



November 1, 2002

Washington, DC



Java Stored Procedures – An Alternative to External Procedures?

Robert F. Edwards

Dulcian, Inc.

The Problem

#1 - Document application

- Deleting files from the server after loading images into the database.
- Requirement: Execute an O/S command

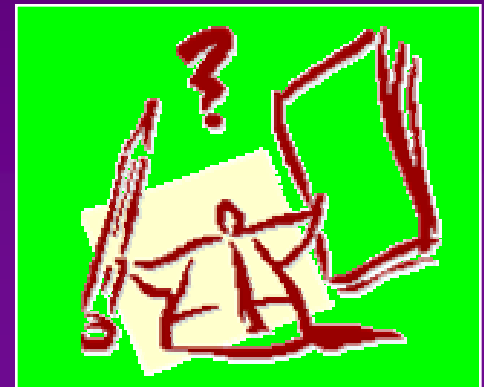
#2 - Checkout application for BRIM™

- Writing binary data to a file for editing (checking out an object).
- Requirement: Write a BLOB object to a file



The Solution – Options

- ◆ External Procedures (Oracle 8.0)
 - Strong candidate
- ◆ Java Stored Procedures (Oracle 8i)
 - New kid on the block



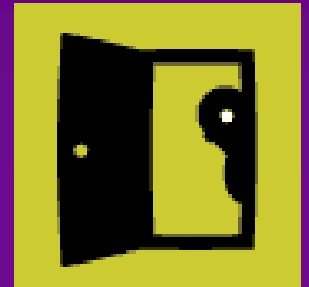
Agenda

- ◆ What are External Procedures?
 - Why use External Procedures?
 - How External Procedures Work
- ◆ What are Java Stored Procedures?
 - Why use Java Stored Procedures?
 - How Java Stored Procedures Work
- ◆ Examples – Application Solutions
- ◆ Summary



What are External Procedures?

- ◆ The External Procedure feature was introduced in Oracle8.0.
- ◆ An external procedure is a C program, stored in a shared library, and called from PL/SQL.
- ◆ An external procedure extends the functionality of PL/SQL beyond the database environment, and beyond the limitations of PL/SQL's own feature set.



Why use External Procedures?

◆ You need functionality not available in PL/SQL:

- Write binary objects to an O/S file.
- Call C library functions
 - Send Email

◆ You require functionality at the O/S level:

- Execute command line utilities
 - Create, copy, move, delete files
 - SQL Loader, Export, Import
- Call O/S functions
 - Get information on space usage on hard disk
 - Control operation of hardware device



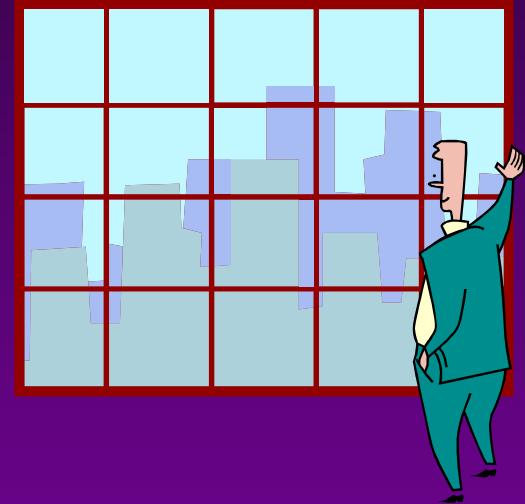


Advantages of External Procedures

- ◆ External procedures:
 - Are reliable
 - Are multi-threaded (Beginning with Oracle 8i)
 - Communication is bi-directional
 - Can be used as user-defined functions in SQL

How do External Procedures Work?

- ◆ Application
- ◆ PL/SQL Call Spec
- ◆ Listener
- ◆ Extproc
- ◆ Oracle Library
- ◆ External Procedure

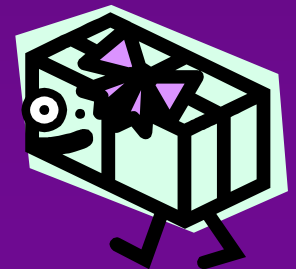


Application

- ◆ Your application – Requestor of External Procedure services
 - Client or server side PL/SQL block or stored procedure, running either on the database or in Oracle Forms
 - The calling code is ignorant of any language-specific calling requirements of the external program.
 - The application calls the call spec, passes data to it, and receives data upon its completion.

