Oracle® JDBC for Rdb Release Notes

Release 7.4.1.1

March 2026

Oracle JDBC for Rdb Release Notes

Release 7.4.1.1.

Copyright © 2005, 2026 Oracle and/or its affiliates. All rights reserved.
Oracle Corporation - Worldwide Headquarters, 2300 Oracle Way, Austin, TX 78741, United States.

Primary Author: Rdb Engineering and Documentation group.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited. The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing. If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government’s use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle, Java, Oracle Rdb, Hot Standby, LogMiner for Rdb, Oracle SQL/Services, Oracle CODASYL DBMS, Oracle RMU, Oracle CDD/Repository, Oracle Trace, and Rdb7 are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Contents

Preface x

Chapter 1 Installation and Documentation 1-1
1.1 Accessing the Documentation 1-1
1.2 System and Software Requirements 1-2
1.2.1 UNZIP is required 1-3
1.3 Installation 1-4
1.3.1 SQL/Services JDBC Server Startup Command Procedures 1-4
1.3.2 Contents of the Oracle JDBC for Rdb Kit 1-5
1.3.3 Installation Procedure 1-9
1.3.4 Rdb Thin Driver Software on Windows 1-17
1.3.5 Rdb Thin Driver Software on Other Operating Systems 1-17
Chapter 2 Enhancements Provided in Oracle JDBC for Rdb Release 7.4.1.1 2-19
2.1 Support for OpenJDK8 2-19
Chapter 3 Problems Corrected 3-20
3.1 ACCVIO in MPServer with Many Open Queries using JDK8 3-20
Chapter 4 Known Problems and Workarounds 4-22
4.1 Thin Server Deadlocks 4-22
4.2 Using Java Fast VM on OpenVMS ALPHA 4-22
4.3 Using the Oracle SQL/Services Management GUI and JDBC Dispatchers 4-23
4.4 Blob Columns and Correlation Names 4-23
4.5 Blob Columns and Update Statements 4-24
4.6 External Procedures and Thin Server 4-25
4.7 Limitations 4-25
4.7.1 Unsupported Methods 4-25
4.7.2 Auto-generated Keys 4-27
4.7.3 String Truncation Warnings 4-27
4.7.4 Numeric and String Functions in JDBC 4-27
Chapter 5 New Features and Corrections in Previous Releases 5-28
5.1 New Features for Release 7.4.1 5-28
5.1.1 General Quality and Performance Improvements 5-28
5.2 Corrections in Release 7.4.1 5-28
5.3 New Features for Release 7.3.5.3 5-28
5.4 Corrections in Release 7.3.5.3 5-28
5.4.1 Extraneous Text on Polling Message 5-29
5.4.2 SilentTruncate Causes Incorrect Update of Columns 5-29
5.4.3 DatabaseMetaData.getColumnPrivileges Problem 5-30
5.4.4 ResultSetMetaData.getColumnDisplaySize Problem 5-30
5.4.5 Connection Option @concurrentThread Problem 5-31
5.4.6 TIMESTAMPADD Function with Milliseconds 5-31
5.4.7 TIMESTAMPDIFF Behaviour is Now Settable 5-31
5.5 New Features for Release 7.3.5.2 5-32

5.5.1 New Connection Options 5-32
5.5.2 Changed Log File Names 5-32
5.5.3 Server Log File Unlocked 5-33
5.6 Corrections in Release 7.3.5.2 5-33
5.6.1 ACCVIO During MP Server Shutdown 5-33
5.6.2 SQL-F-CONVARUND Raised on Multi-context Queries 5-34
5.6.3 Timestamps and Batched Statements 5-35
5.6.4 Incorrect Termination Message on Server Shutdown 5-36
5.6.5 Manager Server Fails to Respond to Poll Request 5-36
5.6.6 SilentTruncate Not Enabled for Some Operations 5-36
5.7 New Features for Release 7.3.5.1 5-37
5.7.1 New Pool Server Balancing Option PoolOrder 5-37
5.7.2 New Connection Options 5-38
5.8 Corrections in Release 7.3.5.1 5-38
5.8.1 Incorrect Date/time Values When TimeZones Mismatch 5-38
5.8.2 New Feature Omission - TimeZone Connection Attributes 5-39
5.8.3 Incorrect Parameter Initialization in RDBJDBC_EXECCLI.COM 5-39
5.8.4 Spurious Error Message with Stop Client 5-41
5.8.5 ORCM Async External Tool Command Failure 5-41
5.8.6 Incorrect Seconds When Date Prior to Java Epoch 5-42
5.8.7 Multithread Problem with DatabaseMetaData 5-42
5.8.8 NLSLANG Problems 5-42
5.8.9 RDBJDBC_STARTSRV.COM Problem 5-43
5.8.10 More Incorrect Seconds When Date Prior to Java Epoch 5-44
5.8.11 SQL92 Dialect Semantics 5-44
5.8.12 Unreferenced CALL Parameters 5-46
5.8.13 ACCVIO When Using CREATE or DROP DATABASE 5-47
5.8.14 IF EXISTS Trimmed Off DROP Statements 5-47
5.8.15 NOSUCHCUR Error When Using ORDP 5-47
5.8.16 Concurrent Threads in Single Connection 5-47
5.9 New Features for Release 7.3.5 5-48
5.9.1 Java 8.0 Support on OpenVMS IA-64 Systems 5-48
5.9.2 Versioned JDBC Driver Jars 5-49
5.9.3 Change in SQL/Services JDBC Dispatcher Naming 5-50
5.9.4 Changes to SHIFT_JIS Java Encoding 5-51
5.10 Corrections in Release 7.3.5 5-51
5.10.1 Small Memory Leak With GetTables() 5-51
5.10.2 DEFAULTSSL Definition Not Inherited 5-51
5.10.3 Incorrect Value for Client and Executor Free Shared Memory 5-52
5.11 New Features for Release 7.3.4.2 5-53
5.11.1 Support for Oracle Rdb Second Password Option 5-53
5.11.2 New srv.usecreprc Configuration Option 5-54
5.11.3 New simplepn Connection Option 5-54
5.11.4 TimeZone Connection Attributes 5-55
5.12 Corrections in Release 7.3.4.2 5-55
5.12.1 Incorrect Number Of Clients in Show Server Pool 5-56

5.12.2 Server Incompatibility With Early Driver Versions 5-56
5.13 New Features for Release 7.3.4.1 5-56
5.13.1 New Transaction Option AUTOFETCH 5-56
5.14 Corrections in Release 7.3.4.1 5-57
5.14.1 Pool Servers Not Working 5-58
5.14.2 Problem with MultiProcess Connection Option 5-58
5.14.3 Bumpy-case Connection String Options Used with Properties. 5-58
5.14.4 Further Blob Problems with Delimited Identifiers 5-59
5.14.5 Access Violation at RDBJDBCMPSHR73+00227E31 5-60
5.15 New Features for Release 7.3.4 5-61
5.15.1 Create / Drop Database Support 5-61
5.16 Corrections in Release 7.3.4 5-61
5.16.1 Like Escape Throws OutOfBounds Exception 5-61
5.16.2 Correlation Name Workaround for Blobs Fails 5-62
5.16.3 PreparedStatement Cache Failure 5-62
5.16.4 Blob Problem with Delimited Identifiers 5-63
5.17 New Features for Release 7.3.3.2 5-63
5.17.1 Change Transaction Start Time. 5-63
5.18 Corrections in Release 7.3.3.2 5-63
5.18.1 NullPointerException When Allowing Anonymous 5-64
5.18.2 UnsupportedEncodingException: Binary 5-64
5.18.3 Alignment Faults When Using MP Server 5-64
5.18.4 ResultSet.absolute() Problems 5-65
5.18.5 Orphaned Executor Processes When Using Persona 5-65
5.19 New Features for Release 7.3.3.1 5-66
5.19.1 Change in Template Configuration Filename 5-66
5.20 Corrections in Release 7.3.3.1 5-67
5.20.1 DatabaseMetadata.getColumns Problem with DefaultValue 5-67
5.20.2 Show Server Configuration File Problem 5-67
5.20.3 Unknown Action Message in Log file 5-68
5.20.4 Severe Performance Degradation in MP Server 5-68
5.20.5 Controller DCL Command Line Failure 5-68
5.20.6 Server Access Restrictions Ignored 5-69
5.20.7 Datetime Insertion Issues 5-70
5.20.8 MP Server May Become CPU Bound 5-71
5.20.9 Controller Null Pointer Exception When No Default Server 5-72
5.20.10 Pool Server Tries to Restart Running Server 5-73
5.20.11 Controller Poll Reopenlogs Hangs 5-74
5.20.12 Pool Server Usage Balancing Problem 5-74
5.20.13 Configuration File Problem Regression 5-75
5.20.14 Server Configuration Deny User Not Enforced 5-75
5.21 New Features for Release 7.3.3 5-76
5.21.1 Oracle Rdb PID now Displayed 5-76
5.21.2 Controller Command Show Executors 5-76
5.21.3 Extra Timestamp Precision 5-77
5.21.4 Nlslang Connection Switch 5-77

5.21.5 Use Query Header as Description 5-78
5.22 Corrections in Release 7.3.3 5-79
5.22.1 Controller START SERVER Problems 5-79
5.22.2 Executor Initialization Problem 5-80
5.22.3 Idle Client Termination Breaks Subsequent Connection 5-81
5.22.4 Controller STOPSERVER Command Problem 5-81
5.22.5 Hang on Connection to Server When Using SSL 5-82
5.22.6 DatabaseMetadata Pattern Matches and Nulls 5-82
5.22.7 DatabaseMetadata Missing Index Information 5-83
5.22.8 Forced Client Termination May Crash Thin Server 5-83
5.22.9 Show Clients May Cause NullPointerException in Server 5-84
5.22.10 Access Violation in MP Server 5-85
5.22.11 Syntax Errors in Insert and Update Statements 5-85
5.22.12 Statement.getGeneratedKeys() Throws RdbException 5-86
5.22.13 SQLException Thrown in ResultSet.isBeforeFirst() 5-86
5.22.14 PreparedStatement executeBatch() Problems 5-87
5.23 New Features for Release 7.3.2 5-88
5.23.1 Server Options List Inheritance 5-88
5.23.2 Oracle JDBC for Rdb Manager Server 5-89
5.23.3 Restricting Server Access by IP 5-90
5.23.4 Executor Balancing 5-90
5.23.5 MinFreeExecutors 5-90
5.23.6 Executor Reuse 5-91
5.23.7 LogFile Patterns 5-91
5.23.8 New Server Configuration Option retainRdbSQLState 5-91
5.23.9 New Connection Option “APP” 5-92
5.24 Corrections in Release 7.3.2 5-92
5.24.1 Event Flag Problem with Long Executor Name Prefixes 5-92
5.24.2 Executor Process Termination Problem 5-93
5.24.3 Some Server Characteristics Not Correctly Inherited from DEFAULT 5-94
5.24.4 Memory Problem with Pool Servers and Java 1.6.0-2 5-95
5.24.5 Exception not Thrown When Record Locked During Update 5-96
5.24.6 Classpath Documentation Error 5-97
5.25 New Features for Release 7.3.1 5-98
5.25.1 “Owner” may be Used in XML Configuration Files 5-98
5.25.2 SQL Statement Restriction and Denial 5-98
5.25.3 Event Notification 5-99
5.26 Corrections in Release 7.3.1 5-99
5.26.1 Multi-process Server Connection Hang When Executor Dies 5-99
5.26.2 Multi-process Executor Process Terminates Unexpectedly 5-100
5.26.3 Pooled Server AutoRestart Problem 5-100
5.26.4 getGeneratedKeys() and OutOfBounds Exception 5-101
5.26.5 Multi-process Server Fails When Persona Used 5-102
5.26.6 RestrictAccess Being Applied Even When Disabled 5-103
5.26.7 Multi-process Problem in the Native Driver 5-103
5.27 New Features for Release 7.3.0.2 5-104

5.27.1 Server Network Keep Alive 5-104
5.28 Corrections in Release 7.3.0.2 5-104
5.28.1 Null Pointer Exception During Server Viability Check by Pool Server 5-104
5.28.2 MINUS and INTERSECT Problem 5-105
5.28.3 Memory Leak When Executor Process Fails to Run 5-105
5.28.4 Lost Executor Not Caught by Multi-process Server 5-105
5.28.5 Multi-process Native Driver Problem 5-106
5.28.6 PreparedStatement.getGeneratedKeys() Problem 5-106
5.28.7 Excessive IOs During Metadata Retrieval 5-106
5.28.8 Syntax Error SQL-F-CONVARUND 5-106
5.28.9 Multi-process Server Hang 5-107
5.28.10 SSL Socket Intrusion Problem 5-107
5.29 New Features for Release 7.3.0.1 5-108
5.29.1 Reopen Server Log Files Using Poll Subcommand 5-108
5.30 Corrections in Release 7.3.0.1 5-108
5.30.1 Show Clients Not Showing Column Names 5-109
5.30.2 Batched Statement Fails With MULTIPLE_RECORDS Exception 5-110
5.30.3 Release 7.3 Thin Driver and Prior Servers Incompatibility 5-110
5.30.4 Global Memory Leak When Executors are Run-down 5-111
5.30.5 Autorestart on Pooled Servers Not Working 5-111
5.30.6 setFetchSize() Hint Ignored by PreparedStatement 5-112
5.30.7 Exception on Subsequent Execution of PreparedStatement 5-112
5.30.8 EOFException on READ_ROW With Nested Statements 5-113
5.30.9 DatabaseMetaData.getUDTs() 5-114
5.30.10 Using Multiprocess With the JDBC Native Driver 5-114
5.30.11 Prepared Statement Not Closing Underlying Cursor 5-115
5.30.12 Release Statement Synchronization Problem 5-116
5.30.13 Unitialized SQLCA Block in MP Server 5-116
5.30.14 Controller SHOW CLIENTS and MP Server Problem 5-117
5.30.15 Documentation Error – Record Streaming 5-117
5.30.16 ResultSet.updateRow() and ResultSet.deleteRow() Problem 5-118
5.30.17 Problem Using Column Renaming With Dbkey 5-118
5.30.18 Cast Error on Scaled Integer After ResultSet.insertRow() 5-119
5.30.19 ResultSet.deleteRow() Behaviour Change 5-120
5.30.20 Blob Handles Not Released When Using ResultSet.getBlob() 5-121
5.30.21 Sequence Values Not Visible When Using Column Name 5-122
5.30.22 Scaled Integer Problem 5-123
5.30.23 Server Matching Exception May Stop Poll Handling 5-124
5.31 New Features for Release 7.3 5-125
5.31.1 Shutdown Thread 5-125
5.31.2 Driver.attach() 5-125
5.31.3 Returning List of Known Databases 5-125
5.31.4 Controller Enhancements 5-125
5.31.5 RDB_EXT.JAR file 5-126
5.31.6 Performance Enhancements 5-127
5.32 Corrections in Release 7.3 5-129

5.32.1 Server Startup Failure When Using CFG File 5-129
5.32.2 AccessViolation on Disconnect When Inserting Blobs 5-129
5.32.3 Parameter Marker Problem Now Resolved 5-130
5.32.4 Controller SHOW CLIENTS and MP Server Problem 5-131
5.32.5 Possible Memory Leak When Updating Blob Columns 5-131
5.32.6 Access Violation With Trace and Network Dump 5-132
5.32.7 Named Input Parameters not Working With CallableStatements 5-132
5.33 New Features for Release 7.2.5.5 5-133
5.34 Corrections in Release 7.2.5.5 5-134
5.34.1 Long-running Query Holds up New Connections in MP Server 5-134
5.35 New Features for Release 7.2.5.4 5-134
5.36 Corrections in Release 7.2.5.4 5-135
5.36.1 Access Violation at Java_rdb_JNI_SetStrVal 5-135
5.36.2 DCL Command Line Too Long 5-135
5.36.3 Access Violation When Database Specification is Missing 5-136
5.36.4 Unaligned Memory Faults on OpenVMS IA-64 5-137
5.36.5 Read-Only Transactions Not Enforced on Connection Switch 5-137
5.36.6 Sockets Not Correctly Closing on OpenVMS Clients 5-138
5.36.7 Cast Problem When Converting String to Date/Time 5-138
5.37 New Features for Release 7.2.5.3 5-139
5.38 Corrections in Release 7.2.5.3 5-140
5.38.1 Blob DatabaseMetaData Access May Crash Thin Server 5-140
5.38.2 BigDecimal Scaling Incorrect in PreparedStatement SetObject() 5-140
5.38.3 Connection.nativeSQL() Method Throws Null Pointer Exception 5-141
5.38.4 Calling Resultset.isLast() Method May Change Transaction Behavior 5-141
5.39 New Features for Release 7.2.5.2 5-141
5.39.1 DEC_KANJI and DEC_HANZI Support Enabled 5-142
5.40 Corrections in Release 7.2.5.2 5-142
5.40.1 ResultSet.getBigDecimal() Not Working With System ResultSets 5-142
5.40.2 Setting TraceLevel Fails When Using Hexadecimal Notation 5-142
5.40.3 Delimited Identifier Problem in AS Clause of Select Statement 5-143
5.40.4 Configuration File Problem in "DEFAULT" Server Definition 5-143
5.40.5 Pool Server User Restriction Problem 5-144
5.40.6 Potential Problem When Dumping SQLDA in Trace 5-145
5.40.7 Connection.getCatalog() Returns Wrong Value 5-145
5.40.8 Potential Memory Leak With Views 5-145
5.41 New Features for Release 7.2.5.1 5-146
5.41.1 SQLDA Dumping 5-146
5.41.2 failSAFE IP With Pool Servers 5-146
5.41.3 HandshakeTries and Wait on Multiprocess Native Connections 5-146
5.41.4 Server Access Security Enhancements 5-147
5.41.5 Restriction on Using Multiple Blob Fields in Join Now Removed 5-147
5.42 Corrections in Release 7.2.5.1 5-148
5.42.1 Incorrect Row Number Returned After ResultSet.getLast() Call 5-148
5.42.2 Pool Server Startup of Pooled Servers May Fail When Persona is Used 5-149
5.42.3 Last Column in Select List May be Inaccessible in Some Queries 5-150

5.42.4 Abnormal Client Termination May Prevent Executor Re-use 5-150
5.42.5 Decimal Column Problem With Native Driver 5-151
5.42.6 'EFN xx is not available' Message on Executor Startup 5-151
5.42.7 Extraneous Log Message During Auto-restart Check by Pool Server 5-152
5.42.8 Logfile Not Correctly Set for Servers Started Using the Controller 5-152
5.43 New Features for Release 7.2.5 5-152
5.43.1 Persona 5-152
5.44 Corrections in Release 7.2.5 5-153
5.44.1 Incorrect STARTUP72 Installed With V7.2-41 Kit 5-153
5.44.2 Multi-process Server Continuous DIO Activity When Idle 5-153
5.44.3 Client idleTimeout Problem in Reused Prestarted Executors 5-154
5.44.4 Syntax Error in Query Generated for DatabaseMetaData.getTables 5-154
5.44.5 Show Clients in Controller May Crash Connected Thin Server 5-155
5.45 New Features for Release 7.2.4.1 5-156
5.45.1 Client and Server Timeout Feature 5-156
5.45.2 Executor Name Prefix 5-156
5.46 Corrections in Release 7.2.4.1 5-156
5.46.1 Incorrect Installation Directory for RDBJDBCCFG.XML 5-156
5.46.2 Persona Not Handled Correctly by the Multi-process and Pool Servers .. 5-157
5.46.3 Multi-process Server Timeout May be Too Short 5-157
5.46.4 SSL Problems With srv.idleTimeout and srv.bindTimout 5-158
5.46.5 IA-64 Array Out of Bounds Exception with String Indexing 5-159
5.46.6 Comments Within SQL Text Not Handled Correctly 5-159
5.46.7 Prepared Statements Memory Leak With Multi-process Servers 5-160
5.47 Corrections in Release 7.2.4 5-161
5.47.1 Maximum Size of Single Data Row Increased to 65,272 Octets 5-161
5.47.2 Another Connection Overlap Window Found With Pool Servers 5-161
5.47.3 SSL Server Problem with XML-Formatted Configuration File 5-162

Preface

Purpose of This Manual

The Oracle JDBC for Rdb 7.4.1.1 release notes summarize new features, corrections to software, restrictions, workarounds, and problems. They also include new features and corrections provided in releases 7.2.4. through 7.4.1. These release notes cover Oracle JDBC for Rdb for OpenVMS on both Alpha and Integrity Servers.

Audience

This document is intended for users responsible for:

- System management
- Database administration
- Application programming

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Reader Comments

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title, release date, chapter, section, and page number (if available).

Please direct all comments, and corrections to this email address: **infordb_mb@oracle.com.**

If you have problems with the software, please contact your local Oracle Support Services.

Document Structure

This document consists of five chapters:

Chapter 1	Describes location of documents and installation directions.
Chapter 2	Describes new features and technical changes in this release.
Chapter 3	Describes corrected software errors in this release.
Chapter 4	Describes known problems, restrictions, and workarounds.
Chapter 5	Describes new features and corrected software errors in releases 7.2.4 through 7.4.1.

Conventions

The Oracle JDBC for Rdb documentation set to which this manual belongs often refers to the following by their abbreviated names:

- Oracle JDBC for Rdb is referred to as Rdb JDBC or sometimes just JDBC.
- Oracle Rdb refers to Oracle Rdb Server for OpenVMS Alpha and OpenVMS IA-64 software.
- Oracle SQL/Services and OCI Services for Oracle Rdb is referred to as Oracle SQL/Services.
- Oracle Rdb Connectivity Manager is referred to as ORCM.
- Oracle Rdb Data Provider for .NET is referred to as ORDP.
- OpenVMS Industry Standard 64 for Integrity Servers is referred to as OpenVMS IA-64.
- OpenVMS refers to both the OpenVMS IA-64 and OpenVMS Alpha operating systems.
- Hewlett-Packard Enterprise Company is referred to as HPE.
- VMS Software Inc. is referred to as VSI.
- The Oracle JDBC for Rdb Multi-process Server is referred to as the MP server.
- The Oracle JDBC for Rdb Pool Server is referred to as the Pool server.

The following conventions are used in this document:

word	A lowercase word in a format example indicates a syntax element that you supply.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
{ }	Braces enclose clauses from which you must choose one alternative.
...	A horizontal ellipsis means you can repeat the previous item.
. . .	A vertical ellipsis in an example means that information not directly related to the example has been omitted.

Conventions in Code Examples

Code examples illustrate Java code, SQL or other command-line statements. They are displayed in a monospace (fixed width) font and separated from normal text as shown in this example:

```
SELECT last_name FROM employees WHERE last_name = 'TOLIVER';
```

The text of exception messages that may be raised are also displayed using the same convention.

▲contents

Chapter 1
Installation and Documentation

This chapter contains installation and documentation information for Oracle JDBC for Rdb 7.4.1.1.

1.1 Accessing the Documentation

You can extract release notes or an Oracle JDBC for Rdb document from the PCSI kit prior to installation by following one of these procedures:

- To extract a copy of the release notes, define PCSI$SOURCE to point to the location (device name and directory) of the PCSI kit. Then, enter the PRODUCT EXTRACT RELEASE_NOTES command followed by the product name at the DCL prompt.

```
$ DEFINE PCSI$SOURCE DKA400:[KITS]
$ PRODUCT EXTRACT RELEASE_NOTES RDBJDBC74
```

- To extract a list of files contained in a software product kit, define PCSI$SOURCE to point to the location (device name and directory) of the PCSI kit. Then, enter the PRODUCT LIST command followed by the product name at the DCL prompt.

```
$ DEFINE PCSI$SOURCE DKA400:[KITS]
$ PRODUCT LIST RDBJDBC74
```

- To extract a specified file, define PCSI$SOURCE to point to the location (device name and directory) of the PCSI kit. Then, enter the PRODUCT EXTRACT FILE command followed by the product name and file name at the DCL prompt.

```
$ DEFINE PCSI$SOURCE DKA400:[KITS]
$ PRODUCT EXTRACT FILE RDBJDBC74/SELECT=filename.ext
```

The Oracle JDBC for Rdb documentation is also available on My Oracle Support.

The installation procedure copies the Oracle JDBC for Rdb release notes to the SYS$HELP directory.

1.2 System and Software Requirements

On the server side, Oracle JDBC for Rdb requires the following software products to be installed:

Software	Version
OpenVMS	V8.4
Java ™ SDK/RTE	V5.0-1 (1.5.0-1)
Oracle Rdb	V7.3

On the client side, you must install the following software product in order to use the Oracle JDBC for Rdb Thin driver:

Software	Minimum Version
Java ™ SDK/RTE	V5.0-1 (1.5.0-1)

In addition, if you need to start and stop Oracle JDBC for Rdb servers using Oracle SQL/Services, the following product must be installed:

Software	Minimum Version	
	OpenVMS Alpha	OpenVMS IA-64
Oracle SQL/Services	V7.3	V7.3

Documentation and information about installing HPE Java for OpenVMS systems may be found at the following web site:

http://www.hpe.com/

An important note for HPE customers using Java may be found at the following information page:

OpenVMS - Java Support on HPE OpenVMS and VSI OpenVMS

Documentation and information about installing VSI Java for OpenVMS systems may be found at the following web site:

https://vmssoftware.com/products/java/

In line with the HPE and VSI recommendations for Java applications, Oracle recommends the following minimum quota setting on accounts used to start up Thin servers, in particular those used to start Multi-process servers.

FILLM	4096
CHANNELCNT	4096
WSDEF	4096
WSQUO	8192
WSEXTENT	32768
WSMAX	32768
PGFLQUO	3145728
BYTLM	4000000
BIOLM	150
DIOLM	150
TQELM	100

Be sure to set your systems quotas appropriately to accommodate these process quotas.

See the Java for OpenVMS release notes for more information on OpenVMS quotas and resources required by Java.

Also refer to your Oracle Rdb documentation for recommendations on OpenVMS quotas required for Oracle Rdb.

1.2.1 UNZIP is required

The installation kit for Oracle JDBC for Rdb is bundled as a ZIP file and requires the use of UNZIP prior to the product installation.

ZIP and UNZIP are Open Source tools; the OpenVMS version of these tools are available for download from VMS Software, Inc. Please download and install these tools prior to proceeding with the Oracle JDBC for Rdb installation.

https://products.vmssoftware.com/zip

1.3 Installation

This section describes how to install Oracle JDBC for Rdb and includes a sample log.

1.3.1 SQL/Services JDBC Server Startup Command Procedures

During installation copies of the SQL/Services JDBC server startup command procedures will be placed in both the installation directory and SYS$MANAGER.

These files are named :

SQLSRV_JDBC_SERVER_STARTUP<*sqs_vers*>.COM

where <*sqs_vers*> reflects the major version numbers of SQL/Services releases.

For example:

SQLSRV_JDBC_SERVER_STARTUP73.COM
SQLSRV_JDBC_SERVER_STARTUP74.COM

Prior to JDBC release 7.3.5 any existing SQL/Services JDBC Server command procedures found within SYS$MANAGER would not be replaced during JDBC installation. However, changes to SQL/Services JDBC processing in release 7.3.5 require that these command procedures may now be replaced.

Starting with JDBC release 7.3.5, during installation JDBC “archives” the old version of the command procedure found in SYS$MANAGER by appending _OLD to the end of the file type, prior to copying over the new command procedures.

Note:

If you have made any site-specific changes to these command procedures, you will have to reapply these changes to the updated command procedures in SYS$MANAGER.

As the releases of Oracle JDBC for Rdb are independent of the releases of Oracle SQL/Services, the currently installed version of Oracle JDBC for Rdb may not have

installed an appropriate SQL/Services JDBC Server command procedure for all SQL/Services versions installed on your system.

If this is the case, JDBC dispatchers may not start up correctly for the installed SQL/Services version.

To fix this problem you can simply copy an existing SQL/Services JDBC Server command procedure within SYS$MANAGER: and alter the version number of its filename to reflect the SQL/Services version you are using.

1.3.2 Contents of the Oracle JDBC for Rdb Kit

The Oracle JDBC for Rdb kit uses OpenVMS POLYCENTER Software Installation utility to simplify the installation of the product. Please refer to your OpenVMS documentation on the use of OpenVMS POLYCENTER Software Installation utility.

The Oracle JDBC for Rdb kit product installation file is named ORCL-pppVMS-RDBJDBC74-V0704-xxxxxx-1.PCSI where ppp will be the platform and xxxxxx will be the build instance of this kit, for example:

```
  ORCL-AXPVMS-RDBJDBC74-V0704-1V1Q34-1.PCSI
or
  ORCL-I64VMS-RDBJDBC74-V0704-1V1Q34-1.PCSI
```

The installation file is contained in the RDBJDBCV74*.ZIP file.

The installation kit is comprised of the following files:

BUILD_CERTS.COM	Command procedure example of building certificates for SSL.
OracleJDBCForRdbClient74000.msi OracleJDBCForRdbClient74000X64.msi	Microsoft installer file for client-side software installation on Windows.
RDBJDBCCHECKUP.CLASS	Use to verify the installation of this kit.
RDBJDBCCHECKUP.JAVA	Use to verify the installation of this kit.
RDBJDBCEXEC74.EXE	Executor image in conjunction with a Multi-process server.
RDBJDBCEXEC6474.EXE	Executor image in conjunction with a Multi-process server on OpenVMS

	IA-64 systems using Java 8.0 or higher.
RDBJDBCCFG.XML	Example XML formatted configuration file.
RDBJDBCSHR74.EXE	Shared image required for Oracle Rdb database access.
RDBJDBCSHR6474.EXE	Shared image required for Oracle Rdb database access on OpenVMS IA-64 systems using Java 8.0 or higher.
RDBJDBCMPSHR74.EXE	Shared image required for Multi-process Oracle Rdb database access.
RDBJDBCMPSHR6474.EXE	Shared image required for Multi-process Oracle Rdb database access on OpenVMS IA-64 systems using Java 8.0 or higher.
RDBJDBC_EXECCLI.COM	CLI helper command procedure.
RDBJDBC_INSTALL.COM	Installation command procedure used by PRODUCT during installation.
RDBJDBC_PRECONFIGURE.COM	Installation command procedure used by PRODUCT during the pre-configuration stage.
RDBJDBC_STARTEXEC.COM	Command procedure used by Oracle JDBC for Rdb Multi-process server to start up an executor process.
RDBJDBC_STARTSRV.COM	Command procedure used when an Oracle JDBC for Rdb server is started up from the Oracle JDBC for Rdb controller.
RDB_EXT.JAR	Java jar file containing classes and Java code for extensions to Oracle JDBC for Rdb.

