SYBASE®
An SAP® Company

程序员参考

jConnect™ for JDBC™

7.07

文档 ID：DC38606-01-0707-01
最后修订日期：2012 年 6 月

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

目录

本书内容

《Sybase jConnect for JDBC 程序员参考》介绍 jConnect™ for JDBC™ 产品，并说明了如何用它访问存储在关系数据库管理系统中的数据。

读者

本手册适用于熟悉 Java 编程语言、 JDBC 和 Transact-SQL® 以及 Sybase® 版的结构化查询语言 (SQL) 的数据库应用程序程序员。

如何使用本手册

本书中的信息是按以下方式组织的：

- 第 1 章 “简介”介绍 jConnect for JDBC 产品的概念和组件。
- 第 2 章 “编程信息”介绍 jConnect for JDBC 编程要求。
- 第 3 章 “安全性”介绍可与 jConnect 结合使用的安全机制。
- 第 4 章 “故障排除”介绍 jConnect 使用过程中可能出现的问题的解决方案和变通解决方法。
- 第 5 章 “性能和调优”介绍如何增强使用 jConnect 的应用程序的性能。
- 第 6 章 “迁移 jConnect 应用程序”介绍如何将应用程序迁移到 jConnect 7.x。
- 第 7 章 “Web 服务器网关”包含有关 Web 服务器网关的信息并解释如何将这些网关与 jConnect 结合使用。
- 附录 B “SQL 例外与警告消息”列出了在使用 jConnect 时可能会遇到的 SQL 例外和警告消息。
- 附录 A “jConnect 示例程序”用作 jConnect 示例程序指南。

相关文档

有关详细信息，请参见以下书籍：

- Sybase jConnect for JDBC Release Bulletin（《Sybase jConnect for JDBC 发行公告》）包含有关 jConnect 的最新信息。
- 针对您的平台的 Software Developer's Kit Release Bulletin （《软件开发工具包发行公告》）包含有关软件开发工具包 (SDK) 的最新信息。

- Software Developer's Kit and Open Server Installation Guide（《软件开发工具包和 Open Server 安装指南》）包含有关安装 SDK 及其 jConnect for JDBC 组件的信息。
- Adaptive Server Enterprise Installation Guide （《Adaptive Server Enterprise 安装指南》）包含有关安装 Adaptive Server 的信息。
- 针对您的平台的 Adaptive Server Enterprise Release Bulletin（《Adaptive Server Enterprise 发行公告》）包含 Adaptive Server® Enterprise 的已知问题和最新更新的信息。
- JDBC 的 jConnect 扩展 javadoc 文档。Java Software 的 Java 开发工具包 (JDK) 包含用于从源代码文件中提取注释的 javadoc 脚本。此脚本已用于从 jConnect 源文件中提取 jConnect 软件包、类和方法的文档。使用完全安装或 javadocs 选项安装 jConnect 时，javadoc 信息位于 *javadocs* 目录 *Installation_directory/docs/en/javadocs* 中。

其它信息来源

使用 Sybase Getting Started CD、SyBooks™ CD 和 Sybase Product Manuals 网站可以了解有关产品的更多信息：

- Getting Started CD 包含 PDF 格式的发行公告和安装指南，还可能包含 SyBooks CD 中未收纳的其它文档或更新信息。它随软件一起提供。要阅读或打印 Getting Started CD 上的文档，您需要使用 Adobe Acrobat Reader，可以通过 CD 上提供的链接从 Adobe 网站免费下载该软件。
- SyBooks CD 含有产品手册，它随软件一起提供。基于 Eclipse 的 SyBooks 浏览器使您能够以易于使用的、基于 HTML 的格式阅读手册。

  有些文档可能是以 PDF 格式提供的，您可以通过 SyBooks CD 上的 PDF 目录访问这些文档。若要阅读或打印 PDF 文件，您需要使用 Adobe Acrobat Reader。

  有关安装和启动 SyBooks 的说明，请参见 Getting Started CD 上的 *SyBooks* 安装指南 或 SyBooks CD 上的 *README.txt* 文件。
- Sybase Product Manuals 网站是 SyBooks CD 的联机版本，您可以使用一种标准 Web 浏览器来访问它。除了产品手册之外，还可以找到有关 EBFs/Maintenance （EBF/维护）、 Technical Documents （技术文档）、 Case Management （案例管理）、 Solved Cases （解决的案例）、 Newsgroups（新闻组）和 Sybase Developer Network（Sybase 开发员网络）的链接。

  若要访问 Sybase Product Manuals 网站，请转到 Product Manuals (http://www.sybase.com/support/manuals/)。

Web 上的 Sybase 认证

Sybase 网站上的技术文档不断在更新。

❖ **查找有关产品认证的最新信息**

1 将 Web 浏览器定位到 Technical Documents (http://www.sybase.com/support/techdocs/)。

2 单击 “认证报告” (Certification Report)。

3 在 “合作伙伴认证报告” (Partner Certification Report) 过滤器中选择相应的产品、平台和时间范围，然后单击 “查找” (Go)。

4 单击合作伙伴认证报告的标题即可显示该报告。

❖ **查找有关组件认证的最新信息**

1 将 Web 浏览器定位到 Availability and Certification Reports (http://certification.sybase.com/)。

2 在 “按基本产品搜索” (Search by Base Product) 下选择产品系列和产品，或在 “按平台搜索” (Search by Platform) 下选择平台和产品。

3 选择 “搜索” (Search) 以显示所选项目的可用性和认证报告。

❖ **创建 Sybase 网站 （包括支持页）的个人化视图**

建立 MySybase 配置文件。 MySybase 是一项免费服务，它允许您创建 Sybase Web 页的个性化视图。

1 将 Web 浏览器定位到 Technical Documents (http://www.sybase.com/support/techdocs/)。

2 单击 “我的 Sybase” (MySybase) 并创建 MySybase 配置文件。

Sybase EBF 和软件维护

❖ **查找有关 EBF 和软件维护的最新信息**

1 将 Web 浏览器定位到 the Sybase Support Page (http://www.sybase.com/support)。

2 选择 “EBF/维护” (EBFs/Maintenance)。如果出现提示信息，请输入您的 MySybase 用户名和口令。

3 选择一个产品。

4 指定时间范围并单击 “查找” (Go)。即会显示 EBF/维护版本的列表。

锁形图标表示因为您没有注册为 “技术支持联系人” (Technical Support Contact)，因此您没有某些 EBF/维护版本的下载授权。如果您尚未注册，但拥有 Sybase 代表提供的或通过支持合同获得的有效信息，请单击 “编辑角色” (Edit Roles) 将 “技术支持联系人” (Technical Support Contact) 角色添加到 MySybase 配置文件中。

5 单击信息图标可显示 EBF/维护报告，或单击产品说明下载软件。

约定

表 1：语法约定

键	定义
命令	命令名、命令选项名、实用程序名、实用程序标志和其它关键字用 sans serif 字体显示。
变量	变量 （即代表您要填充的值的词语）用斜体 显示。
{ }	大括号表示至少选择括号中的一个选项。不要在命令中包括大括号。
[ ]	中括号表示可以选择括号中的一个或多个选项，也可不选。不要在命令中包括大括号。
( )	小括号将作为命令的一部分键入。
|	竖线表示只可以选择一个显示的选项。
,	逗号表示可以选择任意多个显示的选项，可输入逗号作为命令的一部分来分隔选项。

辅助功能特性

为具有可访问性，专门采用 HTML 版本提供本文档。可以利用适应性技术 （如屏幕阅读器）浏览 HTML 文档，也可以用屏幕放大器进行查看。

jConnect for JDBC 和 HTML 文档已进行了测试，符合美国政府“第 508 节辅助功能”的要求。符合 “第 508 节”的文档一般也符合非美国的辅助功能原则，如 World Wide Web 协会 (W3C) 中针对 Web 站点的原则。

此产品的联机帮助也是以 HTM 格式提供的，您可以使用屏幕阅读器进行浏览。

注释 您可能需要对辅助功能工具进行配置以获得最佳使用效果。某些屏幕阅读器按照大小写来辨别文本，例如将 “ALL UPPERCASE TEXT”看作首字母缩写，而将 “MixedCase Text”看作单词。对工具进行配置，规定语法约定，您可能会感觉更方便。有关工具的信息，请查阅文档。

有关 Sybase 如何支持辅助功能的信息，请参见 Sybase Accessibility (http://www.sybase.com/accessibility)。 Sybase 辅助功能站点包括指向“第 508 节”和 W3C 标准的相关信息的链接。

如果需要帮助

对于购买了支持合同的客户安装的每一个 Sybase 产品，都会有一位或多位指定人员获得与 Sybase 技术支持部门联系的授权。如果使用手册或联机帮助不能解决问题，可让指定人员与 Sybase 技术支持部门联系或与所在区域的 Sybase 子公司联系。

第 1 章

简介

本章介绍了 jConnect for JDBC 产品及其概念和组件。

什么是 JDBC?

Oracle Corporation 制定的 Java 数据库连接 (JDBC) 是一套应用程序编程接口 (API) 规范，它允许 Java 应用程序使用结构化查询语言 (SQL) 访问多个数据库管理系统。 JDBC 驱动程序管理器可以处理多个连接到不同数据库的驱动程序。

标准 JDBC API 和 JDBC 标准扩展 API 中包含一组接口，利用这些接口可以打开到数据库的连接，执行 SQL 命令并处理结果。表1-1 中介绍了这些接口。

表 1-1：JDBC 接口

接口	说明
java.sql.Driver	定位数据库 URL 的驱动程序
java.sql.Connection	用于连接到特定数据库
java.sql.Statement	执行 SQL 语句
java.sql.PreparedStatement	处理带参数的 SQL 语句
java.sql.CallableStatement	处理数据库存储过程调用
java.sql.ResultSet	获取 SQL 语句的结果
java.sql.DatabaseMetaData	用于访问关于数据库的连接的信息。
java.sql.ResultSetMetaData	用于访问描述 ResultSet 属性的信息。
javax.sql.Rowset	处理 JDBC RowSet 实现。
javax.sql.DataSource	处理与数据源的连接。
javax.sql.ConnectionPoolDataSource	处理连接归集。

每个关系数据库管理系统都需要一个驱动程序来实现这些接口。有四种类型的 JDBC 驱动程序：

- 类型 *1 JDBC-ODBC* 桥 – 将 JDBC 调用转换成 ODBC 调用，然后将其传递给 ODBC 驱动程序。有些 ODBC 软件必须驻留在客户端计算机中。有些客户端数据库代码可能也驻留在客户端计算机中。
- 类型 *2 native-API partly-Java* 驱动程序 – 将 JDBC 调用转换成数据库特定的调用。该驱动程序能够直接与数据库服务器通信，同时还需要客户端计算机中的一些二进制代码。
- 类型 *3 net-protocol all-Java* 驱动程序 – 利用独立于 DBMS 的网络协议与中间层服务器通信。然后，中间层网关将请求转换成供应商特定的协议。
- 类型 *4 native-protocol all-Java* 驱动程序 – 将 JDBC 调用转换成供应商特定的 DBMS 协议，允许客户端应用程序直接与数据库服务器通信。

有关 JDBC 及其规范的详细信息，请参见Oracle Technology Network for Java (http://www.oracle.com/technetwork/java/index.html)。

什么是 jConnect?

jConnect 是 Sybase 高性能 JDBC 驱动程序。 jConnect 包含以下两种驱动程序：

- native-protocol/all-Java 驱动程序和
- net-protocol/all-Java 驱动程序。

jConnect 使用的协议是 TDS 5.0 （Tabular Data Stream™，第 5 版），它是 Adaptive Server Enterprise 和 Open Server™ 应用程序的本地协议。 jConnect 执行 JDBC 标准，为整个 Sybase 产品系列提供最佳连接，通过它可访问 25 个以上的企业级系统和遗留系统，包括：

- Adaptive Server Enterprise
- SQL Anywhere®
- Sybase® IQ
- Replication Server®
- DirectConnect™

此外， jConnect for JDBC 还能够访问 Oracle、 AS/400 以及其它使用 Sybase DirectConnect 的数据源。

在某些情况下， JDBC 的 jConnect 实现会偏离 JDBC 规范。有关详细信息，请参见第 101 页的 “对 JDBC 标准的约束与说明”。

第 2 章

编程信息

本章描述构成 jConnect for JDBC 的基本组件和编程要求。介绍了如何调用 jConnect 驱动程序、设置连接属性和连接到数据库服务器。还包含有关使用 jConnect 功能的信息。

本章包含下列主题：

有关 JDBC 编程的信息，请转到以下位置中面向 Java 开发人员的资源页：Oracle Technology Network for Java (http://www.oracle.com/technetwork/java/index.html)。

设置 jConnect

本节介绍在使用 jConnect 之前需要执行的任务。

设置 jConnect 版本

jConnect 版本属性 JCONNECT_VERSION 确定驱动程序的行为和激活的功能。例如，Adaptive Server 15.5 支持 jConnect 6.05 和 7.0，但这两个版本处理 datetime 和 time 数据的方式有所不同。在连接到 Adaptive Server 15.5 时，支持微秒级精度的时间数据的 jConnect 7.0 使用 bigdatetime 或 bigtime，即使目标 Adaptive Server 列被定义为 datetime 或 time 也是如此。而不支持微秒级精度的 jConnect 6.05 在连接到 Adaptive Server 15.5 时则始终会转换 datetime 或 time 数据。

可以使用 SybDriver.setVersion 方法或 JCONNECT_VERSION 连接属性设置 jConnect 版本。

使用 SybDriver.setVersion

setVersion 方法影响由 SybDriver 对象创建的*所有*连接的 jConnect 缺省行为。可以多次调用 setVersion 来更改版本设置。新连接会继承在建立连接时与版本设置相关联的行为。在会话过程中更改版本设置不会影响当前连接。可以使用 com.sybase.jdbcx.SybDriver.VERSION_LATEST 常量确保总是请求所使用的 jConnect 驱动程序的最高版本值。但是，在将版本设置为 com.sybase.jdbcx.SybDriver.VERSION_LATEST 后，如果用较新的 jConnect 驱动程序替换当前的 jConnect 驱动程序，可能会发现行为发生变化。

此代码示例演示如何装载 jConnect 驱动程序和设置其版本：

```
import java.sql.DriverManager;
import com.sybase.jdbcx.SybDriver;
SybDriver sybDriver = (SybDriver)
   Class.forName("com.sybase.jdbc4.jdbc.SybDriver")
   .newInstance();
sybDriver.setVersion(com.sybase.jdbcx.SybDriver.
   VERSION_7);
DriverManager.registerDriver(sybDriver);
```

使用 JCONNECT_VERSION

可以使用 JCONNECT_VERSION 连接属性覆盖 SybDriver 版本设置并为特定连接指定不同的版本设置。表 2-1 列出了有效的 JCONNECT_VERSION 值以及与这些值相关联的 jConnect 特征。

表 2-1：jConnect 版本设置及其功能

JCONNECT_VERSION	功能
“7.0”	除以下情况外， jConnect 7.0 与 jConnect 6.05 的行为相同： • jConnect 从服务器请求对 bigdatetime 和 bigtime SQL 数据类型的支持。15.5 之前的 Adaptive Server 版本将忽略该请求。请参见第 66 页的“使用 date 和 time 数据类型”。 • jConnect 支持 JDBC 4.0。请参见第 90 页的 “JDBC 4.0 规范支持”和第 101 页的“对 JDBC 标准的约束与说明”。 • ENABLE_BULK_LOAD 的有效值为空值 （缺省值）、ARRAYINSERT_WITH_MIXED_STATEMENTS、 ARRAYINSERT、 BCP 和 LOG_BCP。
“6.05”	除以下情况外， jConnect 6.05 与 jConnect 6.0 的行为相同： • jConnect 支持计算列，包括其元数据信息。 • jConnect 支持长标识符。通过长标识符，您可以使用最多 255 个字节的标识符或对象名称。长标识符适用于大多数用户定义的标识符，包括表名、列名和索引名称。 • jConnect 支持 JDBC 3.0。请参见第 91 页的 “JDBC 3.0 规范支持”和第 101 页的“对 JDBC 标准的约束与说明”。

JCONNECT_VERSION	功能
“6”	除以下情况外， jConnect 6.0 与 jConnect 5.x 的行为相同： • jConnect 请求对 date 和 time SQL 数据类型的支持。 12.5.1 之前的 Adaptive 版本将忽略该请求。请参见第 66 页的 “使用 date 和 time 数据类型”。 • jConnect 从服务器请求对 unichar 和 univarchar 数据类型的支持。 12.5.1 之前的 Adaptive Server 版本将忽略该请求。请参见第 33 页的 “使用 jConnect 传递 Unicode 数据”。 • jConnect 从服务器请求对宽表的支持。 12.5.1 之前的 Adaptive Server 版本将忽略该请求。第 45 页的 “使用 Adaptive Server 的宽表支持”。 • DISABLE_UNICHAR_SENDING 的缺省值为 false。
“5”	jConnect 5.x 与 jConnect 4.0 的行为相同。
“4”	除以下情况外， jConnect 4.0 与 jConnect 3.0 的行为相同： • LANGUAGE 连接属性的缺省值为 null。 • Statement.cancel 的缺省行为是只取消调用其的 Statement 对象。此行为符合 JDBC 标准。 请使用 CANCEL_ALL 设置 Statement.cancel 的行为。 • 可以使用 JDBC 2.0 方法以列数据的形式存储和检索 Java 对象。请参见第 82 页的 “将 Java 对象作为列数据存储在表中”。
“3”	除以下情况外， jConnect 3.0 与 jConnect 2.0 的行为相同： • 如果 CHARSET 连接属性没有指定字符集， jConnect 将使用数据库的缺省字符集。 • CHARSET_CONVERTER 的缺省值为 CheckPureConverter 类。
“2”	• LANGUAGE 连接属性的缺省值为 us_english。 • 如果 CHARSET 连接属性没有指定字符集，缺省字符集将是 iso_1。 • CHARSET_CONVERTER 的缺省值为 TruncationConverter 类，除非 CHARSET 连接属性指定多字节或 8 位字符集（在这种情况下， CHARSET_CONVERTER 的缺省值为 CheckPureConverter 类。请参见第 35 页的 “jConnect 字符集转换程序”。 • Statement.cancel 的缺省行为是取消调用其的对象，以及任何其它已开始执行且正在等待结果的 Statement 对象。此行为*不*符合 JDBC 标准。 请使用 CANCEL_ALL 设置 Statement.cancel 的行为。

调用 jConnect 驱动程序

若要注册和调用 jConnect，请将 jConnect 添加到 jdbc.drivers 系统属性中。在初始化阶段， DriverManager 类试图装载 jdbc.drivers 中列出的驱动程序。这不如调用 Class.forName 的效率高。可以在此属性中列出多个驱动程序，程序之间用冒号 (:) 分隔。以下代码示例演示如何在程序内向 jdbc.drivers 中添加驱动程序：

```
Properties sysProps = System.getProperties();
String drivers = "com.sybase.jdbc4.jdbc.SybDriver";
String oldDrivers =
sysProps.getProperty("jdbc.drivers");
if (oldDrivers != null)
   drivers += ":"+ oldDrivers;
 sysProps.put("jdbc.drivers", drivers.toString());
```

注释 System.getProperties 不能用于 Java 小程序。请改用 Class.forName 方法。

在 Java 6 和 JDBC 4 中，JDBC 驱动程序的实例化过程已得到简化。您可以使用 Java 系统属性 jdbc.drivers 来指定驱动程序类，例如：

```
java -Djdbc.drivers=com.sybase.jdbc4.jdbc.SybDriver UseDriver
```

UseDriver 程序不需要显式装载驱动程序：

```
public class UseDriver
{
   public static void main(String[] args)
   {
      try {
         Connection conn = java.sql.DriverManager.getConnection
            ("jdbc:sybase:Tds:localhost:5000?USER=sa&PASSWORD=secret");
         // more code to use connection ...
      }
      catch (SQLException se){
         System.out.println("ERROR:SQLException "+se);
      }
   }
}
```

为 J2EE 服务器配置 jConnect

可以使用 com.sybase.jdbc4.jdbc.SybConnectionPoolDataSource 类在 EAServer 等应用程序服务器中配置与 Adaptive Server 服务器的连接池。javax.sql.ConnectionPoolDataSource 接口的 com.sybase.jdbc4.jdbc.SybConnectionPoolDataSource 实现可为每个连接属性提供 getter 和 setter 方法。

还可通过编程方式配置 jConnect，例如：

```
private DataSource getDataSource ()
{
   SybConnectionPoolDataSource connectionPoolDataSource = new
```

```
      SybConnectionPoolDataSource();
   connectionPoolDataSource.setDatabaseName("pubs2");
   connectionPoolDataSource.setNetworkProtocol("Tds");
   connectionPoolDataSource.setServerName("localhost");
   connectionPoolDataSource.setPortNumber(5000);
   connectionPoolDataSource.setUser("sa");
   connectionPoolDataSource.setPassword(PASSWORD);
   return connectionPoolDataSource;
}
private void work () throws SQLException
{
   Connection conn = null;
   Statement stmt = null;
   DataSource ds = getDataSource();
   try {
      conn = ds.getConnection();
      stmt = conn.createStatement();
      // ...
   }
   finally {
      if (stmt != null) {
         try { stmt.close(); } catch (Exception ex) { /* ignore */ }
      }
      if (conn != null) {
         try { conn.close(); } catch (Exception ex) { /* ignore */ }
      }
   }
}
```

建立连接

本节介绍如何使用 jConnect 建立与 Adaptive Server 或 SQL Anywhere 数据库的连接。

连接属性

连接属性指定登录到服务器所需的信息并定义预期的客户端和服务器行为。连接属性名称不区分大小写。

设置连接属性

必须在连接到服务器之前设置连接属性。可使用两种方法设置连接属性：

- 在应用程序中使用 DriverManager.getConnection 方法。
- 在定义 URL 时设置连接属性。

> **注释** 在 URL 中设置的驱动程序连接属性不会覆盖在应用程序中使用 DriverManager.getConnection 方法设置的任何相应的连接属性。

下面的示例代码使用了 DriverManager.getConnection 方法。随 jConnect 提供的示例程序也包含设置这些属性的示例。

```
Properties props = new Properties();
 props.put("user", "userid");
 props.put("password", "user_password");
/*
 * If the program is an applet that wants to access
 * a server that is not on the same host as the
 * web server, then it uses a proxy gateway.
 */
 props.put("proxy", "localhost:port");
/*
 * Make sure you set connection properties before
 * attempting to make a connection.You can also
 * set the properties in the URL.
 */
Connection con = DriverManager.getConnection
  ("jdbc:sybase:Tds:host:port", props);
```

列出当前连接设置

若要列出驱动程序的当前连接设置，请使用 Driver.getDriverPropertyInfo(String url, Properties props)。该代码返回包含以下内容的 DriverPropertyInfo 对象数组：

- 驱动程序属性
- 驱动程序属性所基于的当前设置
- URL 和传入的属性

jConnect 连接属性列表

表 2-2 列出了 jConnect 的连接属性并给出了其缺省值。这些属性不区分大小写。

可使用 getClientInfo() 和 setClientInfo() 标准方法动态设置标记为 **Dynamic** 的连接属性的值。

表 2-2：连接属性

属性	说明	缺省值	Dynamic 或 Static
ALTERNATE_SERVER_NAME	指定镜像 SQL Anywhere 环境中主数据库和辅助数据库使用的备用服务器名称。主数据库和辅助数据库使用相同的备用服务器名称，因此客户端应用程序可以连接到当前主服务器，而无需事先知道两台服务器中的哪一台是主服务器。 JDBC URL 的语法仍然是 `jdbc:sybase:Tds:<hostname>:<port#>/database?connection_property=value;`。但是，如果设置了 ALTERNATE_SERVER_NAME， jConnect 将忽略 *hostname* 和 *port* 变量的值，而是使用 SQL Anywhere UDP 发现协议来确定当前主服务器。 有关数据库镜像的信息，请参见 《*SQL Anywhere* 服务器 – 数据库管理》。 **注释** 也可以对未镜像的 SQL Anywhere 使用 ALTERNATE_SERVER_NAME。不过，您将始终从单独的服务器获取相同的主机和端口值。	Null	Static
APPLICATIONNAME	指定应用程序名称。这是一个用户定义的属性。可以编程使服务器端解释为此属性提供的值。	Null	Static
BE_AS_JDBC_COMPLIANT_AS_POSSIBLE	调整其它属性以确保 jConnect 方法的应答方式尽可能符合 JDBC 3.0 标准。 如果将该属性设置为 “true”，将影响 （并覆盖） 下列属性： • CANCEL_ALL （设置为 “false”） • LANGUAGE CURSOR （设置为 “false”） • SELECT_OPENS_CURSOR （设置为 “true”） • FAKE_METADATA （设置为 “true”） • GET_BY_NAME_USES_COLUMN_LABEL （设置为 “false”）	False	Static

属性	说明	缺省值	Dynamic 或 Static
CACHE_COLUMN_METADATA	如果重复使用执行 SELECT 查询的 PreparedStatement 或 CallableStatement 对象，则将 CACHE_COLUMN_METADATA 设置为 true 可以提高性能。设置为 true 时，该语句将会记住与第一次执行该语句时所返回的 SELECT 查询结果关联的 ResultSet Metadata 信息。以后执行该语句时，将会重用这些元数据，而不必重新构建。这可以通过使用更多内存来节省 CPU 时间。 在连接到 Adaptive Server 15.7 ESD #1 及更高版本时，应使用 SUPPRESS_ROW_FORMAT 连接属性。	False	Static
CANCEL_ALL	指定 Statement.cancel 方法的行为： • 如果 CANCEL_ALL 为 false，那么调用 Statement.cancel 将只取消调用其的 Statement 对象。因此，如果 stmtA 是 Statement 对象，stmtA.cancel 将取消执行数据库中 stmtA 所包含的 SQL 语句，而不影响其它语句。无论 stmtA 是在高速缓存中等待执行还是已开始执行并且正在等待结果，都会取消该命令。 • 如果 CANCEL_ALL 为 true，那么调用 Statement.cancel 将不仅取消调用其的对象，还将取消同一连接上已开始执行并且正在等待结果的任何其它 Statement 对象。 以下示例将 CANCEL_ALL 设置为“false”。*props* 是指定连接属性的 Properties 对象： `props.put("CANCEL_ALL", "false");` **注释** 若要取消执行某个连接上的所有 Statement 对象，而不管其是否已开始在服务器上执行，请使用扩展方法 SybConnection.cancel。	• True – 对于 JCONNECT_VERSION <= “3” • False – 对于 JCONNECT_VERSION >= “4”	Static

属性	说明	缺省值	Dynamic 或 Static
CAPABILITY_TIME	仅在 JCONNECT_VERSION >= 6 时使用。此时 jConnect 与支持 TIME 数据类型的服务器相连，并且所有类型为 java.sql.Time 或 escape literals {t ...} 的参数都被作为 TIME 进行处理。 jConnect 的以前版本将这些参数作为 DATETIME 进行处理并在 java.sql.Time 参数前加上 “1970-01-01”。如果基础数据类型为 datetime 或 smalldatetime，则日期部分也会存储在数据库中。在 jConnect 6.0 或更高版本中，在处理 TIME 时，服务器会将时间转换成基础数据类型并在前面加上其自己的基准年。这可能导致旧数据与新数据之间不兼容。如果要对 java.sql.Time 使用 datetime 或 smalldatetime 数据类型，为了能够向后兼容，您应将 CAPABILITY_TIME 保留为 *false*。将此属性保留为 *false* 可强制 jConnect 将 java.sql.Time 参数或**转义文字** {t ...} 作为 DATETIME 进行处理，而不论服务器处理 TIME 数据类型的能力如何。 将此属性设置为 *true* 会使 jConnect 在连接到 Adaptive Server 时将 java.sql.Time 参数作为 TIME 数据类型进行处理。如果您要使用 smalldatetime 或 datetime 列来存储时间值， Sybase 建议您将此属性保留为 *false*。	False	Static
CAPABILITY_ WIDETABLE	如果您为了提高性能而不使用 JDBC ResultSetMetaData （如列名），则可以将此属性设置为 “false”。这将减少通过网络交换的数据并提高性能。除非要使用 EAServer，否则 Sybase 建议使用缺省设置。请参见第 45 页的 “使用 Adaptive Server 的宽表支持”。	False	Static
CHARSET	为传递给数据库的字符串指定字符集。如果 CHARSET 值为 Null， jConnect 将使用服务器的缺省字符集向服务器发送 string 数据。如果指定 CHARSET，数据库必须能够处理此格式的字符。如果数据库不能处理此格式的字符，将生成一条消息，说明不能正确完成字符转换。 **注释** 如果使用的是 jConnect 6.05 或更高版本，并将 DISABLE_UNICHAR_SENDING 设置为 false，那么当客户端试图向服务器发送无法用连接所使用的字符集表示的字符时， jConnect 将能够检测出来。发生这种情况时， jConnect 会将字符数据作为 unichar 数据发送给服务器，这样可使客户端能够在 unichar/univarchar 列和参数中插入 Unicode 数据。	Null	Static

属性	说明	缺省值	Dynamic 或 Static
CHARSET_CONVERTER_CLASS	指定希望 jConnect 使用的字符集转换程序类。jConnect 使用来自 SybDriver.setVersion 的版本设置或随 JCONNECT_VERSION 属性传入的版本来确定要使用的缺省字符集转换程序类。有关详细信息，请参见第 35 页的 “选择字符集转换程序”。	视版本而定。请参见第 6 页的表 2-1。	Static
CLASS_LOADER	该属性设置为您创建的 DynamicClassLoader 对象。DynamicClassLoader 用于装载在应用程序启动时存储在数据库中但不在 CLASSPATH 中的 Java 类。有关详细信息，请参见第 87 页的 “使用动态类装载”。	Null	Static
CONNECTION_FAILOVER	与 Java 命名和目录接口 (JNDI) 一起使用。请参见第 31 页的 “CONNECTION_FAILOVER 连接属性”。	True	Static
CRC	当该属性设置为 true 时，返回的更新计数为累计计数，其中既包括受执行的语句直接影响的更新，也包括执行语句后调用的所有触发器。	false	Static
DATABASE	当从 Sybase *interfaces* 文件获得连接信息时，使用该属性指定连接的数据库名称。 *interfaces* 文件的 URL 无法提供数据库名称。	*null*	Static
DEFAULT_QUERY_TIMEOUT	当设置该连接属性时，它将作为在该连接上创建的任何语句的缺省查询超时。	*0*（无超时）	Dynamic
DELETE_WARNINGS_FROM_EXCEPTION_CHAIN	指定是要保留 SQLWarning 还是将其从 SQLException 链中删除。 值： • True – jConnect 会将 SQLWarning 对象从 SQLException 链中删除。 • False – jConnect 会将 SQLWarning 对象保留在 SQLException 链中。	True	Static
DISABLE_UNICHAR_SENDING	当客户端应用程序向服务器发送 unichar 字符 （以及非 unichar 字符）时，会对发送到数据库的任何字符数据产生轻微的性能影响。在 jConnect 6.05 和更高版本中，此属性在缺省情况下设置为 false。如果使用 jConnect 较早版本的客户端希望向数据库发送 unichar 数据，则必须将此属性设置为 false。请参见第 33 页的 “使用 jConnect 传递 Unicode 数据”。	视版本而定	Static
DISABLE_UNPROCESSED_PARAM_WARNINGS	禁用警告。在处理存储过程的结果时， jConnect 经常读取行数据之外的返回值。如果不处理返回值，jConnect 将引发一个警告。若要禁用这些警告 （这样有助于提高性能），请将此属性设置为 “true”。	False	Static
DYNAMIC_PREPARE	决定是否在数据库中预编译动态 SQL 预准备语句。请参见第 141 页的 “DYNAMIC_PREPARE 连接属性”。	true	Dynamic

属性	说明	缺省值	Dynamic 或 Static
EARLY_BATCH_READ_THRESHOLD	指定行数阈值，超出该值后，读取器线程应启动，以清除批处理的服务器响应。 如果无需较早读取，则将该值设置为 -1。	-1	Static
ENABLE_BULK_LOAD	指定是否使用 bulk load 向数据库中插入行。值： • 空值 – 禁用批量装载。 • ARRAYINSERT_WITH_MIXED_STATEMENTS – 使用行级别日志记录启用批量装载，并允许应用程序在批量装载操作过程中执行其它语句。 • ARRAYINSERT – 使用行级别日志记录启用 bulk load，但应用程序在执行 bulk load 操作期间无法执行其它语句。 • BCP – 使用页面级别日志记录启用批量装载，但应用程序在批量装载操作过程中无法执行其它语句。 • LOG_BCP – 除对整个事务进行转储以供可能的完全恢复外，其余和 BCP 相同。	Null	Dynamic
ENABLE_LOB_LOCATORS	指定 jConnect 应该创建客户端物化 LOB 还是服务器端 LOB 定位符。有效值： • False：缺省值 – jConnect 使用客户端物化 LOB。即， LOB 的所有数据均在客户端进行处理和高速缓存。 • True：仅在自动提交设置为 False 时有效，否则，会在内部将值更改为 False。设置为 true 后，将使用服务器定位符，而不是在客户端存储 LOB 数据。 请参见第 69 页的 “大对象定位符支持”。	False	Dynamic
ENABLE_SERVER_PACKETSIZE	指定是否将连接包大小设置为服务器建议的值。如果设置为 true，驱动程序将不使用 PACKETSIZE 连接属性，并且服务器可以使用介于 512 和最大包大小之间的任何值。如果设置为 false，将使用 PACKETSIZE 连接属性。	True	Static
ENCRYPT_PASSWORD	允许安全登录。当该属性设置为 true 时，登录和远程站点口令都会被加密，然后再发送到服务器。不再以明文形式发送这些口令。 ENCRYPT_PASSWORD 优先于 RETRY_WITH_NO_ENCRYPTION。有关口令加密的详细信息，请参见第 80 页的 “使用口令加密”。	False	Static

属性	说明	缺省值	Dynamic 或 Static
ESCAPE_PROCESSING_DEFAULT	避免处理 SQL 语句中的 JDBC 函数转义。缺省情况下， jConnect 会分析提交到数据库的所有 SQL 语句，以查找有效的 JDBC 函数转义。如果应用程序不在其 SQL 调用中使用 JDBC 函数转义，可将此连接属性设置为 “false”以避免此处理过程。这样做可以使性能获得略微改善。 此外， ESCAPE_PROCESSING_DEFAULT 可帮助在 SQL 语法中使用大括号的后端服务器 （如 Sybase IQ）	True	Static
EXECUTE_BATCH_PAST_ERRORS	指定 jConnect 允许批处理更新操作忽略在执行各个语句时遇到的非致命错误并完成批处理更新，还是中止批处理更新操作。值： • True – 允许批处理更新操作忽略遇到的非致命错误并完成批处理更新。 • False – 在遇到非致命错误时中止批处理更新。	False	Static
EXPIRESTRING	包含许可证有效 date。除 jConnect 的评估副本之外，其余副本的有效期均设置为 Never。这是一个只读属性。	Never	Static
FAKE_METADATA	返回假元数据。如果在调用 ResultSetMetaData 方法 getCatalogName、 getSchemaName 和 getTableName 时此属性设置为 “true”，由于服务器不提供有用的元数据，调用过程将返回空字符串 ("")。 如果此属性设置为 “false”，那么调用这些方法将引发 “Not Implemented” SQLException。 **注释** 如果已启用宽表且正在使用 Adaptive Server 12.5 或更高版本，则将忽略此属性设置，因为服务器提供了有用的元数据。	False	Static
GET_BY_NAME_USES_COLUMN_LABEL	提供与 jConnect 6.0 之前版本的向后兼容性。 在 Adaptive Server 12.5 和更高版本中， jConnect 跟以前相比可以访问更多的元数据。在 12.5 版本之前， column name 和 column alias 代表同一数据类型。现在在使用 Adaptive Server 12.5 或更高版本且启用宽表的情况下， jConnect 可以区分这两者。 若要保留向后兼容性，请将此属性设置为 “true”。 如果希望调用 getByte、 getInt、 get* (String columnName) 以查看列的实际名称，请将此属性设置为 “false”。	True	Static

属性	说明	缺省值	Dynamic 或 Static
GET_COLUMN_LABEL_FOR_NAME	保持与 jConnect 5.5 或更早版本的向后兼容，其中对 ResultMetaData.getColumnName 的调用将返回列标签而不是列名称。值： • True – ResultMetaData.getColumnName 返回列标签 • False – ResultMetaData.getColumnName 返回列名	False	Static
GSSMANAGER_CLASS	指定 org.ietf.jgss.GSSManager 类的一种第三方实现。 可将此属性设置为字符串或 GSSManager 对象。 如果将此属性设置为字符串，其值应是第三方 GSSManager 实现的全限定类名。如果将此属性设置为对象，则该对象必须扩展 org.ietf.jgss.GSSManager 类。有关详细信息，请参见第 3 章 “安全性”。	Null	Static
HOMOGENEOUS_BATCH	调用新的 ASE 优化批处理协议以加快 PreparedStatement 对象的批处理操作速度。有效值： • True – 使用新的优化批处理协议。 • False – 即使 jConnect 连接到支持新优化批处理协议的 ASE，也使用旧的批处理协议。 请参见第 143 页的 “jConnect 中的优化批处理”。	True	Dynamic
HOSTNAME	标识当前主机名。	无。 最大长度是 30 个字符，如果超过则被截断至 30 个字符。	Static
HOSTPROC	标识主机上的应用程序进程。	无	Static
IGNORE_DONE_IN_PROC	确定不返回中间更新结果 （像在存储过程中那样），而只返回最终结果集。	False	Static

属性	说明	缺省值	Dynamic 或 Static
IGNORE_WARNINGS	指定是否检查并生成警告消息。目前，此属性仅检查有关将时间戳值存储为 Adaptive Server date 和 time 数据类型 （其精度低于 Java 时间戳）时的精度损失的警告。 有效值： • True – jConnect 不检查和生成警告消息，因此提高了性能。 • False – 缺省值，指示 jConnect 检查并生成警告消息。 **注释** 在将 IGNORE_WARNINGS 设置为 true 前，应就此类配置对应用程序产生的影响进行全面测试。	False	Static
IMPLICIT_CURSOR_FETCH_SIZE	将该属性与 SELECT_OPENS_CURSOR 属性结合使用，可强制 jConnect 在发送到数据库的每个 select 查询上打开一个只读游标。该游标具有在此属性中设置的值提取大小，除非使用 Statement.setFetchSize 方法加以覆盖。	*0*	Static
INTERNAL_QUERY_TIMEOUT	使用该属性可设置将由在内部创建并用 jConnect 执行的语句所使用的查询超时。如果内部命令没有在适当的时间内完成，这可以防止应用程序失败。	*0* （无超时）	Dynamic
IS_CLOSED_TEST	允许指定在调用 Connection.isClosed 时向数据库发送何种查询 （如果有）。有关详细信息，请参见 第 101 页的 “使用 Connection.isClosed 和 IS_CLOSED_TEST”。	Null	Static
J2EE_TCK_COMPLIANT	当该属性设置为 true 时， jConnect 驱动程序会启用符合 J2EE 1.4 技术兼容包 (TCK) 测试套件的行为，这会导致性能有所下降。因此， Sybase 建议使用缺省值 false。	false	Static
JAVA_CHARSET_MAPPING	指定用户定义的字符集映射，取代 Adaptive Server 的缺省字符集映射。请参见第 39 页的 “取代缺省字符集映射”。	-	Static
JCE_PROVIDER_CLASS	指定 RSA 加密算法中使用的 Java Cryptography Extension (JCE) 提供程序。	绑定的 JCE 提供程序。	Static
JCONNECT_VERSION	设置版本专有的特性。请参见第 6 页的 “使用 JCONNECT_ VERSION”。	7	Static
LANGUAGE	指定来自 jConnect 和服务器的消息的显示语言。该设置必须与 *syslanguages* 中的语言相符，因为服务器消息将根据您的当地环境中的语言设置进行本地化。支持的语言包括：中文、美国英语、法语、德语、日语、韩语、波兰语、葡萄牙语和西班牙语。	视版本而定。请参见第 6 页的 “使用 JCONNECT_ VERSION”。	Static

属性	说明	缺省值	Dynamic 或 Static
LANGUAGE_CURSOR	确定 jConnect 使用 “语言游标” 而不使用 “协议游标”。 请参见第 144 页的 “游标性能”。	False	Static
LITERAL_PARAMS	如果设置为 “true”，那么 setXXX 方法在 PreparedStatement 接口中设置的任何参数都会在 SQL 语句执行时以文字形式插入该语句。 如果设置为 “false”，参数标记将留在 SQL 语句中，而参数值被单独发送给服务器。	False	Static
NEWPASSWORD	指定口令有效期处理过程中使用的新口令。	Null	Static
OPTIMIZE_FOR_PERFORMANCE	指定是否启用 jConnect 性能增强属性。目前，此属性仅控制 IGNORE_WARNINGS 属性。 有效值： • True – jConnect 在增强性能模式下运行。 • False – 缺省值，表示 jConnect 在正常模式下运行。 **注释** 在将 OPTIMIZE_FOR_PERFORMANCE 设置为 true 前，应就此类配置对应用程序产生的影响进行全面测试。	False	Static
OPTIMIZE_STRING_CONVERSIONS	指定是否启用字符串转换优化。 客户端在 SQL 预准备语句中使用字符数据类型时，此优化行为可提高 jConnect 的性能。 值： • 0 – 缺省值；不启用字符串转换优化。 • 1 – 在 jConnect 使用 utf8 或服务器缺省字符集时，启用字符串转换优化。 • 2 – 在所有情况下均启用字符串转换优化。	0	Static
PACKETSIZE	标识网络包大小。如果使用的是 Adaptive Server 15.0 或更高版本， Sybase 建议您不要设置此属性，并让 jConnect 和 Adaptive Server 选用适合于您的环境的网络包大小。	512	Static
PASSWORD	标识登录口令。 如果使用 getConnection(String, String, String) 方法，将自动设置；如果使用 getConnection(String, Props)，将显式设置。	无	Static

属性	说明	缺省值	Dynamic 或 Static
PRELOAD_JARS	包含以逗号分隔的与指定 CLASS_LOADER 相关联的 *.jar* 文件名的列表。这些 *.jar* 文件在连接时装载，可供使用相同 jConnect 驱动程序的任何其它连接使用。有关详细信息，请参见第 89 页的 “预装载 .jar 文件”。	Null	Static
PROMPT_FOR_NEWPASSWORD	指定是执行透明口令更改还是提示输入新口令。值： • True – 提示手动设置新口令。 • False – jConnect 检查 NEWPASSWORD 的值，如果不是空值，则使用该值替换到期口令。	False	Static
PROTOCOL_CAPTURE	指定用于捕获应用程序和 Adaptive Server 间的 TDS 通信的文件。	Null	Dynamic
PROXY	指定网关地址。对于 HTTP 协议， URL 是 http://host:port。 若要使用支持加密的 HTTPS 协议， URL 应为 https://host:port/servlet_alias。	无	Static
QUERY_TIMEOUT_CANCELS_ALL	强制 jConnect 在读取超时时取消连接上的所有语句。可以在客户端调用 execute() 但由于死锁 （例如，试图读取当前正在另一个事务中更新的表）而超时时使用此行为。	False	Dynamic
RELEASE_LOCKS_ON_CURSOR_CLOSE	指定在游标关闭时 Adaptive Server 是否在隔离级别 2 和 3 释放共享只读游标锁： • False – 在游标关闭时不释放共享游标锁。 • True – 在游标关闭时释放共享游标锁。 请参见第 54 页的 “在游标关闭时释放锁”。	False	Static
REMOTEPWD	包含用于通过服务器到服务器远程过程调用进行访问的远程服务器口令。请参见第 44 页的 “执行服务器到服务器的远程过程调用”。	无	Static
REPEAT_READ	确定驱动程序是否保留列和输出参数的副本，以便可以随机读取或重复读取列。请参见第 136 页的 “REPEAT_READ 连接属性”。	True	Static

