[简介 3](#_Toc18627)

[嵌入式DERBY基础 3](#_Toc16923)

[DERBY JDBC 数据库连接URL 3](#_Toc30974)

[DERBY系统 3](#_Toc16388)

[启动数据库 4](#_Toc27278)

[关闭系统 5](#_Toc25609)

[定义系统目录 5](#_Toc25729)

[错误日志 6](#_Toc21460)

[Derby属性文件 6](#_Toc19801)

[Derby双开 6](#_Toc32057)

[德比数据库文件目录详情 7](#_Toc32634)

[创建删除和备份数据库 8](#_Toc19849)

[单个数据库关闭 8](#_Toc1433)

[存储和恢复 8](#_Toc21662)

[数据库页面 9](#_Toc10277)

[广泛性 9](#_Toc14658)

[数据库限制机制 9](#_Toc14980)

[系统关闭 9](#_Toc20960)

[数据库连接 9](#_Toc20369)

[在系统中连接数据库 9](#_Toc2974)

[在系统外部连接数据库 10](#_Toc30408)

[约定指定数据库名字 10](#_Toc23369)

[特殊的数据库访问 10](#_Toc28663)

[通过类路径进入数据库 10](#_Toc3783)

[通过归档JAR文件访问 11](#_Toc29052)

[数据库连接实例 11](#_Toc19170)

[通过连接地址属性开发数据库 12](#_Toc15692)

[使用数据库属性 12](#_Toc8367)

[关闭数据库 12](#_Toc25431)

[在locale-based排序下创建数据库 13](#_Toc16089)

[创建一个不区分大小写的数据库 13](#_Toc17426)

[创建一个自定义的排序 13](#_Toc9881)

[在一个属性中指定属性对象 14](#_Toc15392)

[使用内存数据库工具 15](#_Toc18726)

[创建内存数据库 15](#_Toc27251)

[查看存在的内存数据库 15](#_Toc23004)

[使用内存数据库 15](#_Toc19557)

[移除内存数据库 15](#_Toc17006)

[保持一个内存数据库 16](#_Toc1936)

[使用derby属性 16](#_Toc20963)

[持久化属性 16](#_Toc14192)

[优先级属性 16](#_Toc17448)

[database-wide属性的保护级 17](#_Toc1303)

[动态属性和静态属性 17](#_Toc28537)

[设置DERBY属性 17](#_Toc16759)

[验证系统属性存在性 18](#_Toc3582)

[在C/S环境下设置属性 18](#_Toc16147)

[设置属性案例 18](#_Toc3395)

DERBY系列6-JDBC下DERBY基础



# 简介

derby采用的是JDBC API接口进行连接的关系型数据库，JDBC API是JAVA平台的一部分，也是连接数据库所需要具备的接口，如果需要更详细的了解DERBY运行过程，你需要去了解JDBC接口的运行过程。

本章节介绍了应用程序编程的详细信息，通过JDBC下URL访问数据库的具体流程

也包括了DERBY系统的概念。

此外，你也有必要去学习类似SQL之类的关系型数据库基本的查询语句，SQL是标准的关系型查询语言，不依赖特定的编程语言，DERBY支持SQL下的基本语言和一些高级功能并且DERBY实现了JDBC API作为JAVA EE兼容的资源管理器。

# 嵌入式DERBY基础

本章节叫你如何配置和使用DERBY编写一个基础的数据库

### DERBY JDBC 数据库连接URL

Derby通过DriverManager.getConnection方法去响应JDBC，通过字符串建立一个URL连接，提供了一个JDBC驱动接口，他让你能跟连接数据库并进行高级操作，下列语句是通过字符串创建URL

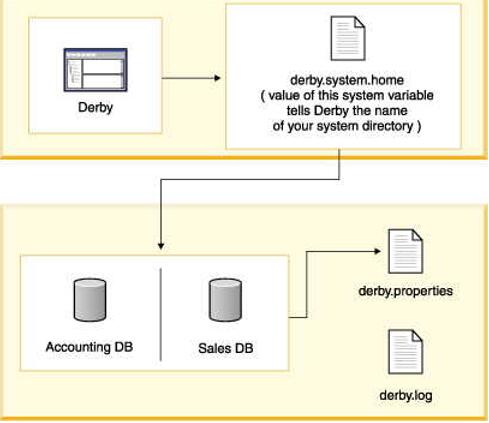
|  |
| --- |
| Connection conn = DriverManager.getConnection("jdbc:derby:sample"); |

