



EsgynDB JDBC Type4 程序员参考指南

2.8.0

版权

© Copyright 2015-2021 贵州易鲸捷信息技术有限公司

公告

本文档包含的信息如有更改，恕不另行通知。

保留所有权利。除非版权法允许，否则在未经易鲸捷预先书面许可的情况下，严禁改编或翻译本手册的内容。易鲸捷对于本文中所包含的技术或编辑错误、遗漏概不负责。

易鲸捷产品和服务附带的正式担保声明中规定的担保是该产品和服务享有的唯一担保。本文中的任何信息均不构成额外的保修条款。

声明

Microsoft® 和 Windows® 是美国微软公司的注册商标。Java® 和 MySQL® 是 Oracle 及其子公司的注册商标。Bosun 是 Stack Exchange 的商标。Apache®、Hadoop®、HBase®、Hive®、openTSDB®、Sqoop® 和 Trafodion® 是 Apache 软件基金会的商标。Esgyn, EsgynDB 和 QianBase 是易鲸捷的商标。

目 录

目 录.....	1
前言.....	9
本文简介.....	9
目标读者.....	9
修订历史.....	9
批评与建议.....	10
相关文档.....	11
符号约定.....	14
1. 简介.....	18
1.1 Type 4 驱动程序 API 程序包	18
1.2 安装.....	18
2. 访问 EsgynDB.....	19
2.1 数据源.....	19
2.1.1 JDBC 数据源（客户端）	19
2.2 安全.....	19
2.3 使用 DataSource 接口连接.....	20
2.3.1 部署 DataSource 对象.....	20
2.3.2 DataSource 对象属性.....	21
2.3.3 以编程方式创建 DataSource 类的实例.....	21
2.3.4 以编程方式注册 DataSource 对象.....	22
2.3.5 通过 JNDI 和连接数据源获取 DataSource 实例	23
2.3.6 指定配置数据源的属性文件	24
2.4 使用 DriverManager 类进行连接.....	25
2.4.1 加载和注册驱动程序	25
2.4.2 建立连接	25
2.4.3 使用 Driver Manager 进行连接.....	26
2.5 连接池.....	27
2.6 语句池.....	28
2.6.1 语句池的使用	28
2.6.2 语句池的故障排除	28

2.7 线程安全数据库访问.....	29
2.8 “Update.....Where Current of”操作	30
2.9 获取查询成本的 INFOSTATS 命令.....	31
2.9.1 使用 INFOSTATS 命令.....	31
2.10 国际化支持.....	33
2.10.1 应用程序使用字符串	33
2.10.2 使用字符集属性控制字符串转换	34
2.10.3 本地化错误信息和状态信息	36
3. Type 4 驱动程序属性	39
3.1 Type 4 驱动程序属性概述.....	39
3.1.1 客户端属性	39
3.1.2 服务器端属性	42
3.2 如何指定 JDBC Type 4 属性.....	44
3.2.1 设置属性	44
3.2.2 创建和使用属性文件	44
3.2.3 在命令行中设置属性	45
3.2.4 属性优先级	46
4. Type 4 驱动程序属性说明	47
4.1 catalog 属性	47
4.2 connectionTimeout 属性.....	48
4.3 clientCharset 属性	49
4.4 fetchBufferSize 属性	49
4.5 initialPoolSize 属性	51
4.6 ISO88591 属性	52
4.7 KANJI 属性	53
4.8 KSC5601 属性.....	54
4.9 language 属性	55
4.10 loginTimeout 属性	56
4.11 maxIdleTime 属性	57
4.12 maxPoolSize 属性	58
4.13 maxStatements 属性	59
4.14 minPoolSize 属性	60
4.15 networkTimeout 属性	61

4.16 password 属性	62
4.17 properties 属性.....	63
4.18 reserveDataLocators 属性	64
4.19 roundingMode 属性.....	66
4.20 schema 属性.....	68
4.21 T4LogFile 属性.....	69
4.22 T4LogLevel 属性.....	71
4.22.1 T4LogLevel 注意事项.....	72
4.23 translationVerification 属性.....	73
4.24 url 属性	74
4.24.1 url 属性注意事项	74
4.25 user 属性.....	75
5. Type4 驱动程序高级属性说明	76
5.1 客户端缓存.....	76
5.2 clipVarchar 属性	77
5.3 sessionDebug 属性	78
5.4 specifiedServer 属性.....	78
5.5 traceTransTime 属性.....	78
6. Type 4 驱动程序兼容性	79
6.1 兼容性概述.....	79
6.2 不支持的功能.....	80
6.3 偏差.....	84
6.4 EsgynDB 扩展	87
6.4.1 消息国际化	87
6.4.2 其他 DatabaseMetaData API.....	87
6.5 DatabaseMetaData 方法处理 Null 参数的一致性	88
6.6 Type 4 驱动程序数据类型与 SQL 数据类型的对应关系	90
6.6.1 JDBC 数据类型.....	90
6.7 浮点支持.....	93
6.8 SQLJ 支持	93
6.9 Type 4 驱动程序不支持的 JDBC 3.0 功能	93
6.10 限制.....	93
7. 跟踪和日志功能	95

7.1 标准 JDBC 跟踪和日志功能.....	95
7.2 Type 4 驱动程序日志功能.....	97
7.2.1 控制 Type 4 驱动程序日志输出.....	97
7.2.2 消息格式	99
7.2.3 日志输出示例	101
8. 消息.....	102
8.1 消息格式.....	102
8.2 获得帮助.....	102
8.3 Type 4 驱动程序错误消息.....	102
8.3.1 01032 08S01	102
8.3.2 01056 25000	102
8.3.3 01118 S1008.....	103
8.3.4 08001 HY000.....	103
8.3.5 08004 HY000.....	103
8.3.6 29001 HYC00	103
8.3.7 29002 08003	104
8.3.8 29003 HY000.....	104
8.3.9 29004 HY024.....	104
8.3.10 29005 HY024.....	104
8.3.11 29006 HY000.....	105
8.3.12 29007 07009	105
8.3.13 29008 24000	105
8.3.14 29009 HY109.....	105
8.3.15 29010 07009	106
8.3.16 29011 07009.....	106
8.3.17 29012 07006	106
8.3.18 29013 HY024.....	106
8.3.19 29015 HY024.....	107
8.3.20 29017 HY004.....	107
8.3.21 29018 22018	107
8.3.22 29019 07002	107
8.3.23 29020 07009	108
8.3.24 29021 HY004.....	108

8.3.25 29022 HY010.....	108
8.3.26 29026 HY000.....	108
8.3.27 29027 HY011.....	109
8.3.28 29029 HY011.....	109
8.3.29 29031 HY000.....	109
8.3.30 29032 23000	109
8.3.31 29033 23000	110
8.3.32 29035 HY000.....	110
8.3.33 29036 HY000.....	110
8.3.34 29037 HY106.....	110
8.3.35 29038 HY107.....	111
8.3.36 29039 HY092.....	111
8.3.37 29040 HY000.....	111
8.3.38 29041 HY000.....	112
8.3.39 29042 HY000.....	112
8.3.40 29043 HY000.....	112
8.3.41 29044 HY000.....	112
8.3.42 29045 01S07	112
8.3.43 29046 22003	113
8.3.44 29047 HY000.....	113
8.3.45 29048 HY009.....	113
8.3.46 29049 25000	113
8.3.47 29050 HY107.....	113
8.3.48 29051 01S02	114
8.3.49 29053 HY000.....	114
8.3.50 29054 HY000.....	114
8.3.51 29056 HY000.....	115
8.3.52 29057 HY000.....	115
8.3.53 29058 HY000.....	115
8.3.54 29059 HY000.....	116
8.3.55 29060 HY000.....	116
8.3.56 29061 HY00.....	116
8.3.57 29063 HY00.....	116

8.3.58 29067 07009	117
8.3.59 29068 07009	117
8.3.60 29069 HY000.....	117
8.3.61 29100 HY000.....	118
8.3.62 29101 HY000.....	118
8.3.63 29102 HY000.....	118
8.3.64 29103 HY000.....	118
8.3.65 29104 HY000.....	119
8.3.66 29105 HY000.....	119
8.3.67 29106 HY000.....	119
8.3.68 29107 HY000.....	119
8.3.69 29108 HY000.....	119
8.3.70 29109 HY000.....	120
8.3.71 29110 HY000.....	120
8.3.72 29111 HY000.....	120
8.3.73 29112 HY000.....	121
8.3.74 29113 HY000.....	121
8.3.75 29114 HY000.....	121
8.3.76 29115 HY000.....	121
8.3.77 29116 HY000.....	122
8.3.78 29117 HY000.....	122
8.3.79 29118 HY000.....	122
8.3.80 29119 HY000.....	122
8.3.81 29120 HY000.....	123
8.3.82 29121 HY000.....	123
8.3.83 29122 HY000.....	123
8.3.84 29123 HY000.....	123
8.3.85 29124 HY000.....	124
8.3.86 29125 HY000.....	124
8.3.87 29126 HY000.....	124
8.3.88 29127 HY000.....	125
8.3.89 29128 HY000.....	125
8.3.90 29129 HY000.....	125

8.3.91 29130 HY000.....	126
8.3.92 29131 HY000.....	126
8.3.93 29132 HY000.....	126
8.3.94 29133 HY000.....	127
8.3.95 29134 HY000.....	127
8.3.96 29135 HY000.....	127
8.3.97 29136 HY000.....	128
8.3.98 29137 HY000.....	128
8.3.99 29138 HY000.....	128
8.3.100 29139 HY000.....	128
8.3.101 29140 HY000.....	129
8.3.102 29141 HY000.....	129
8.3.103 29142 HY000.....	129
8.3.104 29143 HY000.....	130
8.3.105 29144 HY000.....	130
8.3.106 29145 HY000.....	130
8.3.107 29146 HY000.....	130
8.3.108 29147 HY000.....	131
8.3.109 29148 HY000.....	131
8.3.110 29149 HY000.....	131
8.3.111 29150 HY000.....	131
8.3.112 29151 HY000.....	132
8.3.113 29152 HY000.....	132
8.3.114 29153 HY000.....	132
8.3.115 29154 HY000.....	132
8.3.116 29155 HY000.....	133
8.3.117 29156 HY000.....	133
8.3.118 29157 HY000.....	133
8.3.119 29158 HY000.....	133
8.3.120 29159 HY000.....	134
8.3.121 29160 HY000.....	134
8.3.122 29161 S1000	134
8.3.123 29162 S1000	135

8.3.124 29163 08001	135
8.3.125 29164 08001	135
8.3.126 29165 HY000.....	136
8.3.127 29166 HY000.....	136
8.3.128 29167 HY000.....	136
8.3.129 29168 HY000.....	136
8.3.130 29169 HY000.....	137
8.3.131 29170 HY000.....	137
8.3.132 29172 HY000.....	137
8.3.133 29173 HY000.....	137
8.3.134 29174 HY000.....	138
8.3.135 29175 HY000.....	138
8.3.136 29177 HY000.....	138
8.3.137 29178 HY000.....	138
8.3.138 29182 HY000.....	139
8.3.139 S1000 HY000	139
9. 避免驱动程序-服务器版本不匹配	140
9.1 JDBC 客户端连接至 EsgynDB 的注意事项	140
9.2 版本不匹配错误消息.....	140

前言

本文简介

本指南介绍了如何使用 EsgynDB JDBC Type 4 驱动程序（以下简称为 Type 4 驱动程序），该驱动程序使运行在外部平台上的 Java 程序通过 JDBC 访问 EsgynDB。

目标读者

本指南的目标读者为访问 EsgynDB SQL 数据库的 Java 程序员。

本指南默认您已熟悉 Java 文档。更多信息，请参阅 <http://docs.oracle.com/en/java/>。

修订历史

版本	日期	说明
2.8.0	2021 年 3 月	添加第五章高级属性说明。 新增客户端属性 clientCharset。 修改 9.1，新增一条注意事项。
2.7.0	2019 年 11 月	
2.4.0	2018 年 11 月	
2.3.0	2017 年 11 月	
2.1.0	2017 年 5 月	
2.0.1	2016 年 7 月	
2.0.0	2016 年 6 月	
1.3.0	2016 年 1 月	

批评与建议

我们支持您对本指南做出的任何批评与建议，并尽力提供符合您需求的文档。

若您发现任何错误、或有任何改进建议，请发邮件至 support@esgyn.cn。

相关文档

文档名称	说明
《EsgynDB ADO.NET Provider 手册》	本文介绍如何使用 EsgynDB ADO.NET Provider 访问 EsgynDB。
《EsgynDB 备份与恢复指南》	本文介绍 EsgynDB 时间点恢复功能。
《EsgynDB CQD 参考手册》	本文介绍控制查询默认 (Control Query Default) 设置, 它能更改当前编译器会话环境, 影响查询执行计划。
《EsgynDB 安装指南》	本文介绍安装 EsgynDB, 包括安装前准备、安装 Hadoop 发行版、故障排除、配置、启用安全功能、提高安全性和卸载 EsgynDB 等。
《EsgynDB Manager 用户指南》	本文介绍网页管理工具 EsgynDB Manager, 包括仪表盘、工作负载、日志、数据库、连接服务、查询工作台、安全、租户、警报和工具等。
《EsgynDB Windows ODBC 驱动安装指南》	本文介绍如何在 Window 系统安装 EsgynDB ODBC 驱动。
《EsgynDB 客户端安装指南》	介绍如何安装和配置各类客户端应用程序, 例如, Trafodion 命令行界面 (TrafCI)、Trafodion odb、DbVisualizer、SQuirreL SQL

	客户端和 JDBC/ODBC 驱动程序，方便用户连接并使用 EsgynDB。
《EsgynDB SQL 手参考册》	本文介绍 EsgynDB SQL 语句、实用程序、函数和其它 SQL 语言元素的语法。
《Trafodion 命令行界面指南》	本文介绍 Trafodion 命令行界面 (TrafCI)，该界面允许用户连接至 EsgynDB，交互式地或通过脚本访问 EsgynDB。
《EsgynDB 连接服务参考指南》	本文介绍如何对 EsgynDB 连接服务 (Database Connectivity Service, DCS) 进行配置、调优、排故和调试，这为基于 ODBC/JDBC API 开发的应用程序提供了访问 EsgynDB SQL 数据库服务器的框架；另外，本文还介绍了 DCS API。
《Trafodion odb 用户指南》	本文介绍如何配置并使用 Trafodion odb 工具，该工具是一个多线程的 ODBC 命令行工具，能对数据执行 Trickle Load。
《Trafodion REST 服务器参考指南》	本文介绍如何启动、关闭、配置、排故和调试 Trafodion REST (Representational State Transfer) 服务器；另外，本文简要介绍了 REST API。
《EsgynDB 多数据中心复制	本文介绍如何使用 EsgynDB 同步复制功能和

指南》	xdc 工具等。
《EsgynDB 迁移工具用户指南》	本文介绍如何安装和使用 EsgynDB 迁移工具。
《EsgynDB 傻瓜手册系列——如何使用 EsgynDB? 》	本文介绍如何快速了解并使用 EsgynDB。
《EsgynDB 傻瓜手册系列——如何集成 EsgynDB 与 Zeppelin? 》	本文介绍如何安装、配置和使用 Zeppelin。

符号约定

描述语法时，本手册采用以下符号约定。

- 大写字母

表示关键字和保留字。未在方括号中的内容是必选的语法项。



示例

```
SELECT
```

- 小写字母

表示变量。未在方括号中的内容是必选的语法项。



示例

```
file-name
```

- [] 方括号

表示可选的语法项。



示例

```
DATETIME [start-field TO] end-field
```

如果方括号中包括多个语法项，您可以选择某一语法项或不选。

多个语法项可以垂直排列，每个语法选项用方括号括起；也可以水平排列，用竖线隔开。



示例

```
DROP SCHEMA schema [ CASCADE ][ RESTRICT ]
```

```
DROP SCHEMA schema [ CASCADE | RESTRICT ]
```


