

Oracle® JDBC for Rdb

Release Notes
Release 7.3.3.1.
August 2014

Oracle JDBC for Rdb Release Notes, Release 7.3.3.1.

Copyright © 2005, 2014 Oracle and/or its affiliates. All rights reserved.

Primary Author: Jim Murray.

Contributing Author:

Contributor: Wolfgang Kobarg-Sachsse.

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, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

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 and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon 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, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface.....	9
Purpose of This Manual.....	9
Intended Audience	9
Access to Oracle Support.....	9
Document Structure	9
Conventions	9
Chapter 1 Installation and Documentation.....	11
1.1 Accessing the Documentation.....	11
1.2 System and Software Requirements	12
1.3 Installation.....	13
1.3.1 Remove Prior BETA Versions of Oracle JDBC for Rdb Release 7.3.....	13
1.3.2 Contents of the Oracle JDBC for Rdb Kit	14
1.3.3 Installation Procedure	16
1.3.4 Rdb Thin driver software on Windows.....	20
1.3.5 Rdb Thin driver software on Other operating systems	21
Chapter 2 Enhancements Provided in Oracle JDBC for Rdb Release 7.3.3.1.	22
2.1 Change in Template Configuration Filename.....	22
Chapter 3 Problems Corrected.....	23
3.1 DatabaseMetadata.getColumns Problem with DefaultValue	23
3.2 Show Server Configuration file Problem.....	23
3.3 Unknown Action Message in Log file	24
3.4 Severe Performance Degradation in MP Server	24
3.5 Controller DCL Command line Failure	24
3.6 Server Access Restrictions Ignored	25
3.7 Datetime Insertion Issues.....	26
3.8 MP Server may Become CPU Bound.....	27
3.9 Controller Null Pointer Exception when no Default Server	28
3.10 Pool Server Tries to Restart Running Server.....	28
3.11 Controller Poll Reopenlogs Hangs.....	29
3.12 Pool Server Usage Balancing Problem.....	30
3.13 Configuration File Problem Regression.....	30
3.14 Server Configuration Deny User not Enforced.....	30
Chapter 4 Known Problems and Workarounds	32
4.1 Thin Server Deadlocks.....	32
4.2 Using Java Fast VM on OpenVMS ALPHA	32
4.3 Using the Oracle SQL/Services Management GUI and JDBC Dispatchers.....	33
4.4 Blob Columns and Correlation Names	33
4.5 Blob Columns and Update Statements	34
4.6 External Procedures and Thin Server	35
4.7 Limitations	35
4.7.1 Unsupported Methods.....	35
4.7.2 Auto-generated keys	37
4.7.3 String Truncation Warnings.....	37
4.7.4 Numeric and String Functions in JDBC	37

Chapter 5	New Features and Corrections in Previous Releases	38
5.1	New Features for Release 7.3.3.0.	38
5.1.1	Oracle Rdb PID now Displayed.....	38
5.1.2	Controller Command Show Executors	38
5.1.3	Extra Timestamp Precision	38
5.1.4	Nlslang Connection Switch.....	39
5.1.5	Use Query Header as Description.....	40
5.2	Corrections in Release 7.3.3.0.	41
5.2.1	Controller START SERVER Problems	42
5.2.2	Executor Initialization Problem	42
5.2.3	Idle Client Termination breaks Subsequent Connection	43
5.2.4	Controller STOPSERVER command Problem.....	43
5.2.5	Hang on Connection to Server when using SSL.....	44
5.2.6	DatabaseMetadata Pattern Matches and Nulls.....	44
5.2.7	DatabaseMetadata Missing Index Information.....	45
5.2.8	Forced Client Termination may Crash Thin Server.....	45
5.2.9	Show Clients may cause NullPointerException in Server.....	46
5.2.10	AccessViolation in MP Server.....	46
5.2.11	Syntax Errors in Insert and Update Statements	47
5.2.12	Statement.getGeneratedKeys() throws RdbException.....	48
5.2.13	SQLException thrown in ResultSet.isBeforeFirst()	48
5.2.14	PreparedStatement executeBatch() Problems	49
5.3	New Features for Release 7.3.2.0.	50
5.3.1	Server Options List Inheritance	50
5.3.2	Oracle JDBC for Rdb Manager Server	51
5.3.3	Restricting Server Access by IP.....	51
5.3.4	Executor Balancing.....	51
5.3.5	MinFreeExecutors.....	52
5.3.6	Executor Reuse	52
5.3.7	LogFile Patterns	52
5.3.8	New Server Configuration Option retainRdbSQLState.....	52
5.3.9	New Connection Option <i>app</i>	53
5.4	Corrections in Release 7.3.2.0.	53
5.4.1	Event Flag Problem with long Executor name Prefixes	54
5.4.2	Executor Process Termination Problem.....	55
5.4.3	Some Server Characteristics not Correctly Inherited from DEFAULT.....	55
5.4.4	Memory Problem with Pool Servers and Java 1.6.0-2.....	57
5.4.5	Exception not Thrown when Record Locked during Update	57
5.4.6	Classpath Documentation Error	58
5.5	New Features for Release 7.3.1.0.	59
5.5.1	“Owner” may be used in XML Configuration Files	59
5.5.2	SQL Statement Restriction and Denial	59
5.5.3	Event Notification.....	59
5.6	Corrections in Release 7.3.1.0.	60
5.6.1	Multi-process Server Connection Hang when Executor Dies	60
5.6.2	Multi-process Executor Process Terminates Unexpectedly	60

