Dokumentation UIMADatabaseInterface Couchbase

Philipp Gsell

March 12, 2020

1 Verwendungszweck

Das vorliegende Java Programm stellt eine Erweiterung des UIMADatabaseInterface um eine Schnittstelle für Couchbase dar. Couchbase ist eine verteilte Dokumentdatenbank, die primär das JSON-Format verwendet aber auch in der Lage ist binäre Dateien abzuspeichern. Die Schnittstelle ermöglicht das Einlesen von XMI-Dateien, Konvertieren zu JSON und Upload in ein Couchbase-Cluster, sowie das Abfragen von Dokumenten, Download und (Re)konvertierung zu CAS/XMI. Da Couchbase nur eine Dokumentgröße von max. 20MB pro Dokument zulässt, enthält die Implementierung einen Mechanismus, um mit größeren Dateien umzugehen. Dies wird durch das Umwandeln des JSON-Strings in Bytechunks“ (Bytearrays gleicher Größe) mit einer durch den User festgelegten Größe umgesetzt. Diese Dateien werden dann jeweils als Bytedatei hochgeladen und bei Download/Abfrage wieder zusammengesetzt.

2 Funktionalitäten

Die Couchbase Klasse implementiert die Funktionen des UIMADatabaseInterface. Im folgenden werden die Vorgehensweisen für die essentiellen Funktionalitäten der Implementation vorgestellt.

2.1 Konfiguration der Schnittstelle

Dem Programm können über die Konfigurationsdatei (src/main/resources/couchbase_example.conf) die Verbindungsparameter für den Couchbasecluster übergeben werden. Der Parameter chunksize (in Byte) bestimmt die Größe der Binärchunks in die ein Dokument zerlegt wird. Für das Chunking sind auch die Parameter databucket und metabucket von Bedeutung. In den databucket werden die byte Chunks und ein Inhaltsverzeichnis abgelegt, das genutzt wird um die Chunks wieder zusammenzusetzen. Im metabucket werden Metadaten abgelegt, auf denen Queries ausgeführt werden können. Eine kurze Beschreibung für das Aufsetzen eines Couchbase-Clusters durch ein Dockerimage findet sich in Abschnitt 3.

2.2 Verbinden mit Couchbase

Um sich mit dem Couchbase Cluster zu verbinden, wird ein Objekt der Couchbase Klasse mit einem Objekt der CouchbaseConfig Klasse oder manuell spezifizierten Parametern initialisiert. Um eine Verbindung zu einem Bucket zu trennen, kann destroy() auf die Couchbase Instanz angewandt werden.

2.3 Import einer XMI-Datei

Um eine XMI-Datei aus einem Datenpfad einzulesen kann die Funktion XmiFileToJCas(Filepath) der Klasse XmiHandler verwendet werden. Returned wird ein Jcas mit den Inhalten der XMI.

2.4 Upload

2.4.1 Dummy erstellen

Mit createDummy(JCas) aus der Klasse Couchbase muss zunächst ein leeres Datenbankdokument für das Jcas Objekt erschaffen werden. Enthält das UIMA-Dokument noch keine ID (UIMADBID) so wird eine zufällig generierte ID für das Datenbankdokument generiert und zusätzlich zum Jcas hinzugefügt. Andernfalls wird die UIMADBID des Jcas erfasst und für das Datenbankdokument genutzt.

2.4.2 Dokument befüllen/updaten

Um den zuvor erstellten Dummy zu befüllen, können die updateElement(ID) (kein Chunking) bzw. updateElementBlob(ID) (Chunking) Funktionen der Couchbase Klasse verwendet werden. UpdateElement konvertiert das Übergebene Jcas zu einem JsonString und befüllt den dazugehörigen Dummy damit. Es kann genutzt werden, wenn das entstehende JsonDokument eine Größe unter 20MB hat. In diesem Fall genügt die Nutzung eines einzelnen Buckets.
Für Dokumente deren Json Repräsentation eine Größe von 20Mb übersteigt, muss updateElementBlob verwendet werden. Hierbei wird das Jcas zunächst in einen JSON-String konvertiert, dessen Byterepräsentation anschließend in Chunks gleicher Größe zerlegt wird. Jeder dieser Bytechunks wird jeweils in einem Bytedokument im databucket gespeichert. In dem initial erstellten Dummy wird ein Inhaltsverzeichnis angelegt, um die zu einem Jcas gehörenden Chunks später wieder zusammensetzen zu können. Die Chunks erhalten als Schlüssel jeweils die ID des Jcas mit dem Suffix ::partXY.
Anschließend sollte die Verbindung zum databucket getrennt werden und sich mit dem metabucket verbunden werden, um updateMetaData auszuführen, wodurch die Metadaten des Jcas in einem Json-Dokument gespeichert werden. Dieses Dokument erhält ebenfalls die im Jcas abgelegte ID (dies ist möglich, da es sich bei einem Bucket um einen abgeschlossenen Suchraum handelt). Auf diesen Metadaten können später Abfragen ausgeführt werden.