- {} 大括号

表示必选的语法项。

 示例

```
FROM { grantee [, grantee ] ... }
```

如果大括号中包括多个语法选项，您必须选择某一语法项。

多个语法项可以垂直排列，每个语法项用大括号括起；也可以水平排列，用竖线隔开。

 示例

```
INTERVAL { start-field TO end-field } { single-field }  
INTERVAL { start-field TO end-field | single-field }
```

- | 竖线

隔开方括号或大括号中的多个语法项。

 示例

```
{expression | NULL}
```

- ... 省略号

○ 紧跟在方括号或大括号之后，表示括号内的语法项可以重复任意次。

 示例

```
ATTRIBUTE[S] attribute [, attribute] ...  
{, sql-expression } ...
```

- 紧跟在单个语法项之后，表示该语法项可以重复任意次。



示例

```
expression-n ...
```

- 标点符号

- 圆括号、逗号、分号和其它符号，请参照以下示例输入。



示例

```
DAY (datetime-expression)
```

```
@script-file
```

- 符号（例如，方括号或大括号）周围的引号表示该符号是必选项。



示例

```
"{" module-name [, module-name] ... }"
```

- 语法项的间距

- 语法项之间必须有空格。如果某一语法项是圆括号或逗号，则不需要空格。



示例

```
DAY (datetime-expression) DAY(datetime-expression)
```

- 如果语法项之间没有空格，则不允许空格。在以下示例中，句点与任何其他语法项之间不允许有空格。



示例

```
myfile.sh
```

- 行距

如果一条命令过长（超过一行），则每个连续行需缩进三个空格，并通过空行与前一行分隔。

行距用于区分连续行中的语法项与垂直列表（多个语法项）中的项。



示例

```
match-value [NOT] LIKE _pattern
    [ESCAPE esc-char-expression]
```

1. 简介

本指南介绍如何使用 EsgynDB JDBC Type 4 驱动程序，该驱动程序使运行在其他平台上的 Java 应用程序访问 EsgynDB。

支持的 Java 版本：JDK 1.7 或更高版本的 Java 平台。

1.1 Type 4 驱动程序 API 程序包

EsgynDB JDBC Type 4 驱动程序（以下简称“Type 4 驱动程序”）实现了符合标准 JDBC 3.0 数据访问 API 的 JDBC 技术。

Type 4 驱动程序包的名称为 `org.trafodion.jdbc.t4`。

更多关于标准 JDBC API 的信息，请参阅 <http://docs.oracle.com/en/java/>。

1.2 安装

更多关于如何安装 EsgynDB JDBC Type 4 驱动程序的信息，请参阅《EsgynDB 客户端安装指南》。

2. 访问 EsgynDB

2.1 数据源

数据源是数据库或其它数据存储实体。JDBC (客户端) 数据源是一个 Java 对象，它包括数据库的 URL、连接数据库的 catalog 和 schema、以及获取 JDBC 连接至底层数据库的方法。

2.1.1 JDBC 数据源 (客户端)

所有 JDBC 数据源类实现 `javax.sql.DataSource` 接口或 `javax.sql.ConnectionPoolDataSource` 接口。

Type 4 驱动程序数据源类 `org.trafodion.jdbc.t4.TrafT4DataSource` 和 `org.trafodion.jdbc.t4.TrafT4ConnectionPoolDataSource`，这些类由 JDBC 3.0 规范定义。

一般而言，用户或系统管理员使用工具创建数据源，通过 JNDI 服务供应部件注册数据源。运行时，用户应用程序通过 JNDI 获取数据源，并使用数据源方法建立与底层数据库的连接。

`DataSource` 对象映射至数据库实例。在 Type 4 驱动程序产品中，`DataSource` 对象是应用程序代码和数据库之间的接口，并连接 DCS¹ 数据源。

2.2 安全

客户端使用有效的用户名和密码和标准 JDBC 3.0 API 连接至 EsgynDB。应用程序能使用不同的用户 ID 和不同的 `Connection` 对象创建多个连接。

Type 4 驱动程序提供用户名和密码认证。密码已加密。

¹ DCS，即 Data Connectivity Services，数据连接服务。

2.3 使用 DataSource 接口连接

`javax.sql.DataSource` 接口提高了应用程序可移植性，它是与数据库建立连接的首选方法。

可移植性通过允许应用程序为数据源使用逻辑名称（而非提供驱动程序特定信息）实现。

Java 命名和目录接口（Java Naming and Directory Interface, JNDI）的命名服务能将逻辑名称映射到 `javax.sql.DataSource` 对象中。

推荐您对应用程序服务器使用该 `DataSource` 方法。

应用程序通过使用 `DataSource` 中的 `getConnection` 方法请求连接时，该方法将返回一个 `Connection` 对象。

`DataSource` 对象是生产 `Connection` 对象的工厂。实现 `DataSource` 接口的对象需要在 JNDI 服务中注册。

2.3.1 部署 DataSource 对象

在应用程序连接至 `DataSource` 对象之前，系统管理员会部署 `DataSource` 对象以供程序员使用。

系统管理员使用 GUI 工具设置数据源属性，这是安装数据源的步骤之一。用户无需获取或设置属性（如需设置，请使用管理工具的内置的功能）。

以下步骤为创建和注册数据库对象：

1. 创建 `DataSource` 类的实例。
2. 设置 `DataSource` 对象的属性。
3. 通过使用 JNDI API 的命名服务注册 `DataSource` 对象。

应用程序开发人员或系统管理员使用 GUI 工具设置 `DataSource` 类实例和 `DataSource` 对象属性，这是安装数据源的步骤之一。如果您正在使用已安装好的数据源，请参阅[以编程方式创建 DataSource 类的实例](#)。

更多关于使用数据源的信息，请参阅 [JDBC™ Database Access 目录文档](#) 中的 [连接 DataSource 对象](#)，或该文档中的其他信息。

2.3.2 DataSource 对象属性

DataSource 对象具有识别和描述对象表示的实际数据源的属性，例如，URL（数据库的主 IP 地址或主机名）、数据库 schema 和 catalog 名称、数据库服务器位置和数据库名称等。

更多关于 Type 4 驱动程序属性（和 DataSource 对象一同使用）的信息，请参阅 [Type 4 驱动程序属性](#)。

2.3.3 以编程方式创建 DataSource 类的实例

JDBC 应用程序能以编程方式设置 DataSource 属性，并通过 DataSource 对象注册。如果您想要以编程方式获取或设置 DataSource 对象属性，请在 TrafT4DataSource 对象或 TrafT4ConnectionPoolDataSource 对象上使用合适的 getter 或 setter 方法。

示例

```
TrafT4DataSource temp = new TrafT4DataSource() ;  
temp.setCatalog( "TRAFODION" ) ;
```

以下示例说明，如果对象支持 serverDataSource 属性

```
ds.setServerDataSource ("my_server_datasource") ,
```

DataSource 对象 ds 需要包含的方法，以及为了使用 Type 4 驱动程序访问 EsgyDB，需要设置 TrafT4DataSource 对象属性：

示例

```
TrafT4DataSource ds = new TrafT4DataSource() ;  
ds.setUrl( "jdbc:t4jdbc://<primary IP addr or host  
name>:23400/" );
```

```
ds.setSchema( "myschema" ) ;
ds.setUser( "gunnar" ) ;
ds.setPassword( "my_userpassword" ) ;
// Properties relevant for Type 4 connection pooling.
// Set ds.setMaxPoolSize(-1) for turning OFF connection
pooling
ds.setMaxPoolSize( "100" ) ;
ds.setMinPoolSize( "10" ) ;
// Properties relevant for Type 4 statement pooling.
// Set ds.setMaxStatement(0) for turning statement
pooling OFF
// Statement pooling is enabled only when connection
pooling is
// enabled.
ds.setMaxStatements( "7000" ) ;
```

实际上，该方法创建了一个属性文件。更多信息，请参阅[创建和使用属性文件](#)。

2.3.4 以编程方式注册 DataSource 对象

以下示例解释了如何以编程方式注册并通过 JNDI 使用上述代码创建 TrafT4DataSource 对象 ds。

示例

```
java.util.Hashtable env = new java.util.Hashtable() ;
env.put( Context.INITIAL_CONTEXT_FACTORY, "Factory class
name here" ) ;

javax.naming.Context ctx = new
javax.naming.InitialContext( env ) ;
ctx.rebind( "myDataSource", ds ) ;
```

2.3.5 通过 JNDI 和连接数据源获取 DataSource 实例

JDBC 应用程序从上下文对象中查找数据源 JNDI 名称。一旦应用程序获得 DataSource 对象,它将对数据源执行 getConnection() 调用,并获取连接。

以下步骤为 JDBC 应用程序连接并使用与数据库相关的数据源,以及执行操作的应用程序代码。

1. 导入程序包。

示例

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

2. 创建初始上下文环境。

示例

```
Hashtable env = new Hashtable() ;
env.put( Context.INITIAL_CONTEXT_FACTORY,
"com.sun.jndi.fscontext.RefFSContextFactory" ) ;
try
{
    Context ctx = new InitialContext( env ) ;
}
catch( ... )
{
    ...
}
```

2. 访问 EsgynDB

3. 查找与数据源 myDataSource 相关的 JNDI 名称，其中 myDataSource 是与真实数据源服务器相关的逻辑名称。

示例

```
DataSource ds  
  
=(DataSource)ctx.lookup( "myDataSource" ) ;
```

4. 使用数据源创建连接。

示例

```
con = ds.getConnection() ;
```

5. 使用该连接执行操作。以下为简单示例。

示例

```
stmt = con.createStatement() ;  
  
try  
{  
    stmt.executeUpdate( "drop table tdata" ) ;  
}  
  
catch ( SQLException e ) {}
```

2.3.6 指定配置数据源的属性文件

使用属性文件方法配置 DataSource 对象前，属性文件则必须保存在磁盘上且需包含配置数据源的 property_name = property_value 键值对。更多信息，请参阅[创建和使用属性文件](#)。

JDBC 应用程序进行连接时，应用程序将属性文件作为命令行参数进行传递：

```
java -Dt4jdbc.properties=<path of properties file on disk
```

2.4 使用 DriverManager 类进行连接

`java.sql.DriverManager` 类广泛地应用于获取连接,但与 `DataSource` 类相比,它的灵活性稍差。`DriverManager` 类与驱动程序接口共同管理已加载的驱动程序集。当应用程序使用 `DriverManager.getConnection` 方法发出连接请求并提供 URL 时,`DriverManager` 会找到合适的驱动程序来识别该 URL,并使用该驱动程序获取数据库连接。

`org.trafodion.jdbc.t4.T4Driver` 是实现 `java.sql.Driver` 接口的 Type 4 驱动程序类。

2.4.1 加载和注册驱动程序

连接数据库之前,应用程序使用 `DriverManager` 类加载和注册 Type 4 驱动程序,可以使用以下任一方法:

- 在 Java 程序命令行的 `-Djdbc.drivers` 选项中指定 Type 4 驱动程序类:

```
-Djdbc.drivers=org.trafodion.jdbc.t4.T4Driver
```

- 在应用程序中以编程方式使用 `Class.forName` 方法:

```
Class.forName("org.trafodion.jdbc.t4.T4Driver")
```

- 将 Type 4 驱动程序类添加到 `java.lang.System` 属性的 `jdbc.drivers` 属性项中: `jdbc.drivers=org.trafodion.jdbc.t4.T4Driver`

2.4.2 建立连接

`DriverManager.getConnection` 方法接收一个 Type 4 驱动程序 URL 字符串。

Type 4 驱动程序的 JDBC URL 为 `jdbc:t4jdbc://<ip addr or host name>:23400/[:][property=value[;property2=value2]...]`

参数	使用
<code><ip addr or host</code>	EsgynDB 的主 IP 地址或主机名。

2. 访问 EsgynDB

name>	
23400	EsgynDB 的端口号。
property = value 和 property2 = value2	指定 Type 4 驱动程序的键值对属性，必须用分号 (;) 隔开。例如， T4LogLevel=ALL;T4LogFile=temp1.log.

更多信息，请参阅 [Type 4 驱动程序属性](#)。

JDBC 应用程序可以使用以下代码建立连接：

```
Class.forName( "org.trafodion.jdbc.t4.T4Driver" ) ;  
//loads the driver  
String url = "jdbc:t4jdbc://<database primary IP  
address>:23400/"  
Connection con = DriverManager.getConnection( url,  
"userID", "Passwd" ) ;
```

变量 con 表示与数据源的连接，该数据源可用于创建和执行 SQL 语句。

2.4.3 使用 Driver Manager 进行连接

- Type 4 驱动程序定义了一组用于配置驱动程序的属性。更多信息，请参阅 [Type 4 驱动程序属性](#)。
- Java 应用程序能通过以下方式（按优先级排序）指定属性：
 1. 在 DriverManager 类的 getConnection 方法中使用
java.util.Properties 参数。
 2. 在 DriverManager.getConnection 方法中使用数据库 URL，URL
为 jdbc:t4jdbc://<ip addr or host
name>:23400/:property=value
<ip addr or host name> 是 EsgynDB 的主 IP 地址或主机名。

3. 使用 JDBC 驱动程序的属性文件，该文件作为命令行参数传递。在命令
行中输入属性文件的格式为：`-Dt4jdbc.properties=<path of
properties file on disk>`

例如，`-Dt4jdbc.properties=C:\temp\t4props`

更多信息，请参阅[创建和使用属性文件](#)。

4. 在命令行中使用包含-D 选项的 JDBC 属性。如果使用-D 选项，那么在
Java 应用程序中，它适用于使用 DriverManager 的所有 JDBC 连接。

命令行中的格式为：

`-Dt4jdbc.property_name=<property value>`

例如，`-Dt4jdbc.maxStatements=1024`

2.5 连接池

Type 4 驱动程序提供连接池功能，它将数据库连接的缓存分配给客户端会话，并
在数据库活动中再次使用。如果连接池处于激活状态，则连接不会真正断开。调
用 `Connection.close()` 方法时，连接将返回到其连接池。客户端再次请求连
接时，驱动程序将返回已存在连接池中的连接，而不返回新的物理连接。

- 当 JDBC 应用程序使用 DriverManager 类或 DataSource 接口获取 JDBC
连接时，可以使用连接池。maxPoolSize 和 minPoolSize 的属性值确定
连接池的大小。
- 默认情况下，连接池未开启。若要启用连接池功能，请将 maxPoolSize 属性
设置为大于 0 的整数值。
- 使用以下 Type 4 驱动程序属性管理连接池：
 - [maxpoolsize](#) 属性中的 maxPoolSize
 - [minPoolSize](#) 属性中的 minPoolSize
 - [initialPoolSize](#) 属性中的 initialPoolSize
 - [maxStatements](#) 属性中的 maxStatements
- 与 DriverManager 类同时使用时，Type 4 驱动程序的连接池管理器通过特

定值确定将哪些连接放在同一连接池内，特定值为以下属性的组合：

- url
- catalog
- schema
- username
- password

具有相同特定值的连接会放在同一连接池内。

注意：给定属性组合首次连接使用的连接池属性值在整个进程中都有效，在给定组合进行首次连接后，应用程序将不能更改其中任何属性值。

2.6 语句池

语句池功能允许应用程序重复使用 `PreparedStatement` 对象（方式与在连接池中使用连接相同），语句池对应用程序完全透明。

2.6.1 语句池的使用

- 如果想启用语句池，请将 `maxStatements` 属性设置为大于 0 的整数值，并启用连接池。更多信息，请参阅 [initialPoolSize](#) 属性和 [连接池](#)。
- 为 JDBC 应用程序启用语句池可能会极大提高性能。
- 由于 `PreparedStatement` 对象不在范围内且不能被重复使用（除非应用程序显式地关闭），请使用 `Statement.close` 方法显式地关闭。
- 为确保应用程序重复使用 `PreparedStatement`，请调用以下任一方法：
 - `Statement.close` 方法：由应用程序调用。
 - `Connection.close` 方法：由应用程序调用。重复使用连接时，正在使用的所有 `PreparedStatement` 对象都可以重复使用。

