Oracle® JDBC for Rdb
Release Notes

February 2011

Release 7.3.0.1

Oracle JDBC for Rdb Release Notes, Release 7.3.0.1

safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark of Oracle Corporation. All other company or product names mentioned are used for identification purposes only and may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Preface

Purpose of This Manual

The Oracle JDBC for Rdb 7.3.0.1 release notes summarize new features, corrections to software, restrictions, workarounds, and problems. They also include new features and corrections provided in releases 7.2.4 through 7.3. 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

Document Structure

This document consists of five chapters:

Chapter 1	Describes location of documents and installation directions
Chapter 2	Describes new features and technical changes in this release
Chapter 3	Describes corrected software errors in this release
Chapter 4	Describes known problems, restrictions, and workarounds
Chapter 5	Describes new features and corrected software errors in releases 7.2.4 through 7.3.

1.1 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.

Chapter 1 Installation and Documentation

This chapter contains installation and documentation information for Oracle JDBC for Rdb release 7.3.0.1

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 SYS$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

Please note: the minimum version of Java required has changed. JDK 5.0 is now the minimum SDK/RTE version supported on all platforms for both server and client Oracle JDBC for Rdb components

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://www.compaq.com/java/documentation/index.html - Java 2.
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	2048
Wsquo	4096
Wsextent and Wsmax	16384
Pgflquo	1500000
bytlm	1000000
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 V7.3 Oracle JDBC for Rdb.

If you were participating in a BETA trial of V7.3 Oracle JDBC for Rdb then you must remove any previously installed BETA kits before proceeding with the installation of this release.

Failure to remove previous V7.3 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 V7.3 BETA that may

cause connection failures if V7.3 BETA thin driver jars are used to connect to a V7.3 server or vice-versa.

In addition, ensure that all deployed copies of the V7.3 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 RDBJDBCV73xxxxx.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
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
RDBJDBC_<version>_RELNOTES.PDF RDBJDBC_<version>_RELNOTES.HTML	Oracle JDBC for Rdb Release Notes
RDBJDBC_<version>.RELEASE_NOTES	Text version of Oracle JDBC for Rdb Release Notes
RDBJDBC_USERGUIDE.HTML 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

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 RDBJDBCV73xxxx.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 RDBJDBCSHR RDB$JDBC_HOME:RDBJDBCSHR73.EXE
$ DEFINE/SYSTEM RDBJDBCMPSHR RDB$JDBC_HOME:RDBJDBCMPSHR73.EXE
$ DEFINE/SYSTEM RDBJDBCEXEC RDB$JDBC_HOME:RDBJDBCEXEC73.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.
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]. If an rdbjdbccfg.xml file is not already present in the sys$common:[rdb$jdbc] directory, the installation procedure will copy the one included in the installation there.

```
$ 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]
```

RDBJDBCSHR to point to the shared image RDBJDBCSHR73.EXE.

```
$ define/system RDBJDBCSHR –
SYS$COMMON:[RDB$JDBC.0703-0V0A18]RDBJDBCSHR73.EXE
```

RDBJDBCMPSHR to point to the shared image RDBJDBCMPSHR73.EXE.

```
$ define/system RDBJDBCMPSHR –
SYS$COMMON:[RDB$JDBC.0703-0V0A18]RDBJDBCMPSHR73.EXE
```

RDBJDBCEXEC to point to the shared image RDBJDBCEXEC73.EXE.

```
$ define/system RDBJDBCEXEC –
SYS$COMMON:[RDB$JDBC.0703-0V0A18]RDBJDBCEXEC73.EXE
```

You must define the RDB$JDBC_HOME logical name if you want to use a Thin

Multi-process server or use the Thin controller or Pool servers 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 RDBJDBCCHECKUP.CLASS file. During the installation RDBJDBCCHECKUP.CLASS is copied to RDB$JDBC_HOME.
Copy this file to your default directory and then you can invoke it using Java.
You will be prompted for a username and password and an Oracle Rdb database to test the installation against. If the test succeeds, the text "Your JDBC installation is correct." is displayed.

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

Test the Thin server by using the following commands:

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

```
$stop rdbthinsrvtest
```

Note

Because 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.

Chapter 2 Enhancements Provided in Oracle JDBC for Rdb Release 7.3.0.1

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

2.1 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 JDBC server cannot be be read or copied while the log files are currently being used by the server.

The V7.3-01 (or above) thincontroller used in conjunction with JDBC servers from V7.3-01 and above 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.