PL/SQL Call Spec

- ◆ A PL/SQL call spec acts as the “gateway” interface between the application and the external procedure.
- ◆ Hides complexity of the external program call
 - Handles language-specific calling requirements
 - Provides simplified calling parameters for application
 - Hides logging and system activity from the user
- ◆ The call spec initiates the actual call to the External Procedure Listener.





Call Spec for C

```
function ExecOSCcmd (syscmd  VARCHAR2)
return BINARY_INTEGER is
language C
library utloscmd
name "oscall"
parameters ( syscmd STRING,
              syscmd INDICATOR SHORT,
              syscmd LENGTH INT,
              RETURN );
```

Listener

- ◆ The External Procedure Listener runs in the Net8 environment.
- ◆ The Listener:
 - Runs in the background (a daemon, in Unix) – listens for requests
 - Spawns an instance of EXTPROC
- ◆ Entries for extproc included in:
 - TNSNAMES.ORA
 - LISTENER.ORA



Extproc

- ◆ Extproc is Oracle's external procedure calling program (extproc.exe)
 - Oracle's link to the outside environment
 - Calls external program using a library reference passed to it by the call spec
 - Extproc passes data from the call spec to the external program, and returns results back to PL/SQL.

Oracle Library

- ◆ A library object is a data dictionary reference to a file system directory and file name.
 - `'C:\oracle\ora81\rdbms\extproc\osutil.dll'`
- ◆ The library itself is referenced by its name:
 - `CREATE LIBRARY utloscmd AS`
`'C:\oracle\ora81\rdbms\extproc\osutil.dll'`
- ◆ Passed the library name, Extproc gets the file path from the dictionary, to locate and load the library.





The External Program

- ◆ Normally written in C/C++ (or other language)
- ◆ Compiled as a dynamic link library (.dll) or shared object (.so)
 - extern.so (Unix/Linux)
 - extern.dll (Windows)
- ◆ Resides on any file system directory
 - Referenced by Oracle library object