2.6.2 语句池的故障排除

如果您想排除语句池的故障，请注意以下有关 Type 4 驱动程序的详细信息：

- Type 4 驱动程序在语句池中查找匹配的 `PreparedStatement` 对象，并重

复使用 PreparedStatement。

匹配条件包括 SQL 字符串、catalog、当前 schema、当前事务隔离和结果集的可保留性。

如果 Type 4 驱动程序查找到匹配的 PreparedStatement 对象，为了方便重复使用，驱动程序将返回相同的 PreparedStatement 对象到应用程序，并将 PreparedStatement 对象标记为“正在使用”。

- 当语句的数量达到 maxStatements 的上限时，先进先出算法（First in First out）为随后生成的 PreparedStatement 对象留出缓存空间。
- Type 4 驱动程序认为在执行或重复使用时生效的任何 SQL CONTROL 语句与 SQL 编译时生效的语句相同。

如果该条件不为真（not true），则重复使用 PreparedStatement 对象可能会导致错误。

- 避免通过重新编译的方式来提高语句池的性能。满足以下某些条件时，SQL 执行器会自动重新编译查询：
 - 同一表的运行时版本与编译时版本有不同的重新定义时间戳。
 - DDL 或 SQL 实用程序操作能减少针对表的操作。
 - 执行时的事务隔离级别和访问模式与编译时的不同。
- 重新编译查询时，SQL 执行器将存储重新编译的查询。因此，查询只重新编译一次，直到再次满足上述任一条件。
- Type 4 驱动程序不缓存 Statement 对象。

2.7 线程安全数据库访问

为了确保线程安全，在 Type 4 驱动程序中，将 API 层类作为特定的实例对象：

- 为了确保连接时的线程安全，将 `TrafT4DataSource.getConnection()` 作为同步方法。
- 一旦建立连接，`Connection` 对象即为一个特定实例。
- 在单个连接中，如果多个语句在不同线程上运行，为了防止数据损坏，语句

对象将序列化。

2.8 “Update.....Where Current of” 操作

执行游标 SQL 语句的 `update . . . where current of` 操作时，`ResultSet` 的访存大小（`fetch size`）必须为 1。

如果访存大小大于 1，则 `update . . . where current` 操作结果可能是：

- 可能根据实际游标位置更新不正确的行。
- 正在更新的游标可能已关闭，因此可能会引起 `SQL Exception`。

以下示例为将结果集的访存大小设置为 1，执行 `update . . . where current of cursor` 语句。

示例

```
ResultSet rs ;
...
rs.setFetchSize( 1 ) ;
String st1 = rs.getCursorName() ;

Statement stmt2 =
    connection.createStatement( ResultSet.TYPE_
                                FORWARD_ONLY
                                , ResultSet.CONCUR_UPDATABLE
                                ) ;

stmt2.executeUpdate( "UPDATE cat2.sch2.table1
                     SET j = 'update row' WHERE CURRENT
                     OF "
                     + st1
                     ) ;
```

2.9 获取查询成本的 INFOSTATS 命令

INFOSTATS 命令报告特定查询的累计成本。INFOSTATS 为预编译语句收集统计信息。准备完成后，统计信息将作为结果集返回到 JDBC 应用程序。结果集包含以下列：

列	说明
Query ID (SQL_CHAR)	该查询项的唯一标识符。
CPUTIME (SQL_DOUBLE)	执行查询语句指令的处理器器的预估时间（单位为秒）。1.0 表示 1 秒。
IOTime (SQL_DOUBLE)	执行查询语句 I/O(检测附加数据传输)的预估时间。
MsgTime (SQL_DOUBLE)	查询语句发送消息的预估时间(单位为秒)。预估时间与本地消息、远程消息和数据发送量相关。
IdleTime (SQL_DOUBLE)	等待查询语句的事件发生的最长预估时间（单位为秒）。预估时间包括打开表格或启动 ESP 进程的时间。
TotalTime (SQL_DOUBLE)	预估与执行查询相关的成本。
Cardinality (SQL_DOUBLE)	预估将返回的行数。

2.9.1 使用 INFOSTATS 命令

INFOSTATS 命令仅能与 PreparedStatement 对象一起使用。语法为：

```
INFOSTATS cursor_name。
```

其中 cursor_name 是预编译语句的名称。如果 cursor_name 需区分大小写，则用单引号括起。

为了获取 cursor_name，在 EsgynDB JDBC Type 4 驱动程序的

TrafT4PreparedStatement 类中定义 `getStatementLabel()` 方法:

```
org.trafodion.jdbc.t4.TrafT4PreparedStatement: public
String
getStatementLabel() ;
```

2.9.1.1 INFOSTATS 注意事项

- 仅能在以下方法中使用 INFOSTATS:

```
java.sql.Statement.executeQuery(String sql)
java.sql.Statement.execute(String sql)
```

- INFOSTATS 不支持 `setCursorName`。
- 如果错误地调用 INFOSTATS, Type 4 驱动程序会产生以下错误:

```
Message: INFOSTATS command can only be executed
        by calling execute(String sql) method.
        Sqlstate HY000
        Sqlcode 29180
```

2.9.1.2 INFOSTATS 示例

示例

```
Statement s = conn.createStatement( ) ;
TrafT4PreparedStatement p =
(TrafT4PreparedStatement)conn.prepareStatement (
"SELECT * FROM t WHERE i = ?" ) ;
boolean results = s.execute( "INFOSTATS " +
p.getStatementLabel() ) ;

if ( results )
{
    ResultSet rs = s.getResultSet( ) ;
    while ( rs.next( ) )
```

```
{  
    //process data  
}  
}
```

输出

```
QueryID:  
MXID001001128212016369912348191_16_SQL_CUR_9829657  
CPUTime: 0.09975778464794362  
IOTime: 0.10584000146627659  
MsgTime: 0.09800000134418951  
IdleTime: 0.09800000134418951  
TotalTime: 0.10584000146627659  
Cardinality: 100.0
```

2.10 国际化支持

2.10.1 应用程序使用字符串

驱动程序中的国际化支持会影响字符串的处理。Type 4 驱动程序处理以下两种情况的字符串。

- 当驱动程序处理 SQL 语句时。

示例

```
Statement stmt = connection.createStatement() ;  
stmt.execute( "SELECT * FROM table1 WHERE col1 =  
'abcd'" ) ;
```

- 当驱动程序处理 JDBC 参数时。

示例

2. 访问 EsgynDB

```
PreparedStatement pstmt = connection.prepareStatement(
    "SELECT * FROM table1 WHERE col1 = ?" );
pstmt.setString( 1, "abcd" );
```

Type 4 驱动程序使用数据库中的列类型将字符串从 Java 转化为 EsgynDB 处理的字节数组。

2.10.2 使用字符集属性控制字符串转换

Type 4 驱动程序提供字符集映射属性。这些属性允许显式地定义内部 SQL 字符集格式与 Java 字符串 Unicode (UnicodeBigUnmarked) 编码间的转换。

Type 4 驱动程序键值提供字符集映射属性，如下表所示：

关键字	默认值
ISO88591	ISO88591_1
KANJI	SJIS
KSC5601	EUC_KR

下表描述了这些字符集，并总结了 EsgynDB 支持的字符集。

EsgynDB 字符集	对应 Java 编码集 ²	说明
ISO88591	ISO88591_1	单个字符，8 位字符数据类型，ISO88591 支持英语和其他西欧语言。

更多信息，请参阅 [ISO88591 属性](#)。

2.10.2.1 使用字符集属性

java.sql.PreparedStatement 类包含 setString() 和 setCharacterStream() 方法，这些方法分别接收 String 和 Reader 参数。java.sql.ResultSet 类包含 getString() 和 getCharacterStream()

² java.io 和 java.lang API 的标准名称。

方法，这些方法分别返回 `String` 和 `Reader` 参数。

- **检索一列**

如果以字符串形式检索一列（例如，调用 `getString()` 或 `getCharacterStream` 方法），Type 4 驱动程序使用字符集映射属性键实例化字符串对象（其中键对应列的字符集）。

示例

SQL `CREATE TABLE` 语句创建包含 ISO88591 列的表。

```
CREATE TABLE t1 ( c1 CHAR(20) CHARACTER SET
ISO88591 ) ;
```

JDBC 程序使用以下 `java` 命令设置 ISO88591 属性，并发布 `getString()` 方法。

示例

```
java -Dt4jdbc.ISO88591=SJIS test1.java
// The following method invocation returns a String
object, which
// was created using the "SJIS" Java canonical name as
the charset
// parameter to the String constructor.
String s1 = rs.getString( 1 ) ; // get column 1 as a
String
```

- **设置参数**

使用字符串设置参数时（例如，调用 `setString()` 方法），Type 4 驱动程序在生成字符串内部表示时使用键值（键对应于列的字符集）。字符串 `getBytes` 方法的字符集参数是 Java 标准名称，对应该列的字符集。

示例

以下 SQL CREATE TABLE 语句创建包含 ISO88591 列的表格：

```
CREATE TABLE t1 ( c1 CHAR(20) CHARACTER SET
ISO88591) ;
> java -DISO88591=SJIS test1.java
```

以下调用将 stmt 的第一列设置为字符串“abcd”，其中“abcd”编码为 SJIS。字符串getBytes方法的字符集参数是 SJIS stmt.setString(1, “abcd”);

2.10.2.2 控制异常

如果驱动程序无法转换全部或部分 SQL 参数，则可以使用 translationVerification 属性显示地定义驱动程序行为。属性的值可以为 TRUE 或 FALSE（默认值为 FALSE）。

如果 translationVerification 属性的值为 FALSE，且驱动程序无法转换全部或部分 SQL 语句，则转换为未指定。在大多数情况下，不可转换的字符被编码为 ISO88591 单字节问号（'?' 或 0x3F）。不会抛出异常或警告。

如果 translationVerification 属性的值为 TRUE，且驱动程序无法转换全部或部分 SQL 语句，则驱动程序会抛出 SQLException 信息，如下所示：

```
Translation of parameter to {0} failed. Cause: {1}
```

其中{0}为目标字符集，{1}为转换失败的原因。

更多信息，请参阅 [translationVerification 属性](#)。

2.10.3 本地化错误信息和状态信息

Type 4 驱动程序通过资源包为本地化错误信息和状态信息提供国际化支持。驱动程序使用属性文件中的一组静态字符串将错误信息和状态信息映射到其文本表

示。

2.10.3.1 本地化信息文件的文件名格式

属性文件（包含信息）的格式名为 T4Messages_xx.properties

其中 xx 为本地名称。本地名称由当前默认区域位置或语言属性定义。

Type 4 驱动程序自带错误信息和状态信息属性文件，包含英语区域位置的错误和状态信息的文本表示。该文件名为 T4Messages_en.properties。

2.10.3.2 本地化信息字符串格式

本地化信息文件包含以下形式的字符串：

```
message=message_text
```

示例

```
driver_err_error_from_server_msg=An error was returned  
from the server.
```

```
Error: {0} Error detail: {1}
```

其中 message 是 driver_err_error_from_server_msg。

message_text 是 An error was returned from the server.

```
Error: {0} Error detail: {1}
```

message_text 中的 {n} 等于 1、2、3... 依此类推，它是运行时由 Type 4 驱动程序填充的占位符。任何转换必须包括这些占位符。

2.10.3.3 如何创建本地化信息文件

1. 提取位于 jdbcT4 - *.jar 文件中的 T4Messages_en.properties。

示例

在 unix 命令行使用 java 工具 jar:

```
jar -x T4Messages_en.properties < jdbcT4-*.jar
```

2. 复制文件。
3. 编辑文件，将英语文本替换成您所处区域的语言文本。
4. 保存文件，遵循本地化信息文件文件名格式中的要求命名该文件。
5. 将文件放在类路径中任何目录，运行 JDBC 应用程序。

为了运行用户应用程序，新信息文件可以保存在类路径的任何位置。

运行时，如果驱动程序无法读取信息属性文件，则它将使用属性的 message 作为信息文本。更多信息，请参阅[本地化信息字符串格式](#)。

3. Type 4 驱动程序属性

3.1 Type 4 驱动程序属性概述

下表总结了影响客户端操作的 Type 4 驱动程序属性。更多信息，请点击下表中的属性名称。

注意：大部分特定属性适用于 DataSource 对象、DriverManager 对象和 ConnectionPoolDataSource 对象（特例会另外注明）。

3.1.1 客户端属性

3.1.1.1 连接控制属性

属性名称	说明	默认值
dataSourceName	指定注册的 DataSource 或 ConnectionPoolDataSource 名称（只能在 DriverManager 对象中设置）。	无
loginTimeout	设置可以尝试连接的时间上限。如果超过该值，连接将断开。	60（秒）
networkTimeout	设置驱动程序等待数据库服务器回复的时间上限。	0（未指定网络超时）

3.1.1.2 池管理属性

属性名称	说明	默认值
initialPoolSize	当连接池和 Type 4 驱动程序共同使用时，设置初始连接池大小（忽略通过 ConnectionPoolDataSource 对象进行的连接）。	-1（不创建初始连接池）
maxIdleTime	设置在连接关闭前，物理连接在连接池中保持空闲的秒数。	0（指定为无限制）
maxPoolSize	设置连接池可容纳物理连接的最大数量。	-1（禁用连接池）
maxStatements	设置连接池应缓存的 PreparedStatement 对象的总数。	0（禁用语句池）
minPoolSize	设置连接池中物理连接的最小数量。	-1（忽略 minPoolSize 值）

3.1.1.3 国际化属性

属性名称	说明	默认值
clientCharset	设置数据库客户端显示字符串的字符集	如未设置，默认为列定义时的字符集。
ISO88591	设置对应 ISO88591 字符集的字符集映射。	ISO88591_1

3. Type 4 驱动程序属性

KANJI	设置对应 KANJI 字符集的字符集映射。	SJIS (即 shift-JIS, 日语)
KSC5601	设置对应 KSC5601 字符集的字符集映射。	ECU_KR (即 KSC 5601, ECU 编码, 韩语)
language	设置错误信息使用的语言。	无
translationVerification	如果驱动程序无法转化全部或部分 SQL 语句或 SQL 参数, 则定义驱动程序的行为。	FALSE

3.1.1.4 日志记录和跟踪属性

属性名称	说明	默认值
T4LogFile	设置 Type 4 驱动程序的日志记录文件名称。	名称由以下模式定义: %h/t4jdbc%u.log
T4LogLevel	设置日志记录级别以控制 Type 4 驱动程序的日志记录输出。	关闭

3.1.1.5 其他客户端属性

属性名称	说明	默认值
description	指定注册的源名称。	无
fetchBufferSize	从 ResultSet 对象中批量获取行。	4 千字节
properties	指定属性文件的位置, 该属性文件包含键值对	无

3. Type 4 驱动程序属性

	值对,它在配置 Type 4 驱动程序时指定属性值。	
roundingMode	指定 Type 4 驱动程序的取整行为。	ROUND_DOWN

3.1.2 服务器端属性

下表总结了影响服务器端操作的 Type 4 驱动程序属性。

注意: 大部分特定属性适用于 DataSource 对象、DriverManager 对象和 ConnectionPoolDataSource 对象 (特例会另外注明)。

3.1.2.1 Type 4 驱动程序服务器端属性

属性名称	说明	默认值
catalog	如果 SQL 对象未完全限定,则设置默认 catalog,用于访问在 SQL 语句中被引用的 SQL 对象。	无。在当前版本中必须为“TRAFODION”。
connectionTimeout	设置在 DCS 断开连接前,连接可空闲的时间(单位为秒)。	-1 (使用服务器数据源上设置的 ConnTimeout 值)。
password	设置传递至数据库的密码。该密码可更改。	空字符串。
schema	如果 SQL 对象未完全限定,则设置数据库 schema,用于访问在 SQL 语句中被引用的 SQL 对	无。