属性	说明	缺省值	Dynamic 或 Static
REQUEST_HA_SESSION	指示连接客户端是否希望开始高可用性 (HA) 故障切换会话。请参见第 40 页的 “实施高可用性故障切换支持”。 建立连接后将不能重置此属性。如果希望请求故障切换会话时具有更大的灵活性，可以对客户端应用程序进行编码，使其在运行期设置 REQUEST_HA_SESSION。 **注释** 此属性设置为 “true” 将导致 jConnect 尝试进行故障切换登录。如果没有正确设置此连接属性，即使已为故障切换配置了服务器，也不能启动故障切换会话。	False	Static
REQUEST_KERBEROS_SESSION	确定 jConnect 是否使用 Kerberos 进行鉴定。如果此属性设置为 “true”，还必须指定 SERVICE_PRINCIPAL_NAME 属性的值。 可能还要为 GSSMANAGER_CLASS 属性提供值。有关详细信息，请参见第 3 章 “安全性”。	False	Static
RETRY_WITH_NO_ENCRYPTION	允许服务器使用明文口令重新尝试登录 当 ENCRYPT_PASSWORD 和 RETRY_WITH_NO_ENCRYPTION 属性都设置为 true 时， jConnect 会首先使用加密口令登录。如果登录失败， jConnect 将使用明文口令登录。有关口令加密的详细信息，请参见第 80 页的 “使用口令加密”。	False	Static
RMNAME	在使用分布式事务 (XA) 时设置资源管理器名称。此属性将覆盖可能在 LDAP 服务器条目中设置的资源管理器名称。有关详细信息，请参见第 99 页的 “分布式事务管理支持”。	Null	Static
SECONDARY_SERVER_HOSTPORT	在客户端使用 HA 故障切换会话时设置辅助服务器的主机名和端口。此属性的值应采用下列格式：hostName:portNumber。除非 REQUEST_HA_SESSION 也设置为 “true”，否则将忽略此属性。有关详细信息，请参见第 40 页的 “实施高可用性故障切换支持”。	Null	Static

属性	说明	缺省值	Dynamic 或 Static
SELECT_OPENS_CURSOR	确定对 Statement.executeQuery 的调用在查询包含 FOR UPDATE 子句时是否自动生成游标。 如果前面已经在同一语句中调用了 Statement.setFetchSize 或 Statement.setCursorName，那么将 SELECT_OPENS_CURSOR 设置为 “true”将不起作用。 **注释** 将 SELECT_OPENS_CURSOR 设置为 “true”时，系统性能可能会下降。 有关在 jConnect 中使用游标的详细信息，请参见 第 47 页的 “对结果集使用游标”。	False	Static
SEND_BATCH_IMMEDIATE	指定 jConnect 是在调用 PreparedStatement.addBatch() 后立即为当前行发送参数，还是仅在调用 PreparedStatement.executeBatch() 后再发送。 • True – jConnect 在调用 PreparedStatement.addBatch() 后立即为当前行发送参数。这会最大限度地减少客户端内存的使用率，并让服务器有更多时间来处理批处理参数。 • False – jConnect 仅在调用 PreparedStatement.executeBatch() 后才发送批处理参数。	False	Dynamic
SERIALIZE_REQUESTS	确定 jConnect 在发送其它请求之前是否等待服务器的响应。	False	Static
SERVER_INITIATED_TRANSACTIONS	允许服务器控制事务。缺省情况下，该属性设置为 true，并且 jConnect 通过使用 Transact-SQL 命令 set chained on 来允许服务器启动并控制事务。如果设置为 false，则 jConnect 通过使用 Transact-SQL 命令 begin tran 启动并控制事务。Sybase 建议允许服务器控制事务。	True	Static
SERVICENAME	指示 DirectConnect 网关服务的后台数据库服务器的名称。还用于指示在连接到 SQL Anywhere 之后应使用的数据库。	无	Static
SERVERTYPE	在连接到 OpenSwitch 时将此属性设置为 “OSW”。这使得 jConnect 可以向 OpenSwitch 发送某些指令，允许 OpenSwitch 在将连接重定向到另一个服务器实例时记住初始的连接设置，例如隔离级别、 textsize、带引号的标识符和 autocommit。	无	Static

属性	说明	缺省值	Dynamic 或 Static
SERVICE_PRINCIPAL_NAME	在建立到 Adaptive Server 的 Kerberos 连接时使用。此属性的值应与密钥分发中心 (KDC) 中的服务器条目和数据库运行时所使用的服务器名称对应。 如果 REQUEST_KERBEROS_SESSION 属性设置为“false”，将会忽略 SERVICE_PRINCIPAL_NAME 属性的值。有关详细信息，请参见第 3 章 “安全性”。	Null	Static
SESSION_ID	TDS 会话 ID。如果设置此属性，jConnect 将认为应用程序正试图恢复由 TDS 贯通网关保持打开的现有 TDS 会话的通信。jConnect 将跳过登录协商并将所有来自应用程序的请求转发到指定的会话 ID。	Null	Static
SESSION_TIMEOUT	指定 HTTP 贯通会话（使用 jConnect TDS 贯通服务器小程序创建）在空闲时保持活动的时间量 （以秒为单位）。在达到指定时间后，连接会自动关闭。有关 TDS 贯通服务器小程序的详细信息，请参见第 151 页的 “使用 TDS 贯通”。	Null	Static
SETMAXROWS_AFFECTS_SELECT_ONLY	指定 setMaxRows 是否仅限制 select 语句返回的行，以与 JDBC 规范一致。值： • True – Statement.setMaxRows(int max) 仅限制作为 select 语句的结果返回的行数。 • False – Statement.setMaxRows(int max) 对作为 select、 insert、 update 和 delete 语句的结果返回的行数加以限制。 当连接到 Adaptive Server 15.5 或更低版本时，SETMAXROWS_AFFECTS_SELECT_ONLY 会被忽略。	True	Static
SQLINITSTRING	定义要在连接打开时传递给数据库服务器的一组命令。这些命令必须是可以用 Statement.executeUpdate 方法执行的 SQL 命令。	Null	Static
STREAM_CACHE_SIZE	指定用于高速缓存语句响应流的最大大小。	Null （无限制的高速缓存大小）	Dynamic
STRIP_BLANKS	强制服务器在将字符串值存储到表中之前，先删除其前导空白和尾随空白。 值： • 0 – 缺省值；客户端发送的字符串值 “按原样”存储。 • 1 – 在将字符串值存储到表中之前，先删除其前导空白和尾随空白。	0	Static

属性	说明	缺省值	Dynamic 或 Static
SUPPRESS_CONTROL_TOKEN	取消发送控制令牌。 值： • 0 – 缺省值；发送控制令牌。 • 1 – 取消发送控制令牌。	0	Static
SUPPRESS_PARAM_FORMAT	在执行动态 SQL 预准备语句时，jConnect 客户端可使用 SUPPRESS_PARAM_FORMAT 连接字符串属性来抑制参数格式元数据。在可能的情况下，客户端会减少发送的参数元数据以提高性能。 值： • 0 – 在 select、 insert 和 update 操作中不抑制参数格式元数据。 • 1 – 缺省值；在可能的情况下抑制参数格式元数据。	1	Static
SUPPRESS_ROW_FORMAT	在 jConnect 中，客户端可使用 SUPPRESS_ROW_FORMAT 连接字符串属性来强制 Adaptive Server 仅在动态 SQL 预准备语句的行格式更改时发送 TDS_ROWFMT 或 TDS_ROWFMT2 数据。这样， Adaptive Server 可以尽量向客户端发送较少的数据，从而提高性能。 值： • 0 – 即便行格式未发生更改，也发送 TDS_ROWFMT 或 TDS_ROWFMT2 数据。 • 1 – 缺省值；强制服务器仅在行格式更改时才发送 TDS_ROWFMT 或 TDS_ROWFMT2 数据。	1	Static
SUPPRESS_ROW_FORMAT2	指定 Adaptive Server 尽可能使用 TDS_ROWFMT 字节序列而不是 TDS_ROWFMT2 字节序列发送数据。 值： • 0 – 缺省值；不抑制 TDS_ROWFMT2。 • 1 – 强制服务器尽可能以 TDS_ROWFMT 格式发送数据。 **注释** 在连接到 Adaptive Server 15.7 ESD #1 及更高版本时，应改用 SUPPRESS_ROW_FORMAT 连接属性。	0	Static

属性	说明	缺省值	Dynamic 或 Static
SYBSOCKET_FACTORY	使 jConnect 能够使用自定义套接字实现。 将 SYBSOCKET_FACTORY 设置为下列两项之一： ● 实现 com.sybase.jdbcx.SybSocketFactory 的类的名称；或 ● “DEFAULT”，设置为此值将实例化新的 java.net.Socket( ) 使用此属性可建立到数据库的 SSL 连接。	Null	Static
TEXTSIZE	允许设置文本大小。缺省情况下， Adaptive Server 和 SQL Anywhere 允许从图像或文本列中读取 32,627 字节。如果已经安装了 jConnect mda 表，jConnect 会将该值更改为 2GB。但是，如果在连接到 OpenSwitch 时设置该值，则允许连接在 OpenSwitch 将连接重定向到另一个服务器实例时记住此设置。	2GB	Static
USE_METADATA	在建立连接时创建并初始化 DatabaseMetaData 对象。连接到指定的数据库必须要使用 DatabaseMetaData 对象。 jConnect 将 DatabaseMetaData 用于某些功能，如分布式事务管理支持 (JTA/JTS) 和动态类装载 (DCL)。 如果收到错误 010SJ （表明应用程序需要元数据），请安装 jConnect 附带的用于返回元数据的存储过程。请参见 《jConnect for JDBC 安装指南》第 3 章中的 “安装存储过程”。	True	Static
USER	指定登录 ID。 如果使用 getConnection(String, String, String) 方法，将自动设置；如果使用 getConnection(String, Props)，将显式设置。	无	Static
VERSIONSTRING	提供 JDBC 驱动程序的只读版本信息。	jConnect 驱动程序版本	Static

连接到 Adaptive Server

在 Java 应用程序中，定义一个使用 jConnect 驱动程序连接到 Adaptive Server 的 URL。此 URL 的基本格式如下：

```
jdbc:sybase:Tds:host:port
```

其中：

- jdbc:sybase 标识驱动程序。

- Tds 是 Adaptive Server 的 Sybase 通信协议。
- *host:port* 是 Adaptive Server 主机名和监听端口。有关数据库或 Open Server 应用程序使用的条目，请参见 *$SYBASE/interfaces* (UNIX) 或 *%SYBASE%\ini\sql.ini* (Windows)。从 “query” 条目可获取 *host:port*。

可使用下面的格式连接到特定数据库：

```
jdbc:sybase:Tds:host:port/database
```

注释 若要使用 SQL Anywhere 或 DirectConnect 连接到特定数据库，请使用 SERVICENAME 连接属性而不是 “/database” 来指定数据库名称。

示例

以下代码将创建到主机 “myserver” 上的监听端口 3697 的 Adaptive Server 的连接。

```
SysProps.put("user","userid");
SysProps.put("password","user_password");
String url = "jdbc:sybase:Tds:myserver:3697";
Connection_con =
   DriverManager.getConnection(url,SysProps);
```

URL 连接属性参数

可以在定义 URL 时指定 jConnect 驱动程序连接属性的值。

注释 在 URL 中设置的驱动程序连接属性不会覆盖在应用程序中使用 DriverManager.getConnection 方法设置的任何相应的连接属性。

若要在 URL 中设置连接属性，请将属性名及其值附加到 URL 定义中。使用以下语法：

```
jdbc:sybase:Tds:host:port/database?
   property_name=value
```

要设置多个连接属性，请附加每个额外的连接属性及其值 （前面加上 “&”）。例如：

```
jdbc:sybase:Tds:myserver:1234/mydatabase?
   LITERAL_PARAMS=true&PACKETSIZE=512&HOSTNAME=myhost
```

如果其中某个连接属性的值包含 “&”，请在该连接属性值的 “&” 前加一个反斜杠 (\)。例如，如果主机名是 “a&bhost”，则使用下列语法：

```
jdbc:sybase:Tds:myserver:1234/mydatabase?
   LITERAL_PARAMS=true&PACKETSIZE=512&HOSTNAME=
   a\&bhost
```

请勿对连接属性值使用引号，即使这些值是字符串。例如，使用：

```
HOSTNAME=myhost
```

而不是：

```
HOSTNAME="myhost"
```

使用 *sql.ini* 和 *interfaces* 文件目录服务

可以使用 *sql.ini* 文件（对于 Windows）和 *interfaces* 文件（对于 UNIX）为 jConnect for JDBC 提供服务器信息。通过使用 *sql.ini* 或 *interfaces* 文件，企业可以集中管理企业网络中的可用服务的相关信息（包括 Adaptive Server 信息）。

使用连接字符串标识 *sql.ini* 或 *interfaces* 文件。在 jConnect for JDBC 上，只能连接到单个目录服务 URL (DSURL)。

用于 jConnect 的单个 DSURL 的连接字符串

在连接到 DSURL 时，您必须指定 *sql.ini* 或 *interfaces* 文件的路径和服务器名称。否则， jConnect 将返回错误。

以下命令指定 *sql.ini* 文件的路径：

```
String url = "jdbc:sybase:jndi:file://D:/syb1252/ini/mysql.ini?myaseISO1"
```

其中：

- 服务器名称为 myaseISO1
- *sql.ini* 文件路径为 *file://D:/syb1252/ini/sql.ini*

以下命令指定 *interfaces* 文件的路径：

```
String url = "jdbc:sybase:jndi:file:///work/sybase/interfaces?myase"
```

其中：

- 服务器名称为 myase
- *interfaces* 文件路径为 *file:///work/sybase/interfaces*

适用于 SSL 的 *sql.ini* 和 *interfaces* 文件的格式

以下是用于 *SSL* 的 *sql.ini* 文件的格式：

```
[SYBSRV2]
master=nlwnsck,mango1,4100,ssl
query=nlwnsck,mango1,4100,ssl
query=nlwnsck,mango1,5000,ssl
```

interfaces 文件的格式为：

```
sybsrv2
master tcp ether mango1 5000 ssl
query tcp ether mango1 4100 ssl
query tcp ether mango1 5000 ssl
```

注释 jConnect 支持在 *sql.ini* 或 *interfaces* 文件中的同一服务器名称下存在多个查询条目。jConnect 尝试按照 *sql.ini* 或 *interfaces* 文件中的顺序，通过查询条目连接到 host 或 port 值。如果 jConnect 在查询条目中找到 SSL，它将通过指定应用程序特定的套接字工厂来要求对应用程序进行编码以处理 SSL 连接，否则，连接可能会失败。

使用 JNDI 连接到服务器

在 jConnect 中，您可以使用 Java 命名和目录接口 (JNDI) 来提供连接信息，它可以提供：

- 一个集中位置，可以从中指定主机名和端口以连接到服务器。不需要在应用程序中添加特定的主机和端口号的代码。
- 一个集中位置，可以从中指定供所有应用程序使用的连接属性和缺省数据库。
- 用于处理不成功的连接尝试的 jConnect CONNECTION_FAILOVER 属性。 CONNECTION_FAILOVER 设置为 “true” 时， jConnect 将尝试连接到 JNDI 名称空间中的主机/端口服务器地址序列，直到连接成功。

要配合使用 jConnect 和 JNDI，需要确保在 JNDI 访问的任何目录服务中都能获得特定的信息，并且在 javax.naming.Context 类中设置了所需的信息。本节包含以下主题：

- 使用 JNDI 的连接 URL
- 所需的目录服务信息

- CONNECTION_FAILOVER 连接属性
- 提供 JNDI 环境信息

使用 JNDI 的连接 URL

若要指定 jConnect 应使用 JNDI 获得连接信息，请将 “jndi”作为 URL 协议放在 “sybase”后面：

```
jdbc:sybase:jndi:protocol-information-for-use-with-JNDI
```

URL 中 “jndi”后的任何内容都将通过 JNDI 进行处理。例如，若要使用 JNDI 处理轻量目录访问协议 (LDAP)，可以输入以下代码：

```
jdbc:sybase:jndi:ldap://LDAP_hostname:port_number/servername=
  Sybase11,o=MyCompany,c=US
```

此 URL 告知 JNDI 从 LDAP 服务器获取信息，提供要使用的 LDAP 服务器的主机名和端口号，并且以特定于 LDAP 形式的提供数据库服务器的名称。

所需的目录服务信息

当配合使用 JNDI 和 jConnect 时，JNDI 需要为目标数据库服务器返回以下信息：

- 要连接到的主机名和端口号
- 要使用的数据库的名称
- 不允许单个应用程序自己设置的任何连接属性

此信息需要按照固定格式存储在任何用于提供连接信息的目录服务中。要求的格式包括一个数值对象标识符 (OID)，它标识所提供的信息的类型（如目标数据库），随后是格式化信息。

注释 可以使用别名代替 OID 来引用属性。

表 2-3 显示了要求的格式。

表 2-3：JNDI 的目录服务信息

属性说明	别名	OID (object_id)
LDAP 目录服务中的接口条目替换	sybaseServer	1.3.6.1.4.1.897.4.1.1
sybaseServer LDAP 属性的集合点	sybaseServer	1.3.6.1.4.1.897.4.2
版本属性	sybaseVersion	1.3.6.1.4.1.897.4.2.1
服务器名称属性	sybaseServer	1.3.6.1.4.1.897.4.2.2
服务属性	sybaseService	1.3.6.1.4.1.897.4.2.3
状态属性	sybaseStatus	1.3.6.1.4.1.897.4.2.4
地址属性	sybaseAddress	1.3.6.1.4.1.897.4.2.5
安全性机制属性	sybaseSecurity	1.3.6.1.4.1.897.4.2.6
重试次数属性	sybaseRetryCount	1.3.6.1.4.1.897.4.2.7
循环延迟属性	sybaseRetryDelay	1.3.6.1.4.1.897.4.2.8
jConnect 连接协议	sybaseJconnectProtocol	1.3.6.1.4.1.897.4.2.9
jConnect 连接属性	sybaseJconnectProperty	1.3.6.1.4.1.897.4.2.10
数据库名	sybaseDatabasename	1.3.6.1.4.1.897.4.2.11
高可用性故障切换服务器名称属性	sybaseHAservername	1.3.6.1.4.1.897.4.2.15
ResourceManager 名称	sybaseResourceManager Name	1.3.6.1.4.1.897.4.2.16
ResourceManager 类型	sybaseResourceManager Type	1.3.6.1.4.1.897.4.2.17
JDBCDataSource 接口	sybaseJdbcDataSource-Interface	1.3.6.1.4.1.897.4.2.18
ServerType	sybaseServerType	1.3.6.1.4.1.897.4.2.19

注释 用斜体表示的属性是必需的。

以下示例演示为 LDAP 目录服务下的数据库服务器 “SYBASE11” 输入的连接信息。示例 1 使用的是属性 OID。示例 2 使用的是属性别名，它不区分大小写。可以使用 OID 也可以使用别名。

示例 1

```
dn:servername=SYBASE11,o=MyCompany,c=US
    servername:SYBASE11
   1.3.6.1.4.1.897.4.2.5:TCP#1#giotto 1266
   1.3.6.1.4.1.897.4.2.5:TCP#1#giotto 1337
   1.3.6.1.4.1.897.4.2.5:TCP#1#standby1 4444
   1.3.6.1.4.1.897.4.2.10:REPEAT_READ=false&
       PACKETSIZE=1024
    1.3.6.1.4.1.897.4.2.10:CONNECTION_FAILOVER=true
```

```
1.3.6.1.4.1.897.4.2.11:pubs2
1.3.6.1.4.1.897.4.2.9:Tds
```

示例 2

```
dn:servername=SYBASE11,o=MyCompany,c=US
servername:SYBASE11
sybaseAddress:TCP#1#giotto 1266
sybaseAddress:TCP#1#giotto 1337
sybaseAddress:TCP#1#standby1 4444
sybaseJconnectProperty:REPEAT_READ=false&
   PACKETSIZE=1024
sybaseJconnectProperty:CONNECTION_FAILOVER=true
sybaseDatabasename:pubs2
sybaseJconnectProtocol:Tds
```

在这些示例中，可通过主机 “giotto”上的端口 1266 或 1337 访问 SYBASE11，也可通过主机 “standby1”上的端口 4444 对其进行访问。REPEAT_READ 和 PACKETSIZE 这两个连接属性在一个条目中设置。CONNECTION_FAILOVER 连接属性是作为单独条目设置的。连接到 SYBASE11 的应用程序开始时是与 pubs2 数据库连接。不需要指定连接协议，但如果指定了，则必须以 “`Tds`”而不是 “`TDS`”的形式输入该属性。

CONNECTION_FAILOVER 连接属性

CONNECTION_FAILOVER 是布尔值连接属性，可以在 jConnect 使用 JNDI 获取连接信息时使用该属性。

如果 CONNECTION_FAILOVER 设置为 True，jConnect 将多次尝试连接到服务器。如果连接到与服务器关联的主机和端口号的尝试失败，jConnect 将使用 JNDI 获取与该服务器关联的下一个主机和端口号，并通过它们尝试连接。连接尝试将按顺序使用与服务器关联的所有主机和端口。

例如，如果 CONNECTION_FAILOVER 设置为 True，并且数据库服务器与下列主机和端口号相关联 （如前面 LDAP 示例中所述）：

```
1.3.6.1.4.1.897.4.2.5:TCP#1#giotto 1266
1.3.6.1.4.1.897.4.2.5:TCP#1#giotto 1337
1.3.6.1.4.1.897.4.2.5:TCP#1#standby 4444
```

为了连接到服务器， jConnect 会尝试通过端口 1266 连接到主机 “giotto”。如果失败， jConnect 会尝试 “giotto”上的端口 1337。如果再次失败， jConnect 就会尝试通过端口 4444 连接到主机 “standby1”。

CONNECTION_FAILOVER 的缺省值为 True。

如果 CONNECTION_FAILOVER 设置为 False， jConnect 将尝试连接到初始主机和端口号。如果尝试失败，jConnect 将抛出一个 SQL 例外并不再重试。

提供 JNDI 环境信息

若要与 JNDI 一起使用 jConnect，您应该熟悉JNDI specification from Oracle Technology Network for Java (http://www.oracle.com/technetwork/java/jndi/index.html)。

特别是当 JNDI 和 jConnect 一起使用时，需要确保在 javax.naming.directory.DirContext 中设置所需的初始化属性。这些属性可以在系统级或在运行期设置。

两个重要属性是：

- Context.INITIAL_CONTEXT_FACTORY

  此属性包含 JNDI 使用的初始环境工厂的全限定类名。这确定了在 Context.PROVIDER_URL 属性中指定的 URL 使用的 JNDI 驱动程序。

- Context.PROVIDER_URL

  此属性获取驱动程序（如 LDAP 驱动程序）要访问的目录服务的 URL。此 URL 应该是一个字符串，如 “ldap://ldaphost:427”。

以下示例演示如何在运行期设置环境属性以及如何使用 JNDI 和 LDAP 获取连接。在该示例中，INITIAL_CONTEXT_FACTORY 环境属性设置为调用 LDAP 服务提供程序的 Oracle 实施。PROVIDER_URL 环境属性被设置为位于主机 “ldap_server1” 上的 389 端口的 LDAP 目录服务的 URL。

```
Properties props = new Properties();

 /* We want to use LDAP, so INITIAL_CONTEXT_FACTORY is set to the
 * class name of an LDAP context factory.In this case, the
 * context factory is provided by Sun’ s implementation of a
 * driver for LDAP directory service.
 */
 props.put(Context.INITIAL_CONTEXT_FACTORY,
   "com.sun.jndi.ldap.LdapCtxFactory");

 /* Now, we set PROVIDER_URL to the URL of the LDAP server that
 * is to provide directory information for the connection.
 */
 props.put(Context.PROVIDER_URL, "ldap://ldap_server1:389");

 /* Set up additional context properties, as needed.*/
 props.put("user", "xyz");
 props.put("password", "123");

 /* get the connection */
 Connection con = DriverManager.getConnection
```

```
   ("jdbc:sybase:jndi:ldap://ldap_server1:389" +
   "/servername=Sybase11,o=MyCompany,c=US",props);
```

传递给 getConnection 的连接字符串包含开发人员必须提供的特定于 LDAP 的信息。

在运行期设置 JNDI 属性后 （如上例所示）， jConnect 将它们传递给要用于初始化服务器的 JNDI，如以下 jConnect 代码所示：

```
javax.naming.directory.DirContext ctx =
   new javax.naming.directory.InitialDirContext(props);
```

jConnect 然后会通过调用 DirContext.getAtributes 从 JNDI 获取所需的连接信息，如以下示例所示，其中 *ctx* 是一个 DirContext 对象：

```
javax.naming.directory.Attributes attrs =
   ctx.getAttributes("ldap://ldap_server1:389/servername=" +
     "Sybase11", SYBASE_SERVER_ATTRIBUTES);
```

此例中， SYBASE_SERVER_ATTRIBUTES 是在 jConnect 中定义的字符串数组。数组值是表 2-3 中列出的必需目录信息的 OID。

处理国际化和本地化

本节讨论与 jConnect 有关的国际化和本地化问题。

使用 jConnect 传递 Unicode 数据

在 Adaptive Server 12.5 和更高版本中，数据库客户端可以使用 unichar 和 univarchar 数据类型。这两种数据类型可以实现 Unicode 数据的有效存储和检索。

以下为从 Unicode 标准 （版本 2.0）中引用的一段内容：

“Unicode 标准是对字符和文本进行编码的具有固定宽度的一致的编码方案。这种用于信息处理的国际字符代码的指令系统包括当今世界上的主要脚本和常用的技术符号。 Unicode 字符编码将字母字符、表意字符和符号同等对待，这意味着它们可以混合使用，且使用的难易程度相同。Unicode 标准使用 ASCII 字符集的模型，但使用 16 位编码以完全支持多语言文本。”

这意味着用户可以指定数据库表列存储 Unicode 数据，而不用考虑服务器的缺省字符集。

注释 在 Adaptive Server 12.5 到 12.5.0.3 版本中，服务器必须具有缺省字符集 utf-8 才能使用 Unicode 数据类型。但在 Adaptive Server 12.5.1 和更高版本中，数据库用户无需考虑服务器的缺省字符集即可使用 unichar 和 univarchar 数据类型。

服务器接受 unichar 和 univarchar 数据时， jConnect 将执行以下操作：

- 对于客户端要发送到服务器的所有字符数据 – 例如，使用 PreparedStatement.setString (int column, String value) – jConnect 确定字符串是否能转换为服务器的缺省字符集。
- 如果 jConnect 确定这些字符不能转换为服务器的字符集 （例如，有些字符无法用服务器的字符集表示），它会将数据以 unichar/univarchar 数据编码发送给服务器。

例如，如果客户端尝试向以 iso_1 作为缺省字符集的 Adaptive Server 12.5.1 发送 Unicode 日语字符， jConnect 将检测到该日语字符不能转换为 iso_1 字符。 jConnect 随后以 Unicode 数据发送字符串。

客户端向服务器发送 unichar/univarchar 数据会降低计算机的性能。这是因为 jConnect 必须对不能直接映射到服务器的缺省字符集的所有字符串和字符执行两次字符到字节的转换。

如果您使用的是 6.05 以前的 jConnect 版本，并想要使用 unichar 和 univarchar 数据类型，您必须执行以下任务：

1 设置 JCONNECT_VERSION = 6 或更高版本。有关详细信息，请参见第 6 页的 “使用 JCONNECT_ VERSION”。

2 您需要将 DISABLE_UNICHAR_SENDING 连接属性设置为 false。从 jConnect 6.05 开始，此属性在缺省情况下设置为 false。有关详细信息，请参见第 10 页的 “设置连接属性”。

注释 有关 unichar 和 univarchar 数据类型支持的详细信息，请参见 Adaptive Server Enterprise 手册。

jConnect 字符集转换程序

jConnect 对所有字符集转换使用特殊类。通过选择字符集转换程序类，可指定 jConnect 处理单字节和多字节字符集转换的方式以及转换对应用程序性能的影响。

共有两个字符集转换类。 jConnect 使用的转换类基于 JCONNECT_VERSION、 CHARSET 和 CHARSET_CONVERTER_CLASS 连接属性。

- TruncationConverter 类只能用于使用 ASCII 字符的单字节字符集（如 iso_1 和 cp850）。它不能用于多字节字符集或使用非 ASCII 字符的单字节字符集。当 JCONNECT_VERSION 设置为 2 时，TruncationConverter 类是缺省转换程序。

  jConnect 7 使用 TruncationConverter 类以与 jConnect version 2.2 相同的方式处理字符集。当 JCONNECT_VERSION = 2 时，TruncationConverter 类是缺省转换程序。

- PureConverter 类是纯 Java 类多字节字符集转换程序。 jConnect 在 JCONNECT_VERSION = 4 或更高版本时使用此类。如果 jConnect 检测到在 CHARSET 连接属性中指定的字符集与 TruncationConverter 类不兼容，则在 JCONNECT_VERSION = 2 时也使用此转换程序。

  虽然 PureConverter 类能实现多字节字符集转换，但也可能降低 jConnect 驱动程序的性能。如果需要考虑驱动程序性能，请参见 第 36 页的 “提高字符集转换性能”。

选择字符集转换程序

jConnect 使用 JCONNECT_VERSION 来确定要使用的缺省字符集转换程序类。 JCONNECT_VERSION = 2 或更高版本时，缺省值为 PureConverter 和 TruncationConverter。JCONNECT_VERSION = 4 或更高版本时，缺省值为 PureConverter。

也可以通过设置 CHARSET_CONVERTER_CLASS 连接属性指定希望 jConnect 使用的字符集转换程序。如果希望使用 jConnect 版本的缺省字符集转换程序之外的字符集转换程序，此方法将很有用。

例如，如果您设置 JCONNECT_VERSION = 4 或更高版本，但要使用 TruncationConverter 类而不使用多字节的 PureConverter 类，则可以设置 CHARSET_CONVERTER_CLASS：

```
...
 props.put("CHARSET_CONVERTER_CLASS",
   "com.sybase.jdbc4.utils.TruncationConverter")
```

设置 CHARSET 连接属性

可以通过设置 CHARSET 驱动程序属性指定要在应用程序中使用的字符集。如果没有设置 CHARSET 属性：

- JCONNECT_VERSION = 2 时，jConnect 将使用 iso_1 作为缺省字符集。
- JCONNECT_VERSION = 3 和更高版本时，jConnect 将使用数据库的缺省字符集并会在客户端自动调整以执行任何必要的转换。
- 对于从 6.05 开始的 jConnect 版本，如果 jConnect 无法将用户数据成功转换为协商的字符集，则在服务器支持 Unicode 字符时，它会向服务器发送未经转换的 Unicode 字符，否则会抛出异常。

也可以使用 IsqlApp 应用程序的 -J *charset* 命令行选项指定字符集。

若要确定 Adaptive Server 上安装了哪些字符集，请在服务器上发出以下 SQL 查询：

```
select name from syscharsets
 go
```

对于 PureConverter 类，如果客户端 Java 虚拟机 (VM) 不支持指定的 CHARSET，那么连接将失败并引发 SQLException，指出必须将 CHARSET 设置为 Adaptive Server 和客户端都支持的字符集。

如果使用 TruncationConverter 类，则无论指定的 CHARSET 是否是 7 位 ASCII 码，都将进行字符截断。因此，如果您的应用程序需要处理非 ASCII 数据 （例如任何亚洲语言），则不应使用 TruncationConverter，因为这会导致数据损坏。

提高字符集转换性能

如果使用多字节字符集并需要提高驱动程序性能，可以使用 jConnect 示例提供的 SunIoConverter 类。有关详细信息，请参见第 136 页的“SunIoConverter 字符集转换”。

另外，如果您的应用程序仅处理 7 位 ASCII 数据，则可以使用 TruncationConverter 来提高性能。

支持的字符集

表 2-4 列出了 jConnect 支持的 Sybase 字符集。表中还列出了每个支持的字符集的对应的 JDK 字节转换程序。

虽然 jConnect 支持 UCS-2，但目前 Sybase 数据库或 Open Server 都不支持 UCS-2。

Adaptive Server 12.5 和更高版本支持一个 Unicode 版本 （称为 UTF-16 编码）。

表 2-4：支持的 Sybase 字符集

SybCharset 名称	JDK 字节转换程序
ascii_7	ASCII
big5	Big5
big5hk （针对 JDK 1.3 及更高版本）	Big5_HKSCS
cp037	Cp037
cp437	Cp437
cp500	Cp500
cp850	Cp850
cp852	Cp852
cp855	Cp855
cp857	Cp857
cp860	Cp860
cp863	Cp863
cp864	Cp864
cp866	Cp866
cp869	Cp869
cp874	Cp874
cp932	MS932
cp936	GBK
cp949	Cp949
cp950	Cp950
cp1250	Cp1250
cp1251	Cp1251
cp1252	Cp1252
cp1253	Cp1253
cp1254	Cp1254
cp1255	Cp1255
cp1256	Cp1256
cp1257	Cp1257
cp1258	Cp1258
deckanji	EUC_JP
eucgb	EUC_CN
eucjis	EUC_JP
eucksc	EUC_KR

SybCharset 名称	JDK 字节转换程序
gb18030	GB18030
ibm420	Cp420
ibm918	Cp918
iso_1	ISO8859_1
iso88592	ISO8859-2
is088595	ISO8859_5
iso88596	ISO8859_6
iso88597	ISO8859_7
iso88598	ISO8859_8
iso88599	ISO8859_9
iso15	ISO8859_15_FDIS
koi8	KOI8_R
mac	MacRoman
mac_cyr	MacCyrillic
mac_ee	MacCentralEurope
macgreek	MacGreek
macturk	MacTurkish
sjis	MS932
tis620	MS874
ucs2	Unicode
utf8	UTF8

处理不受支持的字符集

jConnect 不支持以下 Sybase 字符集，因为 JDK 字节转换程序与这些 Sybase 字符集都不类似：

- cp1047
- euccns
- greek8
- roman8
- roman9
- turkish8

可以对 TruncationConverter 类使用这些字符集，只要应用程序只使用这些字符的 7 位 ASCII 子集即可。

取代缺省字符集映射

使用 JAVA_CHARSET_MAPPING 连接属性可取代 Adaptive Server 的缺省字符集映射。

示例 1　将服务器字符集 cp949 映射到 ms949：

```
props.put("CHARSET", "cp949"); /* Server character set */
props.put("JAVA_CHARSET_MAPPING", "ms949"); /* Java character set mapping */
```

大多数 Adaptive Server 字符集与其所映射到的 Java 字符集同名。有关映射到 Java 字符集而使用不同名称的字符集，请参见第 37 页的表 2-4。

欧洲货币符号支持

jConnect 支持使用欧洲货币符号（或 “*euro*”），并支持欧洲货币符号与 UCS-2 Unicode 间的相互转换。

euro 已添加到以下 Sybase 字符集中：cp1250、 cp1251、 cp1252、 cp1253、 cp1254、 cp1255、 cp1256、 cp1257、 cp1258、 cp874、 iso885915 和 utf8。

若要使用 *euro* 符号：

- 使用 PureConvertor 或 CheckPureConverter 类，它是一个纯 Java 多字节字符集转换程序。有关详细信息，请参见第 35 页的 “jConnect 字符集转换程序”。
- 检验是否在服务器上安装了新的字符集。

  Adaptive Server 和 SQL Anywhere 都支持 *euro* 符号。
- 在客户端上选择合适的字符集。有关详细信息，请参见第 36 页的 “设置 CHARSET 连接属性”。

处理数据库

本节讨论与 jConnect 相关的数据库问题，包括以下主题：

- 实施高可用性故障切换支持
- 执行服务器到服务器的远程过程调用
- 使用 Adaptive Server 的宽表支持
- 访问数据库元数据

- 对结果集使用游标
- 使用包括 COMPUTE 子句的 Transact-SQL 查询
- 批处理更新支持
- 通过存储过程的结果集更新数据库
- 使用数据类型
- 仅数据锁定表中的可变长度行
- 大对象 (LOB) 支持
- 大对象定位符支持

实施高可用性故障切换支持

jConnect 支持 Adaptive Server 故障切换功能。

注释 高可用性系统中的 Sybase 故障切换与连接故障切换功能不同。如果希望同时使用这两个功能， Sybase 强烈建议您仔细 阅读本节内容。

概述

Sybase 故障切换允许配置两台 Adaptive Server 作为协同服务器。如果主协同服务器发生故障，该服务器的设备、数据库和连接可以由辅助协同服务器接管。

可以对称或非对称地配置高可用性系统：

- 非对称 配置包括两台 Adaptive Server，它们在物理上位于不同的计算机上，但彼此相连，以便当一台服务器出现故障时，可以由另一台服务器承担它的工作负荷。辅助 Adaptive Server 充当 “热备份”，它只有在出现故障切换时才工作。
- 对称 配置也包括两台在不同计算机上运行的 Adaptive Server。但当出现故障切换时，其中每一个 Adaptive Server 都可以充当另一个 Adaptive Server 的主协同服务器或辅助协同服务器。在此配置中，每一个 Adaptive Server 都具有完整的功能，都具有各自的系统设备、系统数据库、用户数据库和用户登录。

在上述两种设置中，两台计算机都被配置为双向访问，这样使两台计算机都可以看到并访问对方的磁盘。

可以在 jConnect 中启用故障切换，然后将客户端应用程序连接到进行过故障切换配置的 Adaptive Server。如果主服务器故障切换到辅助服务器，客户端应用程序也会自动切换到辅助服务器并重新建立网络连接。

注释 有关详细信息，请参见 Adaptive Server Enterprise 中针对 Adaptive Server 的 《在高可用性系统中使用 Sybase 故障切换》手册。

要求、依赖性和限制

- 必须配置两个 Adaptive Server 才能进行故障切换。
- 当客户端发生故障切换时，只保留在发生故障切换前提交给数据库的更改。
- 必须将 REQUEST_HA_SESSION jConnect 连接属性设置为 “true”（请参见第 9 页的 “连接属性”）。
- 发生故障切换时 jConnect 事件通知不起作用。请参见第 72 页的 “使用事件通知”。
- 请关闭所有不再使用的语句。 jConnect 会存储有关语句的信息以启用故障切换。未关闭的语句将导致内存泄漏。

在 jConnect 中实现故障切换

可使用以下两种方法在 jConnect 中实现故障切换支持：

- 使用两个连接属性 （REQUEST_HA_SESSION 和 SECONDARY_SERVER_HOSTPORT），并按以下方式设置：
  - 将 REQUEST_HA_SESSION 设置为 “true”。
  - 将 SECONDARY_SERVER_HOSTPORT 设置为辅助服务器监听的主机名和端口号。请参见第 9 页的 “连接属性”和 “SECONDARY_SERVER_HOSTPORT”连接属性。
- 使用 JNDI 连接到服务器。请参见“使用 JNDI 连接到服务器”。在 JNDI 所需的目录服务信息文件中加入一个主服务器条目和一个辅助服务器条目。主服务器条目有一个引用辅助服务器条目的属性 (HA OID)。

  使用 LDAP 作为 JNDI 的服务提供程序时，此 HA 属性可以有以下三种可能的形式：

a 相对区分名 (RDN) – 此形式假定与此属性值结合使用的搜索库（通常由 java.naming.provider.url 属性提供）足以标识辅助服务器。例如，假定主服务器位于 “hostname:4200”上，而辅助服务器位于 “hostname:4202”上：

```
dn:servername=haprimary, o=Sybase, c=US
1.3.6.1.4.1.897.4.2.5: TCP#1#hostname 4200
1.3.6.1.4.1.897.4.2.15: servername=hasecondary
objectclass:sybaseServer

dn:servername=hasecondary, o=Sybase, c=US
1.3.6.1.4.1.897.4.2.5: TCP#1#hostname 4202
objectclass:sybaseServer
```

b 区分名 (DN) – 此形式假定 HA 属性的值唯一标识辅助服务器，因此可能复制也可能不复制在搜索库中找到的值。例如：

```
dn:servername=haprimary, o=Sybase, c=US
1.3.6.1.4.1.897.4.2.5: TCP#1#hostname 4200
1.3.6.1.4.1.897.4.2.15: servername=hasecondary,
   o=Sybase, c=US ou=Accounting
objectclass:sybaseServer

dn:servername=hasecondary, o=Sybase, c=US, ou=Accounting
1.3.6.1.4.1.897.4.2.5: TCP#1#hostname 4202
objectclass:sybaseServer
```

请注意，`hasecondary` 位于树的其它分支上（请参见附加的 `ou=Accounting` 限定符）。

c *Full LDAP URL* – 此形式对搜索库没有任何假定。 HA 属性应是用于标识辅助服务器的完全限定 LDAP URL （它甚至可以指向不同的 LDAP 服务器）。例如：

```
dn:servername=hafailover, o=Sybase, c=US
1.3.6.1.4.1.897.4.2.5: TCP#1#hostname 4200
1.3.6.1.4.1.897.4.2.15: ldap://ldapserver:386/servername=secondary,
   o=Sybase, c=US ou=Accounting
objectclass:sybaseServer

dn:servername=secondary, o=Sybase, c=US, ou=Accounting
1.3.6.1.4.1.897.4.2.5: TCP#1#hostname 4202
objectclass:sybaseServer
```

d 在 JNDI 所需的目录服务信息文件中，将 REQUEST_HA_SESSION 连接属性设置为 “true”，以使每次建立连接时都启用故障切换会话。

使用 REQUEST_HA_SESSION 连接属性表明连接客户端希望与配置为用于故障切换的 Adaptive Server 开始一个故障切换会话。将此属性设置为 true 指示 jConnect 将尝试进行故障切换登录。如果没有设置此连接属性，即使正确配置了服务器，也不会启动故障切换会话。 REQUEST_HA_SESSION 的缺省值是 false。

像设置其它任何连接属性一样设置此连接属性。建立连接后将不能重置此属性。

如果希望请求故障切换会话时具有更大的灵活性，可以对客户端应用程序进行编码，使其在运行期设置 REQUEST_HA_SESSION。

以下示例演示为 LDAP 目录服务下的数据库服务器 “SYBASE11” 输入的连接信息，其中 “tahiti” 是主服务器，“moorea” 是辅助协同服务器：

```
dn:servername=SYBASE11,o=MyCompany,c=US
1.3.6.1.4.1.897.4.2.5:TCP#1#tahiti 3456
 1.3.6.1.4.1.897.4.2.10:REPEAT_READ=false&PACKETSIZE=1024
1.3.6.1.4.1.897.4.2.10:CONNECTION_FAILOVER=false
 1.3.6.1.4.1.897.4.2.11:pubs2
 1.3.6.1.4.1.897.4.2.9:Tds
1.3.6.1.4.1.897.4.2.15:servername=SECONDARY
1.3.6.1.4.1.897.4.2.10:REQUEST_HA_SESSION=true


dn:servername=SECONDARY, o=MyCompany, c=US
1.3.6.1.4.1.897.4.2.5:TCP#1#moorea 6000
```

- 使用 JNDI 和 LDAP 请求连接：
  - jConnect 使用 LDAP 服务器的目录确定主服务器和辅助服务器的名称和位置：

```
/* get the connection */
 Connection con = DriverManager.getConnection
   ("jdbc:sybase:jndi:ldap://ldap_server1:389" +
    "/servername=Sybase11,o=MyCompany,c=US",props);
```

或者

  - 指定搜索库：

```
props.put(Context.PROVIDER_URL,
   "ldap://ldap_server1:389/ o=MyCompany, c=US");

Connection con=DriverManager.getConnection
   ("jdbc:sybase:jndi:servername=Sybase11", props);
```