Chapter 3
Problems Corrected

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

3.1 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 the 7.3 JDBC release 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>
```

3.2 Batched Statement Fails with MULTIPLE_RECORDS Exception.

Fixed in Instance Build 20100611.

To provide additional JDBC driver functionality in Oracle JDBC for Rdb V7.3, during the preliminary parsing of SQL update or insert statements, the Rdb JDBC 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 succesfully.

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 Rdb JDBC 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.

3.3 Incompatibility between V7.3 Thin driver and Pre-V7.3 Servers.

Fixed in Instance Build 20100611.

Ideally, when upgrading JDBC, 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 V7.3 may prevent applications using the V7.3 Thin driver from connecting correctly to Rdb JDBC servers from earlier Oracle JDBC for Rdb versions.

The connection of applications using pre-V7.3 Thin drivers to a V7.3 server should still work correctly.

These incompatibility issues have now been resolved.

3.4 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.

3.5 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 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.

3.6 setFetchSize() hint ignored by PreparedStatement

Fixed in Instance Build 20100623

In prior versions of JDBC, 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.

3.7 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 V7.3 JDBC and only occurs when using the Oracle JDBC for Rdb thin driver.

3.8 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();
}
```

3.9 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.

3.10 Using Multiprocess with the JDBC Native Driver

Fixed in Instance Build 20100718

A problem was introduced in V7.3 JDBC preventing the use of the Multiprocess option within JDBC Native driver connections.

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

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

This has now been fixed.

3.11 Prepared Statement not Closing underlying Cursor.

Fixed in Instance Build 20100721

A problem introduced in JDBC 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_xxxxxxxxx was already open
```

The following code snippet show 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.

3.12 Release Statement synchronization problem

Fixed in Instance Build 20100802

Version 7.3 JDBC introduced improvements in the network handling of selection statements. A problem in the optimization of network IOs introduce in V7.3 JDBC 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.

3.13 Unitialized SQLCA block in MP server

Fixed in Instance Build 20100906

A problem in how internal SQLCA blocks were initialized by the Multiprocess 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.

3.14 Controller SHOW CLIENTS and MP Server problem

Fixed in Instance Build 20100315.

Note: This problem was reported fixed in V7.3 but unfortunately, the fix was incomplete.

When the command SHOW CLIENTS or SHOW ALL CLIENTS is issued within the Controller, if the recipient server is a Multi-Process server (MP Server), an executor process may be started up by the server to execute the request.

A problem in the executor code prevents the executor from closing down correctly if the maximum number of free executors has already been reached for the MP Server.

This new executor process will not be added to the free executor pool, but will not be shutdown and thus will remain active on the system.

If the controller SHOW CLIENTS command is issued again, the number of executor processes may increase until an Open VMS quota or process quota is exceeded.

This problem has now been fixed.

3.15 Documentation Error – Record Streaming

In the V7.3 release notes the Performance Enhancements subsection of the New Features section mentions that Record Streaming is available from V7.3 onwards.

Unfortunately a number of problems required the removal of this feature prior to the V7.3 release, but the V7.3 releases notes were not correctly updated to remove reference to the feature prior to product shipment.

This feature may be made available in a later version of Oracle JDBC for Rdb.

3.16 ResultSet.updateRow() and ResultSet.deleteRow() problem

Fixed in Instance Build 20110127.

Bug 9668574

A problem introduced in the V7.3 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.

3.17 Problem using column renaming with dbkey.

Fixed in Instance Build 20110127.

Changes to SQL statetment parsing introduced in V7.3 inadvertantly introduced a problem in queries that access the dbkey of records directly.

For example the following query:

```
select dbkey as ID, last_name from employees;
```

may cause the following exception to be raised:

```
Exception: Unhandled exception@Statement.execute :
java.lang.ArrayIndexOutOfBoundsException:S1000
```

This problem only occurs when the dbkey is retrieved as the first named column of the selection expression and the column name is changed by using the 'AS' clause.

Some simple 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.

3.18 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.

3.19 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 versions 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 10^{th} row, then after removing the record, the current row position would still be the 10^{th} 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.

3.20 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 a 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 JDBC 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.

3.21 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.

3.22 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.

3.23 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.

Chapter 4
Known Problems and Workarounds

This chapter describes known problems, restrictions, and workarounds for Oracle JDBC for Rdb release 7.3.0.1

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 that 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

Using Java Fast VM on OpenVMS when you start up 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 Limitations

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

