Schulz IT Services
Gesellschaft für Datenbank- und Softwareentwicklung mbH

DOAG Konferenz 2017

Java-Programme in der Oracle-Datenbank?
Na klar!

Matthias Schulz

Unabhängiger Oracle Experte:

- Consulting
- Schulungen
- Workshops
- Blog:
  oracledeli.wordpress.com

Geschäftsführer:

Schulz IT Services GmbH

Tauberstraße 28
90449 Nürnberg
Telefon (0911) 3849090

www.schulz-it-services.de
schulz@schulz-it-services.de

Matthias Schulz

Entwicklung und Tuning:

- Datenbankdesign
- OLTP, OLAP, ETL
- SQL
- PL/SQL
- APEX
- Java in der Datenbank

Spezialthemen:

- Data Vault
- Exadata Database Machine
- Exasol Database
- Advanced Queueing
- Replikation
- objektorientiertes PL/SQL

Schulz IT Services
Gesellschaft für Datenbank- und Softwareentwicklung mbH

Stored Procedures

Stored Procedures
Vorteile

Stored Procedures bieten eine Reihe bekannter Vorteile:

+ bessere **Kapselung**

+ höhere **Sicherheit**

+ Trennung von „Data Logic“ und „Business Logic“

+ kürzere **Programmausführungszeit**

+ geringere **Netzwerklast**

+ …

Java Stored Procedures

Zusätzliche Vorteile

+ **Unabhängigkeit** vom Datenbankhersteller
+ **Wiederverwendbarkeit** von vorhandenem Java Know How
+ unzählige Java **Bibliotheken und Frameworks**
+ Besserer Zugriff auf das **Filesystem** (im Vergleich zu PL/SQL)
+ **Verbindungen** zu nicht-Oracle-Systemen (JDBC, ...)
+ Zugriff auf **LOB's** in entfernten Datenbanken
+ …

Standalone Programm vs Java Stored Procedure

Administration und Kosten bei bereits vorhandener Datenbank

	Java Server (Standalone)	Oracle DB (JServer)
Softwarelizenzen	 €	✓
Hardware	 €	✓
Backup & Recovery	 €	✓
Maintenance & Patches	 €	✓

 Zeitaufwand € Kosten ✓ vorhanden

Schulz IT Services

Gesellschaft für Datenbank- und Softwareentwicklung mbH

Java Stored Procedures

Anwender

Java Stored Procedures werden von den nationalen Behörden der USA und Großbritanniens zur Abwicklung der Einreisekontrollen verwendet.

Da ein Ausfall der Systeme jegliche Einreise zum Erliegen bringen würde, wurde eine Technologie gewählt die Sicherheit, Stabilität, hohe Performance und langfristige Produktverfügbarkeit gewährleistet:

Java Stored Procedures

Quelle: Oracle

Performance

Standalone Programm vs Java Stored Procedure

Sind Java Stored Procedure schneller als Java Programme
die außerhalb der Datenbank laufen?

Java Testprogramm:

- **Datenbankverbindung** aufbauen
- **Tabelle** „BASIC_LOB_TABLE“ falls vorhanden entfernen und erstellen
- **DML-Operationen**:
  - **Einfügen** von zwei Datensätzen
  - **Selektieren** der Datensätze und Abarbeiten des Cursors in einer Schleife:
    - **Lesen** der Clob- und Blob-Felder der Tabelle und ermitteln der **Größen**
    - **Truncate** auf das CLob- und das BLob-Objekt und ermitteln der **Größen**

Standalone Programm vs Java Stored Procedure

Das Java Testprogramm führt sowohl standalone als auch als Stored Procedure exakt den gleichen Sourcecode aus, bis auf diese Stelle:

```
/* Where is the code running: in the database or outside? */
if (System.getProperty("oracle.jserver.version") != null){
    /* Inside the database: use the default connection */
    conn = DriverManager.getConnection("jdbc:default:connection:");
}
else{
    /* Standalone: connect to the database */
    DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
    conn = DriverManager.getConnection(
                    "jdbc:oracle:thin:@mycompany.com:1521:MYDB",
                    "MY_USER", "MyPassword123");
}
```

