Oracle® JDBC for Rdb
Release Notes

June 2008

Release 7.2.5.3

Oracle JDBC for Rdb Release Notes, Release 7.2.5.3

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 7.2.5.3 release notes summarize new features, corrections to software, restrictions, workarounds, and problems. They also include new features and corrections provided in release 7.2.4, release 7.2.4.1, release 7.2.5, release 7.2.5.1 and release 7.2.5.2 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, 7.2.4.1, 7.2.5 , 7.2.5.1 and 7.2.5.2

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 7.2.5.3

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 RDBJDBC72
```

- 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 RDBJDBC72
```

- 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 RDBJDBC72/SELECT=filename.ext
```

The Oracle JDBC for Rdb documentation is also available on MetaLink 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	V7.3-2	V8.2-1
HP Java tm SDK/RTE	V1.4.1	V1.4-21
Oracle Rdb	V7.1-24	V7.2

Note

In prior V7.2 releases of Oracle JDBC for Rdb a minimum version of V7.2 Oracle Rdb was required. This now has been relaxed and is now the same as the V7.1 releases of Oracle JDBC for Rdb. V7.2 Oracle Rdb is still the required minimum version on Integrity.

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 tmSDK/RTE	V1.4.1

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

Software	Minimum Version	
	Alpha	Integrity
Oracle SQL/Services	V7.1.6	V7.2

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

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

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

http://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 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-RDBJDBC72-V0702-xxxxxx-1.PCSI where ppp will be the platform and xxxxxx will be the build instance of this kit, for example:

```
  ORCL-AXPVMS-RDBJDBC72-V0702-5V0672-1.PCSI
or
  ORCL-I64VMS-RDBJDBC72-V0702-5V0672-1.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 RDBJDBCV72xxxxx.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
RDBJDBCEXEC72.EXE	Executor image in conjunction with a Multi-process server
RDBJDBCCFG.XML	Example XML formatted configuration file
RDBJDBCSHR72.EXE	Shared image required for Oracle Rdb database access
RDBJDBCMPSHR72.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
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_FAQ.TXT(HTML,PDF)	Frequently asked questions
NATIVEDRIVERSANDJDEV.TXT(HTML)	How to change Oracle Developer to handle the Oracle JDBC for Rdb drivers
RDBJDBC_<version>_RELNOTES.PDF(HTML)	Oracle JDBC for Rdb Release Notes
RDBJDBC_<version>.RELEASE_NOTES	Text version of Oracle JDBC for Rdb Release Notes
RDBJDBC_USERGUIDE. HTML(PDF)	User guide
SQLSRV_JDBC_SERVER_STARTUP71.COM SQLSRV_JDBC_SERVER_STARTUP72.COM	Command procedures used by Oracle SQL/Services to start up an Oracle JDBC for Rdb server

1.3.2 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 RDBJDBCV72xxxx.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-RDBJDBC72-V0702-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 5V0672, and that the directory where the PCSI file can be found is MY_DIR.

```
$ PRODUCT INSTALL RDBJDBC72/SOURCE=MY_DIR

The following product has been selected:
     ORCL AXPVMS RDBJDBC72 V7.2-5V0672        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 RDBJDBC71 V7.2-5V0672: Oracle JDBC for Rdb

    Copyright © 1995, 2006,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 RDBJDBC72 V7.2-5V0672
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.0702-5V0672]

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.0702-5V0672]
$ 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:RDBJDBCSHR72.EXE
$ DEFINE/SYSTEM RDBJDBCMPSHR RDB$JDBC_HOME:RDBJDBCMPSHR72.EXE
$ DEFINE/SYSTEM RDBJDBCEXEC RDB$JDBC_HOME:RDBJDBCEXEC72.EXE

...100%

The following product has been installed:

   ORCL AXPVMS RDBJDBC72 V7.2-5V0672        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]

0702-5V0672.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.0702-5V0672]
```

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 RDBJDBCSHR72.EXE.

```
$ define/system RDBJDBCSHR SYS$COMMON:[RDB$JDBC.0702-
5V0672]RDBJDBCSHR72.EXE
```

RDBJDBCMPSHR to point to the shared image RDBJDBCMPSHR72.EXE.

```
$ define/system RDBJDBCMPSHR SYS$COMMON:[RDB$JDBC.0702-
5V0672]RDBJDBCMPSHR72.EXE
```

RDBJDBCEXEC to point to the shared image RDBJDBCEXEC72.EXE.

```
$ define/system RDBJDBCEXEC SYS$COMMON:[RDB$JDBC.0702-
5V0672]RDBJDBCEXEC72.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:
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:
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 7.2.5.3

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

None.

Chapter 3
Problems Corrected

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

3.1 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 metedat 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 metedata 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.

3.2 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.

3.3 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.

3.4 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.

Chapter 4
Known Problems and Workarounds

This chapter describes known problems, restrictions, and workarounds for Oracle JDBC for Rdb release 7.2.5.3

4.1 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.

A workaround is to alter the input query to force SQL to allocate a parameter size that is large enough to hold the input values that the application may use. For example:

```
ps = conn.prepareStatement( "select f1 from tab where" +
     " cast (f1 as VARCHAR(4)) like ?");
```

This will now return the correct value.

We are currently investigating ways of fixing this problem.

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 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 7.2.5.2

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

5.1.1 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.

5.2 Corrections in Release 7.2.5.2

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

5.2.1 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.

5.2.2 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.

5.2.3 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.

5.2.4 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.

5.2.5 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.

5.2.6 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 tracelLevel `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.

5.2.7 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.

5.2.8 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.

5.3 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.

5.3.1 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.

5.3.2 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.

5.3.3 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.

5.3.4 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 specifed.

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.

5.3.5 Retriction 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.

5.4 Corrections in Release 7.2.5.1

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

5.4.1 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
```

5.4.2 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 may 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.

5.4.3 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.

5.4.4 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.

5.4.5 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.

5.4.6 '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.

5.4.7 Extraneous log message during Auto-restart check by Pool Server.

Changed in Instance Build 20070306

A problem in how the Pool Server checked the availablility of its pooled servers when carrying out AutoRestart checking meant that an extraneous CLIENT LOST message would be logged by the pooled server everytime it was checked. The following message would be logged:

```
srv.DBActionHandler <idle> Connection to Client lost
```

This has now been fixed.

5.4.8 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.

5.5 New Features for Release 7.2.5

5.5.1 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.

5.6 Corrections in Release 7.2.5

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

5.6.1 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.

5.6.2 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.

5.6.3 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.

5.6.4 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.

5.6.5 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.

5.7 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.

5.7.1 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.

5.7.2 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.

5.8 Corrections in Release 7.2.4.1

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

5.8.1 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.

5.8.2 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.

5.8.3 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.

5.8.4 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.

5.8.5 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.

5.8.6 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
```

5.8.7 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.

5.9 Corrections in Release 7.2.4

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

5.9.1 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.

5.9.2 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.

5.9.3 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.