5.6.3 Pooled Server AutoRestart Problem	61
5.6.4 getGeneratedKeys() and OutOfBounds Exception	62
5.6.5 Multi-process Server fails when Persona Used	62
5.6.6 RestrictAccess being Applied even when Disabled.....	63
5.6.7 Multi-process Problem in the Native Driver.....	64
5.7 New Features for Release 7.3.0.2.	64
5.7.1 Server Network Keep Alive.....	65
5.8 Corrections in Release 7.3.0.2.	65
5.8.1 Null pointer Exception during Server viability check by Pool Server.....	65
5.8.2 MINUS and INTERSECT Problem.....	65
5.8.3 Memory Leak when Executor Process fails to Run.....	66
5.8.4 Lost Executor not caught by Multi-process Server.....	66
5.8.5 Multi-process Native Driver Problem.....	66
5.8.6 PreparedStatement.getGeneratedKeys() Problem.....	67
5.8.7 Excessive IOs during Metadata Retrieval.....	67
5.8.8 Syntax error SQL-F-CONVARUND.....	67
5.8.9 Multi-process Server Hang	68
5.8.10 SSL Socket Intrusion Problem.....	68
5.9 New Features for Release 7.3.0.1.	69
5.9.1 Reopen Server Log files using Poll Subcommand	69
5.10 Corrections in Release 7.3.0.1.	69
5.10.1 Show Clients not Showing Column Names.....	69
5.10.2 Batched Statement Fails with MULTIPLE_RECORDS Exception	70
5.10.3 Incompatibility between 7.3.0.0 Thin Driver and Prior Release Servers	71
5.10.4 Global Memory leak when Executors are Run-down.....	71
5.10.5 Autorestart on Pooled Servers not Working	72
5.10.6 setFetchSize() hint Ignored by PreparedStatement.....	72
5.10.7 Missing Stmt Exception on Subsequent Execution of PreparedStatement....	73
5.10.8 EOFException on READ_ROW with Nested Statements.....	74
5.10.9 DatabaseMetaData.getUDTs().....	75
5.10.10 Using Multi-process with the JDBC Native Driver.....	75
5.10.11 Prepared Statement not Closing underlying Cursor.....	75
5.10.12 Release Statement Synchronization Problem	76
5.10.13 Uninitialized SQLCA Block in MP server	77
5.10.14 Controller SHOW CLIENTS and MP Server Problem	77
5.10.15 Documentation Error – Record Streaming	78
5.10.16 ResultSet.updateRow() and ResultSet.deleteRow() Problem.....	78
5.10.17 Problem using Column Renaming with dbkey	79
5.10.18 Class cast error on Scaled integer Retrieval after ResultSet.insertRow()....	79
5.10.19 ResultSet.deleteRow() Behaviour Change.....	80
5.10.20 Underlying Blob Handles not Released when using ResultSet.getBlob()... 82	
5.10.21 Sequence Values not Visible to ResultSet get Methods using Column Name	82
5.10.22 Scaled Integer Problem.....	84
5.10.23 Server Matching Exception may Stop Poll Handling	84
5.11 New Features for Release 7.3.0.0.	85

5.11.1 Shutdown Thread	85
5.11.2 Driver.attach()	86
5.11.3 Returning List of Known Databases	86
5.11.4 Controller Enhancements	86
5.11.5 RDB_EXT.JAR file	86
5.11.6 Performance Enhancements	88
5.12 Corrections in Release 7.3.0.0.	89
5.12.1 Server Startup Failure when using CFG File	89
5.12.2 AccessViolation on Disconnect when Inserting Blobs	89
5.12.3 PreparedStatement and Parameter Markers Known Problem now Resolved	90
5.12.4 Controller SHOW CLIENTS and MP Server Problem	91
5.12.5 Possible Memory Leak when Updating Blob Columns	91
5.12.6 Access Violation with Trace and Network Dump	92
5.12.7 Named Input Parameters not Working with CallableStatements	92
5.13 New Features for Release 7.2.5.5.	93
5.14 Corrections in Release 7.2.5.5.	94
5.14.1 Long-running Query Holds up New Connections in MP Server	94
5.15 New Features for Release 7.2.5.4.	94
5.16 Corrections in Release 7.2.5.4.	95
5.16.1 Access Violation at Java_rdb_JNI_SetStrVal	95
5.16.2 DCL Command Line Too Long	95
5.16.3 Access Violation during DriverManager.getConnection() when Database Specification is Missing	96
5.16.4 Unaligned Memory Faults on IA64	97
5.16.5 Read-Only Transactions not Enforced on Connection Switch	97
5.16.6 Sockets not Correctly Closing on OpenVMS Clients causing Accumulation of Mailboxes	98
5.16.7 Cast problem when Converting String to Date/Time	98
5.17 New Features for Release 7.2.5.3.	99
5.18 Corrections in Release 7.2.5.3.	100
5.18.1 Interaction of DatabaseMetaData methods with Blobs may Crash the Thin Server	100
5.18.2 BigDecimal scaling Incorrect when used with PreparedStatement setObject() Methods	100
5.18.3 Connection.nativeSQL() method Throws Null Pointer Exception	101
5.18.4 Calling ResultSet.isLast() method May Change Transaction Behavior	101
5.19 New Features for Release 7.2.5.2.	101
5.19.1 DEC_KANJI and DEC_HANZI Support Enabled	102
5.20 Corrections in Release 7.2.5.2.	102
5.20.1 ResultSet.getBigDecimal() not Working with System ResultSets	102
5.20.2 Setting TraceLevel fails when using Hexadecimal Notation	102
5.20.3 Delimited Identifier Problem in AS clause of Select Statement	103
5.20.4 Configuration file problem in "DEFAULT" Server Definition	103
5.20.5 Pool Server May choose Incompatible Pooled Server when User Restriction Enabled	104
5.20.6 Potential Problem when Dumping SQLDA in Trace	104

5.20.7 Connection.getCatalog() Returns Wrong Value for Single Schema Databases	105
5.20.8 Potential Memory Leak with Views	105
5.21 New Features for Release 7.2.5.1.	106
5.21.1 SQLDA Dumping	106
5.21.2 failSAFE IP with Pool Servers	106
5.21.3 HandshakeTries and HandshakeWait on Multi-process Native Connections	106
5.21.4 Server Access Security Enhancements	107
5.21.5 Restriction on using Multiple Blob fields in Join now Removed	107
5.22 Corrections in Release 7.2.5.1.	108
5.22.1 Incorrect Row Number Returned after ResultSet.getLast() call	108
5.22.2 Pool Server Startup of Pooled Servers may fail When Persona is Used	109
5.22.3 Last Column in Select List may be Inaccessible in Some Queries	109
5.22.4 Abnormal Client Termination may Prevent Executor Re-use	110
5.22.5 Decimal Column Problem with Native Driver	110
5.22.6 'EFN xx is not available' Message on Executor Startup	111
5.22.7 Extraneous log message during Auto-restart check by Pool Server	111
5.22.8 Logfile not Correctly set for Servers Started Using the Controller	112
5.23 New Features for Release 7.2.5.0.	112
5.23.1 Persona	112
5.24 Corrections in Release 7.2.5.0.	112
5.24.1 Incorrect SQLSRV_JDBC_SERVER_STARTUP72 Installed with V7.2-41 Oracle JDBC for Rdb Kit	112
5.24.2 Multi-process Server May Show Continuous DIO Activity Even When Idle	113
5.24.3 Client idleTimeout Does Not Work for Prestarted and Reused Executors.	113
5.24.4 Syntax Error in Query Generated for DatabaseMetaData.getTables	114
5.24.5 Show Clients in Controller may Crash Connected Thin Server	115
5.25 New Features for Release 7.2.4.1.	115
5.25.1 Client and Server Timeout Feature	115
5.25.2 Executor Name Prefix	115
5.26 Corrections in Release 7.2.4.1.	116
5.26.1 Release Notes Specify Incorrect Installation Directory for RDBJDBCCFG.XML	116
5.26.2 Persona Not Handled Correctly by the Multi-process and Pool Servers	116
5.26.3 Multi-process Server / Executor Handshake Timeout May Be Too Short on Heavily Loaded Systems	117
5.26.4 Problems with srv.idleTimeout and srv.bindTimeout Configuration Variables and Their Use with SSL servers	118
5.26.5 IA64 Problem Causes Array Out of Bounds Exception When Handling String Indexing	118
5.26.6 Comments within SQL Text Not Handled Correctly	119
5.26.7 Prepared Statements May Cause a Memory Leak with Multi-process Servers	120
5.27 Corrections in Release 7.2.4.0.	120

