Entwurfsbeschreibung Benchmark

Verantwortlicher: Jakob Kusnick

Daniel Alexander, Jakob Kusnick, Richard Khulusi, Christopher Schott, Tobias Turke, Nikolas Oberhäuser, Hans Dieter Pogrzeba, Felix Conrads

Inhaltsverzeichnis

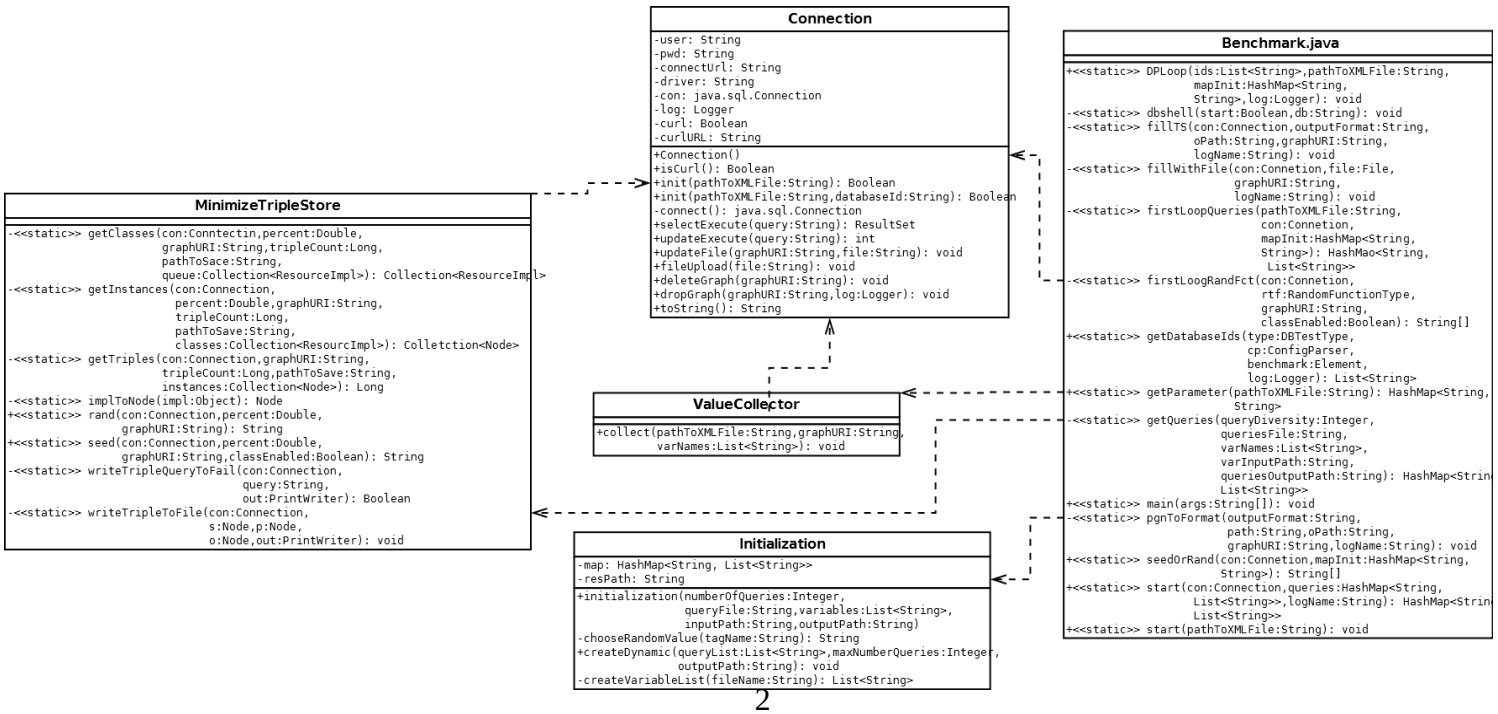
1. Allgemeines 2
2. Aufbau und Ablauf 2
3. Auswertung und Statistik des Benchmarks 4

1. Allgemeines

Das Ziel des Benchmarks ist es, einen möglichst geeigneten Triplestore zu ermitteln, um den Nutzern den Zugang zu den Schachdaten möglichst effizient zu gestalten.
Effizient bedeutet dabei, dass die Anfragezeit für die Benutzer sehr gering ist.
Um den dafür prädestinierten Triplestore zu finden wird ein automatisiertes Test-Programm entwickelt, welches einen zuvor erstellten Test-Datensatz in die Triplestores schreibt und anschließend aufzeichnet, wie schnell die SPARQL-Anfragen auf diesen Daten verarbeitet werden.

2. Aufbau und Ablauf

Der Benchmark besteht aus verschiedenen Phasen, dementsprechend auch aus mehreren Komponenten. Für jeden einzelnen Triplestore wird eine komplette Test-Routine abgearbeitet. Um dem Triplestore eine möglichst realitätsnahe Simulation des Betriebes zu bieten, ist es von Nöten, geeignete Queries anhand von Userszenarien zu erschaffen und diese dann dynamisch zu gestalten. Dies bedeutet, dass Anfragen mit Platzhaltern für verschiedene Variablen erzeugt werden und diese zufällig ersetzt werden. Durch diese Variation soll dem Triplestore die Chance auf die einfache Nutzung des Cachespeichers genommen werden.