登录到主服务器

如果 Adaptive Server 没有配置为用于故障切换或者不能批准故障切换会话，客户端将无法登录并会显示以下警告：

```
'The server denied your request to use the high-
availability feature.

Please reconfigure your database, or do not request a
high-availability session.'
```

故障切换到辅助服务器

发生故障切换时将抛出 SQL 例外 JZ0F2：

```
'Sybase high-availability failover has occurred.The
current transaction is aborted, but the connection is
still usable.Retry your transaction.'
```

然后客户端使用 JNDI 自动重新连接到辅助数据库。

注意：

- 客户端连接到的数据库的标识和任何提交的事务都会得到保留。
- 部分读取的结果集、游标和存储过程调用将丢失。
- 发生故障切换时，应用程序可能需要重新启动过程或返回到上一个完成的事务或活动。

故障恢复到主服务器

有时客户端会从辅助服务器故障恢复到主服务器。何时发生故障恢复是由系统管理员决定的，他在辅助服务器上发出 sp_failback 之后，客户端即可期望在主服务器上发生同样的行为和结果，如第 44 页的 “故障切换到辅助服务器”所述。

执行服务器到服务器的远程过程调用

在一台服务器上运行的 Transact-SQL 语言命令和存储过程可以执行位于另一台服务器上的存储过程。应用程序已连接到的服务器登录到远程服务器，并执行服务器到服务器的远程过程调用。

应用程序可以指定一个 “通用”口令供服务器间通信使用，即用于所有服务器间连接的口令。连接打开后，服务器便可使用此口令登录到任何远程服务器。缺省情况下， jConnect 使用当前连接的口令作为服务器间通信的缺省口令。

但如果同一用户在两台服务器上的口令不同，且该用户打算执行服务器到服务器的远程过程调用，则应用程序必须为每个要使用的服务器显示定义口令。

jConnect 包括一个属性，可以用来设置通用的 “远程”口令或在不同的服务器上设置不同的口令。 jConnect 允许使用 SybDriver 类中的 setRemotePassword 方法设置并配置此属性：

```
Properties connectionProps = new Properties();

public final void setRemotePassword(String serverName,
    String password, Properties connectionProps)
```

若要使用此方法，应用程序必须先导入 SybDriver 类，然后再调用此方法：

```
import com.sybase.jdbcx.SybDriver;
SybDriver sybDriver = (SybDriver)
    Class.forName("com.sybase.jdbc4.jdbc.SybDriver").n
ewInstance();
sybDriver.setRemotePassword
    (serverName, password, connectionProps);
```

注释 若要为不同的服务器设置不同的远程口令，请为每台服务器重复上述调用过程。

此调用将给定的 “服务器名-口令”对添加到给定 Properties 对象中，该对象可以由应用程序在 DriverManager.getConnection (*server_url*, *props*) 中传递给 DriverManager。

如果 serverName 为 null，则将 password 设置为通用口令，供后续到所有服务器 （不包括由前面的 setRemotePassword 调用明确指定的服务器）的连接使用。

如果应用程序设置了 REMOTEPWD 属性， jConnect 将不再设置缺省的通用口令。

使用 Adaptive Server 的宽表支持

Adaptive Server 提供的限制和参数要大于以前版本的数据库服务器。例如：

- 表可以包含 1,024 列。
- Varchar 和 varbinary 列可以包含超过 255 个字节的数据。
- 在调用存储过程或者作为 PreparedStatement 的参数进行调用时，最多可以发送和检索 2048 个参数。

- 在连接到 Adaptive Server 15.7 ESD #1 及更高版本时，最多可以发送和检索 32767 个 PreparedStatement 参数。

若要确保 jConnect 从数据库请求宽表支持，JCONNECT_VERSION 的缺省设置必须为 6 或更高版本。

注释 如果将 JCONNECT_VERSION 设置为低于 6，jConnect 仍可继续使用 Adaptive Server 12.5 版和更高版本。但如果尝试从需要宽表支持才能完全检索数据的表中选择数据，将可能遇到意外的错误或数据截断情形。

从不支持宽表的 Sybase 服务器访问数据时，也可以将 JCONNECT_VERSION 设置为 6 或更高版本。在这种情况下，服务器只是忽略宽表支持请求。

除了大量列和参数之外，宽表支持还提供扩展结果集元数据。例如，在 jConnect 6.0 之前的版本中， ResultSetMetaData 方法 getCatalogName、getSchemaName 和 getTableName 都返回 “Not Implemented” SQLException，因为服务器没有提供元数据。启用宽表支持后，服务器将发送回此信息，上述三个方法将返回有用的信息。

访问数据库元数据

为了支持 JDBC DatabaseMetaData 方法， Sybase 提供了一组存储过程，jConnect 可以调用这些存储过程以获取数据库的元数据。为使 JDBC 元数据方法能正常工作，必须在服务器上安装这些存储过程。

如果 Sybase 服务器上尚未安装用于提供元数据的存储过程，可以使用随 jConnect 提供的存储过程脚本进行安装：

- *sql_server.sql* 在 Adaptive Server 12.0 版之前的数据库中安装存储过程。
- *sql_server12.sql* 在 Adaptive Server 12.0.x 版的数据库中安装存储过程。
- *sql_server12.5.sql* 在 Adaptive Server 12.5.x 版的数据库中安装存储过程。
- *sql_server15.0.sql* 为 Adaptive Server 15.0 到 15.5 版安装存储过程。
- *sql_server15.7.sql* 为 Adaptive Server 15.7.2 或更高版本安装存储过程。
- *sql_server15.7.0.2.sql* 为 Adaptive Server 15.7 ESD #2 或更高版本安装存储过程。

- *sql_asa.sql* – 在 SQL Anywhere 9.x 数据库中安装存储过程
- *sql_asa10.sql* – 在 SQL Anywhere 10.x 数据库中安装存储过程
- *sql_asa11.sql* – 在 SQL Anywhere 11.x 数据库中安装存储过程。
- *sql_asa12.sql* – 在 SQL Anywhere 12.x 数据库中安装存储过程。

注释 这些脚本的最新版本与所有版本的 jConnect 都兼容。

有关安装存储过程的详细说明，请参见 Sybase 《*jConnect for JDBC* 安装指南》和 Sybase 《*jConnect for JDBC* 发行公告》。

此外，若要使用元数据方法，在建立连接时必须将 USE_METADATA 连接属性设置为 “true”（缺省值）。

不能获取有关数据库中临时表的元数据。

注释 DatabaseMetaData.getPrimaryKeys 方法可查找在表定义 (CREATE TABLE) 中声明或使用 alter table (ALTER TABLE ADD CONSTRAINT) 声明的主键。它不查找使用 sp_primarykey 定义的键。

对结果集使用游标

jConnect 实现许多 JDBC 2.0 游标和更新方法。这些方法使得使用游标以及根据结果集中的值更新表的数据行变得更加容易。

在 JDBC 2.0 中， ResultSet 的主要特点在于其类型和并发性。类型和并发值是 java.sql.ResultSet 接口的一部分，由该接口的 javadoc 描述。

表 2-5 描述了 jConnect 中提供的 java.sql.ResultSet 的特点。请求时，如果服务器是 Adaptive Server 15.0 或更高版本，则 jConnect 会打开服务器端可滚动游标。

表 2-5：jConnect 中提供的 java.sql.ResultSet 选项

	类型		
并发	**TYPE_FORWARD_ONLY**	**TYPE_SCROLL_INSENSITIVE**	**TYPE_SCROLL_SENSITIVE**
CONCUR_READ_ONLY	支持	支持	不可用
CONCUR_UPDATABLE	支持	不可用	不可用

本节包括以下主题：

创建游标

使用 jConnect 创建游标有两种方法：

- SybStatement.setCursorName

  可使用 SybStatement.setCursorName 显式指定游标名称。SybStatement.setCursorName 的签名是：

  ```
  void setCursorName(String name) throws SQLException;
  ```

- SybStatement.setFetchSize

  可使用 SybStatement.setFetchSize 创建游标并指定每次读取操作从数据库返回的行数。 SybStatement.setFetchSize 的签名是：

  ```
  void setFetchSize(int rows) throws SQLException;
  ```

  使用 setFetchSize 创建游标时， jConnect 驱动程序会为游标命名。若要获取游标名称，请使用 ResultSet.getCursorName。

另一种创建游标的方法是在连接上使用以下 JDBC 方法指定希望语句返回的 ResultSet 的类型：

```
Statement createStatement(int resultSetType, int
resultSetConcurrency)throws SQL Exception
```

类型和并发值与表 2-5 中列出的、在 ResultSet 接口中找到的类型和并发值相对应。如果请求的是不受支持的 ResultSet，将在连接上链接一个 SQL 警告。执行返回的 Statement 时，您将收到与所请求的类型最接近的 ResultSet 类型。有关此方法的行为的详细信息，请参见 JDBC 规范。

如果不使用 createStatement，则 ResultSet 的缺省类型是：

- 如果只调用 Statement.executeQuery，则返回的 ResultSet 是类型和并发值分别为 TYPE_FORWARD_ONLY 和 CONCUR_READ_ONLY 的 SybResultSet。
- 如果调用 setCursorName，则从 executeQuery 返回的 ResultSet 是类型和并发值分别为 TYPE_FORWARD_ONLY 和 CONCUR_UPDATABLE 的 SybCursorResultSet。

- 如果调用 setFetchSize，则从 executeQuery 返回的 ResultSet 是类型和并发值分别为 TYPE_FORWARD_ONLY 和 CONCUR_READ_ONLY 的 SybCursorResultSet。

若要检验 ResultSet 对象的类型是否是您所需要的类型，请使用以下两种 ResultSet 方法：

```
int getConcurrency() throws SQLException;
int getType() throws SQLException;
```

❖ **创建和使用游标**

1 使用 Statement.setCursorName 或 SybStatement.setFetchSize 创建游标。

2 调用 Statement.executeQuery 为语句打开游标并返回游标结果集。

3 调用 ResultSet.next 读取行并在结果集中定位游标。

下面的示例分别使用上述两种方法创建游标并返回结果集。它还使用 ResultSet.getCursorName 获取通过 SybStatement.setFetchSize 创建的游标的名称。

```
// With conn as a Connection object, create a
// Statement object and assign it a cursor using
// Statement.setCursorName().
Statement stmt = conn.createStatement();
stmt.setCursorName("author_cursor");

// Use the statement to execute a query and return
// a cursor result set.
ResultSet rs = stmt.executeQuery("SELECT au_id,
      au_lname, au_fname FROM authors
      WHERE city = 'Oakland'");
while(rs.next())
{
...
}

// Create a second statement object and use
// SybStatement.setFetchSize()to create a cursor
// that returns 10 rows at a time.
SybStatement syb_stmt = conn.createStatement();
syb_stmt.setFetchSize(10);

// Use the syb_stmt to execute a query and return
// a cursor result set.
SybCursorResultSet rs2 =
      (SybCursorResultSet)syb_stmt.executeQuery
      ("SELECT au_id, au_lname, au_fname FROM
```

```
authors
       WHERE city = 'Pinole'");
while(rs2.next())
{
...
}

// Get the name of the cursor created through the
// setFetchSize() method.
String cursor_name = rs2.getCursorName();
 ...
// For jConnect 6.0, create a third statement
// object using the new method on Connection,
// and obtain a SCROLL_INSENSITIVE ResultSet.
// Note:you no longer have to downcast the
// Statement or the ResultSet.
Statement stmt = conn.createStatement(
                 ResultSet.TYPE_SCROLL_INSENSITIVE,
                 ResultSet.CONCUR_READ_ONLY);
ResultSet rs3 = stmt.executeQuery
   ("SELECT ...[whatever]");
// Execute any of the JDBC 2.0 methods that
// are valid for read only ResultSets.
rs3.next();
rs3.previous();
rs3.relative(3);
rs3.afterLast();
...
```

使用 JDBC 1.x 方法进行定位型更新和删除

下面的示例演示如何使用 JDBC 1.x 中的方法进行定位型更新。该示例创建两个 Statement 对象，一个用于选择游标结果集中的行，另一个用于通过结果集中的行更新数据库。

```
// Create two statement objects and create a cursor
// for the result set returned by the first
// statement, stmt1.Use stmt1 to execute a query
// and return a cursor result set.
Statement stmt1 = conn.createStatement();
Statement stmt2 = conn.createStatement();
stmt1.setCursorName("author_cursor");
ResultSet rs = stmt1.executeQuery("SELECT
    au_id,au_lname, au_fname
    FROM authors WHERE city = 'Oakland'
```

```
        FOR UPDATE OF au_lname");

    // Get the name of the cursor created for stmt1 so
    // that it can be used with stmt2.
    String cursor = rs.getCursorName();

    // Use stmt2 to update the database from the
    // result set returned by stmt1.
    String last_name = new String("Smith");
    while(rs.next())
    {
        if (rs.getString(1).equals("274-80-9391"))
         {
           stmt2.executeUpdate("UPDATE authors "+
           "SET au_lname = "+last_name +
           "WHERE CURRENT OF " + cursor);
        }
    }
```

在结果集中删除

下面的示例使用从上述代码中获得的 Statement 对象 *stmt2* 执行定位型删除：

```
stmt2.executeUpdate("DELETE FROM authors
          WHERE CURRENT OF " + cursor);
```

使用 JDBC 2.0 方法进行定位型更新和删除

本节介绍一些 JDBC 2.0 方法，可用于更新当前游标行中的列以及通过结果集中的当前游标行更新数据库。每个方法后面都有一个示例。

在结果集中更新列

JDBC 2.0 提供了多个方法，用于在客户端更新内存中的结果集的列值。然后可使用更新的值对基础数据库执行更新、插入或删除操作。所有这些方法都在 SybCursorResultSet 类中实现。

下面是 jConnect 中提供的一些 JDBC 2.0 更新方法：

```
void updateAsciiStream(String columnName, java.io.InputStream x, int length)
   throws SQLException;
void updateBoolean(int columnIndex, boolean x) throws SQLException;
void updateFloat(int columnIndex, float x) throws SQLException;
void updateInt(String columnName, int x) throws SQLException;
void updateInt(int columnIndex, int x) throws SQLException;
```

```
void updateObject(String columnName, Object x) throws SQLException;
```

用于通过结果集更新数据库的方法

JDBC 2.0 指定了两种方法，用于根据结果集中的当前值更新或删除数据库中的行。这些方法在形式上比 JDBC 1.x 中的 Statement.executeUpdate 简单，并且不需要使用游标名称。它们在 SybCursorResultSet 中实现：

```
void updateRow() throws SQLException;
void deleteRow() throws SQLException;
```

注释 结果集的并发必须是 CONCUR_UPDATABLE。否则上述方法将引发例外。对于 insertRow，必须指定所有要求非空条目的表列。DatabaseMetaData 提供的方法决定这些更改何时可见。

示例

下面的示例将创建一个用于返回游标结果集的 Statement 对象。对于结果集中的每一行，列值在内存中更新，然后数据库通过该行的新列值得到更新。

```
// Create a Statement object and set fetch size to
// 25. This creates a cursor for the Statement
// object Use the statement to return a cursor
// result set.
SybStatement syb_stmt =
(SybStatement)conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_UPDATABLE);
syb_stmt.setFetchSize(25);
SybCursorResultSet syb_rs =
(SybCursorResultSet)syb_stmt.executeQuery(
    "SELECT * from T1 WHERE ...")

// Update each row in the result set according to
// code in the following while loop. jConnect
// fetches 25 rows at a time, until fewer than 25
// rows are left.Its last fetch takes any
// remaining rows.
while(syb_rs.next())
{
  // Update columns 2 and 3 of each row, where
// column 2 is a varchar in the database and
// column 3 is an integer.
  syb_rs.updateString(2, "xyz");
syb_rs.updateInt(3,100);
//Now, update the row in the database.
  syb_rs.updateRow();
```

```
}
// Create a Statement object using the
// JDBC 2.0 method implemented in jConnect 6.0
Statement stmt = conn.createStatement
(ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_UPDATABLE);
// In jConnect 6.0, downcasting to SybCursorResultSet is not
// necessary.Update each row in the ResultSet in the same
// manner as above
while(rs.next())
{
rs.updateString(2, "xyz");
rs.updateInt(3,100);
  rs.updateRow();
// Use the Statement to return an updatable ResultSet
ResultSet rs = stmt.executeQuery("SELECT * FROM T1 WHERE...");
}
```

从 ResultSet 中删除行

若要从游标结果集中删除某一行，可按如下方式使用 SybCursorResultSet.deleteRow：

```
while(syb_rs.next())
 {
     int col3 = getInt(3);
     if (col3 >100)
     {
     syb_rs.deleteRow();
     }
 }
```

在 ResultSet 中插入行

下面的示例说明如何使用 JDBC 2.0 API 执行插入操作。不需要使用 SybCursorResultSet。

```
// prepare to insert
rs.moveToInsertRow();
// populate new row with column values
rs.updateString(1, "New entry for col 1");
rs.updateInt(2, 42);
// insert new row into db
rs.insertRow();
// return to current row in result set
rs.moveToCurrentRow();
```

在游标关闭时释放锁

Adaptive Server 15.7 扩展了 declare cursor 语法，使其包括 release_locks_on_close 选项，用以在游标关闭时在隔离级别 2 和 3 释放共享游标锁。 jConnect 对 release-lock-on-close 的语义提供相应支持。

要在 jConnect 连接上应用，请将 RELEASE_LOCKS_ON_CURSOR_CLOSE 连接属性设置为 true。 RELEASE_LOCKS_ON_CURSOR_CLOSE 的缺省值为 false。

此设置只有在连接到支持 release_locks_on_close 的服务器时才会生效。

有关 release_locks_on_close 的信息，请参见 Adaptive Server Enterprise 《参考手册：命令》。

select for update 支持

Adaptive Server 15.7 支持 select for update，它可以为同一事务内的后续更新锁定行，并支持可更新游标的排它锁。请参见 Adaptive Server Enterprise 《*Transact-SQL* 用户指南》中的第 2 章 “查询：从表中选择数据”。

当 for update 子句添加到 select 语句以及客户端内打开的任何可更新游标中后，此功能便可自动供客户端使用。

对 PreparedStatement 对象使用游标

PreparedStatement 对象在创建后可以多次使用，每次使用时可为其输入参数指定相同或者不同的值。如果为 PreparedStatement 对象使用游标，则每次使用完游标后必须将其关闭，下次使用时再将其重新打开。关闭游标的结果集 (ResultSet.close) 时也会关闭该游标。执行游标的预准备语句 (PreparedStatement.executeQuery) 时会打开该游标。

下面的示例演示如何创建 PreparedStatement 对象，如何为其指定游标，以及如何执行两次 PreparedStatement 对象 （关闭然后重新打开游标）。

```
// Create a prepared statement object with a
// parameterized query.
PreparedStatement prep_stmt =
conn.prepareStatement(
"SELECT au_id, au_lname, au_fname "+
"FROM authors WHERE city = ?"+
"FOR UPDATE OF au_lname");

//Create a cursor for the statement.
prep_stmt.setCursorName("author_cursor");

// Assign the parameter in the query a value.
```

```
// Execute the prepared statement to return a
// result set.
prep_stmt.setString(1, "Oakland");
ResultSet rs = prep_stmt.executeQuery();

//Do some processing on the result set.
while(rs.next())
{
    ...
}

// Close the result, which also closes the cursor.
rs.close();

// Execute the prepared statement again with a new
// parameter value.
prep_stmt.setString(1,"San Francisco");
rs = prep_stmt.executeQuery();
// reopens cursor
```

在 jConnect 中使用 TYPE_SCROLL_INSENSITIVE 结果集

jConnect 仅支持 TYPE_SCROLL_INSENSITIVE 结果集。

jConnect 使用 Tabular Data Stream (TDS) （即 Sybase 专有协议）与 Sybase 数据库服务器进行通信。 Adaptive Server 15.0 或更高版本支持 TDS 可滚动游标。对于不支持 TDS 可滚动游标的服务器，在每次调用 ResultSet.next 时， jConnect 都会在客户端按需高速缓存行数据。但到达结果集的末尾时，整个结果集将存储到客户端内存中。因为这可能导致性能降低，所以 Sybase 建议您仅在使用 Adaptive Server 15.0 或结果集相当小时才使用 TYPE_SCROLL_INSENSITIVE 结果集。

注释 在 jConnect 中使用 TYPE_SCROLL_INSENSITIVE ResultSets 时，如果服务器不支持 TDS 可滚动游标，则只能在读取完 ResultSet 的最后一行后才能调用 isLast 方法。在未达到最后一行时调用 isLast 会导致抛出 UnimplementedOperationException。

---

jConnect 在 *sample2* 目录中提供了 ExtendResultSet；此示例使用 JDBC 1.0 接口提供了一个受限制的 TYPE_SCROLL_INSENSITIVE ResultSet。

此实现使用标准 JDBC 1.0 方法生成无滚动功能的只读结果集，即基础数据的一个静态视图，它不会即时反映在结果集为打开状态时所进行的更改。 ExtendedResultSet 在客户端高速缓存所有 ResultSet 行。对较大的结果集使用此类时应谨慎。

sample.ScrollableResultSet 接口：

- 是 JDBC 1.0 java.sql.ResultSet 的扩展。
- 定义了与 JDBC 2.0 java.sql.ResultSet 具有相同签名的其它方法。
- *只*包含部分 JDBC 2.0 方法。未包含其中用于修改 ResultSet 的方法。

来自 JDBC 2.0 API 的方法有：

```
boolean previous() throws SQLException;
boolean absolute(int row) throws SQLException;
boolean relative(int rows) throws SQLException;
boolean first() throws SQLException;
boolean last() throws SQLException;
void beforeFirst() throws SQLException;
void afterLast() throws SQLException;
boolean isFirst() throws SQLException;
```

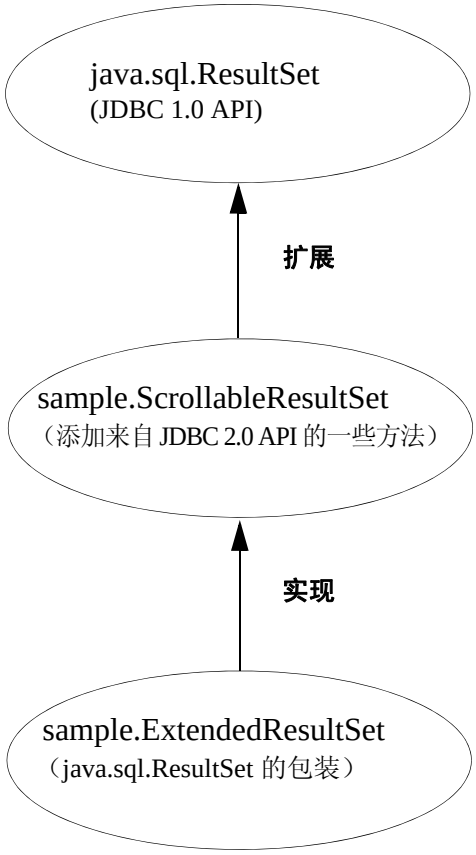
```
boolean isLast() throws SQLException;
boolean isBeforeFirst() throws SQLException;
boolean isAfterLast() throws SQLException;
int getFetchSize() throws SQLException;
void setFetchSize(int rows) throws SQLException;
int getFetchDirection() throws SQLException;
void setFetchDirection(int direction) throws
SQLException;
int getType() throws SQLException;
int getConcurrency() throws SQLException;
int getRow() throws SQLException;
```

若要使用示例类，请使用任意 JDBC 1.0 java.sql.ResultSet 创建一个 ExtendedResultSet。以下为相关的代码段（假定为 Java 1.1 环境）：

```
// import the sample files
import sample.*;
//import the JDBC 1.0 classes
import java.sql.*;
// connect to some db using some driver;
// create a statement and a query;
// Get a reference to a JDBC 1.0 ResultSet
ResultSet rs = stmt.executeQuery(_query);
// Create a ScrollableResultSet with it
ScrollableResultSet srs = new ExtendedResultSet(rs);
// invoke methods from the JDBC 2.0 API
srs.beforeFirst();
// or invoke methods from the JDBC 1.0 API
if (srs.next())
  String column1 = srs.getString(1);
```

图 2-1 是一个类框图，显示了示例类和 JDBC API 之间的关系。

图 2-1：类框图



有关详细信息，请参见 JDBC 2.0 API，网址为：
http://www.oracle.com/technetwork/java/javase/jdbc。

使用包括 COMPUTE 子句的 Transact-SQL 查询

jConnect for JDBC 支持包括 COMPUTE 子句的 Transact-SQL 查询。COMPUTE 子句允许使用一个 select 语句显示明细和摘要结果。摘要行显示在特定组的明细行的下面。例如：

```
select type, price, advance
   from titles
   order by type
   compute sum(price), sum(advance) by type
```

```
type          price      advance
------------  ---------  ----------
UNDECIDED     NULL       NULL

Compute Result:
----------------------- -----------------------
NULL                    NULL

type          price      advance
------------  ---------  ----------
business      2.99       10,125.00
business      11.95      5,000.00

business      19.99      5,000.00
business      19.99      5,000.00

Compute Result:
----------------------- -----------------------
54.92                   25,125.00

...
...

(24 rows affected)
```

在 jConnect 执行包括 COMPUTE 子句的 select 语句时，jConnect 会向客户端返回多个结果集，结果集的数量取决于可用的唯一分组的数量。每个组包含一个用于明细行的结果集和一个用于摘要的结果集。客户端必须处理所有结果集才能完全处理返回的行，否则返回的第一个结果集中将只包括第一个数据组的明细行。

有关 COMPUTE 子句的详细信息，请参见 Adaptive Server Enterprise《*Transact-SQL* 用户指南》。有关处理多个结果集的详细信息，请参见 Oracle Technology Network for Java Web 站点上提供的 JDBC API 文档。

批处理更新支持

批处理更新允许一个 Statement 对象向基础数据库提交多个语句，这些语句作为一个单元（一批）一起进行处理。添加到批处理的任何语句必须仅返回更新计数，不得返回 ResultSet。

有关对 Statement、 PreparedStatement 和 CallableStatement 使用批处理更新的示例，请参见 *sample2* 子目录中的 *BatchUpdates.java*。

jConnect 还支持动态 PreparedStatements 的批处理。

实现说明

jConnect 按照 JDBC 2.0 API 中指定的方式实现批处理更新，但有下列例外：

- EXECUTE_BATCH_PAST_ERRORS 连接属性控制故障在批处理执行中的处理方式。

  缺省情况下， EXECUTE_BATCH_PAST_ERRORS 设置为 false，jConnect 会在出现第一个故障后停止处理。BatchUpdateException.getUpdateCounts 返回长度为 M < N 的 int[] 数组，表示批处理中的前 M 个语句成功，第 M+1 个语句失败，第 M+2..N 个语句没有执行。此处的 “N”等于批处理中的语句总数。

  如果将 EXECUTE_BATCH_PAST_ERRORS 设置为 true，则 jConnect 会在出现非致命故障时继续处理。BatchUpdateException.getUpdateCounts 返回长度为 N 的 int[] 数组，其中 “N”等于批处理中的语句总数。检查各项更新计数便可确定每个语句的执行状态。

- 若要以批处理 （非链式）模式调用存储过程，必须也以非链式模式创建存储过程。有关详细信息，请参见第 133 页的 “在非链式事务模式中执行存储过程”。

- 如果 Adaptive Server 在批处理执行过程中遇到致命错误，则 BatchUpdateException.getUpdateCounts 仅返回长度为零的 int[ ]。发生致命错误时整个事务将回退，因此成功操作的行数为零。

- 不支持批处理更新的数据库中的批处理更新：即使数据库不支持批处理更新， jConnect 仍可在 executeUpdate 循环中执行批处理更新。这样无论指向哪个数据库，都可以使用相同的批处理代码。

有关批处理更新的详细信息，请参见JDBC API documentation (http://www.oracle.com/technetwork/java/index.html)。

通过存储过程的结果集更新数据库

jConnect 提供 update 方法和 delete 方法，用于在由存储过程返回的结果集中获取游标。然后可使用游标的位置更新或删除提供结果集的基础表中的行。这些方法位于 SybCursorResultSet 中：

```
void updateRow(String tableName) throws SQLException;
void deleteRow(String tableName) throws SQLException;
```

tableName 参数标识提供结果集的数据库表。

若要获取存储过程返回的结果集中的游标，需要先使用 SybCallableStatement.setCursorName 或 SybCallableStatement.setFetchSize，然后再执行包含该过程的可调用语句。下面的示例演示如何在存储过程的结果集中创建游标，更新结果集中的值，然后使用 SybCursorResultSet.update 方法更新基础表：

```
// Create a CallableStatement object for executing the stored
// procedure.
CallableStatement sproc_stmt =
   conn.prepareCall("{call update_titles}",
     ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_UPDATABLE);

// Set the number of rows to be returned from the database with
// each fetch.This creates a cursor on the result set.
(SybCallableStatement)sproc_stmt.setFetchSize(10);

//Execute the stored procedure and get a result set from it.
SybCursorResultSet sproc_result = (SybCursorResultSet)
   sproc_stmt.executeQuery();

// Move through the result set row by row, updating values in the
// cursor' s current row and updating the underlying titles table
// with the modified row values.
while(sproc_result.next())
{
   sproc_result.updateString(...);
   sproc_result.updateInt(...);
   ...
   sproc_result.updateRow(titles);
}
```

使用数据类型

本节介绍如何使用 numeric、 image、 text、 date、 time 和 char 数据。

发送数值数据

SybPreparedStatement 扩展支持 Adaptive Server 处理 NUMERIC 数据类型的方式，可为该数据类型指定精度 （总位数）和标度 （小数点后的位数）。

Java 中与此对应的数据类型 （java.math.BigDecimal）稍有不同，当 jConnect 应用程序使用 setBigDecimal 方法控制输入/输出参数的值时，这些差异会引发问题。具体地说，有时参数 （无论是存储过程参数还是列）的精度和标度必须与对应的 SQL 对象的精度和标度完全一致。

SybPreparedStatement 扩展与以下方法结合使用来加强 jConnect 应用程序对 setBigDecimal 的控制：

```
public void setBigDecimal (int parameterIndex, BigDecimal X, int scale,
  int precision) throws SQLException
```

有关详细信息，请参见 jConnect 安装目录下 */sample2* 子目录中的 *SybPrepExtension.java* 示例。

更新数据库中的图像数据

jConnect 的 TextPointer 类包含 sendData 方法，用于更新 Adaptive Server 或 SQL Anywhere 数据库中的 image 列。在 jConnect 的早期版本中，必须在 java.sql.PreparedStatement 中使用 setBinaryStream 方法发送图像数据。在当前版本中， TextPointer.sendData 方法使用 java.io.InputStream 将图像数据发送到 Adaptive Server 数据库中，并且极大地提高了性能。

警告！ 使用 TextPointer 类的 sendData() 方法可能会影响应用程序，因为 TextPointer 不是标准的 JDBC 格式。

Sybase 建议使用标准 JDBC 格式 PreparedStatement.setBinaryStream(int paramIndex, InputStream image) 或 LOB 定位符支持来发送图像数据。但是，在处理大图像数据时， setBinaryStream() 消耗的过程高速缓存中的内存可能要远远多于 TextPointer 类。

在实现 TextPointer 类的替换之前， Sybase 将继续支持它。

若要获取 TextPointer 类的实例，可以在 SybResultSet 中使用两种 getTextPtr 方法之一：

```
public TextPointer getTextPtr(String columnName)
public TextPointer getTextPtr(int columnIndex)
```

TextPointer 类中的公共方法

com.sybase.jdbcx 软件包提供 TextPointer 类。其公共方法接口为：

```
public void sendData(InputStream is, boolean log)
  throws SQLException
public void sendData(InputStream is, int length,
  boolean log) throws SQLException
public void sendData(InputStream is, int offset,
  int length, boolean log) throws SQLException
public void sendData(byte[] byteInput, int offset,
  int length, boolean log) throws SQLEXception
```

其中：

- sendData(InputStream *is*, boolean *log)* 用指定的输入流中的数据更新 image 列。
- sendData(InputStream *is*, int *length*, boolean *log)* 用指定的输入流中的数据更新 image 列，其中 *length* 是发送的字节数。
- sendData(InputStream *is*, int *offset*, int *length*, boolean *log)* 用指定的输入流中的数据更新 image 列，从 *offset* 参数中给定的字节偏移处开始传送，直到传送完 *length* 参数中指定的字节数。
- sendData(byte[ ] *byteInput*, int *offset*, int *length*, boolean *log)* 用 *byteInput* 参数指定的字节数组中所包含的图像数据更新列。更新从 *offset* 参数中给定的字节偏移处开始，一直持续到读取完 *length* 参数中指定的字节数为止。
- *log* 是各种方法的参数，用于指定 image 数据是否要完全记录到数据库的事务日志中。如果 *log* 参数设置为 “true”，则整个二进制图像都会写入事务日志。如果 *log* 参数设置为 “false”，则将记录更新操作，但图像本身并不写入日志。

❖ **使用 *TextPointer.sendData* 更新 *image* 列**

使用图像数据更新列：

1　为要更新的行和列获取一个 TextPointer 对象。

2　使用 TextPointer.sendData 执行更新操作。

以下两节通过示例说明了这些更新步骤。该示例发送 *Anne_Ringer.gif* 文件中的 image 数据，用以更新 pubs2 数据库中 au_pix 表的 pic 列。更新是针对 author ID 为 899-46-2035 的行进行的。

获取 TextPointer 对象

text 和 image 列包含 timestamp 和页位置信息，它们与列的文本和图像数据分开存放。在从 text 或 image 列中选取数据时，此额外信息作为结果集的一部分 “隐含”起来。

用于更新 image 列的 TextPointer 对象需要此隐含信息，但不需要列数据的图像部分。为了获取此信息，需将这一列选取到 ResultSet 对象中，然后使用 SybResultSet.getTextPtr 提取文本指针信息，忽略图像数据，并创建 TextPointer 对象。请参见以下示例代码。

当一列中包含大量的图像数据时，为一行或多行选取列并等待获取所有数据很可能效率很低，因为不需要使用这些数据。为了缩短这一过程，请使用 set textsize 命令最小化数据包中返回的数据量。为达到这一目的，下面的代码示例在获取 TextPointer 对象时使用了 set textsize。

```
/*
 * Define a string for selecting pic column data for author ID
 * 899-46-2035.
 */
String getColumnData = "select pic from au_pix where au_id = '899-46-2035'";

/*
 * Use set textsize to return only a single byte of column data
 * to a Statement object.The packet with the column data will
 * contain the "hidden" information necessary for creating a
 * TextPointer object.
 */
Statement stmt= connection.createStatement();
stmt.executeUpdate("set textsize 1");

/*
 * Select the column data into a ResultSet object--cast the
 * ResultSet to SybResultSet because the getTextPtr method is
 * in SybResultSet, which extends ResultSet.
 */
SybResultSet rs = (SybResultSet)stmt.executeQuery(getColumnData);

/*
 * Position the result set cursor on the returned column data
 * and create the desired TextPointer object.
 */
rs.next();
TextPointer tp = rs.getTextPtr("pic");

/*
 * Now, assuming we are only updating one row, and won’ t need
 * the minimum textsize set for the next return from the server,
 * we reset textsize to its default value.
 */
stmt.executeUpdate("set textsize 0");
```

使用 *TextPointer.sendData* 进行更新

以下代码利用了上一节中的 TextPointer 对象，使用 *Anne_Ringer.gif* 文件中的图像数据来更新 pic 列。

```
/*
 *First, define an input stream for the file.
 */
FileInputStream in = new FileInputStream("Anne_Ringer.gif");
```

```
/*
* Prepare to send the input stream without logging the image data
* in the transaction log.
*/
boolean log = false;

/*
* Send the image data in Anne_Ringer.gif to update the pic
* column for author ID 899-46-2035.
*/
tp.sendData(in, log);
```

有关详细信息，请参见 jConnect 安装目录下 *sample2* 子目录中的 *TextPointers.java* 示例。

使用 *text* 数据

在早期版本中， jConnect 使用 TextPointer 类的 sendData 方法更新 Adaptive Server 或 SQL Anywhere 数据库中的 text 列。

Java 已不再支持 TextPointer 类，即在 Java 的后续版本中不再推荐使用，并且可能会取消该类。

如果使用的数据服务器是 Adaptive Server 或 SQL Anywhere，请使用标准 JDBC 格式发送文本数据：

```
PreparedStatement.setAsciiStream(int paramIndex,
   InputStream text, int length)
```

或

```
PreparedStatement.setUnicodeStream(int paramIndex,
   InputStream text, int length)
```

或

```
PreparedStatement.setCharacterStream(int paramIndex,
   Reader reader, int length)
```

使用 *date* 和 *time* 数据类型

jConnect for JDBC 支持 Adaptive Server datetime、 smalldatetime、 bigdatetime、 bigtime、 date 和 time 数据类型：

- datetime 可保存从 1753 年 1 月 1 日到 9999 年 12 月 31 日之间的日期。在支持 1/300 秒精度级别的平台上， datetime 可精确到该级别。

- smalldatetime 可保存从 1900 年 1 月 1 日到 2079 年 6 月 6 日之间的日期，可以精确到分钟。
- bigdatetime 表示自 0000 年 1 月 1 日 0:00:00.000000 以来所经过的微秒数。bigdatetime 值的合法范围为 0001 年 1 月 1 日 00:00:00.000000 到 9999 年 12 月 31 日 23:59:59.999999。
- bigtime 表示自一天开始所经过的微秒数。 bigtime 值的合法范围为 00:00:00.000000 到 23:59:59.999999。
- date 可保存从 0001 年 1 月 1 日到 9999 年 12 月 31 日之间的日期，与 java.sql.Date 中允许使用的值完全匹配。在 java.sql.Date 与 date 数据类型间存在直接映射。
- time 可保存从 00:00:00:000 到 23:59:59:990 之间的时间。在 java.sql.Time 与 time 数据类型间存在直接映射。

用法 （date、 time、 datetime 和 smalldatetime）

- 如果从包含 date 或 time 列的表中选择数据，并且尚未 （通过设置 jConnect 版本）在 jConnect 中启用 date/time 支持，则服务器会在返回 date 或 time之前尝试将其转换为 datetime 值。如果返回的日期早于 1/1/1753，则会产生问题。在这种情况下，会出现转换错误，数据库会向您通知该错误。
- SQL Anywhere 支持 date 和 time 数据类型，但 date 和 time 数据类型并不直接与 Adaptive Server 12.5.1 和更高版本中的数据类型相兼容。在使用 jConnect 与 SQL Anywhere 进行通信时，应当继续使用 datetime 和 smalldatetime 数据类型。
- 在 SQL Anywhere 中， datetime 列中的最大值是 1-1-7911 00:00:00。
- 使用 jConnect 时，如果试图将早于 1753 年 1 月 1 日的日期插入 datetime 列或参数中，则会收到转换错误。
- 有关 date 和 time 数据类型的详细信息，请参见 Adaptive Server 手册；请特别注意其中有关可执行的隐式转换的一节。
- 如果对 Adaptive Server date、time 或 datetime 列使用 getObject，则返回的值分别为 java.sql.Date、 java.sql.Time 或 java.sql.Timestamp 数据类型。

用法 （bigdatetime 和 bigtime）

- 连接到 Adaptive Server 15.5 及更高版本时， jConnect for JDBC 会使用 bigdatetime 和 bigtime 数据类型传输数据，即使接收 Adaptive Server 列定义为 datetime 和 time 也是如此。

  这意味着， Adaptive Server 可能自动截断来自 jConnect for JDBC 的值以符合 Adaptive Server 列。例如，在数据类型为 time 的 Adaptive Server 列中， bigtime 值 23:59:59.999999 保存为 23:59:59.996。

- 连接到 Adaptive Server 15.0.x 和更早版本时，jConnect for JDBC 使用 datetime 和 time 数据类型传输数据。

使用 *char*/*varchar*/*text* 数据类型和 *getByte*

如果数据不是十六进制、八进制或十进制数据，请勿对 char、univarchar、 unichar、 varchar 或 text 字段使用 rs.getByte。

支持的其它数据类型

jConnect 支持下列 Adaptive Server 数据类型：

- bigint – 一种精确数值数据类型，设计为在现有 int 类型范围不足时使用。
- unsigned int – 无符号形式的精确数值整数数据类型：unsignedsmallint、 unsignedint 和 unsignedbigint。
- unitext – 用于 Unicode 字符的可变长度数据类型。

Bigint 数据类型

Sybase 支持 bigint，它是一种 64 位整数数据类型，作为本机 Adaptive Server 数据类型受到支持。在 Java 中，此数据类型映射到 Java 数据类型 long。若要将此数据类型用作参数，可以调用 PreparedStatement.setLong(int index, long value)，jConnect 即会将数据作为 bigint 发送给 Adaptive Server。在从 bigint 列进行检索时，可以使用 ResultSet.getLong(int index) 方法。

Unitext 数据类型

在使用 unitext 数据类型时， jConnect 中并没有发生 API 更改。使用 unitext 列时， jConnect 可以在内部存储和检索来自 Adaptive Server 的数据。

Unsigned int 数据类型

Adaptive Server 支持将 *unsigned bigint*、 *unsigned int* 和 *unsigned smallint* 作为本机 Adaptive Server 数据类型。由于在 Java 中并没有与之相对应的无符号数据类型，因此，如果要正确处理数据，必须用 set 和 get 语句设置和获取下一个较大整数。例如，如果要从 *unsigned int* 中检索数据，使用 Java 数据类型 int 则会太小，不能包含大的正值，因此，ResultSet.getInt (int index) 可能会返回不正确的数据或抛出例外。若要正确处理数据，应该用 get 语句获取下一个较大整数值 ResultSet.getLong()。可以按照下表用 set 或 get 语句设置或获取数据。

Adaptive Server 数据类型	Java 数据类型
unsigned smallint	setInt()、 getInt()
unsigned int	setLong()、 getLong()
unsigned bigint	setBigDecimal()、 getBigDecimal()

仅数据锁定表中的可变长度行

如果可变长度列从行起始位置后超过 8191 字节处开始，则配置了 16K 逻辑页大小且低于 15.7 版的 Adaptive Server 无法创建含有可变长度行的仅数据锁定 (DOL) 表。从 Adaptive Server 15.7 开始删除了这一限制。请参见 Adaptive Server Enterprise 《性能和调优系列：物理数据库调优》中的第 2 章 “数据存储”。

JDBC 客户端不需要特殊配置即可使用此功能。当连接到配置为接收宽 DOL 行的 Adaptive Server 15.7 版时，这些客户端会自动使用宽偏移插入记录。如果客户端尝试向 Adaptive Server 的早期版本或禁用了 DOL 宽行选项的 15.7 版 Adaptive Server 发送 DOL 宽行，则会收到错误消息。

大对象 (LOB) 支持

jConnect 支持使用大对象 (LOB) 数据类型 （text、 unitext 和 image）作为：

- 具有行内存储的 LOB 列

  在 Adaptive Server 中，当有足够的内存存储整个行时，标记为行内的 LOB 列存储在行内。如果由于对行中的列进行更新而使行大小增大并超过定义的限制值，在行内存储的 LOB 列会移动到行外以使其仍不超过限制值。请参见 Adaptive Server Enterprise 《*Transact-SQL* 用户指南》中的第 21 章 “行内、行外 LOB”。

  jConnect 中的批量插入例程支持 Adaptive Server 中 text、 image 和 unitext LOB 列的行内和行外存储。早期客户端版本中的批量插入例程始终将 LOB 列存储在行外。

- 用作存储过程参数的 LOB 对象

  jConnect 支持使用 text、 unitext 和 image 作为存储过程中的输入参数以及作为参数标记数据类型。

大对象定位符支持

jConnect 支持大对象 (LOB) 定位符。 LOB 定位符中包含指向 LOB 数据的逻辑指针，而不是数据本身，减少了通过网络在 Adaptive Server 和其客户端之间传送的数据量。从 Adaptive Server 15.7 开始，引入了对 LOB 定位符的服务器支持。