```
Blob.setBytes

Blob.setBinaryStream

Clob.setString

Clob.setAsciiStream

Clob.setCharacterStream

Clob.truncate

Connection.setSavepoint

Connection.rollback(savepoint)

Connection.releaseSavepoint

DatabaseMeteData.getSQLKeywords

PreparedStatement.setRef

PreparedStatement.setArray

PreparedStatement.setNull(int,int,String)

PreparedStatement.setURL(int,URL)

ResultSet.getRef

ResultSet.getArray

ResultSet.updateAsciiStream

ResultSet.updateBinaryStream

ResultSet.updateCharacterStream

ResultSet.updateRef

ResultSet.updateArray

ResultSet.rowUpdated

ResultSet.rowInserted

ResultSet.rowDeleted
```

```
ResultSet.updateBytes

Statement.cancel

Statement.setQueryTimeout

Statement.getMoreResults

Statement.executeUpdate(String sql, int autoGeneratedKeys)

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

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

Statement.execute(String sql, int autoGeneratedKeys)

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

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

- The following features or datatypes in JDBC 2.0 and JDBC 3.0 are not supported:
    - Array
    - Ref
    - Clob
    - User Defined datatypes
    - Scroll cursors
    - Savepoints
- Auto-generated keys

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

- 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.

- 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.

Chapter 5
New Features and Corrections in Previous Releases

5.1 New Features for Release 7.3

This chapter describes new and changed features in Oracle JDBC for Rdb release 7.3.

5.1.1 Shutdown Thread

Starting with Oracle JDBC for Rdb release 7.3 the JDBC drivers will create a shutdown thread when they are initially invoked. The purpose of this thread is to use the shutdown-hook feature provided by OpenVMS that will execute the thread at shutdown. The shutdown thread will ensure that any connection left open by the application will be correctly disconnected prior to the application shutdown proceeding, thus preventing application hangs at shutdown.

The application developer does not need to change any code for this shutdown feature to be enabled, as long as the application is using V7.3 JDBC 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.

5.1.2 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.

5.1.3 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.

5.1.4 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.

5.1.5 RDB_EXT.JAR file

Oracle JDBC for Rdb V7.3 installation now includes the RDB_EXT.JAR file that contains classes and java source for extensions to Oracle JDBC for Rdb.

The following extensions are include:

- Hibernate RdbDialect.java source
- Oracle UCP enabling classes

Details of these extensions follow.

5.1.5.1 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.

5.1.5.2 Oracle UCP enabling classes

UCP or, Universal Connection Pool, is a new database feature included in Oracle database 11*g* 11.1.0.7, Oracle database 11.2.0.x and Oracle AS 11*g* R1. UCP works with any java based connections, e.g., JDBC, JCA, LDAP.

The RDB_EXT.JAR file installed with Oracle JDBC for Rdb, provides the necessary classes to enable Oracle JDBC for Rdb drivers to be used in the connection pools created by UCP.
The class

`oracle.rdb.jdbc.rdbExt.RdbDataSource`

found in the RDB_EXT.JAR may be used as a Connection Factory class for UCP, for example:

```
import oracle.ucp.jdbc.PoolDataSource;
import oracle.ucp.jdbc.PoolDataSourceFactory;
.
.
.
      PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
      pds.setConnectionFactoryClassName("oracle.rdb.jdbc.rdbExt.RdbDataSource");
      pds.setURL("jdbc:rdbThin://localhost:1701/my_dir:mf_personnel");
      pds.setUser("jdbc_user");
      pds.setPassword("jdbc_user");
      pds.setInitialPoolSize(0);
      Connection conn2 = null;

      for (int i = 0; i< 5 ; i++ )
      {
         conn2 = pds.getConnection();
         Statement sc = conn2.createStatement();
.
.
.
         rs.close();
         sc.close();
         conn2.close();
      }
.
.
.
```

For more details, see the UCP Users Guide at

http://www.oracle.com/technology/tech/java/sqlj_jdbc/UCP_dev_guide.pdf

5.1.6 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.

5.1.6.1 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.

5.1.6.2 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, V7.3 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.

5.1.6.3 Results Caching

V7.3 will allow limited caching of resultSets on the client side. This is particulary useful in multi-tiered environments for statements that are repeatedly executed giving the same results each time, especially in a pooled connection environment. See the section “Results Cache” in the Oracle JDBC for Rdb User Guide for more details.

5.1.6.4 Server/Executor Synchronization

Starting with release V7.3, the method used to synchronize operations between a Multi-Process server and its sibling executors has been improved to reduce the amount of CPU required and to speed-up operations.