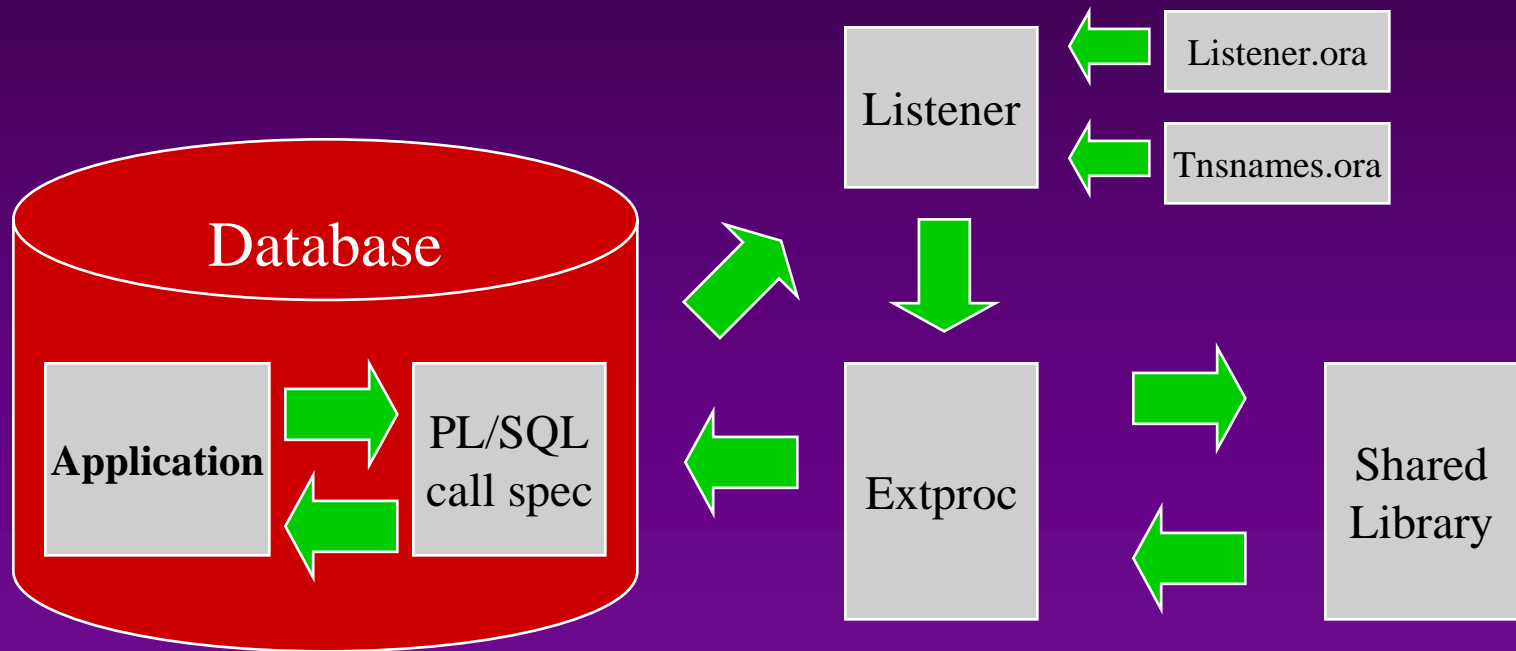


External Procedure Sample Source Code

```
#include <stdio.h>
#include <stdlib.h>
...
int oscall (char *cmd, short cind, int clen)
{
    /* Indicator/length validation code snipped */
    int ret = 0;
    ret = system(cmd);
    /* Return (ret) values:
    * 0 = Successful
    * All other = Error condition
    */
    return ret;
}
```

External Procedure Execution Flow

Server O/S Environment



Benefits of Shared Libraries

- ◆ The external procedure is a shared library rather than an executable.
 - The library is shared by many users.
 - Many routines can be bundled in a library.
- ◆ Uses its own separate memory space
- ◆ Provides full transaction support
- ◆ Database level security is enforced.





Limitations of External Procedures

- ◆ Excess inter-process communication required
 - Single-threaded in Oracle 8.0
 - Multi-threaded in Oracle 8i
- ◆ Cannot pass user-defined data types
 - Parameters must use conventional scalars.
- ◆ Memory usage is not in the SGA; is in O/S space.
- ◆ The shared library is closed after it is called; it is not cached.
 - Subsequent calls are much faster, however, within the same session.



Java Stored Procedures



What are Java Stored Procedures?

- ◆ Java stored procedures are Java class methods, which are “published” to SQL, reside on the database, and execute within the Oracle JVM environment.
- ◆ Introduced in Oracle8i (Rel. 8.1.5)



Why use Java Stored Procedures?

- ◆ Java Stored Procedures are part of the Oracle Java environment, called JServer, which is integrated with the database.
- ◆ Java is tightly integrated with PL/SQL and RDBMS functionality.
- ◆ Hundreds of Java classes available, provide APIs for user applications.



Advantages of Java Stored Procedures

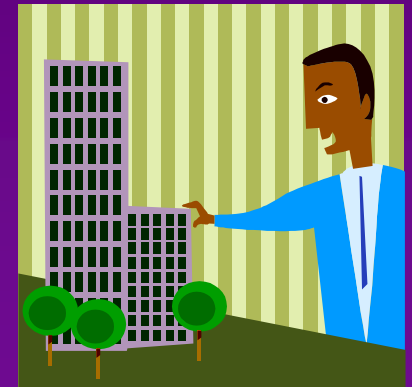
- ◆ Java is an open language – not proprietary
- ◆ Portable = runs on multiple platforms
- ◆ Extends the database functionality and programmability
- ◆ Resides on and executes within the database





How Do Java Stored Procedures Work?

- ◆ Application
 - Java applications
 - Non-Java applications
- ◆ Call Spec for Java
- ◆ Java Stored Procedure
- ◆ Java Environment
 - Development
 - Deployment
 - Execution



Application

◆ SQL-PL/SQL

- SQL DML statements
- PL/SQL blocks, functions, procedures, packages
- Triggers

◆ Java

- JDBC - prepared or callable statements
- Query-based JSPs (Java Server Pages)
- Servlets
- Java Beans



Call Spec for Java

- ◆ A function or procedure is used to invoke a Java stored procedure
 - “Top level” stored function or procedure
 - Packaged function or procedure
 - Member method of SQL object type
- ◆ Creating a call spec is called “publishing” the Java method
- ◆ Sample call spec for Java:

```
CREATE OR REPLACE function OSCommand  
(cmd Varchar2) Return Number  
AS LANGUAGE JAVA  
NAME 'OSUtil.OSCmd.OSExec(java.lang.String) return  
int';
```



