

Preface

Introduction

Part I: Java in the Database

1 Stored Procedures as Database Programming Model

1.1 Rationale for Stored Procedures

1.1.1 Simplifying Database Programming

1.1.2 Centrally Managed Data Logic

1.1.3 Performance: Run JDBC Applications Faster in the Database

1.1.4 Encapsulation

1.1.5 Security: Advanced Data Access Control

1.1.6 Resource Optimization

1.1.7 Low-Cost Deployment

1.1.8 Fully Utilize Database Capabilities

1.2 Obstacles to the Adoption of Stored Procedures

1.2.1 Lack of Portability across RDBMS Vendors

1.2.2 Scalability

1.2.3 Maintenance and Resilience to Schema Change

1.2.4 Hard to Debug

1.2.5 Weak Support for Complex Types

1.3 Languages for Stored Procedures

1.3.1 Proprietary Languages

1.3.2 Java for Stored Procedures

1.3.3 .NET Languages

1.4 PL/SQL or Java

1.4.1 PL/SQL and Java!

2 OracleJVM: Under the Hood

2.1 Design Goals and Architecture

2.1.1 Tight Integration with the RDBMS

2.1.2 J2SE Compatibility

2.1.3 How Is Java Stored in the Database?

2.1.4 Class Sharing

2.1.5 Interapplication Isolation (JSR 121)

2.1.6 Contrasting OracleJVM with the JDK VM

2.1.7 Resource Control

2.1.8 SQL Data Access from Java in the Database

2.1.9 DBMS\_JAVA: The All-Purpose Tool for Administering OracleJVM

2.2 Java Memory Management

2.2.1 Key Memory Structures of the Oracle Database

2.2.2 Java Memory Allocation Techniques

2.2.3 Garbage Collection Techniques

2.2.4 Java Memory Areas

2.2.5 Shared Servers versus Dedicated Processes

2.2.6 The Javapool

2.2.7 Top-Level Calls and Recursive Calls

2.2.8 State Preservation across Calls and End-of-Call Migration

2.2.9 End-of-Call, VM Termination, and Session Termination

2.3 Security in OracleJVM

2.3.1 User Authentication

2.3.2 Database-Schema Security

2.3.3 Resolver Specification and Class-Resolution Security

2.3.4 Login-User and Effective-User Security

2.3.5 Java 2 Security in OracleJVM

2.3.6 Java 2 Security in OracleJVM

- 2.3.7 OracleJVM Security Best Practices
- 2.3.8 JNI Calls
- 2.4 Java VM Life Cycle
  - 2.4.1 OracleJVM Install, Uninstall, and Reinstall
  - 2.4.2 Java VM Initialization and Termination
- 2.5 Java Execution in the Database
  - 2.5.1 The OracleJVM Interpreter
- 2.6 The Native Java Compiler (NCOMP)
  - 2.6.1 What Is NCOMP?
  - 2.6.2 Requirements and Design Choices
  - 2.6.3 The NCOMP Process
  - 2.6.4 The NCOMP Command
  - 2.6.5 The STATUSNC Command
  - 2.6.6 NCOMP Configuration and Planning
  - 2.6.7 NCOMP Performance Tips, Improper Use, and Troubleshooting
- 3 Developing and Running Java in the Database
  - 3.1 Developing Java in the Database
    - 3.1.1 Turning JDBC Applications into Java Stored Procedures
    - 3.1.2 Creating or Loading Java in the Database
    - 3.1.3 Removing Java Sources, Classes, and Resources from the Database
    - 3.1.4 Setting/Querying Environment Variable and System Properties
    - 3.1.5 The Java Compiler within the Database
  - 3.2 Turning Java in the Database into Stored Procedures
    - 3.2.1 Call Spec Types
  - 3.3 Mapping SQL and PL/SQL Types to/from Java Types
    - 3.3.1 Mapping Matrix
    - 3.3.2 Code Segments for Mapping
  - 3.4 Invoking Java in the Database
    - 3.4.1 Setup
    - 3.4.2 Invoking Java in the Database Using OJVMJAVA
    - 3.4.3 Invoking Java in the Database through the PL/SQL Wrapper
    - 3.4.4 Invoking Java in the Database through Client-side Stub
    - 3.4.5 Errors and Exceptions Handling
  - 3.5 Managing Java in the Database
    - 3.5.1 Java Audit
    - 3.5.2 Oracle Enterprise Manager (Database Control) Support for Java in the Database
- 4 Pragmatic Applications Using Java in the Database
  - 4.1 CNXO: Secure Credit Card Processing with Oracle and JSSE
  - 4.2 Using J2EE and Java in the Database Together
    - 4.2.1 Auto-generating Primary Keys for BMP Entity Beans
    - 4.2.2 Calling-out EJB from OracleJVM
    - 4.2.3 HTTP Call-Out: The Poor Man's Cache Invalidation
    - 4.2.4 JMS over Streams/AQ in the Database
  - 4.3 JDBC Call-Out to Non-Oracle Databases
    - 4.3.1 Description and Rationales
    - 4.3.2 How Does It Work?
  - 4.4 SAP Java Connector: Accessing the SAP System from the Oracle Database
  - 4.5 Excel-like Expression Parser in the Database
    - 4.5.1 Rationales for Custom Parsers in the Database
    - 4.5.2 What Is the Mini-Parser?
    - 4.5.3 Implementing the Mini-Parser
- 5 Database Scripting Using Non-Java Languages
  - 5.1 Why Contemplate Non-Java Languages for the Database?