Standalone Programm vs Java Stored Procedure

Laufzeitvergleich – nur **DML**-Operationen

Lauf	Standalone	JServer	schneller
1	80 ms	10 ms	8 mal
2	94 ms	8 ms	12 mal
3	93 ms	8 ms	12 mal
4	93 ms	8 ms	12 mal
5	93 ms	7 ms	13 mal
6	93 ms	8 ms	12 mal
7	94 ms	8 ms	12 mal
8	78 ms	9 ms	9 mal
9	94 ms	8 ms	12 mal
10	94 ms	8 ms	12 mal
	906 ms	**82 ms**	**11 mal**

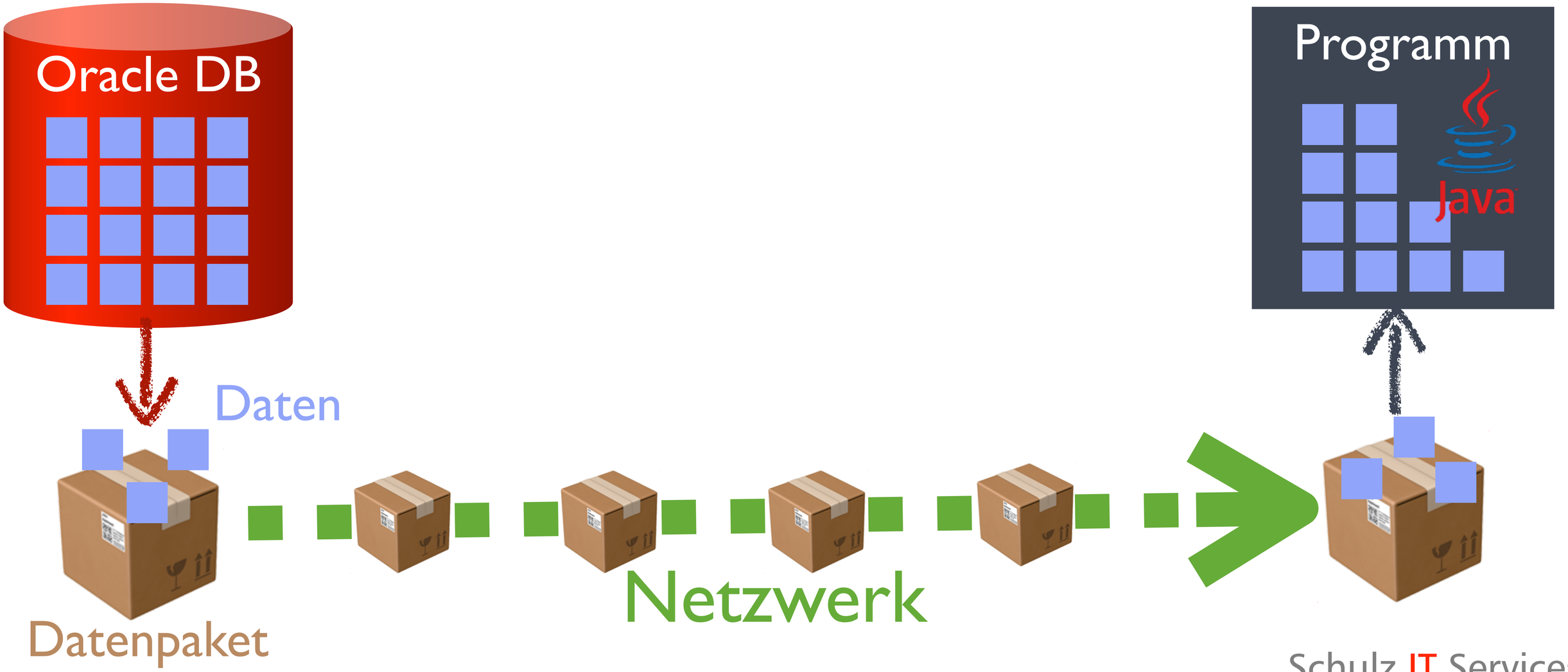
Standalone Programm vs Java Stored Procedure

Laufzeitvergleich – **komplett**

Lauf	Standalone	JServer	schneller
1	689 ms	45 ms	15 mal
2	702 ms	39 ms	18 mal
3	686 ms	39 ms	18 mal
4	686 ms	41 ms	17 mal
5	686 ms	37 ms	19 mal
6	702 ms	39 ms	18 mal
7	687 ms	39 ms	18 mal
8	670 ms	44 ms	15 mal
9	686 ms	39 ms	18 mal
10	687 ms	39 ms	18 mal
	6.881 ms	**401 ms**	**17 mal**