5.27.1 Maximum Size of Single Data Row Increased to 65,272 Octets.....	120
5.27.2 Another Connection Overlap Window Found with Pool Servers.....	121
5.27.3 SSL Server Information Not Correctly Set from XML-Formatted Configuration File.....	121

Preface

Purpose of This Manual

The Oracle JDBC for Rdb 7.3.3.1 release notes summarize new features, corrections to software, restrictions, workarounds, and problems. They also include new features and corrections provided in releases 7.2.4.0 through 7.3.3.0. These release notes cover Oracle JDBC for Rdb for OpenVMS on both Alpha and Integrity Servers.

Intended Audience

This document is intended for users responsible for:

- System management
- Database administration
- Application programming

Access to Oracle Support

Oracle customers 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.

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 7.2.4.0 through 7.3.3.0.

Conventions

Oracle JDBC for Rdb is often referred to as JDBC.

Hewlett-Packard Company is often referred to as HP.

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 7.3.3.1.

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 RDBJDBC73
```

- 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 RDBJDBC73
```

- 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 RDBJDBC73/SELECT=filename.ext
```

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

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

1.2 System and Software Requirements

Oracle JDBC for Rdb requires the following software products to be installed:

Software	Minimum Version	
	Alpha	Integrity
HP OpenVMS	V8.2	V8.2-1
HP Java [™] SDK/RTE	V5.0-1 (1.5.0-1)	V5.0-1 (1.5.0-1)
Oracle Rdb	V7.2.1	V7.2.1

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	V1.5.0-1

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	
	Alpha	Integrity
Oracle SQL/Services	V7.1.6	V7.2

Detailed information about installing Hewlett-Packard Java for OpenVMS system may be found at the following web site:

<http://www.hp.com/java>.

Documentation for HP's Java for OpenVMS system may be found at the following web sites:

<http://h18012.www1.hp.com/java/documentation/index.html>

In line with HP 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.

UAF Fillm	4096
Channelcnt	4096
Wsdef	4096
Wsquo	8192
Wsexent and 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.

1.3 Installation

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

1.3.1 Remove Prior BETA Versions of Oracle JDBC for Rdb Release 7.3.

If you were participating in a BETA trial of Oracle JDBC for Rdb 7.3.x.x.x then you must remove any previously installed BETA kits before proceeding with the installation of this release.

Failure to remove previous 7.3 kits may prevent Oracle JDBC for Rdb from functioning correctly. In particular, several changes have been made to the client/server communication protocol during the life of the 7.3 BETA releases that may cause connection failures if 7.3 BETA thin driver jars are used to connect to a 7.3 server or vice-versa.

In addition, ensure that all deployed copies of the 7.3.x.x.x BETA rdbThin.jar are replaced. If not, connection failures may result in the applications using these BETA driver jars.

1.3.2 Contents of the Oracle JDBC for Rdb Kit

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

The Oracle JDBC for Rdb kit product installation file is named ORCL-pppVMS-RDBJDBC73-V0703-xxxxxx-1.PCSI where ppp will be the platform and xxxxxx will be the build instance of this kit, for example:

```
ORCL-AXPVMS-RDBJDBC73-V0703-0V0A18-1.PCSI
or
ORCL-I64VMS-RDBJDBC73-V0703-0V0A18-1.PCSI
```

The installation file is located in the RDBJDBC directory of the Rdb Software distribution CD. If you obtained the Oracle JDBC for Rdb kit from the Web, the installation file is contained in the RDBJDBC73xxxxx.ZIP file, where xxxxx refers to the build instance of the kit.

The installation kit is comprised of the following files:

BUILD_CERTS.COM	Command procedure example of building certificates for SSL.
OracleJDBCForRdbClient73000.msi OracleJDBCForRdbClient73000X64.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.
RDBJDBCEXEC73.EXE	Executor image in conjunction with a Multi-process server.
RDBJDBCCFG.XML	Example XML formatted configuration file.
RDBJDBCshr73.EXE	Shared image required for Oracle Rdb database access.

RDBJDBCMPSHR73.EXE	Shared image required for Multi-process Oracle Rdb database access.
RDBJDBC_EXECCLI.COM	CLI helper command procedure.
RDBJDBC_INSTALL.COM	Installation command procedure used by Polycenter during installation.
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 Oracle JDBC for Rdb servers are 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.
RDBTHIN.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb thin driver.
RDBTHINSRV.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb servers.
RDBTHINCONTROL.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb controller.
RDBTHINSRVPOOL.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb pool server.
RDBMANSRV.JAR	Java jar file containing the classes for the Oracle JDBC for Rdb manager server.
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.

1.3.3 Installation Procedure

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

1. If you obtained the kit in ZIP format, restore the kit file to a temporary directory:

```
$ unzip RDBJDBC73xxxx.ZIP -d MY_DIR
```

This will unzip the Polycenter kit for Oracle JDBC for Rdb and you will have access to the PCSI file, ORCL-pppVMS-RDBJDBC73-V0703-xxxxxx-1.PCSI, where ppp is the platform and xxxxxx is the build instance of this kit.

2. Use the Polycenter 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 0V0A18, and that the directory where the PCSI file can be found is MY_DIR.