在嵌入式环境中应用数据库，使用B/S和C/S的应用程序会产生不同的连接地址，在以后会详细介绍。

### DERBY系统

德比系统是德比数据库引擎的一个实例，通过一个系统目录，零个或多个数据库和一个全系统配置组成。系统目录中包含任何持续系统范围的配置参数和属性，在他的系统有一个特有的文件叫”derby.properties”,这个文件不是自动生成的，你需要自己创建。

但是derby不是持久的，你必须在每一个启动时指定系统的位置，然而，德比系统和系统目录是运行数据库的一个组成部分，这是成功启动部署德比的关键，如下图



在系统目录中也包含错误日志，在系统中每个数据库都包含子目录，并且他与数据库同名，此外，如果你连接外部数据库时，他会默认使用为当前系统的一部分。

当你使用嵌入式驱动程序时，德比数据库文件和日志文件有指定文件系统的默认权限，如果你使用JAVA SE7或更高版本，你可以通过限制文件访问来提高安全性，通过derby.storage.useDefaultFilePermissions 设置为FALSE实现，详情请看引用手册。

注意：在内存数据库中不出现在系统目录中。

每个JVM实现一个DERBY实例

你可能有两个在同一台机器上运行的系统实例，但每个实例必须运行在不同的JVM。

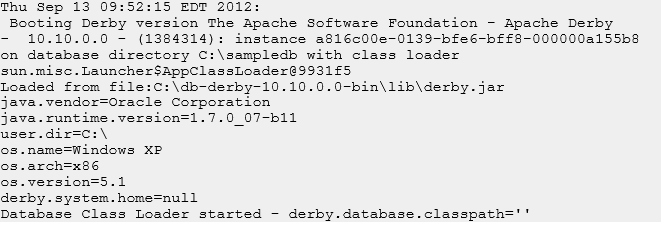
如果使用嵌入式驱动程序，两个独立的德比实例不能访问同一个数据库，如果一个DB实例试图访问一个数据库将会出现一组错误信息，你可以使用网络服务器实现，通过参考相同的数据库名称进行映射数据库。

### 启动数据库

德比的默认配置是在应用程序第一次连接时配置，当DB启动一个数据库时，他会在检查本机是否已经启动.

你可以配置你的系统启动时去自动执行系统，但是这会影响你的性能。

一旦德比启动一个数据库时，在目录中会产生一个日志文件，里面会显示你的德比系统目录的位置，如果你指没有定了derby.system.home 属性，会出现以下信息：



数据库系统JVM中有限制，这可能是性能保护

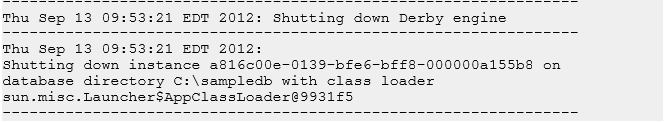
### 关闭系统

在一个嵌入式环境中，当一个程序关闭时，会首先关闭DERBY系统

但当不再JVM环境下运行德比时，他会在继续重新连接启动这个数据库，在嵌入式系统中，应用程序会关闭deerby系统并调用JDBC

|  |
| --- |
| DriverManager.getConnection("jdbc:derby:;shutdown=true"); |

关闭derby的命令，他会提高SQLEXCEPTIONS的性能，在日志文件中会产生如下信息



通常情况下，一个应用程序使用德比引擎时会最后关闭自己，然而他可以在在关闭德比后重新启动相同的JVM会话，通过以下代码去执行启动

|  |
| --- |
| Class.forName("org.apache.derby.jdbc.EmbeddedDriver").newInstance(); |

JDBC不建议在newinstance()建立，但添加方法能在任何JVM下启动

建议：如果你想关闭时重新启动DERBY，你可以在语句下添加deregister=false来重新连接URL，这样可以避免重新加载嵌入式引擎保证运行性能速度

|  |
| --- |
| DriverManager.getConnection("jdbc:derby:;shutdown=true;deregister=false"); |