RDBNATIVE.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb native driver.
RDBNATIVEV<Java_version>.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb native driver built to run on a specified Java Version. This is discussed below.
RDBTHIN.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb thin driver.
RDBTHINV<Java_version>.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb thin driver built to run on a specified Java version. This is discussed below.
RDBTHINSRV.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb servers.
RDBTHINSRVV<Java_version>.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb thin driver built to run on a specified Java version. This is discussed below.
RDBTHINCONTROL.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb controller.
RDBTHINCONTROLV<Java_version>.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb controller built to run on a specified Java version. This is discussed below.
RDBTHINSRVPOOL.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb pool server.
RDBTHINSRVPOOLV<Java_version>.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb pool server built to run on a specified Java

	version. This is discussed below.
RDBMANSRV.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb manager server.
RDBMANSRVV<Java_version>.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb manager server built to run on a specified Java version. This is discussed below.
RDBJDBC_<version>_RELNOTES.PDF	Oracle JDBC for Rdb Release Notes.
RDBJDBC_<version>.RELEASE_NOTES	Text version of Oracle JDBC for Rdb release notes.
RDBJDBC_USERGUIDE.PDF	User guide.
SQLSRV_JDBC_SERVER_STARTUP*.COM	Command procedures used by Oracle SQL/Services to start up an Oracle JDBC for Rdb server.
WATCHEVENTS.JAR	Sample application for watching Oracle JDBC for Rdb server events.

Note:

The HTML version of the user guide and release notes are no longer supplied.

Starting with Oracle JDBC for Rdb Release 7.4.1.1, the installation of Oracle JDBC for Rdb will install varianted versions of most of the JDBC JAR files. These JAR files are built to run in conjunction with a minimum Java version as shown in the following table:

Driver JAR file	Minimum version of Java required	JDBC supported
RDBNATIVEV5.JAR RDBTHINV5.JAR RDBTHINSRVV5.JAR	JDK 1.5	JDBC 3.0, as released in JDK 1.4

RDBTHINCONTROLV5.JAR RDBTHINSRVPOOLV5.JAR RDBMANSRVV5.JAR		
RDBNATIVEV6.JAR RDBTHINV6.JAR RDBTHINSRVV6.JAR RDBTHINCONTROLV6.JAR RDBTHINSRVPOOLV6.JAR RDBMANSRVV6.JAR	Java SE 6	JDBC 4.0 as released in Java SE 6
RDBNATIVEV8.JAR RDBTHINV8.JAR RDBTHINSRVV8.JAR RDBTHINCONTROLV8.JAR RDBTHINSRVPOOLV8.JAR RDBMANSRVV8.JAR	Java SE 8 or openJDK8	JDBC 4.2 as released in Java SE 8 or available using OpenJDK 8

In keeping with prior versions, the installation procedure will also install non-versioned variants of these JAR files:

- RDBNATIVE.JAR
- RDBTHIN.JAR
- RDBTHINSRV.JAR
- RDBTHINCONTROL.JAR
- RDBTHINSRVPOOL.JAR
- RDBMANSRV.JAR

These JAR files are identical to the JDK 1.5 varianted JAR files.

Note:

To start JDBC servers or the controller in an OpenJDK8 Java environment, you must use the *V8 variants of the server or controller JARs. If the *V8 variants of the JARs are not used the invocation of the server or controller will fail.

1.3.3 Installation Procedure

Follow these steps to install the Oracle JDBC for Rdb kit:

1 Download and unzip the Oracle JDBC for Rdb kit from Oracle.

The Oracle JDBC for Rdb kit comes as a ZIP file containing the installation PCSI file and some documentation.

Note:

VMS ZIP utility -ZIP and UNZIP are open-source tools; the OpenVMS version of these tools are available for download from VMS Software, Inc.

Please download and install these tools prior to proceeding with this installation.

https://products.vmssoftware.com/zip

Use the UNZIP utility (the symbol should be defined as a foreign command) to extract the files from the archive into an installation sub-directory.

That directory can then be referenced by the OpenVMS POLYCENTER Software Installation utility as described below.

Restore the kit file to a temporary directory:

```
$ unzip RDBJDBCV74110_IMV.ZIP
Archive:  DISK$USERS:[SOFTWARE.JDBC_KIT]RDBJDBCV74110_IMV.ZIP;1
************************************************************

Oracle Rdb Engineering Group
Oracle JDBC for Rdb V7.4-110(IA64) - Production
Archive created 1-JAN-2026 00:50:00.00
Copyright (c) 2026, Oracle Corporation.  All Rights Reserved.

************************************************************
  inflating: ORCL-I64VMS-RDBJDBC74-V0704-1V1Q34-1.PCSI
  inflating: RDBJDBC_README_7411.HTML
  inflating: RDBJDBC_V07411.RELEASE_NOTES
  inflating: RDBJDBC_V07411_RELNOTES.PDF
```

This will unzip the PRODUCT kit for Oracle JDBC for Rdb and you will have access to the PCSI file, ORCL-pppVMS-RDBJDBC74-V0704-xxxxxx-1.PCSI, where ppp is the platform and xxxxxx is the build instance of this kit.

2 If you have a V73 or prior release of Oracle JDBC for Rdb already installed you must carry out some extra operations prior to installing the new version.

If the prior version of Oracle JDBC for Rdb was installed into the default JDBC installation directory (i.e., the PRODUCT INSTALL/DESTINATION qualifier was

not used during installation) you must choose to either remove this old version using PRODUCT REMOVE or alternatively install the new version using PRODUCT INSTALL/DESTINATION to install the product to a different location. Failure to do so may cause problems in the installation of the new version.

If you choose to carry out a PRODUCT REMOVE of an earlier version, but you wish to save a copy of this existing JDBC installation, you should do this before you remove the old version.

The installation of JDBC as shown below provides an option to take a backup of the prior version. This feature may be used to create your backup before removing the old version by:

a. running the procedure as shown in the following step (step 4) but answer NO when asked if you wish to continue after the backup has been taken
b. next, remove the old version using PRODUCT REMOVE
c. then run the installation (step 4) again. This time, as you have already removed the older version, you will not be prompted to optionally backup the old version.

3 Use the POLYCENTER Software Installation utility PRODUCT command to install the kit.

Details of the version of the kit will be displayed and you will be asked if you want to proceed. The following examples of installation on an ALPHA system assume the kit build instance is 1V1Q34, and that the directory where the PCSI file can be found is MY_DIR. This example also assumes you already have a prior version of Oracle JDBC for Rdb installed on your system.

```
$ PRODUCT INSTALL RDBJDBC74/SOURCE=MY_DIR
The following product has been selected:
    ORCL I64VMS  RDBJDBC74 V7.4-1V1Q34      Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and
for any products that may be installed to satisfy software dependency
requirements.

Configuring ORCL I64VMS RDBJDBC74 V7.4-1V1Q34: Oracle JDBC for Rdb

    Copyright © 1995, 2026, Oracle Corporation.  All Rights Reserved.

Prior to installation you may wish to backup the old version of Oracle JDBC
for Rdb currently installed in DISK1:[SYS0.SYSCOMMON.RDB$JDBC.0704-1V0P1J]

Do you wish to backup the old version (Press Y/N) ? : y
Oracle JDBC for Rdb has been successfully backedup to
```

```
  DISK1:[SYS0.SYSCOMMON.RDB$JDBC]0704-1V0P1J.SAV



* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    ORCL I64VMS RDBJDBC74 V7.4-1V1Q34      DISK1:[VMS$COMMON.]

Portion done: 0%...10%...30%...40%...50%...60%...70%...80%...90%

Oracle JDBC for Rdb has been successfully installed in :

DISK$IA64VMSSYS1:[SYS2.SYSCOMMON.rdb$jdbc.0704-1V1Q34]

To help you setup the required logical names, a file named
RDBJDBC_STARTUP.COM has been added to this installation directory.

If you wish, insert the following line in SYS$MANAGER:SYSTARTUP_VMS.COM:
  @DISK$IA64VMSSYS1:[SYS2.SYSCOMMON.RDB$JDBC.0704-1V1Q34]RDBJDBC_STARTUP.COM

RDBJDBC_STARTUP.COM:
$!
$! Oracle JDBC for Rdb startup command procedure. Built from kit 0704-1V1Q34.
Created  1-JAN-2026 20:23:47.96.
$!
$ DSE = "DEFINE/SYSTEM/EXECUTIVE_MODE"
$ DSE RDB$JDBC_HOME DISK$IA64VMSSYS1:[SYS2.SYSCOMMON.rdb$jdbc.0704-1V1Q34]
$ DSE RDB$JDBC_LOGS DISK$IA64VMSSYS1:[SYS2.SYSCOMMON.rdb$jdbc.logs]
$ DSE RDB$JDBC_COM  DISK$IA64VMSSYS1:[SYS2.SYSCOMMON.rdb$jdbc.com]
$ DSE RDBJDBCSHR RDB$JDBC_HOME:RDBJDBCSHR74.EXE
$ DSE RDBJDBCMPSHR RDB$JDBC_HOME:RDBJDBCMPSHR74.EXE
$ DSE RDBJDBCEXEC RDB$JDBC_HOME:RDBJDBCEXEC74.EXE
$ DSE RDBJDBCSHR64 RDB$JDBC_HOME:RDBJDBCSHR6474.EXE
$ DSE RDBJDBCMPSHR64 RDB$JDBC_HOME:RDBJDBCMPSHR6474.EXE
$ DSE RDBJDBCEXEC64 RDB$JDBC_HOME:RDBJDBCEXEC6474.EXE
$!
$! Enable the use of the SQL/Services JDBC dispatcher name as
$! the default JDBC server name during creation of a SQL/Services
$! JDBC Dispatcher.  If you wish to default to SQS<port> then
$! deassign this logical name.
$!
$ DSE RDB$JDBC_SQS_USE_DISPNAME  "TRUE"

...100%

The following product has been installed:
    ORCL I64VMS RDBJDBC74 V7.4-1V1Q34      Layered Product
```

During the pre-configuration phase of the installation, if a prior version of JDBC has been installed and the logical name RDB$JDBC_HOME can be translated, you will be provided with the opportunity to save a backup copy of this earlier version.

During the installation, the PRODUCT utility will remove the old version of the JDBC software. If you wish to retain a copy of this earlier version, you should answer ‘Y’ when prompted by PRODUCT for the optional backup.

If you have chosen to back up the older version of JDBC a BACKUP SAV file will be created in the installation's parent directory:

```
$ dir  DISK1:[SYS0.SYSCOMMON.rdb$jdbc]0704-1V0P1J.sav

Directory DISK1:[SYS0.SYSCOMMON.RDB$JDBC]

0704-1V0P1J.SAV;1

Total of 1 file.
```

The installation procedure will copy all the kit files to the appropriate Oracle JDBC for Rdb product directory in the SYS$COMMON:[RDB$JDBC] directory, for example:

```
SYS$COMMON:[RDB$JDBC.0704-1V1Q34]
```

If they are not already present, the installation procedure will create two new directories, SYS$COMMON:[RDB$JDBC.LOGS] and SYS$COMMON:[RDB$JDBC.COM].

Note:

During installation a generic configuration file RDBJDBCCFG_TEMPLATE.XML will be copied to the RDB$JDBC_HOME directory. Also, during installation, if the SYS$COMMON:[RDB$JDBC.COM] directory does not already contain a file named RDBJDBCCFG.XML, the contents of the configuration template file will be used to create this file.

By default, as the logical name RDB$JDBC_COM points to the SYS$COMMON:[RDB$JDBC.COM] directory, it is this directory that the Oracle SQL/Services JDBC dispatcher will use when searching for a configuration file to use during server startup. (See the sections *JDBC Dispatcher* and *Determining the server configuration file* in the *Oracle JDBC for Rdb User Guide* for more details).

The RDBJDBCCFG_TEMPLATE.XML found in the RDB$JDBC_HOME directory will be replaced each time you install Oracle JDBC for Rdb, however, any existing RDBJDBCCFG.XML file found in the RDB$JDBC_COM directory will not be replaced.

Oracle recommends using RDB$JDBC_HOME:RDBJDBCCFG_TEMPLATE.XML only as a template file and not to use this file in production.

```
$ dir sys$common:[rdb$jdbc]/col=1

Directory SYS$COMMON:[RDB$JDBC]

0703-5V0H4K.DIR;1
0703-5V0H4K.SAV;1
0704-1V1Q34.DIR;1
COM.DIR;1
LOGS.DIR;1
```

In addition, the command procedures `SQLSRV_JDBC_SERVER_STARTUP*.COM` will be copied to the system specific SYS$MANAGER directory.

Note:

During the removal of the prior version of JDBC, PRODUCT will leave any RDBJDBC_STARTUP.COM file found in the older version's directory. This may be used as a reference file in case you have made any customizations.

---

4 Use the command procedure `RDBJDBC_STARTUP.COM` found in the Oracle JDBC for Rdb product installation directory to define the required system logical names:

RDB$JDBC_HOME to point to the installation home.

```
$ define/system RDB$JDBC_HOME SYS$COMMON:[RDB$JDBC.0704-1V1Q34]
```

RDB$JDBC_LOGS to point to the Oracle JDBC for Rdb log directory.

```
$ define/system RDB$JDBC_LOGS SYS$COMMON:[RDB$JDBC.LOGS]
```

RDB$JDBC_COM to point to the Oracle JDBC for Rdb command directory.

```
$ define/system RDB$JDBC_COM SYS$COMMON:[RDB$JDBC.COM]
```

RDBJDBCSHR to point to the shared image RDBJDBCSHR74.EXE.

```
$ define/system RDBJDBCSHR -
SYS$COMMON:[RDB$JDBC.0704-1V1Q34]RDBJDBCSHR74.EXE
```

RDBJDBCMPSHR to point to the shared image RDBJDBCMPSHR74.EXE.

```
$ define/system RDBJDBCMPSHR –
SYS$COMMON:[RDB$JDBC.0704-1V1Q34]RDBJDBCMPSHR74.EXE
```

RDBJDBCEXEC to point to the shared image RDBJDBCEXEC74.EXE.

```
$ define/system RDBJDBCEXEC –
SYS$COMMON:[RDB$JDBC.0704-1V1Q34]RDBJDBCEXEC74.EXE
```

RDBJDBCSHR64 to point to the shared image RDBJDBCSHR6474.EXE. (Used only on **Integrity** systems.)

```
$ define/system RDBJDBCSHR64 –
SYS$COMMON:[RDB$JDBC.0704-1V1Q34]RDBJDBCSHR6474.EXE
```

RDBJDBCMPSHR64 to point to the shared image RDBJDBCMPSHR6474.EXE. (Used only on **Integrity** systems.)

```
$ define/system RDBJDBCMPSHR64 –
SYS$COMMON:[RDB$JDBC.0704-1V1Q34]RDBJDBCMPSHR6474.EXE
```

RDBJDBCEXEC64 to point to the shared image RDBJDBCEXEC6474.EXE. (Used only on **Integrity** systems.)

```
$ define/system RDBJDBCEXEC64 –
SYS$COMMON:[RDB$JDBC.0704-1V1Q34]RDBJDBCEXEC6474.EXE
```

You must define the RDB$JDBC_HOME logical name if you want to use a Thin Multi-process server or use the Controller or Pool servers or the SQL/Services JDBC dispatcher to start server processes.

5 Include the rdbnative and rdbthin jar files in your Java CLASSPATH by using either the logical names CLASSPATH or JAVA$CLASSPATH or the -classpath option on the Java command line, for example:

```
$ define JAVA$CLASSPATH -
[],RDB$JDBC_HOME:RDBNATIVE.JAR,RDB$JDBC_HOME:RDBTHIN.JAR
```

By default the RDBNATIVE.JAR and RDBTHIN.JAR files contain classes built to run with JDBC 3.0 as found in JDK 1.5. The JAR files are compatible with later releases of Java, however JDBC features that were first introduced for JDBC 4.0 in Java SE 6 or later are not supported by these drivers.

On INTEGRITY systems, if you require the use of JDBC features that were first introduced in Java SE 6 or later then you should choose one of the varianted driver version JAR files.

For example if the JDBC feature was introduced in Java SE 6 then you could use either the V6 or V8 variants of the driver jars:

```
$ define JAVA$CLASSPATH -
[],RDB$JDBC_HOME:RDBNATIVEV6.JAR,RDB$JDBC_HOME:RDBTHINV6.JAR
```

If the feature was introduced in Java SE 7 or Java SE 8 then use the V8 variants:

```
$ define JAVA$CLASSPATH -
[],RDB$JDBC_HOME:RDBNATIVEV8.JAR,RDB$JDBC_HOME:RDBTHINV8.JAR
```

6 Test your installation using the "RdbJdbcCheckup" Java class in the RDBJDBCCHECKUP.CLASS file. During the installation RDBJDBCCHECKUP.CLASS is copied to RDB$JDBC_HOME. Copy this file to your default directory and then you can invoke it using Java.

You will be prompted for a username and password and an Oracle Rdb database to test the installation against. If the test succeeds, the text "*Your JDBC installation is correct.*" is displayed.

```
$ java "RdbJdbcCheckup"
Please enter information to test connection to the database
user: my_username
password: my_password
database: my_db_dir:personnel
Connecting to the database...Connecting
connected.
Hello World
Your JDBC installation is correct.
$
```

Test the Thin server by using the following commands:

```
$spawn/nowait/proc=rdbthinsrvtest java -jar rdbthinsrv.jar
$java "RdbJdbcCheckup" "-t"
Please enter information to test connection to the database
user: my_username
password: my_password
database: my_db_dir:personnel
Connecting to the database...Connecting...
connected.
Hello World
Your JDBC installation is correct.
$stop rdbthinsrvtest
```

Note:

As Java is a case-sensitive language, it is important to specify class and method names exactly as they are described in the various APIs. By default, the OpenVMS operating system uppercases command line parameters unless you surround them with double quotation marks.

1.3.4 Rdb Thin Driver Software on Windows

If you are developing or deploying Windows applications that use the Oracle JDBC for Rdb thin driver, you will have to copy the provided **RDBTHIN.JAR** file to each of the Windows systems you will be using. In addition, you should consult your application, IDE or Microsoft Windows software guides to determine how to let the application or development environment know where the Rdb thin driver jar can be found.

To facilitate installation of the Rdb thin driver on your Windows client systems, included in the Oracle JDBC for Rdb installation kit are two Microsoft Software Installation images, **OracleJDBCForRdbClient74000.msi** and **OracleJDBCForRdbClient74000X64.msi** that may be used on your client Windows systems to install the RdbThin and the RdbThinControl jars.

The **OracleJDBCForRdbClient74000X64.msi** may be used on Windows 64 Bit systems. **OracleJDBCForRdbClient74000.msi** can also be used on Windows 64 Bit systems, however, the installation will be placed in the 32-bit subsystem.

Once the appropriate MSI file has been copied to your Windows system, double-click the filename to invoke the Microsoft installer, and then follow the directions provided by the installation Wizard.

This installation will place a copy of the RdbThin and RdbThinControl jars as well as the current JDBC documentation into the designated installation directory on your Windows system.

In addition, pointers to the uninstall procedure, the Rdb Thin Controller shortcut and documentation will be added to your Program Menu.

1.3.5 Rdb Thin Driver Software on Other Operating Systems

If you are developing or deploying applications that use the Oracle JDBC for Rdb thin driver, you will have to copy the provided **RdbThin.jar** file to each of the systems you will be using.

You should consult your application, IDE, or operating system software guides to determine how to let the application or development environment know where the Rdb thin driver jar can be found.

▲contents

Chapter 2
Enhancements Provided in Oracle JDBC for Rdb Release 7.4.1.1

This chapter describes new and changed features in Oracle JDBC for Rdb release 7.4.1.1.

2.1 Support for OpenJDK8

This release of JDBC includes new JDK versioned JAR files for use with Oracle JDBC for Rdb servers and the controller.

If you wish to run the JDBC servers or the controller under an OpenJDK8 environment on OpenVMS you must use the *V8.JAR variants of the appropriate JDBC server or controller JARs:

- RDBTHINSRVV8.JAR
- RDBTHINCONTROLV8.JAR
- RDBTHINSRVPOOLV8.JAR
- RDBMANSRVV8.JAR

Note:

If the *V8 variants of the JARs are not used when you are using the OpenJDK8 environment on OpenVMS, the invocation of the server or controller will fail.

See the sections ***JDK Versioned Oracle JDBC for Rdb Server Jars*** and ***JDK Versioned Oracle JDBC for Rdb Controller Jars*** in the ***Oracle JDBC for Rdb User Guide*** for more information on these new JAR files.

▲contents

Chapter 3
Problems Corrected

This chapter describes software errors corrected in Oracle JDBC for Rdb release 7.4.1.1.

3.1 ACCVIO in MPServer with Many Open Queries using JDK8

Release 7.4.1.1.

A problem has been found in rare cases in Oracle JDBC for Rdb release 7.4.1 where an MPServer with a number of open queries may ACCVIO when trying to allocate MPServer shared memory. This problem has only been observed when using release 7.4.1 and a MPServer running under JDK8 or OpenJDK8.

The ACCVIO logged by the server will indicate a problem in:

```
RDBJDBC_MP/mp_get_chunk
```

And may be similar to the following:

```
# A fatal error has been detected by the Java Runtime
Environment:
#
#  %SYSTEM-F-ACCVIO (0xc) at pc=73F1E50, pid=549491203,
tid=0x0000000007f63200
#
# JRE version: OpenJDK Runtime Environment (8.0_372-b07)
(build 1.8.0_372-b07)
# Java VM: OpenJDK 64-Bit Server VM (25.372-b07 mixed
mode OpenVMS-ia64 )
# Problematic frame:
# C  [RDBJDBCMPSHR6474+0xf0]
RDBJDBC_MP/mp_get_chunk+0x73f1e50
```

This problem has now been fixed.

▲contents

Chapter 4
Known Problems and Workarounds

This chapter describes known problems, restrictions, and workarounds for Oracle JDBC for Rdb Release 7.4.1.1.

4.1 Thin Server Deadlocks

When using a thin server, the Rdb database environment and binds are within the OpenVMS process context of the server process.

When multiple clients attach to a database using the same server, each client is allocated its own thread and has its own connection context within Rdb, but from the Rdb engine's viewpoint, all of these connections are held by a single process.

The locking behavior of Rdb for connections within a single process differs from that of connections made using discrete OpenVMS processes. In particular when two separate processes try to get the same lock, by default, the second process will wait until the first releases the lock. In contrast when two connections within the same process try to get the same lock, a deadlock may be raised.

Oracle recommends, when using thin servers, that the `lockwait` connection switch or server option be used to prevent deadlocks. By choosing a lockwait duration shorter than your system deadlock wait, and then choosing an appropriate `maxtries` value which determines how many times the server will attempt to get the lock before giving up, deadlocks may be prevented. In addition, keeping your transaction duration small will reduce the time locks are held and thus allow better concurrency.

See *Lockwait and Maxtries* in the *Oracle JDBC for Rdb User Guide* for more information.

4.2 Using Java Fast VM on OpenVMS ALPHA

Using Java Fast VM on OpenVMS Alpha when starting thin servers may limit the number of clients a single server may be able to handle concurrently. This is because using Fast VM drastically reduces the amount of certain system memory that the Oracle Rdb subsystem has access to.

The usual symptom of running out of memory due to this situation is when the server process issues COSI-VASFULL errors.

Refer to the OpenVMS Java documentation on using Fast VM for suggestions on how memory usage may be altered.

Heap size used by the Java VM is important in determining how much memory will be pre-allocated by the Java VM. You can set the size of the heap using the -Xmx option. By default, the Fast VM looks at your quota and the size of physical memory on the system to decide how large a heap to give you. So if both are very large, you may wind up with a larger heap than you really need. You can use **-verbose:gc** on the command line of the command used to start a server to see the current heap size.

Memory usage may also be altered by using the "-Xglobal" switch.

If the thin servers are getting COSI-VASFULL errors when Fast VM is enabled, Oracle suggests trying the following switch settings as a first pass at rectifying the problem.

```
$ java "-Xmx24m" "-Xglobal120m" -jar rdbthinsrv.jar
```

4.3 Using the Oracle SQL/Services Management GUI and JDBC Dispatchers

The unsupported Oracle SQL/Services Management GUI does not recognize dispatchers of the type JDBC. Unfortunately, this means that you will no longer be able to use this GUI once a JDBC dispatcher has been defined.

Removing the JDBC dispatcher from your Oracle SQL/Services definitions will alleviate this problem.

Note: The unsupported Oracle SQL/Services Management GUI has been replaced with the newer Java-based Oracle Rdb Connectivity Manager GUI (ORCM).

4.4 Blob Columns and Correlation Names

Due to the nature of the parsing carried out by the Oracle JDBC for Rdb drivers it is required that all blob columns referenced from the second and subsequent tables in a multi-table join must be qualified using correlation names as shown below:

```
Select ta.blob, tb.blob from table1 ta, table2 tb
```

```
  where ta.name = tb.name
```

Failure to use a correlation name in conjunction with the blob column name may result in SQL parsing errors when data is retrieved from the blob field as the drivers do not have enough information to determine the correct table to access the blob data from.

```
SQL-F-FLDNOTCRS, Column <blob col> was not found in the tables
 in current scope
```

This limitation also means that the use of "*" in the select clause for a join across two or more tables that include blob fields may also cause a similar SQL error.

4.5 Blob Columns and Update Statements

When the SQL statements compiled by the Oracle JDBC for Rdb drivers contain references to list of byte varying columns (Blobs) the drivers must create auxiliary statements to carry out the required operations on the Blob columns within the database.

These statements handle the preparation of list cursors required by Rdb to access the underlying list of byte varying columns and use the dbkeys of the target rows to establish the correct currency on the parent tables when using the list cursors.

To achieve this, code is added to any Select queries containing references to Blob columns in order to retrieve the dbkey of the parent table row. Code is also added to Insert and Update statements to return the dbkey of the affected row.

If a Blob column is updated within an SQL Update statement, it is required that the execution of the statement only updates a single row as only a single dbkey can be returned to the drivers.

For example:

```
.
.
.
 PreparedStatement prep = Conn.PrepareStatement(
   "update resumes set resume = ? where employee_id = '91111'");
 prep.setBinaryStream(1, bs, sss.length());
 prep.execute();
.
.
.
```

If the resumes table contains more than one row satisfying the query selection criteria then an exception will be raised by Rdb during the execute() :

```
%RDB-E-MULTIPLE_MATCH, record selection criteria should identify
 only one record; more than one record found
```

4.6 External Procedures and Thin Server

Due to the way that Rdb carries out internal communication of handles between different components during external procedure execution it is possible that concurrent clients within the same Thin Server trying to use external procedures may see exceptions related to the Rdb error BAD_DB_HANDLE.

Currently it is not possible to rectify this limitation within the Thin Server, as the process model used internally by Rdb for using external procedures does not align with how multi-threading must be carried out within the Thin Server.
As the Multi-process server (MP) uses a separate OpenVMS process for each connection, it aligns better with the expected Rdb process model and consequently external procedures may be used by concurrent clients.

Oracle suggests that if external procedures are expected to be executed by concurrent clients, an MP Server should be used to handle these connections.

4.7 Limitations

4.7.1 Unsupported Methods

The following JDCB 2.0, JDBC 3.0 and JDK 1.4 methods are not currently supported:

```
Blob.setBytes

Blob.setBinaryStream

Clob.setString

Clob.setAsciiStream

Clob.setCharacterStream

Clob.truncate

Connection.setSavepoint
```

```
Connection.rollback(savepoint)

Connection.releaseSavepoint

DatabaseMeteData.getSQLKeywords

PreparedStatement.setRef

PreparedStatement.setArray

PreparedStatement.setNull(int,int,String)

PreparedStatement.setURL(int,URL)

ResultSet.getRef

ResultSet.getArray

ResultSet.updateAsciiStream

ResultSet.updateBinaryStream

ResultSet.updateCharacterStream

ResultSet.updateRef

ResultSet.updateArray

ResultSet.rowUpdated

ResultSet.rowInserted

ResultSet.rowDeleted

ResultSet.updateBytes

Statement.cancel

Statement.setQueryTimeout

Statement.getMoreResults

Statement.executeUpdate(String sql, int autoGeneratedKeys)

Statement.executeUpdate(String sql, int[] columnIndexes)

Statement.executeUpdate(String sql, String[] columnNames)

Statement.execute(String sql, int autoGeneratedKeys)

Statement.execute(String sql, int[] columnIndexes)
```

```
Statement.execute(String sql, String[] columnNames)
```

The following features or datatypes in JDBC 2.0 and JDBC 3.0 are not supported:

- Array
- Ref
- Clob
- User Defined datatypes
- Scroll cursors
- Savepoints

4.7.2 Auto-generated Keys

The total number of markers and fields allowed in a single SQL statement is 250.

4.7.3 String Truncation Warnings

The Oracle JDBC for Rdb drivers follow the SQL-92 rules for string truncation that differ depending on whether it is a store or retrieval.

If a string truncation happens during a store operation, Oracle Rdb signals the error RDB$_TRUN_STORE, unless all of the truncated characters are spaces, in which case there is no error. If a string truncation happens during a retrieval, Oracle Rdb signals the SQL warning RDMS$K_SQLCODE_TRUNCWARN.

4.7.4 Numeric and String Functions in JDBC

A number of JDBC standard Numeric and String functions are not supported within Oracle Rdb unless you have previously prepared the database for use with OCI Services for Oracle Rdb using the SQL_FUNCTIONSnn.SQL, where nn is the Rdb version.

Refer to the Oracle SQL/Services documentation for more details on using this script.

▲contents

Chapter 5
New Features and Corrections in Previous Releases

5.1 New Features for Release 7.4.1

This section describes new and changed features in Oracle JDBC for Rdb 7.4.1.

5.1.1 General Quality and Performance Improvements

This release of JDBC has been built using newer versions of the VSI C++ and C compilers, and we expect to see improvements in code optimization and resilience.

5.2 Corrections in Release 7.4.1

This section describes software errors corrected in Oracle JDBC for Rdb 7.4.1.

None in this release.

5.3 New Features for Release 7.3.5.3

This section describes new and changed features in Oracle JDBC for Rdb 7.3.5.3.

None in this release.

5.4 Corrections in Release 7.3.5.3

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.5.3.