```
$ PRODUCT INSTALL RDBJDBC73/SOURCE=MY_DIR
```

```
The following product has been selected:
```

```
    ORCL AXPVMS RDBJDBC73 V7.3-0V0A18          Layered Product
[Installed]
```

```
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.

ORCL AXPVMS RDBJDBC73 V7.3-0V0A18: Oracle JDBC for Rdb

Copyright © 1995, 2010, Oracle Corporation. All Rights Reserved.

This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:

ORCL AXPVMS RDBJDBC73 V7.3-0V0A18
DISK\$AXPVMSSYS:[VMS\$COMMON.]

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

Oracle JDBC for Rdb has been successfully installed in :

DISK\$AXPVMSSYS:[SYS1.SYSCOMMON.rdb\$jdbc.0703-0V0A18]

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

RDBJDBC_STARTUP.COM:

#! Oracle JDBC for Rdb startup command procedure

#!

\$ DEFINE/SYSTEM RDB\$JDBC_HOME -

DISK\$AXPVMSSYS:[SYS1.SYSCOMMON.rdb\$jdbc.0703-0V0A18]

\$ DEFINE/SYSTEM RDB\$JDBC_LOGS -

DISK\$AXPVMSSYS:[SYS1.SYSCOMMON.rdb\$jdbc.logs]

\$ DEFINE/SYSTEM RDB\$JDBC_COM -

DISK\$AXPVMSSYS:[SYS1.SYSCOMMON.rdb\$jdbc.com]

\$ DEFINE/SYSTEM RDBJDBC_SHR RDB\$JDBC_HOME:RDBJDBC_SHR73.EXE

\$ DEFINE/SYSTEM RDBJDBCMP_SHR RDB\$JDBC_HOME:RDBJDBCMP_SHR73.EXE

\$ DEFINE/SYSTEM RDBJDBCCEXEC RDB\$JDBC_HOME:RDBJDBCCEXEC73.EXE

...100%

The following product has been installed:

ORCL AXPVMS RDBJDBC73 V7.3-0V0A18

Layered Product

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.0703-0V0A18]

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 to use `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]

0703-0V0A18.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.

3. 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.0703-0V0A18]
```

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]
```

RDBJDBC_SHR to point to the shared image RDBJDBC_SHR73.EXE.

```
$ define/system RDBJDBC_SHR -  
SYS$COMMON:[RDB$JDBC.0703-0V0A18]RDBJDBC_SHR73.EXE
```

RDBJDBC_MPSHR to point to the shared image RDBJDBC_MPSHR73.EXE.

```
$ define/system RDBJDBC_MPSHR -  
SYS$COMMON:[RDB$JDBC.0703-0V0A18]RDBJDBC_MPSHR73.EXE
```

RDBJDBC_EXEC to point to the shared image RDBJDBC_EXEC73.EXE.

```
$ define/system RDBJDBC_EXEC -  
SYS$COMMON:[RDB$JDBC.0703-0V0A18]RDBJDBC_EXEC73.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.

4. Include the rdbnative.jar 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:

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

5. Test your installation using the "RdbJdbcCheckup" Java class in the RDBJDBC_CHECKUP.CLASS file. During the installation RDBJDBC_CHECKUP.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.

In order 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, **OracleJDBCForRdbClient73000.msi** and **OracleJDBCForRdbClient73000X64.msi** that may be used on your client Windows systems to install the RdbThin and the RdbThinControl jars.

The **OracleJDBCForRdbClient73000X64.msi** may be used on Windows 64 Bit systems. **OracleJDBCForRdbClient73000.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 7.3.3.1.

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

2.1 Change in Template Configuration Filename

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](#)

Chapter 3

Problems Corrected

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

3.1 DatabaseMetadata.getColumns Problem with Default Value

(BUG 18900456)
Fixed in Instance 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 7.3.3.0 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.

3.2 Show Server Configuration file Problem

Fixed in Instance 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$COMMON: [rdb$jjdbc.0703-3v0e5s] rdb$jjdbc_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.

3.3 Unknown Action Message in Log file

Fixed in Instance Build 20140722.

Starting with Oracle JDBC from Rdb Release 7.3.3.0 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 7.3.3.0 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.

3.4 Severe Performance Degradation in MP Server (BUG 19150908)

Fixed in Instance Build 20140722.

A change in the handling of mutex operations within the Multiprocess server made during the 7.3.3.0 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.

3.5 Controller DCL Command line Failure (BUG 18973965)

Fixed in Instance Build 20140722.

A problem introduced in release 7.3.3.0. 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 similar to 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
```

3.6 Server Access Restrictions Ignored (BUG 19149514)

Fixed in Instance 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>
```

3.7 Datetime Insertion Issues

(BUG 18921457)

Fixed in Instance Build 20140723.

Release 7.3.3.0 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 7.3.3.0 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 some 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 columns such as Timestamp or VMS Date or Time and the source is a date/time 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)
```

3.8 MP Server may Become CPU Bound

Fixed in Instance Build 20140731.

A problem introduced in 7.3.3.0 may cause the MultiProcess 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 execution that will be shown in the server log file:

```
java.util.ConcurrentModificationException
```

This has now been fixed.

3.9 Controller Null Pointer Exception when no Default Server

Fixed in Instance 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.

3.10 Pool Server Tries to Restart Running Server

Fixed in Instance Build 20140812.

The pool server will attempt to restart any server in its pool that is marked for restart and has been found to 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 JDBC 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 JDBC 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.

