[数据库连接池的好处](http://yplsoft.iteye.com/blog/156697)

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：

[连接池的优缺点](http://mofeichen.iteye.com/blog/557417)

优点

使用连接池的最主要的优点是性能。创建一个新的数据库连接所耗费的时间主要取决于网络的速度以及应用程序和数据库服务器的(网络)距离，而且这个过程通常是一个很耗时的过程。而采用数据库连接池后，数据库连接请求可以直接通过连接池满足而不需要为该请求重新连接、认证到数据库服务器，这样就节省了时间。

缺点

数据库连接池中可能存在着多个没有被使用的连接一直连接着数据库(这意味着资源的浪费)。

[Java中数据库连接池原理机制的详细讲解](http://blog.csdn.net/ilibaba/article/details/3996962)

连接池的基本工作原理

　　1、基本概念及原理

　　由上面的分析可以看出，问题的根源就在于对数据库连接资源的低效管理。我们知道，对于共享资源，有一个很著名的设计模式：资源池 (Resource Pool)。该模式正是为了解决资源的频繁分配?释放所造成的问题。为解决上述问题，可以采用数据库连接池技术。数据库连接池的基本思想就是为数据库连接 建立一个“缓冲池”。预先在缓冲池中放入一定数量的连接，当需要建立数据库连接时，只需从“缓冲池”中取出一个，使用完毕之后再放回去。我们可以通过设定 连接池最大连接数来防止系统无尽的与数据库连接。更为重要的是我们可以通过连接池的管理机制监视数据库的连接的数量?使用情况，为系统开发?测试及性能调 整提供依据。

　　2、服务器自带的连接池

　　JDBC的API中没有提供连接池的方法。一些大型的WEB应用服务器如BEA的WebLogic和IBM的WebSphere等提供了连接池的机制，但是必须有其第三方的专用类方法支持连接池的用法。

　　连接池关键问题分析

　　1、并发问题

　　为了使连接管理服务具有最大的通用性，必须考虑多线程环境，即并发问题。这个问题相对比较好解决，因为Java语言自身提供了对并发管理的支 持，使用synchronized关键字即可确保线程是同步的。使用方法为直接在类方法前面加上synchronized关键字，如：

　　public synchronized Connection getConnection()

　　2、多数据库服务器和多用户

　　对于大型的企业级应用，常常需要同时连接不同的数据库(如连接Oracle和Sybase)。如何连接不同的数据库呢?我们采用的策略是：设计 一个符合单例模式的连接池管理类，在连接池管理类的唯一实例被创建时读取一个资源文件，其中资源文件中存放着多个数据库的url地址()?用户名()?密 码()等信息。如 tx.url=172.21.15.123：5000/tx\_it，tx.user=yang，tx.password=yang321。根据资源文件提 供的信息，创建多个连接池类的实例，每一个实例都是一个特定数据库的连接池。连接池管理类实例为每个连接池实例取一个名字，通过不同的名字来管理不同的连 接池。

　　对于同一个数据库有多个用户使用不同的名称和密码访问的情况，也可以通过资源文件处理，即在资源文件中设置多个具有相同url地址，但具有不同用户名和密码的数据库连接信息。

　　3、事务处理

　　我们知道，事务具有原子性，此时要求对数据库的操作符合“ALL-ALL-NOTHING”原则,即对于一组SQL语句要么全做，要么全不做。

　　在Java语言中，Connection类本身提供了对事务的支持，可以通过设置Connection的AutoCommit属性为 false,然后显式的调用commit或rollback方法来实现。但要高效的进行Connection复用，就必须提供相应的事务支持机制。可采用 每一个事务独占一个连接来实现，这种方法可以大大降低事务管理的复杂性。

　　4、连接池的分配与释放

　　连接池的分配与释放，对系统的性能有很大的影响。合理的分配与释放，可以提高连接的复用度，从而降低建立新连接的开销，同时还可以加快用户的访问速度。

　　对于连接的管理可使用空闲池。即把已经创建但尚未分配出去的连接按创建时间存放到一个空闲池中。每当用户请求一个连接时，系统首先检查空闲池内 有没有空闲连接。如果有就把建立时间最长(通过容器的顺序存放实现)的那个连接分配给他(实际是先做连接是否有效的判断，如果可用就分配给用户，如不可用 就把这个连接从空闲池删掉，重新检测空闲池是否还有连接);如果没有则检查当前所开连接池是否达到连接池所允许的最大连接数(maxConn),如果没有 达到，就新建一个连接，如果已经达到，就等待一定的时间(timeout)。如果在等待的时间内有连接被释放出来就可以把这个连接分配给等待的用户，如果 等待时间超过预定时间timeout,则返回空值(null)。系统对已经分配出去正在使用的连接只做计数，当使用完后再返还给空闲池。对于空闲连接的状 态，可开辟专门的线程定时检测，这样会花费一定的系统开销，但可以保证较快的响应速度。也可采取不开辟专门线程，只是在分配前检测的方法。

　　5、连接池的配置与维护

　　连接池中到底应该放置多少连接，才能使系统的性能最佳?系统可采取设置最小连接数(minConn)和最大连接数(maxConn)来控制连接 池中的连接。最小连接数是系统启动时连接池所创建的连接数。如果创建过多，则系统启动就慢，但创建后系统的响应速度会很快;如果创建过少，则系统启动的很 快，响应起来却慢。这样，可以在开发时，设置较小的最小连接数，开发起来会快，而在系统实际使用时设置较大的，因为这样对访问客户来说速度会快些。最大连 接数是连接池中允许连接的最大数目，具体设置多少，要看系统的访问量，可通过反复测试，找到最佳点。

