package com.twitter.search.earlybird;

import java.io.IOException;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.base.Predicate;

import com.google.common.base.Predicates;

import org.apache.lucene.index.IndexWriterConfig;

import org.apache.lucene.store.Directory;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.twitter.decider.Decider;

import com.twitter.search.common.schema.DynamicSchema;

import com.twitter.search.common.schema.base.Schema.SchemaValidationException;

import com.twitter.search.common.schema.earlybird.EarlybirdCluster;

import com.twitter.search.common.schema.earlybird.EarlybirdSchemaCreateTool;

import com.twitter.search.common.schema.thriftjava.ThriftIndexingEvent;

import com.twitter.search.common.util.CloseResourceUtil;

import com.twitter.search.common.util.io.flushable.DataDeserializer;

import com.twitter.search.common.util.io.flushable.FlushInfo;

import com.twitter.search.core.earlybird.index.EarlybirdIndexSegmentData;

import com.twitter.search.core.earlybird.index.extensions.EarlybirdIndexExtensionsFactory;

import com.twitter.search.earlybird.document.DocumentFactory;

import com.twitter.search.earlybird.document.ThriftIndexingEventDocumentFactory;

import com.twitter.search.earlybird.document.ThriftIndexingEventUpdateFactory;

import com.twitter.search.earlybird.exception.CriticalExceptionHandler;

import com.twitter.search.earlybird.partition.PartitionConfig;

import com.twitter.search.earlybird.partition.SearchIndexingMetricSet;

import com.twitter.search.earlybird.partition.SegmentSyncInfo;

import com.twitter.search.earlybird.partition.UserPartitionUtil;

/\*\*

\* Collection of required indexing entities that differ in the various Earlybird clusters.

\*/

public abstract class EarlybirdIndexConfig {

private static final Logger LOG = LoggerFactory.getLogger(EarlybirdIndexConfig.class);

private final EarlybirdCluster cluster;

private final DynamicSchema schema;

private final Decider decider;

private final SearchIndexingMetricSet searchIndexingMetricSet;

protected final CriticalExceptionHandler criticalExceptionHandler;

/\*\*

\* Creates a new index config using an applicable schema built for the provided cluster.

\*/

protected EarlybirdIndexConfig(

EarlybirdCluster cluster, Decider decider, SearchIndexingMetricSet searchIndexingMetricSet,

CriticalExceptionHandler criticalExceptionHandler) {

this(cluster, buildSchema(cluster), decider, searchIndexingMetricSet,

criticalExceptionHandler);

}

@VisibleForTesting

protected EarlybirdIndexConfig(

EarlybirdCluster cluster,

DynamicSchema schema,

Decider decider,

SearchIndexingMetricSet searchIndexingMetricSet,

CriticalExceptionHandler criticalExceptionHandler) {

this.cluster = cluster;

this.schema = schema;

this.decider = decider;

this.searchIndexingMetricSet = searchIndexingMetricSet;

this.criticalExceptionHandler = criticalExceptionHandler;

LOG.info("This Earlybird uses index config: " + this.getClass().getSimpleName());

}

private static DynamicSchema buildSchema(EarlybirdCluster cluster) {

try {

return EarlybirdSchemaCreateTool.buildSchema(cluster);

} catch (SchemaValidationException e) {

throw new RuntimeException(e);

}

}

/\*\*

\* Creates the appropriate document factory for this earlybird.

\*/

public final DocumentFactory<ThriftIndexingEvent> createDocumentFactory() {

return new ThriftIndexingEventDocumentFactory(

getSchema(), getCluster(), decider, searchIndexingMetricSet,

criticalExceptionHandler);

}

/\*\*

\* Creates a document factory for ThriftIndexingEvents that are updates to the index.

\*/

public final DocumentFactory<ThriftIndexingEvent> createUpdateFactory() {

return new ThriftIndexingEventUpdateFactory(

getSchema(), getCluster(), decider, criticalExceptionHandler);

}

/\*\*

\* Return the EarlybirdCluster enum identifying the cluster this config is for.

\*/

public final EarlybirdCluster getCluster() {

return cluster;

}

/\*\*

\* Return the default filter for UserUpdatesTable - for the archive cluster keep

\* users that belong to the current partition.

\*/

public final Predicate<Long> getUserTableFilter(PartitionConfig partitionConfig) {

if (EarlybirdCluster.isArchive(getCluster())) {

return UserPartitionUtil.filterUsersByPartitionPredicate(partitionConfig);

}

return Predicates.alwaysTrue();

}

/\*\*

\* Creates a new Lucene {@link Directory} to be used for indexing documents.

\*/

public abstract Directory newLuceneDirectory(SegmentSyncInfo segmentSyncInfo) throws IOException;

/\*\*

\* Creates a new Lucene IndexWriterConfig that can be used for creating a segment writer for a

\* new segment.

\*/

public abstract IndexWriterConfig newIndexWriterConfig();

/\*\*

\* Creates a new SegmentData object to add documents to.

\*/

public abstract EarlybirdIndexSegmentData newSegmentData(

int maxSegmentSize,

long timeSliceID,

Directory dir,

EarlybirdIndexExtensionsFactory extensionsFactory);

/\*\*

\* Loads a flushed index for the given segment.

\*/

public abstract EarlybirdIndexSegmentData loadSegmentData(

FlushInfo flushInfo,

DataDeserializer dataInputStream,

Directory dir,

EarlybirdIndexExtensionsFactory extensionsFactory) throws IOException;

/\*\*

\* Creates a new segment optimizer for the given segment data.

\*/

public abstract EarlybirdIndexSegmentData optimize(

EarlybirdIndexSegmentData earlybirdIndexSegmentData) throws IOException;

/\*\*

\* Whether the index is stored on disk or not. If an index is not on disk, it is presumed to be

\* in memory.

\*/

public abstract boolean isIndexStoredOnDisk();

/\*\*

\* Whether documents are search in LIFO ordering (RT mode), or default (Lucene) FIFO ordering

\*/

public final boolean isUsingLIFODocumentOrdering() {

return !isIndexStoredOnDisk();

}

/\*\*

\* Whether this index supports out-of-order indexing

\*/

public abstract boolean supportOutOfOrderIndexing();

/\*\*

\* Returns a CloseResourceUtil used for closing resources.

\*/

public abstract CloseResourceUtil getResourceCloser();

/\*\*

\* Returns the schema for this index configuration.

\*/

public final DynamicSchema getSchema() {

return schema;

}

/\*\*

\* Returns the decider used by this EarlybirdIndexConfig instance.

\*/

public Decider getDecider() {

return decider;

}

public SearchIndexingMetricSet getSearchIndexingMetricSet() {

return searchIndexingMetricSet;

}

}