继续昨天研究通过参数去影响druid的连接池行为。

maxActive与activeCount

`maxActive`从字面量来看是“最大连接数”，那么哪些连接可以统计进`maxActive`呢？在源码中，负责创建连接的方法中都有以下这段代码：

```
activeCount + poolingCount >= maxActive
```

在这个判断中，只有当这个表达式为false时才有创建连接的机会。变量`poolingCount`在前面作业中得知是“可用连接”，那么要理解`activeCount`的含义，我认为就了解“哪些连接可以统计进`maxActive`”。

`activeCount`，顾名思义，“活跃线程”，我首先去查找`getConnection()`和`discardConnection(DruidConnectionHolder )`、`DruidPooledConnection.close()`、`DruidDataSource.recycle(DruidPooledConnection)`等，果不其然这里对应着`activeCount`的加减。所以我对`activeCount`的理解是“在连接池中被借出未归还的连接”。

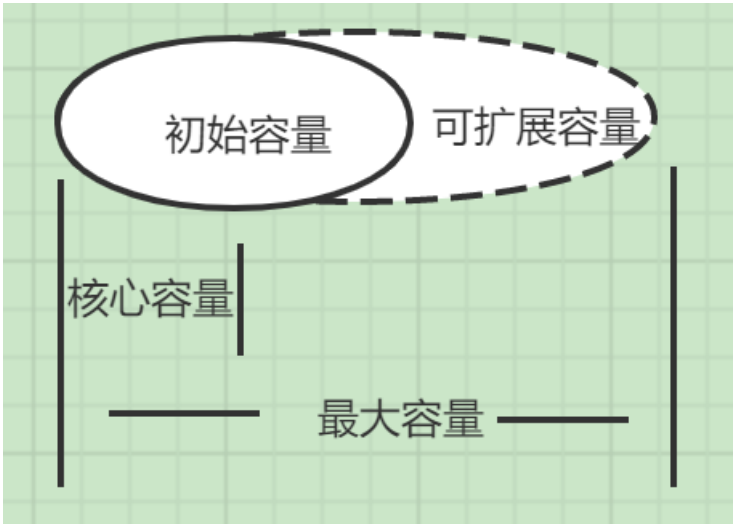
所以`maxActive`对应着是“已使用与可使用的连接数总和上限”。

看Druid源码的目的是想了解如何使用池化技术和管理池中的资源，这里先总结下“利用池化技术时，需要考虑哪些基本元素”。这里比较定制ThreadPoolExecutor的构造函数和Druid的配置参数。

```
ThreadPoolExecutor(int corePoolSize,
                               int maximumPoolSize,
                               long keepAliveTime,
                               TimeUnit unit,
                               BlockingQueue<Runnable> workQueue,
                               ThreadFactory threadFactory,
                               RejectedExecutionHandler handler) {
    ......
}
```

ThreadPoolExecutor的(`corePoolSize, maximumPoolSize`)分别表示线程池的核心值和最大值，在Druid中与之对应的有(`initialSize, minIdle, maxIdle, maxActive`)等。

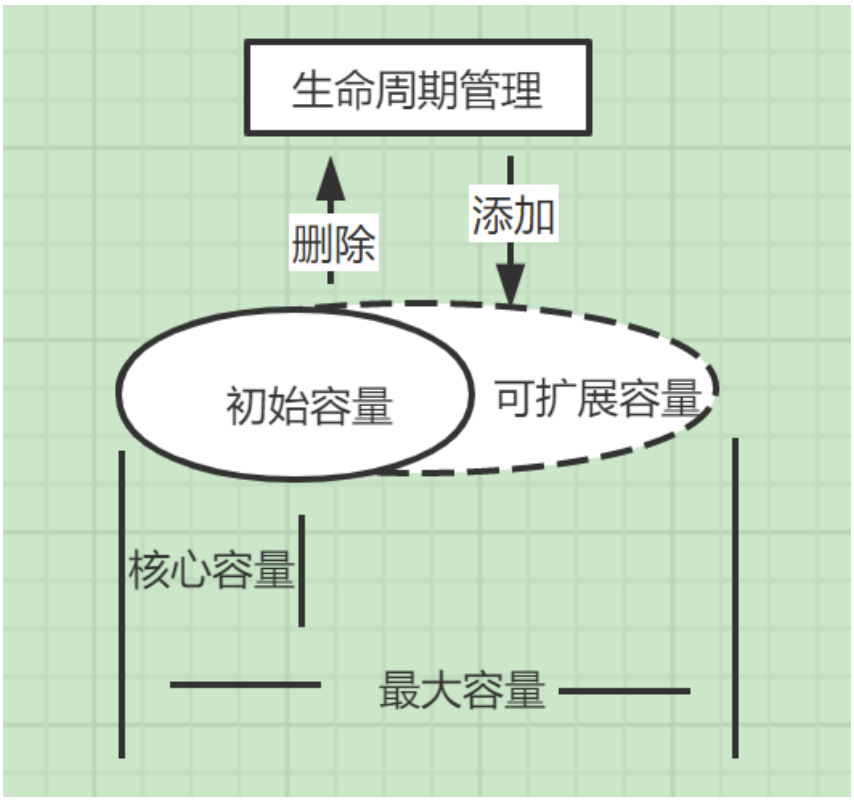
在这里得到的知识是，要被池化的资源，考虑的基本元素是资源池子的可容纳最大限度。其次根据业务，可以允许客户定制初始化容量、核心资源容量等。示意图如下：