5.4.1 Extraneous Text on Polling Message

Release 7.3.5.3.
Also Fixed in Release 7.3.5.2.2 build 20190530.

The extraneous literal string "{1}" is displayed in "Polling servers" message when the POLL command is issued from the controller:

```
rdbthincontrol> poll
Polling servers ... {1}
```

5.4.2 SilentTruncate Causes Incorrect Update of Columns

Release 7.3.5.3.

As part of the JDBC **SilentTruncate** feature JDBC will silently truncate input string parameters to match the maximum column width of the associated columns, preventing Oracle Rdb from raising string truncation errors.

In release JDBC 7.3.5.2 a change was made to how SilentTruncate operations determined how much to pre-truncate strings to prevent Rdb from raising string truncation errors. (See JDBC Release Note **SilentTruncate not enabled for some Operations.**)

To do this JDBC relies on the column width information returned to it by Rdb during the describe operation on the SQL statement that is compiled.

A problem during the determination of the amount of truncation required to be applied to the input strings, may cause JDBC to truncate columns by one more octet than required. This may result in incorrect data being stored in the database.

This problem only occurs when all the following conditions are met:

1. **SilentTruncate** has been enabled.
2. A PreparedStatement is used to set column values.
3. A PreparedStatement.setString() method is called on an input parameter and the string value provided has an octet length equal to or greater than the length of the column it will be written to.

This problem has now been fixed, SilentTruncate now works correctly on both select and update parameters.

In addition, it was possible that the earlier fixes to SilentTruncate may interfere with the JDBC feature described in JDBC Release Note PreparedStatement and Parameter

Markers Known Problem now Resolved. Release JDBC 7.3.5.3 now ensures that this feature still works as documented.

Note that as silent string truncation does not comply with SQL standards and the use of this feature may mask errors in your application code, Oracle strongly recommends to not use or rely on this feature.

5.4.3 DatabaseMetaData.getColumnPrivileges Problem

Fixed Release 7.3.5.3.2 build 20190705

The **getColumnPrivileges** method of the standard JDBC **DatabaseMetaData** returns information about privileges associated with columns for the table specified:

TABLE_CAT table catalog (may be null)
TABLE_SCHEM table schema (may be null)
TABLE_NAME table name
COLUMN_NAME column name
GRANTOR grantor of access (may be null)
GRANTEE grantee of access
PRIVILEGE name of access (SELECT, INSERT, ...)
IS_GRANTABLE "YES" if grantee is permitted

Prior to Release 7.3.5.3.2, JDBC fails to return the correct privilege information, and incorrectly repeats the TABLE_NAME as the COLUMN_NAME for all records.

5.4.4 ResultSetMetaData.getColumnDisplaySize Problem

Fixed Release 7.3.5.3.2 build 20190711

The value returned from the method **ResultSetMetaData.getColumnDisplaySize** may be incorrect under the following situation:

1. The column specified has the Rdb datatype **DATE VMS** and
2. The call to **getColumnDisplaySize** is made subsequent to the execution of a prior statement that accesses a datetime column with datatype other than **DATE VMS** and
3. The column position within the ResultSet of the datetime column in the prior statement is the same relative position as the **DATE VMS** column interrogated by the **getColumnDisplaySize.**

Under the above conditions the **getColumnDisplaySize** returns the value of the column display width of the datetime column of the prior statement instead of the expected value of 26.

5.4.5 Connection Option @concurrentThread Problem

Fixed Release 7.3.5.3.3 build 20210427

When using the connection option "@concurrentThread=true" it is possible that thin client applications may fail when trying to prepare SQL statements that have input bind variables.

The following exception may be raised in the client application:

```
java.io.EOFException
    @rdb.Client.GET_COL_INFO"
```

This problem has now been fixed.

5.4.6 TIMESTAMPADD Function with Milliseconds

Fixed Release 7.3.5.3

When using the extended function feature "{FN TIMESTAMPADD…}" in conjunction with the qualifier "SQL_TSI_FRAC_SECOND" the value provided was added as whole seconds rather than as millseconds resulting in an incorrect value.

This was fixed in the original Release 7.3.5.3 but the release note was omitted.

5.4.7 TIMESTAMPDIFF Behaviour is Now Settable

Fixed Release 7.3.5.3.4 Build 20210518

The extended function feature "{FN TIMESTAMPDIFF (TYPE, TS1, TS2) }" calculates the difference between two timestamps. By default the Oracle JDBC for Rdb drivers calculate the difference by subtracting TS2 from TS1. This however is different from other JDBC drivers that carry out the calculation by subtracting TS1 from TS2.

If your application requires the alternate behaviour you may use the following connection option:

```
   @REVERSETIMESTAMPDIFF=TRUE
```

See the Oracle JDBC for Rdb User Guide for more details on connection options.

▲contents

5.5 New Features for Release 7.3.5.2

This section describes new and changed features in Oracle JDBC for Rdb 7.3.5.2.

5.5.1 New Connection Options

Release 7.3.5.2

A previously undocumented Connection option has been introduced:

- suppressAddingDbkey

See the section *Connection Options* of the *Oracle JDBC for Rdb User Guide* for more details.

5.5.2 Changed Log File Names

Release 7.3.5.2

When a JDBC server is started by the Thin controller or as a SQL/Services JDBC dispatcher, the **RDB$JDBC_HOME: RDBJDBC_STARTSRV.COM** command procedure is used.

As well as setting up the JDBC server process this command procedure also sets the names of the various log files used during the server startup.

As it is possible that the same server names may be used by different nodes of the same cluster, the naming of the log files now include, by default, the nodename of the node they started on.

In addition a new pattern marker has been added to the LogFile pattern in the server configuration options allowing you to specify that the current node name should be used in the logfile name.

See the section *Logfile pattern* of the *Oracle JDBC for Rdb User Guide* for more details.

5.5.3 Server Log File Unlocked

Release 7.3.5.2

Prior to release 7.3.5.2, server log files were locked while the server was running. With release 7.3.5.2 the default behaviour is for the server log file to be readable while the server is running.

To unlock the log file the following two JAVA* logical names have been added to the standard server startup command procedure **RDB$JDBC_HOME:RDBJDBC_STARTSRV.COM :**

```
$    write x "$ DEFINE/USER JAVA$FILE_OPEN_MODE 3"
$    write x "$ DEFINE/USER JAVA$FSYNC_INTERVAL 5"
```

If you wish the server log files to remain locked while the server is running, comment out these two lines.

▲contents

5.6 Corrections in Release 7.3.5.2

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.5.2.

5.6.1 ACCVIO During MP Server Shutdown

Fixed in Release 7.3.5.2.
Also Fixed in Release 7.3.5.1.2 Build 20180427.

The shut-down of a MP Server instigated using the thincontroller or ORCM may cause an AccessViolation in the server process during the close-down procedure.

This problem may occur if, during the shutdown, there are active clients that still have connections to an Rdb database using the server.

An AccessViolation like the following may be seen in the server log file:

```
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=0000000000000020,
PC=FFFFFFFF844BE051, PS=0000001B
----- End of Exception message
                                          0 FFFFFFFF803A7692 FFFFFFFF803A7692
PTHREAD$RTL  THD_MUTEX  pthread_mutex_trylock
                                     229428 0000000000001CB1 FFFFFFFF844BE051
RDBJDBCMPSHR73  RDBJDBC_MP  mp_lock_mutex
                                      79708 0000000000007652 0000000004B33D52
RDBJDBCMPSHR73  RDBJDBC_MP  mp_notify    80060 0000000000008242 0000000004B34942
RDBJDBCMPSHR73  RDBJDBC  JNI_Disconnect
```

```
                                        95780 000000000010D332 0000000004CB9EF2
RDBJDBCMPSHR73  RDBJDBC  java_rdb_JNI_Disconnect
```

Or

```
OpenVMS stack trace:
#
# A fatal error has been detected by the Java Runtime Environment:
#
#  %SYSTEM-F-ACCVIO (0xc) at pc=884BC051, pid=1278, tid=244371968
#
# JRE version: Java(TM) SE Runtime Environment (8.0) (build 1.8.0.03-vms-rc1)
# Java VM: Java HotSpot(TM) 64-Bit Server VM (25.51-b02 mixed mode OpenVMS-ia64 )
# Problematic frame:
# C  [PTHREAD$RTL+0x1]
#
#
#
# An error report file with more information is saved as:
# /sys$scratch//hs_err_pid1278.log
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=000000000000002C,
PC=000000000110F160, PS=0000001B

  Improperly handled condition, image exit forced by last chance handler.
    Signal arguments:   Number = 0000000000000005
                        Name   = 000000000000000C
                                 0000000000000000
                                 000000000000002C
                                 000000000110F160
                                 000000000000001B
```

This problem only occurs with MP Servers with active client connections.

5.6.2 SQL-F-CONVARUND Raised on Multi-context Queries

BUG 28143654
Fixed in Release 7.3.5.2.
Also Fixed in Release 7.3.5.1.3 Build 20180615.

During the preconditioning of SQL queries for use within JDBC, the JDBC drivers may add extra SQL syntax to the provided query to return dbkeys for future potential record access by the driver.

A problem may occur when multiple table contexts are used within the query requiring the driver to add dbkey retrieval code for each of the table contexts used.

The driver may associate the wrong table context with the dbkey request which may in turn cause JDBC to issue the following exception when the query is compiled:

```
oracle.rdb.jdbc.common.RdbException: in <rdbjdbcsrv:prepare_stmt>
%SQL-F-CONVARUND, Column qualifier xxxxxxxx is not defined
```

An example of the type of query that may fail is shown below:

```
SELECT EMPLOYEES.EMPLOYEE_ID, S1.SALARY_START, S2.SALARY_START
FROM EMPLOYEES,
             SALARY_HISTORY S1,
             SALARY_HISTORY S2
WHERE EMPLOYEES.STATUS_CODE = '1'
AND S1.EMPLOYEE_ID = EMPLOYEES.EMPLOYEE_ID
AND S2.EMPLOYEE_ID = EMPLOYEES.EMPLOYEE_ID
AND S1.SALARY_START < S2.SALARY_START
AND S1.SALARY_AMOUNT > S2.SALARY_AMOUNT
```

The following workarounds may be used to prevent this problem from occurring:

a. Add the "{nodbk}" escape clause to the SQL text.

```
PreparedStatement ps = connection.prepareStatement(
"{nodbk}SELECT EMPLOYEES.EMPLOYEE_ID, S1.SALARY_START, " +
" S2.SALARY_START " +
" FROM EMPLOYEES, " +
"              SALARY_HISTORY S1, " +
"              SALARY_HISTORY S2 …
```

See the section *Extended SQL Syntax – {nodbk} escape* of the *Oracle JDBC for Rdb User Guide* for more details.

Or

b. Add the "@`suppressAddingDbkey=true`" connection string qualifier to your connection.

For example:

```
Connection conn = DriverManager.getConnection
("jdbc:rdbThin://bravo:1755/my_db_dir:pers" +
 "@suppressAddingDbkey=true", user, pass);
```

See the section *Connection Options* of the *Oracle JDBC for Rdb User Guide* for more details.

Both these workarounds suppress the addition of the extra dbkey retrieval syntax thus preventing the syntax problem.

5.6.3 Timestamps and Batched Statements

Fixed in Release 7.3.5.2.
Also Fixed in Release 7.3.5.1.4. Build 20180810.

A problem with the adjustments of timestamps relative to the local time zone may cause incorrect Date/Time results to be stored into the Rdb database.

The problem only occurs during Insert or Update operations using PreparedStatements that have been batched, for example:

```
…
  Timestamp ts = new Timestamp(date.getTime());
  ps = conn.prepareStatement("INSERT INTO TEST(TEST_DATE) VALUES (?)");
  ps.setTimestamp(1, ts);
  ps.addBatch();
  ps.executeBatch();
…
```

This has now been fixed.

5.6.4 Incorrect Termination Message on Server Shutdown

Fixed in Release 7.3.5.2.

During the normal closedown of a JDBC server several closing messages are written to the server log file.

Infrequently, the log termination message may be incorrect, and the closedown may take slightly longer to complete. The server closedown does complete correctly.

Log entries like the following may be seen at the end of the log file. These messages can be safely disregarded.

```
00000400> RDB$CLIENT_ID_00000001 C00000000*T 2018-04-25 21:30:10.572 :
closing msg : BroadcastThread
Server terminated due to inactivity
```

5.6.5 Manager Server Fails to Respond to Poll Request

Fixed in Release 7.3.5.2.

The JDBC Manager server may fail to respond to a multicast poll made from the thin controller or ORCM even though it is using the same multicast credentials as the poll requester.

In addition, the following message line may be seen in the server log of the JDBC manager server:

```
oracle.rdb.jdbc.srv.RdbManSrv cannot be cast to oracle.rdb.jdbc.srv.RdbDBSrv
```

5.6.6 SilentTruncate Not Enabled for Some Operations

Fixed in Release 7.3.5.2.

The use of the SILENTRUNCATE connection modifier to silently truncate string parameters prior to sending them to Rdb may not work in some circumstances.

In earlier versions of JDBC and Oracle Rdb, the following code example may retrieve the employee record with the employee_id of “00168” as silent truncate (which was the default behaviour) would truncate the source string “001689” to the size of the employee_id column (5 characters) prior to sending to Rdb for processing:

```
PreparedStatement p = conn.prepareStatement (
   "select last_name from employees  where employee_id =?");
p.setString( 1, "001689" ); // 1 longer than field - should find it
ResultSet r = p.executeQuery();
while (r.next())
    …
```

To comply with SQL Standards, more recent versions of Oracle JDBC for Rdb were changed to no longer carry out this type of silent truncation by default. However, to allow older applications to continue to work as expected if they relied on this implicit truncation, JDBC introduced the SILENTRUNCATE connection modifier.

A problem has been found in the handling of the silent truncation that may prevent the above example code working as expected when silent truncation is enabled. Current versions of JDBC may still send down the non-truncated form of the string, and as no records match the given key, the code above may return an empty record set.

JDBC has now been changed to allow silent truncation of parameters as in the above example if the SILENTRUNCATE connection modifier is set.

Note that as silent string truncation does not comply with SQL standards and the use of this feature may mask errors in your application code, Oracle strongly recommends to not use or rely on this feature.

5.7 New Features for Release 7.3.5.1

This section describes new and changed features in Oracle JDBC for Rdb 7.3.5.1.

5.7.1 New Pool Server Balancing Option PoolOrder

Release 7.3.5.1.
Also available in Release 7.3.5.0.5

JDBC now allows you to specify the new Pool Server Balancing option **PoolOrder**.

JDBC Pool Servers now have the option to use the pool order, i.e., the order in which the pooled servers where defined for the pool to determine which server to redirect to on the next client connection request.

The Pool Balancing option **PoolOrder** tells the pool server to select the first pooled server in the pool that has not yet reached its maximum client limit. Instead of using the default round-robin search it will now do a simple order search of the pool. The order of servers in the pool is the order of the pooled servers in that server's pool specified in the server's configuration file.

5.7.2 New Connection Options

Release 7.3.5.1.

Four new Connection options have been introduced:

- SilentTruncate
- rollbackOnDeadlock
- SQLDialect
- concurrentThread

See the section *Connection Options* of the *Oracle JDBC for Rdb User Guide* for more details.

▲contents

5.8 Corrections in Release 7.3.5.1

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.5.1.

5.8.1 Incorrect Date/time Values When TimeZones Mismatch

Fixed in Release 7.3.5.1.
Also fixed in Instance Build 20170824 Release 7.3.5.0.2.

Due to a problem introduced in release 7.3.4.2.2, JDBC fails to convert date/time values correctly while storing or retrieving data from the Rdb database if the connection attribute "`@timezone`" is used.

A similar problem also occurs when the default time zone of the client application does not match the time zone of the data already stored in the existing database.

There are several ways to change the application default time zone, including using Java system property `user.timezone`, the Java system method `TimeZone.setDefault()` and the `@timezone` connection attribute introduced in JDBC release 7.3.4.2.2.

Although time zone information is not stored in the Rdb database, time zone information obtained from the system is used by Rdb to carry out the conversion of date/time values to the internal numeric format that this data is actually stored as within the database.

In general the time zone used by Rdb is the time zone of the system that the database resides that has been set by using OpenVMS tools such as `SYS$MANAGER:UTC$TIME_SETUP.COM`.

A workaround for this problem is to ensure that the default time zone used by the client application matches the time zone of OpenVMS system the Rdb database resides, and to not use the "`@timezone`" connection attribute.

5.8.2 New Feature Omission - TimeZone Connection Attributes

Fixed in Release 7.3.5.1.
Also fixed in Instance Build 20170824 Release 7.3.5.0.2.

The 7.3.4.2.2. release of *Oracle JDBC for Rdb Release Notes* failed to include the new feature **TimeZone Connection Attributes**. The missing note can now be found in the *New Features for Release 7.3.4.2* section of these release notes.

5.8.3 Incorrect Parameter Initialization in RDBJDBC_EXECCLI.COM

Fixed in Release 7.3.5.1.
Also fixed in Instance Build 20170830 Release 7.3.5.0.3

A problem introduced in release 7.3.5 of Oracle JDBC for Rdb prevents CLI command from executing correctly.

Included in the ORCM installation is the command procedure RDB$JDBC_HOME:RDBJDBC_EXECCLI.COM. This command procedure is used to execute CLI commands by JDBC servers.

When attempting to execute a CLI command an error like the following will be logged:

```
%DCL-W-IVVERB, unrecognized command verb - check validity and
spelling
```

```
\OS$$\
```

In the body of the RDBJDBC_EXECCLI.COM command procedure, the standard DCL parameter “P2” was incorrectly re-initialized as a global symbol rather than a local symbol.

A workaround for this problem is to edit your RDB$JDBC_HOME:RDBJDBC_EXECCLI.COM command procedure and replace the line:

```
$ p2 :==
```

With

```
$p2 :=
```

The content of the RDBJDBC_EXECCLI.COM is show below with the line that should be replaced highlighted in yellow.

```
$ !   Copyright ▒ 2005, 2017, Oracle Corporation.  All Rights Reserved.
$ !
$ !  EXECUTE CLI COMMAND
$ !
$ ! p1 is the command proc that invoked this
$ ! p2 is the command
$ ! p3 thru p8 are the  parameters passed
$ SET MESSAGE /FACILITY /IDENTIFICATION /SEVERITY /TEXT
$ defRdbVersion :== 73
$ !
$ ! Check for onstart command
$ !
$ strt :=""f$extract(0,5,p2)"'
$ if strt.eqs."OS$$ "
$ then
$   onstrt :=""f$extract(5,f$length(p2),p2)"'
$   p2 :==
$   'onstrt'
$ endif
$! if Oracle Rdb/RMU not setup try version 'defRdbVersion'
$ if f$logical("RDB$JDBC_NORDBCHECK").nes."TRUE"
$ then
$   rvar :== 'f$logical("RDMS$RMU_VARIANT")
$   if rvar .eqs.""
$   then
$     @SYS$LIBRARY:RDB$SETVER 'defRdbVersion'
$   else
$     @SYS$LIBRARY:RDB$SETVER RESET
```

```
$   endif
$ endif
$ 'p2' 'p3' 'p4' 'p5' 'p6' 'p7' 'p8'
$ !
$ exit
```

5.8.4 Spurious Error Message with Stop Client

Fixed in Release 7.3.5.1.
Also fixed in Instance Build 20171006 Release 7.3.5.0.4

When using the Thin Controller command STOP CLIENT IN like the following:

```
STOP CLIENT IN MY_DIR:MY_DATABASE
```

an error message is raised by the JDBC server like:

```
Error executing command 'stop client in DIR:MY_DATABASE
<1> Io exception : No such client,  RDB$CLIENT_ID : DIR:MY_DATABASE
```

This error message is spurious, the command completes successfully, and the server will stop all clients bound to that database.

5.8.5 ORCM Async External Tool Command Failure

Fixed in Release 7.3.5.1.
Also fixed in Instance Build 20171030 Release 7.3.5.0.4

When using ORCM external tools that have the “continuous” flags set, ORCM builds and executes an asynchronous connection to the recipient server which executes the required command and remains listening on the server’s output socket for results of the command execution.

If the recipient server is a JDBC server and during the asynchronous command execution operation this server receives a connect request from either the thin controller or from an ORCM session, the asynchronous command will be incorrectly cancelled by the server and the ORCM results display for the command will hang.

JDBC has now been changed to prevent this incorrect command cancellation and resultant output freezing.

5.8.6 Incorrect Seconds When Date Prior to Java Epoch

Fixed in Release 7.3.5.1.
Also fixed in Instance Build 20171107 Release 7.3.5.0.4

A problem in the internal handling of fractions of seconds may cause incorrect datetime values to be stored or retrieved from the database.

This problem only occurs when the date/time value is before the Java epoch date 1st January 1970, and the fractions of seconds in the time value is non-zero. The integral "seconds" value of the time stored will be incorrectly incremented by 1 second.

A workaround for this problem is to use date/time values with an integral number of seconds (no fractions).

5.8.7 Multithread Problem with DatabaseMetaData

Fixed in Release 7.3.5.1.
Also fixed in Instance Build 20171124 Release 7.3.5.0.5

A problem has been found in the thread handling of the **DatabaseMetaData** class methods. Some of the methods in this class were found not to be thread safe.

Applications using multiple concurrent threads accessing the same **DatabaseMetaData** object may find that one or more of the threads may fail with an **EOFException** raised.

JDBC has now changed how **DatabaseMetaData** methods are executed to improve thread safety.

5.8.8 NLSLANG Problems

Fixed in Release 7.3.5.1.
Also fixed in Instance Build 20171124 Release 7.3.5.0.5.

The contents of the comment (description) attribute of Rdb Table and Columns may not be displayed correctly in third party client applications if the connection attribute "@NLSLANG" has been used.

JDBC fails to correctly convert the Rdb segmented strings (Blobs) that hold the description information, and the value of the description will be left as a reference to the segmented string dbkey rather than the textual value of the description.

In addition, if the NLSLANG is set to a character set that has multi-octet characters, literal values embedded in the SQL text to be compiled by **Statements** and

PreparedStatements may not be converted correctly leading to failures when the literals are used in selection and other operations by SQL.

For example, when using "@NLSLANG=UTF8"

```
select decode(a.sex,_dec_mcs'M','男',_dec_mcs'F','女')  from employees;
```

may not provide the expected output.

5.8.9 RDBJDBC_STARTSRV.COM Problem

Fixed in Release 7.3.5.1.
Also fixed in Instance Build 20171205 Release 7.3.5.0.6.

The command procedure RDBJDBC_STARTSRV.COM is used to start JDBC servers by several components of Oracle JDBC for Rdb including the thin controller and SQL/Services JDBC dispatcher start up.

In release 7.3.5 this command procedure was changed to automatically select a Java version to use if it found that none had been setup prior to its execution.

A typographic error made during the modification of RDBJDBC_STARTSRV.COM may prevent the procedure from correctly determining that Java 8 is available on your system.

A workaround for this problem is to alter the RDBJDBC_STARTSRV.COM file in your RDB$JDBC_HOME directory to fix this typographic error.

Locate the highlighted line as shown below (in Yellow) and remove the "@" character from the start of the filename.

```
.
.
.
$! PARAM1 ... PARAM8  will be passed directly to the thinsrv
$! PARAM_TYPE = server type
$! PARAM_LOG = optional logfile
$! PARAM_CFG = optional configfile
$! PARAM_STARTUP = additional setup command
$! PARAM_COPYLN = if true JDBC environment logical names should be copied from caller
$! PARAM_RDB_VERS = if not null the Rdb version to set
$! PARAM_JAVA_VERS = if not null the Java environment to set
$!
$!
$! the following symbols set the def version to use
$!
$    defRdbVersion = 73
$    defJavaSetup = "@SYS$COMMON:[JAVA$80.COM]JAVA$SETUP.COM
$    if "''PARAM_JAVA_VERS'" .nes. ""
$    then
$       defJavaVersion = 'PARAM_JAVA_VERS'
$    else
```

```
$       if f$getsyi("arch_name") .eqs. "IA64"
$       then
$!      see if Java 8 present
$          if f$search(defJAVASetup) .nes. ""
$          then
$            defJAVAVersion = 80
$          else
$            defJAVAVersion = 60
$          endif
$       else
$          defJAVAVersion = 150
$       endif
$    endif.
.
.
.
```

In other words, replace

```
$    defJAVASetup = "@SYS$COMMON:[JAVA$80.COM]JAVA$SETUP.COM
```

with

```
$    defJAVASetup = "SYS$COMMON:[JAVA$80.COM]JAVA$SETUP.COM
```

5.8.10 More Incorrect Seconds When Date Prior to Java Epoch

Fixed in Release 7.3.5.1.
Also fixed in Instance Build 20180112 Release 7.3.5.0.8.

In addition to the problems found and fixed as described in section 3.9 of these release notes, further problems have been noted when dates prior to the Java epoch are used that have time precision greater than milliseconds. Datetime values fitting this description may have their "seconds" values incorrectly incremented by 1 resulting in incorrect datetime values being stored to the Rdb database.

5.8.11 SQL92 Dialect Semantics

Fixed in Release 7.3.5.1.

The documentation for Oracle JDBC for Rdb state that the JDBC servers and Drivers adhere to SQL92 Dialect semantics as described in the Oracle Rdb SQL documentation.

In prior versions of JDBC, this was true for most of the SQL92 semantics described with one exception. In a standard connection using either a native or a thin JDBC driver, the string truncation behavior described for SQL92 in the SET DIALECT section of the *Oracle Rdb SQL Reference Manual*, was not followed. By default, string truncation in prior versions of JDBC follows SQL89 semantics.

Starting with JDBC Release 7.3.5.1 JDBC now ensures that, by default, the SQL92 dialect has been selected for all connections by automatically issuing a SET DIALECT statement after database bind.

A partial workaround for prior versions of JDBC is to execute a SET DIALECT 'SQL92' statement prior to executing other SQL statements in JDBC:

```
Statement stmt = connection.createStatement();
stmt.execute("SET DIALECT 'SQL92'");
```

The workaround will enforce SQL92 string truncation semantics for most SQL statements except for the case of parameters used in PreparedStatements. See the note below.

Note:

A side-effect of setting the SQL Dialect to "SQL92" or other dialect values that may also follow SQL92 string truncation semantics is that Oracle Rdb will now throw an exception on truncations of input parameters in PreparedStatements even if the statement is a select.

For example:

```
PreparedStatement ps = connection.prepareStatement(
  "select * from employees where employee_id = ? ");
ps.setString("001649");
ps.execute("SET DIALECT 'SQL92'");
```

The above example will throw a **TRUN_STORE** exception as the value set for **employee_id** is one character greater than the expected 5 characters in the Rdb demonstration table **employees.**

In JDBC Releases prior to 7.3.5.1, this exception will not be raised even if you have explicitly set the SQL Dialect to SQL92 as JDBC will still automatically truncate parameter string values to the size expected by Rdb for that parameter. Thus, some string values will still be silently truncated when using PreparedStatements.

In JDBC Release 7.3.5.1 or later, silent truncation is NOT carried out, thus by default, SQL statement as in the example above will raise the **TRUN_STORE** exception.

You may prevent this error from being raised by using the connection string modifier "**silentTruncate=true**".

See the section *Connection Options* of the *Oracle JDBC for Rdb User Guide* for more details.

5.8.12 Unreferenced CALL Parameters

Fixed in Release 7.3.5.1.

Trying to set the value of a CALL parameter that does not have a corresponding reference within the underlying called module, results in JDBC throwing the following exception:

```
Internal error: Invalid index for data access
```

The following code snippet shows an example of the type of scenario that may raise this exception:

```
stmt.executeUpdate(" create module p_mod language sql " +
  " procedure p_get(in :id_in varchar(10), out :b integer); " +
  " select count (*) into :b from employees; end module; ");
stmt = conn.prepareCall("call p_get(?,?)" );
stmt.registerOutParameter( 2, Types.INTEGER );
stmt.setString(1,"T");
```

In this example the procedure **p_get** declares the in parameter **:id_in,** however the body of the procedure contains no reference to that parameter.

In prior versions of JDBC the invocation of the code line

```
stmt.setString(1,"T");
```

would raise the exception as described above.

Starting with JDBC Release 7.3.5.1, JDBC will no longer raise this exception. Instead, a warning will be added to the statement's SQLWarnings list:

```
Call Parameter xxxxx not used.
```

You may retrieve the current set of statement warnings by calling the **getWarnings()** method.

The **Statement.getWarnings()** method retrieves the first warning reported by calls on this **Statement** object. Subsequent **Statement** object warnings will be chained to this **SQLWarning** object.

5.8.13 ACCVIO When Using CREATE or DROP DATABASE

Fixed in Release 7.3.5.1.

The following Access Violation may be raised when trying to create or drop a database using a JDBC statement:

```
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual
address=0000000000000002
```

This has now been fixed.

5.8.14 IF EXISTS Trimmed Off DROP Statements

Fixed in Release 7.3.5.1.

The SQL text for Statements containing the DROP clause was incorrectly modified by JDBC to remove any “IF EXISTS” clause that may be present.

This has now been fixed.

5.8.15 NOSUCHCUR Error When Using ORDP

Fixed in Release 7.3.5.1.

A problem introduced in JDBC Release 7.3.5 may cause ORDP queries to fail during the closing of a Rdb-based DataAdapter.

When JDBC connectivity is being used in conjunction with an RdbDataAdapter, the close-down of the RdbDataAdapter may cause the following exception to be raised:

```
Oracle.DataAccess.RdbClient.RdbException : in <rdbjdbcsrv:close_cursor>%SQL-F-NOSUCHCUR,
Cursor CC_00000002
```

5.8.16 Concurrent Threads in Single Connection

Fixed in Release 7.3.5.1.

Although JDBC is thread-safe when concurrent threads use separate connections, several optimization features introduced into JDBC drivers over the last few releases may cause threading issues when concurrent threads are using the same JDBC connection.

Concurrent thread access of a single JDBC connection may fail with several symptoms including Access Violations and prematurely closed connections.

Third-party or client applications using JDBC drivers may carry out concurrent threading on connections to optimize data retrieval. When these applications are used with Oracle JDBC for Rdb drivers, it is possible that operations may fail due to threading issues.

To allow these applications to work correctly with JDBC, a new connection string option "**concurrentThread**" has been introduced.

If your application intends to carry out concurrent thread operations on the same JDBC connection, this new connection option may be used to inform JDBC of the potential of concurrent multithread operations.