3. Type 4 驱动程序属性

	象。	
url	设置数据库的 URL 值。 只能在 DriverManager 对象 中设置。	无。
user	设置数据库的用户值。	无。

3.2 如何指定 JDBC Type 4 属性

Type 4 JDBC 驱动程序属性配置驱动程序，这些属性可以在数据源、连接 URL（数据库的主 IP 地址或主机名）、属性文件或 java 命令行中指定。

Java 属性的格式为 `key=value`

运行时，驱动程序会查找特定的一组属性键，并根据它们的关联值执行操作。

3.2.1 设置属性

- 对于通过 `DataSource` 或 `ConnectionPoolDataSource` 进行的连接，在 `DataSource` 或 `ConnectionPoolDataSource` 对象中设置属性。
- 对于 `DriverManager` 类，按照以下任一方法设置属性：
 - 在命令行中使用选项 `-Dproperty_name = property_value`。
 - 在 `DriverManager` 类的 `getConnection()` 方法中使用 `java.util.Properties` 参数。

3.2.2 创建和使用属性文件

为了配置连接，JDBC 应用程序使用包含 JDBC 驱动程序属性的文件来提供属性值，该属性文件作为 java 命令行参数传递。命令行中输入属性文件的格式为：
`-Dt4jdbc.properties=<path of the properties file on disk>`

示例

```
-Dt4jdbc.properties=C:\temp\t4props\myprops.properties
```

创建文件时，使用编辑器输入属性值。

属性文件中的条目必须有 `property_name=property_value` 键值对格式：
`property_name=property_value`

示例

3. Type 4 驱动程序属性

```
maxStatements=1024
```

为了配置 DataSource 连接，属性文件可能包含以下属性名称和值：

```
url=jdbc:t4jdbc://<primary IP addr or host name of
database>:23400/
user=database_username
password=mypassword
description=<a string>
catalog=TRAFODION
schema=myschema
maxPoolSize=20
minPoolSize=5
maxStatements=20
loginTimeout=15
initialPoolSize=10
connectionTimeout=10
T4LogLevel=OFF
T4LogFile=/mylogdirectory/mylogfile
```

3.2.3 在命令行中设置属性

在命令行通过 `java -D` 选项指定 Type 4 驱动程序属性时，该属性必须包含前缀 `t4jdbc`。

此符号包括句点 (.)，确保所有 Type 4 驱动程序属性名称在 Java 应用程序中是唯一的。

示例

`maxStatements` 属性更改为：

```
-Dt4jdbc.maxStatements=10
```

3.2.4 属性优先级

如果特定属性由应用程序设置为若干种方式, 则被使用的值取决于该值如何被设置 (按照以下步骤) :

1. 设置 DataSource 对象、 DriverManager 对象或 ConnectionPoolDataSource 对象。
2. 通过 DriverManager 类中 getConnection 方法的 java.util.Properties 参数进行设置。
3. 在 t4jdbc.properties 属性指定的属性文件中设置属性。
4. 在 java 命令行中设置 -Dt4jdbc.property_name=<property value>。

4. Type 4 驱动程序属性说明

以下属性说明按英文首字母排序。更多信息，请参阅 [Type 4 驱动程序属性概述](#)。

4.1 catalog 属性

如果 SQL 对象未完全限定，则 catalog 属性设置默认 catalog，该默认 catalog 用于访问 SQL 语句中被引用的 SQL 对象。

在 DataSource 对象、ConnectionPoolDataSource 对象或

DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅 [如何指定 JDBCType 4 属性](#)。

Data type: String

Default: none

示例

指定 catalog TRAFODION:

```
catalog=TRAFODION
```

4.2 connectionTimeout 属性

connectionTimeout 属性设置在 DCS 断开连接前，连接可空闲的时间（单位为秒）。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: int

Units: seconds

Default: -1 (Use the ConnTimeout value set on the server-side data source.)

Range: -1, 0 to 2147483647

- 零 (0) 指定超时值为无穷大。
- 如果连接设置允许，非零正值将覆盖 EsgynDB 数据源上设置的值。
- 负值可视为-1。

示例

考虑以下情况。

即使连接未使用，它依然会占用资源。应用程序将放弃连接，即应用程序使用连接之后，不会关闭连接。

但是，您可通过 connectionTimeout 属性将其设置为 300 秒后自动关闭。如果连接在 300 秒内未被使用，它也会自动关闭。

设置 connectionTimeout 属性为 300 秒：

```
connectionTimeout=300
```

4.3 clientCharset 属性

clientCharset 属性设置数据库客户端显示字符串的字符集。如未设置，默认为列定义时的字符集。

该属性可在 url 连接字符串后设置。

示例

```
jdbc:t4jdbc://<ip or host>:23400/:clientCharset=UTF-8
```

4.4 fetchBufferSize 属性

fetchBufferSize 属性设置读取缓存的大小（单位为 KB）。

适用场景：对语句成功执行 executeQuery() 操作后，从 ResultSet 对象获取行。

在 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: short

Default size: 4

Range: 4 through 32767

- 零值和负值为默认值。
- Type 4 驱动程序确保内部获取的行数大于或等于行大小的最小值（使用 setFetchSize 方法设置），且存储在内存中的行数由 setFetchSize（使用属性设置）指定。

示例

```
fetchBufferSize=32
```

4.5 initialPoolSize 属性

当连接池与 Type 4 驱动程序共同使用时，initialPoolSize 属性设置初始连接池大小。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

当请求进行第一次连接时，驱动程序为每个连接池创建 n 个连接（n 是 initialPoolSize）。例如，如果将数据源的 initialPoolSize 设置为 5，则在应用程序第一次调用数据源的 getConnection() 方法时，驱动程序将尝试创建 5 个连接，并把它们放在同一池中。

Data type: int

Units: number of physical connections

Default: -1 (Do not create an initial connection pool.)

Range: -1 to maxPoolSize

- 任何负值等同-1。
- 值可以小于 minPoolSize，但不能大于 maxPoolSize。如果指定的值大于 maxPoolSize，则使用 maxPoolSize 属性值。

示例

```
initialPoolSize=10
```

4.6 ISO88591 属性

ISO88591 字符集映射属性到相应的 SQL ISO88591 字符集，这是一个适用于字符数据类型的单字节 8 位字符集。该属性支持英语和其他西欧语言。更多信息，请参阅[国际化支持](#)。

在 DataSource 对象或 DriverManager 对象中设置该属性。该属性不适用于通过 ConnectionPoolDataSource 对象进行的连接。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: String

Default: ISO88591_1

该值可以是任何有效的 Java 标准名称（在 [Java 文档](#) 的“java.io 和 java.lang API 标准名称”中列出）。更多信息，请参阅[国际化支持](#)。

4.7 KANJI 属性

KANJI 字符集映射属性到相应的 SQL KANJI 字符集，这是日本主机上广泛使用的双字节字符集。该属性是 Shift JIS 的一个子集：双字符部分。该属性的编码为 big endian。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: String

Default: SJIS (which is shift-JIS, Japanese)

示例

```
java -Dt4jdbc.KANJI=SJIS
```

更多信息，请参阅[国际化支持](#)。

4.8 KSC5601 属性

KSC5601 字符集映射属性到相应的 SQL KSC5601 字符集, 它是一个双字节字符集。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息, 请参阅 [如何指定 JDBCType 4 属性](#)。

Data type: String

Default: ECU_KR (which is KS C 5601, ECU encoding, Korean)

该值可以是任何有效的 Java 标准名称 (在 [Java 文档](#) 的 “java.io 和 java.lang API 标准名称” 中列出)。

示例

```
java -Dt4jdbc.KSC5601=ECU_KR
```

更多信息, 请参阅 [国际化支持](#)。

4.9 language 属性

language 属性设置错误消息的语言。更多信息，请参阅[本地化错误信息和状态信息](#)。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: String

Default: none

该值可以是 [Java 文档](#) 的“java.io 和 java.lang API 标准名称”列中列出的任何有效的 Java 标准名称。

示例

要将语言设置为 shift-JIS，日语：

```
language=SJIS
```

4.10 loginTimeout 属性

loginTimeout 属性设置在连接断开前，尝试连接的时间上限（单位为秒）。当尝试连接的时间大于设置值，连接将断开。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: int

Units: seconds

Default: 60

Range: 0 to 2147483647

如果将该值设置为 0，则表明未指定该值。

4.11 maxIdleTime 属性

maxIdleTime 属性设置连接被关闭前，它在连接池中保持空闲的秒数。0 表示无限制。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: int

Units: seconds

Default: 0 (No timeout)

Range: 0 through 2147483647

任何负值等同于 0，表示没有时间限制。

示例

将最大空置时间设置为 5 分钟（300 秒）：

```
java -Dt4jdbc.maxIdleTime=300
```

4.12 maxPoolSize 属性

maxPoolSize 属性设置连接池容纳连接的最大数量，包括空闲连接和正在使用的连接。

当连接的数量达到上限时，Type 4 驱动程序将抛出 `SQLException` 并发送消息，提示已达到上限。

在 `DataSource` 对象、`ConnectionPoolDataSource` 对象或

`DriverManager` 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: int

Units: number of physical connections

Default: -1 (Disables connection pooling.)

Range: -1, 0 through 2147483647, but greater than minPoolSize

maxPoolSize 属性的值决定连接池的使用情况：

- 任何负值等同于-1，值为-1时将禁用连接池。
- 0表示无上限。
- 任何小于 minPoolSize 的正值都将更改为 minPoolSize 值。

4.13 maxStatements 属性

maxStatements 属性设置连接池缓存 PreparedStatement 对象的总数，包括未使用的对象和正在使用的对象。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: int

Units: number of objects

Default: 0 (Disables statement pooling.)

Range: 0 through 2147483647

0 表示禁用语句池，任何负值等同于 0。

提示： 由于语句池能显著提高应用程序性能，建议您启用语句池。

注意： 语句池只有在启用连接池时才有效。

示例

指定语句池：

```
maxStatements=10
```

4.14 minPoolSize 属性

minPoolSize 属性限制空闲连接池中连接的数量。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: int

Default: -1 (The minPoolSize value is ignored.)

Range: -1, 0 through n, but less than maxPoolSize

- 任何负值等同于-1。
- 任何大于 maxPoolSize 的值都将被更改为 maxPoolSize 值。
- 当 maxPoolSize 为-1 且 minPoolSize 为-1 时:
 - 当空闲池中的物理连接数达到 minPoolSize 值时，Type 4 驱动程序关闭后续连接，不将它们添加到空闲连接池。
 - 0 表示连接未关闭。当连接关闭时，连接始终被添加到空闲连接池。

示例

将 minPoolSize 的值设置为 1，确保始终保留一个连接：

```
minPoolSize=1
```

4.15 networkTimeout 属性

networkTimeout 属性设置驱动程序等待数据库服务器回复的时间限制。当尝试操作的时间长于设置值（单位为秒）时，驱动程序将停止等待回复，并向应用程序返回 SQLException。

在 DataSource 对象、ConnectionPoolDataSource 对象或

DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

注意：

请谨慎使用该属性。网络超时会导致 Type 4 驱动程序和连接服务器之间的套接字连接超时。

如果服务器在处理事务或 SQL 操作，则服务器将继续处理，直至事务或操作失败、事务管理器超时或服务器收到 Type 4 驱动程序客户端不存在的消息。

网络超时可能导致正在运行的事务或操作在失败或回滚之前持续运行很长时间。但由于网络超时，连接将不可用。

Data type: int

Units: seconds

Default: 0 (No network timeout is specified.)

0 through to 2147483647

4.16 password 属性

password 属性设置传递到 DCS 服务器的密码或更改该密码。密码在传递到服务器时会被加密。

指定密码的格式为：

```
password=old [, new, new ]
```

- old 表示当前密码
- new 表示新密码。密码长度必须为 6 到 8 个字符，不能包含双引号 (")。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: String

Default: empty string

示例

```
password=eye0weU$
```

4.17 properties 属性

properties 属性指定属性文件的位置，该文件包含为配置 Type 4 驱动程序指定属性值的关键字-值对。更多信息，请参阅[创建和使用属性文件](#)。

4.18 reserveDataLocators 属性

reserveDataLocators 属性设置数据定位符的数量，这些数据定位符为在 LOB 表中存储数据的进程保留。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: int

Units: number of data locators to be reserved

Default: 100

Range: 1 to 9,223,372,036,854,775,807 ($2^{63} - 1$)

请勿设置比实际所需数据定位符数量大很多的值。如果指定的值为 0 或更小，则使用默认值 (100)。

根据正在执行的应用程序配置文件，设置 reserveDataLocators 值。

如果应用程序插入大量 LOB 项，则更大的 reserveDataLocators 值能防止 LOB 表中的 ZZ_DATA_LOCATOR 值频繁更新。

但是，如果应用程序只插入少量 LOB 项，则 reserveDataLocators 值越小越好。

如果使用较大的值，则可能在 LOB 表中出现 holes（未使用的数据定位符），这些 holes 表示未使用的空间。

此外，过高的值将阻止使用 LOB 表的其他 Type 4 应用程序来保留数据定位符，所以应避免将该值设为过高（例如，万亿）。

如果您想更改 JDBC 应用程序的 reserveDataLocators 值，请使用命令行工具。

示例

以下命令为 myProgramClass 程序类保留 150 个数据定位符。

```
java -DTraft4jdbc.reserveDataLocators=150 myProgramClass
```

4.19 roundingMode 属性

roundingMode 属性指定 Type 4 驱动程序的舍入行为。

例如，如果数据为 1234.127，列定义为 NUMERIC(6,2)，应用程序执行 setDouble() 和 getDouble()，则返回的值为 1234.12，由默认舍入模式 ROUND_DOWN 截尾。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅 [如何指定 JDBCType 4 属性](#)。

Data type: String

Default: ROUND_DOWN

roundingMode 的值为：

ROUND_CEILING

ROUND_DOWN

ROUND_FLOOR

ROUND_HALF_DOWN

ROUND_HALF_EVEN

ROUND_HALF_UP

ROUND_UNNECESSARY

ROUND_UP

更多有关舍入模式值的信息，请参阅 [java.math.BigDecimal](#) 文档。

如果应用程序设置了错误的 roundingMode 值，则 Type 4 驱动程序不会抛出错误，而是使用 ROUND_DOWN 值。

4. Type 4 驱动程序属性说明

如果数据截尾时应用程序收到 DataTruncation 异常，请将 roundingMode 属性设置为 ROUND_UNNECESSARY。

4.20 schema 属性

schema 属性设置数据库 schema, 这些 schema 访问在 SQL 语句中被引用的 SQL 对象 (如果 SQL 对象未完全限定)。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息, 请参阅 [如何指定 JDBCType 4 属性](#)。

Data type: String

Default: none

示例

```
schema=sales
```

4.21 T4LogFile 属性

T4LogFile 属性设置 Type 4 驱动程序的日志文件的名称。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: String

默认文件名由 %h/t4jdbc%u.log 模式定义，其中：

- / 表示本地路径名分隔符。
- %h 表示 user.home 系统属性的值。%u 表示解决冲突的唯一编号。

系统的任何有效文件名都是允许的。

如果明确指定了日志文件，则每次使用该文件名建立 FileHandler 时，该文件都会被覆盖。

如需保留之前创建的日志文件，请使用标准“java.util.logging”文件语法将唯一的数字追加到每个日志文件。

示例 1

您可以在数据源中查找到以下属性：

```
T4LogFile = C:/temp/MyLogFile%u.log
```

该名称使 Type 4 驱动程序创建新的日志文件，为每个通过该数据源构造的连接使用唯一名称。

示例 2

```
C:/temp/MyLogFile43289.log
```

```
C:/temp/MyLogFile87634.log
```

4. Type 4 驱动程序属性说明

C:/temp/MyLogFile27794.log

如果显示地指定未完全限定的日志文件，则 Type 4 驱动程序将在当前工作目录中创建该文件，例如，在调用 JVM 的目录中。

更多有关 java.util.logging 的信息，请参阅[日志概述](#)。

4.22 T4LogLevel 属性

T4LogLevel 属性设置日志级别，这些日志级别控制 Type 4 驱动程序的日志输出。Java 程序包 `java.util.logging` 在驱动程序中记录并跟踪错误消息。