他可能关闭了单一的数据库，而不是整个德比系统，你可以在关闭时重新启动同一个session中的数据库。

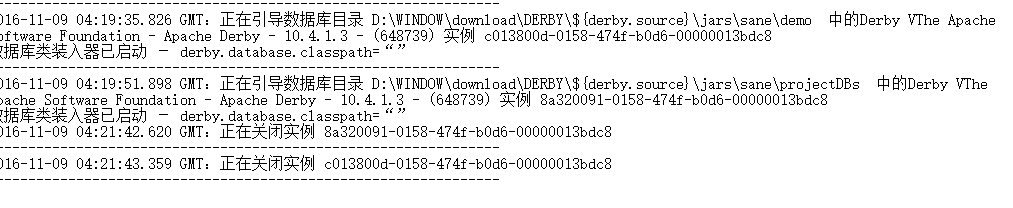
### 定义系统目录

Derby提供了一个系统目录，你可以通过在DERBY.SYSTEM.HOME中指定一个JAVA系统属性

如果你在启动debi时不指定系统目录，他就会默认为当前的系统目录，你可以在系统目录中查看到目录以及各种参数，如果你启动了一个不存在的系统目录，德比会在目录里自动创建一个没有数据库的系统

### 错误日志

当你在系统中创建或连接一个数据库时，德比会将信息输出在一个日志中，通常的情况下，derby会在目录中产生一个叫DERBY.LOG的日志文件



或者你可以通过发送信息到流中，你可以使用derby.stream.error.method或derby.stream.error.field属性，在一个不同的文件使用DERBY.STREAM.ERROR.FILE,你的操作属性会出现在日志中。

### Derby属性文件

在DERBY.PROPERTIES文件中包含为系统配置定义的信息

它不是自动创建的，如果你想写入DERBY文件，你需要为自己创建一个，DERBY.PROPERYIES 会生成JAVA.UTIL.PROPERTIES.SAVE方法，详情请看引用手册

### Derby双开

德比为了防止相同的数据库存在，会在数据库目录中生存一个叫DB.LCK文件

如果一个的比的第二个实例试图启动正在运行的数据库，会产生以下错误信息：

此外，在堆栈跟踪出现的DERBY.LOG文件中，为帮助诊断双开的问题，可以使用

Derby.stream.error.logBootTrace 属性来获得信息引导。

如果DERBY启动数据库时无法建立DB.LCK，数据库将会进入只读模式，这可能会导致由于缺乏磁盘空间或数据库目录的访问权限的错误，

## 德比数据库文件目录详情

在一个基本的德比数据库中包含字典对象，也可以存储自己的配置信息

德比目录

一个德比数据库存储在与数据库同名的系统目录文件中。

当你建立一个数据库时，与数据库同名的文件夹会生成以下文件夹

|  |  |
| --- | --- |
| 目录名 | 解释 |
| log目录 | 包含构成数据库日志的文件，内部用于数据修复(不与错误日志相同的东西) |
| Seg0目录 | 包含用户表，系统表和索引，将以2进制文件存储。 |
| Service配置文件 | 一个具有内部配置信息的文本文件，具体如下   |  | | --- | | SysschemasIndex2Identifier=225  SyscolumnsIdentifier=144  SysconglomeratesIndex1Identifier=49  SysconglomeratesIdentifier=32  SyscolumnsIndex2Identifier=177  SysschemasIndex1Identifier=209  SysconglomeratesIndex3Identifier=81  SystablesIndex2Identifier=129  SyscolumnsIndex1Identifier=161  derby.serviceProtocol=org.apache.derby.database.Database  SysschemasIdentifier=192  derby.storage.propertiesId=16  SysconglomeratesIndex2Identifier=65  derby.serviceLocale=zh\_CN  SystablesIdentifier=96  SystablesIndex1Identifier=113 | |
| TMP目录(可能不存在) | 这是用于大型分类和延迟更新和删除的临时目录，类是由各种SQL语句使用，对于只读的数据库，你可能需要设置一个属性来更改此位置 |
| Jar目录(可能不存在) | 当使用数据库加载时，存储一个存储库文件目录 |