Java Stored Procedure Requirements

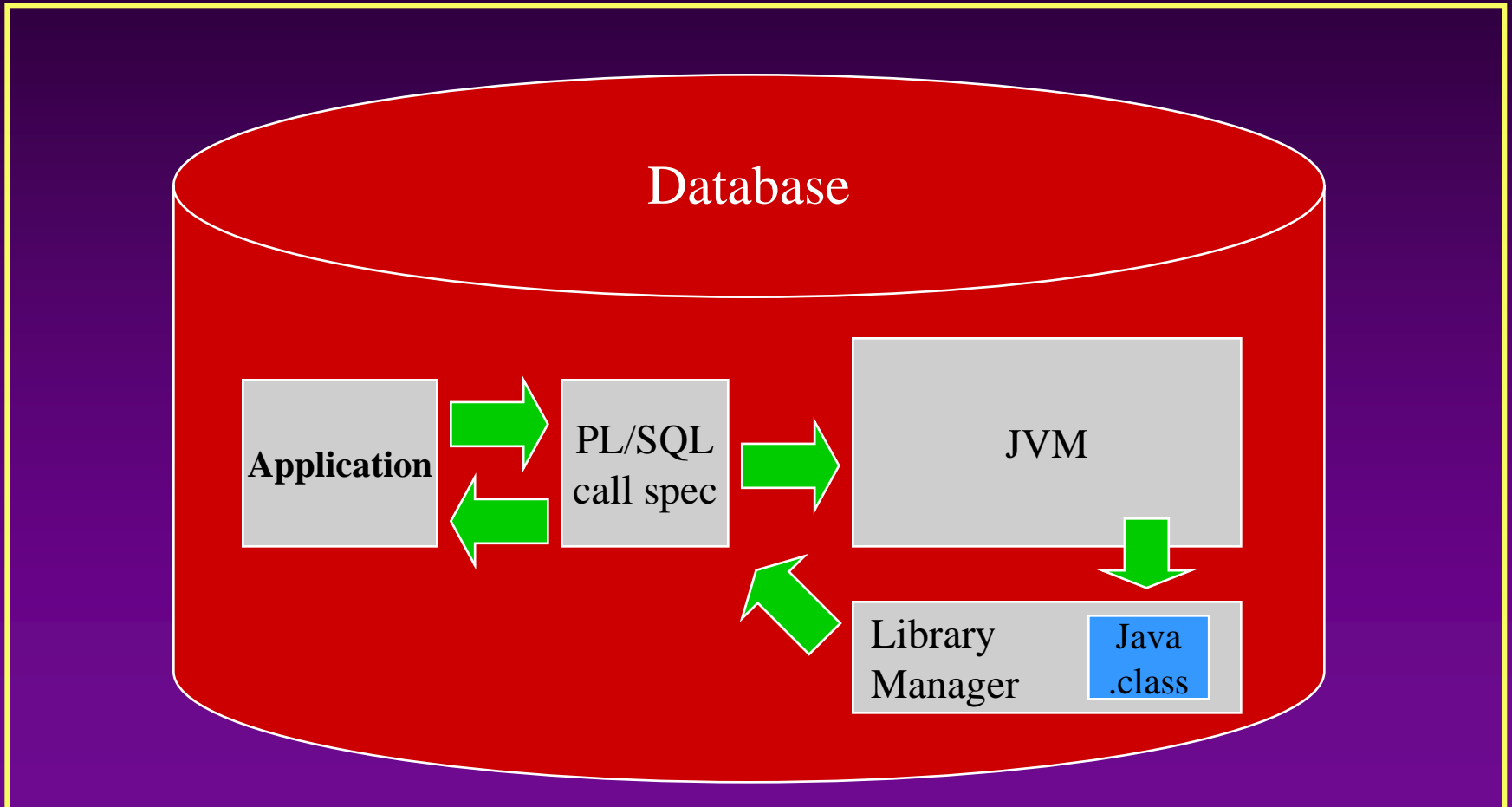
- ◆ Public Java class
- ◆ Public method
- ◆ Static method
 - Exception – instance method OK for SQL object type
- ◆ Mapped “signature” – maps to the call spec parameters and function return
 - Java types must correctly map to PL/SQL types