当连接到支持 LOB 定位符的 Adaptive Server 并关闭 autocommit 时，jConnect 使用服务器端定位符访问 LOB 数据。否则，jConnect 会在客户端物化 LOB 数据。可以将整个 LOB API 用于客户端物化 LOB 数据，但由于数据较大， API 性能可能会与用于 LOB 定位符时不同。

注释 当您使用 LOB 定位符时，检索每个行上都包括 LOB 数据的大结果集可能会影响应用程序的性能。Adaptive Server 将 LOB 定位符作为结果集的一部分返回，为获取 LOB 数据， jConnect 必须高速缓存剩余的结果集。 Sybase 建议您让结果集小一些，或者启用游标支持来限制要高速缓存的数据的大小。

要启用 LOB 定位符支持，请在 ENABLE_LOB_LOCATORS 连接属性设置为 true 的情况下建立与 Adaptive Server 的连接。启用后，客户端应用程序便可以使用 java.sql 软件包中的 Blob、Clob 和 NClob 类访问定位符。

注释 如果 LOB 定位符和 autocommit 都已启用， jConnect 会自动将 LOB 定位符切换为客户端物化 LOB，即使 Adaptive Server 能够支持 LOB 定位符也是如此。这会增大客户端使用的内存，而且可能会降低性能。因此，建议您在设置了 autocommit off 时使用 LOB 定位符。

有关 Blob、 Clob 和 NClob 类的信息，请参见 Java 文档。

实施高级功能

本节介绍如何使用 jConnect 高级功能，包含以下主题：

- 使用 BCP 插入功能
- 受支持的 Adaptive Server Cluster Edition 功能
- 使用事件通知
- 处理错误消息
- 使用口令加密

- 将 Java 对象作为列数据存储在表中
- 使用动态类装载
- JDBC 4.0 规范支持
- JDBC 3.0 规范支持
- JDBC 2.0 选件工具包扩展支持

使用 BCP 插入功能

jConnect 支持使用 bulk load 插入功能将大量行插入 Adaptive Server 12.5.2 和更高版本中。虽然此功能不要求在服务器上进行特殊配置，但是较大的页大小、网络包大小和最大的内存大小可显著改进性能。根据客户端内存，使用较大的批处理文件也可改进性能。

要启用此功能，应将 ENABLE_BULK_LOAD 设置为以下任何一种有效值：

- ARRAYINSERT_WITH_MIXED_STATEMENTS – 使用行级别日志记录启用批量装载，并允许应用程序在批量装载操作过程中执行其它语句。
- ARRAYINSERT – 使用行级别日志记录启用 bulk load，但应用程序在执行 bulk load 操作期间无法执行其它语句。
- BCP – 使用页面级别日志记录启用批量装载，但应用程序在批量装载操作过程中无法执行其它语句。
- LOG_BCP – 使用页面级别日志记录 （使用 ASE 快速记录 BCP 功能）启用批量装载；但应用程序在批量装载操作过程中无法执行其它语句。

使用预准备语句并且将 ENABLE_BULK_LOAD 设置为有效值时，jConnect 会使用 BULK 例程将一批记录插入 Sybase 数据库。

受支持的 Adaptive Server Cluster Edition 功能

jConnect 支持 Adaptive Server Cluster Edition 环境，其中多个 Adaptive Server 连接到一组共享磁盘和高速专用互连。这样， Adaptive Server 即可使用多个物理和逻辑主机进行扩展。

有关 Cluster Edition 的详细信息，请参见 Adaptive Server Enterprise 《集群用户指南》。

登录重定向

在任何给定时间， Cluster Edition 环境中通常有一些服务器的工作负荷比其它服务器高。当客户端应用程序试图连接到繁忙的服务器时，登录重定向功能可允许服务器将客户端连接重定向到集群中不太繁忙的服务器，从而帮助平衡服务器的负荷。登录重定向过程在登录序列中进行，客户端应用程序不会收到被重定向的通知。当客户端应用程序连接到支持登录重定向功能的服务器时，会自动启用此功能。

注释 当客户端应用程序连接到配置为重定向客户端的服务器时，登录时间可能会增加，因为在将客户端连接重定向到另一台服务器时，会重新启动登录过程。

连接迁移

连接迁移功能允许 Cluster Edition 环境中的服务器动态分配负荷，并将现有客户端连接及其环境无缝迁移到集群中的另一台服务器。此功能使 Cluster Edition 环境能够充分利用资源并减少计算时间。因为服务器间的迁移是无缝的，所以连接迁移功能还可帮助创建零停机时间的高可用性环境。当客户端应用程序连接到支持连接迁移功能的服务器时，会自动启用此功能。

注释 在服务器迁移过程中，命令执行时间可能会增加。 Sybase 建议您相应地增加命令超时。

连接故障切换

连接故障切换功能允许客户端应用程序在主服务器因意外事件 （例如断电或套接字失败）变得不可用时切换到备用 Adaptive Server。在集群环境中，客户端应用程序可以使用动态故障切换地址多次故障切换到多台服务器。

启用高可用性选项后，客户端应用程序无需配置为知道可能的故障切换目标。 Adaptive Server 始终使用基于集群成员资格、逻辑集群使用情况和负荷分配的最佳故障切换列表更新客户端。在故障切换过程中，客户端参照有序的故障切换列表来尝试重新连接。如果驱动程序成功连接到服务器，则驱动程序会在内部根据返回的列表更新主机值列表。否则，驱动程序会引发连接失败异常。

启用连接故障切换

可以使用连接字符串通过将 REQUEST_HA_SESSION 设置为 true 来启用连接故障切换。例如：

```
URL="jdbc:sybase:Tds:server1:port1,server2:port2,...,
serverN:portN/mydb?REQUEST_HA_SESSION=true"
```

其中 server1:port1, server2:port2, ..., serverN:portN 是有序故障切换列表。

建立连接时，jConnect 会尝试连接到故障切换列表中指定的第一个主机和端口。如果失败，则会遍历列表，直到建立连接，或直到到达列表末尾。

注释 连接字符串中指定的备用服务器列表只在初始连接过程中使用。使用任何可用实例建立连接后，如果客户端支持高可用性，则客户端会从服务器收到最可能成为故障切换目标的更新列表。此新列表将覆盖指定列表。

使用事件通知

可以使用 jConnect 事件通知功能，让应用程序在执行 Open Server 过程时获得通知。

若要使用这一功能，必须使用 SybConnection 类，该类扩展了 Connection 接口。 SybConnection 包含 regWatch 方法和 regNoWatch 方法，分别用于打开事件通知和关闭事件通知。

应用程序还必须实现 SybEventHandler 接口。该接口包含一个公共方法 void event(String proc_name, ResultSet params)，在发生指定事件时将调用该方法。事件的参数被传递给 event，后者告知应用程序如何进行响应。

若要在应用程序中使用事件通知，请调用 SybConnection.regWatch( ) 将应用程序注册到已注册过程的通知列表中。使用以下语法：

SybConnection.regWatch(*proc_name*,*eventHdlr*,*option*)

其中：

- *proc_name* 是一个字符串，是用于生成通知的注册过程的名称。
- *eventHdler* 是实现的 SybEventHandler 类的实例。
- *option* 是 NOTIFY_ONCE 或 NOTIFY_ALWAYS。如果希望应用程序仅在过程首次执行时得到通知，请使用 NOTIFY_ONCE。如果希望应用程序在过程每次执行时均得到通知，请使用 NOTIFY_ALWAYS。

每当 Open Server 上发生具有指定 *proc_name* 的事件时，jConnect 都会从独立线程中调用 eventHdlr.event。 eventHdlr.event 在执行时会接收传递来的事件参数。因为这是一个独立的线程，所以事件通知不会阻止应用程序的执行。

如果 *proc_name* 不是已注册过程，或者 Open Server 无法将客户端添加到通知列表中，则调用 regWatch 会抛出 SQL 例外。

若要关闭事件通知，请使用以下调用过程：

```
SybConnection.regNoWatch(proc_name)
```

警告！ 使用 Sybase 事件通知扩展时，应用程序需要对连接调用 close 方法，以删除首次调用 regWatch 时创建的子线程。否则，当退出应用程序时可能会导致虚拟机挂起。

事件通知示例

下面的示例演示在建立连接后如何实现事件处理程序，然后向该事件处理程序的某个实例注册事件：

```
public class MyEventHandler implements SybEventHandler
{
  // Declare fields and constructors, as needed.
  ...
  public MyEventHandler(String eventname)
  {
    ...
  }

  // Implement SybEventHandler.event.
  public void event(String eventName, ResultSet params)
  {
    try
    {
      // Check for error messages received prior to event
      // notification.
      SQLWarning sqlw = params.getWarnings();
      if sqlw != null
      {
        // process errors, if any
        ...
      }
      // process params as you would any result set with
      // one row.
```

```
      ResultSetMetaData rsmd = params.getMetaData();
      int numColumns = rsmd.getColumnCount();
      while (params.next())          // optional
      {
        for (int i = 1; i <= numColumns; i++)
        {
          System.out.println(rsmd.getColumnName(i) + " =
            " + params.getString(i));
        }
        // Take appropriate action on the event.For example,
        // perhaps notify application thread.
        ...
      }
    }
    catch (SQLException sqe)
    {
      // process errors, if any
      ...
    }
  }
}

public class MyProgram
{
  ...
  // Get a connection and register an event with an instance
  // of MyEventHandler.
  Connection conn = DriverManager.getConnection(...);
  MyEventHandler myHdlr = new  MyEventHandler("MY_EVENT");

  // Register your event handler.
  ((SybConnection)conn).regWatch("MY_EVENT", myHdlr,
    SybEventHandler.NOTIFY_ALWAYS);
  ...
 conn.regNoWatch("MY_EVENT");
  conn.close();
}
```

处理错误消息

jConnect 提供两个用于返回特定于 Sybase 的错误信息的类：SybSQLException 和 SybSQLWarning，并提供一个 SybMessageHandler 接口，用于自定义 jConnect 处理来自服务器的错误消息的方式。

处理作为警告返回的数字错误

在 Adaptive Server 12.0 至 12.5 中，缺省情况下，按严重级别 10 对数字错误进行处理。严重级别为 10 的消息归类为状态信息消息，而不是归类为错误，并且在 SQLWarning 对象中传输其内容。下面的代码摘录说明此处理过程：

```
static void processWarnings(SQLWarning warning)
{
if (warning != null)
 {
  System.out.println ("\n -- Warning received -- \n");
 }//end if
  while (warning != null)
 {
  System.out.println ("Message:"
+  warning.getMessage());
  System.out.println("SQLState:"
+  warning.getSQLState());
  System.out.println ("ErrorCode:" +
  warning.getErrorCode());
  System.out.println ("----------------------------");
  warning = warning.getNextWarning();
 }//end while
}//end processWarnings
```

当出现数字错误时，返回的 ResultSet 对象中不包含任何结果集数据，必须从 SQLWarning 获取有关此错误的相关信息。因此，在 JDBC 应用程序中，检查并处理 SQLWarning 的代码不应依赖于结果集。例如，下面的代码在用于处理 while 循环的结果集的内部和外部检查并处理 SQLWarning 数据：

```
while (rs.next())
{
 String value = rs.getString(1);
 System.out.println ("Fetched value:" + value);

 // Check for SQLWarning on the result set.
 processWarnings (rs.getWarnings());

}//end while

 // Check for SQLWarning on the result set.
 processWarnings (rs.getWarnings());
```

此处代码在没有结果集数据 （rs.next( ) 为 false）的情况下检查 SQLWarning。下面的示例是正确编写为检测并报告数字错误的程序的输出。错误是除数为零：

```
-- Warning received --

Message:Divide by zero occurred.
SQLState:01012
ErrorCode:3607
```

检索特定于 Sybase 的错误信息

jConnect 提供 EedInfo 接口来指定获取特定于 Sybase 的错误信息的方法。EedInfo 接口在 SybSQLException 和 SybSQLWarning 中实现，它们是 SQLException 和 SQLWarning 的扩展类。

SybSQLException 和 SybSQLWarning 包含以下方法：

- public ResultSet getEedParams，返回包含附带了错误消息的所有参数值的单行结果集。
- public int getStatus，如果消息中包含参数值，它将返回 “1”；如果不包含，则返回 “0”。
- public int getLineNumber，返回引发了错误消息的存储过程或查询的行号。
- public String getProcedureName，返回引发了错误消息的过程的名称。
- public String getServerName，返回生成消息的服务器的名称。
- public int getSeverity，返回错误消息的严重性。
- public int getState，返回有关服务器中错误消息的内部源的信息。（仅用于 Sybase 技术支持部门。）
- public int getTranState，返回以下事务状态之一：
  - 0 连接当前处于扩展事务中。
  - 1 前一事务已成功提交。
  - 3 前一事务中止。

有些错误消息可能是 SQLException 或 SQLWarning 消息，但不是 SybSQLException 或 SybSQLWarning 消息。应用程序应先检查其正在处理的例外类型，然后下转到 SybSQLException 或 SybSQLWarning。

自定义错误消息处理

可以使用 SybMessageHandler 接口来自定义 jConnect 处理服务器生成的错误消息的方式。通过在自己的类中实现 SybMessageHandler 来处理错误消息有如下好处：

- “通用”错误处理

错误处理逻辑可放在错误消息处理程序中，而不需要在整个应用程序中重复。

- “通用”错误记录

  错误消息处理程序可包含处理所有错误记录的逻辑。

- 根据应用程序要求重新映射错误消息严重性

  错误消息处理程序可以包含识别特定错误消息的逻辑，并根据应用程序的要求而不是根据服务器的严重性级别来降低或提高错误的严重级。例如，在进行删除旧行的清除操作期间，可能想降低消息“一行不存在”的严重级。然而，在其它环境中又可能需要提高严重级。

注释 实现 SybMessageHandler 接口的错误消息处理程序仅接收服务器生成的消息。它们不处理由 jConnect 生成的消息。

---

jConnect 接收到错误消息时，它会检查是否已注册 SybMessageHandler 类来处理该消息。如果已注册， jConnect 将调用 messageHandler 方法，该方法接受 SQL 例外作为其参数。然后，jConnect 根据 messageHandler 返回的值处理该消息。错误消息处理程序可以：

- 依原样返回 SQL 例外。
- 返回 null。 jConnect 将忽略此消息。
- 根据 SQL 例外创建 SQL 警告，并将其返回。警告结果被添加到警告消息链中。
- 如果初始消息是 SQL 警告， messageHandler 可将此 SQL 警告评估为紧急，并创建和返回一个 SQL 例外，一旦控制权返回到 jConnect 就会引发此例外。

安装错误消息处理程序

通过从 SybDriver、SybConnection 或 SybStatement 调用 setMessageHandler 方法可安装实现 SybMessageHandler 的错误消息处理程序。如果从 SybDriver 安装错误消息处理程序，则所有后续 SybConnection 对象都会继承此处理程序。如果从 SybConnection 对象安装错误消息处理程序，则由该 SybConnection 对象创建的所有 SybStatement 对象都会继承此处理程序。

此继承关系仅从安装了错误消息处理程序对象后才起作用。例如，如果创建一个名为 “myConnection”的 SybConnection 对象，然后调用 SybDriver.setMessageHandler 来安装错误消息处理程序对象，则 “myConnection”无法使用此对象。

若要返回当前错误消息处理程序对象，请使用 getMessageHandler。

错误消息处理程序示例

```
 import java.io.*;
import java.sql.*;
 import com.sybase.jdbcx.SybMessageHandler;
 import com.sybase.jdbcx.SybConnection;
 import com.sybase.jdbcx.SybStatement;
import java.util.*;

 public class MyApp
 {
   static SybConnection conn = null;
   static SybStatement stmt = null
   static ResultSet rs = null;
   static String user = "guest";
   static String password = "sybase";
   static String server = "jdbc:sybase:Tds:192.138.151.39:4444";
   static final int AVOID_SQLE = 20001;

   public MyApp()
   {
try
      {
              Class.forName("com.sybase.jdbc4.jdbc.SybDriver").newInstance();
              Properties props = new Properties();
              props.put("user", user);
              props.put("password", password);
              conn = (SybConnection)
              DriverManager.getConnection(server, props);
              conn.setMessageHandler(new NoResultSetHandler());
              stmt =(SybStatement) conn.createStatement();
              stmt.executeUpdate("raiserror 20001 'your error'");

              for (SQLWarning sqw = _stmt.getWarnings();
              sqw != null;
              sqw = sqw.getNextWarning());
              {
                 if (sqw.getErrorCode() == AVOID_SQLE);
                 {
```

```
                System.out.println("Error" +  sqw.getErrorCode()+
                " was found in the Statement°Øs warning list.");
                break;
               }
            }
            stmt.close();
            conn.close();
  }
  catch (Exception e)
  {
    System.out.println(e.getMessage());
    e.printStackTrace();
  }
}

class NoResultSetHandler implements SybMessageHandler
{
  public SQLException messageHandler(SQLException sqe)
  {
    int code = sqe.getErrorCode();
    if (code == AVOID_SQLE)
    {
      System.out.println("User " + _user + " downgrading " +
        AVOID_SQLE + " to a warning");
      sqe = new SQLWarning(sqe.getMessage(),
        sqe.getSQLState(),sqe.getErrorCode());
    }
    return sqe;
  }
}

public static void main(String args[])
{
  new MyApp();
}
```

使用口令加密

缺省情况下， jConnect for JDBC 通过网络向 Adaptive Server 发送纯文本口令以进行鉴定。但是， jConnect 也支持对称和非对称口令加密，并可以在通过网络发送口令之前对口令进行加密。对称加密机制使用相同的密钥来加密和解密口令，而非对称加密机制使用一个密钥 （公用密钥）加密口令，使用另一个密钥 （私有密钥）解密口令。因为私有密钥不在网络上共享，所以认为非对称加密比对称加密安全。启用口令加密后，如果服务器支持非对称加密，则使用此格式，而不使用对称加密。

注释 若要使用非对称口令加密功能，必须有一个支持口令加密的服务器，例如 Adaptive Server 15.0.2。

启用口令加密

ENCRYPT_PASSWORD 连接属性指定是否以加密格式传输口令。此同一属性用于启用非对称密钥加密。启用口令加密后，如果服务器支持非对称密钥加密，则使用此格式，而不使用对称密钥加密。

将 ENCRYPT_PASSWORD 连接属性设置为 true 将启用口令加密。缺省值为 false。

注释 如果服务器配置为要求客户端使用加密口令，则输入纯文本口令会导致用户登录失败。

使用明文口令重试登录

当 ENCRYPT_PASSWORD 属性设置为 True，但服务器不支持口令加密时，服务器登录失败。若要对不支持口令加密的服务器使用明文口令，请将 RETRY_WITH_NO_ENCRYPTION 连接属性设置为 True。

当 ENCRYPT_PASSWORD 和 RETRY_WITH_NO_ENCRYPTION 属性都设置为 True 时， jConnect 会首先使用加密口令登录。如果登录失败， jConnect 将使用明文口令登录。

设置 Java Cryptography Extension (JCE) 提供程序

非对称口令加密机制使用 RSA 加密算法加密要传输的口令。若要执行此 RSA 加密，请为 JRE 配置合适的 Java Cryptography Extension (JCE) 提供程序。配置的 JCE 提供程序应能够支持“RSA/NONE/OAEPWithSHA1AndMGF1Padding”转换。

JRE 附带的 JCE 提供程序可能无法处理“RSA/NONE/OAEPWithSHA1AndMGF1Padding”转换。若要在这种情况下使用扩展口令加密功能，请配置一个支持此转换的外部 JCE 提供程序。如果 JCE 无法处理所需转换，您将在登录时收到错误消息。

可以使用 JCE_PROVIDER_CLASS 连接属性指定 JCE 提供程序。您可以从大量商业和开放源代码 JCE 提供程序中进行选择。例如，“Bouncy Castle Crypto APIs for Java”就是一种广泛使用的开放源代码 Java JCE 提供程序。如果选择不指定 JCE_PROVIDER_CLASS 属性，jConnect 将尝试使用任何捆绑的 JCE。

使用 GSE-J 执行 RSA 口令加密

可以使用 Certicom Security Builder GSE-J 执行 RSA 口令加密。Certicom Security Builder GSE-J 是一个符合 FIPS 140-2 的 JCE 提供程序，包括在 jConnect 驱动程序中。此提供程序包括两个 JAR 文件：*EccpressoFIPS.jar* 和 *EccpressoFIPSJca.jar*；可从 *$JDBC_HOME/classes* 和 *$JDBC_HOME/devclasses* 目录访问这两个文件。

要使用 Certicom Security Builder GSE-J 提供程序，应将 JCE_PROVIDER_CLASS 连接属性的值设置为“com.certicom.ecc.jcae.Certicom”。

注释 如果通过设置 ENCRYPT_PASSWORD 连接属性而非 JCE_PROVIDER_CLASS 连接属性启用了口令加密，则 jConnect 会尝试查找并装载 Certicom Security Builder GSE-J 提供程序。此过程仅在 *EccpressoFIPS.jar* 和 *EccpressoFIPSJca.jar*与 jConnect JAR 文件（*jconn4.jar* 或 *jconn4d.jar*）位于相同目录中时才会成功。

指定自定义 JCE 提供程序

要指定自定义 JCE 提供程序：

- 将 JCE_PROVIDER_CLASS 属性设置为要使用的提供程序的定全限定类名。例如，要使用 Bouncy Castle JCE：

```
String url = "jdbc:sybase:Tds:myserver:3697";
Properties props = new Properties();
props.put("ENCRYPT_PASSWORD ", "true");
props.put("JCE_PROVIDER_CLASS",
"org.bouncycastle.jce.provider.BouncyCastleProvider
");

/* Set up additional connnection properties as
```

```
needed */
 props.put("user", "xyz");
 props.put("password", "123");

 /* get the connection */
Connection con = DriverManager.getConnection(url,
props);
```

- 在使用 JCE 提供程序之前对其进行配置，为此，可使用以下两种方法之一：
  - 将 JCE 提供程序 *jar* 文件复制到 JRE 标准扩展目录中：
    - 对于 UNIX 平台：
      ${JAVA_HOME}/jre/lib/ext
    - 对于 Windows：
      %JAVA_HOME%\jre\lib\ext
  - 如果无法将 JCE *jar* 文件复制到适当的目录中，请参见JCE Reference Guide (http://docs.oracle.com/javase/1.4.2/docs/guide/security/jce/JCERefGuide.html)来获得有关设置外部 JCE 提供程序的说明。

如果 jConnect 无法使用指定的 JCE 提供程序，它会尝试使用 JRE 安全配置文件中配置的 JCE 提供程序。如果没有配置任何其它 JCE 提供程序，或者配置的提供程序不支持所需转换并且启用了口令加密，则连接将失败。

将 Java 对象作为列数据存储在表中

某些数据库产品允许直接将 Java 对象作为列数据存储到数据库中。在这样的数据库中，Java 类被当作数据类型，可以声明以 Java 类作为其数据类型的列。

通过实现定义于 PreparedStatement 接口中的 setObject 方法以及定义于 CallableStatement 和 ResultSet 接口中的 getObject 方法， jConnect 可以将 Java 对象存储到数据库中。这样对使用本地 JDBC 类和方法的应用程序使用 jConnect 就可以直接将 Java 对象作为列数据进行存储和检索。

注释 若要使用 getObject 和 setObject，请将 jConnect 版本设置为 com.sybase.jdbcx.SybDriver.VERSION_4 或更高版本。请参见第 6 页的“使用 JCONNECT_ VERSION”。

下面几节介绍了使用 JDBC 与 jConnect 在表中存储和检索对象的要求和过程：

- 将 java 对象作为列数据存储的前提条件
- 将 Java 对象发送到数据库
- 接收来自数据库的 Java 对象

注释 Adaptive Server 12.0 及更高版本和 SQL Anywhere 6.0.x 及更高版本可在表中存储 Java 对象，但有一些限制。有关详细信息，请参见《*jConnect for JDBC* 发行公告》。

将 java 对象作为列数据存储的前提条件

若要将属于用户定义的 Java 类的 Java 对象存储在列中，必须满足三个要求：

- 该类必须实现 java.io.Serializable 接口。这是因为 jConnect 使用本地 Java 序列和非序列将对象发送到数据库，并接收从数据库返回的对象。
- 类定义必须安装在目标数据库中，或者 您必须使用 DynamicClassLoader (DCL) 直接从 SQL Anywhere 或 Adaptive Server 服务器装载类，并在使用类时视其位于本地 CLASSPATH 中。有关详细信息，请参见第 87 页的 “使用动态类装载”。
- 客户端系统必须在 *.class* 文件中包含类定义，该文件可通过本地 CLASSPATH 环境变量访问。

将 Java 对象发送到数据库

若要将用户定义的类的实例作为列数据发送，请使用下面的 setObject 方法之一 （根据 PreparedStatement 接口中的指定）：

```
void setObject(int parameterIndex, Object x, int targetSqlType,
   int scale) throws SQLException;
void setObject(int parameterIndex, Object x, int targetSqlType)
   throws SQLException;
void setObject(int parameterIndex, Object x) throws SQLException;
```

在 jConnect 中，若要发送 Java 对象，可使用 java.sql.Types.JAVA_OBJECT 目标 sql.Type，或者使用 java.sql.Types.OTHER。

下面的示例定义一个 Address 类，显示 Friends 表（该表包含数据类型为 Address 类的 Address 列）的定义，并向该表中插入一行。

```
public class Address implements Serializable
{
  public String   streetNumber;
   public String   street;
   public String   apartmentNumber;
   public String   city;
   public int   zipCode;
  //Methods
  ...
}

/* This code assumes a table with the following structure
**  Create table Friends:
**  (firstname varchar(30)  ,
**   lastname varchar(30),
**  address Address,
**  phone varchar(15))
*/

// Connect to the database containing the Friends table.
Connection conn =
   DriverManager.getConnection("jdbc:sybase:Tds:localhost:5000",
     "username", "password");

// Create a Prepared Statement object with an insert statement
//for updating the Friends table.
PreparedStatement ps = conn.prepareStatement("INSERT INTO
   Friends values (?,?,?,?)");

// Now, set the values in the prepared statement object, ps.
// set firstname to "Joan."
ps.setString(1, "Joan");

// Set last name to "Smith."
ps.setString(2, "Smith");

// Assuming that we already have "Joan_address" as an instance
// of Address, use setObject(int parameterIndex, Object x) to
// set the address column to "Joan_address."
ps.setObject(3, Joan_address);

// Set the phone column to Joan’ s phone number.
ps.setString(4, "123-456-7890");

// Perform the insert.
ps.executeUpdate();
```

接收来自数据库的 Java 对象

客户端 JDBC 应用程序可将接收到的来自数据库的 Java 对象放到结果集中，或作为从存储过程返回的输出参数的值。

如果结果集包含作为列数据的 Java 对象，请在 ResultSet 接口中使用以下 getObject 方法之一检索该对象：

```
Object getObject(int columnIndex) throws SQLException;
Object getObject(String columnName) throws SQLException;
```

如果从存储过程返回的输出参数包含 Java 对象，请在 CallableStatement 接口中使用下面的 getObject 方法检索该对象：

```
Object getObject(int parameterIndex) throws SQLException;
```

下面的示例说明如何使用 ResultSet.getObject(int parameterIndex) 将在结果集中接收的对象分配给类变量。本示例使用前一节中所用的 Address 类和 Friends 表，并提供了一个可在信封上打印姓名与地址的简单应用程序。

```
/*
 ** This application takes a first and last name, gets the
 ** specified person' s address from the Friends table in the
 ** database, and addresses an envelope using the name and
 ** retrieved address.
 */
 public class Envelope
 {
   Connection conn = null;
   String firstName = null;
   String lastName = null;
   String street = null;
   String city = null;
   String zip = null;

   public static void main(String[] args)
   {
     if (args.length < 2)
     {
     System.out.println("Usage:Envelope <firstName>
       <lastName>");
                  System.exit(1);
     }
     // create a 4" x 10" envelope
     Envelope e = new Envelope(4, 10);
     try
     {
       // connect to the database with the Friends table.
       conn = DriverManager.getConnection(
```

```
        "jdbc:sybase:Tds:localhost:5000", "username",
          "password");
      // look up the address of the specified person
      firstName = args[0];
      lastName = args[1];
      PreparedStatement ps = conn.prepareStatement(
        "SELECT address FROM friends WHERE " +
          "firstname = ?AND lastname = ?");
      ps.setString(1, firstName);
      ps.setString(2, lastName);
      ResultSet rs = ps.executeQuery();
                   if (rs.next())
      {
        Address a = (Address) rs.getObject(1);
        // set the destination address on the envelope
        e.setAddress(firstName, lastName, a);
      }
      conn.close();
    }
    catch (SQLException sqe)
    {
      sqe.printStackTrace();
                   System.exit(2);
    }
    // if everything was successful, print the envelope
    e.print();
  }
  private void setAddress(String fname, String lname, Address a)
  {
    street = a.streetNumber + " " + a.street + " " +
      a.apartmentNumber;
    city = a.city;
    zip = "" + a.zipCode;
  }
  private void print()
  {
    // Print the name and address on the envelope.
    ...
  }
}
```

可在 jConnect 安装目录下的 *sample2* 子目录中找到更详细的 HandleObject.java 示例。

使用动态类装载

SQL Anywhere 和 Adaptive Server 允许将 Java 类指定为：

- SQL 列的数据类型
- Transact-SQL 变量的数据类型
- SQL 列的缺省值

在早期版本中，只有出现在 jConnect CLASSPATH 中的类才是可访问的，也就是说，如果 jConnect 应用程序试图访问不在本地 CLASSPATH 中的类的实例，就将产生 java.lang.ClassNotFound 例外。

jConnect 6.05 及更高版本通过实现 DynamicClassLoader (DCL) 来直接从 SQL Anywhere 或 Adaptive Server 服务器装载类，并在使用类时视其位于本地 CLASSPATH 中。

会继承超类中的所有安全性功能。 Java 2 中实现的装载程序委托模型仍然存在 – jConnect 首先尝试从 CLASSPATH 中装载请求的类；如果失败， jConnect 会尝试 DynamicClassLoader。

有关使用 Java 和 Adaptive Server 的详细信息，请参见 《*Adaptive Server Enterprise* 中的 *Java*》。

使用 *DynamicClassLoader*

若要使用 DCL 功能：

1　创建并配置类装载程序。 jConnect 应用程序的代码应该类似如下所示：

```
Properties props = new Properties();

// URL of the server where the classes live.
String classesUrl = "jdbc:sybase:Tds:myase:1200";

// Connection properties for connecting to above server.
props.put("user", "grinch");
props.put("password", "meanone");
...

// Ask the SybDriver for a new class loader.
DynamicClassLoader loader = driver.getClassLoader(classesUrl, props);
```

2　使用 CLASS_LOADER 连接属性，使得新的类装载程序可用于执行查询的语句。创建类装载程序后，即可将其传递给后续连接，如下所示 （接着第 1 步中的代码示例）：

```
// Stash the class loader so that other connection(s)
// can know about it.
```

```
props.put("CLASS_LOADER", loader);

// Additional connection properties
props.put("user", "joeuser");
props.put("password", "joespassword");

// URL of the server we now want to connect to.
String url = "jdbc:sybase:Tds:jdbc.sybase.com:4446";

// Make a connection and go.
Connection conn = DriverManager.getConnection (url, props);
```

假定 Java 类定义如下所示：

```
class Addr {
      String street;
      String city;
      String state;
}
```

假定 SQL 表定义如下所示：

```
create table employee (char(100) name, int empid, Addr address)
```

3 如果客户端应用程序 CLASSPATH 中缺少 Addr 类，请使用下面的客户端代码：

```
Statement stmnt = conn.createStatement();
// Retrieve some rows from the table that has a Java class
// as one of its fields.
ResultSet rs = stmnt.executeQuery(
      "select * from employee where empid = ’ 19’ ");
if (rs.next() {
      // Even though the class is not in our class path,
      // we should be able to access its instance.
      Object obj = rs.getObject("address");
      // The class has been loaded from the server, so let's take a look.
      Class c = obj.getClass();
    // Some Java Reflection can be done here to access the fields of obj.
      ...
}
```

CLASS_LOADER 连接属性提供了一个方便的机制，可在许多连接中共享一个类装载程序。

应确保在许多连接中共享一个类装载程序不会导致类冲突。例如，假定 org.foo.Bar 类的两个不同且不兼容的实例存在于两个不同的数据库中，当使用同一装载程序访问这两个类时，就可能引发问题。在检查来自第一个连接的结果集时装载第一个类。当检查来自第二个连接的结果集时，已经装载了这个类。因此，将不会装载第二个类，而 jConnect 不能检测此情况。

不过， Java 具有一种内置机制，可确保类的版本与非序列化对象的版本信息匹配。 Java 至少会检测并报告上述情况。

类及其实例不需要驻留在同一数据库或服务器中，但装载程序和后续连接没有理由不引用同一数据库或服务器。

使用非序列化

下面的示例说明了如何非序列化本地文件中的对象。该序列化对象是一个驻留在服务器但不存在于 CLASSPATH 中的类的实例。

SybResultSet.getObject( ) 使用 DynamicObjectInputStream （装载来自 DynamicClassLoader 的类定义的 ObjectInputStream 的子类），而不是缺省系统 （“boot”）类装载程序。

```
// Make a stream on the file containing the
//serialized object.
FileInputStream fileStream = new FileInputStream("serFile");
// Make a "deserializer" on it.Notice that, apart
//from the additional parameter, this is the same
//as ObjectInputStreamDynamicObjectInputStream
stream = new DynamicObjectInputStream(fileStream, loader);
// As the object is deserialized, its class is
//retrieved through the loader from our server.
Object obj = stream.readObject();stream.close();
```

预装载 *.jar* 文件

jConnect 6.05 或更高版本具有一个名为 PRELOAD_JARS 的连接属性。将 *.jar* 文件定义为以逗号分隔的 *.jar* 文件名的列表时，这些文件会被全部装载。在这种情况下，“JAR”会引用服务器使用的 “保留的 JARname”。它是在安装 Java 程序中指定的 *.jar* 文件名，例如：

```
install java new jar 'myJarName' from file '/tmp/mystuff.jar'
```

如果设置 PRELOAD_JARS，则 *.jar* 文件会与类装载程序关联，因此不需要为每个连接预装载这些文件。您只需为一个连接指定 PRELOAD_JARS。以后尝试预装载相同的 *.jar* 文件可能会影响性能，因为不必要地从服务器检索 *.jar* 文件数据。

注释 SQL Anywhere 不能将 *.jar* 文件作为一个实体返回，因此 jConnect 会依次迭代检索每个类。不过， Adaptive Server 可检索整个 *.jar* 文件，并装载其包含的所有类。

高级功能

DynamicClassLoader 中有许多公共方法。有关详细信息，请参见 *JDBC_HOME/docs/en/javadocs* 中的 javadocs 信息。

附加功能包括能够在期望进行一系列类装载时使装载程序的数据库连接保持 “活动”状态，并能按照类名称显式装载单个类。

还可使用继承自 java.lang.ClassLoader 的公共方法。 java.lang.Class 中处理装载类的方法也是可用的；不过，使用这些方法时要小心，因为其中某些方法会对使用哪些类装载程序进行假定。尤其应使用 Class.forName 的 3 参数版本，否则会使用系统 （“boot”）类装载程序。请参见 第 75 页的 “处理错误消息”。

JDBC 4.0 规范支持

支持以下 JDBC 4.0 规范：

- 连接管理
- 自动 SQL 驱动程序装载
- 数据库元数据
- 国家字符集转换
- 包装模式
- 标量函数 CHAR_LENGTH、 CHARACTER_LENGTH、CURRENT_DATE、 CURRENT_TIME、 CURRENT_TIMESTAMP、EXTRACT 和 OCTET_LENGTH、 POSITION

有关 JDBC 4.0 规范的信息，请参见Oracle Technology Network for Java (http://www.oracle.com/technetwork/index.html)。

JDBC 3.0 规范支持

本节介绍当前 jConnect 7.0 版本中所支持的 JDBC 3.0 功能。

保存点支持

增加了 Savepoint 接口，其中包含设置、释放事务或将事务回退到指定保存点的方法。

在事务中使用保存点

JDBC 2.0 中的事务支持允许对事务进行控制并回退事务中的每个更改。在 JDBC 3.0 中，保存点带来了更大的控制力：Savepoint 接口允许将一个事务分成几个逻辑断点，以此对回退的事务量加以控制。

设置和回退到某个保存点

JDBC 3.0 API 增加了方法 Connection.setSavepoint，该方法在当前事务内设置一个保存点，并返回一个 Savepoint 对象。重载 Connection.rollback 方法来采用 Savepoint 对象参数。

释放保存点

Connection.releaseSavepoint 方法以 Savepoint 对象作为参数并将其从当前事务中删除。释放 Savepoint 后，如果尝试在回退操作中引用它，则会发生 SQLException。

当事务被提交或整个事务回退时，在事务中创建的任何保存点都自动释放并变成无效保存点。如果将事务回退到某个保存点，则它会自动释放任何其它的在该保存点之后创建的有问题的保存点并使之失效。

注释 可以使用 DatabaseMetaData.supportsSavepoints 方法确定 JDBC API 实现是否支持保存点。

参数元数据检索

添加接口 ParameterMetaData，该接口描述预准备语句参数的数量、类型和属性，并支持新的和修改的 DatabaseMetaData 方法。

自动生成键检索

增加了从包含自动生成值的列中检索值的方式。JDBC 3.0 能够满足获取自动生成键值或自动递增键值的日常需求。

确定生成键的值

若要通知驱动程序您要检索自动生成的键，请将常量 Statement.RETURN_GENERATED_KEYS 作为 Statement.execute() 方法的第二个参数传递。执行该语句后，调用 Statement.getGeneratedKeys() 检索生成的键。结果集将包含检索到的每个生成键的行。

注释 Adaptive Server 无法返回生成键的结果集。如果执行一批 insert 命令，则调用 Statement.getGeneratedKeys() 会只返回最后一个生成键的值。

有关检索自动生成键的详细信息 （包括示例代码），请在 Oracle Java Web 站点上搜索 “检索自动生成的键”。

可以拥有多个打开的 ResultSet 对象

添加了 getMoreResults(int)，它采用的参数指定 Statement 对象所返回的 ResultSet 对象是否应在返回任何后续 ResultSet 对象前关闭。

作为更改的一部分， JDBC 3.0 规范允许 Statement 接口支持多个打开的 ResultSets，从而取消了 JDBC 2 规范限制，即，返回多个结果的语句在任何指定时间只能打开一个 ResultSet。为支持多个打开的结果，Statement 接口增加了方法 getMoreResults() 的一个过载版本。getMoreResults(int) 方法采用整数标志，当调用 getResultSet() 方法时，该标志指定以前打开的 ResultSets 的行为。该接口定义的标志如下所示：

- CLOSE_ALL_RESULTS – 当调用 getMoreResults() 时，关闭以前打开的所有 ResultSet 对象。
- CLOSE_CURRENT_RESULT – 当调用 getMoreResults() 时，关闭当前 ResultSet 对象。
- KEEP_CURRENT_RESULT – 当调用 getMoreResults() 时，不关闭当前 ResultSet 对象。

按照名称将参数传递到 *CallableStatement* 对象

增加了允许字符串标识将为 CallableStatement 对象设置的参数的方法。

可以使用 CallableStatement 接口按照参数名称指定参数，而不使用过去指定参数索引的方法。您会发现当过程中包含很多具有缺省值的参数时新方法非常有效。可以使用命名参数来仅指定没有缺省值的值。

可保持游标支持

增加了指定 ResultSet 对象可保持性的能力。可保持游标 （或结果）是一种当包含该游标的事务被提交后不能自动关闭的对象。JDBC 3.0 增加了对指定游标可保持性的支持。要想指定 ResultSet 的可保持性，必须在使用 createStatement()、 prepareStatement() 或 prepareCall() 方法准备语句时完成这一工作。可保持性可以为下列常量之一：

- HOLD_CURSORS_OVER_COMMIT – ResultSet 对象 （游标）不关闭；当隐式或显式执行 commit 操作时，它们保持打开状态。
- CLOSE_CURSORS_AT_COMMIT – ResultSet 对象 （游标）在隐式或显式执行 commit 操作时关闭。

如果在提交事务时关闭游标，则通常会产生较好的性能。除非在完成事务后需要游标，否则最好在执行 commit 操作时关闭游标。因为规范没有定义 ResultSet 的缺省可保持性，所以其行为将取决于实现。

JDBC 2.0 选件工具包扩展支持

JDBC 2.0 Optional Package（《*JDBC 2.0* 选件工具包》）（旧称 *JDBC 2.0 Standard Extension API* （《*JDBC 2.0* 标准扩展 API》））中定义了多个 JDBC 2.0 驱动程序可实现的功能。 jConnect 6.05 版及更高版本已实现了以下选件工具包扩展功能：

- 用于命名数据库的 JNDI
  （使用 jConnect 支持的任何 Sybase DBMS）
- 连接池
  （使用 jConnect 支持的任何 Sybase DBMS）
- 分布式事务管理支持 仅用于 Adaptive Server。

注释 Sybase 建议使用 JNDI 1.2，它可与 Java 1.1.6 及更高版本兼容。

用于命名数据库的 JNDI

参考

JDBC 2.0 Optional Package （《JDBC 2.0 选件工具包》）（旧称 *JDBC 2.0 Standard Extension API* （《JDBC 2.0 标准扩展 API》））的第 5 章 “JNDI and the JDBC API”（JNDI 与 JDBC API）。

相关接口

- javax.sql.DataSource
- javax.naming.Referenceable
- javax.naming.spi.ObjectFactory

此功能为 JDBC 客户端提供了按标准方法获取数据库连接的替代方法。不再调用 Class.forName （“com.sybase.jdbc4.jdbc.SybDriver”），然后将 JDBC URL 传递到 DriverManager 的 getConnection( ) 方法，客户端可使用逻辑名访问 JNDI 命名服务器来检索 javax.sql.DataSource 对象。此对象负责装载驱动程序，并与它代表的物理数据库建立连接。客户端代码更简单并且是可重用的，因为特定于供应商的信息已放入 DataSource 对象中。

DataSource 对象的 Sybase 实现是 com.sybase.jdbcx.SybDataSource （有关详细信息，请参见 javadocs）。此实现通过使用 JavaBean 组件的设计模式支持以下标准属性：

- databaseName
- dataSourceName
- description
- networkProtocol
- password
- portNumber
- serverName
- user

注释 roleName 不受支持。

---

jConnect 提供 javax.naming.spi.ObjectFactory 接口的实现，因此 DataSource 对象可根据命名服务器条目的属性构造。给定 javax.naming.Reference，或 javax.naming.Name 和 javax.naming.DirContext 时，此 factory 可构造 com.sybase.jdbcx.SybDataSource 对象。若要使用此 factory，请将 java.naming.object.factory 系统属性设置为包括 com.sybase.jdbc4.SybObjectFactory。

用法

可以在不同的应用程序中，以不同的方式使用 DataSource。下面的几小节中介绍所有选项并提供一些代码示例，以指导您完成该过程。有关详细信息，请参见 *JDBC 2.0 Optional Package*（《JDBC 2.0 选件工具包》）（旧称 *JDBC 2.0 Standard Extension API*（《JDBC 2.0 标准扩展 API》））以及 Oracle Java Web 站点上的 JNDI 文档。

1a.管理员进行的配置：LDAP

jConnect 自 4.0 版本开始已经支持 LDAP 连接。因此，建议的方法（不要求自定义软件）是使用 LDAP 数据交换格式 (LDIF) 将 DataSources 配置为 LDAP 条目。例如：

```
dn:servername:myASE, o=MyCompany, c=US
1.3.6.1.4.1.897.4.2.5:TCP#1# mymachine 4000
1.3.6.1.4.1.897.4.2.10:PACKETSIZE=1024&user=me&password=secret
1.3.6.1.4.1.897.4.2.11:userdb
```