只读数据库可以被压缩到JAR包或者解压文件，下面是德比数据库目录的文件和目录，包含了数据库目录和文件结构,在下图中是derby数据库目录与文件的结构

Derby为了保护数据的性能也对数据有一定的长度限制，在图中也列举了列的规范性，以便在开发中把握数据的整体结构合理性

|  |  |
| --- | --- |
| 数据类型 | 限制 |
| 每个数据库的表 | java.lang.LONG.MAX\_VALUE  一些操作系统会限制目录的文件数 |
| 表中的索引 | 32767 |
| 表中的行 | 无限制 |
| 主键 | 16 |
| 表长度 | 无限制，但一些系统下的目录会限制长度 |
| 行长度 | 无限制 |

### 创建删除和备份数据库

在记忆型数据库中，连接地址属性去关闭，但是在数据库的系统文件中并没有删除属性

所以，你只能在当数据库开启时才能去去关闭或删除数据库，你可以启动数据库启动行中的GETPROPERTYINFO去了解

如果你想去备份数据库，你可以使用它内置的联机备份 数据库工具在之后会详细介绍

或者你可以使用前滚恢复(roll-foeward)去修复你的数据库，derby通过使用数据库的完整备份实现向前恢复，详细的机制会在之后介绍

### 单个数据库关闭

一个应用程序可以在derby系统结束运行时去关闭单个数据库

### 存储和恢复

一个derby数据库数据存寸和恢复机制，derby会确保所有提交的事务都是持久性的，甚至如果系统出现问题时，可以通过数据库事务日志使用数据库

然而当一个事务提交时，写入磁盘的数据库操作都没有被缓存，只是进入了临时的磁盘，当如果想进行恢复丢失事务操作时，可以将所有提交失败的事务都应用在数据库中，以及

活跃的回滚操作，使当前数据库保持在一个一致有效的状态。

在进行基础的操作中，derby会通过小日志进行定期检查，检查点标志会将一部分无用的信息去除日志

所以，如果你想恢复数据库时，需要消耗许多的资源，你需要适当的关闭数据库来提高启动的性能

### 数据库页面

Derby中的表和列作为一个集合体包含了2个以上的页面

一个页面可以作为一个系统的可配置的存储单元，或者在一个集体的基础下，一个集合能以增长到八页的用户数据。

### 广泛性

你可以设置许多德比属性作为库级属性，他们保存在数据库中以游离的方式知道系统性能的运用，可以保证数据库的性能

### 数据库限制机制

德比数据库不支持在列中使用CLOB,BLOB,LONG VARCHAR 和XML的数据类型，如果索引中的键列长度大于索引半页的大小长度会失败，在现有的索引中，如果新建的列大于索引页大小的一半会导致插入失败。

你可以通过在数据量小的列上创建索引，通过它来查询数据量大的行。

### 系统关闭

如果数据库日志不能分配给更多的磁盘空间，系统将会关闭，在DERBY.LOG文件中会出现IOEXCEPTION的异常信息，你需要查看你的磁盘空间是否允许进行磁盘写入操作

### 数据库连接

在你连接数据库地址时会在DriverManager.getConnection中响应，更多详细内容会在之后介绍

### 在系统中连接数据库

进入数据库最基本的方法就是通过数据库绝对路径进行访问，JDBC会通过对应的地址进入对应的系统目录进行操作，在一个C/S环境中，路径的地址通常在一个服务器中

在默认的情况下，你可以直接通过访问数据库名字来进行访问当前数据库，比如你已经在系统中使用了一个叫”DERBYDB”的数据库，你可以通过以下命令进行直接访问

|  |
| --- |
| jdbc:derby:DERBYDB; |

在JDBC中，你可以使用以下命令

|  |
| --- |
| Connection conn=DriverManager.getConnection(“jdbc:derby:DERBYDB”); |

### 在系统外部连接数据库

你也可以连接到数据库在其他目录(包括子目录的目录)通过指定一个相对或绝对路径名来识别数据库。你指定一个绝对路径是由主机操作系统定义的。

通过特定的链接地址，你可以在同一时间连接多个数据库

|  |
| --- |
| jdbc:derby:../otherDirectory/myDB |
| Jdbc:derby:c:/otherDirectory/mydb |