在 `DataSource` 对象、`ConnectionPoolDataSource` 对象或 `DriverManager` 对象中设置该属性。更多有关如何设置属性的信息，请参阅[如何指定 JDBCType 4 属性](#)。

Data type: String

Default: OFF

日志级别

级别	说明
OFF	表示关闭日志。该选项为默认值。
SEVERE	表示严重故障。通常为 Type 4 驱动程序产生 SQL 异常时。
WARNING	表示潜在问题。通常为 Type 4 驱动程序产生 SQL 警告时。
INFO	提供信息性消息。 通常涉及连接池、语句池和资源使用，该信息能帮助调整应用程序性能。
CONFIG	提供静态配置信息，包括属性值和其他 Type 4 驱动程序配置信息。
FINE	提供 Type 4 驱动程序 API 中描述的 Type 4 驱动程序方法的跟踪信息。跟踪级别等于调用 <code>DriverManager</code> 类的 <code>setLogWriter()</code> 方法或 <code>DataSource</code> 类时提供的跟踪级别。
FINER	表示由内部 Type 4 驱动程序方法提供的详细跟踪信息，这些信息可用于调试 Type 4 驱动程序。
FINEST	表示非常详细的跟踪信息。驱动程序提供详细的内部数据信息，可用于调试 Type 4 驱动程序。

4. Type 4 驱动程序属性说明

ALL	记录所有信息。
------------	---------

示例

如需启用跟踪功能，请使用命令行工具指定 `traft4jdbc.T4LogLevel` 属性：

```
-Dt4jdbc.T4LogLevel=FINE
```

4.22.1 T4LogLevel 注意事项

- 如果应用程序使用 AppServer 定义安全管理器，则您必须在 `java.policy` 文件中授予 `LoggingPermission`，如下所示：

```
permission java.util.logging.LoggingPermission  
"control", "" ;
```

- 程序启动期间，Type 4 驱动程序不继承 `java.util.logging.FileHandler.level` 的设置。

4.23 translationVerification 属性

如果驱动程序无法转化全部或部分 SQL 语句或 SQL 参数，则 translationVerification 属性定义驱动程序的行为。

在 DataSource 对象、ConnectionPoolDataSource 对象或 DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅 [如何指定 JDBCType 4 属性](#)。

您能将该值设置为 TRUE 或 FALSE。

Data type: String

Default: FALSE

值	含义	说明
FALSE	驱动程序无法转化所有或部分 SQL 语句，转化未指定。	在大多数情况下，无法转化的字符被编码为 ISO88591 单字节问号 (? 或 0x3F)。不会抛出异常或产生警告。
TRUE	驱动程序无法转化全部或部分 SQL 语句或参数。	驱动程序抛出一个 SQLException，如下所示： <pre>Translation of parameter to {0} failed. Cause: {1}</pre> 其中 {0} 为目标字符集，{1} 为转化失败的原因。

提示：如果 translationVerification 属性值为 TRUE，则该进程将使用更多系统资源。如需提高性能，请将该属性设置为 FALSE。

更多信息，请参阅 [国际化支持](#)。

4.24 url 属性

url 属性设置数据库的 URL 值。

该属性在 DriverManager 对象中使用，URL 的格式为：

```
jdbc:t4jdbc://<primary IP addr or hostname of  
database>:23400/[:]  
[ property=value [ ; property2=value ] ... ]
```

其中，<primary IP_addr or hostname of database>:23400>指定数据库的位置。

Data type: String

Default: none

示例

```
url=jdbc:t4jdbc://mynode.mycompanynetwork.net:23400/
```

4.24.1 url 属性注意事项

- 如果未指定 url 参数并调用 DriverManager.getConnection()，则 Type 4 驱动程序将抛出 SQLException。
- 如果您在 url 中使用文字 IPV4 或 IPV6 地址，请注意：
 - 仅针对 IPV6：用方括号 ([]) 将地址括起来。
 - 根据 IPV4 和 IPV6 标准，端口号可选。
 - 数据库默认端口号为 23400。

4.25 user 属性

user 属性设置连接服务的角色值。

有效的角色名称必须对通过连接服务进行访问的 SQL 数据拥有充分访问权限。

在 DataSource 对象、ConnectionPoolDataSource 对象或

DriverManager 对象中设置该属性。更多有关如何设置属性的信息，请参阅

[如何指定 JDBCType 4 属性](#)。

Data type: String

Default: empty string

示例

```
user=System_rolename
```

5. Type4 驱动程序高级属性说明

以下属性是 EsgynDB 独有的 Type4 高级属性，按英文首字母排序。

5.1 客户端缓存

为提升静态数据表的查询性能，减少网络交互次数，因此将静态表数据加载至驱动端内存，当访问到此静态表数据时，并且 SQL 语句中仅包含单表查询，针对这种数据加载至驱动端缓存，此后每条 SQL 都会访问这张客户端缓存表。

客户端缓存指的是将表数据加载至 JDBC 驱动内存。

静态数据指的是是一些极少改变的数据表，例如配置表。

使用客户端缓存，针对反复使用的静态数据表的查询性能有两倍以上的提升。

目前该功能仅限于静态数据或极少改动的配饰数据库。

使用步骤：

1. 设置 URL 属性：

```
jdbc:t4jdbc://[ip or
host]:23400/:cacheTable=table1,table2,table3;
maxCachedRows=1000; cacheRefreshIntervalSecs=-1
```

属性说明如下：

名称	默认值	参数说明
cacheTable	null	指定需要缓存的表，如果有多张表则以逗号分隔
maxCachedRows	1000 (行)	指定最大缓存表大小，行数超过1000 则不缓存此表

cacheRefreshIntervalSecs	-1 (秒)	默认不刷新
--------------------------	--------	-------

2. 引入 h2database 的依赖包 (或直接下载 jar 包引入)

```
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <version>1.4.197</version>
</dependency>
```

配置说明:

1. 指定 URL 关键字 cacheTable (大小写敏感)
2. 当需要缓存多个表时, 以逗号分隔
3. 必须要引入 h2database 的依赖包 (maven pom 或 jar 方式)、

目前使用限制:

- 1 cacheRefreshIntervalSecs>0 采用了 Timer 的方式, 开启后应用不会软退出, 需要 kill -9 杀掉进程。
- 2 个别数据类型不支持, 例如: bigint

5.2 clipVarchar 属性

该属性可提高 varchar 列数据的 IO 性能, 在某些情况下可提高查询和插入的性能。

用法:

1. jdbc:t4jdbc://[ip or host]:23400/:clipVarchar=1
对所有 varchar 列开启 clipVarchar 功能
2. jdbc:t4jdbc://[ip or host]:23400/:clipVarchar=200

对列宽大于等于 200 的 varchar 列开启 clipVarchar 功能

默认为 0，即不开启 clipVarchar 功能

5.3 sessionDebug 属性

该属性可开启 mxosrvr 端的 sessionDebug 日志，仅适用于开发及测试环境。

用法：

```
jdbc:t4jdbc://ip:23400/:sessionDebug=1
```

开启 mxosrvr 端的日志。

默认为 0，即不打开。

5.4 specifiedServer 属性

该属性用于 jdbc 客户端连接指定的 mxosrvr。

用法：

```
jdbc:t4jdbc://ip:23400/:specifiedServer=[ip or  
hostname]:port
```

5.5 traceTransTime 属性

该属性用于全链路追踪日志开启，可调查慢 SQL，仅适用于开发及测试环境。

用法：

1. jdbc:t4jdbc://ip:23400/:traceTransTime=200

日志中打印超过 200ms 的事务执行语句及时间。

2. jdbc:t4jdbc://ip:23400/:traceTransTime=0

日志中打印所有事务执行语句及时间。

默认为-1，即不打开此功能。

6. Type 4 驱动程序兼容性

6.1 兼容性概述

Type 4 驱动程序符合 JDBC 3.0 API 规范，但在某些方面，该驱动程序与 JDBC 标准不同。

本小节描述了不支持的 JDBC 方法、与规范不同的方法和功能，以及标准 JDBC 之外的 EsgynDB 扩展功能。

本小节未描述符合规范的 JDBC 功能。

此外，本章还列出 EsgynDB JDBC Type 4 驱动程序不支持的 EsgynDB SQL 功能、其他不支持的功能和限制。

6.2 不支持的功能

java.sql 程序包中的某些方法抛出 SQLException，并包含信息 Unsupported feature - <method-name>:，如下所示：

方法	说明
<code>CallableStatement.getArray(int parameterIndex)</code>	不支持特定的 CallableStatement 方法。
<code>CallableStatement.getArray(String parameterName)</code>	
<code>CallableStatement.getBlob(int parameterIndex)</code>	
<code>CallableStatement.getBlob(String parameterName)</code>	
<code>CallableStatement.getClob(int parameterIndex)</code>	
<code>CallableStatement.getClob(String parameterName)</code>	
<code>CallableStatement.getObject(int parameterIndex, Map map)</code>	
<code>CallableStatement.getObject(String parameterName, Map map)</code>	
<code>CallableStatement.getRef(int parameterIndex)</code>	
<code>CallableStatement.getRef(String parameterName)</code>	
<code>CallableStatement.getURL(int parameterIndex)</code>	
<code>CallableStatement.getURL(String</code>	

6. Type 4 驱动程序兼容性

<code>parameterName)</code>	
<code>CallableStatement.executeBatch()</code>	
<code>Connection.releaseSavepoint(Savepoint savepoint)</code> <code>Connection.rollback(Savepoint savepoint)</code>	不支持特定的 Connection 方法。
<code>Connection.setSavepoint()</code> <code>Connection.setSavepoint(String name)</code>	
<code>PreparedStatement.setArray(int parameterIndex, Array x)</code> <code>PreparedStatement.setRef(int parameterIndex, Ref x)</code> <code>PreparedStatement.setURL(int parameterIndex, URL x)</code>	不支持特定的 PreparedStatement 方法。
<code>ResultSet.getArray(int columnIndex)</code> <code>ResultSet.getArray(String columnName)</code> <code>ResultSet.getObject(int columnIndex, Map map)</code> <code>ResultSet.getObject(String columnName, Map map)</code> <code>ResultSet.getRef(int columnIndex) ResultSet.getRef(String columnName)</code>	不支持特定的 ResultSet 方法。
<code>ResultSet.getURL(int columnIndex)</code> <code>ResultSet.getURL(String columnName)</code> <code>ResultSet.updateArray(int columnIndex)</code> <code>ResultSet.updateArray(String</code>	

6. Type 4 驱动程序兼容性

columnName) ResultSet.updateRef(int columnIndex) ResultSet.updateRef(String columnName)	
Statement.getQueryTimeout() Statement.setQueryTimeout()	不支持特定的 Statement 方法。

java.sql 程序包中的某些方法抛出 SQLException, 并包含信息 Auto generated keys not supported:, 如下所示:

方法	说明
Connection.prepareStatement(String sql, int autoGeneratedKeys)	不支持自动生成的键。
Connection.prepareStatement(String sql, int[] columnIndexes)	
Connection.prepareStatement(String sql, String[] columnNames)	
Statement.executeUpdate(String sql, int autoGeneratedKeys) Statement.executeUpdate(String sql, int[] columnIndexes)	
Statement.executeUpdate(String sql, String[] columnNames)	
Statement.getGeneratedKeys()	

java.sql 程序包中的某些方法抛出 SQLException, 并包含信息 Data type not

6. Type 4 驱动程序兼容性

supported:, 如下所示:

方法	说明
<code>CallableStatement.getBytes (int parameterIndex)</code>	不支持特定的数据类型。
<code>CallableStatement.setBytes (String parameterIndex, bytes[] x)</code>	

java.sql 程序包中的某些接口未在 Type 4 驱动程序中实现, 如下所示:

方法	说明
<code>java.sql.Array</code>	EsgynDB 不支持底层数据类型。
<code>java.sql.Ref</code>	
<code>java.sql.Savepoint</code>	
<code>java.sql.SQLData</code>	
<code>java.sql.SQLInput</code>	
<code>java.sql.SQLOutput</code>	
<code>java.sql.Struct</code>	

javax.sql 程序包中的某些接口未在 Type 4 驱动程序中实现, 如下所示:

方法	说明
<code>javax.sql.XAConnection</code>	如 JDBC 3.0 API 规范中所述, 分布式事务处理尚未实现。
<code>javax.sql.XADataSource</code>	

更多有关偏差方法的信息, 请参阅[偏差](#)。

6.3 偏差

下表列出了执行方式与 JDBC 规范不同的方法。

如果方法中的参数被忽略，则 Type 4 驱动程序不会抛出 `SQLException`，允许应用程序继续处理，但可能无法获得预期结果。

虽然这些方法与规范不同，但不一定会抛出 `SQLException`。

注意：由于 EsgynDB SQL 不支持 `SQL_ROWVER`（该列函数在指定表中返回一列或多列。如果任何事务的行的值被更新，则数据源将自动更新该列函数），`java.sql.DatabaseMetaData.getVersionColumns()` 方法模拟 `java.sql.DatabaseMetaData.getBestRowIdentifier()` 方法。

方法	说明
<code>java.sql.DatabaseMetaData.getVersionColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)</code>	<p>将列添加至列数据，由于 EsgynDB SQL 不支持以下类型的列类型，所以将值设置为 NULL。</p> <ul style="list-style-type: none"> • <code>SCOPE_CATALOG</code> • <code>SCOPE_SCHEMA</code> • <code>SCOPE_TABLE</code> • <code>SOURCE_DATA_TYPE</code>
<code>java.sql.DatabaseMetaData.getTableTypes(String catalog, String schemaPattern, String[] types)</code>	<p>将列添加至列数据，由于 EsgynDB SQL 不支持以下类型的列类型，所以将值设置为 NULL。</p> <ul style="list-style-type: none"> • <code>TYPE_CAT</code> • <code>TYPE_SCHEMA</code> • <code>TYPE_NAME</code> • <code>SELF_REFERENCING_COL_NAME</code>

	<ul style="list-style-type: none"> REF_GENERATION
<pre>java.sql.DatabaseMetaData.g etUDTs(String catalog, String schemaPattern, String tableNamePattern, int[] types)</pre>	<p>将 BASE_TYPE 添加至列数据，由于 EsgynDBSQL 不支持基本类型，所以将该值设置为 NULL。</p>
<pre>java.sql.DatabaseMetaData.g etVersionColumns()</pre>	<p>模拟 DatabaseMetaData.getBestRowIdentifier() 方法，因为 EsgynDBSQL 不支持 SQL_ROWVER（该列函数在指定表中返回一列或多列。如果任何事务的行的值被更新，则数据源将自动更新该列函数）。</p>
<pre>java.sql.Connection.createS tatement(. . .) java.sql.Connection.prepare Statement(. . .)</pre>	<p>Type 4 驱动程序不支持滚动敏感（scroll-sensitive）结果集类型。如果应用程序请求该类型，则会触发 SQL 警告，结果集将被更改为非滚动敏感（scrollinsensitive）类型。</p>
<pre>java.sql.ResultSet.setFetch Direction(. . .)</pre>	<p>忽略 Fetch Direction 属性。</p>
<pre>java.sql.Statement.cancel()</pre>	<p>在某些实例中，删除与服务器的连接，而不是取消查询。您必须重新连接到服务器。</p> <hr/> <p>注意：如果对正在处理的语句执行 cancel()，则会删除连接，否则将保留连接。</p>

6. Type 4 驱动程序兼容性

<code>java.sql.Statement.setEscapeProcessing(. . .)</code>	EsgynDB SQL 解析转义语法，所以禁用转义处理无效。
<code>java.sql.Statement.setFetchDirection(. . .)</code>	忽略 Fetch Direction 属性。