3.11 Controller Poll Reopenlogs Hangs

Fixed in Instance 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>
```

3.12 Pool Server Usage Balancing Problem

Fixed in Instance 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.

3.13 Configuration File Problem Regression

Fixed in Instance Build 20140820.

A problem with the inheritance of server attributes from the “DEFAULT” server was fixed in release 7.3.2.0. This problem has unfortunately resurfaced.

The problem and workaround is described in release note:

[*5.4.3 Some Server Characteristics not Correctly Inherited from DEFAULT*](#)

This has now been fixed.

3.14 Server Configuration Deny User not Enforced

Fixed in Instance 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](#)

Chapter 4

Known Problems and Workarounds

This chapter describes known problems, restrictions, and workarounds for Oracle JDBC for Rdb Release 7.3.3.1.

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 database using the same server, each client is allocated its own thread and has its own connection context within Rdb, but from the Rdb engines 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 lock wait 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 allowing better concurrency.

See *Lockwait and Maxtries* in your *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 `-verbosegc` 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 existing version of the 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 the GUI once a JDBC dispatcher has been defined.

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

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 reference 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 statement 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 reference to Blob columns in order to retrieve the dbkey of the parent table row. Code is also added to Insert and Update statement 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 component 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 has to 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 JDBC 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`

`DatabaseMetaData.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_functions.sql script. 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 7.3.3.0.

This section describes new and changed features in Oracle JDBC for Rdb 7.3.3.0.

5.1.1 Oracle Rdb PID now Displayed

(Also available 7.3.2.0 - Patch 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 or above, and the JDBC server used is release 7.3.2.0 (patch 2) or above.

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

5.1.2 Controller Command Show Executors

(Also available 7.3.2.0 - Patch 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.

5.1.3 Extra Timestamp Precision

(7.3.3.0 Patch 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 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 version 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 upto 7 decimal places (i.e. OpenVMS ticks, or 100 nanosecond units).

The extra timestamp precision is only available to applications using JDBC 7.3.3.0 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.

5.1.4 NlsLang Connection Switch

(7.3.3.0 Patch 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 actually 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.

5.1.5 Use Query Header as Description

(7.3.3.0 Patch 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](#)

5.2 Corrections in Release 7.3.3.0.

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.3.0.

5.2.1 Controller START SERVER Problems

Fixed in Instance Build 20131126.

The controller will fail with a Java NullPointerException 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 work around 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 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.

5.2.2 Executor Initialization Problem

Fixed in Instance Build 20131126.

Also fixed in release 7.3.2.0 Instance Build 20130517 (Patch 3).

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.

5.2.3 Idle Client Termination breaks Subsequent Connection

Fixed in Instance Build 20131126.

Also fixed in release 7.3.2.0 Instance Build 20130520 (Patch 4).

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 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.

5.2.4 Controller STOPSERVER command Problem

Fixed in Instance Build 20131126.

Also fixed in release 7.3.2.0 Instance Build 20130530 (Patch 5).

A problem introduced in the first non-patched 7.3.2.0 release of JDBC may cause the controller command line command STOPSERVER to raise an exception.

For example:

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

raises the following exception:

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

This has now been fixed.

5.2.5 Hang on Connection to Server when using SSL

Fixed in Instance Build 20131126.

Also fixed in release 7.3.2.0 Instance Build 20130612 (Patch 6).

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.

5.2.6 DatabaseMetadata Pattern Matches and Nulls

Fixed in Instance Build 20131126.

Also fixed in release 7.3.2.0 Instance Build 20131118 (Patch 6).

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.

5.2.7 DatabaseMetadata Missing Index Information

Fixed in Instance Build 20131126.

Also fixed in release 7.3.2.0 Instance Build 20131118 (Patch 6).

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

This method is used by tools such as the Rdb Addin 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.

5.2.8 Forced Client Termination may Crash Thin Server

Fixed in Instance 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 interrupted 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.

5.2.9 Show Clients may cause NullPointerException in Server

Fixed in Instance 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.

5.2.10 AccessViolation in MP Server

Fixed in Instance 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 AccessViolation 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=00000000xxxxx, PS=0000xxxx
RDBJDBCMPSHR73  CJDBCCTX_M  getDaMemUsage
RDBJDBCMPSHR73  CJDBCCTX_M  getStmntMemUsage
```

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.

5.2.11 Syntax Errors in Insert and Update Statements

Fixed in Instance 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,fld2) 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 work around 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,fld2) values ('55555',' TEST INTO ')
```

This has now been fixed.

5.2.12 Statement.getGeneratedKeys() throws RdbException

Fixed in Instance 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.

5.2.13 SQLException thrown in ResultSet.isBeforeFirst()

Fixed in Instance Build 20140520 (PATCH 3).

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.

5.2.14 PreparedStatement executeBatch() Problems

Fixed in Instance Build 20140526 (PATCH 3).

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 tabl 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.

5.3 New Features for Release 7.3.2.0.

This section describes new and changed features in Oracle JDBC for Rdb 7.3.2.0.

5.3.1 Server Options List Inheritance

When a 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 characteristic 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">

<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 7.3.2.0, 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 7.3.2.0.

5.3.2 Oracle JDBC for Rdb Manager Server

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.

5.3.3 Restricting Server Access by IP

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.

5.3.4 Executor Balancing

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.

5.3.5 MinFreeExecutors

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.

5.3.6 Executor Reuse

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.

5.3.7 LogFile Patterns

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.

5.3.8 New Server Configuration Option retainRdbSQLState

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.

5.3.9 New Connection Option *app*

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](#)

5.4 Corrections in Release 7.3.2.0.

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.2.0.

5.4.1 Event Flag Problem with long Executor name Prefixes

Fixed in Instance 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 raise, 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.

5.4.2 Executor Process Termination Problem

Fixed in Instance 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.

5.4.3 Some Server Characteristics not Correctly Inherited from DEFAULT

BUG 14510287.

Fixed in Instance 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.

5.4.4 Memory Problem with Pool Servers and Java 1.6.0-2

Fixed in Instance 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,  
.br/>.br/.
```

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.

5.4.5 Exception not Thrown when Record Locked during Update

Fixed in Instance 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.

5.4.6 Classpath Documentation Error

Fixed in Instance Build 20130426.

In prior versions of Oracle JDBC for Rdb, due to a typographic error, the `Driver Class` section of the `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 was 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 `User Guide` sections:

- `Results Class`
- `Blob Class`

This has now been fixed.

5.5 New Features for Release 7.3.1.0.

This section describes new and changed features in Oracle JDBC for Rdb 7.3.1.0.