注意：在你启动数据库后，一个数据库已经成为了DERBY系统中的一部分，这意味在整个系统都在访问实例，所以你只需要通过访问数据库目录即可进行操作。

### 约定指定数据库名字

当你进入来自文件系统中的数据库，任何在系统目录地址名都不是绝对的，所以要通过约定规则去定义数据库的名字并执行命令

1. 基于用途的
2. 加上”create=true”属性

即使在WINDOW路径名中，路径名也不能只包含一个冒号(:)，详细的在之后介绍

你只能在指定的JVM下运行，NFS文件系统在UNIX和WINDOWS远程共享文件中不保证会工作，你可以通过derby.system.home去查看

当两个不同数据库中路径名称和值，或者相对或者绝对指向同一个实际的目录，他们会被认为是相通的，这意味着通过他的绝对路径名和相对路径名会连接到同一个数据库中，数据库的名称有他的规范路径定义的目录**java.io.File.getCanonicalPath**

在新建数据库时，DERBY会自动生成不存在的目录，如果他没有创建，这表明创建数据库失败.

### 特殊的数据库访问

在特殊的情况下，你也可以通过JAR文件或者类路径进行访问数据库或者进行数据操作，在之后会介绍使用内存数据库进行访问

## 通过类路径进入数据库

在许多的情况下，你是通过文件系统来进入数据库，但是你也可以通过类路径进入数据库，通过已经归档的JAR或者ZIP文件进入，但是所有的数据库都是只读状态。

访问一个数据库的路劲，可以使用类路径subprotocol.

通过以下命令可以访问该类路径

|  |
| --- |
| Jdbc:derby:classpath:sample |

如果路径只包含类路径，你也可以通过此命令访问

|  |
| --- |
| Jdbc:derby:classpath:demo/databses/sample; |

### 通过归档JAR文件访问

这在一定的情况下可以实现，这需要保证类路径存在JAR文件中，请确认路径文件的完整性

### 数据库连接实例

这个例子实在嵌入式环境中使用连接语法实现的，详细的语法结构可以通过DERBY语法手册进行参考

|  |  |
| --- | --- |
| 语法 | 解释 |
| Jdbc:derby:london/sales | 在数据库连接中，LONDON是系统目录的子目录，而SALES是LONDON目录下的子目录 |
| Jdbc:derby:/reference/phrases/french | 尝试连接路径下的数据库  在UNIX系统中，这是一个绝对目录下的路径，但在WINDOW中，这是一个存在于C盘的路径 |
| Jdbc:derby:a:/demo/sample | 在window系统A磁盘中打开/DEMO/SAMPLE目录下的数据库存储目录 |
| jdbc:derby:c:/databases/salesdb jdbc:derby:salesdb | 在window系统A磁盘中打开/DEMO/SAMPLE目录下的数据库存储目录中并访问SALESDB数据库 |
| JDBC:DERBY:SUPPORT/BUGSDB;CREATE=TRUE; | 在系统目录中数据库不存在的情况下，自动生成数据库 |
| shutdown=true | 关闭该数据库 |
| jdbc:derby:memory:myDB | 进入该内存数据库，在服务器连接地址的语法会有所不同，在之后会详细介绍 |
| jdbc:derby:classpath:myDB | 进入只读模式下数据库 |
| jdbc:derby:jar:(C:/dbs.jar)products/boiledfood | 从JAR文件中访问目录下的只读数据库 |
| dbc:derby:directory:myDB | 进入文件系统下的数据库 |

## 通过连接地址属性开发数据库

指定属性连接URL，这个例子是在嵌入式环境下进行连接的，你也可以这些语句在其他链接地址上。

你也可以设置这些属性通过一个属性对象DriverManager.getConnection去连接，如果指定的任何属性都在CONNECTION URL和一个属性对象.