6.4 EsgynDB 扩展

以下是在 Type 4 驱动程序中实现的在 JDBC 标准之外的 EsgynDB 扩展。

6.4.1 消息国际化

Type 4 驱动程序设计合理，各种语言都能采用 Java 消息。错误消息存储在源代码之外的单独属性文件中，并根据区域设置动态获取。不同语言的错误消息存储在基于语言 and 国家的单独属性文件中。该扩展不适用于运行 JDBC 应用程序时产生的所有消息。

更多信息，请参阅[本地化错误消息和状态消息](#)。

6.4.2 其他 DatabaseMetaData API

添加到 T4DatabaseMetaData 类的 API 提供以下功能：

- 获取表的别名。

```
public java.sql.ResultSet getSynonymInfo(String
catalog, String schema, String table) throws
SQLException
```

6.5 DatabaseMetaData 方法处理 Null 参数的一致性

本小节描述 Type 4 驱动程序如何确定 null 参数的值，该值作为 DatabaseMetaData 方法的参数值被传递。

其他数据库可能会以不同方式实现 JDBC 规范，下文解释了 Type 4 驱动程序对受影响查询的不同处理方式。

该实现适用于某些方法，这些方法使用能表示模式的参数，参数名称的格式为 attributePattern

java.sql.DatabaseMetaData 类的许多方法都会受到影响，例如，getColumns() 方法；再例如，schema 是 schemaPattern 参数中的属性，schemaPattern 是 java.sql.ResultSet.getAttributes 方法的一个参数。

```
public ResultSet getAttributes( String catalog
                               , String schemaPattern
                               , String typeNamePattern
                               , String attributeNamePattern
                               ) throws SQLException
```

如果应用程序传递一个 null 值，则 null 将按以下方式处理：

- 如果参数名称包含后缀 Pattern，则 null 被解释为 % 通配符。
- 如果参数名称不包含后缀 Pattern，则 null 被解释为该参数的默认值。

示例

catalog	默认 catalog 名称。
schemaPattern	% 通配符为所有指定 catalog 的 schema

6. Type 4 驱动程序兼容性

	查询数据。
--	-------

6.6 Type 4 驱动程序数据类型与 SQL 数据类型的对应关系

6.6.1 JDBC 数据类型

下表列出了 Type 4 驱动程序支持的 JDBC 数据类型和相应的 EsgynDB SQL 数据类型：

JDBC 数据类型	JDBC 驱动程序支持的 EsgynDB SQL	EsgynDB SQL 数据类型
Types.Array	否	不适用
Types.BIGINT	是	LARGEINT
Types.BINARY	EsgynDB SQL 映射数据类型。数据类型与创建表的数据类型不同。	CHAR (n) ¹
Types.BIT	EsgynDB SQL 映射数据类型。数据类型与创建表的数据类型不同。	CHAR (1)
Types.CHAR	是	CHAR (n)
Types.DATE	是	DATE
Types.DECIMAL	是	DECIMAL (p, s)
Types.DISTINCT	否	不适用
Types.DOUBLE	是	DOUBLE PRECISION
Types.FLOAT	是	FLOAT (p)
Types.INTEGER	是	INTEGER
Types.JAVA_OBJECT	否	不适用。
Types.LONGVARIABLE	EsgynDB SQL 映射数据类型。数据类型与创建表的数据类型不同。	VARCHAR (n) ¹
Types.LONGVARCHAR	是。最大长度为 4018。	VARCHAR [(n)]

6. Type 4 驱动程序兼容性

Types.NULL	否	不适用。
Types.NUMERIC	是	NUMERIC (p, s)
Types.REAL	是	FLOAT (p)
Types.REF	否	不适用
Types.SMALLINT	是	SMALLINT
Types.STRUCT	否	不适用
Types.TIME	是	TIME
Types.TIMESTAMP	是	TIMESTAMP
Types.TINYINT	EsgynDB SQL 映射数据类型。数据类型与创建表的数据类型不同。	SMALLINT
Types.VARBINARY	EsgynDB SQL 映射数据类型。数据类型与创建表的数据类型不同。	VARCHAR (n) ¹
Types.VARCHAR	是	VARCHAR (n)
Types.BOOLEAN	EsgynDB SQL 映射数据类型。数据类型与创建表的数据类型不同。	CHAR (1)
Types.DATALINK	否	不适用

Type 4 驱动程序将以下数据类型映射至 JDBC 数据类型 Types.OTHER:

- INTERVAL YEAR (p)
- INTERVAL YEAR (p) TO MONTH
- INTERVAL MONTH (p)

¹ EsgynDB 提供映射，ResultSet.getObject() 方法返回一个字符串对象，而不是一个字节数组。

- INTERVAL DAY (p)
- INTERVAL DAY (p) TO HOUR
- INTERVAL DAY (p) TO MINUTE
- INTERVAL DAY (p) TO SECOND
- INTERVAL HOUR (p)
- INTERVAL HOUR (p) TO MINUTE
- INTERVAL HOUR (p) TO SECOND
- INTERVAL MINUTE (p)
- INTERVAL MINUTE (p) TO SECOND
- INTERVAL SECOND (p)

6.7 浮点支持

Type 4 驱动程序仅支持在应用程序客户端和 Type 4 驱动程序之间传递的 IEEE 浮点数据。

6.8 SQLJ 支持

Type 4 驱动程序支持非自定义的 SQLJ 应用程序，不支持自定义的 SQLJ 应用程序。

6.9 Type 4 驱动程序不支持的 JDBC 3.0 功能

JDBC 3.0 不要求支持以下功能，EsgynDB JDBC Type 4 驱动程序不支持以下功能：

- 批处理语句返回多个结果
- 数据库保存点支持 (EsgynDB SQL 不提供)
- 获取自动生成的键
- 转换组和类型映射
- 连接器架构与 JDBC 3.0 SPI 之间的关系
- Type 4 驱动程序和 DCS 之间交互的安全套接字通信或加密
- 从 AppServer 到 Type 4 驱动程序的安全上下文环境 (用户名和密码) 的隐式传递
- IPV6 协议栈。在 EsgynDB 服务器端，用 ipv4 地址模拟 ipv6 地址
- 分布式事务

6.10 限制

- Type 4 驱动程序仅支持受 EsgynDB SQL 和 SPJ 支持的数据库功能。因此，Type 4 驱动程序不完全符合 JDBC 3.0 规范。
- Type 4 驱动程序的所有服务器端的管理功能都依赖于 DCS (数据连接服

务)。

7. 跟踪和日志功能

Type 4 驱动程序提供两种跟踪和日志功能：

- 由 JDBC 标准定义的标准 JDBC 跟踪和日志功能
- Type 4 驱动程序日志功能

配置 DCS 可启用服务器端跟踪（日志）功能。

7.1 标准 JDBC 跟踪和日志功能

JDBC 标准提供日志和跟踪功能，通过设置日志写入器，您能跟踪 JDBC 方法的调用情况。

如需设置日志写入器，则需对 `DriverManager` 类调用 `setLogWriter()` 方法，或对 `DataSource` 类（或 `ConnectionPoolDataSource` 类）调用 `setLogWriter()` 方法。

- `DriverManager` 日志写入器是字符输出流，`DriverManager` 建立的所有连接的所有日志和跟踪信息都打印到该字符输出流。

该输出流包括该连接方法打印的信息，以及由该连接产生的其他对象的方法打印的信息等。

`DriverManager` 日志写入器初始值为 `null`，即默认禁用日志功能。

更多有关使用 `setLogWriter` 方法的信息，请参阅 [DriverManager 类 API](#)。

- `DataSource` 日志写入器是字符输出流，该数据源的所有日志和跟踪信息都打印到该字符输出流。

该输出流包括通过对象方法打印的信息，以及由该对象产生的其他对象的方法打印的信息等。

打印到数据源特定日志写入器的信息不会打印到与

`java.sql.DriverManager` 类相关的日志写入器中。

创建 DataSource 对象时,日志写入器初始值为 null,即默认禁用日志功能。

更多有关使用 setLogWriter 方法的信息,请参阅 [DriverSource 接口 API](#)。

7.2 Type 4 驱动程序日志功能

Type 4 驱动程序日志功能允许您检索内部跟踪信息（这些信息能在调试驱动程序时使用）、捕获错误和警告信息。

除了标准 JDBC 跟踪和日志功能，Type 4 驱动程序还提供了独立的日志功能（Type 4 驱动程序日志）。

Type 4 驱动程序日志提供与标准 JDBC 跟踪和日志功能相同级别的跟踪和日志功能，并提供以下信息：

- 有关 Type 4 驱动程序的内部组件和内部跟踪信息的详情
- Type 4 驱动程序性能调试信息
- 更精细地控制日志信息的数量和类型
- 错误和警告信息

7.2.1 控制 Type 4 驱动程序日志输出

Type 4 驱动程序提供了以下两个属性，用于控制日志输出。

- `T4LogLevel`：指定日志记录的级别。
更多有关如何使用该属性的信息，请参阅 [T4LogLevel 属性](#)。
- `T4LogFile`：指定驱动程序将要写入日志信息的文件。
更多有关如何使用该属性的信息，请参阅 [T4LogFile 属性](#)。

当应用程序设置了几个属性值时，如需确定哪些设置适用，请参阅[属性规范的优先级](#)。

示例

以下属性文件将日志记录级别设置为 SEVERE，并指定日志文件名：

7. 跟踪和日志功能

T4LogLevel= SEVERE

T4LogFile=c:/T4logfile1.log

7.2.2 消息格式

跟踪输出的格式为：

```
sequence-number ~ time-stamp ~ thread-id
~ [connection-id] ~ [server-id] ~ [dialogue-id]
~ [class].[method] [(parameters)] ~ [text]
```

标识符	说明
sequence-number	呈递增顺序的唯一序列号。
time-stamp	消息的时间。 示例 10/17/2004 12:48:23
thread-id	Java VM 中的线程标识符。
connection-id	如果适用，该标识符是与消息相关的连接的唯一 ID。
server-id	如果适用，该标识符是与消息相关的连接服务器的信息。 该标识符的格式为： TCP:node-name.server-name/port-number:ODBC 其中 <ul style="list-style-type: none"> • node-name 是 EsgynDB 节点的名称 • server-name 是 EsgynDB 平台的名称 • port-number 是服务器所连接的端口 示例 TCP:\banshee-tcp.\$Z0133/46003:ODBC
dialogue-id	如果适用，该标识符用于 DCS 连接。

7. 跟踪和日志功能

<code>class</code>	如果适用，该标识符是发出日志请求的类的名称。
<code>method</code>	如果适用，该标识符是发出日志请求的方法的名称。
<code>parameters</code>	与方法相关的可选参数集。
<code>text</code>	消息的可选文本信息。

提示：波浪号 (~) 字符分隔消息文本。该分隔符允许您使用工具（例如，Excel，Word，UNIX sort 等）格式化信息。

例如，您可以根据序列号或线程 ID 将消息格式化和分类。

您可以编辑日志文件并将分隔符（波浪号）更改为任何字符。

如有需要，为了使格式可读，您能在数字（例如，thread id 和序列号）前面添加 0。

7.2.3 日志输出示例

- T4LogLevel 设置为 SEVERE 的输出：

```
00000036 ~ Dec 8, 2006 10:05:55 AM PST ~ 10 ~ 4508606 ~
null
    ~ null ~ T4Messages.createSQLException("en_US",
    "socket_write_error", "null") ~
```

- T4LogLevel 设置为 FINER 的输出：

```
0000006 ~ 10/22/2004 10:34:45 ~ 001234 ~ 0049934 ~
FetchRowSetMessage ~ marshal
    ~ Entering FetchRowSetMessage.marshal( en_US
    , 48345
    , STMT_MX_8843
    , 5
    , 4192,
    , 0
    , 0 )
```

8. 消息

8.1 消息格式

消息按 SQLCODE 顺序（首位数字）列出，包括以下内容：

SQLCODE SQLSTATE message-text

Cause [What occurred to trigger the message.]

Effect [What is the result when this occurs.]

Recovery [How to diagnose and fix the problem.]

8.2 获得帮助

某些消息没有相应的解决措施，请联系 user@trafodion.incubator.apache.org 获取帮助。

8.3 Type 4 驱动程序错误消息

8.3.1 01032 08S01

01032 08S01 Communication link failure. The server timed out or disappeared.

- **原因：**连接超时。
- **结果：**操作失败。
- **措施：**重新连接。将连接超时值设置为合适的值。

8.3.2 01056 25000

01056 25000 Invalid transaction state.

- **原因：**事务状态错误。
- **结果：**操作失败。
- **措施：**重试。

8.3.3 01118 S1008

01118 S1008 Operation canceled.

- **原因:** 操作已取消。
- **结果:** 操作失败。
- **措施:** 重试操作。

8.3.4 08001 HY000

08001 HY000 Retry attempts to connect to the datasource failed. May be ODBC server not able to register to the ODBC service process.

- **原因:** 服务器错误。
- **结果:** 操作失败。
- **措施:** 请联系 user@trafodion.incubator.apache.org 检查服务器错误日志, 分析错误和警告。

8.3.5 08004 HY000

08004 HY000 Data source rejected establishment of connection since the ODBC server is connected to a different client now

- **原因:** 已丢失与服务器的连接。服务器现已连接至其他连接。
- **结果:** 操作失败。
- **措施:** 重新连接。

8.3.6 29001 HYC00

HYC00 Unsupported feature - {0}

- **原因:** JDBC 驱动程序不支持所列功能。
- **结果:** 抛出不支持的异常, 返回 Null resultSet。
- **措施:** 删除程序未实现的函数。

8.3.7 29002 08003

29002 08003 Connection does not exist

- **原因:** 断开连接数据库后, 尝试执行操作。
- **结果:** 数据库无法访问。
- **措施:** 重新连接数据库后, 重试操作。

8.3.8 29003 HY000

29003 HY000 Statement does not exist

- **原因:** 对已关闭语句调用 `getter` 或 `exec`。
- **结果:** `getter` 或 `exec` 调用验证失败。
- **措施:** 语句可用时, 发出 `validateGetInvocation()` 或 `validateExecDirectInvocation`。

8.3.9 29004 HY024

29004 HY024 Invalid transaction isolation value.

- **原因:** 将事务隔离级别设置为无效值。
- **结果:** `TraffT4Connection.setTransactionIsolation` 未设置事务隔离值。
- **措施:** 有效的隔离值是 `SQL_TXN_READ_COMMITTED`、`SQL_TXN_READ_UNCOMMITTED`、`SQL_TXN_REPEATABLE_READ` 和 `SQL_TXN_SERIALIZABLE`。
如果未指定隔离值, 则默认值为 `SQL_TXN_READ_COMMITTED`。

8.3.10 29005 HY024

29005 HY024 Invalid ResultSet type

- **原因:** 设置了无效的 `ResultSet` 类型值。
- **结果:** 调用了 `resultSetType` 参数的 SQL 语句失败。

- **措施:** 有效的 `ResultSet` 类型是 `TYPE_FORWARD_ONLY`、`TYPE_SCROLL_INSENSITIVE` 和 `TYPE_SCROLL_SENSITIVE`。

8.3.11 29006 HY000

29006 HY000 Invalid Result Set concurrency

- **原因:** 设置了无效的 `result-set` 并发值。
- **结果:** 调用 `resultSetConcurrency` 的 `TrafT4Statement` 失败。
- **措施:** 有效的 `resultSetConcurrency` 值为 `CONCUR_READ_ONLY` 和 `CONCUR_UPDATABLE`。

8.3.12 29007 07009

29007 07009 Invalid descriptor index

- **原因:** `ResultSetMetadata` 列参数或 `ParameterMetaData` `param` 参数不在描述符范围。
- **结果:** 未返回 `ResultSetMetadata` 或 `ParameterMetaData` 方法数据。
- **措施:** 验证提供给方法的列或参数。

8.3.13 29008 24000

29008 24000 Invalid cursor state

- **原因:** 连接关闭时调用 `ResultSet` 方法。
- **结果:** 方法调用不成功。
- **措施:** 确保在调用 `ResultSet` 方法之前连接可用。

8.3.14 29009 HY109

29009 HY109 Invalid cursor position

- **原因:** 当 `ResultSet` 行游标位于插入行时, 执行了 `deleteRow()` 方法、`updateRow()` 方法或 `cancelRowUpdates` 方法。或当 `ResultSet` 行游标不在插入行, 执行了 `insertRow()` 方法。
- **结果:** 行改变且游标操作失败。

- **措施:** 如需插入行, 请将游标移动至该插入行。如需删除、取消或更新行, 请移除该插入行的游标。

8.3.15 29010 07009

29010 07009 Invalid column name

- **原因:** 列搜索不包含 columnName 字符串。
- **结果:** 列对比或搜索失败。
- **措施:** 为 findColumn()、validateGetInvocation() 和 validateUpdInvocation() 方法提供有效的 columnName 字符串。

8.3.16 29011 07009

29011 07009 Invalid column index or descriptor index

- **原因:** 发出 ResultSet 方法, 该方法的列参数超出有效范围。
- **结果:** 未返回 ResultSet 方法数据。
- **措施:** 确保验证提供给方法的列。

8.3.17 29012 07006

29012 07006 Restricted data type attribute violation.