Es ist möglich simultan mit beiden Buckets verbunden zu sein, was jedoch aufgrund der benötigten Ressourcen nicht zu empfehlen ist.
Die update Funktionen können sowohl für das initiale Befüllen, als auch das Updaten einer Datei verwendet werden, da sie den Inhalt des Datenbankdokumentes überschreiben.

2.5 Download

Um ein Dokument aus der Datenbank abzurufen, stehen die Funktionen getElement(ID) und getElementBlob(ID) zur Verfügung, die jeweils ein Jcas zurückgeben. Erstere sollte verwendet werden, wenn das Dokument ohne Chunking gespeichert wurde, andernfalls zweitere. Eine Verbindung zum entsprechenden Bucket muss vorher etabliert werden (im Falle von getElementBlob zum databucket).

2.6 Export als XMI-Datei

Um ein Jcas als XMI zu exportieren kann JCasToXmiFile(Jcas, FileName) aus der Klasse XMiHandler genutzt werden. Dabei wird die XMI Repräsentation des übergebenen Jcas im Ordner xmi_output unter dem angegebenen Dateinamen erstellt.

2.7 Abfragen

Um Dokumente per N1QL abzufragen stehen die Funktionen aus der Klasse CouchbaseQueries zur Verfügung. Sowohl für den data- als auch den metabucket existieren dort Funktionen, denen ein N1QL-Query als String übergeben werden kann, dessen Ergebnis dann je nach Wahl der Funktion in der Konsole ausgegeben (printQueryResultsData/printQueryResultsMeta) oder einfach returned werden (returnQueryResultsData/returnQueryResultsMeta).

Anmerkung: Weitere Dokumentation einzelner Funktionen und Methoden findet sich im Quellcode.

3 Beispiele

Zum nachvollziehen des Programmablaufs kann die Funktion CouchbaseTest verwendet werden, welche die in xmi_input enthaltenen Dateien hochlädt und anschließend wieder abfragt. Die heruntergeladenen Dateien werden in xmi_output ablegt. Beispiel N1QL Abfragen sind dort ebenfalls zu finden. Um einen fehlerfreien Ablauf zu gewährleisten, müssen die entsprechenden Typsysteme vorhanden sein.
Ein Docker Image sowie eine Anleitung zum Aufsetzen eines Couchbase Servers mit Docker findet sich unter:
https://docs.couchbase.com/server/current/install/getting-started-docker.html
Nachdem ein entsprechender Container gestartet wurde, kann die Datenbank via

localhost:8091 (default port) konfiguriert werden. Eine Anleitung dazu findet sich unter:
https://docs.couchbase.com/server/current/manage/manage-nodes/create-cluster.html
Um Chunking zu nutzen, ist das Erstellen von zwei Buckets erforderlich (metabucket und databucket), für die Funktionalität ohne chunking genügt ein Bucket.

4 Performanz

Nachfolgend eine Übersicht über reine Upload/Downloadzeiten der 5 gegebenen Beispiel XMI-Dateien (ohne XMI-CAS Deserialisierung, bzw. CAS-XMI Serialisierung) für verschiedene Chunkgrößen.

Chunksize	Chunks	Upload time	Download time	Total
0,01 MB	7666	21 s	14 s	35 s
0,1 MB	773	15 s	11 s	26 s
0,2 MB	390	17 s	11 s	28 s
0,5 MB	160	15 s	13 s	28 s
1 MB	84	16 s	13 s	29 s
2 MB	46	13 s	12 s	25 s
4 MB	27	16 s	11 s	27 s
8 MB	18	17 s	10 s	27 s
16 MB	14	19 s	10 s	29 s