5.5.1 “Owner” may be used in XML Configuration Files

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>
.
.
.
```

5.5.2 SQL Statement Restriction and Denial

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.

5.5.3 Event Notification

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.

5.6 Corrections in Release 7.3.1.0.

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.1.0.

5.6.1 Multi-process Server Connection Hang when Executor Dies

Fixed in Instance 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-existent 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 your *Oracle JDBC for Rdb User Guide* for more information on the server parameter `srv.lostExecSweep`.

5.6.2 Multi-process Executor Process Terminates Unexpectedly

Fixed in Instance 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 effect executors associated with active connections and not executors within the MP server free executor pool.

A work-around 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.

5.6.3 Pooled Server AutoRestart Problem

Fixed in Instance Build 20111117.

A problem introduced in release 7.3.0.2 (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 work-around 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.

5.6.4 getGeneratedKeys() and OutOfBounds Exception

Fixed in Instance 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.

5.6.5 Multi-process Server fails when Persona Used

Fixed in Instance Build 20111221.

A problem introduced in 7.3.0.2 (V7.3-02) 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 a 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+0xfffffb2]
#
# 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 00000000000001EF0
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 RJDDBC_PROC rjdbc_proc_GetPid
                                42606 000000000000029A1
00000000035C5481
RDBJDBCMPSHR73 RDBJDBC Java_rdb_JNI_StartProcess
```

The only work-around is to not use Persona for Multi-process servers.

This has now been fixed.

5.6.6 RestrictAccess being Applied even when Disabled

Fixed in Instance Build 20120201.

The JDBC 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.

5.6.7 Multi-process Problem in the Native Driver

Fixed in Instance Build 20120208.

Due to changes in thread interleaving introduced in Java JDK6.0 it is possible that applications using the `multi-process` 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 depend 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](#)

5.7 New Features for Release 7.3.0.2.

This section describes new and changed features in Oracle JDBC for Rdb 7.3.0.2.

5.7.1 Server Network Keep Alive

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 your *Oracle JDBC for Rdb User Guide* for more information.

5.8 Corrections in Release 7.3.0.2.

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.0.2.

5.8.1 Null pointer Exception during Server viability check by Pool Server

Fixed in Instance 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 7.3.0.0 (V7.3) 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.

5.8.2 MINUS and INTERSECT Problem

Fixed in Instance 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.

5.8.3 Memory Leak when Executor Process fails to Run

Fixed in Instance 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.

5.8.4 Lost Executor not caught by Multi-process Server

Fixed in Instance 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.

5.8.5 Multi-process Native Driver Problem

Fixed in Instance Build 20110623.

During the creation of connection contextes when using the Multi-process 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.

5.8.6 PreparedStatement.getGeneratedKeys() Problem

Fixed in Instance 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.

5.8.7 Excessive IOs during Metadata Retrieval

Fixed in Instance Build 20111002 (PATCH2).

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 revision during the life of the database.

This problem was introduced in 7.3.0.2 (first release) and has now been fixed.

5.8.8 Syntax error SQL-F-CONVARUND

Fixed in Instance Build 20111005 (PATCH3).

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 (first release) and has now been fixed.

5.8.9 Multi-process Server Hang

Fixed in Instance Build 20111006 (PATCH4).

The Multi-process (MP) server may hang during Resultset close operations.

A problem introduced in 7.3.0.2 (first release) 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 (first release) and has now been fixed.

5.8.10 SSL Socket Intrusion Problem

Fixed in Instance Build 20111102 (PATCH5).

Data injection into an existing open SSL sockets may cause problems with SSL-enabled Thin Servers.

Socket and Port security scanning software may inject data into exiting 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](#)

5.9 New Features for Release 7.3.0.1.

This section describes new and changed features in Oracle JDBC for Rdb 7.3.0.1.

5.9.1 Reopen Server Log files using Poll Subcommand

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 read or copied while the log files are currently being used by the server.

The thincontroller from Oracle JDBC for Rdb 7.3.0.1 (V7.3-01) , or later from later releases, used in conjunction with Oracle JDBC for Rdb servers from 7.3.0.1 and later releases 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](#)

5.10 Corrections in Release 7.3.0.1.

This section describes software errors corrected in Oracle JDBC for Rdb release 7.3.0.1.

5.10.1 Show Clients not Showing Column Names

Fixed in Instance 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 7.3.0.0 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>

```

5.10.2 Batched Statement Fails with MULTIPLE_RECORDS Exception

Fixed in Instance Build 20100611.

To provide additional JDBC driver functionality in Oracle JDBC for Rdb 7.3.0.0, 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 which 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.

5.10.3 Incompatibility between 7.3.0.0 Thin Driver and Prior Release Servers

Fixed in Instance 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 7.3.0.0 (V7.3) may prevent applications using the 7.3.0.0 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 release prior to 7.3.0.0 to a 7.3.0.0 server should still work correctly.

These incompatibility issues have now been resolved.

5.10.4 Global Memory leak when Executors are Run-down

Fixed in Instance Build 20100611.

Bug 9787051

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 a MP server runs out of global memory it cannot start-up new executor processes.

As this problem occurs when an executor is run-down when it is no longer required, a work-around 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.

5.10.5 Autorestart on Pooled Servers not Working

Fixed in Instance Build 20100611.

BUG 9656632

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 7.3.0.0 (V7.3) 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.

5.10.6 setFetchSize() hint Ignored by PreparedStatement

Fixed in Instance 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.

5.10.7 Missing Stmt Exception on Subsequent Execution of PreparedStatement

Fixed in Instance 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 7.3.0.0 (V7.3) JDBC and only occurs when using the Oracle JDBC for Rdb thin driver.

5.10.8 EOFException on READ_ROW with Nested Statements

Fixed in Instance 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 record 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();  
}
```

.
. .

5.10.9 DatabaseMetaData.getUDTs()

Fixed in Instance 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.

5.10.10 Using Multi-process with the JDBC Native Driver

Fixed in Instance Build 20100718.

A problem was introduced in Oracle JDBC for Rdb 7.3.0.0 (V7.3) preventing the use of the `multi-process` option within JDBC Native driver connections.

When a connection is attempted using the JDBC Native driver with a connection URL that contains the `@multi-process=true` switch the following exception was raised:

```
INFO: server version needs to be V7.3 or later
```

This has now been fixed.

5.10.11 Prepared Statement not Closing underlying Cursor

Fixed in Instance Build 20100721.

A problem introduced in Oracle JDBC for Rdb 7.3.0.0 (V7.3) 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_XXXXXXXXXX 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.

5.10.12 Release Statement Synchronization Problem

Fixed in Instance Build 20100802.

Oracle JDBC for Rdb release 7.3.0.0 (V7.3) introduced improvements in the network handling of selection statements. A problem in the optimization of network IOs introduced in 7.3.0.0 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.

5.10.13 Unitialized SQLCA Block in MP server

Fixed in Instance 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.