### 使用数据库属性

你可以通过三种方法的属性去连接数据库

1. 通过数据库名字连接

|  |
| --- |
| jdbc:derby:;databaseName=databaseName |

1. 通过归档文件进入只读数据库

|  |
| --- |
| jdbc:derby:jar:(pathToArchive)databasePathWithinArchive |

1. 包含类路径的归档文件

|  |
| --- |
| jdbc:derby:classpath:databasePathWithinArchive |

### 关闭数据库

在嵌入式环境中你可以通过一下指令在连接地址中进行关闭操作

|  |
| --- |
| jdbc:derby:;shutdown=true |

如果你关闭成功时，会报出一个SQL异常信息，包括了关闭数据库的内容，这是正常现象。

如果你想在管理员模式下在JDK 8或更高的环境下关闭，你必须通过关闭整个derby系统来关闭数据库。

在JDBC中，通过一下指令实现关闭（在程序应用中）

|  |
| --- |
| // shutting down a database from your application  DriverManager.getConnection(  "jdbc:derby:sample;shutdown=true"); |

或者你想在JDBC中关闭单独的数据库，可以通过以下命令

|  |
| --- |
| // shutting down an authenticated database as database owner  DriverManager.getConnection(  "jdbc:derby:securesample;user=joeowner;password=secret;shutdown=true"); |

如果你在使用java.sql.DriverManager.setLoginTimeout这个方法是出现登录超时，可能是这些问题造成的

1. LDAP身份验证的网络问题
2. 连接超时，你可以重启连接数据库

### 在locale-based排序下创建数据库

在默认的情况下，derby运用UNICODE 编码格式进行排序，但是，你可以使用LOCALE-BASED排序创建数据库

你可以使用collation=collation和territory=ll\_CC属性去说明LOCALE-BASED排序。

这个操作只接受用户定义的表，系统定义的可以使用默认的编码排序

**属性限制**：你不能利用此属性去给更高权限的数据库说明

下面是创建数据库的例子

|  |
| --- |
| Jdbc:derby:MexicanDb;create=true;territory=es\_MX;collation=TERRITORY\_BASED |

你可以通过derby参考手册去了解详细参数

### 创建一个不区分大小写的数据库

当你使用locale-based排序创建数据库，TERRITORY\_BASED在去榆中使用默认的排序方式，在通常的情况下，会进行排序和比较

为了让数据库使用不区分大小写的搜索方式，显式强度降低了属性。

|  |
| --- |
| jdbc:derby:SwedishDB;create=true;territory=sv\_SE;collation=TERRITORY\_BASED:PRIMARY |

在PRIMARY的强度中，字符将会不区分大小写，你可以使用SECONDARY去区分大小写。

### 创建一个自定义的排序

在特殊的情况下，你可能需要定义一个排序去区分强度，你可以经过以下步骤去实现

1. 创建一个类去扩展java.text.spi.CollatorProvider.class去定义一个排序方法
2. 创建一个文本叫META-INF/services/java.text.spi.CollatorProvider里面包含了排序所欲要的类
3. 将写好的类放进JRE's lib/ext目录中或者在JAVA.EXT.DIRS属性中声明

比如在GREEK字符编码格式中使用拉丁文，利用相近定义字符格式，下面是例子

1. 在JDBC下使用：

|  |
| --- |
| public class MyCollatorProvider extends CollatorProvider {  public Locale[] getAvailableLocales() {  return new Locale[] {  new Locale("en", "US", "greek")  };  }  public Collator getInstance(Locale locale) {  StringBuilder rules = new StringBuilder();  // alpha should go between a and b  rules.append("< a,A < \u03b1,\u0391 < b,B");  // beta should go between b and c  rules.append("& b,B < \u03b2,\u0392 < c,C");  // add more rules here ....  try {  return new RuleBasedCollator(rules.toString());  } catch (ParseException pe) {  throw new Error(pe);  }  }  } |

在IJ下使用

|  |
| --- |
| ij> connect  'jdbc:derby:GreekDB;territory=en\_US\_greek;collation=TERRITORY\_BASED;create=true';  ij> create table t (x varchar(12));  0 rows inserted/updated/deleted  ij> insert into t values 'a', 'b', 'c', '#', '#';  5 rows inserted/updated/deleted  ij> select \* from t order by x;  X  ------------  a  #  b  #  c  5 rows selected |

### 在一个属性中指定属性对象

你可以去声明对象去指定属性并连接数据库

|  |
| --- |
| Properties p = new Properties();  p.setProperty("user", "sa");  p.setProperty("password", "manager");  p.setProperty("create", "true");  Connection conn = DriverManager.getConnection(  "jdbc:derby:mynewDB", p); |