- **原因:** 数据类型无效或数据类型与 SQL 列类型不匹配时执行了方法。
- **结果:** 不执行接口方法。
- **措施:** 确保列类型使用正确的方法和 Java 数据类型。

8.3.18 29013 HY024

29013 HY024 Fetch size is less than 0.

- **原因:** 设置获取 ResultSet.setFetchSize 行的值小于零。
- **结果:** 当 ResultSet 对象需要更多行时, 未设置从数据库获取的行数。
- **措施:** 将 setFetchSize() 方法行参数值设置为大于零。

8.3.19 29015 HY024

29015 HY024 Invalid fetch direction

- **原因:** 将 `setFetchDirection()` 方法方向参数设置成了无效值。
- **结果:** 未设置 `ResultSet` 对象中行的处理方向。
- **措施:** 有效获取方向是 `ResultSet.FETCH_FORWARD`,
`ResultSet.FETCH_REVERSE` 和 `ResultSet.FETCH_UNKNOWN`。

8.3.20 29017 HY004

29017 HY004 SQL data type not supported

- **原因:** 使用 `BINARY`、`VARBINARY` 或 `LONGVARBINARY` 数据类型调用了不支持的 `getBytes()` 或 `setBytes()` JDBC 方法。
- **结果:** 不支持 `BINARY`、`VARBINARY` 和 `LONGVARBINARY` 数据类型。
- **措施:** 无需执行更改措施。

8.3.21 29018 22018

29018 2018 Invalid character value in cast specification

- **原因:** 将字符串转换成数字类型，但格式错误。
- **结果:** 通过 `getter` 方法获取的字符串无法转换为方法类型。
- **措施:** 验证数据库中的字符串，确保它能兼容。

8.3.22 29019 07002

29019 07002 Parameter {0, number, integer} for {1, number, integer} set of parameters is not set.

- **原因:** 输入的描述符包含没有值集的参数。
- **结果:** `checkIfAllParamsSet()` 方法报告未设置的参数。
- **措施:** 为列出的参数设置相应的值。

8.3.23 29020 07009

29020 07009 Invalid parameter index.

- **原因:** getter 或 setter 方法参数计数索引超出有效的输入描述符范围, 或输入描述符范围为 null。
- **结果:** getter 和 setter 方法调用验证失败。
- **措施:** 将 getter 或 setter 参数索引更改为有效的参数值。

8.3.24 29021 HY004

29021 HY004 Object type not supported

- **原因:** 调用 `PreparedStatement.setObject()` 方法包含不支持的对象类型。
- **结果:** `setObject()` 方法没有为指定的参数设置值。
- **措施:** 无需执行更改措施。有效的对象类型包括 `null`、`BigDecimal`、`Date`、`Time`、`Timestamp`、`Double`、`Float`、`Long`、`Short`、`Byte`、`Boolean`、`String` 以及 `byte []`、`Blob` 和 `Clob`。

8.3.25 29022 HY010

29022 HY010 Function sequence error.

- **原因:** `PreparedStatement.execute()` 方法不支持使用 `PreparedStatement.addBatch()` 方法。
- **结果:** 报告异常; 操作未完成。
- **措施:** 使用 `PreparedStatement.executeBatch()` 方法。

8.3.26 29026 HY000

29026 HY000 Transaction can't be committed or rolled back when AutoCommitmode is on.

- **原因:** 启用 AutoCommit 模式时, 提交了事务。

- **结果:** 事务未提交。
- **措施:** 禁用 AutoCommit 模式, 仅在此时使用该方法。

8.3.27 29027 HY011

29027 HY011 SetAutoCommit not possible, since a transaction is active.

- **原因:** 事务活跃时, 调用了 setAutoCommit() 模式。
- **结果:** 未修改当前 AutoCommit 模式。
- **措施:** 事务完成后, 设置 AutoCommit 模式。

8.3.28 29029 HY011

29029 HY011 SetTransactionIsolation not possible, since a transaction is active.

- **原因:** 事务活跃时, 设置了事务隔离级别。
- **结果:** 更改该连接对象的事务隔离级别失败。
- **措施:** 事务完成后, 设置事务隔离级别。

8.3.29 29031 HY000

29031 HY000 SQL SELECT statement in batch is illegal

- **原因:** 在 executeBatch() 方法中使用了 SELECT SQL 语句。
- **结果:** 报告异常; SELECT SQL 查询无法在批量查询中使用。
- **措施:** 使用 executeQuery() 方法执行 SELECT SQL 语句。

8.3.30 29032 23000

29032 23000 Row has been modified since it is last read.

- **原因:** 游标位于插入行时, 更新或删除 ResultSet 对象行。
- **结果:** 修改 ResultSet 行失败。
- **措施:** 更新或删除行之前, 将 ResultSet 对象游标从该行移除。

8.3.31 29033 23000

29033 23000 Primary key column value can't be updated.

- **原因:** 更新表中的主键列。
- **结果:** 该列未更新。
- **措施:** 即使忽略了列定义中的 NOT NULL 字句, 主键定义中的列也无法更新, 且不能包含 Null 值。

8.3.32 29035 HY000

29035 HY000IO Exception occurred {0}

message_text

- **原因:** ASCII、二进制或字符流的 setter 或 updater 方法导致了 `java.io.IOException`。
- **结果:** 指定 setter 或 updater 方法未修改 ASCII、二进制或字符流。
- **措施:** 无需执行更改措施。

8.3.33 29036 HY000

29036 HY000 Unsupported encoding {0}

- **原因:** 不支持字符编码。
- **结果:** 不支持请求的字符编码时, 抛出异常。
- **措施:** 仅支持 ASCII (ISO88591)、KANJI、KSC5601 和 UCS2 字符编码。

8.3.34 29037 HY106

29037 HY106 ResultSet type is TYPE_FORWARD_ONLY.

- **原因:** 对象类型设置为 `TYPE_FORWARD_ONLY` 时, 将 `ResultSet` 游标指

向上一行。

- **结果:** 未对 `ResultSet` 对象游标执行操作。
- **措施:** `TYPE_FORWARD_ONLY` `ResultSet` 对象类型游标只能向前移动。可滚动 `TYPE_SCROLL_SENSITIVE` 和 `TYPE_SCROLL_INSENSITIVE` 类型游标。

8.3.35 29038 HY107

29038 HY107 Row number is not valid.

- **原因:** 当行号设置为 0 时, 调用了 `ResultSet absolute()` 方法。
- **结果:** 游标未移动到指定行号。
- **措施:** 提供正行号 (指定从结果集开头计数的行号), 或提供负行号 (指定从结果集末尾计数的行号)。

8.3.36 29039 HY092

29039 HY092 Concurrency mode of the `ResultSet` is `CONCUR_READ_ONLY`.

- **原因:** 并发性设置为 `CONCUR_READ_ONLY`, 无法更新 `ResultSet` 对象。
- **结果:** 未修改 `ResultSet` 对象。
- **措施:** 更新时必须将 `ResultSet` 对象的并发性设置为 `CONCUR_UPDATABLE`。

8.3.37 29040 HY000

29040 HY000 Operation invalid. Current row is the insert row.

- **原因:** 获取当前插入行的更新、删除或插入信息。
- **结果:** `ResultSet` 行信息获取失败。
- **措施:** 获取行信息时, 将 `ResultSet` 对象游标从插入行移除。

8.3.38 29041 HY000

29041 HY000 Operation invalid. No primary key for the table.

- **原因:** 对未定义主键列的表执行 `getKeyColumns()` 方法失败。
- **结果:** 该表未返回任何主键数据。
- **措施:** 更改该表, 使它包括主键列。

8.3.39 29042 HY000

29042 HY000 Fetch size value is not valid.

- **原因:** 设置获取行大小的值小于 0。
- **结果:** 需要更多行时, 未设置从数据库获取的行数。
- **措施:** 为 `setFetchSize()` 方法提供大于或等于 0 的有效行值。

8.3.40 29043 HY000

29043 HY000 Max rows value is not valid.

- **原因:** 将任何 `ResultSet` 对象能包含的最大行数值设置为小于 0。
- **结果:** 未设置最大行数限制。
- **措施:** 为 `setMaxRows()` 方法使用大于或等于 0 的有效值。

8.3.41 29044 HY000

29044 HY000 Query timeout value is not valid.

- **原因:** 将驱动程序等待一个 `Statement` 对象执行的秒数设置为小于 0。
- **结果:** 未设置查询超时限制。
- **措施:** 为 `setQueryTimeout()` 方法提供大于或等于 0 的有效值。

8.3.42 29045 01S07

29045 01S07 Fractional truncation.

- **原因:** `ResultSet` getter 方法获取的数据被截尾。
- **结果:** 获取的数据被截尾。
- **措施:** 确保将要获取的数据是有效的数据类型。

8.3.43 29046 22003

29046 22003 Numeric value out of range.

- **原因:** ResultSet getter 方法获取的数值是无效的数据类型。
- **结果:** ResultSet getter 方法未获取相应数据。
- **措施:** 确保将要获取的数据是有效的数据类型。

8.3.44 29047 HY000

29047 HY000 Batch update failed. See next exception for details.

- **原因:** 无法成功执行某个批量更新命令。
- **结果:** 部分批量更新命令失败。
- **措施:** 查看后续异常和异常信息。

8.3.45 29048 HY009

29048 HY009 Invalid use of null.

- **原因:** 将包含预期表名称的参数设置为 Null。
- **结果:** DatabaseMetadata 方法未报告任何结果。
- **措施:** 为 DatabaseMetaData 方法提供非 Null 值的有效表名称。

8.3.46 29049 25000

29049 25000 Invalid transaction state.

- **原因:** 处理事务时, 调用了 begintransaction() 方法。
- **结果:** 未启动新事务。
- **措施:** 调用 begintransaction() 方法之前, 验证是否已启动其他事务。

8.3.47 29050 HY107

29050 HY107 Row value out of range.

- **原因:** 调用 getCurrentRow 超出第一行和最后一行的范围。

- **结果:** 未获取当前行。
- **措施:** 无需执行更改措施。请联系 user@trafodion.incubator.apache.org 并报告整个信息。

8.3.48 29051 01S02

29051 01S02 ResultSet type changed to
TYPE_SCROLL_INSENSITIVE.

- **原因:** 结果集类型已更改。
- **结果:** 无。
- **措施:** 该消息为 SQL 警告。无需执行更改措施。

8.3.49 29053 HY000

29053 HY000 SQL SELECT statement is invalid in
executeUpdate() methodCause.

- **原因:** 在 executeUpdate() 方法中使用了 SELECT SQL 语句。
- **结果:** 报告异常: 未执行 SQL 查询。
- **措施:** 使用 executeQuery() 方法发出 SELECT SQL 语句。

8.3.50 29054 HY000

29054 HY000 Only SQL SELECT statements are valid in
executeQuery() method.

- **原因:** 在 executeQuery() 方法中使用了 non-select SQL 语句。
- **结果:** 报告异常: 未执行 SQL 查询。
- **措施:** 使用 executeUpdate() 方法发出 non-select SQL 语句。

8.3.51 29056 HY000

29056 HY000 Statement is already closed.

- **原因:** 对已关闭语句使用 `validateSetInvocation()` 或 `validateExecuteInvocation` 方法。
- **结果:** 验证语句失败, 返回异常报告。
- **措施:** 在关闭语句之前使用 `validateSetInvocation()` 或 `validateExecuteInvocation` 方法。

8.3.52 29057 HY000

29057 HY000 Auto generated keys not supported.

- **原因:** 使用自动生成键功能。
- **结果:** 操作失败。
- **措施:** 不支持自动生成键功能。

8.3.53 29058 HY000

29058 HY000 Connection is not associated with a PooledConnection object.

- **原因:** 创建 PooledConnection 对象前调用了 `getPooledConnection()` 方法。
- **结果:** 无法获取池中的连接。
- **措施:** 创建 PooledConnection 对象后, 再使用 `getPooledConnection()` 方法。

8.3.54 29059 HY000

29059 HY000 'blobTableName' property is not set or set to null value or set to invalid Value.

- **原因:** 在未设置 `t4jdbc.blobTableName` 属性或该属性为无效值时访问了 BLOB 列。
- **结果:** 应用程序无法访问 BLOB 列。
- **措施:** 将 `t4jdbc.blobTableName` 属性设置为有效的 LOB 表名称。

LOB 表名称的格式为 `catalog.schema.lobTableName`。

8.3.55 29060 HY000

29060 HY000 't4jdbc.clobTableName' property is not set or set to null value or set to invalid value.

- **原因:** 在未设置 `t4jdbc.clobTableName` 属性或该属性为 Null 或无效值时，访问了 CLOB 列。
- **结果:** 应用程序无法访问 CLOB 列。
- **措施:** 将 `t4jdbc.clobTableName` 属性设置为有效的 LOB 表名称。

LOB 表名称的格式为 `catalog.schema.lobTableName`。

8.3.56 29061 HY00

29061 HY00 Lob object {0} is not current.

- **原因:** 在移动游标或关闭获取 LOB 数据的结果集之后，访问 LOB 列数据。
- **结果:** 应用程序无法访问 LOB 数据。
- **措施:** 在移动游标或关闭结果集对象之前，访问 LOB 数据。

8.3.57 29063 HY00

29063 HY00 Transaction error {0} - {1} while obtaining start data locator.

- **原因:** 插入或更新 LOB 列时，Type 4 驱动程序保留给定进程的数据定位器

时发生事务错误。

- **结果:** 应用程序无法插入或更新 LOB 列。
- **措施:** 检查消息中的文件系统错误, 采取相应措施。

8.3.58 29067 07009

2067 07009 Invalid input value in the method {0}.

- **原因:** 给定方法中的一个或多个输入值无效。
- **结果:** 给定输入方法失败。
- **措施:** 检查给定方法的输入值。

8.3.59 29068 07009

29068 07009 The value for position can be any value between 1 and one more than the length of the LOB data.

