对于一个简单的数据库应用，由于对于数据库的访问不是很频繁。这时可以简单地在需要访问数据库时，就新创建一个连接，用完后就关闭它，这样做也不会带来什么明显的性能上的开销。但是对于一个复杂的数据库应用，情况就完全不同了。频繁的建立、关闭连接，会极大的减低系统的性能，因为对于连接的使用成了系统性能的瓶颈。

    连接复用。通过建立一个数据库连接池以及一套连接使用管理策略，使得一个数据库连接可以得到高效、安全的复用，避免了数据库连接频繁建立、关闭的开销。

    对于共享资源，有一个很著名的设计模式：资源池。该模式正是为了解决资源频繁分配、释放所造成的问题的。把该模式应用到数据库连接管理领域，就是建立一个数据库连接池，提供一套高效的连接分配、使用策略，最终目标是实现连接的高效、安全的复用。

数据库连接池的基本原理是在内部对象池中维护一定数量的数据库连接，并对外暴露数据库连接获取和返回方法。如：

外部使用者可通过getConnection 方法获取连接，使用完毕后再通过releaseConnection 方法将连接返回，注意此时连接并没有关闭，而是由连接池管理器回收，并为下一次使用做好准备。

数据库连接池技术带来的优势：

1． 资源重用

由于数据库连接得到重用，避免了频繁创建、释放连接引起的大量性能开销。在减少系统消耗的基础上，另一方面也增进了系统运行环境的平稳性（减少内存碎片以及数据库临时进程/线程的数量）。

2． 更快的系统响应速度

数据库连接池在初始化过程中，往往已经创建了若干数据库连接置于池中备用。此时连接的初始化工作均已完成。对于业务请求处理而言，直接利用现有可用连接，避免了数据库连接初始化和释放过程的时间开销，从而缩减了系统整体响应时间。

3． 新的资源分配手段

对于多应用共享同一数据库的系统而言，可在应用层通过数据库连接的配置，实现数据库连接池技术，几年钱也许还是个新鲜话题，对于目前的业务系统而言，如果设计中还没有考虑到连接池的应用，那么…….快在设计文档中加上这部分的内容吧。某一应用最大可用数据库连接数的限制，避免某一应用独占所有数据库资源。

4． 统一的连接管理，避免数据库连接泄漏

在较为完备的数据库连接池实现中，可根据预先的连接占用超时设定，强制收回被占用连接。从而避免了常规数据库连接操作中可能出现的资源泄漏。一个最小化的数据库连接池实现.

连接池类是对某一数据库所有连接的“缓冲池”，主要实现以下功能：①从连接池获取或创建可用连接；②使用完毕之后，把连接返还给连接池；③在系统关闭前，断开所有连接并释放连接占用的系统资源；④还能够处理无效连接（原来登记为可用的连接，由于某种原因不再可用，如超时，通讯问题），并能够限制连接池中的连接总数不低于某个预定值和不超过某个预定值。

DBCP连接池参数：

| **参数** | **默认值** | **说明** |
| --- | --- | --- |
| username | root | 传递给JDBC驱动的用于建立连接的用户名 |
| password | root | 传递给JDBC驱动的用于建立连接的密码 |
| url | jdbc:[mysql://localhost:3306/adname](https://link.jianshu.com?t=mysql://localhost:3306/adname" \t "D:/note/%E4%B8%BA%E7%9F%A5%E7%AC%94%E8%AE%B0/temp/d09aa126-fe44-402d-90b0-900829f40fff/128/_blank) | 传递给JDBC驱动的用于建立连接的URL |
| driverClassName | com.mysql.jdbc.Driver | 使用的JDBC驱动的完整有效的[Java](https://link.jianshu.com?t=http://lib.csdn.net/base/javaee" \t "D:/note/%E4%B8%BA%E7%9F%A5%E7%AC%94%E8%AE%B0/temp/d09aa126-fe44-402d-90b0-900829f40fff/128/_blank) 类名 |
| initialSize | 0 | 初始化连接:连接池启动时创建的初始化连接数量,1.2版本后支持 |
| maxActive | 8 | 最大活动连接:连接池在同一时间能够分配的最大活动连接的数量, 如果设置为非正数则表示不限制 |
| maxIdle | 8 | 最大空闲连接:连接池中容许保持空闲状态的最大连接数量,超过的空闲连接将被释放,如果设置为负数表示不限制 |
| minIdle | 0 | 最小空闲连接:连接池中容许保持空闲状态的最小连接数量,低于这个数量将创建新的连接,如果设置为0则不创建 |
| maxWait | 无限 | 最大等待时间:当没有可用连接时,连接池等待连接被归还的最大时间(以毫秒计数)超过时间则抛出异常,如果设置为-1表示无限等待 |
| testOnReturn | false | 是否在归还到池中前进行检验 |
| testWhileIdle | false | 连接是否被空闲连接回收器(如果有)进行检验.如果检测失败,则连接将被从池中去除.设置为true后如果要生效,validationQuery参数必须设置为非空字符串 |
| minEvictableIdleTimeMillis | 1000 \* 60 \* 30 | 连接在池中保持空闲而不被空闲连接回收器线程(如果有)回收的最小时间值，单位毫秒 |
| numTestsPerEvictionRun | 3 | 在每次空闲连接回收器线程(如果有)运行时检查的连接数量 |
| timeBetweenEvictionRunsMillis | -1 | 在空闲连接回收器线程运行期间休眠的时间值,以毫秒为单位.如果设置为非正数,则不运行空闲连接回收器线程 |
| validationQuery | null | SQL查询,用来验证从连接池取出的连接,在将连接返回给调用者之前.如果指定,则查询必须是一个SQL SELECT并且必须返回至少一行记录 |
| testOnBorrow | true | 是否在从池中取出连接前进行检验,如果检验失败,则从池中去除连接并尝试取出另一个. |