5.10.14 Controller SHOW CLIENTS and MP Server Problem

Fixed in Instance Build 20100315.

Note

This problem was reported fixed in Oracle JDBC for Rdb 7.3.0.0 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.

5.10.15 Documentation Error – Record Streaming

In the Oracle JDBC for Rdb 7.3.0.0 release notes the Performance Enhancements subsection of the New Features section mentions that Record Streaming is available from 7.3.0.0 onwards.

Unfortunately a number of problems required the removal of this feature prior to the 7.3.0.0 release, but the 7.3.0.0 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.

5.10.16 ResultSet.updateRow() and ResultSet.deleteRow() Problem

Fixed in Instance Build 20110127.

Bug 9668574

A problem introduced in the Oracle JDBC for Rdb 7.3.0.0 release 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.

5.10.17 Problem using Column Renaming with dbkey

Fixed in Instance Build 20110127.

Changes to SQL statement parsing introduced in Oracle JDBC for Rdb 7.3.0.0 inadvertently 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 work-arounds 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.

5.10.18 Class cast error on Scaled integer Retrieval after ResultSet.insertRow()

Fixed in Instance 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.

5.10.19 `ResultSet.deleteRow()` Behaviour Change

Fixed in Instance 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.

5.10.20 Underlying Blob Handles not Released when using ResultSet.getBlob()

Fixed in Instance 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.

5.10.21 Sequence Values not Visible to ResultSet get Methods using Column Name

Fixed in Instance Build 20110204.

Bug 9659036

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 work-around for the first problem is to use the column index instead of the name in the get method.

An alternate work-around 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 work-around for the `ResultSetMetaData.getColumnName()` problem.

These problems have now been fixed.

5.10.22 Scaled Integer Problem

Fixed in Instance 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.

5.10.23 Server Matching Exception may Stop Poll Handling

Fixed in Instance 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 down 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 request until it has been restarted.

This behaviour 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](#)

5.11 New Features for Release 7.3.0.0.

This section describes new and changed features in Oracle JDBC for Rdb 7.3.0.0.

5.11.1 Shutdown Thread

Starting with Oracle JDBC for Rdb release 7.3.0.0 (V7.3) 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 7.3.0.0 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.

5.11.2 Driver.attach()

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.

5.11.3 Returning List of Known Databases

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.

5.11.4 Controller Enhancements

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.

5.11.5 RDB_EXT.JAR file

Oracle JDBC for Rdb 7.3.0.0 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 include:

- Hibernate RdbDialect.java source
- Oracle UCP enabling classes

Details of these extensions follow.

5.11.5.1 Hibernate RdbDialect.java Source

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 your Hibernate documentation for details on using third party JDBC drivers and Dialects.

5.11.5.2 Oracle UCP Enabling Classes

UCP or, Universal Connection Pool, is a new database feature included in Oracle database 11g 11.1.0.7, Oracle database 11.2.0.x and Oracle AS 11g 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>

5.11.6 Performance Enhancements

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.

5.11.6.1 Internal client/server message protocol changes

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.

5.11.6.2 Enhanced SQL Statement Caching

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.

5.11.6.3 Results Caching

7.3.0.0 will allow limited caching of resultSets on the client side. This is particularly 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.

5.11.6.4 Server/Executor Synchronization

Starting with release 7.3.0.0, 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](#)

5.12 Corrections in Release 7.3.0.0.

This section describes software errors corrected in Oracle JDBC for Rdb 7.3.0.0.

5.12.1 Server Startup Failure when using CFG File

Fixed in Instance 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 work-around for this problem is to use an XML-based configuration file instead.

5.12.2 AccessViolation on Disconnect when Inserting Blobs

Fixed in Instance 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.

5.12.3 PreparedStatement and Parameter Markers Known Problem now Resolved

Fixed in Instance Build 20090702

In previous version 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 7.2.4.0 (V7.2-4), 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 versions 7.2.4.0 and higher.

5.12.4 Controller SHOW CLIENTS and MP Server Problem

Fixed in Instance 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 7.3.0.0 release. This has subsequently been rectified in the 7.3.0.1 release.

5.12.5 Possible Memory Leak when Updating Blob Columns

Fixed in Instance 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.

5.12.6 Access Violation with Trace and Network Dump

Fixed in Instance 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.

5.12.7 Named Input Parameters not Working with CallableStatements

Fixed in Instance 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 work-around for the problem is to specify the parameter index value instead of the name.

▲ [contents](#)

5.13 New Features for Release 7.2.5.5.

This chapter describes new and changed features in Oracle JDBC for Rdb release 7.2.5.5.

None.

▲ [contents](#)

5.14 Corrections in Release 7.2.5.5.

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.5.

5.14.1 Long-running Query Holds up New Connections in MP Server

Fixed in Instance 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](#)

5.15 New Features for Release 7.2.5.4.

This chapter describes new and changed features in Oracle JDBC for Rdb release 7.2.5.4.

None.

▲ [contents](#)

5.16 Corrections in Release 7.2.5.4.

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.4.

5.16.1 Access Violation at Java_rdb_JNI_SetStrVal

Fixed in Instance 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.

5.16.2 DCL Command Line Too Long

Fixed in Instance 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.

5.16.3 Access Violation during DriverManager.getConnection() when Database Specification is Missing

Fixed in Instance 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.

5.16.4 Unaligned Memory Faults on IA64

Fixed in Instance Build 20090826.

Several unaligned memory faults have been located and fixed in the shared images associated with Oracle JDBC for Rdb on IA64.

5.16.5 Read-Only Transactions not Enforced on Connection Switch

Fixed in Instance 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.

5.16.6 Sockets not Correctly Closing on OpenVMS Clients causing Accumulation of Mailboxes

Fixed in Instance 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.

5.16.7 Cast problem when Converting String to Date/Time

Fixed in Instance 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](#)

5.17 New Features for Release 7.2.5.3.

This chapter describes new and changed features in Oracle JDBC for Rdb release 7.2.5.3.

None.

▲ [contents](#)

5.18 Corrections in Release 7.2.5.3.

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.3

5.18.1 Interaction of DatabaseMetaData methods with Blobs may Crash the Thin Server

Fixed in Instance 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 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.

5.18.2 BigDecimal scaling Incorrect when used with PreparedStatement setObject() Methods

Fixed in Instance 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 work-around for this problem is to use the `PreparedStatement.SetBigDecimal()` methods instead.