For example:

```
Connection conn = DriverManager.getConnection(
"jdbc:rdbThin://bravo:1755/my_db_dir:pers"+
"@concurrentThread=true", user, pass);
```

A workaround, if it is available, is to use your application configuration settings to disable concurrent threads from accessing the same connection object.

See the *Oracle JDBC for Rdb User Guide* for more details on connection options.

▲contents

5.9 New Features for Release 7.3.5

This section describes new and changed features in Oracle JDBC for Rdb 7.3.5.

(Also see section *Enhancements Provided in Oracle JDBC for Rdb Release 7.3.5.1* for details of any enhancements that may be included in patch releases of 7.3.5).

5.9.1 Java 8.0 Support on OpenVMS IA-64 Systems

Release 7.3.5.

Oracle JDBC for Rdb now supports Java 8.0 on OpenVMS IA-64 systems.

To comply with OpenVMS Java 8.0 requirements JDBC now provides 64Bit version of the three JDBC shareable images:

- RDBJDBC6473.EXE
- RDBJDBCMP6473.EXE

- RDBJDBCEXEC6473.EXE

These shared images are only used on OpenVMS IA-64 systems where Java 8.0 (or later) has already been installed.

They provide the required 64Bit JNI access that is the bridge between your Java applications and the underlying Rdb database system when using Java 8.0 or higher on OpenVMS.

JDBC will automatically select the correct shared images to use depending on the current Java version setup in the environment you start the JDBC servers within, or your default Java environment for applications using the Rdb Native driver.

On OpenVMS IA-64 systems the JDBC installation will also install variants of the RdbNative and RdbThin driver JAR files. See Contents of the Oracle JDBC for Rdb Kit and Installation Procedure for more details.

5.9.2 Versioned JDBC Driver Jars

Release 7.3.5.

Starting with release 7.3.5 Oracle JDBC for Rdb ships with a larger number of jar files allowing for better multi-version support of Java and JDBC releases.

Versioned variants of the RdbThin and RdbNative JAR files are provided to allow for version specific support for JDK 5 (JDBC 3.0), JDK 6 (JDBC 4.0) and JDK 8 (JDBC 4.2).

See the section *Contents of the Oracle JDBC Rdb Kit* in the *Oracle JDBC for Rdb User Guide* for more information on these new JAR files.

In addition, to allow you to double check the JDBC version information associated with the contents of a specific JDBC jar file, each JDBC jar now ships with an additional class "`rdb.JDBCVersion`". This class may be used to determine the JDBC version information of the JAR file.

As this class exposes a "`main`" method, it may be used on the command line to show the JDBC version information for the selected JAR file.

For Example:

```
$java -cp rdbthin.jar "rdb.JDBCVersion"
Title       : Oracle JDBC for Rdb Thin Driver
Vendor      : Oracle Corporation
Version     : Release 7.3.5.0.0 20170529 BH5T
```

```
Load Path   : file:/JMURRAY_USER/murray/jdbc/test/i73/rdbthin.jar
Supports    : JDK 5
JVM         : 1.6.0 Java HotSpot(TM) 64-Bit Server VM
```

Or

```
$@sys$common:[java$80.com]java$setup
$java -cp rdbthinV8.jar "rdb.JDBCVersion"
Title       : Oracle JDBC for Rdb Thin Driver
Vendor      : Oracle Corporation
Version     : Release 7.3.5.0.0 20170529 BH5T
Load Path   : file:/JMURRAY_USER/murray/jdbc/test/i73/rdbthinv8.jar
Supports    : JDK 8
JVM         : 1.8.0.03-OpenVMS Java HotSpot(TM) 64-Bit Server VM
```

This information is also contained the the JAR Manifest.

5.9.3 Change in SQL/Services JDBC Dispatcher Naming

Release 7.3.5.

The process to resolve JDBC server charactistics for the JDBC server started during SQL/Services JDBC Dispatcher startup has been enhanced to include a simplified server characteristics defaulting mechanism.

The SQL/Services dispatcher name now can be used as the JDBC server name. This name in turn may be used to obtain server characteristics from the default XML-based JDBC configuration files found on your system.

The chapter *Oracle SQL/Services and Oracle JDBC for Rdb Servers* in the *Oracle JDBC for Rdb User Guide* has been changed to describe this mechanism.

The use of the dispatcher name as the server name is now the default behaviour. If your existing system relies on the old behaviour of defaulting to the JDBC server name of SQS<port>, you may reset this naming behaviour back by issuing the following DCL command:

```
$ deassign/system/exec RDB$JDBC_SQS_USE_DISPNAME
```

You can still tailor the JDBC server characteristics by using the `RDB$JDBC_SQS*` logical names, however Oracle recommends using the XML-based JDBC configuration files to set JDBC server characteristics.

See the *Oracle JDBC for Rdb User Guide* for more details.

5.9.4 Changes to SHIFT_JIS Java Encoding

Release 7.3.5.

Prior to release 7.3.5 JDBC used the Java “SJIS” character set encoding tables to carry out the encoding and decoding of characters held in Rdb variables and columns that had the Rdb character set “SHIFT_JIS” assigned to them.

The “SJIS” encoding tables used did not convert characters that are considered extensions to the standard ***SHIFT JIS*** encodings such as those found in the Windows Code Page 932.

Note:

> **Windows code pages** are sets of characters (also called character encodings) used in Microsoft Windows.

As it is expected that most ***SHIFT JIS*** users of JDBC will require the support of the the ***SHIFT JIS*** extension characters found in the Windows 932 code page, the encoding tables used internally by JDBC have now been changed to use the Java character set “MS932” instead of “SJIS”.

See the Java internationalization documentation for more information on the Java character sets “SJIS” and “MS932”.

5.10 Corrections in Release 7.3.5

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.5.

5.10.1 Small Memory Leak With GetTables()

Fixed in Instance Build 20161125 release 7.3.5.

A small memory leak may be seen when calling DatabaseMetaData.GetTables().

This has now been fixed.

5.10.2 DEFAULTSSL Definition Not Inherited

Fixed in Instance Build 20170714 Release 7.3.5.

The SSL attributes associated with the DEFAULTSSL server definition within the XML-based configuration file were not correctly inherited by SSL-type server definitions.

JDBC failed to correctly establish SSL credentials for the server since SSL attributes were not correctly copied from the DEFAULTSSL definition, which will subsequently prevent the server from starting correctly.

The server may terminate with the following exception:

```
Failed I/O:javax.net.ssl.SSLException: No available
certificate or key corresponds to the SSL cipher suites
which are enabled.
```

A workaround is to place the SSL attributes directly into the specific server's definition rather than relying on inheritance from the DEFAULTSSL server.

5.10.3 Incorrect Value for Client and Executor Free Shared Memory

Fixed in Instance Build 20170810 Release 7.3.5.

When you issue a `SHOW CLIENTS` or `SHOW EXECUTORS` command in the ThinController, information about the connected clients and executors is displayed.

If the connected server is one of the MP type servers, shared memory statistics will also be displayed.

The `SHOW SERVER, SHOW CLIENTS` and `SHOW EXECUTORS` information for an MP Server will show the amount of global shared memory that is currently free for the server, client or executor respectively, for example:

```
rdbthincontrol> show executors
//localhost:1799/ is currently running 1  executor

RDB$EXECUTOR_NAME       : JDBCDEV00000001
RDB$PID                 : 0x22A7DF71(581427057)
RDB$STATE               : Client 00000004 connected
raid1:[jdbc.regtest.721]mf_personnel as murray
RDB$NUM_STMTS           : 1
RDB$NUM_CURSORS         : 0
RDB$MEM_SQLDAS          : 139496
RDB$MEM_BUFFERS         : 0
RDB$MEM_SEG_STRINGS     : 0
RDB$MEM_NUM_FREE_CHUNKS : 2
RDB$MEM_LARGEST_CHUNK   : 165008
RDB$MEM_TOTAL_FREE      : 165800
RDB$MEM_PAGEFILE_QUOTA  : 718750
RDB$MEM_PAGEFILE_COUNT  : 576751
RDB$MEM_GLOBPAGE_COUNT  : 945
RDB$MEM_WS_COUNT        : 2070
RDB$MEM_PAGETBL_COUNT   : 0
RDB$MEM_PAGE_FAULTS     : 3987
```

```
RDB$MEM_CPU_TIME        : 55
rdbthincontrol>
```

For both `SHOW CLIENTS` and `SHOW EXECUTORS` the total free global shared memory is shown after the **RDB$MEM_TOTAL_FREE** label. For SHOW SERVER it is shown after the **RDB$FREE_SHARED_MEMORY** label.

These values may also be seen in ORCM within the appropriate SERVER or CLIENT displays.

A problem in the calculation of this value may cause MP Server to return total free global shared memory with a higher than actual value.

The problem only affects the SHOW CLIENT and SHOW EXECUTOR displays; the SHOW SERVER display will show the correct shared memory values.

This is a MP Server problem only.

5.11 New Features for Release 7.3.4.2

This section describes new and changed features in Oracle JDBC for Rdb 7.3.4.2.

(Also see section Enhancements Provided in Oracle JDBC for Rdb Release 7.3.5 for details of any enhancements that may be included in patch releases of 7.3.4.2).

5.11.1 Support for Oracle Rdb Second Password Option

Release 7.3.4.2.

Since Release 7.3.2 of Oracle Rdb, database connection authorization may require the use of a second password. (See the *Oracle Rdb Release Notes* for release 7.3.2 for more details.)

JDBC has now been enhanced to allow the presence of a second authorization password during database connection setup.

See the section *Oracle Rdb Second Password* under *Configuration Options* within the *Oracle JDBC for Rdb User Guide* for more details.

5.11.2 New srv.usecreprc Configuration Option

Release 7.3.4.2.
Also available in Release 7.3.4.1.2.

The MP Server may now be configured to use detached processes when starting executors. This is similar to how executors are started by the server when PERSONA is used.

See *Executor Process Startup* in the *Oracle JDBC for Rdb Multi-Process Server* section of the *Oracle JDBC for Rdb User Guide* for more details.

5.11.3 New simplepn Connection Option

Release 7.3.4.2 Build 20160902.

During the initial stages of database connection many applications interrogate the **DatabaseMetaData** to determine the name of the underlying database product.

By default, from release 7.3.4 onwards Oracle Rdb JDBC servers return the value "Oracle Rdb" for the database product name. Earlier versions of the servers will return the simplified name "Rdb".

The presence of the word "Oracle" in the product name may cause some applications to incorrectly assume that the underlying database is fully Oracle RDBMS compliant and may send SQL statements to Oracle Rdb that are not syntactically correct for Oracle Rdb.

A new connection option "**simplepn**" is now available. By setting this connection option to true, the JDBC servers will return "Rdb" instead of "Oracle Rdb" when interrogated for the database product name.

For example:

```
Connection conn = DriverManager.getConnection(
    "jdbc:rdbThin://bravo:1755/my_db_dir:pers.rdb@simplepn=true",
    user, pass);
```

You could also place the connection option in the info Properties object passed to the get Connection method.

See the *Oracle JDBC for Rdb User Guide* for more details on connection options.

5.11.4 TimeZone Connection Attributes

Release 7.3.4.2 Build 20160902.

The connection attribute "**timezone**" has been added to allow you to specify the time zone to use when formatting and interpreting date/time strings.

For example:

```
Connection conn = DriverManager.getConnection(
"jdbc:rdbThin://bravo:1755/my_db_dir:pers"+
"@timezone=Europe/Paris", user, pass);
```

Previously, JDBC interpreted the values of Oracle Rdb date/time variables and columns to be relative to the OpenVMS date zero but using the default time zone of the Java VM the client driver was running under.

This meant that to correctly display or insert date/time data to and from an Rdb database, the client application had to ensure that the default time zone set for the application at the time of JDBC Statement execution matched the expected time zone of the Rdb database data. If the time zones did not match, the data stored or retrieved may be misintrepeted by the application or other applications using the same database data.

For various reasons an application may wish to interpret or display date/time data using a time zone that differs from the timezone that was used when the Rdb database data was inserted.

JDBC now allows the specification of the time zone to be used for the Rdb data, i.e. how to interpret the date/time data stored within the database. This value may differ from the default time zone used by the application.

The new connection attribute **sessiontimezone** is now available to tell JDBC the database time zone to use.

See the *Oracle JDBC for Rdb User Guide* for more details on connection options.

▲contents

5.12 Corrections in Release 7.3.4.2

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.4.2.

5.12.1 Incorrect Number Of Clients in Show Server Pool

Fixed in Release 7.3.4.2 Build 20160810.

The thincontroller **Show Server Pool** command provides information about the pool of servers that a Pool Server controls. Part of this information is the count of clients that are currently connected to each pooled server.

A problem introduced in the 7.3.4 release of Oracle JDBC for Rdb causes the results of this command to display an incorrect value for the number of current clients. The value is incorrectly decremented by 1.

5.12.2 Server Incompatibility With Early Driver Versions

Fixed in Release 7.3.4.2 Build 20160926.

A problem has been found when using Thin Servers release 7.3 and later with earlier releases of the Rdb thin client drivers. Some statement release operations may not correctly release all resources associated with the SQL statement eventually leading to memory problems on the JDBC server.

This problem does not prevent the drivers from working normally, however the eventual depletion of memory on the server may lead to COSI-VASFULL errors.

Although not documented nor supported as a feature, Oracle JDBC for Rdb attempts to maintain upward and downward compatibility between its servers and the Rdb Thin drivers. However, when upgrading Oracle JDBC for Rdb, Oracle recommends that all components on both the server systems and all client systems should be updated to the same version at the same time.

5.13 New Features for Release 7.3.4.1

This section describes new and changed features in Oracle JDBC for Rdb 7.3.4.1.

5.13.1 New Transaction Option AUTOFETCH

Release 7.3.4.1.
Also available in Release 7.3.4.0.4.

When AUTOCOMMIT is enabled, JDBC will automatically commit transactions at the end of the execution of a statement.

Sometimes during long-duration client-side read operations it may be advantageous to commit the current transaction more frequently, to prevent the read-only transaction from locking out other database operations.

For example, when using *SQLDeveloper* to read through Rdb records using Oracle JDBC for Rdb, records are delivered to the client application in groups, where the number of records in a record group depends on the FETCHSIZE specified for the statement or session. As you scroll down a list of records, the application may request the next group of records from the server. This may mean that if you have not finished scrolling through all the records in the retrieval set, there may still be a transaction outstanding on the server waiting for the rest of the records to be requested. If you leave this browse window open for an extended duration, this could cause Read-Only transaction lockouts on the database.

JDBC now allows you to specify that transactions should be auto-committed each time the server sends a FETCHSIZE number of records back to the client. This means that while the client is browsing the current group of records, that no transaction will be outstanding on the server.

A new connection string option "@TRANSACTION=AUTOFETCH" may be used to inform JDBC that auto-commit is required each time the server sends FETCHSIZE groups of records.

In addition, the server configuration option "TRANSACTION=AUTOFETCH" is also available to set the default transaction action for all connections that are made to that server.

A second connection string option "@TRANSACTION=AUTOREADFETCH" is also available and can be used to inform JDBC that, like the AUTOFETCH option, auto-commit is required each time the server sends FETCHSIZE groups of records, but when establishing a new transaction, if not otherwise explicitly specified, a Read-Only transaction should be started.

See the *Oracle JDBC for Rdb User Guide* for information on FETCHSIZE and how to specify the number of records to retrieve during client/server fetch operations. The application you are using may also have documentation on how to specify the FETCHSIZE when using JDBC drivers.

5.14 Corrections in Release 7.3.4.1

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.4.1.

(Also see section Corrections in Release 7.3.4.2 for details of any fixes that may be included in patch releases 7.3.4.1.2 through 7.3.4.1.2).

5.14.1 Pool Servers Not Working

Fixed in Release 7.3.4.1 Build 20160205.
Also fixed in Release 7.3.4.0.2 Build 20150626.

A problem introduced in release 7.3.4 prevents pool servers from correctly passing on client connection requests to the chosen pooled server.

The client application may receive an **EOFException** when trying to connect to Rdb using a pool server.

This has now been fixed.

5.14.2 Problem with MultiProcess Connection Option

Fixed in Release 7.3.4.1 Build 20160205.
Also fixed in Release 7.3.4.0.3 Build 20150825.

A problem introduced in release 7.3.4 prevents the use of the `@multiProcess` connection option when used in conjunction with the JDBC Native driver.

The following exception will be raised when the @multiProcess option is used:

```
java.lang.NullPointerException
        at oracle.rdb.jdbc.util.Executor.runFromJAVA(Executor.java:155)
        at oracle.rdb.jdbc.common.NativeConnect.connect(NativeConnect.java:284)
        at oracle.rdb.jdbc.common.NativeConnect.connect(NativeConnect.java:94)
```

This has now been fixed.

5.14.3 Bumpy-case Connection String Options Used with Properties.

Fixed in Release 7.3.4.1 Build 20160205.
Also fixed in Release 7.3.4.0.3 Build 20150825.

As documented in the *Oracle JDBC for Rdb User Guide*, some connection string options may contain both upper- and lower-cased characters in their names (see section *Connection Options* in the *Oracle JDBC for Rdb User Guide*).

A problem exists when using an option that has at least one upper-cased character in its name. If the option is used within a Properties object passed to the **DriverManager.getConnection()** method, JDBC may fail to recognize the option and silently ignore it. For example, the option *multiProcess* used within a JDBC Native driver connection may fail to be recognized if used in Properties:

```
String personnel="jdbc:rdbNative:personnel$db";
   Properties props=new Properties();
props.put("user",user);
props.put("password",pass);
props.put("multiProcess","true");
Connection conn=DriverManager.getConnection(personnel, props);
```

A workaround is to use the option directly on the URL connection string:

```
String personnel="jdbc:rdbNative:personnel$db@multiProcess=true";
Connection conn=DriverManager.getConnection(personnel,
                                            user,pass);
```

5.14.4 Further Blob Problems with Delimited Identifiers

Fixed in Release 7.3.4.1 Build 20160205.
Also fixed in Release 7.3.4.0.3 Build 20150827.

The problem as discussed in release note *3.4 Blob Problem with Delimited Identifiers* above was not completely solved. It is possible that problems may still exist when blobs take part in an Insert statement.

If the table name associated with a segmented string referenced in the text of an INSERT statement is not fully uppercased and no delimited identifiers are used in the statement then JDBC will fail with a syntax error:

```
oracle.rdb.jdbc.common.RdbException: Invalid column name
```

In addition, due to a problem introduced in release 7.3.4 and release 7.3.3.2.5. it is possible that simple selection of segmented strings from Oracle Rdb tables may fail with an exception similar to:

```
%SQL-F-RELNOTDEF, Table xxxxxxx is not defined in database or schema
```

This has now been fixed.

5.14.5 Access Violation at RDBJDBCMPSHR73+00227E31

BUG 22233633
Fixed in Release 7.3.4.1 Build 20160205.
Also fixed in Release 7.3.4.0.5 Build 20151206.

During the run down of executors by the MP Server it is possible that an internal error in memory handling may cause an Access Violation within the MP Server process like the following:

```
%SYSTEM-F-ACCVIO, access violation, reason mask=04,
virtual address=00000000000000CA, PC=FFFFFFFF904ED491,
PS=0000001B
```

If a stack trace is present or a system dump analysis of the errant process is done, it may show reference to a PC at or near the following location:

```
RDBJDBCMPSHR73+00227E31
```

(This PC offset may differ depending on the version of JDBC installed; the value above was seen when running Oracle JDBC for Rdb release 7.3.4 kit 0703-4V0F6P)

This problem appears to occur only on Itanium systems and may occur when the server node is very busy and is more likely to occur during the time an executor process is being run-down.

Although this is the primary access violation, due to an error in the exception frame handling by Java, the server process may also encounter a rapid iteration of access violations like:

```
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual
address=00000000000000CA, PC=FFFFFFFF904ED491, PS=0000001B
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000000, PC=FFFFFFFF806BC5D0, PS=0000001B
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0300B7590200BB2A, PC=00000000011D9820, PS=0000001B
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000000, PC=FFFFFFFF806BC5D0, PS=0000001B
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000000, PC=FFFFFFFF806BC5D0, PS=0000001B
.
.
.
```

HPE OpenVMS engineering are aware of this Java exception handling looping problem and are currently investigating it.

5.15 New Features for Release 7.3.4

This section describes new and changed features in Oracle JDBC for Rdb 7.3.4.

5.15.1 Create / Drop Database Support

Release 7.3.4.

Starting with release 7.3.4 of Oracle JDBC for Rdb, the JDBC drivers and servers support the ability to create and drop databases.

JDBC will accept the standard Oracle Rdb SQL syntax for **create** and **drop database,** however**,** syntactical keywords must not be abbreviated.

In addition the statement may also contain the clauses "**IF EXISTS**" or "**IF NOT EXISTS**" to conditionalize the operation.

By default, JDBC servers will not allow CREATE or DROP database statements, the server configuration must be changed to enable these operations.

See the *Create and Drop Database* section in the *Oracle JDBC for Rdb User Guide* for more information.

▲contents

5.16 Corrections in Release 7.3.4

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.4.

(Also see section Problems Corrected in release 7.3.4.1 for details of any fixes that may be included in patch releases 7.3.4.0.2 through 7.3.4.0.5).

5.16.1 Like Escape Throws OutOfBounds Exception

Fixed in Release 7.3.4 Build 20150424.
Also fixed in Release 7.3.3.2.2.

The use of the escape clause within a SQL statement containing the **like** operator may cause the JDBC driver to throw an OutOfBounds exception.

The problem only occurs if the escape character chosen is the same default escape character used by Java in string literals, the backslash character "\", for example:

```
select 'not ok' from rdb$database where 'a%a' like '%\%%' escape '\'
```

A workaround is to use a character other than backslash for the escape character:

```
select 'ok' from rdb$database where 'a%a' like '%|%%' escape '|'
```

This has now been fixed.

5.16.2 Correlation Name Workaround for Blobs Fails

Fixed in Release 7.3.4 Build 20150424.
Also fixed in Release 7.3.3.2.3.

A problem introduced prior to Release 7.3.3 of Oracle JDBC for Rdb prevents the workaround as described in the *Known Problems and Workarounds* section Blob Columns and Correlation Names from working.

This has now been fixed.

5.16.3 PreparedStatement Cache Failure

Fixed in Release 7.3.4 Build 20150424.
Also fixed in Release 7.3.3.2.4.

PreparedStatement handles can be cached by using the `sqlcacheps` connection property (See *Caching Statement Handles* in the *Oracle JDBC for Rdb User Guide*).

If statement handle caching has been enabled, a problem with return parameter handling may cause a cached PreparedStatement to fail during execution with the following exception:

```
oracle.rdb.jdbc.common.RdbException: Missing IN or OUT parameter at index:  n
```

Where the index 'n' shown is a number 1 greater than the actual number of parameter markers in the compiled statement.

This has now been fixed.

5.16.4 Blob Problem with Delimited Identifiers

Fixed in Release 7.3.4 Build 20150424.
Also fixed in Release 7.3.3.2.5.

A problem in the production of SQL statements during the processing of blobs internally by JDBC may cause a SQL syntax exception to be raised.

When storing or updating blobs JDBC must create SQL statements to handle the Rdb segmented string operations. If the table name or column name associated with the blob field requires to be handled as a delimited identifier, JDBC fails to recognize this, and generates SQL text that is not syntactically correct.

▲contents

5.17 New Features for Release 7.3.3.2

This section describes new and changed features in Oracle JDBC for Rdb 7.3.3.2.

5.17.1 Change Transaction Start Time.

Build 20141215 release 7.3.3.2.2.

By default, when a new connection is made, JDBC will immediately start a `READ_ONLY` transaction in anticipation of subsequent read-only operations.

The type of the default transaction may be changed to `READ_WRITE` using connection string options.

The start of the default transaction may now be deferred by using the new connection string transaction options, READONLYDEFER and READWRITEDEFER. See the section *Connection Options* in the *Oracle JDBC for Rdb User Guide* for more details.

▲contents

5.18 Corrections in Release 7.3.3.2

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.3.2.

(Also see Corrections in Release 7.3.4 section for details of any fixes that may be included in patch releases 7.3.3.2.2 through 7.3.3.2.5).

5.18.1 NullPointerException When Allowing Anonymous

Fixed in Release 7.3.3.2 Build 20140905.
Also fixed in Release 7.3.3.1.2.

When anonymous access to a JDBC server has been enabled, the server should allow clients to use a blank username to access the Rdb database anonymously.

A problem introduced during the addition of user restriction features will cause the server to throw a **NullPointerException**, if the server has anonymous access enabled and the client provides a blank username.

This has now been fixed.

5.18.2 UnsupportedEncodingException: Binary

Fixed in Release 7.3.3.2 Build 20140916.
Also fixed in Release 7.3.3.1.4.

During the processing of Blob data from the database the Rdb thin driver may throw the following exception:

```
java.io.UnsupportedEncodingException: BINARY
        at java.lang.StringCoding.encode(StringCoding.java:269)
```

or

```
java.io.UnsupportedEncodingException: BINARY : BINARY at
oracle.rdb.jdbc.common.Column.getObjectFromDb(Column.java:1818
```

This has now been fixed.

5.18.3 Alignment Faults When Using MP Server

BUG 20046411
Fixed in Release 7.3.3.2 Build 20141119.
Also fixed in Release 7.3.3.1.5.

A fix made in release 7.3.3.1 to solve a problem with MUTEX handling within the MP Server has exposed a misaligned internal structure that may cause excessive alignment faults on OpenVMS IA-64 systems.

This problem only affects MP Servers.

This has now been fixed.

5.18.4 ResultSet.absolute() Problems

BUG 19780721
Fixed in Release 7.3.3.2 Build 20141121.
Also fixed in Release 7.3.3.1.6.

Several problems have been found when using the `ResultSet.absolute()` method when specifying negative row numbers or row numbers that are greater than the number of rows returned by the ResultSet.

An error like the following may be seen:

```
SQL Exception Connection lost : n >= n
```

In addition the `absolute()` method may return "TRUE" when no row is available.

5.18.5 Orphaned Executor Processes When Using Persona

BUG 20119695
Fixed in Release 7.3.3.2 Build 20141216.

A problem with lost executor checking within the MP Server may cause the server to prematurely attempt to terminate viable executors.

The problem is seen only when the MP Server is started using the **`-persona`** configuration option.

A problem in the termination protocol may also prevent the correct termination of the executor process. As the executor process is not terminated fully, resources may be held, which may cause resource problems if a number of these orphaned executor processes exist concurrently on your system.

A workaround to prevent the premature executor termination is to disable the lost executor checking by setting the MP Server configuration option:
`-srv.lostexecutorsweep`

to the value “0”.

Another possible work-round is to not use **persona** when using an MP Server.

Standard thin servers using **persona** are not affected.

▲contents

5.19 New Features for Release 7.3.3.1

This section describes new and changed features in Oracle JDBC for Rdb 7.3.3.1.

5.19.1 Change in Template Configuration Filename

Release 7.3.3.1.

In prior releases the JDBC installation would place a copy of the template file `RDBJDBCCFG.XML` in the RDB$JDBC_HOME directory and, if not already present, a copy will be placed in the RDB$JDBC_COM directory.

The copy in RDB$JDBC_HOME is meant to be a reference template copy only and should not be used directly in production. In order to reduce possible confusion about the use of this file, starting with JDBC release 7.3.3.1, the copy installed to RDB$JDBC_HOME is renamed to `RDBJDBCCFG_TEMPLATE.XML`.

The `RDBJDBCCFG_TEMPLATE.XML` found in the `RDB$JDBC_HOME` directory will be replaced each time you install Oracle JDBC for Rdb, however, any existing `RDBJDBCCFG.XML` file found in the `RDB$JDBC_COM` directory will not be replaced.

Note:

By default, as the logical name `RDB$JDBC_COM` points to the `SYS$COMMON:[RDB$JDBC.COM]` directory, it is this directory that the Oracle SQL/Services JDBC dispatcher will use when searching for a configuration file to use during server startup. (See the sections *JDBC Dispatcher* and *Determining the server configuration file* in the *Oracle JDBC for Rdb User Guide* for more details).

▲contents

5.20 Corrections in Release 7.3.3.1

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.3.1.

(See section Corrections in Release 7.3.3.2 for details of any fixes that may be included in patch releases 7.3.3.1.2 through 7.3.3.1.6).

5.20.1 DatabaseMetadata.getColumns Problem with DefaultValue

BUG 18900456
Fixed in Release 7.3.3.1 Build 20140717.

The **DatabaseMetadata** class method **getColumns** should return the default value for each column found for the table selection specified.

A problem introduced in Oracle JDBC for Rdb Release 7.3.3 in the handling of internal segmented string columns used to derive the underlying default value may cause JDBC to either indicate that there is no default value for the column when one actually exists, or may throw the following exception:

```
java.io.UnsupportedEncodingException: BLR : BLR
```

JDBC now correctly returns the default value found for either the table column or the domain that the column is based on.

5.20.2 Show Server Configuration File Problem

Fixed in Release 7.3.3.1 Build 20140718.

When the **Show Server** command is used within the Oracle JDBC for Rdb controller, details about the connected server are displayed. Included in the standard display is the name of the configuration file the server used on startup.

In prior versions, JDBC used the standard Java **File.getCanonicalPath()** method to retrieve the current configuration file specification. However, on OpenVMS, if the file specification contains a logical name and depending on whether the logical name points to a rooted directory, the canonical form may contain extraneous path elements within the same path string, for example:

```
SYS$COMMON:[rdb$jdbc.0703-3v0e5s]rdb$jdbc_com:rdbjdbccfg.xml
```

As shown in the above example, the path may display both the logical name and the translation of that name.

Show Server has now been changed to no longer use the canonical form to display the configuration file specification.

5.20.3 Unknown Action Message in Log file

Fixed in Release 7.3.3.1 Build 20140722.

Starting with Oracle JDBC for Rdb Release 7.3.3 it is possible that the following message text may be displayed in the server log file:

```
JNI_Action 10 <unknown ACTION>: 10
```

This action code relates to a new feature (executor checking) introduced in 7.3.3 and the log message displayed is due to a missing description string within the debug output code of the server.

This message does not represent a true error and may be safely disregarded.

5.20.4 Severe Performance Degradation in MP Server

BUG 19150908
Fixed in Release 7.3.3.1 Build 20140722.

A change in the handling of mutex operations within the MP Server made during the 7.3.3 release may cause a severe reduction in the performance of the server seen especially when the server is being utilized on a very busy system.