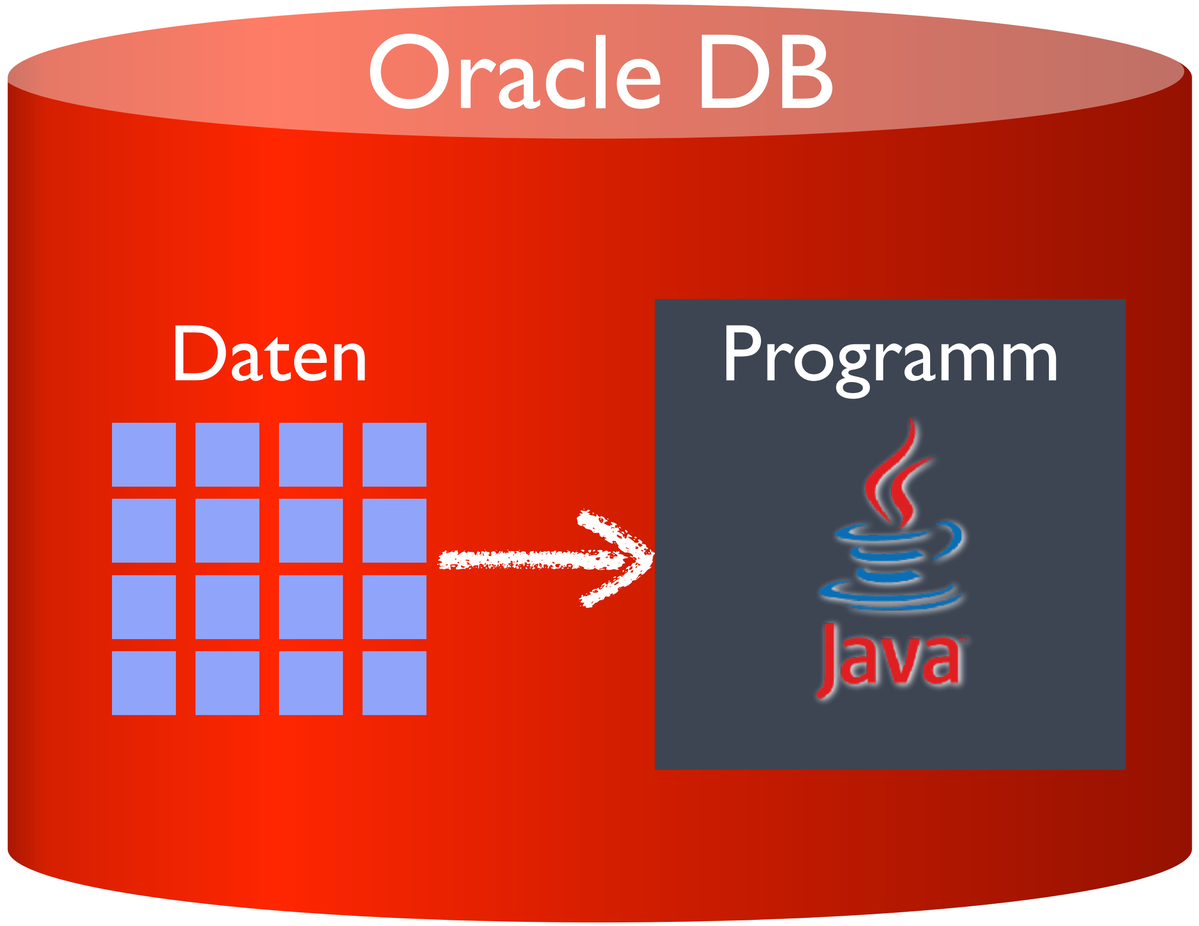
Standalone Java Programm

Laufzeitverluste durch Netzwerkübertragung



Java Stored Procedure

direkter Zugriff auf die Daten ohne Netzwerk



Ausfallsicherheit

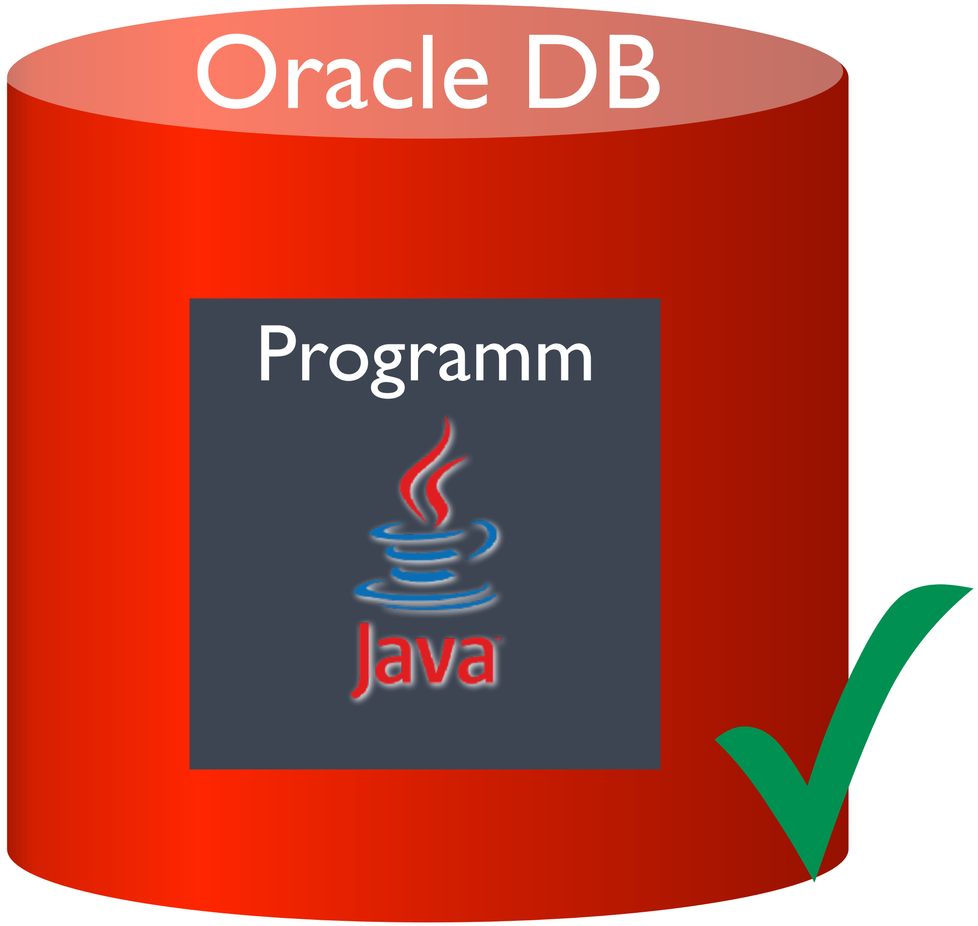
Standalone Java Programm

Zusätzliche Ausfallrisiken

Obwohl die Datenbank verfügbar ist, fällt das System aus, wenn das Netzwerk oder der Application Server ausfallen.

Java Stored Procedure

Ist die Datenbank verfügbar, ist auch der JServer verfügbar



Beispiel: Java Stored Procedures

Beispiel: Java Stored Procedures

1. Erstellen einer Java-Klasse:

```
CREATE OR REPLACE AND RESOLVE JAVA SOURCE NAMED „TestClass" AS
public class TestClass {
   public static void hello (String name) {
      System.out.println ("Hello " + name + "!");
   }
}
/
```

2. Erstellen eines passenden PL/SQL-Wrapper:

```
CREATE OR REPLACE PROCEDURE test_java_hello(name VARCHAR2)
AS LANGUAGE JAVA NAME 'TestClass.hello(java.lang.String)';
/
```

Beispiel: Java Stored Procedures

3. Aufruf der Java-Klasse durch den PL/SQL-Wrapper:

```
-- Konsolenausgabe aktivieren:
set serveroutput on size 1000000 ;

-- Java-Ausgaben auf die Konsole umleiten:
exec dbms_java.set_output(1000000) ;

BEGIN
    test_java_hello('DOAG');
END;
/
```

Ergebnis:

```
Hello DOAG!
```

Beispiel: Java Stored Procedures

4. Alles wieder entfernen:

Java-Source und Klasse verwerfen:

```
DROP JAVA SOURCE "TestClass";
```

PL/SQL-Wrapper verwerfen:

```
DROP PROCEDURE test_java_hello;
```

Deployment

CREATE JAVA SOURCE

Laden und kompilieren eines Java-Sources:

```
CREATE OR REPLACE AND RESOLVE JAVA SOURCE NAMED „TestClass"
AS
public class TestClass {
   public static void hello (String name) {
      System.out.println ("Hello " + name + "!");
   }
}
/
```

Es kann nahezu jede Java SE Klasse erstellt werden, Grafik- oder UI-Methoden sind jedoch nicht möglich.