5.2 Corrections in Release 7.3

This section describes software errors corrected in Oracle JDBC for Rdb release 7.3.

5.2.1 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.

5.2.2 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.

5.2.3 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 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 V7.2-4 and higher.

5.2.4 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 V7.3 release. This has subsequently been rectified in the V7.3-01 release.

5.2.5 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.

5.2.6 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

Bit	Hexadecimal Value	Decimal Value	Traces
			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.

5.2.7 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.

5.3 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.

5.4 Corrections in Release 7.2.5.5

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

5.4.1 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.

5.5 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.

5.6 Corrections in Release 7.2.5.4

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

5.6.1 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.

5.6.2 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.

5.6.3 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.

5.6.4 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.

5.6.5 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.

5.6.6 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.

5.6.7 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.

5.7 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.

5.8 Corrections in Release 7.2.5.3

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

5.8.1 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.

5.8.2 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.

5.8.3 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.

5.8.4 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.

5.9 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.

5.9.1 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.

5.10 Corrections in Release 7.2.5.2

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

5.10.1 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.

5.10.2 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.

5.10.3 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.

5.10.4 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.

5.10.5 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.

5.10.6 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.

5.10.7 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.

5.10.8 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.

5.11 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.

5.11.1 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.

5.11.2 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.

5.11.3 HandshakeTries and HandshakeWait on Multiprocess Native Connections

The multiprocess option on native connections allows the use of executor sub-processes to carryout Rdb connections on behalf of your application using the RdbNative driver. You now have the capability of specifying handshake options during the initial communication handshake protocol used by the main and associated sub-processes.

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

5.11.4 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 addition password before access via the server will be granted.

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

5.11.5 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.

5.12 Corrections in Release 7.2.5.1

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

5.12.1 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
```

5.12.2 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.

5.12.3 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 V7.1.3 of the Oracle JDBC for Rdb drivers.

5.12.4 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.

5.12.5 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.

5.12.6 '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.

5.12.7 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.

5.12.8 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.

5.13 New Features for Release 7.2.5

5.13.1 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, but was omitted from the release notes.

5.14 Corrections in Release 7.2.5

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.5

5.14.1 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 V7.2-41 installation kit for Oracle JDBC for Rdb.

This version of the file does not set up the RDB$JDBC_SQSNAM_* logical name properly and may cause problems when you try to use this file with SQL/Service Thin server startup.

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

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

It should read:

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

This has now been fixed.

5.14.2 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.

5.14.3 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.

5.14.4 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.

5.14.5 Show Clients in Controller may Crash Connected Thin Server

Fixed in Instance Build 20060620

A change in handshake protocol in V7.1-41 of Oracle JDBC for Rdb 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.

5.15 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.

5.15.1 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.

5.15.2 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.

5.16 Corrections in Release 7.2.4.1

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

5.16.1 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 incorrectly specified that the RDBJDBCCFG.XML file would be copied to SYS$COMMON:[RDB$JDBC] directory. The RDBJDBCCFG.XML is actually coped 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.

5.16.2 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 Oracle JDBC for Rdb 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.

5.16.3 Multi-Process Server / Executor Handshake Timeout May Be Too Short on Heavily Loaded Systems

Fixed in Instance Build 20060130

When a Multiprocessor 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.

5.16.4 Problems with srv.idleTimeout and srv.bindTimout Configuration Variables and Their Use with SSL servers

Fixed in Instance Build 20060208)

The *Oracle JDBC for Rdb User Guide for Version 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.

5.16.5 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 *getTableName* method has now been changed to a mechanism other than *indexOf* to carry out its operation. This problem should no longer be seen.

5.16.6 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 <rdbjdbcsrv:execute_immediate> %SQL-F-EXESELSTA,
Attempted to EXECUTE a SELECT statement:RR000
```

The JDBC driver could not correctly determine the type of statement and used the wrong underlying SQL operation to attempt to execute it.

The drivers now extract out comments prior to determining the statement type and sending the native SQL down to Oracle Rdb. The drivers will now correctly parse out C and SQL type comments, for example:

```
/* comment */
! this comment will be terminated at the next line break
-- this comment will be terminated at the next line break
// this comment will be terminated at the next line break
```

5.16.7 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.

5.17 Corrections in Release 7.2.4

This section describes software errors corrected in Oracle JDBC for Rdb release 7.2.4.

5.17.1 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.

5.17.2 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 V7.1.3.3 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.

5.17.3 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.