- **原因:** Blob.setBinaryStream、Clob.setCharacterStream 或 Clob.setAsciiStream 中的位置输入值可能在 1 与大于 LOB 数据长度的数值之间。
- **结果:** 应用程序无法在指定位置写入 LOB 数据。
- **措施:** 更改位置输入值。

8.3.60 29069 HY000

29069 HY000 Autocommit is on and LOB objects are involved.

- **原因:** 启用 AutoCommit 模式时访问 LOB 列。
- **结果:** 应用程序无法访问 LOB 列。

- **措施:** 禁用 AutoCommit 模式。

8.3.61 29100 HY000

29100 HY000 An internal error occurred.

- **原因:** 内部错误。
- **结果:** 操作失败。
- **措施:** 无。请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.62 29101 HY000

29101 HY000 Contact your service provider.

- **原因:** 内部错误。
- **结果:** 操作失败。
- **措施:** 无。请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.63 29102 HY000

29101 HY000 Error while parsing address <address>.

- **原因:** 无法识别地址格式。
- **结果:** 操作失败。
- **措施:** 请参阅 [url 属性](#), 获取有效的地址格式。

8.3.64 29103 HY000

29103 HY000 Address is null.

- **原因:** 地址为空。
- **结果:** 操作失败。
- **措施:** 请参阅 [url 属性](#), 获取有效的地址格式。

8.3.65 29104 HY000

29104 HY000 Expected suffix: <suffix>.

- **原因:** 地址后缀不正确或缺失。
- **结果:** 操作失败。
- **措施:** 请参阅 [url 属性](#), 获取有效的地址格式。

8.3.66 29105 HY000

29105 HY000 Unknown prefix for address.

- **原因:** 地址前缀不正确或缺失。
- **结果:** 操作失败。
- **措施:** 请参阅 [url 属性](#), 获取有效的地址格式。

8.3.67 29106 HY000

29106 HY000 Expected address format:

jdbc:subprotocol::subname.

- **原因:** 不适用。
- **结果:** 不适用。
- **措施:** 请参阅 [url 属性](#), 获取有效的地址格式。

8.3.68 29107 HY000

29107 HY000 Address not long enough to be a valid address.

- **原因:** 地址长度太短, 无法成为有效地址。
- **结果:** 操作失败。
- **措施:** 请参阅 [url 属性](#), 获取有效的地址格式。

8.3.69 29108 HY000

29108 HY000 Expecting \\<machine-name><process-name>/<port-number>.

- **原因:** DCS 地址格式无效。

- **结果:** 操作失败。
- **措施:** 请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.70 29109 HY000

29109 HY000 //{<IP Address|Machine Name>[:port]/database name>

- **原因:** 参考消息。
- **结果:** 不适用。
- **措施:** 不适用。

8.3.71 29110 HY000

29110 HY000 Address is missing an IP address or machine name.

- **原因:** 缺失必须的 IP 地址或机器名。
- **结果:** 操作失败。
- **措施:** 包括有效的 IP 地址或机器名。请参阅 [url 属性](#), 获取有效的地址格式。

8.3.72 29111 HY000

29111 HY000 Unable to evaluate address <address> Cause: <cause>.

- **原因:** 驱动程序无法确定主机的 IP 地址。
- **结果:** 操作失败。
- **措施:** 可能未正确指定地址或主机名称, 或存在安全限制。

请参阅 `java.net.InetAddress` 类中 `getAllByName` 方法的文档。指定有效的 IP 地址或机器名。

请参阅 [url 属性](#), 获取有效的地址格式。

8.3.73 29112 HY000

29112 HY000 Missing ']'.

- **原因：**驱动程序无法确定主机的 IP 地址。
- **结果：**操作失败。
- **措施：**地址或机器名的格式错误。

请参阅 [url 属性](#)，获取有效的地址格式。

8.3.74 29113 HY000

29113 HY000 Error while opening socket. Cause: <cause>.

- **原因：**套接字错误。
- **结果：**操作失败。
- **措施：**对 Exception 使用 `getCause` 方法，确定合适的措施。

8.3.75 29114 HY000

29114 HY000 Error while writing to socket.

- **原因：**套接字写入错误。
- **结果：**操作失败。
- **措施：**对 Exception 使用 `getCause` 方法，确定合适的措施。

8.3.76 29115 HY000

29115 HY000 Error while reading from socket. Cause:
<cause>.

- **原因：**套接字读取错误。
- **结果：**操作失败。
- **措施：**对 Exception 使用 `getCause` 方法，确定合适的措施。

8.3.77 29116 HY000

29116 HY000 Socket is closed.

- **原因:** 套接字关闭错误。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.78 29117 HY000

29117 HY000 Error while closing session. Cause: <cause>.

- **原因:** 关闭会话时遇到错误。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.79 29118 HY000

29118 HY000 A write to a bad map pointer occurred.

- **原因:** 内部错误。
- **结果:** 操作失败。
- **措施:** 无。请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.80 29119 HY000

29119 HY000 A write to a bad par pointer occurred.

- **原因:** 内部错误。
- **结果:** 操作失败。
- **措施:** 无。请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.81 29120 HY000

29120 HY000 An association server connect message error occurred.

- **原因:** 无法连接到 DCS 关联服务器。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.82 29121 HY000

29121 HY000 A close message error occurred. Cause: <cause>.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.83 29122 HY000

29122 HY000 An end transaction message error occurred.

- **原因:** 无法执行操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.84 29123 HY000

29123 HY000 An execute call message error occurred.
Cause: <cause>.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.85 29124 HY000

29124 HY000 An execute direct message error occurred.

Cause: <cause>.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.86 29125 HY000

29125 HY000 An execute direct rowset message error occurred.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.87 29126 HY000

29126 HY000 An execute N message error occurred. Cause: <cause>.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.88 29127 HY000

29127 HY000 An execute rowset message error occurred.

Cause: <cause>.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.89 29128 HY000

29128 HY000 A fetch perf message error occurred. Cause:

<cause>.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.90 29129 HY000

29129 HY000 A fetch rowset message error occurred. Cause:

<cause>.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.91 29130 HY000

29130 HY000 A get sql catalogs message error occurred.

Cause: <cause>.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.92 29131 HY000

29131 HY000 An initialize dialogue message error occurred. Cause: <cause>.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.93 29132 HY000

29132 HY000 A prepare message error occurred. Cause: <cause>.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.94 29133 HY000

29133 HY000 A prepare rowset message error occurred.

Cause: <cause>.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 getCause() 方法的评估值, 采取合适的恢复方法。

8.3.95 29134 HY000

29134 HY000 A set connection option message error

occurred. Cause: <cause>.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 getCause() 方法的评估值, 采取合适的恢复方法。

8.3.96 29135 HY000

29135 HY000 A terminate dialogue message error occurred.

Cause: <cause>.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 getCause() 方法的评估值, 采取合适的恢复方法。

8.3.97 29136 HY000

29136 HY000 An association server connect reply occurred.

Exception: <exception> Exception detail:

<exception_detail>

Error text/code: <error text or code>.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 检查该消息附带的所有错误或错误详细信息, 联系 user@trafodion.incubator.apache.org 检查服务器错误日志, 分析附带的错误和警告。

8.3.98 29137 HY000

29137 HY000 A close reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 getCause() 方法的评估值, 采取合适的恢复方法。

8.3.99 29138 HY000

29138 HY000 An end transaction reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 getCause() 方法的评估值, 采取合适的恢复方法。

8.3.100 29139 HY000

29139 HY000 An execute call reply error occurred.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.101 29140 HY000

29140 HY000 An execute direct reply error occurred.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.102 29141 HY000

29141 HY000 An execute direct rowset reply error occurred.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.103 29142 HY000

29142 HY000 An execute N reply error occurred.

- **原因：**无法执行该操作。
- **结果：**操作失败。
- **措施：**根据 Exception 中使用 `getCause()` 方法的评估值，采取合适的恢复方法。

8.3.104 29143 HY000

29143 HY000 An execute rowset reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.105 29144 HY000

29144 HY000 A fetch perf reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.106 29145 HY000

29145 HY000 A fetch rowset reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.107 29146 HY000

29146 HY000 A get sql catalogs reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.108 29147 HY000

29147 HY000 An initialize dialogue reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.109 29148 HY000

29148 HY000 A prepare reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.110 29149 HY000

29149 HY000 A prepare rowset reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.111 29150 HY000

29150 HY000 A set connection option reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.112 29151 HY000

29151 HY000 A terminate dialogue reply error occurred.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.113 29152 HY000

29152 HY000 No more ports available to start ODBC servers.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 请联系 user@trafodion.incubator.apache.org 检查服务器错误日志。根据 Exception 中使用 `getCause()` 方法的评估值, 采取合适的恢复方法。

8.3.114 29153 HY000

29153 HY000 Invalid authorization specification.

- **原因:** 错误的用户名和/或密码。
- **结果:** 操作失败。
- **措施:** 使用正确的用户名和/或密码重试。

8.3.115 29154 HY000

29154 HY000 Timeout expired.

- **原因:** 无法执行该操作。
- **结果:** 操作失败。
- **措施:** 重试和/或更改操作的超时值。

8.3.116 29155 HY000

29155 HY000 Unknown message type.

- **原因:** 内部错误。
- **结果:** 操作失败。
- **措施:** 请联系 user@trafodion.incubator.apache.org 检查服务器错误日志，分析附带的错误和警告。

8.3.117 29156 HY000

29156 HY000 An error was returned from the server. Error:

<error>

Error detail: <error_detail>.

- **原因:** 服务器报告错误。
- **结果:** 操作失败。
- **措施:** 检查该消息附带的所有错误或错误详细信息，请联系 user@trafodion.incubator.apache.org 检查服务器错误日志，分析附带的错误和警告。

8.3.118 29157 HY000

29157 HY000 There was a problem reading from the server.

- **原因:** 服务器报告错误。
- **结果:** 操作失败。
- **措施:** 检查该消息附带的所有错误或错误详细信息，联系 user@trafodion.incubator.apache.org 检查服务器错误日志，分析附带的错误和警告。

8.3.119 29158 HY000

29158 HY000 The message header contained the wrong version.

Expected: <expected_version> Actual: <actual_version>.

- **原因:** 服务器版本与预期版本不同。
- **结果:** 操作失败。
- **措施:** 检查该消息附带的所有错误或错误详细信息。安装合适版本的驱动程序和 EsgynDB 连接服务。

8.3.120 29159 HY000

29159 HY000 The message header contained the wrong signature.

Expected: <expected_signature> Actual:
<actual_signature>.

- **原因:** 服务器签名与预期版本的不同。
- **结果:** 操作失败。
- **措施:** 检查该消息附带的所有错误或错误详细信息。安装合适版本的驱动程序和 EsgynDB 连接服务。

8.3.121 29160 HY000

29160 HY000 The message header was not long enough.

- **原因:** 服务器返回的消息过短, 无法作为有效信息。
- **结果:** 操作失败。
- **措施:** 无。请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.122 29161 S1000

29161 S1000 Unable to authenticate the user because of an NT error: {0}

- **原因:** 服务器返回的信息。
- **结果:** 操作失败。
- **措施:** 无。请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.123 29162 S1000

29162 S1000 Unexpected programming exception has been found: <exception>.

更多信息，请检查节点的服务器事件日志。

- **原因：**服务器返回的消息。
- **结果：**操作失败。
- **措施：**请联系 user@trafodion.incubator.apache.org 检查服务器错误日志，分析附带的错误和警告。

8.3.124 29163 08001

29163 08001 ODBC Services not yet available: <server>.

- **原因：**服务器返回的消息。
- **结果：**操作失败。
- **措施：**重试和/或等待服务器变为可用。使用更多服务器配置服务器端数据源。

8.3.125 29164 08001

29164 08001 DataSource not yet available or not found: <error>.

- **原因：**服务器返回的消息。
- **结果：**操作失败。
- **措施：**创建服务器数据源和/或使用更多服务器配置服务器数据源。

8.3.126 29165 HY000

29165 HY000 Unknown connect reply error: <error>.

- **原因:** 服务器返回的信息。
- **结果:** 操作失败。
- **措施:** 请联系 user@trafodion.incubator.apache.org 检查服务器错误日志, 分析附带的错误和警告。

8.3.127 29166 HY000

29166 HY000 This method is not implemented.

- **原因:** 内部错误。
- **结果:** 操作失败。
- **措施:** 无。请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.128 29167 HY000

29167 HY000 Internal error. An internal index failed consistency check.

- **原因:** 内部错误。
- **结果:** 操作失败。
- **措施:** 无。请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.129 29168 HY000

29168 HY000 Unknown reply message error: <error> error detail: <error_detail>.

- **原因:** 服务器返回错误。
- **结果:** 操作失败。
- **措施:** 请联系 user@trafodion.incubator.apache.org 检查服务器事件日志中的错误。

8.3.130 29169 HY000

29169 HY000 Invalid connection property setting

- **原因:** 服务器返回的消息过短, 无法作为有效消息。
- **结果:** 操作失败。
- **措施:** 无。请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.131 29170 HY000

29170 HY000 Invalid parameter value.

- **原因:** 内部错误。
- **结果:** 操作失败。
- **措施:** 无。请联系 user@trafodion.incubator.apache.org, 报告错误消息。

8.3.132 29172 HY000

29172 HY000 Translation of parameter to {0} failed.

- **原因:** 将参数转化为目标字符集时发生转化错误, 其中{0}是目标字符集。
- **结果:** 该方法失败。
- **措施:** 在合适的字符集中使用字符设置该参数。如果您想关闭转换验证, 请将 `translationVerification` 属性设置为 `FALSE`。

8.3.133 29173 HY000

29173 HY000 Translation of SQL statement {0} failed.

- **原因:** 将 SQL 语句转换成目标字符集时发生转换错误, 其中{0}是目标字符集。
- **结果:** 该方法失败。
- **措施:** 为了在合适的字符集中使用字符, 请编辑 SQL 语句。如果您想关闭转换验证, 请将 `translationVerification` 属性设置为 `FALSE`。

8.3.134 29174 HY000

29174 HY000 Autocommit is on and updateRow was called on the ResultSetobject.

- **原因:** 开启 AutoCommit 时, 调用 `ResultSet.updateRow()` 方法。
- **结果:** 抛出警告。由于已关闭所有 `ResultSet` (游标), 之后调用 `ResultSet.next()` 都失败。
- **措施:** 在 AutoCommit 关闭的情况下调用 `ResultSet.updateRow()` 方法。

8.3.135 29175 HY000

29175 HY000 Unknown Error {0}.

- **原因:** 连接{0}发生未知错误。
- **结果:** 连接失败。
- **措施:** 重试连接。

8.3.136 29177 HY000

29177 HY000 Data cannot be null.

- **原因:** 以 `String` 格式获取列值数据, 但传递了 `Null` 输入值。
- **结果:** 操作失败。
- **措施:** 请联系 user@trafodion.incubator.apache.org 检查服务器事件日志中的错误。

8.3.137 29178 HY000

29178 HY000 No column value has been inserted.

- **原因:** 未指定必需列的值。
- **结果:** 操作失败。
- **措施:** 确保指定了所有必需列的值, 重试操作。

8.3.138 29182 HY000

29182 HY000 General warning. Connected to the default data source: TDM_Default_DataSource

- **原因:** 用户应用程序指定了服务器端不存在的数据源。
- **结果:** 连接使用默认数据源 TDM_Default_DataSource。
- **措施:** 忽略该警告或联系数据库管理员, 添加应用程序指定的服务器端数据源。

8.3.139 S1000 HY000

S1000 HY000 A TIP transaction error <error> has been detected. Check the server event log on Node <segment> for Transaction Error details.

- **原因:** 服务器返回了一条信息。
- **结果:** 操作失败。
- **措施:** 请联系 user@trafodion.incubator.apache.org 检查服务器事件日志中的错误。

9. 避免驱动程序-服务器版本不匹配

EsgynDB JDBC Type 4 驱动程序只能连接至相同版本的平台（服务器），不能连接至较早版本的平台。

9.1 JDBC 客户端连接至 EsgynDB 的注意事项

- 如果您已安装了 EsgynDB JDBC Type4 驱动程序 2.7，并设置了客户端环境，请将 2.7 驱动程序类加入 java CLASSPATH。
- 由于 EsgynDB JDBC Type4 驱动程序 2.7 和 2.8 不兼容，如使用 EsgynDB2.8，请务必使用 EsgynDB JDBC Type4 驱动程序 2.8。

9.2 版本不匹配错误消息

如果 EsgynDB JDBC 客户端尝试连接到无效的 DCS 版本，则驱动程序将返回错误消息：

```
SQLCODE: 29162
```

```
SQLSTATE S1000
```

```
Error text:
```

```
Unexpected programming exception has been found:
```

```
<errortext>. Check the server event log on node
```

```
<logfile_location> for details.
```

- <errortext>是错误消息的文本。
- <logfile_location>是日志文件的位置。