Die Option "**RESOLVE**" kompiliert den Source und erstellt ein Java Class Object.

PL/SQL Wrapper

Erstellen eines PL/SQL-Wrappers:

```
CREATE OR REPLACE PROCEDURE test_java_hello(name VARCHAR2)
AS LANGUAGE JAVA NAME 'TestClass.hello(java.lang.String)';
/
```

Ein **PL/SQL Wrapper** ermöglicht den Zugriff auf Java Methoden.

Jede **"static" Java Methode** kann über eine passende PL/SQL Function oder Procedure aufgerufen werden.
Die Anzahl der **Parameter** muss gleich und der Datentyp kompatibel sein.

DROP JAVA SOURCE

Java-Source und Klasse verwerfen:

```
DROP JAVA SOURCE "TestClass";
```

loadjava Tool

Mit dem Tool "**loadjava**" können **einzelne Files** (Java Source, Java Class, Resource File, ...) und ganze **jar-Files** in die Oracle Datenbank geladen werden.

Wird ein **jar-File** geladen, so wird der komplette Inhalt als Objekte des aktuellen Users angelegt, wobei Package-Strukturen erhalten bleiben.

Das Tool ist ein Bestandteil des **Oracle Clients** (nicht jedoch des Instant Clients!)

loadjava Tool

Beispiel: Laden eines **jar-Files**

Beispiel:

```
D:\Oracle\product\12.1.0\client_1\bin\loadjava.bat
example.jar -thin -force -resolve -verbose
-user MY_USER/my_password@MYDB
```

Parameter	Bedeutung
`example.jar`	Name des jar-Files
`-thin`	JDBC Thin Client zum Verbindungsaufbau verwenden
`-force`	Bereits vorhandene Klassen überschreiben
`-resolve`	Sourcen sofort kompilieren
`-verbose`	Detailierte Ausgaben
`-user`	Datenbankverbindung: Username/Passwort@Datenbank

loadjava Tool

Beispiel: Laden eines **Resource-Files**

Beispiel:

```
D:\Oracle\product\12.1.0\client_1\bin\loadjava.bat
myconfig.properties –thin –force –verbose
–user MY_USER/my_password@MYDB
```

Parameter	Bedeutung
`myconfig.properties`	Name des Property-Files
`–thin`	JDBC Thin Client zum Verbindungsaufbau verwenden
`–force`	Bereits vorhandene Klassen überschreiben
`–verbose`	Detailierte Ausgaben
`–user`	Datenbankverbindung: Username/Passwort@Datenbank

DROP JAVA CLASS

Java-Klasse verwerfen mit „DROP JAVA CLASS“:

```
DROP JAVA CLASS "TestClass";
```

DROP JAVA CLASS

Alle Java-Klassen des aktuellen Users verwerfen:

```
BEGIN
    FOR rec IN (SELECT *
                  FROM USER_OBJECTS o
                 WHERE o.object_type LIKE 'JAVA%')
    LOOP
        EXECUTE IMMEDIATE 'DROP '
                || rec.object_type
                || '"' || rec.object_name || '"';
    END LOOP;
END;
/
```

JVM / JServer

JServer

Oracle Liefert seit Version 8.1.5 der Oracle Database
eine integrierte Java Virtual Machine (JVM) names **JServer** mit aus.

JServer

Ist eine JVM in meiner Datenbank installiert?:

```
SELECT * FROM all_registry_banners;
```

```
BANNER
Oracle Database Catalog Views Release 11.2.0.4.0 …
Oracle Database Packages and Types Release 11.2.0…
Oracle Workspace Manager Release 11.2.0.4.0 - Dev…
JServer JAVA Virtual Machine Release 11.2.0.4.0 …
…
```

Java Versionen

Aktuelle Version abfragen:

```
SELECT dbms_java.get_ojvm_property
           (propstring=>'java.version')
       AS java_version
  FROM DUAL;
```

```
JAVA_VERSION
1.6.0_43
```

Java Versionen

Oracle Datenbank	Java Version	
Version	standard	unterstützt
11g	JRE 1.5	-
11g, ab 11.2.0.4	JDK 6	-
12c	JDK 6	JDK 7
12c, ab 12.2.0.1	Java 8	Nashorn (JavaScript Engine)

Multithreading

Alle Java Stored Procedures laufen innerhalb einer Oracle **Datenbank Session**.