This has now been fixed.

5.20.5 Controller DCL Command Line Failure

BUG 18973965
Fixed in Release 7.3.3.1 Build 20140722.

A problem introduced in release 7.3.3 may cause the DCL command line parameter –**node (**or **–n)** to be incorrectly parsed by the Oracle JDBC for Rdb controller. DCL command line statements like the following used to invoke the controller will fail:

```
$ java -jar rdb$jdbc_home:rdbthincontrol.jar –
–n localhost -p 1701 -showclients
```

The statement fails showing the following exception:

```
Exception: java.lang.NumberFormatException: For input string: "localhost"
```

A possible workaround is to use the **–url** command line option instead:

```
$ java -jar rdb$jdbc_home:rdbthincontrol.jar –
–url //localhost:1701/ -showclients
```

5.20.6 Server Access Restrictions Ignored

BUG 19149514
Fixed in Release 7.3.3.1 Build 20140723.

A problem in how restriction checks are parsed from the server configuration file may cause the JDBC server to ignore certain restrictions.

For example, the following restriction may be ignored by the server if there are no other allow or deny restrictions specified for the server; the server will incorrectly allow all IPs instead of only local node users:

```
  <server
           name                = "RestrictServer"
           type                = "RdbThinSrv"
           url                 = "//localhost:1701/"
           restrictAccess      = "true">
           <allowIP IP         = "127.0.0.1"/>
      </server>
```

This problem only occurs when the older access restriction syntax is used, in particular:

- AllowIP IP=“…”
- AllowUser name=“…”

A workaround is to use the newer access restriction syntax:

- Allow IP="…"
- Allow User="…"

For example:

```
<server
         name                = "RestrictServer"
         type                = "RdbThinSrv"
         url                 = "//localhost:1701/"
         restrictAccess      = "true">
         <allow IP           = "127.0.0.1"/>
    </server>
```

5.20.7 Datetime Insertion Issues

BUG 18921457
Fixed in Release 7.3.3.1 Build 20140723.

Release 7.3.3 introduced the ability to maintain timestamp information at the nanosecond level. This feature is only available if both the client and the server sides of the JDBC connection are at Oracle JDBC for Rdb release 7.3.3 or higher. If lower JDBC releases are used, timestamp handling will resort to the default millisecond level.

Unfortunately the code required to ensure JDBC upward and downward release compatibility introduced a problem in the handling of some datetime operations.

The **PreparedStatement setObject()** method may fail to insert the correct datetime value into the Rdb database when the destination column is a datetime column such as Timestamp, VMS Date or Time and the source is a datetime object.

For example, the following may load the timestamp column with incorrect datetime values:

Given the following table: date1 (f1 integer, ts timestamp(2))

```
      String inTimestamp = "2014-07-21 09:22:10.1234567";
      java.sql.Timestamp cvtTimestamp =
          java.sql.Timestamp.valueOf(inTimestamp);
      DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
```

```
java.util.Date cvtDate = dateFormat.parse("2014-07-21");
java.sql.Date cvtsdate =
   java.sql.Date.valueOf("2014-07-21");
java.sql.Time cvtstime = java.sql.Time.valueOf("09:22:10");
long cvtlong =  cvtTimestamp.getTime();

PreparedStatement ps3 = conn.prepareStatement(
  "insert into tsbug1 (f1,ts) values (?,?)");
ps3.setObject(1, 1);
ps3.setObject(2, cvtTimestamp);
ps3.executeUpdate();

ps3.setObject(1, 2);
ps3.setObject(2, cvtDate);
ps3.executeUpdate();

ps3.setObject(1, 3);
ps3.setObject(2, cvtsdate);
ps3.executeUpdate();

ps3.setObject(1, 4);
ps3.setObject(2, cvtstime);
ps3.executeUpdate();

ps3.setObject(1, 5);
ps3.setObject(2, cvtlong);
ps3.executeUpdate();
```

Each record created will have an incorrect datetime value stored in the database.

Note:

When the **setObject()** source is a datatype that will naturally cast to a long value, the long value derived will be considered to be a milliseconds value representing the number of milliseconds that have passed since January 1, 1970 00:00:00.000 GMT.

A workaround is to use the appropriate datatyped SET method instead of the generic **setObject()** method, for example:

```
ps3.setTimestamp(1, cvtTimestamp)
```

5.20.8 MP Server May Become CPU Bound

Fixed in Release 7.3.3.1 Build 20140731.

A problem introduced in 7.3.3 may cause the MP Server to go into a tight CPU loop and consume excessive CPU even when no clients are attached.

This problem is a result of an unhandled array list exception that may manifest as the following exception that will be shown in the server log file:

```
java.util.ConcurrentModificationException
```

This has now been fixed.

5.20.9 Controller Null Pointer Exception When No Default Server

Fixed in Release 7.3.3.1 Build 20140812.

When starting a server using the controller DCL command line option `-startserver`, the name of the server provided is used to locate the server characteristics within the configuration file used by the controller.

If the named server does not exist, the controller will then use the characteristics of the server named “Default”. Due to a problem in the server configuration setup, if the configuration file does not contain a Default server definition, a Null Pointer exception is thrown.

The workaround for this is to either specify the name of a server that does exist within the configuration file, or alternatively, ensure that the configuration file does have a “Default” server specified, for example:

```
<?xml version = '1.0'?>
<!-- Configuration file for Rdb Thin JDBC Drivers and Servers -->
<config>
   <!-- SERVERS -->
   <servers>
      <!-- DEFAULT server characteristics-->
      <server
        name="DEFAULT"
        url="//localhost:1701/"
        />

      <server
.
.
.
```

This problem has now been fixed. If the configuration file has no “Default” server specified and the named server does not exist, the server will be started using the default thin server port 1701.

5.20.10 Pool Server Tries to Restart Running Server

Fixed in Release 7.3.3.1 Build 20140812.

The pool server will attempt to restart any server in its pool that is marked for restart and has been found to be unresponsive. A problem in the viability check for servers may lead the pool server to incorrectly mark a server as non-responsive even though the server is running and responsive.

In releases prior to 7.3.3 the pool server will immediately remove servers from its pool it thinks are unresponsive, thus preventing them from taking part in the pool. The server will no longer be chosen as a candidate for new client connections.

In release 7.3.3 and above, the pool server will maintain the server in its pool but will periodically check to see if it has become responsive again. As the server is still in the pool it may still receive and process new connection requests from the pool server, however after a predefined number of viability checks, if the pooled server is still deemed unresponsive it will then be removed from the pool, and no further connections will be redirected to it.

In addition, if the pool server finds an unresponsive server, and that server is marked for restart, the pool server will attempt to restart this server, however if the server is still running the restart attempt will fail and an error will be logged in the pool server log.

A workaround for this problem is to change the pool server configuration file and set all pooled servers autorestart to false, for example:

```
<server
         name                = "srv1"
         type                = "RdbThinSrv"
         url                 = "//localhost:1911/"
         autostart           = "true"
         autorestart         = "false"
 />
```

The pool server has now been fixed to improve its pool viability checking.

5.20.11 Controller Poll Reopenlogs Hangs

Fixed in Release 7.3.3.1 Build 20140815.

If the controller command POLL REOPENLOGS is issued as the first POLL command within the controller session, the controller will hang.

A workaround is to issue a standard POLL command first prior to issuing the POLL REOPENLOGS:

```
rdbthincontrol> poll
Polling servers ...
... Polling complete : 1 responded.
RdbThinSrv1701(0) //192.1.32.212:1701/ (0x2542DAD9<625138393>)
 node = alfred
rdbthincontrol> poll reopenlogs
RdbThinSrv1701(0) //192.1.32.212:1701/ (0x2542DAD9<625138393>)
 node = alfred :  Logs Reopened
rdbthincontrol>
```

5.20.12 Pool Server Usage Balancing Problem

Fixed in Release 7.3.3.1 Build 20140815.

When the server balancing attribute for a Pool Server is set to “USAGE”, the pool server should choose the server from its pool that has the most free user slots available. The free user slots count is the number of potential connections the server can still allow before the server MAXCLIENTS is reached.

In addition, any server that currently has no client connections should be preferentially chosen.

A problem in the usage algorithm may cause the pool server to disregard servers that have no current connections and select the least used server from the list of servers that have at least one current connection.

This has now been fixed, as documented in the *Oracle JDBC for Rdb User Guide*. If a server does not currently have any client connections, it will be chosen in preference to other busy servers.

5.20.13 Configuration File Problem Regression

Fixed in Release 7.3.3.1 Build 20140820.

A problem with the inheritance of server attributes from the “DEFAULT” server was fixed in release 7.3.2. This problem has unfortunately resurfaced.

The problem and workaround are described in release note:

5.4.3 Some Server Characteristics not Correctly Inherited from DEFAULT

This has now been fixed.

5.20.14 Server Configuration Deny User Not Enforced

Fixed in Release 7.3.3.1 Build 20140826.

The server attribute ‘`deny user="xxxxx"`’ in conjunction with the attribute ‘`restrictAccess="true"` ‘ may be used to deny specific users from accessing a JDBC server:

```
<server
         name                 = "srv1"
         type                 = "RdbThinSrv"
         url                  = "//localhost:1911/"
         restrictAccess       = "true"
          >
         <deny user          = "murray"/>
 </server>
```

However, a problem in the configuration attribute parsing may prevent the server from correctly denying access to the specified users.

A workaround for this problem is to also include a valid IP restriction for the same server definition. The wildcard IP address may be used:

```
<server
         name                 = "srv1"
         type                 = "RdbThinSrv"
         url                  = "//localhost:1911/"
         restrictAccess       = "true"
```

```
            >
           <deny user          = "murray"/>
           <allow ip           = "*.*.*.*"/>
   </server>
```

▲contents

5.21 New Features for Release 7.3.3

This section describes new and changed features in Oracle JDBC for Rdb 7.3.3.

5.21.1 Oracle Rdb PID now Displayed

Fixed in Release 7.3.3.
Also available Release 7.3.2.0.2.

The process/stream ID used internally by Oracle Rdb to uniquely identify client connections and that is displayed in output from operations such as `RMU/SHOW USERS` is now displayed by JDBC in server logs and in controller SHOW CLIENT output.

The Rdb process/stream information is only available to JDBC if the client has connected to a database within an Rdb environment of Version 7.2.5.0.0 or above, and the JDBC server used is release 7.3.2.0.2 or above.

See the *Show Clients* section of the *Oracle JDBC for Rdb User Guide* for more information.

5.21.2 Controller Command Show Executors

Fixed in Release 7.3.3.
Also available Release 7.3.2.0.2.

A new controller command `show executors` is now available to display information about executors hosted by the currently connected Multi-process server.

See the *Showing Executors* section of the *Oracle JDBC for Rdb User Guide* for more information.

5.21.3 Extra Timestamp Precision

Fixed in Release 7.3.3.0.2.

JDBC has now been enhanced to allow the use of extra precision in the fractions of seconds associated with timestamps.

Prior to this release, when transferring data between Rdb datetime or timestamp columns and Java timestamp variables, JDBC would limit the precision of second fractions to 3 decimal places, i.e., milliseconds.

If the Rdb timestamp or datetime column was set by Rdb, for example when using CURRENT_TIMESTAMP, the precision of the seconds value would be further limited to the maximum precision supported by Oracle Rdb SQL, i.e., 2 decimal places (hundreds of seconds).

It is possible to load higher precision time data into the timestamp columns programmatically, however earlier versions of JDBC would truncate the value to milliseconds during datetime operations.

JDBC has now been changed and will now allow the use of timestamp data with up to 7 decimal places (i.e., OpenVMS ticks, or 100 nanosecond units).

The extra timestamp precision is only available to applications using JDBC 7.3.3 Patch 2 (or later) native or thin drivers, and only available to the thin driver if it is connected to a JDBC server running release 7.3.3.0 Patch 2 (or later).

If required, the older precision of 3 decimal places for the fraction of seconds may be retained for individual database connections by using the "`@TICKS`" connection string qualifier set to ***false***.

For example:

```
Connection conn = DriverManager.getConnection(
"jdbc:rdbThin://bravo:1755/my_db_dir:pers.rdb@ticks=false",
user, pass);
```

See the *Connection Options* section of the *Oracle JDBC for Rdb User Guide* for more information.

5.21.4 NIslang Connection Switch

Fixed in Release 7.3.3.0.2.

When the values of text columns or literals are moved between Rdb and the JDBC internal buffers, JDBC must convert the value to and from Unicode. To do this JDBC needs to know the expected character set encoding that the Rdb data is using. JDBC will derive the encoding required using the character set information supplied to it by Rdb.

Historically, especially prior to the current character set support found in Rdb, non-MCS characters such as Kanji characters may have been stored in columns or literals that have the DEC_MCS character set associated with them. If this is the case, when JDBC converts these values to Unicode the original character may be lost as JDBC will assume that the character was DEC_MCS.

To provide a way for customers that have non-MCS characters stored in DEC_MCS literals or columns to correctly pass the original characters to JDBC applications, JDBC now provides a connection string option **`nlslang`**.

This option may be used to tell JDBC the encoding used in columns or literals that have the DEC_MCS character set attribute.

For example:

```
Connection conn = DriverManager.getConnection(
"jdbc:rdbThin://bravo:1755/my_db_dir:pers.rdb@nlslang=DEC_KANJI",
user, pass);
```

See the *Connection Options* section of the *Oracle JDBC for Rdb User Guide* for more information.

Note:

This option does not alter the SQL National Language.

5.21.5 Use Query Header as Description

Fixed in Release7.3.3.0.2.

The method `DatabaseMetadata.getColumns()` provides information about table columns to the calling JDBC client. One of the standard properties returned by the `getColumns()` method is **Remarks** which JDBC will fill from the column description field found for that column returned by Oracle Rdb. By default, JDBC uses the **RDB$DESCRIPTION** field of the **RDB$RELATION_FIELDS** system table for the specified column.

JDBC will now allow you to specify that the Remarks property should be derived from the Query Header field (the **RDB$QUERY_HEADER** column) of the **RDB$RELATION_FIELDS** instead of the **RDB$DESCRIPTION** column.

The new connection string qualifier `@useQueryHeader` may be used to tell JDBC to use the alternate source of the column's Remarks field. If the value is set to ***true***, JDBC will fill the Remarks field from the **RDB$QUERY_HEADER** column of the **RDB$RELATIONS_FIELDS** table. For example:

```
Connection conn = DriverManager.getConnection(

"jdbc:rdbThin://bravo:1755/my_db_dir:pers.rdb@useQueryHeader=true",
    user, pass);
```

Optionally, you may also specify a specific character set to use to encode the contents of the query header column when returned to your application. For example:

```
Connection conn = DriverManager.getConnection(
   "jdbc:rdbThin://bravo:1755/my_db_dir:pers.rdb"+
  "@useQueryHeader=Dec_Kanji", user, pass);
```

If you do not specify a character set to use but have specified an alternate character set using the "`@NLSLANG`" connection string qualifier, JDBC will use that character set to carry out appropriate encoding of the query header.

If no character set is specified in the "`@useQueryHeader`" connection string qualifier or specified by the "`@NLSLANG`" connection string qualifier, then LATIN1 will be used during encoding.

See the *Connection Options* section of the *Oracle JDBC for Rdb User Guide* for more information.

▲contents

5.22 Corrections in Release 7.3.3

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.3.

5.22.1 Controller START SERVER Problems

Fixed in Release 7.3.3 Build 20131126.

The controller will fail with a Java NullPointer exception if the `Start Server` command is used within a controller session that does not have a configuration file specified:

```
$ java -jar RDB$JDBC_HOME:RDBTHINCONTROL.JAR
rdbthincontrol> start server  1701
NullPointerException:
java.lang.NullPointerException
```

A workaround for this problem is to start the controller application using a configuration file that contains at least one server definition.

An additional problem may be seen if the `Start Server` command is issued without any other options. This command will fail with the parsing error:

```
$ java -jar RDB$JDBC_HOME:RDBTHINCONTROL.JAR
rdbthincontrol> start server
Starting server ...
SQLException: Error parsing URL :  //localhost:0/:S1000
```

These problems have now been fixed.

5.22.2 Executor Initialization Problem

Fixed in Release 7.3.3 Build 20131126.
Also fixed in Release 7.3.2.0.3 Build 20130517.

Depending on the system load it is possible that a newly created executor process may not have fully completed its initial response to the Multi-process server before the server checks the executor response data.

The incomplete response information may cause a problem with the initialization of the connection that is waiting on the executor process creation.

The server log may show an RdbException logged for the thread where a zero value for the executor version and/or instance may be seen similar to :

```
oracle.rdb.jdbc.common.RdbException: Executor images version
mismatch, server shr = 73202:20130501 , executor = 0:20130501
```

or :

```
oracle.rdb.jdbc.common.RdbException: Executor images version
mismatch, server shr = 73202:20130501 , executor = 0:0
```

It is possible, depending on the loading of the server system, that this problem may in turn cause various access violations to be raised during the connection request, which in turn will cause connection requests to be terminated in error.

5.22.3 Idle Client Termination Breaks Subsequent Connection

Fixed in Release 7.3.3 Build 20131126.
Also fixed in Release 7.3.2.0.4 Build 20130520.

A problem in the handling of idle client termination may cause subsequent connection requests to fail.

During the termination of an idle client, the Multi-process server will force a disconnection of the connection and terminate the executor process used by that connection. This executor is no longer available for re-use, however a problem in the executor cleanup code caused the server to incorrectly mark the executor as valid for re-use. A subsequent connection request may get this unusable executor which will cause the request to fail with an exception similar to the following being raised in the client application:

```
oracle.rdb.jdbc.common.RdbException: Connection lost :
java.io.EOFException
    @rdb.Client.CONNECT_SECURE_V73
```

It is also possible that this same problem may eventually cause the Multi-process server to hang requiring a forced termination of the server.

This problem only occurs when using Multi-process servers that have a non-zero client idle timeout value set for the "cli.idleTimeout" server configuration option.

5.22.4 Controller STOPSERVER Command Problem

Fixed in Release 7.3.3 Build 20131126.
Also fixed in Release 7.3.2.0.5 Build 20130530.

A problem introduced in the first non-patched 7.3.2 release of JDBC may cause the controller command line command STOPSERVER to raise an exception.

For example:

```
java -jar rdb$jdbc_home:rdbthincontrol.jar -stopserver 1818
```

raises the following exception:

```
SQLException: Error parsing URL :  //localhost:0/:S1000
```

This has now been fixed.

5.22.5 Hang on Connection to Server When Using SSL

Fixed in Release 7.3.3 Build 20131126.
Also fixed in Release 7.3.2.0.6 Build 20130612.

A code change made for the 7.3 release of JDBC may cause a client connection request to wait indefinitely for a new connection when SSL is used.

If the client requests a normal , non-SSL connection to an SSL enabled JDBC server or attempts to connect using SSL to a non-SSL JDBC server, the client connection will wait indefinitely.

Prior to release 7.3 JDBC, the server would determine that there is a mismatch in the SSL handshake after about 1 second , and force the connection request to terminate immediately. A code change made during the 7.3 release accidentally disabled this handling.

A workaround is to set the servers *`cli.idletimeout`* configuration attribute to a positive value. If the SSL handshake does not complete within that time, the server will abort the connection request. However, this also means that bound clients will be forcibly disconnected if they are idle for longer than the *`cli.idletimeout`* period.

This has now been fixed.

5.22.6 DatabaseMetadata Pattern Matches and Nulls

Fixed in Release 7.3.3 Build 20131126.

Also fixed in Release 7.3.2.0.6 Build 20131118.

In prior versions, Oracle JDBC for Rdb drivers did not correctly handle null values for pattern parameters used in various DatabaseMetadata methods.

The JDBC standard states: "*If a search pattern argument is set to* `null`*, that argument's criterion will be dropped from the search*".

The DatabaseMetadata methods that were not correctly handling nulls include:

- GetProcedures()
- GetProcedureColumns()
- GetDomains()
- GetFunctions()

These methods now handle null patterns correctly.

5.22.7 DatabaseMetadata Missing Index Information

Fixed in Release 7.3.3 Build 20131126.
Also fixed in Release 7.3.2.0.6 Build 20131118.

In prior versions of Orace JDBC for Rdb, the DatabaseMetadata.GetIndexInfo() method may fail to return information about existing indexes.

This method is used by tools such as the Oracle Rdb Extension for SQL Developer, to return a list of indexes associated with each table found within the database schema.

Due to this problem, tools such as SQLDeveloper may fail to display existing indexes.

This has now been fixed.

5.22.8 Forced Client Termination May Crash Thin Server

Fixed in Release 7.3.3 Build 20131126.

It is possible that the forced termination of client applications by stopping or destroying the client application process may cause an access violation to occur within the JDBC server that the client was connected to.

Forcible termination of application using the thin driver may prevent the proper rundown of the client side of the JDBC connection. The server will determine that the client connection has been lost and will try to close down the connection in a controlled manner, however it is possible that this cleanup processing may get interupted by other server activities and may result in an access violation within the server's shared image.

The JDBC servers have now been changed to carry out more aggressive locking of connection closedown operations which will prevent other server threads from interfering with memory cleanup.

Oracle recommends to not forcibly close down client applications by process termination using STOP/ID etc. Clients using the thin driver should either be allowed to close-down normally or you may use the JDBC controller to stop the client's connection within the server, which will carry out a more regulated cleanup of the connection.

5.22.9 Show Clients May Cause NullPointerException in Server

Fixed in Release 7.3.3 Build 20140304.

Using the controller command **SHOW CLIENTS** may cause an exception to be raised in the server the command was issued to:

```
java.lang.NullPointerException
oracle.rdb.jdbc.srv.ActionHandler.doControlPacket
oracle.rdb.jdbc.srv.SrvActionHandler.doControlPacket
```

This problem occurs infrequently and only happens when a client happens to disconnect from the server within a small time window during the building of the **show client** results by the server.

This problem does not affect the execution or viability of the active server, but will cause the **show client** request to fail within the controller.

Note:

As the ORCM application uses a variant of the thincontrol **show clients** command it is also liable to show this same problem.

This has now been fixed.

5.22.10 Access Violation in MP Server

Fixed in Release 7.3.3 Build 20140313.

A problem in thread synchronization during some server control operations may cause the MP Server to throw an Access Violation exception and terminate.

Under relatively rare circumstances the MP Server may fail due to an Access Violation while it is trying to deliver client or executor information to the thincontroller or ORCM.

The following exception may be seen in the server log prior to the server terminating:

```
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=00000000xxxxxxx, PC=000000000xxxxx, PS=0000xxxx

RDBJDBCMPSHR73  CJDBCCTX_M  getDaMemUsage

RDBJDBCMPSHR73  CJDBCCTX_M  getStmtMemUsage
```

This problem may be seen more often in MP Servers that create and destroy executor processes frequently, and may happen immediately following the execution of SHOW CLIENTS or SHOW EXECUTORS in a thincontroller session.

As ORCM also issues similar commands to obtain information about current server use, it may also instigate this server problem.

The possibility of the server failure may be reduced by providing adequate free executor processes for MP Server clients to use, thus reducing the need for the server to create or destroy executor processes.

5.22.11 Syntax Errors in Insert and Update Statements

Fixed in Release 7.3.3 Build 20140314.

A problem in the dbkey retrieval code generated by the JDBC drivers in order to return the dbkey of newly updated or inserted record results in a syntax error.

The Rdb JDBC drivers add additional code to INSERT and UPDATE statements during SQL statement preparation to return the dbkey of the resultant record. To determine the position of the additional SQL syntax, the JDBC drivers do a search for any existing “INTO” clause. Unfortunately the drivers may mistake literals or delimited identifiers may contain the whitespace-separated word “INTO” for the searched keyword resulting in incorrect SQL syntax being generated.

For example, the following SQL text may fail:

```
insert into tab9(fld1,fl2) values ('55555',' TEST INTO ')
```

The SYNTAX_ERR exception raised shows the modified SQL statement text,

```
%SQL-F-SYNTAX_ERR, Syntax error : in "insert into tab9(fld1,fld2) values
('55555',' TEST , DBKEY INTO '), ?"
```

A workaround is to add the undocumented prefix “{nodbk}” at the start of the SQL text , which will disable the creation of the extra dbkey retrieval code:

```
{nodbk}insert into tab9(fld1,fl2) values ('55555',' TEST INTO ')
```

This has now been fixed.

5.22.12 Statement.getGeneratedKeys() Throws RdbException

Fixed in Release 7.3.3 Build 20140314.

Calling the `Statement.getGeneratedKeys()` method on an executed statement that does not auto-generate keys should return an empty ResultSet.

The Rdb JDBC drivers do return an empty ResultSet, but incorrectly mark the ResultSet as closed. Accessing this ResultSet may raise the following exception:

```
oracle.rdb.jdbc.common.RdbException: Closed Resultset
```

The Rdb JDBC drivers now mark the ResultSet as open.

5.22.13 SQLException Thrown in ResultSet.isBeforeFirst()

Fixed in Release 7.3.3.0.3 Build 20140520.

When using the method **ResultSet.isBeforeFirst**() on a ResultSet produced by some **DatabaseMetaData** methods and the **Statement.getGeneratedKeys** method, the following exceptions may be raised:

```
    java.sql.SQLException, Missing Curs in rdbFetchRow
```

or

```
    java.lang.NullPointerException
```

This has now been fixed.

5.22.14 PreparedStatement executeBatch() Problems

Fixed in Release 7.3.3.0.3 Build 20140526.

A problem in the handling of batch entries for **PreparedStatement** may produce incorrect results after batch execution.

Certain batch entries may be executed twice when **PreparedStatement.executeBatch()** method is called. If the batch entry contains an update or delete SQL statement, the entry may be executed twice in immediate succession.

This may cause problems as the second instance of the execution may change the intended behaviour of the original statement. For example the following SQL text :

```
update tab1 set F1 = F1 +1
```

will result in the F1 column being updated twice and end up as F1 +2.

In addition the updated count for that batch entry may be returned incorrectly, for example the following SQL text:

```
delete from t1b where F1 = ?
```

will result in a value of 0 being returned in its update count value, as the first execution of the statement will remove the records and the subsequent execution will then return the value 0 as no records will be affected by its execution.

This problem only affects **PreparedStatements** that have update or delete SQL statements batched. Normal **Statements** using **executeBatch()** are not affected.

Another problem may be seen when executing an update SQL statement using the PreparedStatement executeBatch. If the statement is not a singleton operation, that

is, if it updates more than one record, the batched statement may fail with the following error:

```
java.sql.BatchUpdateException: Problem with one or more batched query
```

And show the following underlying Rdb error:

```
%RDB-E-MULTIPLE_MATCH, record selection criteria should identify only one
record; more than one record found
```

These problems have now been fixed.

▲contents

5.23 New Features for Release 7.3.2

This section describes new and changed features in Oracle JDBC for Rdb 7.3.2.

5.23.1 Server Options List Inheritance

Release 7.3.2.

When an XML-Formatted configuration file is used, many server characteristics may be inherited from either the DEFAULT or the DEFAULTSSL server definitions.

If a characteristic is specified within a specific server definition, that specification will take precedence. If the characteristic is not specified in the specific server definition, then if it exists in the default server definition it will be inherited.

In previous versions of Oracle JDBC for Rdb, some characteristics were not inherited and even if specified within the default server definition they would not be passed on to other servers in the same configuration file.

The characteristics that did not get inherited were all list options that included:

Server Configuration List Options

Option
<allowDatabase name="database-name">
<allowPrivUser name="user-name">
<allowUser name="user-name">
<deny sql ="sql-pattern">

Option
<enableEvent name="event-name">

List options provide a mechanism to specify one or more named options that may be used by the server for validity, event checking and other operations. Each server definition may contain multiple values for the same named option, for example, multiple `allowUser` entries can be used to provide a list of users that will be allowed to access that server.

Starting with release 7.3.2, the list options specified above may now be inherited from the default server.

The inheritance of these options is cumulative, in that the resultant list of options is the combination of those present in both the default and the specific server definitions.

If a server definition already has an option with the same type and name, it will not be replaced.

Note:

An error in prior documentation specified that the `enableEvent` configuration option would be inherited; this is not true for versions of Oracle JDBC for Rdb prior to release 7.3.2.

5.23.2 Oracle JDBC for Rdb Manager Server

Release 7.3.2.

The Oracle JDBC for Rdb manager server is a server-side component that services JDBC management requests.

The main purpose of the manager server is to provide a mechanism that will allow JDBC servers to be started up on nodes remote to the one the controlling application, such as the Oracle JDBC for Rdb controller, is running.

Please see the *Oracle JDBC for Rdb Manager Server* section of the *Oracle JDBC for Rdb User Guide* for more information on manager servers.

5.23.3 Restricting Server Access by IP

Release 7.3.2.

JDBC servers may now check incoming connection IPs to determine if the client connection will be accepted.

New server configuration options:

- <allow ip = "<valid ip mask>" />
- <deny ip = "<valid ip mask>" />

Using the new server configuration options, you may restrict access to the server by having the server check the originating IP of the client connection to either accept or reject that IP. The valid IP mask may be either a simple IP specification or may be Java regular expression.

See the *Restricting Server, Database and Operational Access* section of the *Oracle JDBC for Rdb User Guide* for more details.

5.23.4 Executor Balancing

Release 7.3.2.

By default, the Multi-process server uses a first-in/first-out (FIFO) scan of its free executor list to select the next executor process to allocate to the new connection request.

Oracle JDBC for Rdb now allows you to choose from several new executor balancing protocols.

Please see the *Executor Balancing* Section of the *Oracle JDBC for Rdb User Guide* for more information.

5.23.5 MinFreeExecutors

Release 7.3.2.

Multi-process servers may now be configured to maintain a minimum set of free executor processes when the server is cleaning up executor processes that have exceeded their maximum idle time.

Please see the *Multi-process Server Configuration Options* and the *Executor Maintenance* sections of the *Oracle JDBC for Rdb User Guide* for more information.

5.23.6 Executor Reuse

Release 7.3.2.

Multi-process servers may now be configured to change the way they reuse executor processes.

Please see the *Multi-process Server Configuration Options* and the *Executor Reuse* sections of the *Oracle JDBC for Rdb User Guide* for more information.

5.23.7 LogFile Patterns

Release 7.3.2.

The server `logfile` configuration option provides an absolute or relative file specification to use as a log file. This specification may also be a special pattern, in which case certain key character sequences may be used in conjunction with normal ASCII filename characters to specify the automatic creation of the log file name.