其次，如果业务对资源的生命周期或者时效性有要求（例如资源空闲的最长时间，资源有效期等），那么需要提供对应的实现和参数配置。ThreadPoolExecutor的(`keepAliveTime`, `unit`)在线程数量超出核心线程数后，会把超过存活时间的线程移除掉。在Druid中与之对应的是(`minIdle, maxIdle, phyTimeoutMillis, minEvictableIdleTimeMillis`等)。在Druid初始化的时候，会启动一个负责destory的线程，部分代码如下：

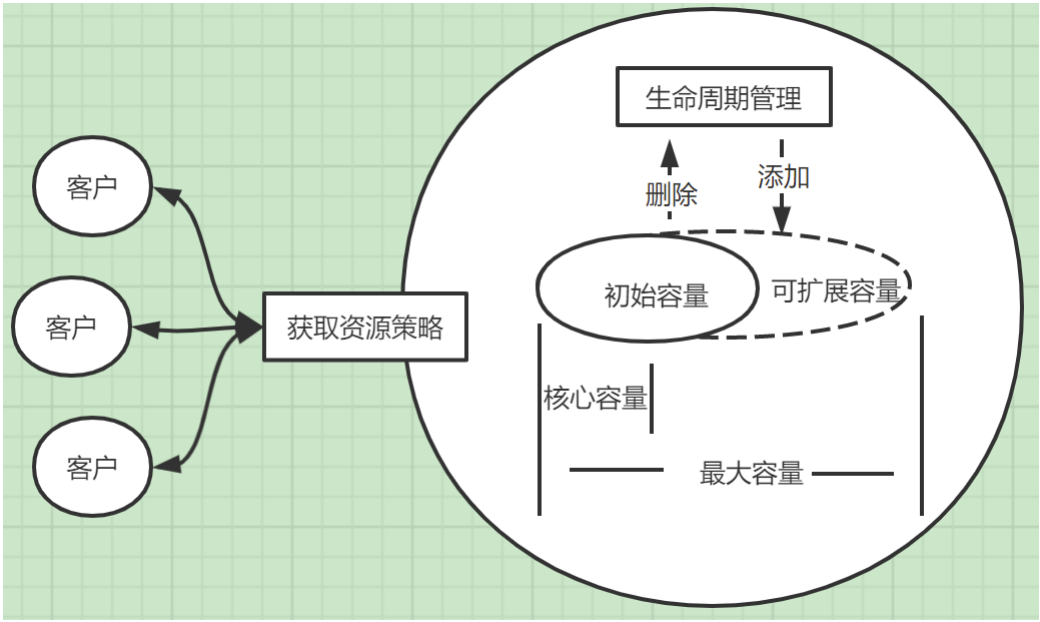
```
public void init() throws SQLException {
    // 忽略部分代码
    createAndStartDestroyThread();
    // 忽略部分代码
}

protected void createAndStartDestroyThread() {
    // 忽略了部分代码
    destroyTask = new DestroyTask();
    destroyConnectionThread = new
DestroyConnectionThread(threadName);
    destroyConnectionThread.start();
}
public class DestroyConnectionThread extends Thread {
    // 忽略了部分代码
    public void run() {
        for (;;) {
            destroyTask.run();
        }
    }
}
public class DestroyTask implements Runnable {
    public void run() {
        shrink(true, keepAlive);// shrink方法前面说过，按照规则负责“收
缩”连接池。
```

```
            if (isRemoveAbandoned()) {
                removeAbandoned();
            }
        }
    }
```



再考虑一个因素是：“客户从池子中获取资源”这个行为。最基本要考虑“当无资源可用”时客户该如何处理。在ThreadPoolExecutor中有RejectedExecutionHandler，在Druid与之对应的有maxmaxWait, maxWaitThreadCount等。前者指的是在线程池饱和时，面对新提交的任务的应对策略。后者Druid则是允许客户自定义等待连接时长或者一直等待，同时还可以配置等待连接最大值，当达到等待连接最大值时，再有客户获取连接时就直接抛异常，这部分代码如下：

```
private DruidPooledConnection getConnectionInternal(long maxWait)
throws SQLException {
    // 忽略了部分代码
    if (maxWaitThreadCount > 0
        && notEmptyWaitThreadCount >= maxWaitThreadCount) {
        connectErrorCountUpdater.incrementAndGet(this);
        throw new SQLException("maxWaitThreadCount " +
maxWaitThreadCount + ", current wait Thread count " +
lock.getQueueLength());
    }
}
```


客户
客户
客户
获取资源策略
生命周期管理
删除
添加
初始容量
可扩展容量
核心容量
最大容量