Einzelne Threads sind völlig voneinander **isoliert**. Ein Datenaustausch kann z.B. über Tabellen oder Messages (Oracle Advanced Queuing) erfolgen.

Multithreading-Programme sind zwar lauffähig, verwenden aber nur eine CPU.

Werden mehrere Threads benötigt, so kann dies durch **Oracle Scheduler Jobs** realisiert werden.

JDBC

Für den Zugriff auf die **„eigene“ Datenbank** existiert eine immer geöffnete JDBC-Verbindung:

```
Connection con =
DriverManager.getConnection("jdbc:default:connection:");
```

Auf **andere Datenbanken** kann wie gewohnt via JDBC zugegriffen werden.

Standalone oder JServer?

Um Festzustellen ob das Programm innerhalb einer Oracle Datenbank (JServer) läuft kann eine System-Property abgefragt werden:

```
if (System.getProperty("oracle.jserver.version") != null){
    /* program runs inside the Oracle database */
}
else{
    /* program runs as standalone */
}
```

Beispiel: Test Socket

Test Socket

Aufgabe

Testen ob eine Verbindung zu einem anderen Server hergestellt werden kann

Test Socket

Java Source

```
CREATE OR REPLACE AND RESOLVE JAVA SOURCE NAMED "SocketTest" AS
import java.net.Socket;
public final class SocketTest
{
    public static void connectTest(final String pHost, final int pPort)
    {
        Socket socket;
        System.out.println("testSourceSocketConnection: host: " + pHost
                           + ", port:" + pPort);
        try {
            socket = new Socket(pHost, pPort);

            if (socket.isConnected()) {
                System.out.println("socket.isConnected() = true\n");
            }
            else {
                System.out.println("socket.isConnected() = false !!!\n");
            }
            socket.close();
        }
        catch (Exception e) {e.printStackTrace();}
    }
}
/
```

Test Socket

PL/SQL Wrapper

```
CREATE OR REPLACE PROCEDURE
    SOCKET_TEST(p_host VARCHAR2, p_port NUMBER)
AS LANGUAGE JAVA
    NAME 'SocketTest.connectTest(java.lang.String, int)';
/
```

Test Socket

Test 1 – alles ok

```
SET SERVEROUTPUT ON ;
BEGIN
    DBMS_JAVA.set_output(1000000);
    SOCKET_TEST('myserver1.mycompany.com', 61616);
END;
/
```

Ergebnis:

```
testSourceSocketConnection: host: myserver1.mycompany.com,
port:61616
socket.isConnected() = true
```

Test Socket

Test 2 – Fehler

```
SET SERVEROUTPUT ON ;
BEGIN
    DBMS_JAVA.set_output(1000000);
    SOCKET_TEST('myserver2.mycompany.com', 1234);
END;
/
```

Ergebnis:

```
testSourceSocketConnection: host: myserver2.mycompany.com, port:123
java.security.AccessControlException: the Permission (java.net.SocketPermission
myserver.mycompany.com resolve) has not been granted to DISTRIBUTOR_PROCESSING.
The PL/SQL to grant this is
dbms_java.grant_permission( 'DISTRIBUTOR_PROCESSING',
'SYS:java.net.SocketPermission', 'myserver.mycompany.com', 'resolve' )
  at
java.security.AccessControlContext.checkPermission(AccessControlContext.java)
  at java.security.AccessController.checkPermission(AccessController.java)
...
```

Test Socket

Test 2 – Fehlermeldung im Detail

Ergebnis:

```
testSourceSocketConnection: host: myserver2.mycompany.com,
port:123
java.security.AccessControlException: the Permission
(java.net.SocketPermission myserver.mycompany.com resolve)
has not been granted to DISTRIBUTOR_PROCESSING.
The PL/SQL to grant this is
dbms_java.grant_permission( 'DISTRIBUTOR_PROCESSING',
'SYS:java.net.SocketPermission', 'myserver.mycompany.com',
'resolve' )
 at java.security.AccessControlContext...
```

Für den Zugriff auf den Server fehlt eine Berechtigung.

Der Befehl zur Erteilung steht bereits fertig in der Fehlermeldung!

Java oder PL/SQL?

Java oder PL/SQL?

Technik

Thema	PL/SQL	Java	Beispiel
SQL	+++	+	
Number Crunching	+	+++	
Portabilität - Oracle DB	+++	+	
- andere DB	–	+++	DB2, Exasolution, …
Zugriff auf andere Systeme	+	+++	DB-Link, JDBC, REST,…
Mitgelieferte Biliotheken	+	++	
Drittanbieter Frameworks	+	+++	

Java oder PL/SQL?