**注意：**重载

### 使用内存数据库工具

为了测试和开发应用程序或者处理持久化数据，你可以使用内存数据库工具

一个内存数据库不在文件系统中而是缓存在内存中，这样你可以在重复创建使用数据库中提高性能。

如果你想需要提高处理性能，你可以使用内存数据库

### 创建内存数据库

跟基本的创建几乎相似，只需在路径中加入”MEMORY”属性即可，下面是例子

|  |
| --- |
| jdbc:derby:memory:myDB;create=true |

在网络服务器驱动下，使用此命令

|  |
| --- |
| jdbc:derby://myhost:1527/memory:myDB;create=true |

### 查看存在的内存数据库

当你不在系统目录中指定绝对路径下的内存数据库，你可以通过以下命令查找数据库

|  |
| --- |
| jdbc:derby:memory:db  jdbc:derby:memory:C:\myderby\db |

### 使用内存数据库

当你在使用内存数据库时，你需要保证堆的配置和derby页面缓存，在之后会详细介绍使用方式

### 移除内存数据库

|  |
| --- |
| jdbc:derby:memory:myDB;drop=true |
| jdbc:derby://myhost:1527/memory:myDB;drop=true |

你可以使用shutdown=true属性去关闭内存数据库，当你执行关闭命令后，derby会报出一个异常，但这是显示关闭成功的异常，你可以去WWDEMEDDED应用程序中去抓获异常。

不过，在以下情况下内存数据库会自动关闭

1. 当你退出IJ后
2. JVM崩溃
3. 系统崩溃，蓝屏

### 保持一个内存数据库

如果你在创建内存数据库时想保留，你可以在桌面中使用此存储过程（SYSCS\_UTIL.SYSCS\_BACKUP\_DATABASE）去保留。

### 使用derby属性

这章是告诉你怎么使用derby属性

你可以使用一下属性去配置derby系统

|  |  |
| --- | --- |
| system-wide | 许多属性建立在SYSTEM-WIDE的基础下，包含了，启动的自动化进程和异常 |
| Database-wide | 一些属性会建立在DATABASE-WIDE的基础下，可以进行区分数据库，保证了部署程序的操作。 |

### 持久化属性

一个database-wide属性通常会持久化保留在数据库中，通常的情况下，在你写入数据时会显示改变，你也可以去任意设置属性，这个属性将会保留规则下的设置

1. 在值改变的情况下，会重新启动
2. 在删除系统文件时，会重新启动
3. 在系统外部启动数据库

### 优先级属性

1. SYSTEM-WIDE属性设置程序化
2. DATABASE-WIDE属性
3. SYSTEM-WIDE 属性写入DERBY-PROPRTIES文件

这表明，在**derby-properties**文件中system-wide属性会重写database-wide属性，而deabase-wide属性会重写系统文件下的SYSTEM-WIDE属性

### database-wide属性的保护级

当你在设置**derby.database.propertiesOnly**属性为TRUE时，database-wide属性不会重写system-wide属性

这个属性保证了启动环境下的安全性，使内部属性不会被恶意修改。

### 动态属性和静态属性

在derby中，许多属性都是Dynamic，这意味着你可以在derby不关闭的情况下设置属性，但有些属性是静态的，这也表明你不能在derby运行时进行设置，你只能在derby启动之前设置属性。

### 设置DERBY属性

这章涵盖了设置属性的几种方式

设置system-wide属性

在DERBY.PROPERTIES文本文件写入SYSTEM-WIDE属性

通过编程改变syste-wide属性性能

你可以在编程通过这几种方式改变属性性能

1. 作为一个命令行参数

通过使用**-D OPTION**来设置

|  |
| --- |
| java -Dderby.system.home=C:\home\Derby\ -Dderby.storage.pageSize=8192 JDBCTest |

1. **在应用程序中使用一个属性对象或声明**

在嵌入模式中，你的derby程序运用在同一个JVM下，你可以通过一下属性去加载JDBC驱动：

|  |
| --- |
| Properties p = System.getProperties();  p.setProperty("derby.system.home", "C:\databases\sample"); |