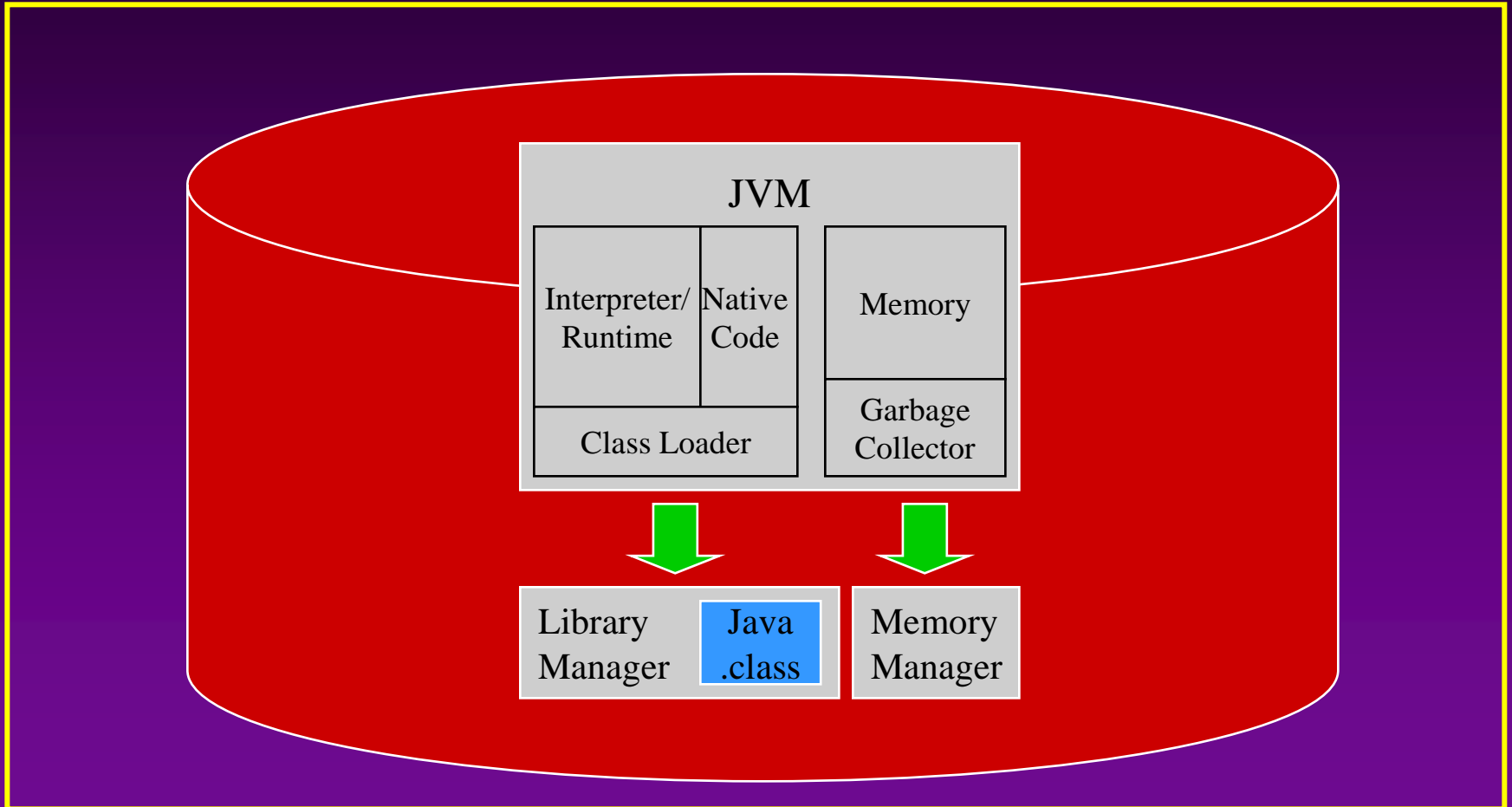
Java Stored Procedure Sample Code

```
package OSUtil;
import java.io.*;
import java.lang.Runtime;
public class OSCmd extends object {
    public static String osExec(String in_cmd) {
        String sret = "Return code ";
        Process p = null;
        int response = 0;
        try { p = Runtime.getRuntime().exec(in_cmd);
        }
        catch (IOException io) { return "IO process
failed"; }
        try { response = p.waitFor(); }
        ...
        return sret + response;
    }
}
```