Aufgabenbereiche

Thema	PL/SQL	Java	Vorteil
SQL	●		keine zusätzliche JDBC-Schicht
SQL-Datentypen	●		keine Umwandlungen
Berechnungen		●	schneller
Moderne Sprachelemente		●	Vererbung, Closures, …
Filehandling		●	mehr Möglichkeiten/Frameworks
Mailversand		●	mehr Möglichkeiten/Frameworks
Zugriff auf Web-Server		●	mehr Möglichkeiten/Frameworks
Zugriff auf andere Systeme		●	JDBC, REST,…

Java oder PL/SQL?

Skills

Thema	PL/SQL	Java
Nutzung vorhandener Kenntnisse	?	?
Verfügbare Entwickler im Markt	+	+++
Code-Wiederverwendung	–	+++

Anhang

Dokumentation

Dokumentation

Database Java Developer's Guide

https://docs.oracle.com/database/122/JJDEV/toc.htm

The loadjava Tool

https://docs.oracle.com/database/122/JJDEV/loadjava-tool.htm#JJDEV10060

DBMS_JAVA Package

https://docs.oracle.com/database/122/JJDEV/DBMS-JAVA-package.htm#JJDEV13000

Java Berechtigungen

Java Berechtigungen

Java Berechtigungen werden über Methoden des System-Packages **DBMS_JAVA** verwaltet.

Java Berechtigungen

Rechte erteilen – Beispiel **Filesystemzugriff:**

```
BEGIN
    dbms_java.grant_permission('MY_USER',
'SYS:java.io.FilePermission', '/my_directory/*', 'read');

    dbms_java.grant_permission('MY_USER',
'SYS:java.io.FilePermission', '/my_directory/*', 'write');

    dbms_java.grant_permission('MY_USER',
'SYS:java.io.FilePermission', '/my_directory/*', 'execute');
END;
/
```

Java Berechtigungen

Rechte erteilen – Beispiel **Runtime Permission:**

```
BEGIN
    dbms_java.grant_permission('MY_USER',
'SYS:java.lang.RuntimePermission', 'getClassLoader', '');
END;
/
```

Java Berechtigungen

Rechte erteilen – Beispiel **Serverzugriff:**

```
BEGIN
    dbms_java.grant_permission('MY_USER',
                               'SYS:java.net.SocketPermission',
                               'myserver.mycompany.com',
                               'connect,resolve');
END;
/
```

Es können mehrere Rechte in einem Aufruf von „grant_permission“ erteilt werden (hier „connect“ und „resolve“).

Java Berechtigungen

Aktuelle Java Berechtigungen abfragen:

```
SELECT *
  FROM USER_JAVA_POLICY;
```

KIND	GRANTEE_NAME	TYPE_SCHEMA	TYPE_NAME
GRANT	MY_USER	SYS	java.net.SocketPermission

NAME	ACTION	ENABLED	SEQ
myserver.mycompany.com	connect,resolve	ENABLED	258

Java Berechtigungen

Rechte deaktivieren – Beispiel **Serverzugriff:**

```
BEGIN
    dbms_java.revoke_permission('MY_USER',
                                'SYS:java.net.SocketPermission',
                                'myserver.mycompany.com',
                                'resolve');
END;
/
```

Hier wird das Recht "resolve" **deaktiviert**.

Java Berechtigungen

Rechte aktivieren – Beispiel **Serverzugriff:**

```
BEGIN
     dbms_java.enable_permission(278);
END;
/
```

Um das Recht "resolve" (wieder) zu aktivieren, wird die Sequence Nummer des Rechts benötigt.
Diese findest sich in der **Spalte "SEQ"** des Views "**USER_JAVA_POLICY**".

Java Berechtigungen

Rechte deaktivieren (via Sequence Nummer) – Beispiel **Serverzugriff:**

```
BEGIN
    dbms_java.disable_permission(278);
END;
/
```

Ein Recht "resolve" kann auch mittels der Sequence Nummer deaktiviert werden.
Diese findest sich in der **Spalte "SEQ"** des Views "**USER_JAVA_POLICY**".

