package com.twitter.servo.repository

object ChunkingStrategy {

/\*\*

\* A chunking strategy for breaking a query into fixed size chunks, with the last

\* chunk possibly being any size between 1 and chunkSize.

\*/

def fixedSize[K](chunkSize: Int): Seq[K] => Seq[Seq[K]] = {

fixedSize(chunkSize, keysAsQuery[K])

}

/\*\*

\* A chunking strategy for breaking a query into fixed size chunks, with the last

\* chunk possibly being any size between 1 and chunkSize.

\*/

def fixedSize[Q <: Seq[K], K](

chunkSize: Int,

newQuery: SubqueryBuilder[Q, K]

): Q => Seq[Q] = { query =>

query.distinct.grouped(chunkSize) map { newQuery(\_, query) } toSeq

}

/\*\*

\* A chunking strategy for breaking a query into roughly equal sized chunks no

\* larger than maxSize. The last chunk may be slightly smaller due to rounding.

\*/

def equalSize[K](maxSize: Int): Seq[K] => Seq[Seq[K]] = {

equalSize(maxSize, keysAsQuery[K])

}

/\*\*

\* A chunking strategy for breaking a query into roughly equal sized chunks no

\* larger than maxSize. The last chunk may be slightly smaller due to rounding.

\*/

def equalSize[Q <: Seq[K], K](

maxSize: Int,

newQuery: SubqueryBuilder[Q, K]

): Q => Seq[Q] = { query =>

{

if (query.size <= maxSize) {

Seq(query)

} else {

val chunkCount = math.ceil(query.size / maxSize.toDouble)

val chunkSize = math.ceil(query.size / chunkCount).toInt

query.distinct.grouped(chunkSize) map { newQuery(\_, query) } toSeq

}

}

}

}