Vor dem eigentlichen Benchmark wird eine Initialisierung durchgeführt, in deren Rahmen die dynamischen Queries erzeugt und für eine effiziente Abarbeitung abgespeichert werden. Dies geschieht im Vorfeld, damit die Queries für alle Triplestores identisch sind, um so im späteren Verlauf eine maximale Fairness sowie Effizienz zu gewährleisten.
Hierfür werden zunächst in der Klasse „ValueCollector.java“ aus einem zuvor eingerichteten Triplestore jegliche Werte für alle Prädikate gesammelt und in Textdateien unter dem entsprechenden Prädikat-Namen abgespeichert.
In einem weiteren Initialisierungs-Schritt werden in der „Initialization.java“ die Platzhalter in den vorbereiteten Test-Queries zufällig mit Werten aus den zuvor gesammelten Daten ersetzt. Auch diese nun dynamischen Queries werden jeweils in einzelne Dateien abgespeichert.

Um zu ermitteln wie die einzelnen Triplestores mit verschiedenen Datensätzen arbeiten, werden neben dem vollen Bestand (100%) auch Testläufe mit 50%, 20% und 10% durchgeführt.
Die Erstellung dieser beschränkten Datensätze ist an das Verfahren das DBPedia-SPARQL-Benchmark angelehnt. Es werden also zwei Funktionen namens Rand und Seed genutzt.
Bei der Rand-Funktion werden zufällig Elemente aus dem Datenbestand gewählt, bis der gewünschte Bestand für den neuen Datensatz erreicht ist.
Die Seed-Funktion hingegen ermittelt für alle drei vorliegenden Klassen 10% der Instanzen für welche alle Triple ausgewählt werden.
Dieser Vorgang wird wiederholt bis der neue Datenbestand die gewünschte Größe erreicht hat.

Im Folgenden wird ermittelt, welche der beiden Funktionen zu wählen ist.
Hierbei gilt es, dass für die neuen, verkleinerten Datenbestände ein möglichst hoher Grad an Repräsentativität zu dem ehemaligen Gesamtdatenbestand besteht.
Ob für die Minimierung die Seed- oder Rand-Funktion gewählt wird, hängt nun also von der durchschnittlichen Distanz von der Anzahl der Triple, Objekte und Subjekte, sowie von dem durchschnittlichen Eingangs- und Ausgangsgrad (sowohl mit Literal, als auch ohne Literal) ab.
Gewählt wird dabei nun die kleinere Distanz, da dies bedeutet, dass die gewünschte Repräsentativität näher liegt.

Jeder der Triplestores wird mit dem gleichen Ausschnitt dieses Datenbestandes befüllt. Dies geschieht über eine eigens dafür geschriebene Methode, welche sich in der Klasse „Benchmark.java“ befindet.

Nachdem die benötigten Queries und der Testdatenbestand vorliegen beginnt der eigentliche Test für jeden Tripelstore:
Dieser besteht aus zwei Teilen, die sich in der Testlaufdauer unterscheiden, sowie in der Menge an genutzten Queries.

Es wird für jede Query gemessen, wie viele Queries dieser Art binnen einer Sekunde beantwortet werden können. Dabei ist es sehr wichtig, das oben genannte Caching zu umgehen, um realistische Werte zu erhalten. Dies wird dadurch ermöglicht, dass die Query-Listen vorher dynamisch generiert wurden.
Ebenso werden für jeden Triplestore im Test die gleichen dynamischen Queries in der gleichen Reihenfolge gewählt, sodass das Ergebnis und somit die Statistik nicht von der Art der Anfrage beeinflusst wird.

Ist dies abgeschlossen, erfolgt ein Test in einem Intervall von 60 Minuten, bei dem der Triplestore permanent mit Anfragen belastet wird, um die Performance über einen größeren Zeitraum zu untersuchen.
Hierbei werden für jede Query-Art jeweils 2000 Queries erstellt. Die Platzhalter werden durch zufällig aus dem zuvor ermittelten Datenbestand (s.o.) ersetzt. Diese 2000 Queries bilden jeweils eine von 38 Query-Listen.
Vor dem ersten Test eines Triplestores legt ein Zufallsgenerator fest, in welcher Reihenfolge die Queries gewählt werden. Hierzu wird zunächst eine Query-Liste und aus dieser schließlich eine zufällige Query ausgewählt.
Somit werden die gleichen Anfragen in der gleichen Reihenfolge für alle Triplestores benutzt, um die Testbedingungen fair zu halten.
Bei diesem Durchlauf wird nun ermittelt, wie hoch die Gesamtanzahl an verschiedenen Queries in einer Stunde ist, die der Triplestore auswerten kann.
Das Ergebnis wird in QMpH (Query Mixes per Hour - verschiedene Queries in der Stunde) angegeben

3. Auswertung und Statistik des Benchmarks

Die abschließende Auswertung der gesammelten Daten und erstellten Statistiken des Benchmarks erfolgt, sobald der Benchmark für alle Triplestores abgeschlossen ist. Zwischenergebnisse können bei Bedarf per Mail angefordert werden.