1b.由客户端访问

这是典型的 JDBC 客户端应用程序。唯一的区别是获取对 DataSource 对象的引用的方式是通过访问命名服务器，而不是通过访问 DriverManager 并提供 JDBC URL。获取连接后，客户端代码就和所有其它 JDBC 客户端代码相同了。代码非常通用，且设置对象 factory 属性（可设置为环境的一部分）时仅引用 Sybase。

jConnect 安装包含了示例程序 *sample2/SimpleDataSource.java* 以说明 DataSource 的使用。此示例仅用于参考，也就是说，除非适当地配置环境并编辑此示例，否则不能运行它。 *SimpleDataSource.java* 包含以下重要代码：

```
import javax.naming.*;
import javax.sql.*;
import java.sql.*;

// set necessary JNDI properties for your environment (same as above)
Properties jndiProps = new Properties();

// used by JNDI to build the SybDataSource
jndiProps.put(Context.OBJECT_FACTORIES,
    "com.sybase.jdbc4.jdbc.SybObjectFactory");

// nameserver that JNDI should talk to
jndiProps.put(Context.PROVIDER_URL, "ldap:some_ldap_server:238/" +
"o=MyCompany,c=Us");

// used by JNDI to establish the naming context
jndiProps.put(Context.INITIAL_CONTEXT_FACTORY,
    "com.sun.jndi.ldap.LdapCtxFactory");

// obtain a connection to your name server
```

```
Context ctx = new InitialContext(jndiProps);
DataSource ds = (DataSource) ctx.lookup("servername=myASE");

// obtains a connection to the server as configured earlier.
// in this case, the default username and password will be used
Connection conn = ds.getConnection();

// do standard JDBC methods
...
```

如果已经在虚拟机中定义属性，则不必将 Properties显式传递给 InitialContext 构造函数，也就是说，要么在将 Java 设置为浏览器属性的一部分时传递，要么使用下面的方法传递：

```
java -Djava.naming.object.factory=com.sybase.jdbc4.jdbc.SybObjectFactory
```

有关设置环境属性的详细信息，请参见 Java VM 文档。

2a.编程配置

此阶段通常由为其公司执行数据库系统管理或应用程序集成的人员来完成。目的是定义一个数据源，然后以一个逻辑名部署到命名服务器。如果需要重新配置服务器（例如，移到其它计算机、端口等等），管理员运行此配置实用程序（概述如下），并将此逻辑名重新分配给新的数据源配置。因此，客户端代码不会更改，因为它只认识此逻辑名。

```
import javax.sql.*;
import com.sybase.jdbcx.*;
.....

// create a SybDataSource, and configure it
SybDataSource ds = new com.sybase.jdbc4.jdbc.SybDataSource();
ds.setUser("my_username");
ds.setPassword("my_password");
ds.setDatabaseName("my_favorite_db");
ds.setServerName("db_machine");
ds.setPortNumber(4000);
ds.setDescription("This DataSource represents the Adaptive Server
    Enterprise server running on db_machine at port 2638.  The default
    username and password have been set to 'me' and 'mine' respectively.
    Upon connection, the user will access the my_favorite_db database on
    this server.");
Properties props = newProperties()
props.put("REPEAT_READ","false");
props.put("REQUEST_HA_SESSION","true");
ds.setConnectionProperties(props);
// store the DataSource object.Typically this is
// done by setting JNDI properties specific to the
// type of JNDI service provider you are using.
// Then, initialize the context and bind the object.
Context ctx = new InitialContext();
ctx.bind("java:comp/env/jdbc/myASE", ds);
```

设置 DataSource 后，需要决定信息的存储位置和存储方式。SybDataSource 提供了 java.io.Serializable 和 javax.naming.Referenceable 来帮助您作出决定，但仍需管理员根据使用的 JNDI 服务提供程序来决定如何存储数据。

2b.由客户端访问

客户端通过采用配置 DataSource 的方法来设置 DataSource 对象的 JNDI 属性，从而检索该对象。客户端需要一个可转换该对象的可用对象 factory，因为它会被存储 （例如，序列化）到 Java 对象中。

```
Context ctx = new InitialContext();
DataSource ds = (DataSource) ctx.lookup("java:comp/env/jdbc/myASE");
Connection conn = ds.getConnection();
```

连接池

参考

JDBC 2.0 Optional Package（《*JDBC 2.0* 选件工具包》）（旧称 *JDBC 2.0 Standard Extension API* （《*JDBC 2.0* 标准扩展 API》））的第 6 章“Connection Pooling”（连接池）。

相关接口

- javax.sql.ConnectionPoolDataSource
- javax.sql.PooledConnection

概述

传统数据库应用程序可与用于应用程序的每次会话的数据库创建一个连接。不过，使用此应用程序时，基于 Web 的数据库应用程序可能需要多次打开和关闭新连接。

处理基于 Web 的数据库连接的一个有效方法是使用连接池，它可维护打开的数据库连接并管理在不同用户请求间共享的连接，从而维护性能并减少空闲连接的数目。对于每个连接请求，连接池首先确定池中是否有空闲连接。如果有，连接池会返回空闲连接，而不是与数据库建立新连接。

提供 com.sybase.jdbc4.jdbc.ConnectionPoolDataSource 类来与连接池实现交互。使用 ConnectionPoolDataSource 时，池实现会监听 PooledConnection。当您关闭连接或有错误破坏了连接时，会通知该实现。此时，池实现会决定如何处理 PooledConnection。

如果没有连接池，事务会：

1 创建与数据库的连接。

2 向数据库发送查询。

3 获得结果集。

4 显示结果集。

5 破坏连接。

有连接池时，序列大致如下：

1 查看连接 “池”中是否存在未用连接。

2 如果有，则使用此连接；否则创建新连接。

3 向数据库发送查询。

4 获得结果集。

5 显示结果集。

6 将连接返回 “池”。用户仍然调用 “close( )”，但连接保持打开状态，且池会得到 close 请求的通知。

与每次客户端需要建立与数据库的连接时都创建一个新的连接相比，重新使用连接的开销要少。

为允许第三方实现连接池， jConnect 实现让 ConnectionPoolDataSource 接口生成 PooledConnections，类似于 DataSource 接口生成 Connections 的方法。

池实现使用 ConnectionPoolDataSource 的 getPooledConnection( ) 方法创建 “真正”的数据库连接。然后，池实现将自己注册为针对 PooledConnection 的监听器。

目前，当客户端请求连接时，池实现就会调用可用 PooledConnection 上的 getConnection( )。当客户端完成连接并调用 close 时，池实现就会通过 ConnectionEventListener 接口得到通知，告知连接空闲，可以重用。

如果客户端因为某种原因破坏了数据库连接，池实现也会通过 ConnectionEventListener 接口得到通知，这样池实现就会将连接从池中删除。

有关详细信息，请参见 *JDBC 2.0 Optional Package* （《*JDBC 2.0* 选件工具包》）（旧称 *JDBC 2.0 Standard Extension API* （《*JDBC 2.0* 标准扩展 *API*》））中的 “Appendix B”（附录 B）。

管理员进行的配置：LDAP

此方法与 “用于命名数据库的 JNDI”中描述的 “1a.管理员进行的配置：LDAP”相似，只是向 LDIF 条目中输入了附加行。在下面的示例中，添加的代码行以粗体显示以供参考。

```
dn:servername=myASE, o=MyCompany, c=US
1.3.6.1.4.1.897.4.2.5:TCP#1# mymachine 4000
1.3.6.1.4.1.897.4.2.10:PACKETSIZE=1024&user=me&password=secret
1.3.6.1.4.1.897.4.2.11:userdb
1.3.6.1.4.1.897.4.2.18:ConnectionPoolDataSource
```

通过中间层客户端进行访问

此过程初始化三个属性 （如第 78 页所示的 INITIAL_CONTEXT_FACTORY、 PROVIDER_URL 和 OBJECT_FACTORIES），并检索 ConnectionPoolDataSource 对象。有关更完整的代码示例，请参见 *sample2/SimpleConnectionPool.java*。基本区别是：

```
...
ConnectionPoolDatabase cpds = (ConnectionPoolDataSource)
    ctx.lookup("servername=myASE");
PooledConnection pconn = cpds.getPooledConnection();
```

分布式事务管理支持

此功能为 Adaptive Server 提供了用于执行分布式事务的标准 Java API。

注释 此功能设计用于大型多层环境。

参考

请参见 *JDBC 2.0 Optional Package* （《*JDBC 2.0* 选件工具包》）（旧称 *JDBC 2.0 Standard Extension API* （《*JDBC 2.0* 标准扩展 *API*》））的第 7 章 “Distributed Transactions”（分布式事务）。

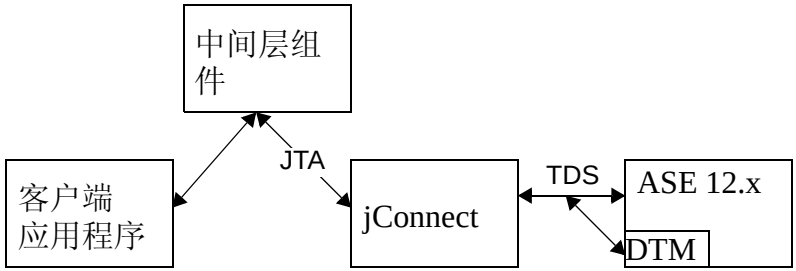
相关接口

- javax.sql.XADataSource
- javax.sql.XAConnection
- javax.transaction.xa.XAResource

背景和系统要求

- 因为在 Sybase Adaptive Server 12.0 和更高版本中 jConnect 会直接与资源管理器通信，所以安装必须具有分布式事务管理支持。
- 任何想参与分布式事物的用户都必须具有 “dtm_tm_role”授权，否则事务将失败。
- 若要使用分布式事务，必须在 */sp* 目录中安装存储过程。请参见《*jConnect for JDBC* 安装指南》第 1 章中的 “安装存储过程”。

图 2-2：12.x 版本中的分布式事务管理支持



管理员进行的配置：LDAP

此方法与第 93 页的 “用于命名数据库的 JNDI”中描述的 “1a.管理员进行的配置：LDAP”相似，只是向 LDIF 条目中输入了附加行。在下面的示例中，添加的代码行显示为粗体。

```
dn:servername:myASE, o=MyCompany, c=US
1.3.6.1.4.1.897.4.2.5:TCP#1# mymachine 4000
1.3.6.1.4.1.897.4.2.10:PACKETSIZE=1024&user=me&password=secret
1.3.6.1.4.1.897.4.2.11:userdb
1.3.6.1.4.1.897.4.2.18:XADataSource
```

通过中间层客户端进行访问

此过程初始化三个属性（INITIAL_CONTEXT_FACTORY、PROVIDER_URL 和 OBJECT_FACTORIES），并检索 XADataSource 对象。例如：

```
...
XADataSource xads = (XADatasource) ctx.lookup("servername=myASE");
XAConnection xaconn = xads.getXAConnection();
```

或覆盖用户名和口令的缺省设置：

```
...
XADataSource xads = (XADatasource) ctx.lookup("servername=myASE");
XAConnection xaconn = xads.getXAConnection("my_username","my_password");
```

对 JDBC 标准的约束与说明

本节讨论 JDBC 的 jConnect 实现是怎样偏离 JDBC 标准的。涉及以下主题：

不受支持的 JDBC 4.0 规范要求

此版本不支持以下内容：

- java.sql.RowID
- JDBC 4.0 中引入的 XML API

使用 Connection.isClosed 和 IS_CLOSED_TEST

根据 JDBC 2.1 规范的第 11.1 节：

“仅能确保 Connection.isClosed 方法在调用 Connection.close 后返回 true。通常情况下不能调用 Connection.isClosed 来确定数据库连接有效还是无效。典型客户端可通过捕获试图执行操作时引发的例外来确定连接无效。”

jConnect 对 isClosed 方法提供了一种缺省的解释，这与规范中定义的行为不同。调用 Connection.isClosed 时，jConnect 首先检验是否已对此连接调用了 Connection.close。如果已调用 close，则 jConnect 会为 isClosed 返回 true。

但是，如果未调用 Connection.close，则 jConnect 接下来将尝试在数据库上执行 sp_mda 存储过程。sp_mda 存储过程是 jConnect 用户在数据库上使用 jConnect 时必须安装的元数据的一部分。

调用 sp_mda 的目的是使 jConnect 可执行已知（或至少是期望）驻留在数据库服务器上的过程。如果存储过程正常执行，则 jConnect 会为 isClosed 返回 false，因为我们已经检验过数据库连接是有效的，且正在工作。但是，如果调用 sp_mda 引发 SQLException，则 jConnect 会捕获该例外并为 isClosed 返回 true，因为连接似乎有问题。

如果想强制 jConnect 更加遵循 isClosed() 的标准 JDBC 行为，则可将 IS_CLOSED_TEST 连接属性设置为特殊值 “INTERNAL”。INTERNAL 设置表示只有在已调用 Connection.close 时，或者当 jConnect 检测到禁用了连接的 IOException 时， jConnect 才会为 isClosed 返回 true。

也可指定在 isClosed 被调用时使用除 sp_mda 外的其它查询。例如，如果想让 jConnect 在调用 isClosed 时尝试 select 1，可将 IS_CLOSED_TEST 连接属性设置为 select 1。

对未处理的结果使用 Statement.close

关于首先调用 Statement.execute，然后对该同一语句对象调用 close，而没有处理 Statement 返回的所有结果 （更新计数和 ResultSets）时驱动程序应具有的行为， JDBC 规范的表述有点模糊。

例如，假定数据库上有一个进行七次行插入的存储过程。应用程序于是使用 Statement.execute 执行该存储过程。在这种情况下， Sybase 数据库向应用程序返回七个更新计数 （为每个插入的行返回一个计数）。根据常规 JDBC 应用程序逻辑，应该使用 getMoreResults、 getResultSet 和 getUpdateCount 方法在循环中处理这些更新计数。这些在Java SE documentation (http://www.oracle.com/technetwork/java/index.html)上的 java.sql.* 软件包的 javadocs 中解释得很清楚。

不过，应用程序程序员可能会在通览返回的所有更新计数前错误地选择调用 Statement.close。在这种情况下， jConnect 会向数据库发送 cancel，而这会造成无法预料的和不希望的负面影响。

在这个特殊的示例中，如果应用程序在数据库完成插入前调用 Statement.close，则数据库可能不会执行所有插入。例如，它可能会在插入 5 行后就停止，因为存储过程尚未结束就在数据库上取消了。

这种情况下就不会向您报告丢失的插入。在仍然具有未处理的结果时试图关闭 Statement 时， jConnect 的更高版本会抛出 SQLException，但在这之前，仍然强烈建议 jConnect 程序员遵循以下指南：

- 调用 Statement.close 时，如果您未处理完所有结果 （更新计数和 ResultSets）时向服务器发送了 cancel。在只执行 select 语句的情况下，这完全可以。但是，在执行 insert/update/delete 操作的情况下，这会导致某些操作不能如期望的那样完成。
- 因此，执行除 select 之外的其它操作时，切勿在还有未处理结果时调用 close。

- 因此，如果调用 Statement.execute，请用 getUpdateCount、getMoreResults 和 getResultSet 方法确认处理了所有结果。

调整多线程

如果多个线程同时调用同一 Statement 实例 （CallableStatement 或 PreparedStatement）上的方法 （Sybase 不建议这样做），您必须手动同步对 Statement 上的方法的调用；jConnect 不会自动执行此过程。

例如，如果有两个在同一 Statement 实例上运行的线程 （一个线程发送查询，另一个线程处理警告），则必须同步对 Statement 上的方法的调用，否则会发生冲突。

使用 *ResultSet.getCursorName*

有些 JDBC 驱动程序可为任何 SQL 查询生成游标名，这样总是可以返回一个字符串。不过，调用了 ResultSet.getCursorName 时 jConnect 不会返回名称，除非您：

- 调用相应的 Statement 上的 setFetchSize 或 setCursorName，或者
- 将 SELECT_OPENS_CURSOR 连接属性设置为 “true”，且查询的格式为 SELECT...FOR UPDATE。例如：

```
select au_id from authors for update
```

如果不调用相应 Statement 上的 setFetchSize 或 setCursorName，或不将 SELECT_OPENS_CURSOR 连接属性设置为 “true”，则会返回 null。

根据 JDBC 2.0 API 文档 （请参见第 11 章 “Clarifications”（说明）），所有其它 SQL 语句都不需要打开游标及返回名称。

有关如何在 jConnect 中使用游标的详细信息，请参见第 47 页的 “对结果集使用游标”。

执行存储过程

如果执行用问号代表参数值的 CallableStatement 对象中的存储过程，会比对参数既使用问号又使用实际值获得更好的性能。而且，如果混合使用实际值和问号，就不能对存储过程使用输出参数。

下面的示例将 *sp_stmt* 创建为 CallableStatement 对象，以执行存储过程 MyProc：

```
CallableStatement sp_stmt = conn.prepareCall(
   "{call MyProc(?,?)}");
```

MyProc 中的两个参数用问号表示。可以使用 CallableStatement 接口中的 registerOutParameter 方法将其中的一个或所有两个参数注册为输出参数。

在下面的示例中， *sp_stmt2* 是 CallableStatement 对象，用于执行存储过程 MyProc2。

```
CallableStatement sp_stmt2 = conn.prepareCall(
   {"call MyProc2(?,'javelin')}");
```

在 *sp_stmt2* 中，其中一个给定参数值为实际值，另一个为问号。不能将两个参数中的任一个注册为输出参数。

若要使用参数的名称绑定来通过 RPC 命令执行存储过程，请使用以下两个过程之一：

- 使用语言命令，用 PreparedStatement 类将输入参数直接从 Java 变量传递给语言命令。如以下代码段所示：

```
// Prepare the statement
System.out.println("Preparing the statement...");
String stmtString = "exec " + procname + " @p3=?, @p1=?";
PreparedStatement pstmt = con.preparedStatement(stmtString);

// Set the values
pstmt.setString(1, "xyz");
pstmt.setInt(2, 123);

// Send the query
System.out.println("Executing the query...");
ResultSet rs = pstmt.executeQuery();
```

- 对于 jConnect 6.05 及更高版本，请使用 com.sybase.jdbcx.SybCallableStatement 接口，如下面的示例所示：

```
import com.sybase.jdbcx.*;
....
// prepare the call for the stored procedure to execute as an RPC
String execRPC = "{call " + procName + " (?, ?)}";
SybCallableStatement scs = (SybCallableStatement)
con.prepareCall(execRPC);

// set the values and name the parameters
// also (optional) register for any output parameters
scs.setString(1, "xyz");
scs.setParameterName(1, "@p3");
scs.setInt(2, 123);
```

```
scs.setParameterName(2, "@p1");

// execute the RPC
// may also process the results using getResultSet()
// and getMoreResults()

// see the samples for more information on processing results
ResultSet rs = scs.executeQuery();
```

第 3 章

安全性

本章介绍 jConnect 的安全性问题。

概述

jConnect 提供了以下选项来保护客户端与服务器之间的通信：

- *SSL* – 使用 SSL 加密客户端和服务器应用程序之间的通信，包括登录交换。
- *Kerberos* – 使用 Kerberos 为 Adaptive Server 鉴定 Java 应用程序或 Java 应用程序的用户，而不需要通过网络发送用户名或口令。还可以使用 Kerberos 设置单点登录 (SSO) 环境，并提供 Java 应用程序的数字标识和 Adaptive Server Enterprise 的数字标识之间的相互鉴定。

  注释 Kerberos 可用于加密通信并提供数据完整性检查，但尚未对 jConnect 实现这些功能。

Kerberos 和 SSL 也可一起使用，这样可以提供 SSO 以及客户端和服务器应用程序之间传输的数据加密的优点。

限制

Kerberos 和 SSL 可以与 Adaptive Server 结合使用；SQL Anywhere 当前不支持 SSL 或 Kerberos 安全机制。

Sybase 建议在 jConnect 中使用 SSL 或 Kerberos 前先阅读其相关文档。本章中的信息假定要使用的服务器已经进行配置，可正常使用 SSL、Kerberos 或上述两者。

有关 Kerberos、SSL 和配置 Adaptive Server Enterprise 的详细信息，请参见第 124 页的 “相关文档”。还请参见有关设置 Kerberos 的白皮书。该文档的 URL 可在 jConnect for JDBC Release Bulletin （《jConnect for JDBC 发行公告》）中找到。

实现自定义套接字插件

本节讨论如何将自定义套接字实现插入应用程序以自定义客户端和服务器间的通信。 javax.net.ssl.SSLSocket 是一个套接字示例，可以通过自定义使其实现加密功能。

com.sybase.jdbcx.SybSocketFactory 是包含返回 java.net.Socket 的 createSocket(String, int, Properties) 方法的 Sybase 扩展接口。若要在 jConnect 中使用自定义套接字工厂，应用程序必须通过定义 createSocket() 方法来实现此接口。

jConnect 使用新套接字完成其后续输入/输出操作。实现 SybSocketFactory 的类将创建套接字并提供一般框架以添加公共套接字级功能，如下所示：

```
/**
 * Returns a socket connected to a ServerSocket on the named host,
 * at the given port.
 * @param host  the server host
 * @param port  the server port
 * @param props  Properties passed in through the connection
 * @returns Socket
 * @exception IOException, UnknownHostException
 */
public java.net.Socket createSocket(String host, int port, Properties props)
   throws IOException, UnknownHostException;
```

传入属性将允许 SybSocketFactory 的实例使用连接属性实现智能套接字。

在实现 SybSocketFactory 以产生套接字时，通过向应用程序传递创建套接字的不同种类的工厂或伪工厂可以使相同的应用程序代码使用不同种类的套接字。

可以使用在套接字结构中使用的参数自定义工厂。例如，可以使用已配置的不同的网络超时或安全参数自定义工厂以返回套接字。返回到应用程序的套接字可以是 java.net.Socket 的子类，以直接公开一些功能（如压缩、安全性、记录标记、统计信息收集或防火墙贯通）的新 API (javax.net.SocketFactory)。

注释 SybSocketFactory 是过度简化的 javax.net.SocketFactory，可以使应用程序从 java.net.* 过渡到 javax.net.*

❖ **在 jConnect 中使用自定义套接字**

1 提供一个实现 com.sybase.jdbcx.SybSocketFactory 的 Java 类。请参见第 109 页的 “创建和配置自定义套接字”。

2 设置 SYBSOCKET_FACTORY 连接属性以使 jConnect 可以使用您的实现获取套接字。

若要在 jConnect 中使用自定义套接字，请将 SYBSOCKET_FACTORY 连接属性设置为以下内容之一：

- 实现 com.sybase.jdbcx.SybSocketFactory 的类名称
- DEFAULT （这将实例化新的 java.net.Socket）

有关如何设置 SYBSOCKET_FACTORY 的说明，请参见第 9 页的 “连接属性”。

创建和配置自定义套接字

jConnect 获得自定义套接字后将使用该套接字连接到服务器。对套接字的任何配置都必须在 jConnect 获得该套接字之前完成。

本节介绍如何使用 jConnect 插入 SSL 套接字实现（如 javax.net.ssl.SSLSocket）。

以下示例显示 SSL 的实现如何创建、配置和返回 SSLSocket 实例。在该示例中， MySSLSocketFactory 类实现 SybSocketFactory 并扩展 javax.net.ssl.SSLSocketFactory 以实现 SSL。它包含两个 createSocket 方法：一个用于 SSLSocketFactory，另一个用于 SybSocketFactory，功能如下：

- 创建 SSL 套接字

- 调用 SSLSocket.setEnableCipherSuites 以指定可用于加密的密码成套程序
- 返回 jConnect 要使用的套接字

示例

```
public class MySSLSocketFactory extends SSLSocketFactory
   implements SybSocketFactory
 {
 /**
 * Create a socket, set the cipher suites it can use, return
 * the socket.
 * Demonstrates how cither suites could be hard-coded into the
 * implementation.
 *
 * See javax.net.SSLSocketFactory#createSocket
 */
public Socket createSocket(String host, int port)
   throws IOException, UnknownHostException
 {
   // Prepare an array containing the cipher suites that are to
   // be enabled.
   String enableThese[] =
   {
       "SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA",
       "SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5",
       "SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA"
   }
   ;
   Socket s =
       SSLSocketFactory.getDefault().createSocket(host, port);
   ((SSLSocket)s).setEnabledCipherSuites(enableThese);
   return s;
 }
/**
 * Return an SSLSocket.
 * Demonstrates how to set cipher suites based on connection
 * properties like:
 * Properties _props = new Properties();
 * Set other url, password, etc. properties.
 * _props.put(("CIPHER_SUITES_1",
 *    "SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA");
 * _props.put("CIPHER_SUITES_2",
 *    "SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5");
 * _props.put("CIPHER_SUITES_3",
```

```
 *     "SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA");
 * _conn = _driver.getConnection(url, _props);
 *
 * See com.sybase.jdbcx.SybSocketFactory#createSocket
 */
public Socket createSocket(String host, int port,
   Properties props)
   throws IOException, UnknownHostException
 {
   // check to see if cipher suites are set in the connection
   // properites
   Vector cipherSuites = new Vector();
   String cipherSuiteVal = null;
   int cipherIndex = 1;
   do
   {
       if((cipherSuiteVal = props.getProperty("CIPHER_SUITES_"
           + cipherIndex++)) == null)
       {
           if(cipherIndex <= 2)
           {
               // No cipher suites available
               // return what the object considers its default
               // SSLSocket, with cipher suites enabled.
               return createSocket(host, port);
           }
           else
           {
               // we have at least one cipher suite to enable
               // per request on the connection
               break;
           }
           else
           }
               // add to the cipher suit Vector, so that
               // we may enable them together
               cipherSuites.addElement(cipherSuiteVal);
           }
       }
       while(true);
      // lets you create a String[] out of the created vector
       String enableThese[] = new String[cipherSuites.size()];
       cipherSuites.copyInto(enableThese);
       Socket s =
           SSLSocketFactory.getDefault().createSocket
             (host, port);
```

```
       // enable the cipher suites
       ((SSLSocket)s).setEnabledCipherSuites(enableThese);
      // return the SSLSocket
       return s;
   }
  // other methods
}
```

由于 jConnect 不需要套接字的种类信息，因此必须在返回套接字之前完成所有配置。

有关详细信息，请参见：

- *EncryptASE.java* – 位于 jConnect 安装的 *sample2* 子目录下，此示例显示如何在 jConnect 应用程序中使用 SybSocketFactory 接口。
- *MySSLSocketFactoryASE.java* – 也位于 jConnect 安装的 *sample2* 子目录下，这是一个 SybSocketFactory 接口的实现示例，可以插入应用程序并使用。

Kerberos

Kerberos 是一种网络鉴定协议，对客户端/服务器应用程序的鉴定使用加密。 Kerberos 为用户和系统管理员带来了以下优点：

- Kerberos 数据库可用作用户的集中仓库。
- Kerberos 便于建立 SSO 环境，在此环境中用户系统登录可提供访问数据库所必需的凭据。
- Kerberos 是一种 IETF 标准。Kerberos 的不同实现间的互操作是可能的。

为 Kerberos 配置 jConnect 应用程序

尝试为 jConnect 配置 Kerberos 前，应确保拥有以下组件：

- JDK 6 或更高版本
- Java 通用安全服务 (GSS) Manager：
  a 缺省 GSS Manager （JDK 的一部分），或
  b Wedgetail JCSI Kerberos 2.6 或更高版本，或
  c CyberSafe TrustBroker Application Security Runtime Library 3.1.0 或更高版本，或

d　来自其它供应商的 GSS Manager 实施。

- 在服务器端得到支持，可与 GSS 库互操作，同时在客户端也得到支持，可与 GSSManager 互操作的 KDC。

若要启用 jConnect 的 Kerberos 登录，请使用以下步骤。

❖ **为 jConnect 配置 Kerberos**

1　将 REQUEST_KERBEROS_SESSION 属性设置为 “true”。

2　将 SERVICE_PRINCIPAL_NAME 属性设置为正在运行的 Adaptive Server Enterprise 的名称。通常就是服务器启动时用 -s 选项设置的名称。服务主体名称也必须用 KDC 注册。如果不为 SERVICE_PRINCIPAL_NAME 属性设置任何值，缺省情况下 jConnect 会使用客户端计算机的主机名。

3　也可以选择设置 GSSMANAGER_CLASS 属性。

有关 REQUEST_KERBEROS_SESSION 和 SERVICE_PRINCIPAL_NAME 属性的详细信息，请参见第 2 章 “编程信息”有关 GSSMANAGER_CLASS 属性的详细信息，请参见“GSSMANAGER_CLASS 连接属性”

GSSMANAGER_CLASS 连接属性

使用 Kerberos 时，jConnect 依赖于实现通用安全服务 (GSS) API 的几个 Java 类。这个功能的大部分都是由 org.ietf.jgss.GSSManager 类提供的。

供应商实现

Java 允许供应商提供其自己的 GSSManager 类实施。例如， Wedgetail Communications 和 CyberSafe Limited 提供的实施即属于供应商提供的 GSSManager 实施。用户可配置供应商编写的 GSSManager 类，以使其在特定的 Kerberos 环境中工作。供应商提供的 GSSManager 类提供的与 Windows 的互操作性可能比标准的 Java GSSManager 类提供的还要多。

在使用供应商提供的 GSSManager 实现前，一定要阅读供应商文档。供应商使用属性设置而不是用于 Kerberos 的标准 Java 系统属性，而且可能会定位领域名和密钥分发中心 (KDC) 条目而不使用配置文件。

设置 GSSMANAGER_CLASS

通过设置 GSSMANAGER_CLASS 连接属性可以在 jConnect 中使用供应商提供的 GSSManager 实现。有两种设置此属性的方法：

- 创建一个 GSSManager 实例，并将此实例设置为 GSSMANAGER_CLASS 属性的值。
- 将 GSSMANAGER_CLASS 属性的值设置为指定 GSSManager 对象的完全限定类名称的字符串。 jConnect 使用该字符串调用 `Class.forName().newInstance()`，并将返回的对象转换为 GSSManager 类。

在任一情况下，应用程序 CLASSPATH 变量必须包含供应商实现的类和 *.jar* 文件的位置。

注释 如果不设置 GSSMANAGER_CLASS 连接属性， jConnect 将使用 org.ietf.jgss.GSSManager.getInstance 方法装载缺省 Java GSSManager 实现。

使用 GSSMANAGER_CLASS 连接属性传递全限定类名时， jConnect 将调用 GSSManager 的不带参数的构造方法。此处例示了供应商实现的一个处于缺省配置的 GSSManager，因此您无法控制 GSSManager 对象的精确配置。如果创建自己的 GSSManager 实例，就可以使用构造方法参数来设置配置选项。

jConnect 使用 GSSMANAGER_CLASS 的方式

首先， jConnect 检查要用于 Kerberos 鉴定的 GSSManager 类对象的 GSSMANAGER_CLASS 的值。

如果已将 GSSMANAGER_CLASS 的值设置为字符串而不是类对象，则 jConnect 将使用该字符串创建指定类的实例，并在 Kerberos 鉴定中使用新实例。

如果已将 GSSMANAGER_CLASS 的值设置为 GSSManager 类对象和字符串以外的其它内容，或如果 jConnect 遇到 ClassCastException，则 jConnect 将抛出指明问题的 SQLException。

示例

以下示例说明了当 GSSMANAGER_CLASS 连接属性设置为完全限定类名称时，如何创建自己的 GSSManager 实例，以及如何让 jConnect 创建 GSSManager 对象。两个示例都使用了 Wedgetail GSSManager。

❖ **示例：创建自己的 GSSManager 实例**

1 在应用程序代码中实例化 GSSManager。例如：

```
GSSManager gssMan =  new com.dstc.security.kerberos.gssapi.GSSManager();
```

此示例使用了不带参数的缺省构造方法。也可以使用允许设置各种配置选项的其它供应商提供的构造方法。

2 将新 GSSManager 实例传递给 GSSMANAGER_CLASS 连接属性。例如：

```
Properties props = new Properties();
props.put("GSSMANAGER_CLASS", gssMan);
```

3 在连接中使用这些连接属性（包括 GSSMANAGER_CLASS）。例如：

```
Connection conn = DriverManager.getConnection (url, props);
```

❖ **示例：将字符串传递给 GSSMANAGER_CLASS**

1 在应用程序代码中，创建一个指定 GSSManager 对象的完全限定类名称的字符串。例如：

```
String gssManClass = "com.dstc.security.kerberos.gssapi.GSSManager";
```

2 将该字符串传递给 GSSMANAGER_CLASS 连接属性。例如：

```
Properties props = new Properties();
props.put("GSSMANAGER_CLASS", gssManClass);
```

3 在连接中使用这些连接属性（包括 GSSMANAGER_CLASS）。例如，

```
Connection conn = DriverManager.getConnection (url, props);
```

设置 Kerberos 环境

本节提供了一些设置环境的建议，以便在三种不同的 Kerberos 实现中使用 jConnect：

- CyberSafe
- MIT
- Microsoft Active Directory

注释 在阅读本节之前，请参见Kerberos white paper (http://www.sybase.com/detail?id=1029260)

CyberSafe

加密密钥

在 CyberSafe KDC 中创建要由 Java 使用的主体时应指定数据加密标准 (DES) 密钥。 Java 参考实现不支持三倍数据加密标准 (3DES) 密钥。

注释 如果是在 CyberSafe KDC 中使用 CyberSafe 且设置了 GSSMANAGER_CLASS 属性，就可以使用 3DES 密钥。

地址映射和领域信息

CyberSafe Kerberos 不使用 *krb5.conf* 配置文件。缺省情况下， CyberSafe 使用 DNS 记录来定位 KDC 地址映射和领域信息。或者， CyberSafe 分别在 *krb.conf* 和 *krb.realms* 文件中定位 KDC 地址映射和领域信息。有关详细信息，请参见 CyberSafe 文档。

如果使用的是标准 Java GSSManager 实现，还必须创建 *krb5.conf* 文件以供 Java 使用。 CyberSafe *krb.conf* 文件的格式与 *krb5.conf* 文件的格式不同。根据 Java SE 文档或 MIT 文档中指定的内容创建 *krb5.conf* 文件。如果正在使用 CyberSafe GSSManager，就不需要 *krb5.conf* 文件。

有关 *krb5.conf* 文件的示例，请参见有关设置 Kerberos 的白皮书。该文档的 URL 可在 jConnect for JDBC Release Bulletin （《jConnect for JDBC 发行公告》）中找到。

Solaris

在 Solaris 上使用 CyberSafe 客户端库时，应确保库搜索路径在任何其它 Kerberos 库前包含 CyberSafe 库。

MIT

加密密钥

创建 MIT KDC 中要由 Java 使用的主体时应指定 DES 密钥。Java 参考实现不支持 3DES 密钥。

如果计划只使用标准 Java GSSManager 实现，应指定 `des-cbc-crc` 或 `des-cbc-md5` 类型的加密密钥。按如下方法指定加密类型：

```
des-cbc-crc:normal
```

这里 `normal` 是密钥 salt 的类型。也可以使用其它 salt 类型。

注释 如果正在使用 Wedgetail GSSManager，则可以在 MIT KDC 中创建类型为 `des3-cbc-sha1-kd` 的主体。

Microsoft Active Directory

用户帐户和服务主体

确保已在 Active Directory 中为用户主体 （用户）和服务主体 （代表数据库服务器的帐户）设置了帐户。用户主体和服务主体都应在 Active Directory 中创建为 “Users”。

加密

如果要使用 Java 参考 GSS Manager 实现，就必须对用户和服务主体使用 DES 加密。

❖ **设置 DES 加密**

1. 在 Active Directory “用户”列表中右击特定用户主体或服务主体的名称。
2. 选择 “属性”。
3. 单击 “帐户” (Account) 选项卡。将出现 “帐户选项” (Account Options) 列表。
4. 为用户主体和服务主体指定应使用的 DES 加密类型。

客户端计算机

如果计划使用 Java 参考实现设置 SSO 环境，可能需要根据Microsoft support site (http://support.microsoft.com/)中提供的说明修改 Windows 注册表。

配置文件

在 Windows 上， Kerberos 配置文件称为 *krb5.ini*。缺省情况下， Java 会在 *C:\WINNT\krb5.ini* 中查找 *krb5.ini*。 Java 允许指定该文件的位置。*krb5.ini* 的格式与 *krb5.conf* 的格式相同。

有关 *krb5.conf* 文件的示例，请参见有关设置 Kerberos 的白皮书。该文档的 URL 可在 jConnect for JDBC Release Bulletin （《jConnect for JDBC 发行公告》）中找到。

有关 Microsoft Active Directory 的 Kerberos 的详细信息，请参见 Microsoft Developer Network (http://msdn.microsoft.com)。

示例应用程序

在 *jConnect-7_0/sample2* 目录中提供了以下两个注释的代码示例，说明了如何建立与 Adaptive Server Enterprise 的 Kerberos 连接：

- *ConnectKerberos.java* – 登录 Adaptive Server Enterprise 的简单 Kerberos 示例
- *ConnectKerberosJAAS.java* – 更详细的示例，显示了如何在应用程序/服务器代码中实现 Kerberos 登录

ConnectKerberos.java

若要运行 *ConnectKerberos.java* 示例应用程序，请使用以下步骤。

❖ **运行 *ConnectKerberos.java***

1 确保计算机具有有效 Kerberos 凭据。该任务根据计算机和环境的不同而不同。

Windows – 通过使用 Kerberos 鉴定成功登录，可以确定 Active Directory 环境中的计算机具有 Kerberos 凭据。

UNIX 或 Linux – 可以使用 Kerberos 客户端的 kinit 实用程序为 UNIX 或 Linux 计算机确定 Kerberos 凭据。如果未使用 kinit 获取初始认证，在试图运行示例应用程序时，系统会提示您输入用户名和口令。

注释 一般情况下，由标准 JDK 提供的 GSSManager 提供程序实施只能使用 DES_CBC_MD5 和 DES_CBC_CRC 加密类型。通过使用第三方软件和设置 GSSMANAGER_CLASS 可能可以使用其它加密类型。

2 确定计算机认证的位置。

Windows – 对于运行在 Active Directory 环境中的计算机， Kerberos 凭据存储在内存中的票据高速缓存中。

UNIX 或 Linux – 对于使用 Kerberos 的 JRE、 CyberSafe、 Solaris 或 MIT 实施的 UNIX 或 Linux 计算机，缺省情况下， kinit 将凭据放在 */tmp/krb5cc_{user_id_number}* 中，其中 *{user_id_number}* 对于您的用户名来说是唯一的。

如果凭据位于其它位置，必须通过设置 ticketCache 属性在 *sample2/exampleLogin.conf* 文件中指定该位置。

3　向 Java 参考实现指定 KDC 计算机的缺省领域和主机名。Java 可从 *krb5.conf* 或 *krb5.ini* 配置文件或从 Java System 属性获取该信息。如果使用供应商提供的 GSS Manager 实现，则此实现可能从 DNS SRV 记录获取主机及领域信息。

Sybase 推荐使用 Kerberos 配置文件，它允许对 Kerberos 环境进行更多控制，包括向 Java 指定鉴定期间请求加密的类型的能力。

注释 在 Linux 上，Java 参考实现在 */etc/krb5.conf* 中查找 Kerberos 配置文件。

如果不使用 Kerberos 配置文件，且未将 Kerberos 配置设置为使用 DNS SRV 记录，就可以使用 java.security.krb5.realm 和 java.security.krb5.kdc 系统属性指定领域和 KDC。

4　编辑 *ConnectKerberos.java* 以使连接 URL 指向您的数据库。

5　编译 *ConnectKerberos.java*。

确保您使用的是 JDK 版本 6 或更高版本。通览源代码注释，并确保已在 CLASSPATH 环境变量中指定了 jConnect 安装中的 *jconn4.jar*。

6　执行 ConnectKerberos.class：

```
java ConnectKerberos
```

确保您使用的是 java 版本 6 可执行文件。示例应用程序输出说明已成功建立连接并执行下面的 SQL：

```
select 1
```

- 若要在不使用 *Kerberos* 配置文件的情况下执行示例，请使用以下命令：

  ```
  java -Djava.security.krb5.realm=your_realm
  -Djava.security.krb5.kdc=your_kdc
  ConnectKerberos
  ```

  其中， *your_realm* 是缺省领域，而 *your_kdc* 是您的 KDC。

- 如有必要，可在调试模式下运行示例应用程序，以查看 Java Kerberos 层的调试输出：

  ```
  java -Dsun.security.krb5.debug=true
  ConnectKerberos
  ```

也可以使用位于 *jConnect-7_0/classes* 目录的 IsqlApp （isql 的 Java 版本）来建立 Kerberos 连接：

```
java IsqlApp -S jdbc:sybase:Tds:hostName:portNum
-K service_principal_name
-F path_to_JAAS_login_module_config_file
```

有关使用 IsqlApp 的详细信息，请参见

krb5.conf 配置文件

下面是 *krb5.conf* 文件的示例。

CyberSafe 或 MIT KDC

这是客户端可能用于 CyberSafe 或 MIT KDC 的 *krb5.conf* 文件示例。

```
# Please note that customers must alter the
# default_realm, [realms] and [doamin_realm]
# information to reflect their Kerberos environment.
# Customers should *not* attempt to use this file as is.
#

[libdefaults]
        default_realm = ASE
        default_tgs_enctypes = des-cbc-crc
        default_tkt_enctypes = des-cbc-crc
        kdc_req_checksum_type = 2
        ccache_type = 2

[realms]

        ASE = {
            kdc = kdchost
            admin_server = kdchost
        }

[domain_realm]
        .sybase.com = ASE
        sybase.com = ASE

[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log
 kdc_rotate = {
```

```
# How often to rotate kdc.log. Logs will get rotated
# no more often than the period, and less often if the
# KDC is not used frequently.

  period = 1d

# how many versions of kdc.log to keep around
# (kdc.log.0, kdc.log.1, ...)

  versions = 10
 }

[appdefaults]
 kinit = {
  renewable = true
  forwardable= true
 }
```

Active Directory KDC

这是客户端可能将其作为 KDC 用于 Active Directory 的 *krb5.conf* 文件示例。

```
# Please note that customers must alter the
# default_realm, [realms] and [domain_realm]
# information to reflect their Kerberos environment.
# Customers should *not* attempt to use this file as is.
#

[libdefaults]
        default_realm = W2K.SYBASE.COM
        default_tgs_enctypes = des-cbc-crc
        default_tkt_enctypes = des-cbc-crc
        kdc_req_checksum_type = 2
        ccache_type = 2

[realms]

        W2K.SYBASE.COM = {
           kdc = 1.2.3.4:88
           admin_server = adserver
         }

[domain_realm]
        .sybase.com = W2K.SYBASE.COM
```

```
        sybase.com = W2K.SYBASE.COM

[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log
 kdc_rotate = {

# How often to rotate kdc.log. Logs will get rotated no
# more often than the period, and less often if the KDC
# is not used frequently.

  period = 1d

# how many versions of kdc.log to keep around
# (kdc.log.0, kdc.log.1, ...)

  versions = 10
 }

[appdefaults]
 kinit = {
  renewable = true
  forwardable= true
 }
```

互操作性

表 3-1 显示了 KDC、GSS 库和平台的组合，Sybase 在这些平台上成功地建立了到 Adaptive Server Enterprise 的连接。缺少任何特定组合并不表示不能与该组合建立连接。可在jConnect for JDBC Web site (http://www.sybase.com/products/allproductsa-z/softwaredeveloperkit/jconnect)中找到最新状态：

表 3-1：互操作性组合

客户端平台	KDC	GSSManager	GSS C 库 [a]	ASE 平台
Solaris 8[b]	CyberSafe	Java GSS	CyberSafe	Solaris 8
Solaris 8	Active Directory[c]	Java GSS	CyberSafe	Solaris 8
Solaris 8	MIT	Java GSS	CyberSafe	Solaris 8
Solaris 8	MIT	Wedgetail GSS[d]	MIT	Solaris 8
Solaris 8	CyberSafe	Wedgetail GSS[e]	CyberSafe	Solaris 8
Windows 2000	Active Directory	Java GSS	CyberSafe	Solaris 8
Windows XP	Active Directory	Java GSS[f]	CyberSafe	Solaris 8