　　如何确保连接池中的最小连接数呢?有动态和静态两种策略。动态即每隔一定时间就对连接池进行检测，如果发现连接数量小于最小连接数，则补充相应数量的新连接,以保证连接池的正常运转。静态是发现空闲连接不够时再去检查。

　　连接池的实现

　　1、连接池模型

　　本文讨论的连接池包括一个连接池类(DBConnectionPool)和一个连接池管理类 (DBConnetionPoolManager)。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。

　　连接池管理类是连接池类的外覆类(wrapper),符合单例模式，即系统中只能有一个连接池管理类的实例。其主要用于对多个连接池对象的管 理，具有以下功能：①装载并注册特定数据库的JDBC驱动程序;②根据属性文件给定的信息，创建连接池对象;③为方便管理多个连接池对象，为每一个连接池 对象取一个名字，实现连接池名字与其实例之间的映射;④跟踪客户使用连接情况，以便需要是关闭连接释放资源。连接池管理类的引入主要是为了方便对多个连接 池的使用和管理，如系统需要连接不同的数据库，或连接相同的数据库但由于安全性问题，需要不同的用户使用不同的名称和密码。

　　2、连接池实现

　　下面给出连接池类和连接池管理类的主要属性及所要实现的基本接口：

**[java]** [view plaincopy](http://blog.csdn.net/ilibaba/article/details/3996962)

1. **public** **class** DBConnectionPool **implements** TimerListener{
2. **private** **int** checkedOut;//已被分配出去的连接数
3. **private** ArrayList freeConnections=**new** ArrayList();
4. //容器，空闲池，根据//创建时间顺序存放已创建但尚未分配出去的连接
5. **private** **int** minConn;//连接池里连接的最小数量
6. **private** **int** maxConn;//连接池里允许存在的最大连接数
7. **private** String name;//为这个连接池取个名字，方便管理
8. **private** String password;//连接数据库时需要的密码
9. **private** String url;//所要创建连接的数据库的地址
10. **private** String user;//连接数据库时需要的用户名
11. **public** Timer timer;//定时器
12. **public** DBConnectionPool(String name,String URL,String user,
13. String password,**int** maxConn)//公开的构造函数
14. **public** **synchronized** **void** freeConnection(Connection con)
15. //使用完毕之后，把连接返还给空闲池
16. **public** **synchronized** Connection getConnection(**long** timeout)
17. //得到一个连接，timeout是等待时间
18. **public** **synchronized** **void** release()
19. //断开所有连接，释放占用的系统资源
20. **private** Connection newConnection()
21. //新建一个数据库连接
22. **public** **synchronized** **void** TimerEvent()
23. //定时器事件处理函数
24. }
25. **public** **class** DBConnectionManager {
26. **static** **private** DBConnectionManager instance;
27. //连接池管理类的唯一实例
28. **static** **private** **int** clients;//客户数量
29. **private** ArrayList drivers=**new** ArrayList();
30. //容器，存放数据库驱动程序
31. **private** HashMap pools = **new** HashMap();
32. //以name/value的形式存取连接池对象的名字及连接池对象
33. **static** **synchronized** **public** DBConnectionManager getInstance()
34. /\*\*如果唯一的实例instance已经创建，直接返回这个实例;否则，调用私有构造函数，
35. 创建连接池管理类的唯一实例\*/
36. **private** DBConnectionManager()
37. //私有构造函数,在其中调用初始化函数init()
38. **public** **void** freeConnection(String name,Connection con)
39. //释放一个连接，name是一个连接池对象的名字
40. **public** Connection getConnection(String name)
41. //从名字为name的连接池对象中得到一个连接
42. **public** Connection getConnection(String name,**long** time)
43. //从名字为name的连接池对象中取得一个连接，time是等待时间
44. **public** **synchronized** **void** release()//释放所有资源
45. **private** **void** createPools(Properties props)
46. //根据属性文件提供的信息，创建一个或多个连接池
47. **private** **void** init()//初始化连接池管理类的唯一实例，由私有构造函数调用
48. **private** **void** loadDrivers(Properties props)//装载数据库驱动程序
49. }

　　3、连接池使用

　　上面所实现的连接池在程序开发时如何应用到系统中呢?下面以Servlet为例说明连接池的使用。

　　Servlet的生命周期是：在开始建立servlet时，调用其初始化(init)方法。之后每个用户请求都导致一个调用前面建立的实例的service方法的线程。最后，当服务器决定卸载一个servlet时，它首先调用该servlet的destroy方法。

　　根据servlet的特点，我们可以在初始化函数中生成连接池管理类的唯一实例(其中包括创建一个或多个连接池)。如：

**[java]** [view plaincopy](http://blog.csdn.net/ilibaba/article/details/3996962)

1. **public** **void** init() **throws** ServletException
2. {
3. connMgr=DBConnectionManager.getInstance();
4. }
5. 然后就可以在service方法中通过连接池名称使用连接池，执行数据库操作。最后在destroy方法中释放占用的系统资源，如：
6. **public** **void** destroy(){
7. connMgr.release();
8. **super**.destroy();
9. }

结束语　在使用JDBC进行与数据库有关的应用开发中，数据库连接的管理是一个难点。很多时候，连接的混乱管理所造成的系统资源开销过大成为制约大型企 业级应用效率的瓶颈。对于众多用户访问的Web应用，采用数据库连接技术的系统在效率和稳定性上比采用传统的其他方式的系统要好很多。本文阐述了使用 JDBC访问数据库的技术?讨论了基于连接池技术的数据库连接管理的关键问题并给出了一个实现模型。文章所给出的是连接池管理程序的一种基本模式，为提高 系统的整体性能，在此基础上还可以进行很多有意义的扩展。