See the *Logfile Pattern* section of the *Oracle JDBC for Rdb User Guide* for more information.

5.23.8 New Server Configuration Option retainRdbSQLState

Release 7.3.2.

In keeping with third party application requirements, during the processing of INSERT statements, the SQL State returned when using the `SQLException.getSQLState()` method is set to the value "S1000" indicating an error has occurred. The message text of the exception contains the details of the error raised.

Oracle Rdb delivers more specific SQL State values including Rdb specific ("Rxxxx") state values, but these may be suppressed by JDBC.

The server `retainRdbSQLState` configuration option may be used to change this behaviour. When set to "true", JDBC will return the original SQL State raised by the underlying database.

For example:

```
<server
       name = "DEFAULT"
       type = "RdbThinSrv"
       url = "//localhost:1701/"
       retainRdbSQLState = "true" />
```

See the *Server Configuration Options* section of the *Oracle JDBC for Rdb User Guide* for more information.

5.23.9 New Connection Option "APP"

Release 7.3.2.

To help identify applications usage of JDBC servers, a new connection option, `app`, has been introduced.

For example:

```
Connection conn = DriverManager.getConnection(
"jdbc:rdbThin://bravo:1755/my_db_dir:pers@app=MyOwnApplication",
user, pass);
```

See the *Connection Options* section of the *Oracle JDBC for Rdb User Guide* for more information.

▲contents

5.24 Corrections in Release 7.3.2

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.2.

5.24.1 Event Flag Problem with Long Executor Name Prefixes

Fixed in Release 7.3.2 Build 20120821.

During the creation of an executor process by the Multi-process server, the server attempts to provide a unique process name for the executor so that it may be easier to recognize when using system tools such as SHOW SYSTEM.

By default, the server will create the process name by utilizing the server name and the unique connection id.

To allow better control of process naming, the prefix of the executor process name may also be explicitly specified by using the server configuration option `srv.execPrefix`. The server will use this prefix and append a sequence number to it in an attempt to provide an unique name.

As there is a limit of 15 characters allowed for OpenVMS process names it is possible that longer prefixes may cause the unique portion of the name to cycle around faster and increase the probability that a process name may be chosen that matches an existing running executor process.

If the process name matches an existing process, the startup of the new executor will fail and the executor process will not be present to acknowledge startup completion to the server.

Normally an executor process failure will be correctly caught by the server and the appropriate exceptions raised. However if the failure happens after the process is correctly created but before this initial acknowledgement, the server will not see the failure for several minutes.

Eventually, an exception similar to the following will be raised:

```
oracle.rdb.jdbc.common.RdbException: Executor not responding on
 EFN 66  EFNMASK 4 after 302 seconds
```

This message will be written to the server log. In addition other messages describing the reason for the failure may be found in the log files associated with the executor in the JDBC log area which by default is pointed to by the logical name RDB$JDBC_LOGS.

Multi-process servers have now been changed to try to better handle process name clashes by doing further name checking on the running system prior to attempting to start the new process.

5.24.2 Executor Process Termination Problem

Fixed in Release 7.3.2 Build 20120821.

During the normal shutdown of an executor, the Multi-process server will request the executor process to self-terminate to ensure that all resources are correctly released prior to the OpenVMS process terminating.

After this self-termination is requested, the server will then check to see if the process is still there, and if it is, will forcibly stop the process.

In heavily utilized systems it is possible that the executor process self-termination has not completed by the time the server does its secondary check and subsequent process termination. In this situation the process may get into a “limbo” state where it fails to closedown correctly and depending on when the second termination is issued the executor process may get trapped in an endless exception handler loop.

The result of this problem is that the executor process may start to consume large amounts of CPU, or may remain idle on the system without releasing resources. In addition this process may cause other problems if the server is restarted and tries to use the same process name for a newly created executor.

This has now been fixed, the server will now allow a greater amount of time for the self-termination to occur before forcing an image exit.

5.24.3 Some Server Characteristics Not Correctly Inherited from DEFAULT

BUG 14510287
Fixed in Release 7.3.2 Build 20120821.

Some server characteristics are not correctly copied from the DEFAULT or DEFAULTSSL server definitions when named servers are used.

For example if `mycfg.xml` contains the following:

```
.
.
.
 <server
      name="DEFAULT"
      type="RdbThinSrv"
      url="//localhost:1701/"
      maxClients="-1"
      restrictAccess = "true"
      >
      <allowUser name = "JDBC_USER"/>
 </server>

 <server name="srv1" />
.
.
.
```

Starting the server using this configuration and using the –verify switch to verify the server characteristics:

```
java -jar rdb$jdbc_home:rdbthinsrv.jar -cfg mycfg.xml -name "srv1"  -verify
```

will raise the following exception:

```
java.lang.NullPointerException
     at oracle.rdb.jdbc.srv.RdbSrv.logServerDetails
```

Although the `restrictAccess` option is correctly inherited , the `allowUser` option from the DEFAULT server is not, causing this problem.

Other server characteristics may also fail to be inherited, although several may not cause immediately obvious problems.

The following is a list of server configuration options that are not correctly inherited from either of the default server definitions:

Server Configuration Option

Option
<allowDatabase name="database-name">
<allowPrivUser name="user-name">
<allowUser name="user-name">
<deny sql ="sql-pattern">
<enableEvent name="event-name">

A workaround is to place these options explicitly in each server definition that requires them.

This has now been fixed.

5.24.4 Memory Problem with Pool Servers and Java 1.6.0-2

Fixed in Release 7.3.2 Build 20120822.

A problem in Java 1.6.0-2 in the recovery of memory when byte streams are garbage collected may prevent the associated memory from being recovered correctly which may lead to a memory leak in the Java VM. This problem does not occur if the byte streams are explicitly closed prior to garbage collection.

JDBC servers try to ensure that all byte streams used are correctly closed prior to releasing the socket connection to the client. However, a problem within Pool Servers may cause the explicit closure of a byte stream to be bypassed.

The failure of the byte stream to be garbage collected correctly may eventually result in Java VM problems and may cause an exception similar to the following which will terminate the Pool Server:

```
%SYSTEM-F-ASTFLT, AST fault, SP=0082D250, param=7FFEFFC8,
.
.
.
```

The Pool Server has now been modified to ensure that all byte streams are correctly closed prior to socket closure.

OpenVMS Java engineering is aware of this problem.

5.24.5 Exception not Thrown When Record Locked During Update

Fixed in Release 7.3.2 Build 20130301.

Normally, if JDBC finds a record locked during an insert, update or delete Statement operation, a Record Locked exception will be raised.

However, a problem in the handling of record locks by the Native driver prevents the driver from raising a locked record exception correctly.

Instead, if a locked record is found, the update operation will be aborted and the statement's update count, obtained using the `Statement.getUpdateCount(),` method will be set to -4.

Although an exception is not raised, you may still determine if a record lock has occurred by calling the `Statement.getUpdateCount()` after calling the statement execute. If the value returned is -4, a locked record was found. If no locked records were found, the value returned will be the number of records affected by the update.

This problem affects the Native driver only. The Thin driver is not affected.

This problem has been fixed.

5.24.6 Classpath Documentation Error

Fixed in Release 7.3.2 Build 20130426.

In prior versions of Oracle JDBC for Rdb, due to a typographic error, the `Driver Class` section of the *Oracle JDBC for Rdb User Guide* referenced the classpath for the JDBC drivers as:

Driver Class

Classpath

```
oracle.jdbc.rdb.rdbThin.Driver
```

or

```
oracle.jdbc.rdb.rdbNative.Driver
```

The classpath values documented were incorrect, and should read:

Driver Class

Classpath

```
oracle.rdb.jdbc.rdbThin.Driver
```

or

```
oracle.rdb.jdbc.rdbNative.Driver
```

The subsequent code example in the same section shows the classpath correctly.

A similar error may be found in the following *Oracle JDBC for Rdb User Guide* sections:

- `Results Class`
- `Blob Class`

This has now been fixed.

▲contents

5.25 New Features for Release 7.3.1

This section describes new and changed features in Oracle JDBC for Rdb 7.3.1.

5.25.1 “Owner” may be Used in XML Configuration Files

Release 7.3.1.

The keyword `OWNER` may be used in certain sections that require a username in the XML Server configuration file.

The keyword `OWNER` specifies the OpenVMS account username of the account used to start the server and may be used instead of a username in sections such as `AllowUser` or `AllowPrivUser`. For example:

```
.
.
.
<server
            name = "rdbthnsrv8"
            type = "RdbThinSrv"
            url = "//localhost:1708/"
            restrictAccess = "true"
            srv.showPoll = "true"
            allowAccessToCL = "true"
        >
            <allowUser name = "jdbc_user"/>
            <allowUser name = "owner"/>
            <allowPrivUser name = "owner"/>
</server>
.
.
.
```

5.25.2 SQL Statement Restriction and Denial

Release 7.3.1.

Servers may be configured to restrict or deny certain SQL statements.
See section *Restricting SQL Statements* in the *Oracle JDBC for Rdb User Guide* for more details.

5.25.3 Event Notification

Release 7.3.1.

Servers may be configured to notify enlisted event listeners of events as they occur on the server. Events such as connect and disconnect, and memory usage may be monitored.

See section *Event Logging and Notification* in the *Oracle JDBC for Rdb User Guide* for more details.

5.26 Corrections in Release 7.3.1

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.1.

5.26.1 Multi-process Server Connection Hang When Executor Dies

Fixed in Release 7.3.1 Build 20111110.

If an executor process dies or is forcibly terminated it is possible that the associated connection within the Multi-process (MP) server may hang indefinitely waiting on a response from the non-existant executor.

Executor process termination is notified to the server through several different internal mechanisms. However inherent in Pthread mutex operations is the possibility that locking operations may prevent the associated thread from acting on the termination notification. If this termination notification is missed or blocked, the thread may wait indefinitely.

The MP Server has been changed to include an executor process sweeper thread that will check each active connection utilizing an MP executor to determine if it is still viable.

If the sweeper finds that an executor has been flagged as terminated but the connection is still active and has not been rundown, the thread for the active connection will be forcibly terminated causing a socket exception on the client side.

The new server parameter `srv.lostExecSweep` may be used to specify the frequency of the sweeper thread execution.

See the *Oracle JDBC for Rdb User Guide* for more information on the server parameter `srv.lostExecSweep`.

5.26.2 Multi-process Executor Process Terminates Unexpectedly

Fixed in Release 7.3.1 Build 20111111.

A problem in the server viability checking carried out by the executor process may cause the executor process to self-terminate unexpectedly.

If an executor is associated with an active connection and has remained idle for a predefined amount of time as detailed below, it may incorrectly determine that the parent server is no longer running and self-terminate.

The amount of time the executor will remain in an idle state before termination depends on the MP Server attribute `srv.MPMaxTries`.

If the executor is idle for more than `srv.MPMaxTries * 3` seconds, it may self-terminate. Thus if the MP Server is using the default `srv.MPMaxTries` of 500 then the executor will self-terminate after 25 minutes of idle time if it still holds an active connection.

This problem only effects executors associated with active connections and not executors within the MP Server free executor pool.

A workaround is to set the `srv.MPMaxTries` to a very large number to delay this termination.

This problem was introduced in release 7.3.0.0 and is now fixed.

5.26.3 Pooled Server AutoRestart Problem

Fixed in Release 7.3.1 Build 20111117.

A problem introduced in release 7.3.0.2.0 (V7.3-02) may cause the Pool server to incorrectly flag its children pooled servers as unavailable and prevent new connections from being made to these servers.

If a pooled server has “Autorestart” enabled, the Pool server will periodically check to see if the server is viable and if it determines that the server is no longer responding, the Pool server will mark the pooled server as unavailable and then attempt to restart it.

A problem in how the Pool server determines the viability of the child server causes the Pool server to incorrectly mark the child as unavailable. The attempted restart will fail as the original pooled server is still present and connected to the socket that the new instance is trying to allocate. Thus the pooled server will then remain marked as unavailable and will no longer participate in the pool.

This problem will prevent all pooled servers marked as “Autorestart” from participating in the server pool.

A workaround is to set `Autorestart="FALSE"` for each of the pooled servers in the server XML configuration file. This will prevent the Pool server from checking and incorrectly marking as invalid.

5.26.4 getGeneratedKeys() and OutOfBounds Exception

Fixed in Release 7.3.1 Build 20111219.

The call to `ResultSet.getNext()` on a resultset returned by the `Statement.getGeneratedKeys()` may throw the following exception:

```
java.lang.ArrayIndexOutOfBoundsException: -1
```

This exception only occurs when the `getNext()` method is called on the resultset after a prior `getNext()` returned false indicating an end of recordstream.

The following code example will show this problem:

```
.
.
.
 ps2.execute();
 rs = ps2.getGeneratedKeys();
 if (rs == null)
    log ( "No resultset from execute");
 else
 {
    while (rs.next())
    {
.
.
.
    }
    rs.next(); // this will throw an array exception
 }
.
.
.
```

This has now been fixed.

5.26.5 Multi-process Server Fails When Persona Used

Fixed in Release 7.3.1 Build 20111221.

A problem introduced in Release 7.3.0.2 may cause the Multi-process server to terminate while attempting to start an executor process.

This problem only occurs if the server has the Persona attribute set.

If the Persona attribute is designated for the server, executor processes will be created using the correct Persona. However a problem in determining the process identifier of the executor process may cause an Access Violation within the server process which will terminate the Server image.

The following is an example of the type of exception raised:

```
#
# An unexpected error has been detected by HotSpot Virtual Machine:
#
#  %SYSTEM-F-ACCVIO (0xc) at pc=35C5481, pid=564797798, tid=2070302400
#
# Java VM: Java HotSpot(TM) Server VM (1.5.0.20 08/30/2010-13:43 IA64 mixed
mode)
# Problematic frame:
# C  [RDBJDBCMPSHR73+0xffffffb2]
#
# An error report file with more information is saved as
hs_err_pid564797798.log
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000000, PC=000000000113DC30, PS=0000001B
%TRACE-F-TRACEBACK, symbolic stack dump follows
image     module    routine               line      rel PC           abs PC
JAVA$HOTSPOT_SHR  OS  print_hex_dump    133224 0000000000001EF0
000000000113DC30
.
.
.
----- Above condition handler called with exception 0000000C
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000010, PC=00000000035C5481, PS=0000001B
----- End of Exception message
                                          0 FFFFFFFF80393F42
FFFFFFFF80393F42
RDBJDBCMPSHR73  RJDBC_PROC  rjdbc_proc_GetPid
                                       42606 00000000000029A1
00000000035C5481
RDBJDBCMPSHR73  RDBJDBC  Java_rdb_JNI_StartProcess
```

The only workaround is to not use Persona for Multi-process servers.

This has now been fixed.

5.26.6 RestrictAccess Being Applied Even When Disabled

Fixed in Release 7.3.1 Build 20120201.

The Oracle JDBC for Rdb User Guide specifies that if the server attribute `restrictAccess` is not defined or is set to `false` then database and user restrictions will not be carried out.

A problem in how server restriction policy was applied may cause restrictions to be applied even if the server attribute `restrictAccess` is not defined or is set to `false`.

If there is at least one `allowDatabase` element in the server specification then database restrictions were applied to the server. Similarly, the presence of `allowUser` entries would cause the appropriate restrictions to be applied irrespective of the `restrictAccess` settings.

This has now been fixed.

5.26.7 Multi-process Problem in the Native Driver

Fixed in Release 7.3.1 Build 20120208.

Due to changes in thread interleaving introduced in Java JDK6.0 it is possible that applications using the `multiprocess` option of the rdbNative driver connections may no longer work correctly when run in a JDK6.0 environment.

Interference may occur between simultaneously running threads accessing JDBC connections within the same application object instance which may cause Access Violations within the application.

The nature of the problem depends very much on where the interference occurs and what each thread was trying to do at the time. However more generally the problems may be seen during connection startup.

Various Access Violations, problems with OpenVMS Event Flags and Pthread mutex problems have been noted during connection, with the subsequent prevention of JDBC connections being made correctly or resulting in the hanging or the termination of the running application.

The JDBC rdbNative driver has now been changed to carry out more aggressive thread locking at the initial handshake phase of the JDBC connection with the Multi-process executor, to prevent these inter-thread problems.

This problem affects the rdbNative driver only. Applications using the rdbThin driver connecting to Multi-process servers are not affected.

▲contents

5.27 New Features for Release 7.3.0.2

This section describes new and changed features in Oracle JDBC for Rdb 7.3.0.2.

5.27.1 Server Network Keep Alive

Release 7.3.0.2.

The ability to set SO_KEEPALIVE on the socket used on client–side to communicate with a server has been available for a number of releases. This release introduces the same feature on the server-side socket used to communicate back to the client.

See `srv.networkKeepAlive` in the *Server Configuration Options* section of the *Oracle JDBC for Rdb User Guide* for more information.

5.28 Corrections in Release 7.3.0.2

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.0.2.

5.28.1 Null Pointer Exception During Server Viability Check by Pool Server

Fixed in Release 7.3.0.2 Build 20110223.

When the Pool server checks to see if a pooled server is available and has a free client slot, it is possible that a problem in a disconnect code feature that was added for the Oracle JDBC for Rdb Release 7.3 upgrade may cause a `NullPointerException` to be raised by the pooled server.

The pooled server recovers from this exception, but an entry showing the exception is written to the server log.

This has now been fixed.

5.28.2 MINUS and INTERSECT Problem

Fixed in Release 7.3.0.2 Build 20110303.

The use of the set operators MINUS and INTERSECT may cause the JDBC SQL parser to incorrectly allocate dbkey and alias variables within the compound statement which may raise SQL syntax errors when the statement is compiled by Rdb.

For example, an SQL syntax error will be raised by the following simple query:

```
select * from employees MINUS select *  from employees;
```

```
SQLException: in <rdbjdbcsrv:prepare_stmt>
F-RELNOTDEF, Table MINUS is not defined in database or schema:42000
```

This has now been fixed.

5.28.3 Memory Leak When Executor Process Fails to Run

Fixed in Release 7.3.0.2 Build 20110419.

A small memory leak has been found in the Multi-process(MP) server when the server attempts to start a new executor process but fails due to a System or a resource problem. An example of a resource limitation may be insufficient process slots for the additional executor process.

The MP Server correctly signals the exception. However it then fails to recover a small amount of memory (157 bytes) which may cause process memory depletion if this situation is repeated many times within the life-time of the server.

This has now been fixed.

5.28.4 Lost Executor Not Caught by Multi-process Server

Fixed in Release 7.3.0.2 Build 20110623.

If an executor process terminates unexpectedly while still connected to an active connection, it is possible that the Multi-process server parent process may not get notification of the terminations and will wait indefinitely for the executor process to respond.

This has now been fixed.

5.28.5 Multi-process Native Driver Problem

Fixed in Release 7.3.0.2 Build 20110623.

During the creation of connection contexts when using the `multiprocess` feature of the Native driver, the driver did not adequately synchronize the connect operation.

This may result in unexpected and incorrect behaviour during connection if multiple threads are concurrently trying to connect to databases within the same process.

This has now been fixed.

5.28.6 PreparedStatement.getGeneratedKeys() Problem

Fixed in Release 7.3.0.2 Build 20110712.

On the second or subsequent execution of a PreparedStatement, the call to the method getGeneratedKeys() will fail with the following exception:

```
oracle.rdb.jdbc.common.RdbException: Closed Resultset
```

This has now been fixed.

5.28.7 Excessive IOs During Metadata Retrieval

Fixed in Release 7.3.0.2.2 Build 20111002.

Several SQL queries used by JDBC to retrieve metadata associated with compiled queries may cause excessive IOs on database systems where the column or table definitions are volatile and have had many revisions during the life of the database.

This problem was introduced in 7.3.0.2 and has now been fixed.

5.28.8 Syntax Error SQL-F-CONVARUND

Fixed in Release 7.3.0.2.3 Build 20111005.

A problem in the determination of the correct table or alias name to use for dbkey references in the additional SQL syntax added by JDBC to allow correct record identification may produce SQL syntax exceptions to be raised.

This problem prevents the query from executing correctly, for example:

```
select * from employees e left join degrees on
     (degrees.employee_id = e.employee_id)
```

will raise the following error:

```
SQLException: SQLState(RR000) vendor code(-1)
@ oracle.rdb.jdbc.common.RdbException: in <rdbjdbcsrv:prepare_stmt>
@ %SQL-F-CONVARUND, Column qualifier EMPLOYEES is not defined
```

This problem only occurs in queries that reference two or more tables.

This problem was introduced in 7.3.0.2 and has now been fixed.

5.28.9 Multi-process Server Hang

Fixed in Release 7.3.0.2.4 Build 20111006.

The Multi-process (MP) server may hang during Resultset close operations.

A problem introduced in 7.3.0.2 may cause the server to try to execute code outside the standard application space and cause either a hang in the server process or an Access Violation.

The MP Server may encounter this problem at any time while trying to close ResultSets , however in the majority of cases the ResultSet close will operate correctly without problem.

The problem may be seen more readily in very active MP Server processes.

Spurious Rdb Dispatcher errors have also been noted due to this problem.

This problem was introduced in 7.3.0.2 and has now been fixed.

5.28.10 SSL Socket Intrusion Problem

Fixed in Release 7.3.0.2.5 Build 20111102.

Data injection into an existing open SSL socket may cause problems with SSL-enabled Thin Servers.

Socket and Port security scanning software may inject data into existing open SSL socket channels that may cause SSLExceptions on that socket. Depending on the state of the Thin Server at the time of the injection, this intrusion may be handled incorrectly causing the server process to use excessive CPU.

The SSL connection is not compromised by the intrusion attempt, and the underlying database connection is not affected.

This has now been fixed.

▲contents

5.29 New Features for Release 7.3.0.1

This section describes new and changed features in Oracle JDBC for Rdb 7.3.0.1.

5.29.1 Reopen Server Log Files Using Poll Subcommand

Release 7.3.0.1.

The thincontroller now has the capability to request servers to re-open their log files allowing the previous version of the log file to be examined or copied.

Due to a restriction within Java on OpenVMS, log files opened by a Oracle JDBC for Rdb server cannot be be read or copied while the log files are currently being used by the server.

The thincontroller from Oracle JDBC for Rdb Release 7.3.0.1 (or later), used in conjunction with Oracle JDBC for Rdb servers from Release 7.3.0.1 (or later) will allow a control user to request that the server re-open its log file by using the POLL REOPENLOGS subcommand.

See the *POLL Sub-commands* section of the *Oracle JDBC for Rdb User Guide* for more information.

▲contents

5.30 Corrections in Release 7.3.0.1

This section describes software errors corrected in Oracle JDBC for Rdb release 7.3.0.1.

5.30.1 Show Clients Not Showing Column Names

Fixed in Release 7.3.0.1 Build 20100315.

The *Show Clients* command within the Controller displays information about clients currently using JDBC servers.

A problem introduced in Oracle JDBC for Rdb Release 7.3 prevents the column header information from being displayed correctly.

The data for each client is still displayed correctly but without column identification.

```
rdbthincontrol> show active clients

Server rdbthnsrv3 (//192.168.1.3:1706/)

00000004*
<CONTROL CONNECTION>
jdbc_user
192.168.1.3:1459
0x16A0(5792)
0x16A0(5792)

2010-03-15 09:59:59.296 : INIT_CONTROL

0 00:00
0
1970-01-01 10:00:00.0
rdbthincontrol>
```

The following shows how the display should look:

```
rdbthincontrol> show active clients

Server rdbthnsrv3 (//192.168.1.3:1706/)

RDB$CLIENT_ID                 : 00000006*
RDB$URL                       : <CONTROL CONNECTION>
RDB$USER                      : jdbc_user
RDB$IP                        : 192.168.1.3:1665
RDB$PID                       : 0x16A0(5792)
RDB$PID_AT_EXECUTOR           : 0x16A0(5792)
RDB$LAST_SQL                  :
RDB$LAST_ACTION               : 2010-03-15 10:55:01.312 : INIT_CONTROL
RDB$LAST_EXCEPTION            :
RDB$TIME_SINCE_LAST_ACTION    : 0 00:00
RDB$MINUTES_SINCE_LAST_ACTION : 0
```

```
RDB$LAST_OPEN                  : 1970-01-01 10:00:00.0
rdbthincontrol>
```

5.30.2 Batched Statement Fails With MULTIPLE_RECORDS Exception

Fixed in Release 7.3.0.1 Build 20100611.

To provide additional JDBC driver functionality in Oracle JDBC for Rdb Release 7.3, during the preliminary parsing of SQL update or insert statements, the Oracle JDBC for Rdb drivers may append extra SQL syntax to the query to deliver the dbkey of the updated or inserted record back to the driver for later processing.

If during the execution of that query, Rdb raises a MULTIPLE_RECORDS exception, the JDBC drivers will re-issue the statement again without the dbkey retrieval clause, allowing the update statement to complete successfully.

If, however, the update statement is issued as a batched statement by using the `Statement.addBatch()` method, the additional dbkey retrieval code may cause that batch statement to fail. In this case Oracle JDBC for Rdb does not re-issue the query and the update will not be done.

The Rdb JDBC drivers have been changed to not add the extra dbkey retrieval code if the statement is being processed as a batched statement.

5.30.3 Release 7.3 Thin Driver and Prior Servers Incompatibility

Fixed in Release 7.3.0.1 Build 20100611.

Ideally, when upgrading Oracle JDBC for Rdb, all components on both the server systems and all client systems should be updated to the same version at the same time.

Although not documented nor supported as a feature, Oracle JDBC for Rdb attempts to maintain upward and downward compatibility between its servers and the Rdb Thin drivers.

However a change in the initial connection hand-shake between the Thin driver and server introduced in Release 7.3 may prevent applications using the Release 7.3 Thin driver from connecting correctly to Oracle JDBC for Rdb servers from earlier Oracle JDBC for Rdb versions.

The connection of applications using Thin drivers from releases prior to Release 7.3 to a Release 7.3 server should still work correctly.

These incompatibility issues have now been resolved.

5.30.4 Global Memory Leak When Executors are Run-down

BUG 9787051
Fixed in Release 7.3.0.1 Build 20100611.

A problem in the release of global shared memory when an executor process is run-down by a Multi-process (MP) server may cause the server to eventually run out of global memory.

If this happens the following exception will be raised:

```
System Error : Insufficient global memory
```

When an MP Server runs out of global memory it cannot start-up new executor processes.

As this problem occurs when an executor is run-down and is no longer required, a workaround for this problem is to increase the `maxFreeExecutor` count for the server to a number that will be sufficient to handle the peak load of the server and to increase the server `sharedMem` appropriately.

This has now been fixed.

5.30.5 Autorestart on Pooled Servers Not Working

BUG 9656632
Fixed in Release 7.3.0.1 Build 20100611.

When a Pool server starts a child server that is marked for autorestart, the child server information is placed on a queue that is periodically checked by the Pool server to ensure the child server is still viable.

If the Pool server finds that the child is not available and that child server is marked for autorestart, the Pool server will try to restart that server.

A problem was introduced in Oracle JDBC for Rdb Release 7.3 within the Pool server code that caused the Pool server to incorrectly mark a newly started child server as “failed in startup”.

If the pool server thinks that the child server did not start correctly the server will not be placed in the check queue used to determine if the child server is still available.

This in turn means that the pool server will not automatically restart the child server if the child server process subsequently dies.

This has been fixed.

5.30.6 setFetchSize() Hint Ignored by PreparedStatement

Fixed in Release 7.3.0.1 Build 20100623.

In prior versions of Oracle JDBC for Rdb, the execution of `setFetchSize()` method on a `PreparedStatement` would be silently ignored.

The connected server process would determine the fetch size of subsequent operations on the PreparedStatement and ignore the fetch size hint provided.

This behavior has now been changed. The fetch size hint provided by the execution of the `setFetchSize()` method on a `PreparedStatement` prior to the execution of the statement may now be taken into account by the server when determining the number of records that should be sent in one network IO operation.

As this is only a hint, the connected server may still choose to ignore it.

5.30.7 Exception on Subsequent Execution of PreparedStatement

Fixed in Release 7.3.0.1 Build 20100623.

The second or subsequent execution of a `PreparedStatement` may raise the following exception:

```
java.sql.SQLException: Missing Stmt in rdbSelect
   @rdb.Client.SELECT
```

This exception may be raised on any subsequent executions of the same PreparedStatement object if the prior execution of the same statement retrieved a greater number of records than the connected server's fetch size.

The following example will show this errant behavior if the fetchSize specified for the connected server is less than 10 rows:

```
.
.
.
 try
 {
       ps = conn.prepareStatement(
             "select last_name from employees limit to 10 row");

       ResultSet rs = ps.executeQuery();
       while (rs.next())
       {
          System.out.println("  result1 <" + rs.getString(1) + ">");
       }
       rs = ps.executeQuery(); //<<<<<< this would fail
       while (rs.next())
       {
          System.out.println("  result2 <" + rs.getString(1) + ">");
       }
 }
 finally
 {
    if ( ps != null ) ps.close();
 }
.
.
.
```

This problem was introduced in Release 7.3. Oracle JDBC for Rdb and only occurs when using the Oracle JDBC for Rdb thin driver.

5.30.8 EOFException on READ_ROW With Nested Statements

Fixed in Release 7.3.0.1 Build 20100629.

A problem in statement context handling may cause the following exception to be raised:

```
java.io.EOFException @rdb.Client.READ_ROW
```

This problem may occur when the following conditions are met.

1. A select query is executed as the outer query
2. An update statement is executed as an inner statement
3. The fetch size of the select query is less than the total number of records returned by the query
4. Fetch Size number of records have already been returned by the outer query

An example of code which may show this problem is shown below:

```
.
.
.
 Statement stmt = conn.createStatement();
 stmt.setFetchSize(10); // less than the total number of records
 PreparedStatement ps = conn.prepareStatement (
   "update job_history set supervisor_id = '00245' where "+
   "employee_id= ?");
 ResultSet rset = stmt.executeQuery(
    "select employee_id from employees limit to 20 rows");
 int i = 0;
 try
 {
    while (rset.next())
    {
       i = i + 1;
       System.out.println(i + "     " +
                          rset.getString("employee_id"));
       ps.setString(1, rset.getString("employee_id"));
       ps.executeUpdate();
    }
 }
 catch (SQLException sqle)
 {
    sqle.printStackTrace();
 }
.
.
.
```

5.30.9 DatabaseMetaData.getUDTs()

Fixed in Release 7.3.0.1 Build 20100702.