客户端平台	KDC	GSSManager	GSS C 库 [a]	ASE 平台

a. 这些是 Adaptive Server Enterprise 用于提供 GSS 功能的库。
b. 表中所有 Solaris 8 平台均为 32 位。
c. 表中的所有 Active Directory 条目均指运行于 Windows 2000 上的 Active Directory 服务器。若要获得 Kerberos 互操作性， Active Directory 用户必须设置为 “为此帐户使用 DES 加密类型”。
d. 使用 Wedgetail JCSI Kerberos 2.6。加密类型为 3DES。
e. 使用 Wedgetail JCSI Kerberos 2.6。加密类型为 DES。
f. Java 1.4.x 有一个错误，它要求客户端使用 `System.setProperty("os.name", "Windows 2000");` 来确保 Java 可在 Windows XP 客户端找到内存中的认证。

Sybase 建议使用这些库的最新版本。如果想要使用旧版本，或者非 Sybase 产品有问题，请联系供应商。

加密类型

典型 JRE 提供的标准 Java GSS 实施仅支持 DES 加密。若要使用 3DES、RC4-HMAC、AES-256 或 AES-128 加密标准，就必须使用 CyberSafe 或 Wedgetail GSSManager。

有关 Wedgetail 和 CyberSafe 的详细信息，请参见各自的文档。

故障排除

本节提供进行 Kerberos 安全性故障排除时要考虑的问题的文档资料。

Kerberos

进行 Kerberos 安全性问题故障排除时要考虑以下内容：

- Java 参考实现仅支持 DES 加密类型。必须配置 Active Directory 和 KDC 主体使用 DES 加密。
- SERVICE_PRINCIPAL_NAME 属性的值必须设置为启动数据服务器时用 -s 选项指定的同一名称。
- 检查 *krb5.conf* 和 *krb5.ini* 文件。对于 CyberSafe 客户端，检查 *krb.conf* 和 *krb.realms* 文件或 DNS SRV 记录。
- 可在 JAAS 登录配置文件中将 debug 属性设置为 “true”。
- 可在命令行中将 debug 属性设置为 “true”：

  ```
  -Dsun.security.krb5.debug=true
  ```

- JAAS 登录配置文件提供了多个可进行设置的选项，以满足各种特殊需要。有关 JAAS 和 Java GSS API 的信息，请参考：

- JAAS login configuration file (http://docs.oracle.com/javase/1.4.2/docs/guide/security/jgss/tutorials/LoginConfigFile.html)
- Class Krb5LoginModule (http://docs.oracle.com/javase/1.4.2/docs/guide/security/jaas/spec/com/sun/security/auth/module/Krb5LoginModule.html)
- Troubleshooting JGSS (http://docs.oracle.com/javase/1.4.2/docs/guide/security/jgss/tutorials/Troubleshooting.html)

相关文档

以下文档提供了有关 Kerberos 安全性的详细信息。

- Java tutorial on JAAS and the Java GSS API (http://docs.oracle.com/javase/1.4.2/docs/guide/security/jgss/tutorials/index.html)
- MIT Kerberos documentation and download site (http://web.mit.edu/kerberos/www/index.html)
- CyberSafe Limited (http://www.cybersafe.ltd.uk)
- CyberSafe Limited document on Windows-Kerberos interoperability (http://www.cybersafe.ltd.uk/docs_cybersafe/Kerberos%20Interoperability%20-%20Microsoft%20W2k%20&%20ActiveTRUST.pdf)
- Description of how Windows implements authentication, including information about Active Directory Kerberos (http://www.windowsitlibrary.com/Content/617/06/1.html)
- Kerberos RFC 1510 (http://www.linuxdig.com/rfc/individual/1510.php)

第 4 章

故障排除

本章描述了对 jConnect 使用过程中可能出现问题的解决方案。

使用 jConnect 进行调试

jConnect 包括一个 Debug 类，该类包含一组调试函数。 Debug 方法包括多个断言函数、跟踪函数和计时器函数，用于定义调试过程的范围以及调试结果的输出位置。

jConnect 安装程序还包括一组完整的具有调试功能的类。这些类位于 jConnect 安装目录的 *devclasses* 子目录下。为了进行调试，必须重定向 CLASSPATH 环境变量以引用调试模式运行期类 (*devclasses/jconn4d.jar*)，而不是引用标准的 jConnect *classes* 目录。也可以在运行 Java 程序时，通过将 -classpath 参数显式提供给 java 命令来实现。

获取 Debug 类的一个实例

若要使用 jConnect 调试功能，应用程序必须导入 Debug 接口并通过调用 SybDriver 类的 getDebug 方法获取 Debug 类的一个实例。

```
import com.sybase.jdbcx.Debug;
//
...
SybDriver sybDriver = (SybDriver)
```

```
Class.forName("com.sybase.jdbc4.jdbc.SybDriver").newInstance();
Debug sybdebug = sybDriver.getDebug();
...
```

在应用程序中打开调试程序

若要使用 Debug 对象的 debug 方法打开应用程序中的调试程序，请添加以下调用：

```
sybdebug.debug(true, [classes], [printstream]);
```

classes 参数是一个字符串，其中列出了要调试的特定类（用冒号分隔各个类）。例如：

```
sybdebug.debug(true,"MyClass")
```

and

```
sybdebug.debug(true,"MyClass:YourClass")
```

在类字符串中使用 “STATIC”将会为 jConnect 中的所有静态方法以及指定的类打开调试程序。例如：

```
sybdebug.debug(true,"STATIC:MyClass")
```

可以指定 “ALL”为所有类打开调试程序。例如：

```
sybdebug.debug(true,"ALL");
```

printstream 参数是可选的。如果未指定 printstream 参数，调试输出将传送到 DriverManager.setLogStream 所指定的输出文件中。

在应用程序中关闭调试程序

若要关闭调试程序，请添加以下调用：

```
sybdebug.debug(false);
```

为调试程序设置 CLASSPATH

在运行启用了调试的应用程序之前，请将经过优化的 jConnect 驱动程序 jar 文件 *jconn4.jar* 替换为调试版本 *jconn4d.jar*，后者位于 jConnect 安装目录下的 *devclasses* 子目录中。

如果使用 CLASSPATH 环境变量：

- 在 UNIX 系统中，将 *$JDBC_HOME/classes/jconn4.jar* 替换为 *$JDBC_HOME/devclasses/jconn4d.jar*。
- 在 Windows 系统中，将 *%JDBC_HOME%\classes\jconn4.jar* 替换为 *%JDBC_HOME%\devclasses\jconn4d.jar*。

使用 Debug 方法

若要自定义调试过程，需添加对其它 Debug 方法的调用。

在这些方法中，第一个 （对象）参数通常是 this，用以指定调用对象。如果这些方法都是静态的，对象参数需使用 null。

- println

  如果已启用调试程序并且对象包含在要调试的类列表中，请使用该方法定义要在输出日志中输出的消息。调试输出将传送到 *sybdebug*.debug 所指定的文件中。

  语法为：

  ```
  sybdebug.println(object,message string);
  ```

  例如：

  ```
  sybdebug.println(this,"Query:"+ query);
  ```

  在输出日志中产生类似如下的消息：

  ```
  myApp(thread[x,y,z]):Query:select * from authors
  ```

- assert

  使用该方法声明条件，并在不满足该条件时抛出一个运行期例外。也可以定义在不满足条件时将消息输出到输出日志中。语法为：

  ```
  sybdebug.assert(object,boolean condition,message
     string);
  ```

  例如：

  ```
  sybdebug.assert(this,amount<=buf.length,amount+"
     too big!");
  ```

  如果 “amount”超出了 buf.length 的值，就会在输出日志中产生类似如下的消息：

  ```
  java.lang.RuntimeException:myApp(thread[x,y,z]):
  Assertion failed:513 too big!
  at jdbc.sybase.utils.sybdebug.assert(
  sybdebug.java:338)
  at myApp.myCall(myApp.java:xxx)
  at .... more stack:
  ```

- startTimer
  stopTimer

  使用这些方法启动和停止计时器，计时器用于测量事件所占用的时间（以毫秒计）。该方法为每个对象保留一个计时器，并为所有静态方法保留一个计时器。启动计时器的语法为：

  ```
  sybdebug.startTimer(object);
  ```

  停止计时器的语法为：

  ```
  sybdebug.stopTimer(object,message string);
  ```

  例如：

  ```
  sybdebug.startTimer(this);
  stmt.executeQuery(query);
  sybdebug.stopTimer(this,"executeQuery");
  ```

  在输出日志中产生类似如下的消息：

  ```
  myApp(thread[x,y,z]):executeQuery elapsed time =
    25ms
  ```

动态日志记录

自 15.7 ESD #4 开始， jConnect for JDBC 通过实施标准 Java Logger 机制支持日志记录机制。现在，应用程序可控制 jConnect 记录程序并根据需要启动或关闭日志记录。

示例

```
try
{
// Get logger for all classes present in
//"com.sybase.jdbc4.jdbc" package

Logger LOG = Logger.getLogger("com.sybase.jdbc4.jdbc");

// To log class-specific log message,
// provide complete class name, for example:
//Logger.getLogger("com.sybase.jdbc4.jdbc.
//SybConnection");
//Get handle as per user's requirement
Handler handler = new ConsoleHandler();

//Set logging level
handler.setLevel(Level.ALL);

//Added user specific handler to logger object
LOG.addHandler(handler);

//Set logging level
LOG.setLevel(Level.ALL);

Class.forName("com.sybase.jdbc4.jdbc.SybDriver");
Properties properties = new Properties();
properties.put("USER", USER_NAME);
properties.put("PASSWORD", PASSWORD);
Connection con =
  DriverManager.getConnection("jdbc:sybase:Tds:"+
    HOST_PORT, properties);
              Statement stmt = con.createStatement();
stmt.execute("select @@version");

//Dynamically turn off logging mechanism
LOG.setLevel(Level.OFF);
con.close();
...
}
```

注释 目前，只有公共 API 会通过 Level.FINE 进行记录；没有任何语句以 Level.INFO、 Level.FINER 和 Level.FINEST 记录。

捕获 TDS 通信

Tabular Data Stream (TDS) 是用于处理客户端应用程序和 Adaptive Server 之间通信的 Sybase 专有协议。 jConnect 包含 PROTOCOL_CAPTURE 连接属性，该属性可以将多个原始 TDS 包捕获到一个文件中。

如果应用程序出现故障，而且无法在应用程序或服务器内部加以解决，则可使用 PROTOCOL_CAPTURE 捕获客户端和服务器之间的通信，并将其存放到一个文件中。该文件包含二进制数据，不能直接解读。您可以将该文件发送给 Sybase 技术支持部门进行分析。

注释 捕获并保存到文件中的 TDS 协议数据包含敏感的用户鉴定信息，也可能含有公司或客户数据等机密信息。要保护此机密数据使其免遭非授权或意外泄露，必须使用文件权限或加密对含有捕获数据的文件加以正确保护。

PROTOCOL_CAPTURE 连接属性

使用 PROTOCOL_CAPTURE 连接属性指定一个文件，用于接收应用程序和 Adaptive Server 之间交换的 TDS 包。 PROTOCOL_CAPTURE 会立即生效，这样在建立连接过程中交换的 TDS 包就会被写入到指定的文件中。所有的 TDS 包将继续被写入到该文件中，直到执行 Capture.pause 或关闭该会话为止。

以下示例显示如何使用 PROTOCOL_CAPTURE 将 TDS 数据发送到文件 *tds_data* 中：

```
...
props.put("PROTOCOL_CAPTURE", "tds_data")
Connection conn = DriverManager.getConnection(url,
props);
```

其中，url 是连接的 URL，props 是用于指定连接属性的 Properties 对象。

Capture 类中的 *pause* 和 *resume* 方法

Capture 类包含在 com.sybase.jdbcx 包中。该类包含两个公共方法：

- public void pause
- public void resume

Capture.pause 停止将原始 TDS 包捕获到文件中；Capture.resume 重新开始捕获。

整个会话的 TDS 捕获文件可能会变得很大。如果要限制捕获文件的大小，并且知道要捕获的 TDS 数据在应用程序中的位置，则可执行如下操作。

❖ **限制捕获文件的大小**

1 在建立连接后，立即获取用于该连接的 Capture 对象，并使用 pause 方法停止捕获 TDS 数据：

```
Capture cap = ((SybConnection)conn).getCapture();
 cap.pause();
```

2 在要开始捕获 TDS 数据的位置前放置 cap.resume。

3 在要停止捕获 TDS 数据的位置后放置 cap.pause。

解决连接错误

本节介绍如何解决在试图建立连接或启用网关时出现的问题。

网关连接被拒绝

网关连接被拒绝：
`HTTP/1.0 502` 网关错误 | 重新启动连接

该错误消息表明，用于连接到 Adaptive Server 的 *hostname* 或 *port#* 出错。检查 *$SYBASE/interfaces* (UNIX) 或 *%SYBASE%\ini\sql.ini* (Windows) 中的 [query] 项。

如果在检验了 *hostname* 和 *port#* 之后问题仍然存在，请使用 “verbose” 系统属性启动 HTTP 服务器以进一步了解相关信息。

在 Windows 系统中，转至 DOS 提示符并输入：

```
httpd -Dverbose=1 > filename
```

在 UNIX 系统中，输入：

```
sh httpd.sh -Dverbose=1 > filename &
```

其中， *filename* 是调试消息输出文件。

您的 Web 服务器可能不支持 connect 方法。小程序仅能够连接到可下载这些小程序的主机。

HTTP 网关和 Web 服务器必须在同一主机上运行。在这种情况下，小程序可以通过 HTTP 网关 （HTTP 网关能够将请求路由到相关数据库）控制的端口连接到同一主机。

若要查看该过程是如何实现的，请查阅位于 jConnect 安装目录下的 *sample2* 子目录中的 *Isql.java* 和 *gateway.html* 源文件。搜索 “proxy”。

管理 jConnect 应用程序所使用的内存

以下情形及其解决方案可能有助于解决 jConnect 应用程序使用越来越多内存的问题。

- 在 jConnect 应用程序中，应当显式关闭最近一次使用的所有 Statement 对象以及子类 （例如， PreparedStatement、CallableStatement）以阻止语句累积在内存中。仅关闭 ResultSet 是不够的。

  例如，下面的语句会导致出现问题：

  ```
  ResultSet rs = _conn.prepareCall(_query).execute();
  ...
  rs.close();
  ```

  而应当使用如下语句：

  ```
  PreparedStatement ps = _conn.prepareCall(_query);
  ResultSet rs = ps.executeQuery();
  ...
  rs.close();
  ps.close();
  ```

- 根据所连接到的 Adaptive Server 或 SQL Anywhere 数据库版本的不同，可能不存在对可滚动游标或可更新式可滚动游标的内在支持。为了在后端服务器原本不支持可滚动游标或可更新式可滚动游标的情况下对这些游标提供支持， jConnect 会根据需要，在每次调用 ResultSet.next 时在客户端上高速缓存行数据。但到达结果集的末尾时，整个结果集将存储到客户端内存中。由于这可能会导致性能降低，因此， Sybase 建议仅当结果集相当小的时候，才使用 TYPE_SCROLL_INSENSITIVE 结果集。在该版本中， jConnect 可确定 Adaptive Server 连接是否支持本机可滚动游标功能，并使用该功能代替客户端高速缓存。因此，大多数应用程序都可以在访问无序行的过程中获得显著性能提高并可降低客户端内存要求。

解决存储过程错误

本节介绍如何解决在尝试使用 jConnect 和存储过程时可能出现的问题。

RPC 返回比已注册参数少的输出参数

SQLState:JZ0SG - RPC 未返回与应用程序注册的参数相同数量的输出参数。

出现该错误的原因在于，通过调用 CallableStatement.registerOutParam 注册的参数多于在存储过程中声明为 “OUTPUT”的参数。确保已将所有相应参数声明为 “OUTPUT”。注意内容如下的代码行：

```
create procedure yourproc (@p1 int OUTPUT, ...
```

注释 如果在使用 SQL Anywhere 时出现此错误，请升级到 SQL Anywhere 版本 5.5.04 或更高版本。

在返回输出参数时出现获取/状态错误

如果查询没有返回行数据，则应该使用 CallableStatement.executeUpdate 或 execute 方法而不应使用 executeQuery 方法。

根据 JDBC 标准的要求，如果 executeQuery 没有结果集， jConnect 会抛出一个 SQL 例外。

在非链式事务模式中执行存储过程

Sybase 错误 7713 - 只能在非链式事务模式中执行存储过程。

当 JDBC 试图将连接置于 autocommit(true) 模式时会出现该错误。应用程序可使用 Connection.setAutoCommit(false) 或通过使用 “set chained on” 语言命令将连接更改为链式模式。如果存储过程不是在兼容模式中创建的，则会出现该错误。

若要修复该问题，请使用：

sp_procxmode *procedure_name,"*anymode"

解决自定义套接字执行错误

如果试图在调用 sun.security.ssl.SSLSocketImpl.setEnabledCipherSuites 时设置 SSL 套接字，则可能会接收到类似如下的例外：

```
java.lang.IllegalArgumentException:
 SSL_SH_anon_EXPORT_WITH_RC4_40_MDS
```

检验 SSL 库是否位于系统库路径中。

第 5 章

性能和调优

本章介绍如何在使用 jConnect 时优化或改善性能。

改善 jConnect 性能

以下多种方法用于优化使用 jConnect 的应用程序的性能：

- 使用 TextPointer.sendData 方法将文本和图像发送到 Adaptive Server 数据库。请参见第 62 页的 “更新数据库中的图像数据”。
- 创建在会话过程中重复使用的动态 SQL 语句的预编译 PreparedStatement 对象。请参见第 137 页的 “对动态 SQL 中的预准备语句的性能调优”。
- 使用批处理更新通过减少网络通信量来改善性能；具体地说，就是所有查询被发送到一个组的服务器，而且返回到客户端的所有响应被发送到一个组中。请参见第 59 页的 “批处理更新支持”。
- 对于可能会移动图像数据、大的行集以及过长的文本数据的会话，使用 PACKETSIZE 连接属性设置最大可用包大小。
- 对于 TDS-tunneled 的 HTTP，设置最大 TDS 包大小并配置 Web 服务器以支持 HTTP1.1 Keep-Alive 功能。此外，将 *SkipDoneProc* 服务器小程序参数设置为 “true”。
- 使用协议游标 （LANGUAGE_CURSOR 连接属性的缺省设置）。有关详细信息，请参见第 145 页的 “LANGUAGE_CURSOR 连接属性”。
- 如果使用 TYPE_SCROLL_INSENSITIVE 结果集，应只在结果集很小时才使用它们。有关详细信息，请参见第 56 页的 “在 jConnect 中使用 TYPE_SCROLL_INSENSITIVE 结果集”。

在后续章节中将介绍改善性能的其它需要考虑的事项。

BigDecimal 范围重设

JDBC 1.0 规范要求 getBigDecimal 有一个范围因子。然后，当从服务器返回 BigDecimal 对象时，必须使用 getBigDecimal 已使用的原范围因子对其重新进行范围设置。

若要减少范围重设所需的时间，请使用 JDBC 2.0 getBigDecimal 方法（jConnect 在 SybResultSet 类中实现，且不需要 *scale* 值）：

```
public BigDecimal getBigDecimal(int columnIndex)
  throws SQLException
```

例如：

```
SybResultSet rs =
   (SybResultSet)stmt.executeQuery("SELECT
   numeric_column from T1");
while(rs.next())
 {
   BigDecimal bd rs.getBigDecimal(
     "numeric_column");
      ...
 }
```

REPEAT_READ 连接属性

如果将 REPEAT_READ 连接属性设置为 “false”，则可改善从数据库中检索结果集的性能。但是，当 REPEAT_READ 为 “false”时：

- 必须按照列索引顺序读取列值。如果要按名称而不是按列编号访问列将是很困难的。
- 不能多次读取行中的列值。

SunIoConverter 字符集转换

如果使用多字节字符集并需要改善驱动程序性能，可以使用 jConnect 示例提供的 SunIoConverter 类。此转换程序以 Oracle Corporation 提供的 sun.io 类为基础。

SunIoConverter 类不是字符集转换程序功能的纯 Java 实现，而且因此未集成到标准 jConnect 产品中。不过，Sybase 已提供了该转换程序以供参考之用，并且可与 jConnect 驱动程序一起使用来改善字符集转换性能。

注释 根据 Sybase 的测试，SunIoConverter 类在所测试的所有虚拟机上都改善了性能。不过，Oracle Corporation 保留在 JDK 未来版本中删除或更改 sun.io 类的权利。因此，此 SunIoConverter 类可能会与 JDK 的更高版本不兼容。

若要使用 SunIoConverter 类，必须先安装 jConnect 示例应用程序。安装完毕后，设置 CHARSET_CONVERTER_CLASS 连接属性，使其指向 jConnect 安装目录的 *sample2* 子目录中的 SunIoConverter 类。有关安装 jConnect 及其组件 （包括示例应用程序）的完整指导，请参见 Sybase jConnect for JDBC Installation Guide （《Sybase jConnect for JDBC 安装指南》）。

若正在使用缺省字符集为 iso_1 的数据库或仅前 7 位 ASCII，则通过使用 TruncationConverter 可获得显著的性能优势。请参见第 35 页的“jConnect 字符集转换程序”。

对动态 SQL 中的预准备语句的性能调优

在 Embedded SQL™ 中，动态语句是需要在运行期编译而不是静态编译的 SQL 语句。通常，动态语句包含输入参数，但这不是必需的。在 SQL 中， prepare 命令用于预编译动态语句并将其保存，以使其在会话期间不必重新编译便可重复执行。

如果语句在一个会话中使用多次，预编译将比每次使用时将其发送到数据库并进行编译提供更好的性能。语句越复杂，性能优势就越显著。

如果可能仅使用几次语句，预编译可能会降低效率，因为在数据库中的预编译、保存以及随后的释放都会增加开销。

预编译要执行的动态 SQL 语句并将其保存到内存中会耗用时间和资源。如果在会话中不太可能多次使用同一个语句，执行数据库 prepare 的开销可能会大大超过其性能优势。另外需要考虑的是，只要数据库中预准备了动态 SQL 语句，它就非常类似于存储过程。在某些情况下，创建存储过程并使其驻留在服务器中可能会比在应用程序中定义预准备语句更可取。这将在第 138 页的“选择预准备语句和存储过程”中讨论。

可以使用 jConnect 优化 Sybase 数据库中动态 SQL 语句的性能，方法如下：

- 在同一语句在会话中可能多次执行的情况下，创建包含预编译语句的 PreparedStatement 对象。
- 在同一语句在会话中很少使用的情况下，创建包含未编译 SQL 语句的 PreparedStatement 对象。

如以下各节所述，设置 DYNAMIC_PREPARE 连接属性并创建 PreparedStatement 对象的最佳方法可能取决于应用程序是否需要跨 JDBC 驱动程序移植，或者所编写的应用程序是否允许特定于 jConnect 的 JDBC 扩展。

jConnect 提供了动态 SQL 语句的性能调优功能。

选择预准备语句和存储过程

如果创建包含预编译动态 SQL 语句的 PreparedStatement 对象，一旦该语句在数据库中编译，它实际上就变成了存储过程，驻留在内存中，并附加到与会话相关的数据结构中。在决定是否维护数据库中的存储过程或在应用程序中创建包含已编译 SQL 语句的 PreparedStatement 对象时，资源需求以及数据库和应用程序维护都是需要考虑的重要因素：

- 存储过程一旦被编译，就跨所有的连接在全局都可用。相反，PreparedStatement 对象中的动态 SQL 语句在每个使用它的会话中都需要进行编译和释放。
- 如果应用程序访问多个数据库，使用存储过程意味着相同的存储过程需要在所有的目标数据库上都可用。这样便产生了数据库维护问题。如果对动态 SQL 语句使用 PreparedStatement 对象，就可避免出现这一问题。
- 如果应用程序为调用存储过程创建了 CallableStatement 对象，则可在该存储过程中封装 SQL 代码和表引用。然后可修改基础数据库或 SQL 代码而无需更改应用程序。

可移植应用程序中的预准备语句

如果在来自不同供应商的数据库上运行应用程序，而且想要一些 PreparedStatement 对象包含预编译语句，而其它对象包含未编译语句，进行如下处理：

- 在访问 Sybase 数据库时，确保已将 DYNAMIC_PREPARE 连接属性设置为 “true”。

- 若要返回包含预编译语句的 PreparedStatement 对象，请以标准方式使用 Connection.prepareStatement：

  ```
  PreparedStatement ps_precomp =
     Connection.prepareStatement(sql_string);
  ```

- 若要返回包含未编译语句的 PreparedStatement 对象，请使用 Connection.prepareCall。

  Connection.prepareCall 返回 CallableStatement 对象，但因为 CallableStatement 是 PreparedStatement 的一个子类，所以您可以将 CallableStatement 对象向上转换为 PreparedStatement 对象，如下所示：

  ```
  PreparedStatement ps_uncomp =
     Connection.prepareCall(sql_string);
  ```

  确保 PreparedStatement 对象 *ps_uncomp* 包含未编译语句，因为仅执行 Connection.prepareStatement 以返回包含预编译语句的 PreparedStatement 对象。

具有 jConnect 扩展的预准备语句

如果不考虑跨驱动程序的可移植性，可编写使用 SybConnection.prepareStatement 的代码，以指定 PreparedStatement 对象是否包含预编译或未编译语句。在这种情况下，如何编码预准备语句可取决于应用程序中的大多数动态语句在会话中可能执行多次还是仅执行几次。

如果大多数动态语句不常执行

对于大多数动态 SQL 语句在会话中可能仅执行一两次的应用程序：

- 将连接属性 DYNAMIC_PREPARE 设置为 “false”。
- 若要返回包含未编译语句的 PreparedStatement 对象，请以标准方式使用 Connection.prepareStatement：

  ```
  PreparedStatement ps_uncomp =
     Connection.prepareStatement(sql_string);
  ```

- 若要返回包含预编译语句的 PreparedStatement 对象，请使用 SybConnection.prepareStatement 并将 *dynamic* 设置为 “true”，如下所示：

```
PreparedStatement ps_precomp =
   (SybConnection)conn.prepareStatement(sql_string, true);
```

如果大多数动态语句在会话中执行多次

如果应用程序中的大多数动态语句在会话中可能执行多次，则进行如下处理：

- 将连接属性 DYNAMIC_PREPARE 设置为 “true”。
- 若要返回包含预编译语句的 PreparedStatement 对象，请以标准方式使用 Connection.prepareStatement：

```
PreparedStatement ps_precomp = Connection.prepareStatement(sql_string);
```

- 若要返回包含未编译语句的 PreparedStatement 对象，则可以使用 Connection.prepareCall （请参见可移植应用程序中的预准备语句）或 SybConnection.prepareStatement，并将 *dynamic* 设置为 “false”：

```
PreparedStatement ps_uncomp =
   (SybConnection)conn.prepareStatement(sql_string, false);

PreparedStatement ps_uncomp = Connection.prepareCall(sql_string);
```

Connection.prepareStatement

jConnect 执行 Connection.prepareStatement，因此可对其进行设置，以返回 PreparedStatement 对象中的预编译 SQL 语句或未编译 SQL 语句。如果设置 Connection.prepareStatement 返回 PreparedStatement 对象中的预编译 SQL 语句，它会将动态 SQL 语句发送到数据库中进行预编译，并且如同直接执行 prepare 命令时一样，被准确地保存下来。如果设置 Connection.prepareStatement 返回未编译 SQL 语句，它将返回 PreparedStatement 对象中的未编译 SQL 语句，而不将其发送到数据库中。

Connection.prepareStatement 返回的 SQL 语句的类型由连接属性 DYNAMIC_PREPARE 确定，而且适用于整个会话。

对于 Sybase 特定应用程序， jConnect 6.05 及更高版本在 jConnect SybConnection 类下提供了 prepareStatement 方法。SybConnection.prepareStatement 允许用户指定是否要对各动态 SQL 语句进行预编译，而不受 DYNAMIC_PREPARE 连接属性会话级设置的影响。

DYNAMIC_PREPARE 连接属性

DYNAMIC_PREPARE 是用于启用动态 SQL 预准备语句的布尔值连接属性：

- 如果将 DYNAMIC_PREPARE 设置为 “true”，则在会话期间每次调用 Connection.prepareStatement 都将试图返回 PreparedStatement 对象中的预编译语句。

  在这种情况下， PreparedStatement 执行时，它所包含的语句已在数据库中进行了预编译，拥有动态赋值的占位符，而且仅需执行该语句。

- 如果将连接的 DYNAMIC_PREPARE 设置为 “false”，则由 Connection.prepareStatement 返回的 PreparedStatement 对象不包含预编译语句。

  在这种情况下，每次执行 PreparedStatement 时，它所包含的动态 SQL 语句必须发送到数据库中进行编译和执行。

DYNAMIC_PREPARE 的缺省值为 “true”。

在以下示例中， DYNAMIC_PREPARE 设置为 “false”以禁用动态 SQL 语句预编译。在该示例中， props 是用于指定连接属性的 Properties 对象。

```
...
props.put("DYNAMIC_PREPARE", "false")
Connection conn = DriverManager.getConnection(url, props);
```

将 DYNAMIC_PREPARE 设置为 “true”时，应注意：

- 并非所有的动态语句均可在 prepare 命令下预编译。 SQL-92 标准对可用于 prepare 命令的语句做了一些限制，而且每个数据库供应商可能会有各自不同的约束。
- 如果数据库因为不能预编译和不能保存通过 Connection.prepareStatement 发送到该数据库的语句而产生错误，jConnect 会捕获该错误，并返回包含未编译动态 SQL 语句的 PreparedStatement 对象。每次执行 PreparedStatement 对象时，该语句都会重新发送到数据库进行编译和执行。
- 在会话结束或显式关闭预编译语句的 PreparedStatement 对象之前，预编译语句将一直驻留在数据库的内存中。 PreparedStatement 对象的碎片收集不能从数据库中删除预准备语句。

作为一般规则，应当在最后一次使用 PreparedStatement 对象之后显式关闭它，以避免预准备语句在会话期间累积在服务器的内存中并降低性能。

SybConnection.prepareStatement

如果应用程序允许特定于 jConnect 的 JDBC 扩展，则可使用 SybConnection.prepareStatement 扩展方法返回 PreparedStatement 对象中的动态 SQL 语句：

```
PreparedStatement SybConnection.prepareStatement(String sql_stmt,
   boolean dynamic) throws SQLException
```

SybConnection.prepareStatement 会根据 *dynamic* 参数的设置，返回包含预编译或未编译 SQL 语句的 PreparedStatement 对象。如果 *dynamic* 为“true”，则 SybConnection.prepareStatement 返回具有预编译 SQL 语句的 PreparedStatement 对象。如果 *dynamic* 为 “false”，则它会返回具有未编译 SQL 语句的 PreparedStatement 对象。

以下示例显示了如何使用 SybConnection.prepareStatement 返回包含预编译语句的 PreparedStatement 对象：

```
PreparedStatement precomp_stmt = ((SybConnection) conn).prepareStatement
   ("SELECT * FROM authors WHERE au_fname LIKE ?", true);
```

在该示例中，连接对象 *conn* 被转换为 SybConnection 对象，以允许使用 SybConnection.prepareStatement。传递给 SybConnection.prepareStatement 的 SQL 字符串在数据库中预编译，即使连接属性 DYNAMIC_PREPARE 设置为 “false”。

如果数据库因为不能预编译通过 SybConnection.prepareStatement 发送到该数据库的语句而产生错误，则 jConnect 会抛出 SQLException，并且调用无法返回 PreparedStatement 对象。这与 Connection.prepareStatement 不同，后者会捕获 SQL 错误，并且如果产生上述错误，它会返回包含未编译语句的 PreparedStatement 对象。

ESCAPE_PROCESSING_DEFAULT 连接属性

缺省情况下，jConnect 会分析提交到数据库的所有 SQL 语句，以查找有效的 JDBC 函数转义。如果应用程序不在其 SQL 调用中使用 JDBC 函数转义，可将此连接属性设置为 “false”以回避此分析过程。这样做可以稍微改善性能。

jConnect 中的优化批处理

jConnect 实施内部算法以加快 PreparedStatement 对象的批处理操作速度。此算法在 HOMOGENEOUS_BATCH 连接属性设置为 true 时被调用。

注释 只有客户端应用程序连接到支持同类批处理的服务器时，此功能才可用。 Adaptive Server Enterprise 15.7 引入了对同类批处理的支持。

以下示例说明了使用 addBatch 和 executeBatch 方法的 PreparedStatement 批处理操作：

```
String sql = "update members set lastname = ? where
member_id = ?";
prep_stmt = connection.prepareStatement(sql);

prep_stmt.setString(1, "Forrester");
prep_stmt.setLong(2, 45129);
prep_stmt.addBatch();

prep_stmt.setString(1, "Robinson");
prep_stmt.setLong(2, 45130);
prep_stmt.addBatch();

prep_stmt.setString(1, "Servo");
prep_stmt.setLong(2, 45131);
prep_stmt.addBatch();

prep_stmt.executeBatch();
```

其中， `connection` 表示连接实例， `prep_stmt` 表示预准备语句实例，而 `?` 表示预准备语句的参数占位符。

大对象 (LOB) 列的同类批处理

如果 HOMOGENEOUS_BATCH 和 ENABLE_LOB_LOCATORS 属性设置为 true，客户端应用程序就无法将 LOB 和非 LOB 预准备语句 setter 方法混合在同一批处理中。例如，以下语句是无效的：

```
String sql = "update members SET catchphrase = ?WHERE
member_id = ?";
prep_stmt = connection.prepareStatement(sql);

prep_stmt.setString(1, "Push the button, Frank!");
prep_stmt.setLong(2, 45129);
prep_stmt.addBatch();

Clob myclob = con.createClob();
myclob.setString(1, "Hi-keeba!");
prep_stmt.setClob(1, myclob);
prep_stmt.setLong(2, 45130);
prep_stmt.addBatch();

pstmt.executeBatch();
```

其中， catchphrase 表示类型为 text 的列。此代码失败的原因是 setString 方法和 setClob 方法用在同一列的同一批处理中。

游标性能

当在 SybCursorResultSet 类中使用 Statement.setCursorName 方法或 setFetchSize( ) 方法时， jConnect 会在数据库中创建游标。使用其它方法可以使 jConnect 打开、读取和更新游标。

jConnect 可通过将 SQL 语句发送到数据库或通过将游标命令编码为 TDS 通信协议内部的标识来创建和操纵游标。第一种类型的游标是 “语言游标”，第二种类型的游标是 “协议游标”。

协议游标能够提供比语言游标更高的性能。另外，并非所有数据库都支持语言游标。例如， SQL Anywhere 数据库就不支持语言游标。

在 jConnect 中，缺省条件是所有游标都是协议游标。不过，LANGUAGE_CURSOR 连接属性提供了通过数据库中的语言命令创建和操纵游标的选项。

LANGUAGE_CURSOR 连接属性

LANGUAGE_CURSOR 在 jConnect 中是一个布尔值连接属性，用于确定将游标创建为协议游标还是语言游标：

- 如果 LANGUAGE_CURSOR 设置为 “false”，则在会话期间创建的所有游标均为能够提供更好性能的协议游标。 jConnect 通过将游标命令作为 TDS 协议中的标识发送来创建和操纵游标。

  缺省情况下， LANGUAGE_CURSOR 设置为 “false”。

- 如果 LANGUAGE_CURSOR 设置为 “true”，则在会话期间创建的所有游标均为语言游标。 jConnect 通过将 SQL 语句发送到数据库进行分析和编译来创建和操纵游标。

  将 LANGUAGE_CURSOR 设置为 “true”无任何已知优点，但如果将其设置为 “false”时应用程序显示意外行为，在这种情况下可将其设置为 “true”。

第 6 章

迁移 jConnect 应用程序

本章介绍如何将应用程序从 jConnect 5.x 和 6.x 迁移到 jConnect 7.x。

向 jConnect 7.x 迁移应用程序

使用以下过程升级到 jConnect 7.x。

❖ **迁移到 jConnect 7.x**

1 如果代码使用 Sybase 扩展，或者您要在代码中显式导入任何 jConnect 类，则可以根据需要更改软件包导入语句。

例如，将导入语句

```
import com.sybase.jdbc.*
```

和

```
import com.sybase.jdbc2.jdbc.*
```

更改为

```
import com.sybase.jdbcx.*
```

有关使用 Sybase 扩展 API 的信息，请参见第 148 页的 “更改 Sybase 扩展”。

2 将 JDBC_HOME 设置为 jConnect 驱动程序的顶层安装目录：

```
JDBC_HOME=jConnect-7_0
```

3 更改 CLASSPATH 环境变量以反映新的安装。类路径必须包括以下内容：

```
JDBC_HOME/classes/jconn4.jar
```

4 更改用于装载驱动程序的源代码，并重新编译该应用程序以使用新的驱动程序：

```
Class.forName("com.sybase.jdbc4.jdbc.SybDriver");
```

5 检验 jConnect 7.0 驱动程序是否为 CLASSPATH 环境变量中指定的第一个 jConnect 驱动程序。

更改 Sybase 扩展

jConnect 版本 4.1 及更高版本包括软件包 com.sybase.jdbcx，该软件包含 JDBC 的所有 Sybase 扩展。在 jConnect 4.1 以前的版本中，可在 com.sybase.jdbc 和 com.sybase.utils 软件包中找到这些扩展。

com.sybase.jdbcx 软件包为不同版本的 jConnect 提供一致的接口。所有 Sybase 扩展都定义为 Java 接口，从而可以在不影响使用这些接口建立的应用程序的情况下更改底层实现。

当开发使用 Sybase 扩展的新应用程序时，可使用 com.sybase.jdbcx。该软件包中的接口允许以最小的更改将应用程序升级到 jConnect 4.0 以后的版本。

某些 Sybase 扩展已经过更改以容纳 com.sybase.jdbcx 接口。

扩展更改示例

如果应用程序使用 SybMessageHandler，代码的区别如下：

- **jConnect 4.0** 代码：

```
import com.sybase.jdbc.SybConnection;
import com.sybase.jdbc.SybMessageHandler;
.
.
Connection con = DriverManager.getConnection(url, props);
SybConnection sybCon = (SybConnection) con;
sybCon.setMessageHandler(new ConnectionMsgHandler());
```

- **jConnect 6.0** 代码：

```
import com.sybase.jdbcx.SybConnection;
 import com.sybase.jdbcx.SybMessageHandler;
.
.
Connection con = DriverManager.getConnection(url, props);
SybConnection sybCon = (SybConnection) con;
sybCon.setSybMessageHandler(new ConnectionMsgHandler());
```

有关如何使用 Sybase 扩展的更多示例，请参见随 jConnect 一起提供的示例。

方法名称

表 6-1 列出了方法在新接口中的新名称。

表 6-1：方法名称更改

Class	原有名称	新名称
SybConnection	getCapture( )	createCapture( )
SybConnection	setMessageHandler( )	setSybMessageHandler( )
SybConnection	getMessageHandler( )	getSybMessageHandler( )
SybStatement	setMessageHandler( )	setSybMessageHandler( )
SybStatement	getMessageHandler( )	getSybMessageHandler( )

Debug 类

不再支持对 Debug 类的直接静态引用，但在 com.sybase.utils 软件包中存在此不受支持的引用。要使用 jConnect 调试功能，应使用 SybDriver 类的 getDebug 方法获得对 Debug 类的引用。例如：

```
import com.sybase.jdbcx.SybDriver;
import com.sybase.jdbcx.Debug;
.
.
.
SybDriver sybDriver =
    SybDriver)Class.forName
    ("com.sybase.jdbc4.jdbc.SybDriver") newInstance();
Debug sybDebug = sybDriver.getDebug();
sybDebug.debug(true, "ALL", System.out);
```

jConnect javadoc 文档中有 Sybase 扩展的完整列表，该文档位于 jConnect 安装目录的 *docs/* 目录中。

第 7 章

Web 服务器网关

本章介绍 Web 服务器网关并说明如何在 jConnect 中使用这些网关。

关于 Web 服务器网关

如果数据库服务器与 Web 服务器运行在不同的主机上，或者正在开发的 Internet 应用程序必须通过防火墙连接到安全的数据库服务器，则需要一个网关充当代理，以提供到数据库服务器的路径。

为使用安全套接字层 (SSL) 协议连接到服务器， jConnect 提供了一个 Java 服务器小程序，该小程序可安装在支持 javax.servlet 接口的任何 Web 服务器上。该服务器小程序启用 jConnect 支持加密并将 Web 服务器用作网关。

注释 jConnect 支持客户端系统上的 SSL。有关详细信息，请参见第 108 页的 “实现自定义套接字插件”。

使用 TDS 贯通

jConnect 使用 TDS 与数据库服务器通信。HTTP 贯通 TDS 可用于转发请求。从客户端到后端数据库的请求经过了整个网关，并在请求的正文中包含 TDS。请求的标头指示请求包中包含的 TDS 的长度。

TDS 是一种面向连接的协议，但 HTTP 不是。为支持安全性功能（如为 Internet 应用程序加密），jConnect 使用 TDS 贯通服务器小程序来维护各 HTTP 请求间的逻辑连接。服务器小程序在初始登录请求的过程中生成一个会话 ID，并且每个后续请求的标头中都包含此会话 ID。使用会话 ID 可以标识活动会话甚至恢复会话，只要服务器小程序拥有一个使用该特定会话 ID 的开放式连接即可。

TDS 贯通服务器小程序提供的逻辑连接使 jConnect 能够支持两个系统间的加密通信，例如， jConnect 客户端将 CONNECT_PROTOCOL 连接属性设置为 “https”后可连接到运行 TDS 贯通服务器小程序的 Web 服务器。

配置 jConnect 和网关

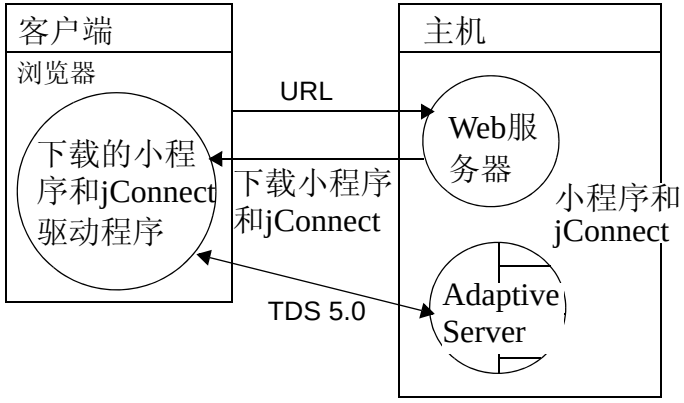
有几个用于设置 Web 服务器和 Adaptive Server 的选项。下面是关于四种常见配置的示例，显示了在何处安装 jConnect 驱动程序以及何时使用带有TDS 贯通服务器小程序的网关。