This problem has now been fixed.

5.18.3 Connection.nativeSQL() method Throws Null Pointer Exception

Fixed in Instance Build 20080623

Calling the `Connection.nativeSQL()` method will result in a `NullPointerException` exception being raised.

This has now been fixed.

5.18.4 Calling Resultset.isLast() method May Change Transaction Behavior

Fixed in Instance 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](#)

5.19 New Features for Release 7.2.5.2.

This chapter describes new and changed features in Oracle JDBC for Rdb release 7.2.5.2.

5.19.1 DEC_KANJI and DEC_HANZI Support Enabled

Support was added to V7.1.3 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](#)

5.20 Corrections in Release 7.2.5.2.

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.2

5.20.1 ResultSet.getBigDecimal() not Working with System ResultSets

Fixed in Instance 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.

5.20.2 Setting TraceLevel fails when using Hexadecimal Notation

Fixed in Instance 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.

5.20.3 Delimited Identifier Problem in AS clause of Select Statement

Fixed in Instance Build 20070731

A problem introduced in 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 work-around 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.

5.20.4 Configuration file problem in "DEFAULT" Server Definition

Fixed in Instance 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 work around 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.

5.20.5 Pool Server May choose Incompatible Pooled Server when User Restriction Enabled

Fixed in Instance 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.

5.20.6 Potential Problem when Dumping SQLDA in Trace

Fixed in Instance 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.

5.20.7 Connection.getCatalog() Returns Wrong Value for Single Schema Databases

Fixed in Instance 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.

5.20.8 Potential Memory Leak with Views

Fixed in Instance 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](#)

5.21 New Features for Release 7.2.5.1.

This chapter describes new and changed features in Oracle JDBC for Rdb release 7.2.5.1.

5.21.1 SQLDA Dumping

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.

5.21.2 failSAFE IP with Pool Servers

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.

5.21.3 HandshakeTries and HandshakeWait on Multi-process Native Connections

The `multi-process` 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.

5.21.4 Server Access Security Enhancements

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.

5.21.5 Restriction on using Multiple Blob fields in Join now Removed

In previous version the following limitation was specified:

Blobs will only be returned correctly from a 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](#)

5.22 Corrections in Release 7.2.5.1.

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.1.

5.22.1 Incorrect Row Number Returned after ResultSet.getLast() call

Fixed in Instance 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
```

5.22.2 Pool Server Startup of Pooled Servers may fail When Persona is Used

Fixed in Instance 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.

5.22.3 Last Column in Select List may be Inaccessible in Some Queries

Fixed in Instance 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.

5.22.4 Abnormal Client Termination may Prevent Executor Re-use

Fixed in Instance Build 20061221

If a client application using a 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.

5.22.5 Decimal Column Problem with Native Driver

Fixed in Instance 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
```

Application using the rdbThin driver should not encounter this problem.

5.22.6 'EFN xx is not available' Message on Executor Startup

Changed in Instance 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.

5.22.7 Extraneous log message during Auto-restart check by Pool Server

Changed in Instance 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.

5.22.8 Logfile not Correctly set for Servers Started Using the Controller

Fixed in Instance 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](#)

5.23 New Features for Release 7.2.5.0.

5.23.1 Persona

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 7.1.4.0, but was omitted from the release notes.

5.24 Corrections in Release 7.2.5.0.

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5.0.

5.24.1 Incorrect SQLSRV_JDBC_SERVER_STARTUP72 Installed with V7.2-41 Oracle JDBC for Rdb Kit

Fixed in Instance 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_QSNAM_* logical name properly and may cause problems when you try to use this file with SQL/Service Thin server startup.

The following line of DCL command in this file is incorrect:

```
$ nam      := 'f$logical("RDB$JDBC_QSNAM_'port'")
```

It should read:

```
$ nam      = f$logical("RDB$JDBC_QSNAM_'port'")
```

This has now been fixed.

5.24.2 Multi-process Server May Show Continuous DIO Activity Even When Idle

Fixed in Instance 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.

5.24.3 Client idleTimeout Does Not Work for Prestarted and Reused Executors

Fixed in Instance 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.

5.24.4 Syntax Error in Query Generated for DatabaseMetaData.getTables

Fixed in Instance 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.

5.24.5 Show Clients in Controller may Crash Connected Thin Server

Fixed in Instance Build 20060620

A change in handshake protocol in Oracle JDBC for Rdb 7.1.4.1 (V7.1-41) 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](#)

5.25 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.

5.25.1 Client and Server Timeout Feature

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.

5.25.2 Executor Name Prefix

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](#)

5.26 Corrections in Release 7.2.4.1.

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.4.1.

5.26.1 Release Notes Specify Incorrect Installation Directory for RDBJDBCCFG.XML

Fixed in Instance Build 20060130

The release notes for Oracle JDBC for Rdb release 7.1.2.0 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.

5.26.2 Persona Not Handled Correctly by the Multi-process and Pool Servers

Fixed in Instance 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.

5.26.3 Multi-process Server / Executor Handshake Timeout May Be Too Short on Heavily Loaded Systems

Fixed in Instance 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 version 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.

5.26.4 Problems with `srv.idleTimeout` and `srv.bindTimeout` Configuration Variables and Their Use with SSL servers

Fixed in Instance Build 20060208)

The *Oracle JDBC for Rdb User Guide* for release 7.1.3.0 (V7.1-3) 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.

5.26.5 IA64 Problem Causes Array Out of Bounds Exception When Handling String Indexing

Fixed in Instance Build 20060208

A problem in the way Java on IA64 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 *getTableNames* method has now been changed to a mechanism other than *indexOf* to carry out its operation. This problem should no longer be seen.

5.26.6 Comments within SQL Text Not Handled Correctly

Fixed in Instance 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 <rdbjdbcdrv: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
```

5.26.7 Prepared Statements May Cause a Memory Leak with Multi-process Servers

Fixed in Instance 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](#)

5.27 Corrections in Release 7.2.4.0.

This section describes software errors corrected in Oracle JDBC for Rdb 7.2.4.0.

5.27.1 Maximum Size of Single Data Row Increased to 65,272 Octets

Fixed in Instance 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.

5.27.2 Another Connection Overlap Window Found with Pool Servers

Fixed in Instance 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 7.1.3.3 release 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.

5.27.3 SSL Server Information Not Correctly Set from XML-Formatted Configuration File

Fixed in Instance 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](#)