Although Rdb does not currently support UDTs you may still retrieve the column descriptions from the empty `Resultset` produced when the method `DatabaseMetaData.getUDTs()` is called.

However a problem in the retrieval of the last column from the `ResultSetMetaData` may result in a Null Pointer exception.

```
java.lang.NullPointerException
```

This has now been fixed.

5.30.10 Using Multiprocess With the JDBC Native Driver

Fixed in Release 7.3.0.1 Build 20100718.

A problem was introduced in Oracle JDBC for Rdb Release 7.3 preventing the use of the `multiprocess` option within JDBC Native driver connections.

When a connection is attempted using the JDBC Native driver with a connection URL that contains the `@multiprocess=true` switch the following exception was raised:

```
INFO: server version needs to be V7.3 or later
```

This has now been fixed.

5.30.11 Prepared Statement Not Closing Underlying Cursor

Fixed in Release 7.3.0.1 Build 20100721.

A problem introduced in Oracle JDBC for Rdb Release 7.3 prevents PreparedStatements containing select statements from correctly closing cursors when the statement is re-executed but no results set read was issued on the previous execution of the same prepared statement instance.

The following exception may be raised:

```
%SQL-F-CURALROPE, Cursor C_xxxxxxxxx was already open
```

The following code snippet shows an example of the type of code that may show this problem:

```
.
.
.
String sql = "select dbkey from employees";
ResultSet rs;
int maxi = 10;
PreparedStatement st=conn.prepareStatement(sql);
for(int j=0;j<maxi;j++)
{
   rs=st.executeQuery();
   rs.close(); // note: no read on the resultset
               // issued within the iterating block
}
.
.
.
```

The problem would be seen on the second and subsequent iterations of the `executeQuery`.

This has now been fixed.

5.30.12 Release Statement Synchronization Problem

Fixed in Release 7.3.0.1 Build 20100802.

Oracle JDBC for Rdb release Release 7.3 introduced improvements in the network handling of selection statements. A problem in the optimization of network IOs introduced in Release 7.3 may produce the following exception during statement release:

```
oracle.rdb.jdbc.common.RdbException:  received 503316549(0x4500001E)
 : expected 1 @rdb.Client.RELEASE_STMT
```

This problem may occur only when the following conditions have been met:

1. A select statement has been executed.
2. Statement select optimization has not been turned off.
3. The total size of a FetchSize number of records is greater than 9600 bytes.
4. The statement is closed before any records have been read from the associated resultset.

This has now been fixed.

5.30.13 Unitialized SQLCA Block in MP Server

Fixed in Release 7.3.0.1 Build 20100906.

A problem in how internal SQLCA blocks were initialized by the Multi-process server may cause a nested exception to be raised during exception handling.

The SQLCA block failed to initialize correctly which may cause incorrect message size information to be used when the server tries to dump out the underlying error message, which in turn may cause the server to try to access incorrect memory addresses.

This problem may lead to various access violation exceptions including :

```
%CXXL-F-TERMINATE, terminate() or unexpected() called

Improperly handled condition, image exit forced by last chance
 handler.
```

This has now been fixed.

5.30.14 Controller SHOW CLIENTS and MP Server Problem

Fixed in Release 7.3.0.1 Build 20100315.

Note

This problem was reported fixed in Oracle JDBC for Rdb Release 7.3 but unfortunately, the fix was incomplete.

When the command SHOW CLIENTS or SHOW ALL CLIENTS is issued within the Controller, if the recipient server is a Multi-process server (MP server), an executor process may be started up by the server to execute the request.

A problem in the executor code prevents the executor from closing down correctly if the maximum number of free executors has already been reached for the MP Server. This new executor process will not be added to the free executor pool, but will not be shutdown and thus will remain active on the system.

If the controller SHOW CLIENTS command is issued again, the number of executor processes may increase until an Open VMS quota or process quota is exceeded.

This problem has now been fixed.

5.30.15 Documentation Error – Record Streaming

In the Oracle JDBC for Rdb Release 7.3 release notes the Performance Enhancements subsection of the New Features section mentions that Record Streaming is available from Release 7.3 onwards.

Unfortunately a number of problems required the removal of this feature prior to the 7.3 Release, but the Release 7.3 releases notes were not correctly updated to remove reference to the feature prior to product shipment.

This feature may be made available in a later release of Oracle JDBC for Rdb.

5.30.16 ResultSet.updateRow() and ResultSet.deleteRow() Problem

BUG 9668574
Fixed in Release 7.3.0.1 Build 20110127.

A problem introduced in Release 7.3 prevents ResultSet.updateRow() and ResultSet.deleteRow() from working correctly.

During the execution of the update one of the following errors may be raised:

```
Internal Error
```

or

```
SQL-F-SYNTAX_ERR, SYNTAX ERROR
```

This has now been fixed.

5.30.17 Problem Using Column Renaming With Dbkey

Fixed in Release 7.3.0.1 Build 20110127.

Changes to SQL statetment parsing introduced in Release 7.3 inadvertantly introduced a problem in queries that access the dbkey of records directly.

For example the following query:

```
select dbkey as ID, last_name from employees;
```

may cause the following exception to be raised:

```
Exception: Unhandled exception@Statement.execute :
java.lang.ArrayIndexOutOfBoundsException:S1000
```

This problem only occurs when the dbkey is retrieved as the first named column of the selection expression and the column name is changed by using the ‘AS’ clause.

Some simple workarounds are:

1. move the dbkey clause to the second or subsequent column reference:

```
select last_name, dbkey as ID from employees;
```

2. Remove the column renaming:

```
select dbkey, last_name from employees;
```

This has now been fixed.

5.30.18 Cast Error on Scaled Integer After ResultSet.insertRow()

Fixed in Release 7.3.0.1 Build 20110128.

During the conversion of a scaled integer invoked by a ResultSet object retrieval the following error may be raised:

```
oracle.rdb.jdbc.common.RdbException: Connection lost
    :   java.lang.Long
   @rdb.Client.GET_INT_VAL
```

This problem occurs when:

- The object retrieved is a scaled integer
- Immediately prior to the retrieval a ResultSet.insertRow() was carried out on the same ResultSet

The following code example will show this problem during the call to the getInt() method:

```
.
.
.
Statement stmt = conn.createStatement();
ResultSet rset = stmt.executeQuery("select * from JOBS");
rset.moveToInsertRow();
rset.updateString("JOB_CODE", "JAVA");
rset.updateString("WAGE_CLASS","1");
rset.updateString("JOB_TITLE","Java Tester");
rset.updateInt("MINIMUM_SALARY",90000);
rset.updateInt("MAXIMUM_SALARY",250000);
rset.insertRow();

while (rset.next())
{
    System.out.println(rset.getString("JOB_TITLE"));
```

```
     System.out.println( "min salary = " +
     rset.getInt("MINIMUM_SALARY"));
 }
.
.
.
```

This has now been fixed.

5.30.19 ResultSet.deleteRow() Behaviour Change

Fixed in Release 7.3.0.1 Build 20110201.

Prior to the V4.0 JDBC specification, the current row positioning after resultset-based record deletion was not clearly defined in the JDBC specification.

In prior releases of Oracle JDBC for Rdb the row position would remain at the same absolute position after the row has been removed. Thus, for example, if the deletion row was the 10th row, then after removing the record, the current row position would still be the 10th record within the resultset, effectively moving the remaining records down one position.

This may cause unexpected behaviour while traversing the resultset. For example, the following code will delete every second record in the resultset as after each deletion the remaining rows would be shifted down one position:

```
.
.
.
 Statement stmt = conn.createStatement(
 ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);

 ResultSet rset = stmt.executeQuery ("SELECT * FROM employees");
 while (rset.next ())
 {
      System.out.println("Select: " + rset.getString (1) );
      rset.deleteRow();
 }
.
.
.
```

To remove each row the row position would have to be reset each time back to the prior row:

```
.
.
.
 Statement stmt = conn.createStatement(
```

```
ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);

ResultSet rset = stmt.executeQuery ("SELECT * FROM employees");
while (rset.next ())
{
   System.out.println("Select: " + rset.getString (1) );
   rset.deleteRow();
   rset.previous();
}
.
.
.
```

In V4.0 of the JDBC specification, the row position after deletion using the `ResultSet.deleteRow()` is now specified:

> After the method deleteRow has been called, the cursor will be positioned before the next valid row. If the deleted row is the last row, the cursor will be positioned after the last row.

Oracle JDBC for Rdb drivers have now been changed to comply with this behaviour. The resetting of the row position using `rset.previous()` as shown in the example above is no longer required.

5.30.20 Blob Handles Not Released When Using ResultSet.getBlob()

Fixed in Release 7.3.0.1 Build 20110201.

The `ResultSet.getBlob()` method may be used to return the contents of an Rdb segmented string column to your application as a Blob object.

A problem in the underlying Rdb statements associated with the retrieval of segmented strings prevented the handles from being released correctly.

This may lead to memory problems in the Oracle JDBC for Rdb server or the Java application if many Blobs are being retrieved.

In addition, as these statements are compiled and not released, operations such as DROP TABLE may be blocked if attempted subsequent to and within the same connection as Blob operations.

The following exception may be raised:

```
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-ACTQUERY, there are queries compiled that reference relation
…
```

This has now been fixed.

5.30.21 Sequence Values Not Visible When Using Column Name

BUG 9659036
Fixed in Release 7.3.0.1 Build 20110204.

The retrieval of data from ResultSet columns using any of the ‘by name’ methods fail when the column is one of the special sequence operations such as nextval or currval.

The following code will raise an exception when the second println statement is called:

```
.
.
.
 ResultSet rs = s.executeQuery(
   "select reportseq.nextval from rdb$database" );
 if (rs.next())
 {
    System.out.println(" by id   = " + rs.getBigDecimal(1) );
    System.out.println(" by name="+
    rs.getBigDecimal("reportseq.nextval"));
 }
.
.
.
```

The following exception will be raised:

```
Invalid column name
```

A similar problem may be seen when trying to retrieve column names using the `ResultSetMetaData.getColumnName()` method:

```
.
.
.
 ResultSet rs = s.executeQuery(
      "select reportseq.nextval from rdb$database" );
 ResultSetMetaData rsMetaData = rs.getMetaData();
 String columnName = rsMetaData.getColumnName(1);
 while (rs.next ())
 {
    System.out.println("Select: " + rs.getBigDecimal(columnName));
```

```
 }
 rs.close();
.
.
.
```

The column name returned by the ResultSetMetaData method will be an empty string and subsequent use of this value will fail to retrieve any column information.

As the first example shows, a workaround for the first problem is to use the column index instead of the name in the get method.

An alternate workaround is to use a column alias in the query.

```
.
.
.
 ResultSet rs = s.executeQuery(
       "select reportseq.nextval as NEXT_ID from rdb$database" );
 if (rs.next())
 {
     System.out.println("by name = " +rs.getBigDecimal("NEXT_ID"));
 }
.
.
.
```

There is no workaround for the `ResultSetMetaData.getColumnName()` problem.

These problems have now been fixed.

5.30.22 Scaled Integer Problem

Fixed in Release 7.3.0.1 Build 20110208.

The Oracle JDBC for Rdb native driver may fail to return the values of scaled integer columns correctly.

If the column size is less than 4 octets and has a scaling factor other than 0 the native driver may return wrong values.

Columns with the following datatypes are affected:

- SMALLINT(n)
- TINYINT(n)

This problem only occurs when using the native driver; the thin driver will return the values correctly.

SMALLINT and TINYINT columns not using scaling are not affected.

This has now been fixed.

5.30.23 Server Matching Exception May Stop Poll Handling

Fixed in Release 7.3.0.1 Build 20110210.

If a server finds an exception during the handling of a server control function such as a POLL request, the server will log the exception and then close asynchronous broadcast request handling.

This means that from that point onwards, operations such as ThinController POLLing will no longer be acknowledged by the server.

Standard client and control function are not affected, and the server will keep on handling client requests correctly, however it will no longer respond to control requests such as POLL.

A simple example of this behavior can be shown by issuing a server name matched POLL request containing an invalid pattern string:

```
rdbthincontrol> poll #name:*srv*.
```

The server will log the following exception:

```
Dangling meta character '*' near index 0
   *srv*.
   ^
```

The server will not respond to further POLL requests until it has been restarted.

This behavior has now been changed. The server will still log the exception but unless the problem is a network IO problem or the server is in the process of shutting down, the asynchronous broadcast request handling will still be enabled. Thus, operations such as POLL request handling will still be carried out by the server.

The only workaround is to restart the server again.

This has now been fixed.

▲contents

5.31 New Features for Release 7.3

This section describes new and changed features in Oracle JDBC for Rdb 7.3.

5.31.1 Shutdown Thread

Starting with Oracle JDBC for Rdb Release 7.3 the JDBC drivers will create a shutdown thread when they are initially invoked. The purpose of this thread is to use the shutdown-hook feature provided by OpenVMS that will execute the thread at shutdown. The shutdown thread will ensure that any connection left open by the application will be correctly disconnected prior to the application shutdown proceeding, thus preventing application hangs at shutdown.

The application developer does not need to change any code for this shutdown feature to be enabled, as long as the application is using Oracle JDBC for Rdb Release 7.3 driver libraries (or later versions) the shutdown hook will be in place.

See the section *Shutdown Thread* in the *Oracle JDBC for Rdb User Guide* for details.

5.31.2 Driver.attach()

Release 7.3.

A new method has been added to the Oracle JDBC for Rdb drivers, that will allow second and subsequent databases to be attached to an existing database connection.

See the *Oracle JDBC for Rdb User Guide* for details.

5.31.3 Returning List of Known Databases

Release 7.3.

The Oracle JDBC for Rdb Thin Driver will now allow client applications to obtain a list of databases known to Oracle JDBC for Rdb servers.

See the *Oracle JDBC for Rdb User Guide* for details.

5.31.4 Controller Enhancements

Release 7.3.

Several enhancements have been made to the Oracle JDBC for Rdb Thin Controller application including:

- Obfuscate command
- Server matching patterns for the POLL command

See the *Oracle JDBC for Rdb User Guide* for details.

5.31.5 RDB_EXT.JAR file

Release 7.3.

Oracle JDBC for Rdb Release 7.3 installation now includes the RDB_EXT.JAR file that contains classes and Java source for extensions to Oracle JDBC for Rdb.

The following extensions are included:

- Hibernate RdbDialect.java source
- Oracle UCP enabling classes

Details of these extensions follow.

5.31.5.1 Hibernate RdbDialect.java Source

Release 7.3.

Hibernate by Red Hat is a persistence engine that provides an alternative to standard entity beans.

To allow Oracle JDBC for Rdb drivers to be used in conjunction with Hibernate the RdbDialect.java source file is provided. You will find this source file in the top-level folder of the RDB_EXT.jar.

See the Hibernate documentation for details on using third party JDBC drivers and Dialects.

5.31.5.2 Oracle UCP Enabling Classes

Release 7.3.

UCP or, Universal Connection Pool, is a new database feature included in Oracle database 11*g* 11.1.0.7, Oracle database 11.2.0.x and Oracle AS 11*g* R1. UCP works with any Java based connections, e.g., JDBC, JCA, LDAP.

The RDB_EXT.JAR file installed with Oracle JDBC for Rdb, provides the necessary classes to enable Oracle JDBC for Rdb drivers to be used in the connection pools created by UCP.
The class:

```
oracle.rdb.jdbc.rdbExt.RdbDataSource
```

found in the RDB_EXT.JAR may be used as a Connection Factory class for UCP, for example:

```
import oracle.ucp.jdbc.PoolDataSource;
import oracle.ucp.jdbc.PoolDataSourceFactory;
.
.
.
        PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
        pds.setConnectionFactoryClassName(
               "oracle.rdb.jdbc.rdbExt.RdbDataSource");
        pds.setURL("jdbc:rdbThin://localhost:1701/my_dir:mf_personnel");
        pds.setUser("jdbc_user");
        pds.setPassword("jdbc_user");
        pds.setInitialPoolSize(0);
        Connection conn2 = null;

        for (int i = 0; i< 5 ; i++ )
        {
            conn2 = pds.getConnection();
            Statement sc = conn2.createStatement();
.
.
.
            rs.close();
            sc.close();
            conn2.close();
        }
.
.
.
```

For more details, see the UCP Users Guide at

http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html

5.31.6 Performance Enhancements

Release 7.3.

Several enhancements have been made to the client/server connectivity and to server/executor communications to improve performance including:

- Internal client/server message protocol changes
- Enhanced SQL Statement caching
- Results caching

- Server/executor synchronization

Details of these enhancements may be found in the following sub-sections.

5.31.6.1 Internal Client/Server Message Protocol Changes

Release 7.3.

Network round-trip times for communication between client applications and JDBC servers may constitute a large portion of wait time for applications. To help reduce this wait time Oracle JDBC for Rdb has changed its client/server message protocol in order to reduce the number of round-trips taken to:

- Attach to the database
- Compile and execute a SQL statement.

These enhancements are enabled automatically from V7.3 onwards.

5.31.6.2 Enhanced SQL Statement Caching

Release 7.3.

As well as caching of certain statement information on the client-side, reducing the number of network IOs required for statement compilation, 7.3.0.0 allows the caching of statement handles. See the sections *SQL Statement Cache* and *Caching Statement Handles* in the *Oracle JDBC for Rdb User Guide* for more details.

5.31.6.3 Results Caching

Release 7.3.

Oracle JDBC for Rdb Release 7.3 will allow limited caching of resultSets on the client side. This is particulary useful in multi-tiered environments for statements that are repeatedly executed giving the same results each time, especially in a pooled connection environment. See the section *Results Cache* in the *Oracle JDBC for Rdb User Guide* for more details.

5.31.6.4 Server/Executor Synchronization

Release 7.3.

Starting with Oracle JDBC for Rdb release 7.3, the method used to synchronize operations between a Multi-process server and its sibling executors has been improved to reduce the amount of CPU required and to speed-up operations.

▲contents

5.32 Corrections in Release 7.3

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.

5.32.1 Server Startup Failure When Using CFG File

Fixed in Release 7.3 Build 20090201.

A problem in parsing the configuration file during server startup may cause the server to fail to startup correctly.

The following is an example of the error message seen:

```
$ java "-jar" "rdb$jdbc_home:rdbthinsrvpool.jar" -
 "-cfg" rdbjdbc_cfg.cfg "-srv.mcGroupIP" "239.8.124.3"
Configuration file problem at line 1
Content is not allowed in prolog.
```

This problem only occurs when using JDK 1.5-0 and above and using a Properties file (*.cfg) to hold the server configuration properties.

A workaround for this problem is to use an XML-based configuration file instead.

5.32.2 AccessViolation on Disconnect When Inserting Blobs

Fixed in Release 7.3 Build 20090301.

A problem in determining the memory allocation for Blob variables may result in a memory Access Violation during disconnect.

The problem only occurs if during the connection, data was inserted into Blob variables and subsequently inserted into a segmented string column in the database.

This has now been fixed.

5.32.3 Parameter Marker Problem Now Resolved

Fixed in Release 7.3 Build 20090702.

In previous versions of Oracle JDBC for Rdb the following known problem was noted:

Using PreparedStatement and Parameter Markers

During the creation of a prepared statement using the `Connection.PrepareStatement()` method, the Oracle JDBC for Rdb drivers call Oracle Rdb SQL to compile the SQL statement and describe its select fields and parameter markers. At this time SQL builds internal message representations of the parameter markers that may be passed to Oracle Rdb when the prepared statement is executed.

The maximum size of character values that may be passed using each parameter marker is fixed by SQL at this stage. This may cause inconsistent results when the application attempts to use character string values that are longer than the maximum size determined by SQL for that parameter.

If the input value is longer, the value will be truncated by SQL prior to being sent to Oracle Rdb for processing.

This does not pose any problems if the query selection is equality, however, other Boolean comparisons may cause unexpected results. For example, this query will return the record:

```
.
.
.
 Statement stmt = conn.createStatement();

 stmt.execute("create table tab (f1 char(3))");

 stmt.execute("insert into tab values ('123')");
 PreparedStatement ps;
 ps = conn.prepareStatement( "select f1 from tab where f1 like ?");
 ps.setString(1, "123");
 ps.execute();
.
.
.
```

This query will not return the record:

```
.
.
.
  ps.setString(1, "%123");
```

```
  ps.execute();
.
.
.
```

The reason the above query fails is that SQL will set the maximum size of the parameter text string to 3 characters (the size of field F1). The input value will be truncated to %12 before being sent to Oracle Rdb and will not match the record.

In conjunction with changes made to Oracle Rdb Release 7.2.4, this problem has now been resolved and queries similar to the ones shown above should now deliver the correct results.

This fix will only work with Oracle Rdb version 7.2.4 and higher.

5.32.4 Controller SHOW CLIENTS and MP Server Problem

Fixed in Release 7.3 Build 20090821.

When the command SHOW CLIENTS or SHOW ALL CLIENTS is issued within the Controller, if the recipient server is a Multi-process server (MP Server), an executor process may be started up by the server to execute the request.

A problem in the executor code prevents the executor from closing down correctly if the maximum number of free executors has already been reached for the MP Server. This new executor process will not be added to the free executor pool, but will not be shutdown and thus will remain active on the system.

If the controller SHOW CLIENTS command is issued again, the number of executor processes may increase until an Open VMS quota or process quota is exceeded.

Note

This problem was not completely fixed in the 7.3 Release. This has subsquently been rectified in the 7.3.0.1 Release.

5.32.5 Possible Memory Leak When Updating Blob Columns

Fixed in Release 7.3 Build 20090827.

During the processing of update statements containing references to Blobs (list of byte varying) columns, the JDBC drivers incorrectly compile the statement twice.

Only the second statement handle is released when the statement is closed which will lead to Rdb statement handles still being active within the current connection.

These active handles use memory and resources within the drivers and the Rdb system which may eventually lead to problems due to insufficient resources in long running connections.

In addition these compiled queries may prevent metadata operations from occurring on the affected tables within the current connection.

5.32.6 Access Violation With Trace and Network Dump

Fixed in Release 7.3 Build 20090904.

When the following trace flags are set together in TraceLevel, it is possible that an access violation may occur if the size of the data being flushed to the network is very large:

Bit	Hexadecimal Value	Decimal Value	Traces
9	0x00000200	512	Network sends
30	0x40000000	1073741824	Full provides more details on certain flags

For example, when inserting a Blob with more than 3MB of data the dump of the network buffer may overflow memory and cause memory problems that will terminate the server unexpectedly.

This problem only occurs with RdbThin connections using either a standard or a Multi-process server.

5.32.7 Named Input Parameters not Working With CallableStatements

Fixed in Release 7.3 Build 20100210.

Using named parameters to set input parameter values of CallableStatements may fail with the following exception:

```
oracle.rdb.jdbc.common.RdbException: Invalid column name :
```

The JDBC code fails to locate input parameters by name correctly.

An example of the type of code statement that may be affected:

```
.
.
.
      stmt.executeUpdate("create module myproc " +
              "LANGUAGE SQL " +
              "procedure myproc(IN :val_in integer; OUT :val_out" +
              "BEGIN " +
              "    SET :val_out = :val_in; " +
              "END; END MODULE");

      CallableStatement proc = conn.prepareCall(
          "{ call myproc(?,?) }");

      proc.setInt("val_in",  999);
.
.
.
```

Exception raised when executing the proc.setInt() method:

```
oracle.rdb.jdbc.common.RdbException: Invalid column name : val_in
```

Input parameters are marked as IN in the procedure definition.

Parameters marked as OUT or INOUT will be located correctly by the JDBC code.

A workaround for the problem is to specify the parameter index value instead of the name.

▲contents

5.33 New Features for Release 7.2.5.5

This section describes new and changed features in Oracle JDBC for Rdb release 7.2.5.5.

None in this release.

▲contents

5.34 Corrections in Release 7.2.5.5

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.5.

5.34.1 Long-running Query Holds up New Connections in MP Server

Fixed in Release 7.2.5.5 Build 20091202.

A problem in retrieving the Process Id of the current thread during the creation of a new connection by the MP Server may cause the connection to hang.

This problem may be seen when an existing connection using the same MP Server is concurrently preparing a very large SQL statement or is executing a query that is taking substantial time within Rdb to fetch the very first record.

Any concurrent new connection requests will hang until the long-running query returns the query compilation results or returns the first record back to the MP Server.

All other pre-established connections using the same MP Server will execute queries normally and will not hang, only new connection requests are affected.

This has now been fixed.

▲contents

5.35 New Features for Release 7.2.5.4

This section describes new and changed features in Oracle JDBC for Rdb release 7.2.5.4.

None in this release.

▲contents

5.36 Corrections in Release 7.2.5.4

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.4.

5.36.1 Access Violation at Java_rdb_JNI_SetStrVal

Fixed in Release 7.2.5.4 Build 20080720.

A problem in a memory copy used within the JNI routine SetStrVal() may cause an access violation similar to the following:

```
 SYSTEM-F-ACCVIO, access violation, reason mask=00
.
.
.
 %TRACE-F-TRACEBACK, symbolic stack dump follows
 image    module    routine  line rel PC abs PC
.
.
.
 RDBJDBCSHR72  RDBJDBC  Java_rdb_JNI_SetStrVal
.
.
.
```

This problem is infrequent as it depends on how memory is allocated during the native code execution, and is usually only associated with JDBC code executing the **setString()** method of a **PreparedStatement or CallableStatement**.

This has now been fixed.

5.36.2 DCL Command Line Too Long

Fixed in Release 7.2.5.4 Build 20080826.

During the startup of a server either by a Pool Server or by using the Controller, DCL command procedures are used to build the DCL command to invoke the server image.

It is possible that long directory names or deep directory structures associated with the log file or the configuration files for the server may cause the size of the DCL command created to exceed the length limits set for a command line by OpenVMS.

In this case the startup of the server will fail and the following exception may be noted in the output of the server startup process:

```
%DCL-W-TKNOVF, command element is too long – shorten
```

The RDBJDBC_STARTSRV.COM has now been changed to reduce the number of options that need to present on the command line by introducing another configuration file that will be built each time the RDBJDBC_STATSRV.COM command procedure is executed. This new command procedure will contain the required configuration information necessary to correctly start the server.

5.36.3 Access Violation When Database Specification is Missing

Fixed in Release 7.2.5.4 Build 20080826.

If during a call to DriverManager.getConnection() the specified connection string does not contain a database file specification a problem in the building of the connection string to be sent to Rdb may force an unexpected termination of the connecting server with the following exception:

```
An unexpected exception has been detected in native code outside the VM.
Unexpected Signal : EXCEPTION_ACCESS_VIOLATION (0xc0000005) occurred at
 PC=0x18965EEB
Function=[Unknown.]Library= … \rdbjdbcshr.dll

NOTE: We are unable to locate the function name symbol for the error
     just occurred. Please refer to release documentation for possible
     reason and solutions.
Current Java thread:
       at rdb.JNI.Connect(Native Method)
       at
oracle.rdb.jdbc.common.AbstractNativeRdb.Connect(AbstractNativeRdb.java:301)
       at
oracle.rdb.jdbc.srv.DBActionHandler.handleConnect(DBActionHandler.java:3844)
```

The actual exception raised depends on the platform and Java VM and may include:

```
#
# An unexpected error has been detected by HotSpot Virtual Machine:
#
#  %SYSTEM-F-ACCVIO (0xc) at pc=84C1A4A0, pid=543163880, tid=63640192
```

and

```
SIGBUS    10*  bus error
.
.
.
%SYSTEM-F-OPCCUS, opcode reserved to customer fault at PC=FFFFFFFF80B6EF14,
PS=0000001B
.
.
.
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address
=000000000000
 0000, PC=FFFFFFFF80A708E8, PS=0000001B
```

and similar variants of the Access Violation messages.

This has now been fixed.

5.36.4 Unaligned Memory Faults on OpenVMS IA-64

Fixed in Release 7.2.5.4 Build 20090826.

Several unaligned memory faults have been located and fixed in the shared images associated with Oracle JDBC for Rdb on OpenVMS IA-64.

5.36.5 Read-Only Transactions Not Enforced on Connection Switch

Fixed in Release 7.2.5.4 Build 20090902.

When the connection switch:

```
Transaction=readonly
```

is used, all transactions started by the Oracle JDBC for Rdb drivers should be forced to READ-ONLY. However this is not correctly enforced if AUTOCOMMIT is subsequently turned OFF and it is possible that READ-WRITE transactions may be started on subsequent select statements.

This has now been fixed.

5.36.6 Sockets Not Correctly Closing on OpenVMS Clients

Fixed in Release 7.2.5.4 Build 20091002.

During the `Close()` of `Connections` using the Thin driver, sockets connecting the client to the server will be closed and resources released. However a problem with the release of data streams associated with these sockets prevents the sockets from closing down completely.

The failure for the sockets to completely close down may cause problems on clients running on OpenVMS. The mailboxes associated with TCP/IP sockets on OpenVMS will not be closed and will accumulate for processes that do multiple connects and disconnects using the JDBC Thin driver.

This problem only happens with clients running on OpenVMS and may depend on the Java version and type of VM used.

The associated data streams are correctly closed down when the `Connection` class is disposed, however unless explicitly disposed by the client application, the `Connection` class object will continue to exist after the connection is closed and until the Java garbage collection causes it to be disposed.

It is possible that the process may run out of TCP/IP ports or OpenVMS resources due to this accumulation of mailbox channels, but this depends on how often the Java garbage collection is run and whether or not the objects are collected.

This problem has now been fixed.

5.36.7 Cast Problem When Converting String to Date/Time

Fixed in Release 7.2.5.4 Build 20091007.

A problem in the JDBC driver code that converts `String` to the internal data format required for storing to an Rdb date/time column may cause the following exception to be raised.

```
java.lang.ClassCastException: java.util.Date
```

This problem occurs when a date/time value stored in its text form is used to set a parameter in a PreparedStatement or CallableStatement that is associated with an Rdb column or variable that has one of the following SQL datatypes:

- DATE ANSI
- DATE VMS
- TIME
- TIMESTAMP

For following code example will raise the exception described above.

```
.
.
.
 //  SQL :   create table tabx (f1 date vms);
 PreparedStatement  ps = conn.prepareStatement
   ("insert into TABX values (?)");

 String dtStr = "2003-11-07 12:34:56.780";
  ps.setString(1,dtStr);
.
.
.
```

A workaround for this problem is to use a Java date/time conversion method to convert from the String to a Java timestamp and then use the setTimestamp method instead of the setString method.

```
String dtStr = "2003-11-07 12:34:56.780";
Timestamp  dt = Timestamp.valueOf(dtStr);
ps.setTimestamp(1,dt);
```

This problem has now been fixed.