Web 服务器和 Adaptive Server 在同一主机上

在两层配置中， Web 服务器和 Adaptive Server 安装在同一主机上：

- 在 Web 服务器主机上安装 jConnect。
- 不需要任何网关。

图 7-1：Web 服务器和 Adaptive Server 在同一主机上

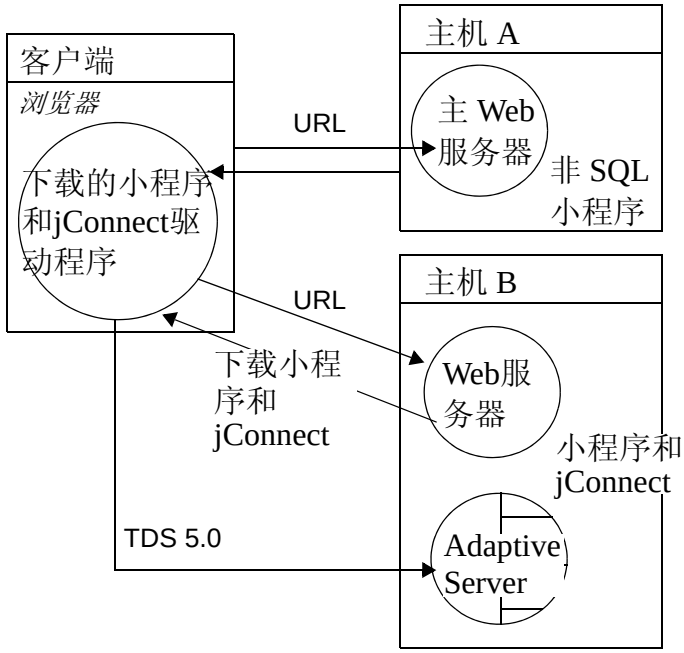


专用 JDBC Web 服务器和 Adaptive Server 在同一主机上

在此配置中，主 Web 服务器在一台单独的主机上。另一台主机由专用于 Adaptive Server 访问的 Web 服务器和 Adaptive Server 共享。来自主服务器的链接发送请求，要求 SQL 访问专用 Web 服务器。在另一台主机上安装：

- 在第二台 (Adaptive Server) 主机上安装 jConnect。
- 不需要任何网关。

图 7-2：专用 JDBC Web 服务器和 Adaptive Server 在同一主机上

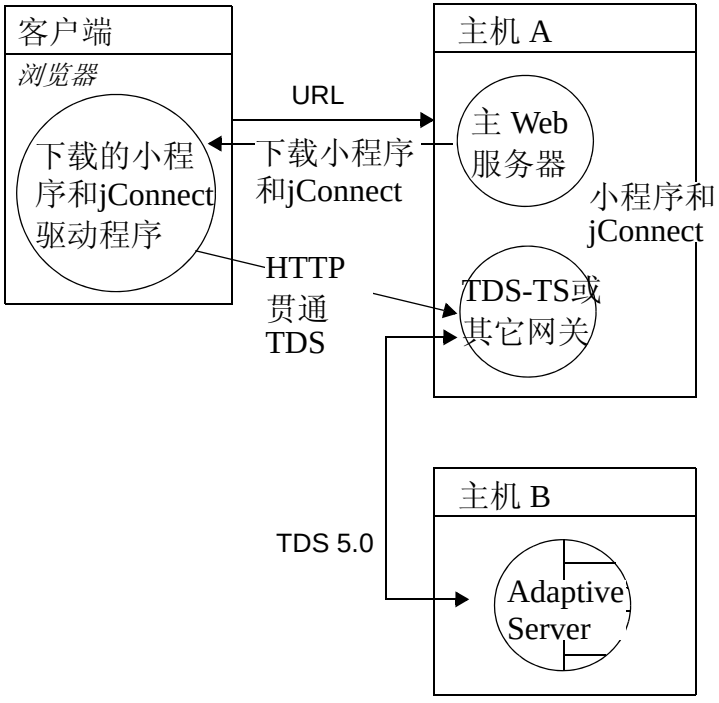


Web 服务器和 Adaptive Server 在不同的主机上

在三层配置中， Adaptive Server 和 Web 服务器在不同的主机上。jConnect 需要一个网关来充当 Adaptive Server 的代理。

- 在 Web 服务器主机上安装 jConnect。
- 安装 TDS 贯通服务器小程序或其它网关。

图 7-3：Web 服务器和 Adaptive Server 在不同的主机上

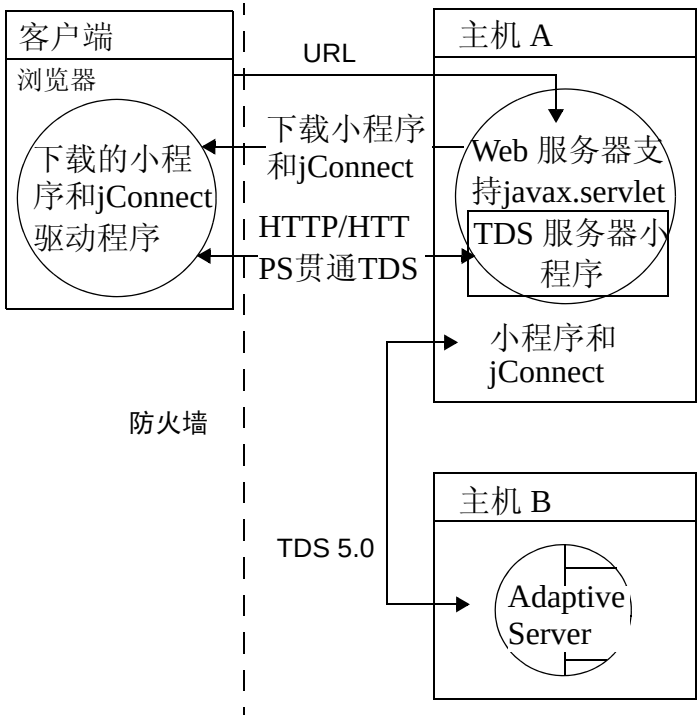


通过防火墙连接到服务器

若要连接到受防火墙保护的服务器，必须使用带有 TDS 贯通服务器小程序的 Web 服务器来支持在 Internet 上传输数据库请求响应。

- 在 Web 服务器主机上安装 jConnect。
- 需要一台支持 javax.servlet 接口的 Web 服务器。

图 7-4：通过防火墙连接到服务器



使用要求

以下几节介绍 Web 服务器网关的使用要求。

阅读 *index.html* 文件

使用 Web 浏览器查看 jConnect 安装目录中的 *index.html* 文件。*index.html* 中提供了指向 jConnect 文档和示例代码的链接。

注释 如果在安装有 jConnect 的同一台计算机上使用 Netscape，请确保浏览器无权访问 CLASSPATH 环境变量。请参见 Sybase jConnect for JDBC Installation Guide and Release Bulletin （《Sybase jConnect for JDBC 安装指南和发行公告》）第 3 章中的 “Restrictions on Setting CLASSPATH When You Use Netscape”（使用 Netscape 时设置 CLASSPATH 的限制）。

❖ **查看 index.html 文件**

1 打开 Web 浏览器。

2 输入与安装相匹配的 URL。例如，如果浏览器和 Web 服务器运行在同一主机上，则输入：

```
http://localhost:8000/index.html
```

如果浏览器和 Web 服务器运行在不同的主机上，则输入：

```
http://host:port/index.html
```

其中， *host* 是运行 Web 服务器的主机的名称， *port* 是侦听端口。

运行示例 Isql 小程序

在浏览器中装载 *index.html* 文件后：

❖ **运行示例小程序**

1 单击 “运行示例 JDBC 小程序” (Run Sample JDBC Applets)。

此操作将带您进入 “jConnect 示例程序” (jConnect Sample Programs) 页面。

2 向下移动 “示例程序” (Sample Programs) 页面以找到 “可执行的示例” (Executable Samples) 下的表。

3 在该表中找到 “Isql.java” (Isql.java) 并单击行末的 “运行” (Run)。

示例 Isql.java 小程序提示对示例数据库执行简单查询并显示结果。小程序显示缺省的 Adaptive Server 主机名、端口号、用户名 (*guest*)、口令 (*sybase*)、数据库和查询。小程序使用缺省值连接到 Sybase 示例数据库。单击 “执行” (Go) 后将返回结果。

故障排除

在 UNIX 上，如果小程序未能按预期显示，则可以修改小程序的屏幕维度：

❖ **修改小程序的屏幕维度**

1　使用文本编辑器编辑下面的内容：

$JDBC_HOME/sample2/gateway.html

2　将第 7 行的高度参数更改为 650。可尝试使用不同的高度设置。

3　在浏览器上重装 Web 页。

使用 TDS 贯通服务器小程序

要使用 TDS 贯通服务器小程序，需要一台支持 javax.servlet 接口的 Web 服务器（如 Oracle Corporation Java Web 服务器）。安装 Web 服务器时，把 jConnect TDS 贯通服务器小程序添加到活动服务器小程序列表中。还可以通过设置服务器小程序参数来定义连接超时和最大包大小。

使用 TDS 贯通服务器小程序时，从客户端到后端服务器的请求贯通网关，这样的请求包括 GET 或 POST 命令、 TDS 会话 ID （在初始请求后）、后端地址和请求状态。

TDS 在请求正文中。两个标头字段表示 TDS 流的长度和网关指派的会话 ID。

当客户端发送请求时， Content-Length 标头字段表示 TDS 内容的大小，请求命令为 POST。如果由于客户端正在检索来自服务器的响应数据的下一部分内容或正在关闭连接，从而造成请求中没有任何 TDS 数据，此时请求命令为 GET。

以下示例演示了如何使用 TDS 贯通 HTTPS 协议在客户端和 HTTPS 网关之间传递信息；该示例显示了一个到名为 “DBSERVER”的后端服务器的端口号为 “1234”的连接。

表 7-1：客户端到网关的登录请求。无会话 ID。

查询	POST/tds?ServerHost=dbserver&ServerPort=1234& Operation=more HTTP/1.0
标头	内容长度：605
内容 (TDS)	登录请求

表 7-2：网关到客户端。标头包含由 TDS 服务器小程序指派的会话 ID。

查询	200 SUCCESS HTTP/1.0
标头	内容长度：210 TDS 会话：TDS00245817298274292
内容 (TDS)	登录确认 EED

表 7-3：客户端到网关。所有后续请求的标头都包含会话 ID。

查询	POST/tds?TDS- Session=TDS00245817298274292&Operation=more HTTP/1.0
标头	内容长度：32
内容 (TDS)	查询 “SELECT * from authors”

表 7-4：网关到客户端。所有后续响应的标头都包含会话 ID。

查询	200 SUCCESS HTTP/1.0
标头	内容长度：2048 TDS 会话：TDS00245817298274292
内容 (TDS)	行格式与某些来自查询响应的行

检查要求

若要对 TDS 贯通的 HTTP 使用 jConnect 服务器小程序，需要：

- 一台支持 javax.servlet 接口的 Web 服务器。若要安装该服务器，请遵循它提供的操作说明。

安装服务器小程序

jConnect 安装包括 *classes* 目录下的 *gateway2* 子目录。该子目录包含 TDS 贯通服务器小程序所需的文件。

将 jConnect gateway 包复制到 Web 服务器的 *servlets* 目录下的 *gateway2* 子目录中。复制好服务器小程序后，按照 Web 服务器操作说明激活服务器小程序。

设置服务器小程序参数

将服务器小程序添加到 Web 服务器时，可输入可选参数定制性能：

- *SkipDoneProc [true|false]* – Sybase 数据库通常会在查询执行期间执行中间处理步骤时返回行计数信息。通常，客户端应用程序会忽略此数据。如果将 *SkipDoneProc* 设置为 “true”，服务器小程序会随即从响应中删除此额外信息，这将减少网络使用和客户端上的处理要求。这在使用 HTTPS/SSL 时尤其有效，因为不需要的数据在被忽略前不会被加密/解密。
- *TdsResponseSize* – 为贯通的 HTTPS 设置最大 TDS 包大小。如果只有几个用户有大量数据，较大的 *TdsResponseSize* 会更加有效。如果有许多执行小事务的用户，请使用较小的 *TdsResponseSize*。
- *TdsSessionIdleTimeout* – 定义在服务器连接自动关闭前该连接能够维持空闲状态的时间量 （以毫秒为单位）。 *TdsSessionIdleTimeout* 缺省值为 600,000 （10 分钟）。

  如果交互式客户端程序可能长时间处于空闲状态而您又不希望中断该连接，则增加 *TdsSessionIdleTimeout*。

  还可从 jConnect 客户端使用 SESSION_TIMEOUT 连接属性设置连接超时值。这在存在可能长时间处于空闲状态的特定应用程序时很有用。在这种情况下，可通过 SESSION_TIMEOUT 连接属性为连接而不是为服务器小程序设置更长的超时值。
- *Debug* – 打开调试程序。请参见第 125 页的 “使用 jConnect 进行调试”。

输入服务器小程序参数 （以逗号分隔的字符串）。例如：

```
TdsResponseSize=[size],TdsSessionIdleTimeout=
  [timeout],Debug=true
```

有关输入服务器小程序参数的完整指导，请参见 Web 服务器文档。

调用服务器小程序

jConnect 根据 *proxy* 连接属性的路径扩展，确定何时使用安装有 TDS 贯通服务器小程序的网关。 jConnect 能够识别 *proxy* 的服务器小程序路径扩展并调用指定网关上的服务器小程序。

用下面的格式定义连接 URL：

```
http://host:port/TDS-servlet-path
```

jConnect 通过调用 Web 服务器上的 TDS 贯通服务器小程序使 TDS 贯通 HTTP。服务器小程序的路径必须是服务器别名列表中为 Web 服务器定义的路径。

跟踪活动的 TDS 会话

可查看活动 TDS 会话的相关信息，其中包括每个会话的服务器连接。使用 Web 浏览器打开管理 URL：

```
http://host:port/TDS-servlet-path?Operation=list
```

例如，如果服务器为 “myserver”， TDS 服务器小程序路径为 */tds*，则输入：

```
http://myserver:8080/tds?Operation=list
```

这将显示活动 TDS 会话的列表。单击会话可查看更多信息 （包括服务器连接）。

终止 TDS 会话

可使用上述 URL 终止任何活动的 TDS 会话。在第一页上单击会话列表中的某个活动会话，然后单击 “终止会话” (Terminate This Session)。

恢复 TDS 会话

必要时，可通过设置 SESSION_ID 连接属性来恢复现有的开放式连接。指定 SESSION_ID 时， jConnect 会跳过协议的登录阶段，并使用指定的会话 ID 恢复与网关的连接。如果指定的会话 ID 不在服务器小程序中，jConnect 会在用户第一次尝试使用连接时抛出 SQL 异常。

附 录 A

jConnect 示例程序

该附录是 jConnect 示例程序的指南：

运行 IsqlApp

IsqlApp 允许从命令行发出 isql 命令，而且允许运行 jConnect 示例程序。

IsqlApp 的语法如下：

```
IsqlApp [-U username]
   [-P password]
   [-S servername]
   [-G gateway]
   [-p {http|https}]
   [-D debug_class_list]
   [ -v ]
   [-I input_command_file]
   [-c command_terminator]
   [-C charset]
   [-L language]
   [-K service_principal_name]
   [-F JAAS_login_config_file_path]
   [-T sessionID]
   [-V <version {2,3,4,5}>]
```

参数	说明
-U	用于连接到服务器的登录 ID。
-P	指定的登录 ID 的口令。
-S	要连接的服务器的名称。
-G	网关地址。对于 HTTP 协议， URL 是：http://*host:port*。 若要使用支持加密的 HTTPS 协议， URL 应为 https://*host*:*port*/*servlet_alias*。

参数	说明
-p	指定是要使用 HTTP 协议还是使用支持加密的 HTTPS 协议。
-D	为所有类或指定的类打开调试程序，类之间用逗号分隔。例如， *-D ALL* 显示所有类的调试输出。 *-D SybConnection, Tds* 仅显示 SybConnection 和 Tds 类的调试输出。
-v	打开详细输出以显示或打印输出。
-I	让 IsqlApp 接受来自文件而不是来自键盘的命令。 在此参数之后，指定用于 IsqlApp 输入的文件的名称。文件必须包含命令终结符 （缺省为 “go”）。
-c	允许指定一个关键字 （如 “go”），当独占一行输入该关键字时可终止命令。这使得在使用终结符关键字之前可以输入多行命令。如果不指定命令终结符，每个新行都会终止命令。
-C	为通过 TDS 的字符串指定字符集。 如果不指定字符集， IsqlApp 将使用服务器的缺省字符集。
-L	为 jConnect 消息和从服务器返回的错误消息指定显示语言。
-K	表示用户想要通过 Kerberos 登录到 Adaptive Server。此参数设置服务主体名称。例如： `-K myASE` 该示例表示您希望执行 Kerberos 登录，且服务器的服务主体名称为 myASE。 有关详细信息，请参见第 3 章 “安全性”。
-F	指定 JAAS 登录配置文件的路径。如果使用 -K 选项，则必须设置此属性。例如： `-F /foo/bar/exampleLogin.conf` 请参见 jConnect 安装的 *sample2* 目录中的 *ConnectKerberos.java* 示例。有关详细信息，请参见第 3 章 “安全性”。
-T	如果设置了此参数， jConnect 将假定某个应用程序正试图在由 TDS 贯通网关保持打开的现有 TDS 会话上重新开始通信。 jConnect 将跳过登录协商并将所有来自应用程序的请求转发到指定的会话 ID。
-V	启用特定于版本的使用特性。请参见第 6 页的 “使用 JCONNECT_VERSION”。

注释 必须在每个选项标志后面输入一个空格。

若要获取命令行选项的完整描述，请输入：

```
java IsqlApp -help
```

下面的示例显示了如何通过端口 “3756”连接到主机 “myserver”上的数据库，并运行名为 “myscript”的 isql 脚本：

```
java IsqlApp -U sa -P sapassword
   -S jdbc:sybase:Tds:myserver:3756
   -I $JDBC_HOME/sp/myscript -c run
```

注释 提供了对 isql 命令进行 GUI 访问的小程序，如下所示：
$JDBC_HOME/sample2/gateway.html (UNIX)
%JDBC_HOME%\sample2\gateway.html (Windows)

运行 jConnect 示例程序和代码

jConnect 包含多个示例程序，这些示例程序说明了本章所涉及的多个主题，旨在帮助您理解 jConnect 如何处理各种 JDBC 类和方法。此外，本节还提供一个示例代码段作为参考。

示例应用程序

安装 jConnect 时，还可以安装示例程序。这些示例包含源代码，因此可以查看 jConnect 是如何实现各种 JDBC 类和方法的。有关安装示例程序的完整指导，请参见 *jConnect for JDBC* 安装指南。

注释 jConnect 示例程序仅用于示范。

示例程序安装在 jConnect 安装目录的 *sample2* 子目录下。 *sample2* 子目录中的文件 *index.html* 包含可用示例的完整列表以及每个示例的说明。*index.html* 还允许将示例程序作为小程序查看和运行。

运行示例小程序

使用 Web 浏览器，就可以将有些示例程序作为小程序运行。这使您可在查看输出结果的同时查看源代码。

如果想要以小程序形式运行示例程序，请在 Web 浏览器中输入 http://localhost:8000/sample2/index.html，以启动 Web 服务器网关。

与 SQL Anywhere 一起运行示例程序

所有示例程序均与 Adaptive Server 兼容，但只有有限数目的示例程序与 SQL Anywhere 兼容。若要获得与 SQL Anywhere 兼容的示例程序的当前列表，请参见 *sample2* 子目录中的 *index.html*。

若要运行可用于 SQL Anywhere 的示例程序，必须在 SQL Anywhere 服务器上安装 *pubs2_any.sql* 脚本。此脚本位于 *sample2* 子目录中。

在 Windows 系统中，进入 DOS 命令窗口并输入：

```
java IsqlApp -U dba -P password
 -S jdbc:sybase:Tds:[hostname]:[port]
 -I %JDBC_HOME%\sample2\pubs2_any.sql -c go
```

在 UNIX 系统中，输入：

```
java IsqlApp -U dba -P password
 -S jdbc:sybase:Tds:[hostname]:[port]
 -I $JDBC_HOME/sample2/pubs2_any.sql -c go
```

示例代码

以下示例代码说明了如何调用 jConnect 驱动程序、建立连接、发出 SQL 语句，以及处理结果。

```
import java.io.*;
import java.sql.*;

   public class SampleCode
   {
 public static void main(String args[])
       {
try
           {
           /*
            * Open the connection.May throw a SQLException.
            */
              DriverManager.registerDriver(
              (Driver) Class.forName(
               "com.sybase.jdbc4.jdbc.SybDriver").newInstance());

               Connection con = DriverManager.getConnection(
                  "jdbc:sybase:Tds:myserver:3767", "sa", "");
           /*
            * Create a statement object, the container for the SQL
            * statement.May throw a SQLException.
```

```
            */
              Statement stmt = con.createStatement();
            /*
             * Create a result set object by executing the query.
             * May throw a SQLException.
             */
              ResultSet rs = stmt.executeQuery("Select 1");
            /*
             * Process the result set.
             */

              if (rs.next())
              {
                 int value = rs.getInt(1);
                 System.out.println("Fetched value " + value);
              }
             rs.close()
             stmt.close()
             con.close()
          }//end try
      /*
       * Exception handling.
       */
catch (SQLException sqe)
          {
System.out.println("Unexpected exception :" +
                           sqe.toString() + ", sqlstate = " +
                               sqe.getSQLState());
              System.exit(1);
          }//end catch


          catch (Exception e)
          {
e.printStackTrace();
              System.exit(1);
          }//end catch

              System.exit(0);
      }
  }
```

附 录 B

SQL 例外与警告消息

下表列出了在使用 jConnect 时会遇到的 SQL 例外与警告消息。

SQL 状态	消息/说明/操作
010AF	严重警告：断言失败，请使用 `devclasses` 确定此严重错误的起因。消息 = _____。 **说明：** jConnect 驱动程序中的内部断言失败。 **操作：** 使用 devclasses 调试类确定显示此消息的原因并向 Sybase 技术支持部门报告此问题。
010CP	`AutoCommit` 选项已更改为 `true`。此事务上的所有待执行语句（如果有）都已提交。 **操作：** 参见消息文本。
010DF	在登录时设置数据库失败。错误消息：_____。 **说明：** jConnect 无法连接到在连接 URL 中指定的数据库。 **操作：** 确保 URL 中的数据库名称正确无误。此外，如果连接到 SQL Anywhere，请使用 SERVICENAME 连接属性指定数据库。
010DP	忽略重复的连接属性 _____。 **说明：** 某个连接属性被定义了两次。可能是在驱动程序连接属性列表中用不同的大小写形式对该连接属性定义了两次，例如 “password”和 “PASSWORD”。连接属性名不区分大小写，因此， jConnect 不会在名称相同但大小写不同的属性名间进行区分。 也可以同时在连接属性列表和 URL 中定义中连接属性。在这种情况下，连接属性列表中的属性值优先。 **操作：** 确保应用程序只定义一次连接属性。不过，或许您想利用属性列表中定义的连接属性优先于 URL 中定义的属性这一特性。这时，可以放心地忽略此警告。
010HA	服务器拒绝了使用高可用性功能的请求。请重新配置数据库，或不要请求高可用性会话。 **说明：** 服务器拒绝了 jConnect 试图建立高可用性连接的操作。 **操作：** 重新配置服务器使其支持高可用性故障切换，或者不要将 REQUEST_HA_SESSION 设置为 “true”。
010HD	这种数据库服务器不支持 `Sybase` 的高可用性故障切换。 **说明：** jConnect 试图连接到的数据库不支持高可用性故障切换。 **操作：** 应当只连接到支持高可用性故障切换的数据库服务器。
010HN	客户端未指定 `SERVICE_PRINCIPAL_NAME` 连接属性。因此， `jConnect` 使用主机名 _____ 作为服务主体名称 **操作：** 确保通过连接属性显式指定服务主体名称。

SQL 状态	消息/说明/操作
010HT	`Hostname` 属性被截断，最大长度为 `30`。 **说明：**您为 HOSTNAME 连接属性提供的字符串超过 30 个字符，或者运行 jConnect 应用程序的主机拥有长度超过 30 字节的名称。 **操作：**无需任何操作，因为这只是提醒您 jConnect 将把名称截断到 30 个字节。然而，如果希望避免此警告，应将 HOSTNAME 设置为长度小于等于 30 个字节。
010KF	服务器拒绝了 `Kerberos` 登录。这很可能是因为发生了通用安全服务 `(GSS)` 例外。请检查 `Kerberos` 环境和配置。 **操作：**检查 Kerberos 环境，确保已正确鉴定到 KDC。有关详细信息，请参见第 3 章“安全性”。
010MX	找不到有关此数据库的元数据访问程序信息。请安装 `jConnect` 文档中所述的必需表。试图检索元数据信息时出错：`_____` **说明：**服务器可能不具有返回元数据信息所需的存储过程。 **操作：**确保服务器上安装有用于提供元数据的存储过程。请参见《*jConnect for JDBC* 安装指南》第 3 章中的“安装存储过程”。
010P4	收到并忽略了一个输出参数。 **说明：**执行的查询返回一个输出参数，但应用程序的结果处理代码未读取该参数，因此将其忽略。 **操作：**如果应用程序需要输出参数数据，则必须重写该应用程序以便能获取数据。这可能需要使用 CallableStatement 来执行查询，并添加对 registerOutputParameter 和 getXXX 的调用。也可以将 DISABLE_UNPROCESSED_PARAM_WARNINGS 连接属性设置为“true”，从而禁止 jConnect 返回此警告，这样做或许还可以提高性能。
010P6	收到一行并忽略该行。 **说明：**正在处理的结果集中出现类型为 0xD1 的意外对象，该对象被忽略。 **操作：**检查生成结果集的查询，如有需要予以更正。
010PF	无法装载在 `PRELOAD_JARS` 连接属性中指定的一个或多个 `jar` 文件。 **说明：**如果在使用 DynamicClassLoader 时将 PRELOAD_JARS 连接属性设置为以逗号分隔的 *.jar* 文件名列表，就会出现此问题。当 DynamicClassLoader 打开与要装载的类所在服务器的连接时，会尝试“预装载”此连接属性中提到的所有 *.jar* 文件。如果服务器上不存在指定的一个或多个 *.jar* 文件名，则会出现上述错误消息。 **操作：**验证在应用程序的 PRELOAD_JARS 连接属性中提到的所有 *.jar* 文件是否均位于服务器上且均可访问。
010PO	属性 `LITERAL_PARAM` 已设置为 `“false”`，因为 `DYNAMIC_PREPARE` 的设置为 `“true”`。 **说明：**如要使用预编译的动态语句，则必须允许向这些语句发送参数（在语句带参数的情况下）。将 LITERAL_PARAMS 设置为“true”会强制以送往服务器的 SQL 中的文本值的形式发送所有参数。所以不能将这两个属性都设置为“true”。 **操作：**为避免出现此警告，在要使用动态 SQL 时，不要将 LITERAL_PARAMS 设置为“true”。有关详细信息，请参见第 137 页的“对动态 SQL 中的预准备语句的性能调优”。

SQL 状态	消息/说明/操作
010RC	不支持请求的 `ResultSet` 类型和并发。它们已被转换。 **说明：**您所请求的结果集类型和并发组合不受支持。已转换所请求的值。有关 jConnect 中可用结果集类型和并发的详细信息，请参见第 47 页的 “对结果集使用游标”。 **操作：**请求受支持的结果集类型和并发组合。
010SJ	找不到有关此数据库的元数据访问程序信息。请安装 `jConnect` 文档中所述的必需表。 **说明：**服务器上没有配置元数据信息。 **操作：**如果应用程序需要元数据，请安装 jConnect 中附带的用于返回元数据的存储过程（参见 jConnect for JDBC Installation Guide （《jConnect for JDBC 安装指南》）的第 3 章中的 “Installing Stored Procedures”（安装存储过程））。如果不需要元数据，请将 USE_METADATA 属性设置为 “false”。
010SK	数据库无法设置连接选项 _____。 **说明：**连接的数据库不支持应用程序尝试的操作。 **操作：**可能需要升级数据库，或者确保安装了最新版本的元数据信息。
010SL	发现此数据库的元数据访问程序信息已过时。请数据库管理员装载最新的脚本。 **说明：**服务器上的元数据信息已过时，需要更新。 **操作：**安装 jConnect 中附带的用于返回元数据的存储过程（请参见《*jConnect for JDBC* 安装指南》第 3 章中的 “安装存储过程”）。
010SM	此数据库不支持最初提议的功能集，正在重试。 **说明：**Adaptive Server Enterprise 11.9.2 版及更低版本有一个错误，该错误使它们在服务器没有客户端请求的功能时拒绝客户端登录。此警告说明 jConnect 检测到这种情况，且正以服务器可接受的最多功能数重试该连接。当 jConnect 遇到这项错误时，它会两次连接服务器。 **操作：**客户端可以放心地忽略此警告；但若想消除此警告并确保 jConnect 只进行一次连接尝试，客户端可以将 ELIMINATE_010SM 连接属性设置为 “true”。注意：在连接到 Adaptive Server 12.0 及更高版本时，不应将此属性设置为 “true”。
010SN	写入文件权限被拒绝。文件：_____.错误消息：_____ **说明：**因 VM 中的安全冲突，对 PROTOCOL_CAPTURE 连接属性中指定的文件的写入权限被拒绝。当一个小程序尝试写入指定文件时，会出现此消息。 **操作：**如果要通过小程序写入文件，必须确保该小程序可以访问目标文件系统。
010SP	无法打开文件进行写入。文件：_____.错误消息：_____ **操作：**确保文件名正确无误且文件可写。
010SQ	连接或登录被拒绝，正在通过主机/端口地址重试连接。 **说明：**CONNECTION_FAILOVER 连接属性被设置为 “true”， jConnect 无法连接到要连接的服务器列表中的某个数据库服务器。因此， jConnect 现在尝试连接到列表中的下一个服务器。 **操作：**只要 jConnect 能连接到另一数据库服务器，就无需任何操作。不过，您应当确定 jConnect 为何无法连接到导致连接警告的特定服务器。

SQL 状态	消息/说明/操作
010TP	服务器无法转换该连接的初始字符集 _____。将使用服务器提议的字符集 _____，且 jConnect 执行了转换。 **说明：**服务器无法使用最初由 jConnect 请求的字符集，已经用不同的字符集响应。jConnect 接受这一更改并执行必要的字符集转换。 本消息只用来提供信息，无其它影响。 **操作：**为避免出现此消息，可将 CHARSET 连接属性设置为服务器支持的字符集。
010TQ	jConnect 无法确定服务器的缺省字符集。这可能是元数据问题引起的。请安装 jConnect 文档中所述的必需表。该连接缺省采用 ascii_7 字符集，该字符集只能处理 0x00 到 0x7F 之间的字符。 **说明：**jConnect 无法确定服务器的缺省字符集。当发生这种情况时，只有前 127 个 ASCII 码字符能够确保得以正确转换。因此，这时 jConnect 恢复到 7 位 ASCII 码。本消息只用来提供信息，无其它影响。 **操作：**安装 jConnect 中附带的用于返回元数据的存储过程（请参见《*jConnect for JDBC* 安装指南》第 3 章中的 “安装存储过程）。
010UF	尝试执行 use database 命令失败。错误消息：_____ **说明：**jConnect 无法连接到在连接 URL 中指定的数据库。两种可能的原因是： • URL 中输入的名称有误。 • USE_METADATA 设置为 “true”（缺省设置），但未安装用于返回元数据的存储过程。结果，jConnect 尝试对 URL 中的数据库执行 use *database* 命令，但命令失败。这可能是由于您尝试访问 Adaptive Anywhere 数据库。SQL Anywhere 数据库不支持 use *database* 命令。 **操作：**确保 URL 中的数据库名称正确无误。确保服务器上安装了用于返回元数据的存储过程 （参见 jConnect for JDBC Installation Guide （《jConnect for JDBC 安装指南》）的第 3 章中的“Installing Stored Procedures”（安装存储过程）和 jConnect for JDBC Release Bulletin（《jConnect for JDBC 发行公告》）。若要尝试访问 SQL Anywhere 数据库，则不要在 URL 中指定数据库名称，或者将 USE_METADATA 设置为 “false”。
010UP	忽略无法识别的连接属性 _____。 **说明：**您试图在 URL 中设置一个 jConnect 目前无法识别的连接属性。jConnect 将忽略该无法识别的属性。 **操作：**检查应用程序中的 URL 定义，确保其只引用有效的 jConnect 驱动程序连接属性。
0100V	正在使用的 TDS 协议版本过旧。 版本：_____ **说明：**服务器不支持所需 TDS 协议版本。 jConnect 要求 5.0 版或更高版本。 **操作：**使用支持所需 TDS 版本的服务器。有关详细信息，请参见 jConnect 安装指南中的系统要求部分。
01S07	Adaptive Server 可能会舍入或截断纳秒值。 **说明：**遇到一个精度高于 1/300 秒的时间值。由于 Adaptive Server 不支持如此高的精度， jConnect 拒绝了这个值。 **操作：**确保时间值的精度不高于 1/300 秒。

SQL 状态	消息/说明/操作
01S08	全局事务中已经征用该连接。当前本地事务上所有挂起的语句 （如果有）都已经回退。 **说明：**jConnect 发出回退命令以清除所有当前本地事务。如果在调用 XAResource.start() 方法后征用全局事务，则会出现这种情况。 **操作：**如果在调用 XAResource.start() 方法之前有本地事务处于活动状态，则需要提交或回退这些本地事务。
01S09	该连接中的全局事务处于活动状态时不能使用本地事务方法 _____。 **说明：**发出警告，指出正在全局事务中执行本地操作。对连接调用 commit() 方法就属于一种本地操作。其它不能使用的操作有：rollback()、 rollback(Savepoint)、 setSavepoint()、 setSavepoint(String)、 releaseSavepoint(Savepoint) 和 setAutoCommit()。 **操作：**本地操作需要与全局事务分开。确保先完成所有本地事务及其操作，然后再启动全局事务。
01S10	本地事务方法 _____ 不能用在系统 12 以前的 XAConnection 中。 **说明：**您所使用的本地事务方法对 Sybase SQL Anywhere 12 之前的版本无效。 **操作：**不要使用该方法。
01S11	WARNING:数据可能会被截断。 **说明：**用户指定的流或 LOB 长度大于 ResultSet.updateXXX 方法中的限制 (Integer.MAX_VALUE)。 **操作：**确保长度在限制范围内。
01S12	无法继续执行 HOMOGENEOUS_BATCH 协议，后退到正常批处理。 **说明：**如果将 DYNAMIC_PREPARE 设置为 false， ASE 不会发送参数元数据。如果 HOMOGENEOUS_BATCH 设置为 true，jConnect 需要此信息进行优化。因此，jConnect 会恢复为正常批处理。 **操作：**仅以预编译的动态 SQL 预准备语句 （将 DYNAMIC_PREPARE 设置为 true）使用优化的批处理 （将 HOMOGENEOUS_BATCH 设置为 true）。
01S13	连接的 ASE 服务器不支持记录 BCP 所需要的设置选项“logbulkcopy”。在不进行记录的情况下后退到正常的批量装载与设置 ENABLE_BULK_LOAD=BCP 相等。 **说明：**连接的 ASE 版本不支持带记录功能的批量装载 （15.7 ESD #1 之前的版本）。因此， jConnect 已恢复到正常批处理。 **操作：**以正确的 ASE 版本 （15.7 ESD #1 或更高版本）使用 ENABLE_BULK_LOAD=LOG_BCP 设置。
01ZZZ	口令已过期。请以新口令设置 NEWPASSWORD 属性或使用 sp_password 更改口令。 **说明：**用于连接 ASE 的口令已过期。 **操作：**再次设置口令。
JZ001	用户名属性 “_____”过长。最大长度为 30。 **操作：**不要超出 30 字节的最大长度限制。
JZ002	口令属性 “_____”过长。最大长度为 30。 **操作：**不要超出 30 字节的最大长度限制。

SQL 状态	消息/说明/操作
JZ003	错误的 `URL` 格式。`URL：_____` **操作：**检验 URL 格式。请参见第 26 页的 “URL 连接属性参数”。 如果使用的是 PROXY 连接属性并且该属性的格式有误，则在尝试连接时会产生 JZ003 例外。 该级联代理的 PROXY 格式为： *ip_address*:*port_number* TDS 贯通服务器小程序的 PROXY 格式为： http[s]://*host*:*port*/*tunneling_servlet_alias*
JZ004	`DriverManager.getConnection(..., Properties)` 中缺少 `User name` 属性 **操作：**提供必需的用户属性。
JZ006	捕获到 `IOException：_____` **说明：**从低层检测到意外的 I/O 错误。当捕获这种 I/O 例外时，它们将通过 ERR_IO_EXCEPTION JZ006 sqlstate 作为 SQL 例外被再次抛出。这些错误通常是由网络通信问题引起的。如果 I/O 例外导致数据库连接被关闭，则 jConnect 会将 JZ0C1 例外链接到 JZ006。客户端应用程序可以查找链中的 JZ0C1 例外，以查看该连接是否仍然可用。 **操作：**检查原始 I/O 异常消息的文本，并从该处继续。
JZ008	无效的列索引值 _____。 **说明：**请求的列索引值小于 1 或大于最大可用值。 **操作：**检查对 getXXX 方法的调用和原始查询的文本，或确保调用 rs.next。
JZ009	转换时发生错误。错误消息：_____ **说明：**可能的部分原因有： • 尝试在两个不兼容的数据类型间进行转换，例如将 date 转换为 int。 • 试图将包含非数值字符的字符串转换为数值类型。 • 存在格式错误，例如 time/date 字符串格式有误。 **操作：**确保 JDBC 规范支持尝试进行的类型转换。确保字符串格式正确无误。如果字符串包含非数值字符，不要试图将其转换为数值类型。
JZ00A	为数值指定的精度和标度无效。 **说明：**使用 setBigDecimal 方法时，将 BigDecimal 值设置为以下几个精度值范围：小于 1、负数、小于标度值的精度，或大于 127 的精度值。 **操作：**检查查询并予以更正，以指定合法的精度/标度值。
JZ00B	数值溢出。 **说明：**试图将 BigInteger 作为 TDS 数值发送，而该值过大；或者试图将 Java long 作为 int 发送，而该值过大。 **操作：**不能在 Sybase 中存储这些值。对于 long，请考虑使用 Sybase 数值类型。尚无法解决 Bignum 的问题。

SQL 状态	消息/说明/操作
JZ00C	指定的精度和标度不匹配数值 _____。 **说明：**在使用 setBigDecimal 方法时， BigDecimal 值的精度或标度超出指定范围。 **操作：**确保指定的精度和标度与 BigDecimal 值相匹配。
JZ00E	视图在已调用 `setCursorName()` 的语句中调用 `execute()` 或 `executeUpdate()`。 **操作：**不要尝试在已设置游标名称的语句中调用 execute 或 executeUpdate。请单独使用一条语句删除或更新游标。有关详细信息，请参见第 47 页的 “对结果集使用游标”
JZ00F	已通过 `setCursorName()` 设置了游标名。 **操作：**不要为同一语句设置两次游标名称。关闭当前游标语句的结果集。
JZ00G	未为此行更新设置任何列值。 **说明：**您试图更新一行，但未在该行中更改任何列值。 **操作：**要更改行中的列值，应在调用 updateRow 前调用 updateXX 方法。
JZ00H	结果集不可更新。使用 `Statement.setResultSetConcurrencyType()`。 **操作：**要将结果集从只读更改为可更新，应使用 Statement.setResultSetConcurrencyType 方法或向 SQL select 语句添加 for update 子句。
JZ00I	无效精度。指定的精度必须 `>=0`。 **说明：**标度值必须大于零。 **操作：**确保标度值非负。
JZ00L	登录失败。检查链接到此例外的 `SQLWarnings` 来查找原因。 **操作：**查看消息文本；根据给出的登录失败原因继续操作。
JZ00M	登录超时。检查数据库服务器是否在指定的主机和端口号上运行。同时检查可能导致数据库服务器挂起的其它情况（例如 `tempdb` 已满）。 **操作：**按照错误消息中提供的建议进行操作。
JZ010	无法将 `Object` 值非序列化。错误文本：_____ **操作：**确保数据库中的 Java 对象实施 Serializable 接口并位于本地 CLASSPATH 变量中。
JZ011	在分析数值连接属性 _____ 时发生数字格式例外。 **说明：**为数值连接属性指定了非整数值。 **操作：**为连接属性指定整数值。
JZ012	内部错误。请将错误报告给 `Sybase` 技术支持。连接属性 _____ 的访问类型有误。 **操作：**与 Sybase 技术支持部门联系。
JZ013	获取 `JNDI` 条目时出错：_____ **操作：**更正 JNDI URL，或在目录服务中创建一个新条目。
JZ014	您不能运行 `setTransactionIsolation(Connection.TRANSACTION_NONE)`。这一级别不能进行设置，只能由服务器返回。 **操作：**检查调用 Connection.setTransactionIsolation 的应用程序代码，并检验传递给方法的值。

SQL 状态	消息/说明/操作
JZ015	为 GSSMANAGER_CLASS 连接属性设置了非法值。属性值必须是扩展 org.ietf.jgss.GSSManager 的 String 或 Object。 **操作：**检查为 GSSMANAGER_CLASS 属性设置的值。
JZ017	保存点无效。 **说明：**为回退或释放指定的保存点不存在。 **操作：**检查查询并予以更正，指定存在的保存点。
JZ018	无法将此方法应用于此类型的保存点。 **说明：**getSavepointId() 方法不适用于命名保存点 （没有 ID）， getSavepointName() 方法不适用于未命名保存点 （没有名称）。 **操作：**检查查询并予以更正。
JZ019	获取 SERVERNAME 时出错：_____. **说明：**用 jdbc:sybase:jndi:file 设置的 URL 未指定 *sql.ini* 文件 (Windows) 或 *interfaces* 文件 (UNIX) 或者服务器名。 **操作：**检查 URL 命令并予以更正。
JZ021	未找到指定的_____文件。 **说明：**未找到在连接 URL 中指定的 *sql.ini* 文件 (Windows) 或 *interfaces* 文件 (UNIX)。 **操作：**检查连接 URL 并予以更正。
JZ022	指定的_____文件格式未知。 **说明：***sql.ini* 文件 (Windows) 或 *interfaces* 文件 (UNIX) 中的连接 URL 字符串格式不正确。 **操作：**检查连接 URL 字符串并予以更正。
JZ024	指定服务器：_____在 interfaces/sql.ini 文件中没有条目：_____。 **说明：**在连接 URL 中指定的服务器名在 *sql.ini* 文件 (Windows) 或 *interfaces* 文件 (UNIX) 中没有对应条目。 **操作：**检查连接 URL 字符串并予以更正。
JZ025	interfaces/sql.ini 中指定服务器的 TLI 格式无效。 **说明：**TLI 格式 *interfaces* 文件 (UNIX) 中的服务器详细信息不正确。 **操作：**检查设置并予以更正。
JZ026	interfaces/sql.ini 文件：_____中服务器：_____的指定协议：_____不受支持。 **说明：**在 *sql.ini* 文件 (Windows) 或 *interfaces* 文件 (UNIX) 中指定的协议不受支持 （tli、tcp 和 nlwnsck 以外的协议）。 **操作：**仅指定受支持的协议。
JZ027	interfaces/sql.ini 文件：_____中服务器：_____的指定 SECMECH 条目：_____不受支持。 **说明：**在 Kerberos 连接 URL 中指定的值无效。 **操作：**检查 URL 并予以更正。

