oracle的oci和thin区别 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/645/2021_2022_oracle_E7_9A_84o_c102_645699.htm 我是今天看到tomcat数据源的配置时，想起来这个问题，刚开始还不晓得thin是什么东西！ database.url=jdbc：oracle：thin：angel/oracle@192.168.55.11：1530：monitordb 经过上网查询，得到如下结果： jdbc：oracle连接的是oracle数据库 thin是一种方法 angel/oracle那个angel是用户名，oracle是密码 192.168.55.11是你要连的电脑ip 1530是oracle的连端口（1521貌似是默认端口） monitordb这是数据库的名字 下面是我转载的文章，是关于tomcat数据源中oracle的oci和thin区别： 前几天同事跑过来跟我说， 机房中的一台tomcat服务器跟oracle数据库机连接很慢，查看控制台中的hibernate日志， 基本上是一条sql出来要等个1-2秒再出第二条。但同样的程序在他自己机器上的tomcat运行，同样是连那台数据库机器，就快很多，不会出现前面的每执行1条sql就卡一次壳的情况。 初步分析，我就想到可能是网络原因， 机房两台机器连接不畅通， 程序和机器差的原因基本可以排除， 机房的tomcat机比我们开发机要强多了， 而且程序在他的机器上运行又没有问题。于是我就劝他到机房去检查一下网络状态， 但他一时也无法进入，因为机房的管理人员不在。 过了一会， 他告诉我问题解决了， 把数据库访问的url更换成了oci方式就好了， oci对我来说有些陌生， 我一直是用的thin，也没想过其他连接方式。对于oci我也只能想到oracle 的client中貌似是有oci什么的，当时有其他事情也没管了。 今天有意了解一下区别，先看看thin和oci的url写法上

的区别：jdbc：oracle：thin：@server ip： service jdbc：oracle：oci：@service看来oci的还更加简洁，ip可以省掉不写了。接下来再找找oci和thin的其他区别，发现有如下解释：引用 Oracle provides four different types of JDBC drivers， for use in different deployment scenarios. The 10.1.0 drivers can access Oracle 8.1.7 and higher. While all Oracle JDBC drivers are similar， some features apply only to JDBC OCI drivers and some apply only to the JDBC Thin driver. JDBC OCI client-side driver： This is a JDBC Type 2 driver that uses Java native methods to call entrypoints in an underlying C library. That C library， called OCI （Oracle Call Interface）， interacts with an Oracle database. The JDBC OCI driver requires an Oracle client installation of the same version as the driver. The use of native methods makes the JDBC OCI driver platform specific. Oracle supports Solaris， Windows， and many other platforms. This means that the Oracle JDBC OCI driver is not appropriate for Java applets， because it depends on a C library. Starting from 10.1.0， the JDBC OCI driver is available for install with the OCI Instant Client feature， which does not require a complete Oracle client-installation. Please refer to Oracle Call Interface for more information. JDBC Thin client-side driver： This is a JDBC Type 4 driver that uses Java to connect directly to Oracle. It implements Oracles SQL*Net Net8 and TTC adapters using its own TCP/IP based Java socket implementation. The JDBC Thin driver does not require Oracle client software to be installed， but does require the server to be configured with a TCP/IP listener. Because it is written entirely in Java， this driver is

platform-independent. The JDBC Thin driver can be downloaded into any browser as part of a Java application. （Note that if running in a client browser， that browser must allow the applet to open a Java socket connection back to the server.） JDBC Thin server-side driver： This is another JDBC Type 4 driver that uses Java to connect directly to Oracle. This driver is used internally within the Oracle database. This driver offers the same functionality as the client-side JDBC Thin driver （above）， but runs inside an Oracle database and is used to access remote databases. Because it is written entirely in Java， this driver is platform-independent. There is no difference in your code between using the Thin driver from a client application or from inside a server. 连接方式有以下几种： Oralce provides four types of JDBC driver. Thin Driver， a 100% Java driver for client-side use without an Oracle installation， particularly with applets. The Thin driver type is thin. To connect user scott with password tiger to a database with SID （system identifier） orcl through port 1521 of host myhost， using the Thin driver， you would write ：Connection conn = DriverManager.getConnection （"jdbc：oracle：thin：@myhost：1521：orcl"， "scott"， "tiger"）； OCI Driver for client-side use with an Oracle client installation. The OCI driver type is oci. To connect user scott with password tiger to a database with SID （system identifier） orcl through port 1521 of host myhost， using the OCI driver， you would write ：Connection conn = DriverManager.getConnection（"jdbc ：oracle：oci：@myhost：1521：orcl"， "scott"， "tiger"）； Note that you can also specify the database by a TNSNAMES entry.

You can find the available TNSNAMES entries listed in the file tnsnames.ora on the client computer from which you are connecting. For example, if you want to connect to the database on host myhost as user scott with password tiger that has a TNSNAMES entry of MyHostString, enter：Connection conn = DriverManager.getConnection（"jdbc：oracle：oci8：@MyHostString", "scott", "tiger"）；If your JDBC client and Oracle server are running on the same machine, the OCI driver can use IPC （InterProcess Communication） to connect to the database instead of a network connection. An IPC connection is much faster than a network connection. Connection conn = DriverManager.getConnection（"jdbc：oracle：oci8：@", "scott", "tiger"）；Server-Side Thin Driver, which is functionally the same as the client-side Thin driver, but is for code that runs inside an Oracle server and needs to access a remote server, including middle-tier scenarios. The Server-Side Thin driver type is thin and there is no difference in your code between using the Thin driver from a client application or from inside a server. Server-Side Internal Driver for code that runs inside the target server, that is, inside the Oracle server that it must access. The Server-Side Internal driver type is kprb and it actually runs within a default session. You are already "connected". Therefore the connection should never be closed. To access the default connection, write：DriverManager.getConnection（"jdbc：oracle：kprb："）；or：DriverManager.getConnection（"jdbc：default：connection："）；You can also use the Oracle-specific defaultConnection（）

method of the OracleDriver class which is generally recommended ：OracleDriver ora = new OracleDriver（） ；Connection conn = ora.defaultConnection（） ； Note： You are no longer required to register the OracleDriver class for connecting with the Server-Side Internal driver， although there is no harm in doing so. This is true whether you are using getConnection（） or defaultConnection（） to make the connection. Any user name or password you include in the URL string is ignored in connecting to the server default connection. The DriverManager.getConnection（） method returns a new Java Connection object every time you call it. Note that although the method is not creating a new physical connection （only a single implicit connection is used）， it is returning a new object. Again， when JDBC code is running inside the target server ， the connection is an implicit data channel， not an explicit connection instance as from a client. It should never be closed. 这下基本明白了1）从使用上来说，oci必须在客户机上安装oracle客户端或才能连接，而thin就不需要，因此从使用上来讲thin还是更加方便，这也是thin比较常见的原因。 2）原理上来看，thin是纯java实现tcp/ip的c/s通讯；而oci方式，客户端通过native java method调用c library访问服务端，而这个c library就是oci（oracle called interface），因此这个oci总是需要随着oracle客户端安装（从oracle10.1.0开始，单独提供OCI Instant Client，不用再完整的安装client） 3）它们分别是不同的驱动类别，oci是二类驱动， thin是四类驱动，但它们在功能上并无差异。 4）虽然很多人说oci的速度快于thin，但找了半天没有找到相关的测试报告。 百考试题温馨提示：本内容