▲contents

5.37 New Features for Release 7.2.5.3

This section describes new and changed features in Oracle JDBC for Rdb release 7.2.5.3.

None in this release.

▲contents

5.38 Corrections in Release 7.2.5.3

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.3.

5.38.1 Blob DatabaseMetaData Access May Crash Thin Server

Fixed in Release 7.2.5.3 Build 20080327.

During the retrieval of RDB$DESCRIPTION data from Rdb System relations for the inclusion into the resultSets returned by various methods within the DatabaseMetaData class, the Oracle JDBC for Rdb Thin Server must create and execute List Cursors.

A problem in the synchronization of access during List Cursor operations during metadata retrieval meant that windows of opportunity exist where the thread currently accessing the List Cursor may interfere with other concurrent threads accessing the database from the same Thin Server.

This may result in a variety of exceptions and/or bugchecks being raised. In some cases the Thin Server may terminate unexpectedly with or without log or bugcheck messages.

This problem only occurs within Thin Servers and only during DatabaseMetaData method calls where description or comment data is being returned for the database object, and only if there is another concurrent connection executing SQL statements.

For example, drilling-down Rdb database connection metadata within JDeveloper using an Oracle JDBC for Rdb thin driver connection to a Thin Server may interfere with concurrent clients on that same server.

This has now been fixed.

5.38.2 BigDecimal Scaling Incorrect in PreparedStatement SetObject()

Fixed in Release 7.2.5.3 Build 20080623.

The scaling of BigDecimal objects may be incorrect when used as the source data objects for `PreparedStatement.SetObject()` methods.

During the conversion of the BigDecimal to the underlying database datatype, scaling information is lost which may result in the wrong values being applied.

A workaround for this problem is to use the `PreparedStatement.SetBigDecimal()` methods instead.

This problem has now been fixed.

5.38.3 Connection.nativeSQL() Method Throws Null Pointer Exception

Fixed in Release 7.2.5.3 Build 20080623.

Calling the `Connection.nativeSQL()` method will result in a `NullPointerException` exception being raised.

This has now been fixed.

5.38.4 Calling Resultset.isLast() Method May Change Transaction Behavior

Fixed in Release 7.2.5.3 Build 20080625.

This problem affects the Oracle JDBC for Rdb Native driver only. Applications using the Oracle JDBC for Rdb Thin driver will not see this problem.

When the native driver is used, calling the Resultset.isLast() method will cause the driver to fetch all the records of the resultSet to determine which is the last record.

During this processing the internal transaction status is set incorrectly. If a SQL statement requiring a read-write transaction is executed subsequent to this but prior to the resultSet's statement being closed, the update statement may fail with the following exception:

```
%RDB-E-READ_ONLY_TRANS, attempt to update during a read-only
 transaction
```

This has now been fixed.

▲contents

5.39 New Features for Release 7.2.5.2

This section describes new and changed features in Oracle JDBC for Rdb release 7.2.5.2.

5.39.1 DEC_KANJI and DEC_HANZI Support Enabled

Release 7.2.5.2.

Support was added to Release 7.1.3 Oracle JDBC for Rdb Drivers for accessing DEC_KANJI and DEC_HANZI data from Oracle Rdb databases but until now had not been adequately tested, so Oracle advised against using Oracle JDBC for Rdb drivers to access DEC_KANJI and DEC_HANZI data from Oracle Rdb databases.

Testing of Oracle JDBC for Rdb drivers using these character sets in conjunction with SHIFT_JIS on PC platforms has now been completed and the prior limitation of use of these characters sets has now been removed.

▲contents

5.40 Corrections in Release 7.2.5.2

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.2.

5.40.1 ResultSet.getBigDecimal() Not Working With System ResultSets

Fixed in Release 7.2.5.2 Build 20070714.

Calling the getBigDecimal() methods on a column in a resultset returned by DatabaseMetaData methods may fail with the following exception:

```
java.lang.ArrayIndexOutOfBoundsException: 2
```

This has now been fixed.

5.40.2 Setting TraceLevel Fails When Using Hexadecimal Notation

Fixed in Release 7.2.5.2 Build 20070731.

A problem in parsing hexadecimal values for trace level may cause an exception to be raised when trace level values greater than 0x7FFFFFFF are used:

```
java.lang.NullPointerException
```

This has now been fixed.

5.40.3 Delimited Identifier Problem in AS Clause of Select Statement

Fixed in Release 7.2.5.2 Build 20070731.

A problem introduced in Release 7.2.5.1 in parsing delimited identifiers used in the AS clause of a select statement may cause an exception to be raised:

```
select last_name as "name 1", first_name from employees
```

```
SQLException: in <rdbjdbcsrv:prepare_stmt>
%SQL-F-RELNOTDEF, Table FIRST_NAME is not defined in database
 or  schema:42000
```

A workaround is to not use delimited identifiers in the AS clause:

```
select last_name as name1, first_name from employees
```

This problem has now been fixed.

5.40.4 Configuration File Problem in "DEFAULT" Server Definition

Fixed in Release 7.2.5.2 Build 20080122.

A problem in how properties were copied from the "DEFAULT" server definition during the instantiation of server information may cause servers to have inappropriate configuration settings.

This problem may only arise if the "DEFAULT" server definition in the server configuration file contains any of the following properties:

```
 ssl.default
ssl.context
ssl.keyManagerFactory
ssl.keyStoreType
ssl.keyStore
ssl.keyStorePassword
ssl.trustStore
```

ssl.trustStorePassword
<allowPrivUser>
<allowUser>
<allowDatabase>
<allowPrivUser>

If any of the above properties are used in the "DEFAULT" server definition it is possible that server configuration properties such as Allowed Databases, or Allowed Users may be incorrectly propagated from the server they were specified for, to all other servers described in the same configuration file.

A workaround for this problem is to not use any of the above properties in the "DEFAULT" server definition and instead place the property in each of the server definitions that require it.

Note that the examples of configurations files used in the Oracle JDBC for Rdb documentation may be susceptible to this problem as they do show the use of the following property in the "DEFAULT" server definition:

```
ssl.default="true"
```

The problem has now been fixed.

5.40.5 Pool Server User Restriction Problem

Fixed in Release 7.2.5.2 Build 20080122.

If a server taking part in a pool of servers controlled by a Pool Server has restricted access enabled and one or more AllowedUser entries are present for that server, the Pool Server may choose the server as a possible candidate server for the connection request, even if the connection username is not one of the specified AllowedUsers.

If this happens, the connection request will be redirected to that chosen server but the connection attempt will be immediately terminated with the following exception:

```
SQLException: Access to server denied
```

Changes have now been made to ensure that information is passed correctly to the Pool Server during the initial connection request to allow it to correctly determine if a candidate pooled server will accept the user requesting the connection prior to redirecting the connection request to that server.

5.40.6 Potential Problem When Dumping SQLDA in Trace

Fixed in Release 7.2.5.2 Build 20080125.

A problem in allocation of an internal buffer for the dumping of a SQLVAR data area may cause unexpected and unusual problems during the execution of the server that may result in the server terminating unexpectedly.

This problem may only be seen if the traceLevel `DUMP SQLDA` flag bit is set for the connection or set server-wide and the data area of the SQLVAR being dump is greater than 512 octets in length.

Bit	Hexadecimal Value	Decimal Value	Traces
14	0x00004000	16384	Dump SQLDA information

A workaround for this problem is to not set the tracelevel `DUMP SQLDA` trace flag bit.

This has now been fixed.

5.40.7 Connection.getCatalog() Returns Wrong Value

Fixed in Release 7.2.5.2 Build 20080212.

The Connection.getCatalog() method incorrectly returns a single quote delimited string instead of a NULL object when connected to a single schema database. When used in conjunction with a multi-schema database the correct catalog value is returned.

This problem prevents JDeveloper 10.x from correctly displaying table and views within the Database branch in the Connections Navigator.

A similar problem may occur when using the Connection.getSchema() method.

5.40.8 Potential Memory Leak With Views

Fixed in Release 7.2.5.2 Build 20080227.

A problem in the handling of statement preparation for SQL statements that contain views where the view result tuples cannot have a dbkey associated with them, caused a memory leak in the Oracle JDBC for Rdb drivers and servers.

The problem may manifest itself in a number of ways including access violations and exceptions stating that shared memory has been used up.

The following example shows a typical view that may cause this problem:

```
create view view1 (c1 integer, c2 integer) as select c1,c2 from t1
 union select c3,c4 from t2;
```

Queries on this view will execute correctly, however during the internal preparation of each query, memory may be allocated that may stay allocated until the connection is closed. In addition the underlying SQL statement may not be released correctly, which in turn may prevent metadata updates from being carried out on the referred tables.

▲contents

5.41 New Features for Release 7.2.5.1

This section describes new and changed features in Oracle JDBC for Rdb release 7.2.5.1.

5.41.1 SQLDA Dumping

Release 7.2.5.1.

Setting the tracelevel to 0x00004000 (Decimal 16384) will provide information about the SQLDA information passed to and from SQL.

See the *Oracle JDBC for Rdb User Guide* for details.

5.41.2 failSAFE IP With Pool Servers

Release 7.2.5.1.

Pool servers may be configured to ensure that redirected connection requests will still correctly redirect during failSAFE IP fail over.

See the *Oracle JDBC for Rdb User Guide* for details.

5.41.3 HandshakeTries and Wait on Multiprocess Native Connections

Release 7.2.5.1.

The `multiprocess` option on native connections allows the use of executor sub-processes to carryout Rdb connections on behalf of your application using the RdbNative driver. You now have the capability of specifying handshake options during the initial communication handshake protocol used by the main and associated sub-processes.

See the *Oracle JDBC for Rdb User Guide* for details.

5.41.4 Server Access Security Enhancements

Release 7.2.5.1.

Servers may be configured to restrict the access of their served databases to a list of allowed usernames. The server configuration `allowUser` has been added to the server section of the configuration files restricting access to databases via that server to only those users specified.

In addition a server password can be specified using the `srv.password` configuration option which forces all users of that server to provide an additional password before access via the server will be granted.

See the *Oracle JDBC for Rdb User Guide* for details.

5.41.5 Restriction on Using Multiple Blob Fields in Join Now Removed

Release 7.2.5.1.

In previous releases of Oracle JDBC for Rdb the following limitation was specified:

> Blobs will only be returned correctly from SQL join statements for the first table mentioned in the join set. For example, given the SQL statement:
>
> ```
> Select ta.blob, tb.blob from table1 ta, table2 tb
>  where ta.name = tb.name
> ```
>
> ta.blob will be returned correctly as it is from the first table referenced in the join set. Trying to access tb.blob may result in the following SQL error:
>
> ```
> %SQL-F-BADPREPARE, Cannot use DESCRIBE or EXECUTE on a statement
>  that is not prepared
> ```

This restriction has now been lifted, the Oracle JDBC for Rdb drivers now handle blob fields from multiple tables within a single join statement.

However due to the nature of the parsing carried out by the Oracle JDBC for Rdb drivers it is required that all blob columns referenced from the second and subsequent tables in the join must be qualified using correlation names as shown in the above example of select.

Failure to use a correlation name in conjunction with the blob column name may result in SQL parsing errors when data is retrieved from the blob field as the drivers do not have enough information to determine the correct table to access the blob data from.

```
SQL-F-FLDNOTCRS, Column <blob col> was not found in the tables in
 current scope
```

This limitation also means that the use of "*" in the select clause for a join across two or more tables that include blob fields may also cause a similar SQL error.

▲contents

5.42 Corrections in Release 7.2.5.1

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.1.

5.42.1 Incorrect Row Number Returned After ResultSet.getLast() Call

Fixed in Release 7.2.5.1 Build 20060906.

A problem in the determination of the current row number when using Scrolling ResultSets caused the ResultSet.getRow() method to return an incorrect row number after absolute positioning of cursor after the end of stream.

The problem is only in the Oracle JDBC for Rdb Native Driver and does not show up in when using the Oracle JDBC for Rdb Thin driver.

The problem may be seen only after a call has been made to ResultSet.afterLast() method followed by a call to ResultSet.last().

The call to ResultSet.afterLast() incorrectly sets an internal record counter to one greater than the actual count .

The following is an example of this problem.

```
.
.
.
 Statement s2   = conn.createStatement(
            ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_UPDATABLE);

 ResultSet rs = s2.executeQuery("select * from employees");
 rs.afterLast();
 rs.last();
 System.out.println("row number  :" + rs.getRow());
 System.out.println("employee_id :" + rs.getString(1));
 rs.close();
 s2.close();
.
.
.
```

May return the following information when used with the Employees table in the PERSONNEL or MF_PERSONNEL databases provided as sample database in the Oracle Rdb installation (the row number of the last record should be 100) :

```
row number  :101
employee_id :00471
```

5.42.2 Pool Server Startup of Pooled Servers May Fail When Persona is Used

Fixed in Release 7.2.5.1 Build 20061011.

A problem in the naming of the subordinate processes used to create a server process during the automatic startup of servers by the Pool Server may cause the following exception:

```
%RUN-F-CREPRC, process creation failed
-SYSTEM-F-DUPLNAM, duplicate name
```

The following related exception might also be seen during the attempted startup of the pooled server process:

```
java.sql.SQLException: Unable to start process, status: 0x164 :
 substatus -4
```

These problems may be seen only if PERSONA is used to change the server authorization characteristics of the started servers.

5.42.3 Last Column in Select List May be Inaccessible in Some Queries

Fixed in Release 7.2.5.1 Build 20061124.

A problem in the handling of internal dbkey information may prevent the application access to the last column in a select list. This problem only occurs if the select query used cannot provide unique Dbkeys for the resultant tuples. Queries containing derived tables or views from multiple tables may show this problem.

For example:

```
select c1.last_name from (select * from employees c where
 c.employee_id='00170') c1
```

may fail to return the last_name correctly when the Resultset.GetString(String columnName) method is used.

```
select c1.last_name, c1.first_name from (select * from employees c
where c.employee_id='00170') c1
```

may correctly return the last name but not the first name as the problem only affects the last column in the outermost select list.

This problem was introduced in code changes made for release 7.1.3.0 of the Oracle JDBC for Rdb drivers.

5.42.4 Abnormal Client Termination May Prevent Executor Re-use

Fixed in Release 7.2.5.1 Build 20061221.

If a client application using an MP Server terminates abnormally or the client socket is lost, the associated database connection will be disconnected by the server. However due to an internal problem, the executor process associated with the terminated client may remain present on the system in LEF state.

The handling of abnormal termination did not correctly terminate the executor process, nor did it place the free executor back in the free list for re-use. This results

in orphaned executor processes that will remain on the system in LEF waiting state but will never be re-used.

If the client abnormal terminations occur frequently, the number of inactive executor processes will grow and may eventually cause system resource problems and excessive swapping.

5.42.5 Decimal Column Problem With Native Driver

Fixed in Release 7.2.5.1 Build 20061221.

A problem in the nativeRdb driver caused Decimal columns to be returned incorrectly. This problem only affects applications using the rdbNative driver.

The Decimal datatype may be used by SQL when scaled integers are returned after manipulation by an internal function or aggregate operation. For example, the following query may return incorrect values when executed through the rdbNative driver.

```
select distinct salary_amount from salary_history
```

Applications using the rdbThin driver should not encounter this problem.

5.42.6 'EFN xx is not available' Message on Executor Startup

Changed in Release 7.2.5.1 Build 20070302.

In earlier versions of Oracle JDBC for Rdb, if the MP Server found that the common event flag number it was using to start the handshake process with a newly created executor was not available after trying to obtain it for reasonable length of time, the server would abort the client connection attempt with the following exception:

```
'EFN xx is not available'
```

In such a case the event flag used by the server was probably left set by a previous attempt by the server to create an executor process but failed during the process startup due to resource problems.

The server code has now been changed to correctly clean up its event flags usage if an executor process fails to start up correctly. In addition the server now does not abort when it finds the event flags already in use, but now assumes that as the amount of time the event flag is unavailable is much longer than the amount of time it might

take the executor process to be created and executor image run-up, that the flag may be reset and the current executor start-up may proceed.

5.42.7 Extraneous Log Message During Auto-restart Check by Pool Server

Changed in Release 7.2.5.1 Build 20070306.

A problem in how the Pool Server checked the availability of its pooled servers when carrying out AutoRestart checking meant that an extraneous CLIENT LOST message would be logged by the pooled server every time it was checked. The following message would be logged:

```
srv.DBActionHandler <idle> Connection to Client lost
```

This has now been fixed.

5.42.8 Logfile Not Correctly Set for Servers Started Using the Controller

Fixed in Release 7.2.5.1 Build 20070412.

The logfile used by the server to record trace message and other output may be set in the server specification section of the configuration file used in conjunction with the servers and the Controller.

A problem in how the logfile information was passed to the newly started server process prevented the correct logfile specification to be used by the server when the server was started using the Controller Start Server command.

This has now been fixed.

▲contents

5.43 New Features for Release 7.2.5

This section describes new features found in Oracle JDBC for Rdb release 7.2.5.

5.43.1 Persona

Release 7.2.5.

When a Thin or Pool server starts up, it automatically inherits the rights identifiers, quotas, and authorization attributes of the process under which it was started. You may now override this default behavior by specifying a persona to use on the startup of the server. This persona will then be used by both the server and the underlying OpenVMS operating system to determine the rights and authorities of the server process and any executor processes that the server may start up.

This feature was introduced in Release 7.1.4, but was omitted from the release notes.

5.44 Corrections in Release 7.2.5

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.

5.44.1 Incorrect STARTUP72 Installed With V7.2-41 Kit

Fixed in Release 7.2.5 Build 20060505.

An incorrect version of the SQLSRV_JDBC_SERVER_STARTUP72.COM file was inadvertently placed in the installation kit for Oracle JDBC for Rdb 7.2.4.1.

This version of the file does not set up the RDB$JDBC_SQSNAM_* logical name properly and may cause problems when you try to use this file with SQL/Services Thin server startup.

The following line of DCL command in this file is incorrect:

```
$ nam    :='f$logical("RDB$JDBC_SQSNAM_''port'")
```

It should read:

```
$ nam    = f$logical("RDB$JDBC_SQSNAM_''port'")
```

This has now been fixed.

5.44.2 Multi-process Server Continuous DIO Activity When Idle

Fixed in Release 7.2.5 Build 20060505.

A problem with the way error and output channels are assigned during the creation of the executor subprocess by a detached Multi-process server may cause the server process to continually issue direct I/Os to the associated mailboxes. This can be seen

as a continuous rise in the "Direct I/O" count for that process even when the server is idle.

Although this does not interfere with the correct functionality of the server, it could incorrectly show up as activity on a quiet server.

A workaround is to start up the Multi-process server directly in a login session rather than detached.

This has now been fixed.

5.44.3 Client idleTimeout Problem in Reused Prestarted Executors

Fixed in Release 7.2.5 Build 20060505.

The amount of time that a client connection may be idle can be limited by using the cli.idleTimeout parameter for the Thin server.

However, the client idle timeout value set for the server will be ignored when a Multi-process server is used with prestarted executors.

If the client gets a prestarted executor on connection, the client idle timeout for the server does not get properly transferred to the client context and no timeout will be issued.

Additionally even if an executor was not prestarted, if it is reused then a similar problem will occur and the inactivity timer will not be set.

The client idle timeout set for a server is now correctly observed by prestarted and re-used executors.

5.44.4 Syntax Error in Query Generated for DatabaseMetaData.getTables

Fixed in Release 7.2.5 Build 20060620.

The JDBC DatabaseMetaData.getTables() method allows the caller to obtain information about the tables and views found in a connected database. When you call this method, you can supply a list of table types to search for.

Currently the Oracle JDBC for Rdb drivers recognize the following types of tables for this method:

- TABLE

- VIEW
- SYSTEM
- SYSTEM TABLE
- SYSTEM VIEW
- LOCAL TEMPORARY
- LOCAL TEMPORARY TABLE
- GLOBAL TEMPORARY
- GLOBAL TEMPORARY TABLE
- INFORMATION
- INFORMATION TABLE

The drivers should ignore any table type not in the above list.

However, due to a problem in the driver code, if the list of table types starts with a type that is not recognized by the driver, a SQL syntax exception will be generated. For example, the following example will result in a SQL syntax error:

```
String types[] = {DERIVED, "TABLE", "VIEW", "GLOBAL TEMPORARY"};
ResultSet rs = dbmd.getTables("","", "%" , types);
```

One possible workaround for this problem is to re-order the types so that the first type specified is one from the list of recognized table types, for example:

```
String types[] = ("TABLE",  DERIVED, "VIEW", "GLOBAL TEMPORARY"};
ResultSet rs = dbmd.getTables("","", "%" , types);
```

This example does not generate a SQL error.

This problem has now been fixed.

5.44.5 Show Clients in Controller May Crash Connected Thin Server

Fixed in Release 7.2.5 Build 20060620.

A change in handshake protocol in Oracle JDBC for Rdb Release 7.1.4.1 drivers introduced a problem in how thin servers respond to requests for client information.

Issuing a SHOW CLIENT command in the Oracle JDBC for Rdb Controller command line may cause the connected thin server to access violate and consequently terminate the server process.

This problem has now been fixed.

▲contents

5.45 New Features for Release 7.2.4.1

This section contains new features and technical changes for Oracle JDBC for Rdb release 7.2.4.1.

5.45.1 Client and Server Timeout Feature

Release 7.2.4.1.

You can now specify the amount of time a server or a client connection may remain inactive before the connection will be terminated or the server closed down.

See the *Oracle JDBC for Rdb User Guide* for details.

5.45.2 Executor Name Prefix

Release 7.2.4.1.

You can now specify the name prefix for executors started up by the Multi-process server. This can help in identifying executor processes on your system.

See the *Oracle JDBC for Rdb User Guide* for details.

▲contents

5.46 Corrections in Release 7.2.4.1

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.4.1.

5.46.1 Incorrect Installation Directory for RDBJDBCCFG.XML

Fixed in Release 7.2.4.1 Build 20060130.

The release notes for Oracle JDBC for Rdb Release 7.1.2 incorrectly specified that the RDBJDBCCFG.XML file would be copied to SYS$COMMON:[RDB$JDBC] directory. The RDBJDBCCFG.XML is actually copied to two directories during product installation:

- The product installation directory found under the main JDBC directory, for example,

```
SYS$COMMON:[RDB$JDBC.0701-4V0614]
```

- The SYS$COMMON:[RDB$JDBC.COM] directory

In addition, the installation procedure incorrectly replaced the RDBJDBCCFG.XML file in the SYS$COMMON:[RDB$JDBC.COM] directory, overwriting any already existing file of the same name.

The release notes have been fixed, and the installation procedure will only copy the RDBJDBCCFG.XML file to the SYS$COMMON:[RDB$JDBC.COM] directory if the file does not already exist in that directory.

5.46.2 Persona Not Handled Correctly by the Multi-process and Pool Servers

Fixed in Release 7.2.4.1 Build 20060130.

When the Persona feature is used in conjunction with a Multi-process or Pool server, a problem in the way either the executor processes or the pooled server processes are created prevented the correct Persona identification from being passed to the created processes. This problem may result in the following error being raised:

```
java.io.IOException: Child creation error: not owner
```

Due to a restriction in the use of the Java System.exec() method that was used by the JDBC servers to start executor sub-processes and pooled servers, the security information and Persona details were not copied across to the newly created process.

The JDBC servers now use the OpenVMS system service CREPRC to start processes. CREPRC correctly transfers the security information to the new process.

5.46.3 Multi-process Server Timeout May be Too Short

Fixed in Release 7.2.4.1 Build 20060130.

When a Multi-process server talks to an executor it uses a handshake protocol to check that the executor is still alive and accepting direction. By default, if the executor has not responded to the server's synchronization request within five seconds it will raise the following exception and terminate the connection :

```
Lost connection to executor
```

This synchronization handshake is done after the executor has replied to the server that it has completed the task requested and is waiting for the next operation to carry out. This synchronization failure will not be raised while the executor is busy within the database and thus is unaffected by such things as database locks or the duration required to compile or execute queries. It will only occur when the executor is known to be waiting for the next action to carry out.

In heavily loaded systems, especially on single-cpu systems, it is possible that the executor process may not be scheduled for execution within the window of this synchronization handshake and the exception may be raised.

In order to carry out this synchronization, in previous versions of the drivers the server polls the executor up to 500 times with a 10 millisecond delay between each poll request. If no response is found after 500 tries, the server raises the above exception.

This version of the Oracle JDBC for Rdb drivers now allows you to specify, at the server level, the maximum number of poll tries and the delay between each try. If you know that the system on which the server is executing could possibly have extended process scheduling delays, you can ensure that the server will not time out on the synchronization handshake. Two new switches have been added to the server definition and startup.

- Srv.MPmaxTries---Use to specify the maximum number of poll tries
- Srv.MPtryWait---Use to specify the delay between each try

See the *Oracle JDBC for Rdb User Guide* for more information.

5.46.4 SSL Problems With srv.idleTimeout and srv.bindTimout

Fixed in Release 7.2.4.1 Build 20060208.

The *Oracle JDBC for Rdb User Guide* for Release 7.1.3 incorrectly referred to the *srv.idleTimeout* as affecting the inactivity timeout for a connection. This switch actually refers to the timeout period for server inactivity. In addition, this feature was not fully functional in previous versions.

The *srv.bindTimeout* configuration variable was meant to limit the time the server will wait for an acknowledgement from the client that the database attach should proceed. The default value is 0, which means that the server will wait indefinitely.

This timeout is useful when dealing with SSL communication, as the server uses it to limit the time it will wait for the client to send down an attach request after a new socket connection has been requested. If the client fails to use an SSL secure socket when trying to communicate with a server that has SSL enabled, the client thread within the server will hang as the connection cannot complete. The *srv.bindTimeout* value specifies how long this wait should be before giving up.

Unfortunately the default for the value was incorrectly set to 1 second, and the *srv.bindTimeout* server attribute was ignored in the XML configuration file. This meant that on CPU-bound systems it was possible that the initial SSL negotiation could take longer than one second and thus cause a TIMOUT failure on the new connection request.

These problems have now been fixed. See the *Oracle JDBC for Rdb User Guide* for more information.

5.46.5 IA-64 Array Out of Bounds Exception with String Indexing

Fixed in Release 7.2.4.1 Build 20060208.

A problem in the way Java on OpenVMS IA-64 carries out string index operations in association with static final string constants may infrequently cause the following type of exception to be raised:

```
Caused by: java.lang.ArrayIndexOutOfBoundsException: 1054649176 at
java.lang.String.indexOf(String.java:1266) at
java.lang.String.indexOf(String.java:1236) at
java.lang.String.indexOf(String.java:1218) at
oracle.rdb.jdbc.common.Statement.getTableName(Statement.java:3148)
```

In all cases, the index value shown after the exception name is very large, in the same order of magnitude as seen above.

The *getTableName* method has now been changed to a mechanism other than *indexOf* to carry out its operation. This problem should no longer be seen.

5.46.6 Comments Within SQL Text Not Handled Correctly

Fixed in Release 7.2.4.1 Build 20060301.

Executing or preparing a statement that has SQL text containing leading or embedded comments may cause errors during parsing of the statement.

Some third-party products may use comments such as */* *comment* */* in the text they send down to the JDBC drivers for compilation. Although handled correctly by Oracle Rdb, comments of this style caused a problem in the determination of statement types during the preliminary parsing of the statement by the JDBC driver.

For example the following SQL text:

```
stmt.Execute(/* This is a comment */ select * from jobs);
```

would cause an SQLException:

```
SQLException: in <rdbjdbcsrv:execute_immediate> %SQL-F-EXESELSTA,
Attempted to EXECUTE a SELECT statement:RR000
```

The JDBC driver could not correctly determine the type of statement and used the wrong underlying SQL operation to attempt to execute it.

The drivers now extract out comments prior to determining the statement type and sending the native SQL down to Oracle Rdb. The drivers will now correctly parse out C and SQL type comments, for example:

```
/* comment */
! this comment will be terminated at the next line break
-- this comment will be terminated at the next line break
// this comment will be terminated at the next line break
```

5.46.7 Prepared Statements Memory Leak With Multi-process Servers

Fixed in Release 7.2.4.1 Build 20060301.

During the preparation of PreparedStatements, the Multi-process server has to allocate memory from the servers' global shared memory pool that will hold some information about columns and parameter markers in the statement that is being prepared.

Due to a coding problem, some of this memory was incorrectly allocated each time the prepared statement was executed, instead of only once at statement compilation time. This wrongly allocated memory was never freed after use. Executing the same prepared statement multiple times will slowly diminish the shared memory available

to the server, eventually causing a problem when the shared memory allocation is all used up.

This has now been fixed.

▲contents

5.47 Corrections in Release 7.2.4

This section describes software errors corrected in Oracle JDBC for Rdb 7.2.4.

5.47.1 Maximum Size of Single Data Row Increased to 65,272 Octets

Fixed in Release 7.2.4 Build 20051114.

During copying rows of data from Oracle Rdb, the Oracle JDBC for Rdb drivers incorrectly limited the number of octets copied to 36863 octets. This can cause problems when there are more than 36863 octets in the row.

The following exception is a symptom of this data row truncation:

```
Statement creation failed: java.sql.SQLException: Connection lost :
java.lang.NegativeArraySizeException @rdb.Client.fillCache
```

The maximum size of a data row supported by the drivers has now been increased to 65,272 octets in keeping with the maximum row size supported by Oracle Rdb.

5.47.2 Another Connection Overlap Window Found With Pool Servers

Fixed in Release 7.2.4 Build 20051209.

Another potential overlap of connections between the connection made by the Oracle JDBC for Rdb Pool server and its pooled servers has been found which may cause the incorrect rejection of a client connection even when a free connection slot is available.

This is similar to the problem referred to in the Oracle JDBC for Rdb Release 7.1.3.3 notes as *Spurious Maximum Number of Clients Exceeded Exception*.

The handshake protocol during server check by the Pool server has now been changed to prevent this overlap of connections.

5.47.3 SSL Server Problem with XML-Formatted Configuration File

Fixed in Release 7.2.4 Build 20051220.

A problem in the parsing of XML configuration file data prevented the correct port and node information from being assigned to named servers of the type "RdbThinSrvSSL", "RdbThinSrvMPSSL" and "RdbThinSrvPoolSSL.

Any URL specification provided for the individual server would be ignored, and the default port and node used instead.

This has now been fixed.

▲contents