SQL 状态	消息/说明/操作
JZ028	为 JCE_PROVIDER_CLASS 连接属性设置的值非法。属性值必须是作为字符串或 java.security.Provider 实例传递的完全限定的提供程序类名称。 **操作：**指定合法值。
JZ029	查找 ALTERNATE_SERVER_NAME _____，(_____) 的地址时出错。 **说明：**jConnect 无法使用 SQL Anywhere UDP 发现协议查找以 ALTERNATE_SERVER_NAME 属性指定的服务器。 **操作：**检查以 ALTERNATE_SERVER_NAME 连接属性指定的服务器名并予以更正。
JZ030	方法 _____ 不受支持。 **说明：**您所使用的公共 API 目前不受支持。
JZ031	打开 _____ 的对象失败。 **说明：**jConnect 无法打开自定义类的对象，因为该自定义类不在类路径中。 **操作：**将类添加到类路径中。
JZ032	Date 或 Timestamp 参数超出 BigDateTime/BigTime 范围。服务器只能支持 0001/01/01 12:00:00:000000AM 到 9999/12/31 11:59:59.999999PM 之间的 BigDateTime 值或 12:00:00:000000AM 到 11:59:59.999999PM 之间的 BigTime 值。 **操作：**参见消息文本。
JZ033	服务器返回未知的 Blob 类型。 **说明：**jConnect 无法将列的 ASE 数据类型映射到 BLOB 数据类型。 **操作：**确保 ASE 列可以转换为 BLOB 数据类型。
JZ034	连接的服务器无法处理大对象 [LOB]。 **说明：**连接的 ASE 版本不支持 LOB 方法。 **操作：**使用常规流方法访问 LOB。
JZ035	要处理大对象 [LOB]，请将连接属性 “ENABLE_LOB_LOCATOR”设置为 true。 **操作：**参见消息文本。
JZ036	对此大对象 [LOB] 的引用在数据库中不再有效。检查是否调用了 free() 或检查事务是否已结束。 **操作：**参见消息文本。
JZ037	offset/position/start 的值应该在 [1, len] 的范围内，其中 len 表示大对象 [LOB] 的长度。 **操作：**参见消息文本。
JZ038	对象长度应 >= 0。 **说明：**在操作 （getBytes、 truncate、 getSubString）中对 LOB 对象指定的大小为负 **操作：**仅指定非负值。

SQL 状态	消息/说明/操作
JZ040	_____操作失败。 _____已关闭。 **说明：**读取 （写入）操作失败，因为输入流或 LOB 读取程序 （输出流或 LOB 写入程序）已关闭。 **操作：**检查应用程序找出冲突原因并予以更正。
JZ041	_____操作在_____上失败。 **说明：**read(write)(available()) 操作失败，因为输入流或读取程序 （输出流或写入程序）（输入流）已关闭。 **操作：**检查应用程序找出冲突原因并予以更正。
JZ042	如果 ENABLE_LOB_LOCATOR 和 HOMOGENEOUS_BATCH 设置为 TRUE，大对象 setter 与其它 setter 便不能混合。java.sql.Types _____ 已和 java.sql.Types _____ 混合。 **操作：**参见消息文本。
JZ043	对于 “ENABLE_BULK_LOAD 属性”的任何可能变体， LOB 对象均不受支持，而为 false。请考虑使用其它 setter API 插入数据。 **操作：**参见消息文本。
JZ044	在 SEND_BATCHPARAMS_IMMEDIATE 设置为 TRUE 的情况下，无法在批处理中创建服务器端的定位符。尝试使用客户端的 LOB 或将 SEND_BATCHPARAMS_IMMEDIATE 设置为 FALSE。 **操作：**参见消息文本。
JZ0BD	方法参数中使用了越界值或无效值。 **操作：**检验方法中的参数值是否正确无误。
JZ0BI	**消息：**setFetchSize：获取大小应该在以下范围内设置：0 <= 行数 <= （结果集中的最大行数）。 **说明：**客户端应用程序在调用 setFetchSize 时使用了无效的行数。 **操作：**检验调用 setFetchSize 时使用的参数值是否位于上述值范围内。
JZ0BJ	**消息：**为 IMPLICIT_CURSOR_FETCH_SIZE 连接属性设置的值必须为 > 0。 **操作：**参见消息文本。
JZ0BP	批处理更新语句中不允许使用输出参数。 **操作：**检查应用程序代码，并确保未尝试在批处理中声明输出参数。
JZ0BR	游标所在的行不支持 _____ 方法。 **说明：**尝试调用的 ResultSet 方法对当前行位置无效 （例如，调用 insertRow 时游标不在插入行上）。 **操作：**不要调用对当前行位置无效的 ResultSet 方法。
JZ0BS	不支持批处理语句。 **操作：**在数据库中通过最新版本安装或更新 jConnect 元数据存储过程。
JZ0BT	_____ 类型的 ResultSet 不支持 _____ 方法。 **说明：**您尝试调用的 ResultSet 方法对此类型的 ResultSet 无效。 **操作：**不要尝试调用对 ResultSet 类型无效的 ResultSet 方法。

SQL 状态	消息/说明/操作
JZ0C0	连接已关闭。 **说明：**应用程序已对此连接对象调用 Connection.close ，无法再使用此连接。 **操作：**修正代码，在连接关闭时清空连接对象引用。
JZ0C1	出现 `IOException`，此错误将使连接关闭。 **说明：**出现了不可恢复的 IOException 例外，连接已被关闭。该连接不能再用于其它任何数据库操作。如果出现此例外，则它总会与 JZ006 例外一起出现在一个例外链中（前面解释过）。 **操作：**确定导致连接中断的 IOException 的起因。
JZ0CL	在使用 `PRELOAD_JARS` 属性时必须定义 `CLASS_LOADER` 属性。 **操作：**在将 PRELOAD_JARS 设置为非空值时，请务必指定一个 CLASS_LOADER。
JZ0D4	`Sybase JDBC URL` _____ 中无法识别的协议。 **说明：**您使用非 TDS 协议指定了连接 URL，但目前 jConnect 只支持 TDS 协议。 **操作：**检查 URL 定义。如果 URL 指定 TDS 作为子协议，请确保该条目使用以下格式和大小写形式： jdbc:sybase:Tds:*host*:*port* 如果 URL 指定 JNDI 作为子协议，请确保它的开头为： jdbc:sybase:jndi:
JZ0D5	装载协议 _____ 时出错。 **操作：**检查 CLASSPATH 系统变量的设置。
JZ0D6	`setVersion` 中指定了无法识别的版本号 _____。选择 `SybDriver.VERSION_*` 值之一，并确保所用 `jConnect` 版本等于或高于指定的版本。 **操作：**参见消息文本。
JZ0D7	装载 `url` 提供程序 _____ 时出错。错误消息：_____ **操作：**检查 JNDI URL，确保其正确无误。
JZ0D8	初始化 `URL` 提供程序时出错：_____ **操作：**检查 JNDI URL，确保其正确无误。
JZ0EM	数据结束。 **操作：**请向 Sybase 技术支持部门报告此错误。
JZ0F1	请求了 `Sybase` 高可用性故障切换连接，但协同服务器地址丢失。 **说明：**将 REQUEST_HA_SESSION 连接属性设置为 “true” 时，必须同时指定故障切换服务器。 **操作：**可以用 SECONDARY_SERVER_HOSTPORT 连接属性指定辅助服务器，或者用 JNDI 设置辅助服务器（请参见第 40 页的 “实施高可用性故障切换支持”）。

SQL 状态	消息/说明/操作
JZ0F2	已发生 Sybase 高可用性故障切换。当前事务已中止，但连接仍可用。请重试事务。 **说明：**之前连接的后端数据库服务器已关闭，但您已经切换到一个辅助服务器。数据库连接仍可用。 **操作：**客户端代码应捕获这一例外，再从上次提交点重新启动事务。如果正确处理了此例外，就可以对同一连接对象继续执行 JDBC 调用。
JZ0FP	为参数 _____ 传递的值不正确 **说明：**为当前结果集的状态指定的参数值无效。 **操作：**确保指定合法的有效值 （CLOSE_CURRENT_RESULT、KEEP_CURRENT_RESULT、 CLOSE_ALL_RESULTS）。
JZ0GC	将 _____ 作为 GSSManager 转换时出错。请检查设置给 GSSMANAGER_CLASS 连接属性的值。该值必须为一个 String，它要指定 GSSManager 实现的完全限定的类名称。或者必须是一个扩展 org.ietf.jgss.GSSManager 的 Object。 **操作：**参见消息文本。
JZ0GK	_____阵列不得为空值，且必须仅包含一个键。 **说明：**自动生成的键列名/索引阵列为 NULL 或含有多个键。在阵列中只允许使用一个键，因为它与 IDENTITY 列相关。 **操作：**检查查询并予以更正。
JZ0GN	将类 ____ 作为 GSSManager 例示时出错。例外是 _____。请检查 CLASSPATH 并确保 GSSMANAGER_CLASS 属性值引用了 GSSManager 实现的完全限定类名称。 **操作：**确保 CLASSPATH 环境变量包含第三方 GSSManager 实现所需的所有 *.jar* 文件。
JZ0GS	出现通用安全服务 API 例外。主要错误代码为 ____。主要错误消息为 ____。次要错误代码为 ____。次要错误消息为 _____。 **操作：**检查主要和次要错误代码及错误消息。检查 Kerberos 配置。有关详细信息，请参见第 3 章 “安全性”。
JZ0H0	无法启动事件处理程序的线程；事件名称 = _____。 **操作：**请向 Sybase 技术支持部门报告此错误。
JZ0H1	收到事件通知但没有找到事件处理程序；事件名称 = _____。 **操作：**请向 Sybase 技术支持部门报告此错误。
JZ0HC	分析十六进制数时遇到非法字符 “_____”。 **说明：**用于表示二进制值的字符串包含了十六进制数限定范围 （0 - 9、 a - f）以外的字符。 **操作：**检查字符串中的字符值，确保它们在要求的范围内。
JZ0I3	未知属性。此消息表明存在内部产品问题。请向 Sybase 技术支持部门报告此错误。 **操作：**表明存在内部产品问题。请向 Sybase 技术支持部门报告此错误。
JZ0I5	指定了无法识别的 CHARSET 属性：_____. **说明：**您为 CHARSET 连接属性指定了一个不受支持的字符集编码。 **操作：**为该连接属性输入有效的字符集代码。请参见第 35 页的 “jConnect 字符集转换程序”。

SQL 状态	消息/说明/操作
JZ0I6	将 UNICODE 转换为服务器使用的字符集时出错。错误消息：_____ **操作：**在 jConnect 客户端为 CHARSET 连接属性选择另外的字符集代码，所选字符集代码应支持要发送到服务器的所有字符。可能也需要在服务器上安装另一个字符集。此外，如果您使用的是 jConnect 6.05 或更高版本以及 Adaptive Server Enterprise 12.5 或更高版本，则可以在向服务器发送数据时使用 unichar/univarchar 数据类型。请参见第 33 页的 “使用 jConnect 传递 Unicode 数据”。
JZ0I7	代理网关没有响应。 **说明：**无法建立连接，因为以 PROXY 连接属性指定的代理网关无响应。 **操作：**检查 PROXY 设置并予以更正。
JZ0I8	代理网关连接被拒绝。网关响应：%1s。 **说明：**代理网关连接因指定原因被拒绝。 **操作：**检查代理网关设置。
JZ0I9	此 InputStream 已关闭。 **说明：**您试图读取一个来自 getAsciiStream、 getUnicodeStream 或 getBinaryStream 的 InputStream，但该 InputStream 已关闭。该流被关闭的原因可能是您移到了另一列或取消了结果集，且没有足够的资源来缓存数据。 **操作：**增加高速缓存大小，或按顺序读取列。
JZ0IA	试图发送 _____ 时出现截断错误。 **说明：**在发送字符串之前进行字符集转换时发生截断错误。被转换的字符串的长度超过了分配给它的大小。 **操作：**在 jConnect 客户端为 CHARSET 连接属性选择其它字符集代码，所选字符集代码应支持要发送到服务器的所有字符。可能也需要在服务器上安装另一个字符集。
JZ0IB	服务器的_____缺省字符集不能映射到客户端 Java 环境中的可用编码。由于 jConnect 无法进行客户端转换，因此连接无法使用，正在关闭。尝试使用较新的 Java 版本，或尝试在类路径中包含 Java 安装的 i18n.jar 或 charsets.jar 文件。 **操作：**参见消息文本。
JZ0IR	不能对已通过 java.io.Reader 在结果集中更新过的列调用 getXXX。 **操作：**删除对使用 Reader 更新过的 ResultSet 列的 getXXX 调用。
JZ0IS	不能对已在结果集中更新过的列调用 getXXXStream。 **说明：**更新结果集中的一个列之后，您试图用以下 SybResultSet 方法之一读取更新后的列值：getAsciiStream、 getUnicodeStream、 getBinaryStream。 jConnect 不支持这种用法。 **操作：**不要试图从正在更新的列中获取输入流。
JZ0J0	偏移和/或长度值超出了实际的文本/图像长度。 **操作：**检查所用的偏移和/或长度值是否正确无误。
JZ0LA	实例化 Cipher 对象失败。所装载的任何 JCE 提供程序均不实施转换 %1s。 **说明：**以 JCE_PROVIDER_CLASS 连接属性指定的 JCE 提供程序的实施不在类路径中。 **操作：**确保实施位于类路径中。

SQL 状态	消息/说明/操作
JZ0LC	不能对正在使用语言游标读取行的 ResultSet 调用 _____ 方法。尝试将 LANGUAGE_CURSOR 连接属性设置为 false。 **说明：**应用程序试图对一个通过语言游标创建的 ResultSet 调用某种 ResultSet 游标滚动方法。 **操作：**参见错误消息。
JZ0MD	ResultSet 元数据不可用。 **说明：**J2EE_TCK_COMPLIANT 属性已设置，但结果集元数据不可用。 **操作：**安装元数据存储过程。
JZ0NC	wasNull 调用前没有其它调用来获取列。 **说明：**只能在获取列的调用 （例如 getInt 或者 getBinaryStream）后调用 wasNull。 **操作：**更改代码，移动对 wasNull 的调用。
JZ0NE	错误的 URL 格式。 URL：_____.错误消息：_____ **操作：**检查 URL 的格式。确保端口号仅包含数值字符。
JZ0NF	无法装载 SybSocketFactory。确保类名称拼写无误，软件包被完全指定，类在类路径中可用，而且有公共的零参数构造函数。 **操作：**参见消息文本。
JZ0NK	生成的键不可用，因为使用了 Statement.NO_GENERATED_KEYS 或者未自动生成任何键。 **说明：**getGeneratedKeys() 方法无法返回自动生成的键，因为语句是通过 .NO_GENERATED_KEYS 执行的或者该语句未产生自动生成的键。 **操作：**仅对以 .RETURN_GENERATED_KEYS 执行的语句或应该自动生成键的语句使用 getGeneratedKeys()。
JZ0NS	方法 _____ 不受支持，不应调用。 **操作：**参见消息文本。
JZ0P1	异常的结果类型。 **说明：**数据库返回的结果不能由语句返回给应用程序，或者应用程序此时需要的不是该结果。通常这表明应用程序在错误地使用 JDBC 执行查询或存储过程。如果 JDBC 应用程序连接到一个 Open Server 应用程序，这可能表明该 Open Server 应用程序中有错误，该错误导致 Open Server 发送异常结果序列。 **操作：**使用 com.sybase.utils.Debug(true, "ALL") 调试工具，尝试确定异常结果及其发生原因。
JZ0P4	协议错误。此消息表明存在内部产品问题。请向 Sybase 技术支持部门报告此错误。 **操作：**参见消息文本。

SQL 状态	消息/说明/操作
JZ0P7	没有对列进行高速缓存；请使用 RE-READABLE_COLUMNS 属性。 **说明：**当 REPEAT_READ 连接属性设置为 “false”时，尝试再次读取列或以错误顺序读取列。 当 REPEAT_READ 为 “false”时，只能读取一次某行的列值，且必须按列索引升序顺序读取。例如，在读取一行的第 3 列后，就不能再次读取该列的值，也不能读取该行第 2 列。 **操作：**将 REPEAT_READ 设置为 “true”，或者不要试图重新读取某列值并确保按列索引升序顺序读取列。
JZ0P8	请求的 RSMDA 列类型名称未知。 **说明：**jConnect 无法在 ResultSetMetaData.getColumnTypeName 方法中确定列类型的名称。 **操作：**确保数据库具有用于元数据的最新存储过程。
JZ0P9	检测到 COMPUTE BY 查询。该类型的结果不受支持，已被取消。 **说明：**执行的查询返回的 COMPUTE 结果不受 jConnect 支持。 **操作：**更改查询或存储过程，使其不使用 COMPUTE BY。
JZ0PA	查询已被取消，相同的响应被放弃。 **说明：**可能有连接中的另一语句执行了取消操作。 **操作：**检查各语句中的 SQL 例外和警告链，以找出原因。
JZ0PB	服务器不支持请求的操作。 **说明：**当 jConnect 建立与服务器的连接时，它会将需要服务器支持的功能通知服务器，服务器再将自身支持的功能通知 jConnect。当应用程序请求的操作在最初的功能协商中被拒绝时，会发出此错误消息。 例如，如果数据库不支持动态 SQL 语句预编译，而代码调用 SybConnection.prepareStatement(*sql_stmt*, *dynamic*)，并且 *dynamic* 设置为 “true”，那么 jConnect 将生成此消息。 **操作：**修改代码，使其不请求不受支持的功能。
JZ0PC	查询中参数的数量和大小要求宽表支持。但是服务器不提供这种支持，或者查询请求不是在登录序列中发出的。如果想请求宽表支持，请尝试将 JCONNECT_VERSION 属性设置为 >=6。 **说明：**您正试图执行一个拥有大量参数的语句，但服务器没有进行相应配置来处理如此之多的参数。能引起此例外的参数数量因传送的数据类型不同而变化。在传送 481 或更少的参数时，肯定不会产生这种例外。 **操作：**必须在 Adaptive Server 12.5 或更高版本服务器中运行此查询。在连接到数据库时，请将 JCONNECT_VERSION 属性设置为 “6”。

SQL 状态	消息/说明/操作
JZ0PD	动态准备中的查询过大，需要宽表支持。但是服务器不提供这种支持，或者查询请求不是在登录序列中发出的。如果想请求宽表支持，请尝试将 JCONNECT_VERSION 属性设置为 >=6。 **说明：**您正试图执行一个拥有大量参数的动态预准备语句，但服务器没有进行相应配置来处理如此之多的参数。 **操作：**必须在 Adaptive Server 12.5 或更高版本服务器中运行此查询。在连接到数据库时，请将 JCONNECT_VERSION 属性设置为 “6”。
JZ0PE	游标声明中列的数目过大，或者游标声明本身过大，需要宽表支持。但是服务器不提供这种支持，或者查询请求不是在登录序列中发出的。如果想请求宽表支持，请尝试将 JCONNECT_VERSION 属性设置为 >=6。 **说明：**当 SELECT 语句试图从 255 个以上的列中返回数据时会出现此错误。或者 SELECT 语句的实际长度很大时 （超过大约 65500 个字符）也会出现此错误。 **操作：**必须在 Adaptive Server 12.5 或更高版本中运行此查询。在连接到数据库时，请将 JCONNECT_VERSION 属性设置为 “6”。
JZ0PN	指定的端口号 _____ 超出范围。端口号必须满足以下条件：0<= 端口号 <=65535。 **操作：**检查数据库 URL 中指定的端口号。
JZ0R0	结果集已关闭。 **说明：**已对结果集对象调用 ResultSet.close 方法，该结果集不能另作他用。 **操作：**修正代码，从而在每当关闭结果集时将 ResultSet 对象引用设置为 null。
JZ0R1	结果集处于 IDLE 状态，因为目前没有访问行。 **说明：**应用程序调用了 ResultSet.getXXX 列数据检索方法之一，但没有当前行；应用程序没有调用 ResultSet.next，或 ResultSet.next返回 “false”，指示没有数据存在。 **操作：**确保先将 rs.next 设置为 “true”后再调用 rs.getXXX。
JZ0R2	查询的结果集不存在。 **说明：**您使用了Statement.executeQuery，但语句没有返回任何行。 **操作：**对未返回任何行的语句使用 executeUpdate。
JZ0R3	列处于 DEAD 状态。这是内部错误。请将错误报告给 Sybase 技术支持。 **操作：**参见消息文本。
JZ0R4	列中不含文本指针。该列不是文本/图像列，或该列为 NULL。 **说明：**不能更新值为 null 的 text/image 列。空 text/image 列中不含文本指针。 **操作：**确保不要试图更新或获取指向不支持文本/图像数据的列的文本指针。确保不要更新值为 null 的 text/image 列。先插入数据，再进行更新。
JZ0R5	ResultSet 当前的定位超出了最后一行。此状态下不能通过执行 get* 操作读取数据。 **说明：**应用程序已将 ResultSet 行指针移到最后一行之后。该位置没有数据可读，所有 get* 操作均为非法。 **操作：**更改代码，使其在 ResultSet 定位在最后一行之后时不再读取列数据。

SQL 状态	消息/说明/操作
JZ0RD	不能对已通过 `deleteRow()` 方法删除的行调用任何 `ResultSet.get*` 方法。 **说明：**应用程序试图从已经删除的行中检索数据。没有有效数据可供检索。 **操作：**更改代码，使应用程序不从已删除的行中检索数据。
JZ0RM	不能在调用 `updateRow` 或 `deleteRow` 之后调用 `refreshRow`。 **说明：**在用 SybCursorResult.updateRow 更新数据库中的行之后，或用 SybCursorResult.deleteRow 删除行之后，又用 SybCursorResult.refreshRow 刷新了数据库中的行。 **操作：**在更新或删除数据库中的行以后，不要再刷新该行。
JZ0S0	语句状态机：语句处于 `BUSY` 状态。 **说明：**出现此错误的唯一情况是在使用 Statement.setCursorname 方法时。如果应用程序试图在语句已在使用中且需要读取非游标结果时设置游标名称，则会出现此错误。 **操作：**在用语句执行任何查询之前先为其设置游标名称，或者在设置游标名称之前调用 Statement.cancel，以确保该语句不处于繁忙状态。
JZ0S1	语句状态机：正试图在 `IDLE` 语句中执行 `FETCH`。 **说明：**语句中发生内部错误。 **操作：**关闭该语句，打开另一个语句。
JZ0S2	语句对象已关闭。 **说明：**已对语句对象调用 Statement.close 方法，该语句不能另作他用。 **操作：**修正应用程序，从而每当关闭语句时都将语句对象引用设置为 null。
JZ0S3	此子类中不能使用继承的方法 _____。 **说明：**PreparedStatement 不支持 executeQuery(String)°¢executeUpdate(String) 和 execute(String)。 **操作：**要传递查询字符串，应使用 Statement 而非 PreparedStatement。
JZ0S4	不能执行空 （零长度）查询。 **操作：**不要执行空查询 ("")。
JZ0S5	该连接中的全局事务处于活动状态时不能使用本地事务方法 _____。 **说明：**使用分布式事务时可能会出现该例外。 **操作：**有关诊断此问题的详细信息，请参见 *JDBC 2.0 Optional Package* （《*JDBC 2.0* 选件工具包》）（旧称 *JDBC 2.0 Standard Extension API* （《*JDBC 2.0* 标准扩展 *API*》））中的第 7 章 “Distributed Transactions”（分布式事务）。
JZ0S6	本地事务方法 _____ 不能用在系统 `12` 以前的 `XAConnection` 中。 **说明：**使用分布式事务时可能会出现该例外。 **操作：**有关诊断此问题的详细信息，请参见 *JDBC 2.0 Optional Package* （《*JDBC 2.0* 选件工具包》）（旧称 JDBC 2.0 Standard Extension API （《JDBC 2.0 标准扩展 API》））中的第 7 章 “Distributed Transactions”（分布式事务）。
JZ0S8	`SQL` 查询中的转义序列格式有误：‘_____’ . **说明：**此错误由错误的转义语法引起。 **操作：**检查 JDBC 文档的语法是否正确。

SQL 状态	消息/说明/操作
JZ0S9	不能执行空 （零长度）查询。 **操作：**不要执行空查询 ("")。
JZ0SA	预准备语句：未设置输入参数，索引：_____. **操作：**确保为每个输入参数赋值。
JZ0SB	参数索引超出范围：_____. **说明：**您尝试获取、设置或注册一个参数，但参数数目已经超越了最大数目限制。 **操作：**检查查询中的参数数量。
JZ0SC	可调用语句：尝试将返回状态设置为 `InParameter`。 **说明：**您已经准备调用一个用于返回状态的存储过程，但却试图设置参数 1，该参数是返回状态。 **操作：**此类调用中可供设置的参数是从 2 开始的。
JZ0SD	未为输出参数找到注册的参数。 **说明：**这表明应用程序存在逻辑错误。您尝试对参数调用 getXXX 或 wasNull，但尚未读取任何参数，或没有输出参数。 **操作：**检查以确保应用程序已经对 CallableStatement 注册了输出参数，且语句已被执行，同时输出参数已读取。
JZ0SE	为 `setObject()` 指定了无效的对象类型。 **说明：**向 PreparedStatement.setObject 传递了非法类型参数。 **操作：**检查 JDBC 文档。参数必须是来自 java.sql.Types 的一个常数。
JZ0SF	未找到任何参数。是否已发送查询？ **说明：**您试图对不含参数的语句设置参数。 **操作：**在设置参数前确保查询已发送。
JZ0SG	RPC 未返回与应用程序注册的参数相同数量的输出参数。 **说明：**如果通过调用 CallableStatement.registerOutParam 注册的参数多于在存储过程中声明为 “OUTPUT” 的参数，则会出现此错误。有关详细信息，请参见第 133 页的 “RPC 返回比已注册参数少的输出参数”。 **操作：**检查存储过程和 registerOutParameter 调用。确保已将所有相应参数声明为 “OUTPUT”。注意内容如下的代码行： `create procedure yourproc (@p1 int OUTPUT, ...` **注释** 如果在使用 SQL Anywhere 时出现此错误，请升级到 SQL Anywhere 版本 5.5.04。
JZ0SH	使用了静态函数转义，但此服务器上找不到元数据访问程序信息。 **操作：**在使用静态函数转义之前，先安装元数据访问程序信息。
JZ0SI	此服务器不支持使用的静态函数转义 _____。 **操作：**不要使用此转义。
JZ0SJ	找不到有关此数据库的元数据访问程序信息。 **操作：**在进行元数据调用前，先安装元数据信息。

SQL 状态	消息/说明/操作
JZ0SK	此类数据库服务器不支持 `oj` 转义。解决方法：使用服务器特定的外部连接语法 （如果支持）。查阅服务器文档。 **操作：**读取错误消息。同时，请安装最新版本的 jConnect 元数据。
JZ0SL	不支持的 `SQL` 类型 `_____`。 **说明：**jConnect 不支持应用程序声明的参数类型。 **操作：**若有可能，尝试用其它类型声明该参数。不要使用 Types.NULL 或 PreparedStatement.setObject (null)。
JZ0SM	`jConnect` 未能执行存储过程，因为发送参数时发生了错误。导致此问题的原因可能是服务器不支持特定的数据类型，或者 `jConnect` 未在连接时为该数据类型请求支持。可以尝试将 `JCONNECT_VERSION` 连接属性设置为更高的值。或者，若有可能，尝试以语言语句的形式发送过程执行指令。
JZ0SN	`setMaxFieldSize`：字段长度不能为负。 **操作：**在调用 setMaxFieldSize 时使用正值或零 （无限制）。
JZ0SO	无效的 `ResultSet` 并发类型：`_____`。 **操作：**确保声明的并发为 ResultSet.CONCUR_READ_ONLY 或 ResultSet.CONCUR_UPDATABLE。
JZ0SP	无效的 `ResultSet` 类型：`_____`。 **操作：**检查声明的 ResultSet 类型是否为 ResultSet.TYPE_FORWARD_ONLY 或 ResultSet.TYPE_SCROLL_INSENSITIVE。 jConnect 不支持 ResultSet.TYPE_SCROLL_SENSITIVE ResultSet 类型。
JZ0SQ	无效的 `UDT` 类型 `_____`。 **说明：**在调用 DatabaseMetaData.getUDTs 方法时，如果用户定义的类型不是 Types.JAVA_OBJECT、 Types.STRUCT 或 Types.DISTINCT， jConnect 便会抛出此异常。 **操作：**使用上述三种 UDT 之一。
JZ0SR	`setMaxRows`：最大行数不能为负。 **操作：**在调用 setMaxRows 时使用正值或零 （无限制）。
JZ0SS	`setQueryTimeout`：查询超时时间不能为负。
JZ0ST	`jConnect` 不能在查询中以文字参数的形式发送 `Java` 对象。执行此查询时，请确保数据库服务器支持 `Java` 对象，且 `LITERAL_PARAMS` 连接属性设置为 `false`。
JZ0SU	`Date` 或 `Timestamp` 参数设置成 `_____` 年，但服务器只支持 `_____` 至 `_____` 之间的年份值。如果在 `Adaptive Server Anywhere` 上将数据发往日期、时间戳列或参数，最好以字符串的形式发送，让服务器进行转换。 **说明：**Adaptive Server Enterprise 和 SQL Anywhere 对 datetime 和 date 的取值范围有不同的规定。 datetime 值必须大于等于 1753，而 date 数据类型可以采用大于等于 1 的年份值。 **操作：**确保发送的 date/timestamp 值在可接受的范围内。
JZ0SV	不允许在同一个 `CallableStatement` 中同时按名称和索引设置参数。 **说明：**CallableStatement 拥有按名称和索引 （顺序位置）指定的两种参数。混合使用无效。 **操作：**仅按名称或仅按索引 （顺序位置）指定参数。

SQL 状态	消息/说明/操作
JZ0SW	ResultSet 可保持性类型无效：_____. **说明：**您以 setHoldability() 方法指定的值无效。 **操作：**仅使用合法值 – HOLD_CURSORS_OVER_COMMIT 或 CLOSE_CURSORS_AT_COMMIT。
JZ0T2	监听器线程读错误。 **操作：**检查网络通信。
JZ0T3	读操作超时。 **说明：**超出了为读取查询响应所分配的时限。 **操作：**调用 Statement.setQueryTimeout 增大超时期限。
JZ0T4	写操作超时。超时期限 （毫秒）：_____. **说明：**超出了为发送请求所分配的时限。 **操作：**调用 Statement.setQueryTimeout 增大超时期限。
JZ0T5	用于存储响应的高速缓存已满。 **操作：**为 STREAM_CACHE_SIZE 连接属性使用缺省值或更大的值。
JZ0T6	读取贯通 TDS URL 时出错。 **说明：**读取 URL 标头时贯通协议失败。 **操作：**检查为连接定义的 URL。
JZ0T7	监听器线程读错误 -- 捕获 ThreadDeath。检查网络连接。 **操作：**检查网络连接，尝试重新运行应用程序。如果线程仍被中止，请联系 Sybase 技术支持。
JZ0T8	收到未知请求的数据。请向 Sybase 技术支持部门报告此错误。
JZ0T9	发送请求不同步。请向 Sybase 技术支持部门报告此错误。 **操作：**参见消息文本。
JZ0TC	尝试在两种类型间进行非法转换。 **说明：**在 Java 类型和 SQL 类型间进行的转换失败。 **操作：**检查请求的类型转换，确保 JDBC 规范支持该转换。
JZ0TD	捕获 ThreadDeath。 **说明：**在 jConnect 执行定时 IO 操作时，调用的应用程序线程被注销。 **操作：**检查应用程序代码找出冲突并予以更正。
JZ0TE	尝试在两种类型间进行非法转换。有效的数据库类型为：‘_____.’ **说明：**数据库列的数据类型和 ResultSet.getXXX 调用中请求的数据类型不能进行隐式转换。 **操作：**使用错误消息中列出的有效数据类型之一。
JZ0TI	jConnect 无法在 _____ 数据库类型和请求的 _____ 类型间进行有意义的转换。 **说明：**例如，应用程序试图对从数据库返回的 time 值调用 ResultSet.getObject(int, Types.DATE) 时，可能会发生这种异常。 **操作：**确保数据库的数据类型能隐式转换成要检索的对象类型。

SQL 状态	消息/说明/操作
JZ0TO	读操作超时。 **说明：**如果套接字读取超时，则会发生此异常。 **操作：**调用 Statement.setQueryTimeout 增大超时期限。同时，检查正在执行的查询或存储过程以确定超时原因。
JZ0TS	试图发送 _____ 时出现截断错误。 **说明：**应用程序指定的字符串的长度大于应用程序要发送的字符串长度。因此，字符串被截断为声明的长度。 **操作：**正确设置长度以避免发生截断。
JZ0US	设置了 SybSocketFactory 连接属性，PROXY 连接属性被设置为服务器小程序的 URL。jConnect 驱动程序不支持这种组合。如果想从在浏览器内运行的小程序中发送安全 HTTP，请使用以 “https://” 开头的代理 URL。 **操作：**参见消息文本。
JZ0XC	_____ 是无法识别的事务协调器类型。 **说明：**元数据信息指示服务器支持分布式事务，但 jConnect 不支持所用的协议。 **操作：**检验是否安装了最新的元数据脚本。如果此错误仍然存在，请联系 Sybase 技术支持。
JZ0XS	服务器不支持 XA 类型的事务。请确保事务功能已在服务器上启用并得到许可。 **说明：**jConnect 尝试连接的服务器不支持分布式事务。 **操作：**不要为此服务器使用 XADataSource，或者对服务器进行升级或配置以使其支持分布式事务。
JZ0XU	当前用户无权执行 XA 类型的事务。确保用户具有 _____ 角色。 **说明：**连接到数据库的用户无权执行分布式事务，最可能的原因是该用户不具备适当的角色 （空白处显示的角色）。 **操作：**按错误消息中的提示授予用户相应的角色，或让具有该角色的另一用户执行此事务。
JZBK1	SybBCP 类未初始化，请重新运行 MDA sql 更新 MDA 存储过程。 **操作：**安装 MDA 存储过程。
JZBK3	批量装载表不存在。 **说明：**以 BCP 指定的表不在数据库中。 **操作：**更正表名。
JZBK4	在 bcp/arrayinsert 模式下的批处理中使用 sql 语句属非法操作。 **说明：**在批处理操作过程中，尝试执行一项非批处理操作。 **操作：**等待批处理操作完成后再尝试执行非批处理操作。
JZBK5	在 bcp 模式下运行批量装载时，应将 autocommit 设置为 true。 **操作：**参见消息文本。
JZBK6	必须使用 ASE 15.7 及更高版本同时启用 “allow wide dol rows” 数据库选项，才能插入偏移量大于 8191 的行。 **操作：**参见消息文本。

SQL 状态	消息/说明/操作
JZBK7	插入数据失败。数据总大小（_____ 字节）超过表 _____ 允许使用的最大行大小（_____ 字节）。 **操作：**参见消息文本。
JZBKI	为属性 ENABLE_BULK_LOAD 设置的值无效。 **操作：**将 ENABLE_BULK_LOAD 设置为以下有效值之一 – ARRAYINSERT_WITH_MIXED_STATEMENTS、ARRAYINSERT、BCP 或 LOG_BCP。
JZNNA	列不允许使用空值。 **说明：**您尝试在预准备语句中使用 setNull() 将位类型列设置为 NULL 值。 **操作：**检查查询并予以更正，将位类型列的值设置为 0 或 1。
S0022	无效的列名称 “_____”。 **说明：**您试图通过名称引用列，但不存在与该名称对应的列。 **操作：**检查列名的拼写是否正确。
ZZ00A	方法 _____ 尚未完成，不应调用它。 **说明：**您试图使用未实现的方法。 **操作：**查阅您的 jConnect 版本附带的发行公告以获取更多信息。也可以访问 jConnect Web page (http://www.sybase.com)，查看是否有最新版本的 jConnect 实现了该方法。如果没有，请不要使用该方法。

词汇表

JDBC	Java 数据库连接 (JDBC) 是允许 Java 程序执行 SQL 语句的 Java API。
jConnect 驱动程序	JDBC 驱动程序，用于 Sybase 服务器，例如使用 Tabular Data Stream (TDS) 通信协议的 Adaptive Server Enterprise。
J2EE 服务器	Sun 公司的 Java 2 平台企业版 (J2EE) 是一种独立于平台的以 Java 为中心的环境，用于在线开发、构建和部署基于 Web 的企业应用程序。
Tabular Data Stream (TDS)	Tabular Data Stream (TDS) 是描述两台计算机之间数据传输的应用程序级协议。TDS 定义可发送的消息类型以及发送消息的顺序。TDS 依赖于面向连接的传输服务。
分布式事务管理支持 (JTA/JTS)	Java Transaction Service (JTS) 指定如何实现一个支持 Java Transaction API (JTA) 并在 API 以下级别实现对象管理组织对象事务服务 1.1 规范的 Java 映射的事务管理器。 Java Transaction API (JTA) 是允许应用程序和 J2EE 服务器访问事务的 API。
SSL	安全套接字层 (SSL) 是管理 Internet 上消息传输安全性的常用协议。
Java 命名和目录接口 (JNDI)	Java 命名和目录接口 (JNDI) 使基于 Java 平台的应用程序能够访问多个命名和目录服务。 JNDI 是来自 Oracle 的 API，用于将 Java 程序连接到命名和目录服务 （如 DNS、 LDAP 和 NDS）。
轻量目录访问协议 (LDAP)	轻量目录访问协议 (LDAP) 是一套软件协议，能让任何人在网络（无论是公共 Internet 还是在公司内部网）中找到组织、个人及其它资源 （如文件和设备）。
对象标识符 (OID)	对象标识符 (OID) 是用于命名对象的标识符。从结构上说， OID 由按层级分配的命名空间中的节点构成。
Java 虚拟机 (JVM)	Java 虚拟机 (JVM) 是提供独立于平台的执行环境的虚拟机，在此环境中可将 Java 字节码转换为计算机语言并将其执行。

ASCII 美国信息交换标准代码 (ASCII) 是以数字形式表示英文字符的代码，每个字母分配一个从 0 到 127 的数字。例如，表示大写 M 的 ASCII 代码为 77。大多数计算机都使用 ASCII 代码表示文本，以实现在计算机之间传输数据。

JDK Java 开发工具包 (JDK) 是用于制作 Java 程序的软件开发工具包 (SDK)。

UCS-2 通用字符集是用于对字符集进行编码的 ISO/IEC 格式。 ISO/IEC 10646 过去与 Unicode同步；但 Unicode 现在添加了一些其它约束，所以符合 10646 并不保证兼容 Unicode。

UTF-16 Unicode 转换格式 16 (UTF-16) 是 Unicode 编码系统中的一种双字节格式。

相对区分名 (RDN) 相对区分名 (RDN) 是区分名的一个组成部分。它包含一个或多个名值对，其中名称和值由等号分隔 （例如 RDN 为 *“uid=ann”* 时，名称为 *“uid”*，值为 *“ann”*），如果存在多个名值对，则应以加号将其分隔（例如 RDN 为 *“cn=Jon Doe+employeeNumber=12345”* 时，名值对为 *“cn=John Doe”* 和 *“employeeNumber=12345”*）。实际上，包含多个名值对的 RDN （称为 “多值 RDN”）很少见，但是在条目中不存在唯一属性或要确保条目的 DN 包含一些有用的标识信息时，这种 RDN 非常有用。

区分名 (DN) 区分名 (DN) 是在目录服务器中唯一标识某个条目的字符串。它包含多个 （或不包含）相对区分名 (RDN) 组成部分，此部分标识目录信息树 (DIT) 中条目的位置。可将条目的区分名视为文件系统中绝对路径的一种类似物，因为它指定名称和层级位置。

Java Cryptography Extension (JCE) Java Cryptography Extension (JCE) 是为实现 Java 安全功能提供统一框架的 API。

RSA 加密 一种高度安全的加密方法。

Java Runtime Environment (JRE) Java Runtime Environment (JRE) 也称作 Java Runtime，是 Java 开发工具包 (JDK) （一组用于开发 Java 应用程序的编程工具）的一部分。

Certicom Security Builder GSE-J 支持 FIPS 140-2 验证的加密算法的 Java Cryptography Extension (JCE) 软件加密提供程序。

LDAP 数据交换格式 (LDIF) LDAP 数据交换格式 (LDIF) 是以文本形式表示目录数据的机制形式。LDIF 规范包含在 RFC 2849 中，该规范不仅描述了一种目录数据表示格式，还描述了一种数据更改机制。

数据类型 一个描述对于变量来说合法的类型、值和操作的定义属性。

GSS 库 实现通用安全服务应用程序编程接口 (GSS-API) 的库。

大对象 (LOB) 数据类型	LOB 数据类型通常为大型字符对象 （文本）或二进制对象 （图像）。
大对象 (LOB) 定位符	LOB 定位符中包含指向 LOB 数据的逻辑指针而不是数据本身，减少了通过网络在 Adaptive Server 与其客户端之间传送的数据量。
SSL	安全套接字层 (SSL) 是为通过 Internet 安全发送信息而开发的安全协议。
Kerberos	Kerberos 是用于鉴定计算机网络中的服务请求的安全方法。
单点登录 (SSO)	单点登录 (SSO) 是一种会话/用户鉴定过程，允许用户输入名称和口令以访问多个应用程序。该过程针对用户拥有权限的所有应用程序对其进行鉴定，当用户在特定会话过程中切换应用程序时消除其它提示。
IETF	Internet Engineering Task Force (IETF) 是 Internet 的主要标准化组织。IETF 是由网络设计人员、操作人员、供应商和研究员组成的大型开放式国际社区，着重于 Internet 的体系结构演变及平稳运行。
Java 通用安全服务 (GSS) 管理器	Java 通用安全服务 (JGSS) 为进行鉴定和安全消息传送提供通用接口。
Wedgetail JCSI	可用于 jConnect 的通用安全服务 (GSS) 管理器。
CyberSafe TrustBroker	可用于 jConnect 的通用安全服务 (GSS) 管理器。
密匙分发中心 (KDC)	密匙分发中心 (KDC) 是执行鉴定和票据生成任务的单点登录 (SSO) 设置的一部分。
TDS 贯通服务器小程序	通过 HTTP 或 HTTPS 数据包穿过 TDS 流的服务器小程序。
RPC	远程过程调用 (RPC) 是一项协议，一个程序可使用它向网络中另一台计算机上的程序请求服务，而无需了解网络详情。（有时，过程调用也称作函数调用或子例程调用。）RPC 使用客户端/服务器模式。发出请求的程序是客户端，提供服务的程序是服务器。与常规或本地过程调用类似，RPC 也是同步操作，需要挂起发出请求的程序直至返回远程过程的结果。但是，使用轻量进程或共享相同地址空间的线程可同时执行多个 RPC。
死锁	在有权锁定一部分数据的两位用户都尝试锁定对方那部分数据时出现的一种情况。
主服务器	在高可用性 (HA) 环境中，主服务器是客户端应首先尝试连接的服务器。
辅助服务器	在高可用性 (HA) 环境中，辅助服务器是客户端在连接主服务器失败的情况下，应尝试连接的服务器。

应用程序编程接口 (API)

应用程序编程接口 (API) 是基于源代码的规范，旨在用作软件组件相互通信的接口。

数据库服务器

数据库服务器这一术语用于表示使用客户端/服务器架构的数据库应用程序的后端系统。

net-protocol

用于在中间层网关之间交换请求/响应，进而与数据库进行通信的协议。

native-protocol

DBMS 支持的本地协议，用于在客户端和服务器之间交换请求/响应。

Adaptive Server Enterprise

Adaptive Server Enterprise (ASE) 是 Sybase, Inc. 推出的在 Linux 及其它基于 Unix 的操作系统、Windows NT 与 Windows 2000 以及 Mac OS 上运行的关系数据库管理系统 (RDBMS)。

SQL Anywhere®

Sybase SQL Anywhere® 是功能全面的关系数据库和数据管理工具。

Sybase® IQ

Sybase® IQ 是专为数据仓储应用设计的高性能决策支持服务器。

Sybase® IQ 是包括 Adaptive Server Enterprise 和 SQL Anywhere® 的 Sybase 产品系列的一部分。 Sybase® IQ 中的组件集成服务提供对主机、UNIX 或 Windows 服务器上关系和非关系数据库的直接访问。

Replication Server™

Replication Server® 维护多个数据库中的复制数据，同时确保这些数据的完整性和一致性。它为使用复制系统中数据库的客户端提供本地数据访问，从而降低网络和集中计算机系统的负载。

DirectConnect®

DirectConnect 是提供与非 Sybase 数据源的基本连接的 ECDA 组件。特别地，它通过 DirectConnect 管理器提供访问管理、事务管理以及远程系统管理。

索引

符号

英文

A

B

C

D

F

G

H

J

K

L

O

P

Q

S

T

W

X

Y

Z