Java Berechtigungen

Rechte löschen – Beispiel **Serverzugriff:**

```
BEGIN
     dbms_java.disable_permission(278);
     dbms_java.delete_permission(278);
END;
/
```

Um ein Recht zu löschen muss es zunächst deaktiviert werden, danach kann es gelöscht werden. Dies geschieht wieder über die Sequence Nummer (**Spalte "SEQ"**, View "**USER_JAVA_POLICY**").

Objekt- und System-Views

Java Objekte

```
SELECT dbms_java.longname(o.object_name),
       o.*
  FROM USER_OBJECTS o
 WHERE o.object_type LIKE 'JAVA%'
 ORDER BY dbms_java.longname(o.object_name);
```

Intern werden die Java-Klassennamen in verkürzter Form abgelegt.

Mit der Funktion **dbms_java.longname** kann der vollständige Name des Packages und der Klasse abgerufen werden.

Java Objekte eines Packages

```
SELECT dbms_java.longname(o.object_name),
       o.*
  FROM USER_OBJECTS o
 WHERE o.object_type LIKE 'JAVA%'
   AND dbms_java.longname(o.object_name)LIKE 'com/mycompany/myproject/%'
 ORDER BY dbms_java.longname(o.object_name);
```

Java Objekte mit Fehlern

```
SELECT dbms_java.longname(e.name), e.*
  FROM USER_ERRORS e
 WHERE e.TYPE LIKE 'JAVA%'
 ORDER BY dbms_java.longname(e.name);
```

Java System Views I

USER_JAVA_ARGUMENTS	argument information of stored java class owned by the user
USER_JAVA_CLASSES	class level information of stored java class owned by the user
USER_JAVA_COMPILER_OPTIONS	native compiler options provided by the user
USER_JAVA_DERIVATIONS	this view maps java source objects and their derived java class objects and java resource objects for the java class owned by user
USER_JAVA_FIELDS	field information of stored java class owned by the user
USER_JAVA_IMPLEMENTS	interfaces implemented by the stored java class owned by user
USER_JAVA_INNERS	list of inner classes refered by the stored java class owned by user

Java System Views 2

USER_JAVA_ARGUMENTS	argument information of stored java class owned by the user
USER_JAVA_CLASSES	class level information of stored java class owned by the user
USER_JAVA_COMPILER_OPTIONS	native compiler options provided by the user
USER_JAVA_DERIVATIONS	this view maps java source objects and their derived java class objects and java resource objects for the java class owned by user
USER_JAVA_FIELDS	field information of stored java class owned by the user
USER_JAVA_IMPLEMENTS	interfaces implemented by the stored java class owned by user
USER_JAVA_INNERS	list of inner classes refered by the stored java class owned by user

Datentypen

Datentyp-Mapping

Bereich	Java Types	Oracle Types
Boolean	boolean	NUMBER
Ganze Zahlen	byte short int long java.math.BigDecimal	NUMBER
Kommazahlen	float double	NUMBER

Datentyp-Mapping

Bereich	Java Types	Oracle Types
Text	java.lang.String	CHAR VARCHAR2 LONG
Bytes	byte[ ]	RAW LONGRAW
Datum	java.sql.Date java.sql.Time	DATE
Timestamp	java.sql.Timestamp	TIMESTAMP

Datentyp-Mapping

Bereich	Java Types	Oracle Types
Array	java.sql.Array	ARRAY
Clob	java.sql.Clob	CLOB
Blob	java.sql.Blob	BLOB
Ref	java.sql.Ref	REF
Struct	java.sql.Struct	STRUCT

Fragen?

Urheberrecht

Haftungsausschluss

Die Schulz IT Services GmbH übernimmt keine Haftung für Vollständigkeit, Aktualität und inhaltliche Richtigkeit der zur Verfügung gestellten Informationen. Ebenso wird keine Haftung für Fehler redaktioneller und technischer Art übernommen.

Haftungsansprüche aus materiellen und immateriellen Schäden, welche sich durch die Nutzung fehlerhafter oder unvollständiger Informationen oder durch die Nutzung der angebotenen Informationen ergeben könnten, sind grundsätzlich ausgeschlossen, es sei denn, die Schulz IT Services GmbH trifft Vorsatz oder grob fahrlässiges Verschulden.

Sollten Formulierungen im Haftungsausschluss oder Teile davon mit der geltenden Rechtslage nicht mehr oder nur noch partiell übereinstimmen, so bleibt hiervon die Gültigkeit der übrigen Teile dieses Textes mit seinem Inhalt unberührt.

Danke!