1. 在DERBY.PROPERTIES文件下更改system-wide属性

在每一个derby系统目录中都有一个叫system-wide的配置文件，而不在数据库里，但在C/S环境下，目录在服务器中存在，他也不会自动帮你生成一个文件和配置属性，你必须要自己创建和配置，通过**java.util.Properties.save**方法中，下面是类似的文件属性

|  |
| --- |
| derby.infolog.append=true  derby.storage.pageSize=8192  derby.storage.pageReservedSpace=60 |

这些属性会在你更改或者删除时改变，或者你在其它目录中启动系统，如果一个数据库在一个系统中移除，SYSTEM-WIDE属性不会以游离的心态存在数据库，而是在你重新设置时存在。

### 验证系统属性存在性

你可以在编程时查询到系统属性的值，你可以通过以下代码实现

|  |
| --- |
| Properties sprops = System.getProperties();  System.out.println("derby.storage.pageSize value: "  + sprops.getProperty("derby.storage.pageSize")); |

你可以使用JMX工具去查找系统详细信息。

设置database-wide属性

database-wide属性在数据库中影响单个数据库的操作，你可以对属性部署安全性，也可以使用系统存储过程，在连接数据库建立实体关系中，你可以使用**SYSCS\_UTIL.SYSCS\_SET\_DATABASE\_PROPERTY**这个存储过程创建声明，在检查属性当前值时，你可以建立**SYSCS\_UTIL.SYSCS\_GET\_DATABASE\_PROPERTY**去声明，当你在声明无效的值时，derby提供了默认的属性值，通过使用**SYSCS\_UTIL.SYSCS\_GET\_DATABASE\_PROPERTY**方法声明。

### 在C/S环境下设置属性

在C/S环境中，你必须为服务器系统中设置系统属性，这表明这当你使用时DERBY.PROPERTIES文件时，文件将存在于server's derby.system.home目录，服务器应用程序可以设置DATABASE-WIDE属性

下面是设置属性的方式

|  |  |
| --- | --- |
| 属性 | 设置方式 |
| SYSTEM-WIDE | 1. 在derby属性中 2. 命令行 |
| DATABASE-WIDE | 通过声明和系统存储过程实现 |

更改动态或者静态属性

属性在derby.Properties文件中通常保持静态的形式，derby将在启动时读取文件信息。

在以下方式中，属性以动态形式使用

1. database-wide属性中
2. 作为系统范围属性对象的derby引擎嵌入进应用程序中

### 设置属性案例

在Derby中，你可以使用许多方式去配置你的系统，这有帮助于你对性能的理解，你也可以在不设置属性的情况下使用，derby提供了默认的方案兼容各种方式。

在你的系统中，有个叫“DERBY.PROPERTIES”的文件存在系统目录，在你创建一个叫”SYSTEM\_DIRECTORY”的文件时，你的数据库也会创建一个相同的文件，在文件中会设置一下属性

|  |
| --- |
| derby.storage.pageSize=8192 |

你启动你的程序时会设置一下属性

|  |
| --- |
| java -Dderby.system.home=c:\system\_directory MyApp |

在进行创建表时，会产生以下信息

|  |
| --- |
| CREATE TABLE table1 (a INT, b VARCHAR(10)) |

在关闭应用程序时，会产生以下命令行

|  |
| --- |
| java -Dderby.system.home=c:\system\_directory  -Dderby.storage.pageSize=4096 MyApp  CREATE TABLE anothertable (a INT, b VARCHAR(10)) |

包含了4096字节的文件属性

在你启动数据库时会在DATABASE-WIDE属性中产生32768页面值

|  |
| --- |
| CallableStatement cs =  conn.prepareCall("CALL SYSCS\_UTIL.SYSCS\_SET\_DATABASE\_PROPERTY(?, ?)");  cs.setString(1, "derby.storage.pageSize");  cs.setString(2, "32768");  cs.execute();  cs.close(); |

在创建表时会自动产生页面值去保留状态

|  |
| --- |
| CREATE TABLE table2 (a INT, b VARCHAR(10)) |

这张表中包含了32768字节的大小

在关闭数据库时，会产生以下结果

|  |
| --- |
| java -Dderby.system.home=c:\system\_directory MyApp |

在创建另外一张表时

|  |
| --- |
| CREATE TABLE table4 (a INT, b VARCHAR(10)) |

Derby在dataase-wide属性创建表时生成了32768子节大小的表，在database-wide重写时进行改变。