Java Stored Procedure Execution Flow



Java Virtual Machine (JVM) Components



◆ JDeveloper

- Write, compile, test and debug code
- Deploy to database and create call spec
- Deployment wizard – create multiple profiles
- JDev 3.2.3 and 9.0.2

◆ SQL*Plus (can be done, but very difficult to use)

- Write source code
- Deploy to database using LoadJava utility
- Create PL/SQL call spec

◆ Third party products

- JBuilder
- Visual Cafe
- Sun ONE Studio (Forte)



LoadJava

- ◆ Command line method in `dbms_java` class
- ◆ Use to load Java classes and archive files into DB.
 - Source (.java) – Java source code
 - Class (.class) – Bytecode runtime files
 - Resource (.jar/.zip) – Java archive files
- ◆ Compile and resolve classes
- ◆ Set permissions
- ◆ Code:
 - `loadjava -u scott/tiger -resolve OSCmd.class`
- ◆ DropJava method – Delete classes from DB



Performance Comparison

- ◆ Java class procedures are generally slower than a compiled C external procedure.
- ◆ Why? Java procedures are interpreted bytecode
 - Bytecode is portable across supported platforms
 - Bytecode is interpreted into native code
 - On-the-fly interpretation adds overhead to execution
- ◆ Need speed? Convert Java classes to native code
 - Solution: JServer Accelerator
 - Compiles bytecode into platform independent C code
 - Compile/link C code, using C compiler for target platform
 - Native code modules 2-10X faster than Java bytecode



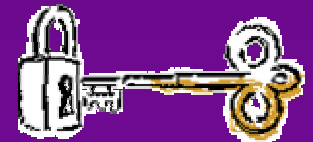
Java Security

◆ Roles (Rel. 8.1.5)

- JavaSysPriv – Maximum permissions
- JavaUserPriv – Fewer permissions

◆ Permissions (Rel. 8.1.6)

- Java 2 Security
- Provides a finer grain of security than roles
- Grant – allow access to specified resource
- Restrict – specify limit/exception from a general rule



Java Security - 2

◆ Invoker/Definer Privileges

- Apply to execution of class methods
- Invoker – Methods execute with privileges of their invoker (default)
- Definer – Methods execute with privileges of their definer

◆ Oracle8i implementation differs from Java 2

- O8i: Class by class J2: CLASSPATH
- O8i: PolicyTable J2: security policy file

◆ Java Security Manager

- Administers security policy for Java
- Runs concurrently with the JVM





Problem #1

Execute O/S Command

◆ Problem:

- Delete image files from server after data is read into the database.

◆ Solution attempted (using JDev 3.2.3):

- Pass command string to Java stored procedure
- Time to delete a file too slow (12 secs on NT server)
- Procedure stopped working on migration to Linux

◆ Final solution:

- Implemented external procedure
- Execution time per file: acceptable (milliseconds)

Problem #2

Write BLOB to File

◆ Problem:

- Store and retrieve binary objects in a library manager checkout application for BRIM™

◆ Solution developed - PL/SQL:

- Read BLOB from table, into a RAW variable
- Convert Raw data to Hex character (RawtoHex)
- Pass Hex data and file information to call spec

◆ Java stored procedure processing (JDev 3.2.3):

- Converts hex characters into binary data, one byte at a time
- Write-append binary data byte to output file
- Return success/fail code to call spec



Summary Comparison

◆ External Procedure

- Resides / executes outside of the database
- Runs in same transaction as calling application
- Performance – fast, native compiled code; much faster in Oracle8i

◆ Java Stored Procedure

- Resides / executes inside of the database
- Runs in same transaction as calling application
- Bytecode slower than C; is 2-10x faster when natively compiled

Summary Final Analysis

- ◆ Performance – External procedure
- ◆ Platform portability – Java
- ◆ Simpler implementation – Java
- ◆ C environment – External procedure
- ◆ C programmers – External procedure
- ◆ Java programmers – Java





Dulcian BRIM™

- ◆ To learn more about Dulcian's Business Rules Information Manager (BRIM™) product, see the Dulcian website: www.dulcian.com
 - Application development tool & runtime environment
 - Download documents about BRIM™
 - BRIM™ is FREE!
 - Interested? Write "BRIM" on your business card





Contact Information

Robert F. Edwards

Dulcian, Inc.

732-744-1116

redwards@dulcian.com

www.dulcian.com