- 5.1.1 Common Language Runtime RDBMS
- 5.1.2 Scripting Languages Support in RDBMS
- 5.2 Database Scripting with OracleJVM: Just for Fun!
- 5.2.1 Proof of Concept #1: Running TCL (JAQL) Scripts in the Database
- 5.2.2 Proof of Concept #2: Running Jython (Python) in the Database
- 5.2.3 Proof of Concept #3: Running Kawa (Scheme) in the Database
- 5.2.4 Proof of Concept #4: Running Groovy in the Database

Part II: Java Persistence and Java SQL Data Access  
- Database Programming with Oracle JDBC

- 6 Introducing the JDBC Technology and Oracle's Implementation
  - 6.1 JDBC Primer
    - 6.1.1 First Steps in JDBC
    - 6.1.2 JDBC within J2SE and J2EE Environments
  - 6.2 Overview of JDBC Specifications
    - 6.2.1 Overview of JDBC 1.22 Specification (Where Things Started!)
    - 6.2.2 Overview of JDBC 2.0 Specification (A Major Spec!)
    - 6.2.3 Overview of JDBC 3.0 Specification
    - 6.2.4 Overview of Upcoming JDBC 4.0 Specification
    - 6.2.5 JDBC Standards Support in the Oracle JDBC Drivers
  - 6.3 Architecture and Packaging of Oracle JDBC Drivers
    - 6.3.1 Rearchitected Oracle JDBC Drivers
    - 6.3.2 Packaging of Oracle JDBC Drivers
    - 6.3.3 Features Differences Between Driver Types
    - 6.3.4 JDBC Drivers and Database Interoperability

- 7 URL, DataSource, Connection, and Statements
  - 7.1 JDBC URL
  - 7.2 DataSources
    - 7.2.1 The OracleDataSource
    - 7.2.2 DataSources and JNDI
  - 7.3 Connections and Connection Services
    - 7.3.1 JDBC Connections and Oracle Extensions
    - 7.3.2 Connection Caching: Implicit Connection Cache
    - 7.3.3 The Connection Cache Manager
    - 7.3.4 RAC Events and Fast Application Notification
    - 7.3.5 High Availability: Fast Connection Failover
    - 7.3.6 Scalability: Connection Load Balancing
    - 7.3.7 JDBC Support for Transparent Application Fail-over
    - 7.3.8 Proxy Authentication
    - 7.3.9 Connection Wrapping
    - 7.3.10 JDBC Connections in Grid Environment
  - 7.4 JDBC Statements and Oracle Extensions
    - 7.4.1 JDBC Statement Types
    - 7.4.2 Statement
    - 7.4.3 PreparedStatement
    - 7.4.4 CallableStatement (Calling Stored Procedures)
    - 7.4.5 Retrieval of Auto-Generated Keys and DML with Returning
    - 7.4.6 Statement Caching
    - 7.4.7 DML Batching

- 8 SQL Data Access and Manipulation
  - 8.1 Key Metadta in JDBC
    - 8.1.1 DatabaseMetaData: OracleDatabaseMetaData
    - 8.1.2 ResultSetMetaData: OracleResultSetMetaData

- 8.1.3 ParameterMetaData
- 8.1.4 StructMetaData
- 8.2 Manipulating Oracle Data Types with JDBC
  - 8.2.1 Manipulating SQL Null Data
  - 8.2.2 Manipulating Character Data Types
  - 8.2.3 Oracle JDBC Support for Number Data Types
  - 8.2.4 JDBC Support for Long and Raw Data Types
  - 8.2.5 JDBC Support for SQL Datetime Data Types
  - 8.2.6 JDBC Support for LOB Datatypes
  - 8.2.7 JDBC Support for ROWID
  - 8.2.8 JDBC Support for OPAQUE Type
  - 8.2.9 JDBC Support for XMLType
  - 8.2.10 JDBC Support for SQL Object Types and References Types
  - 8.2.11 JDBC Support for User-Defined Collections
  - 8.2.12 JDBC Support for Spatial Types
  - 8.2.13 Unsupported Types
- 8.3 Result Set Support in Oracle JDBC
  - 8.3.1 The Result Set API in a Nutshell
  - 8.3.2 The Oracle Result Set Interface
  - 8.3.3 Oracle JDBC Support for Scrollable Result Sets
  - 8.3.4 Oracle JDBC Support for Updatable Result Sets
  - 8.3.5 Prefetching and Auto Refresh
  - 8.3.6 Changes Detection and Visibility
- 8.4 RowSet
  - 8.4.1 Introducing the RowSet API
  - 8.4.2 JDBCRowSet and OracleJDBCRowSet
  - 8.4.3 CachedRowSet and OracleCachedRowSet
  - 8.4.4 WebRowSet and OracleWebRowSet
  - 8.4.5 FilteredRowSet and OracleFilteredRowSet
  - 8.4.6 JoinRowSet and OracleJoinRowSet
- 8.5 Conclusion

## 9 JDBC Quality of Services and Best Practices

- 9.1 Transaction Services
  - 9.1.1 Transactions
  - 9.1.2 AutoCommit
  - 9.1.3 Transaction Isolation Levels
  - 9.1.4 Transaction SavePoint Support
  - 9.1.5 Global/Distributed Transaction
  - 9.1.6 Connection Sharing between Local and Global Transactions
- 9.2 Security Services
  - 9.2.1 Oracle JDBC Support for SSL
- 9.3 Tips and Best Practices
  - 9.3.1 End-to-End Tracing
  - 9.3.2 Common Errors
  - 9.3.3 Optimizing Result Set Retrieval
  - 9.3.4 Logging Service
- 9.4 Conclusion

## Part III: Oracle Database Programming with SQLJ

### 10 Introducing the SQLJ Technology and Oracle's Implementation

- 10.1 Overview
  - 10.1.1 What Is SQLJ?
  - 10.1.2 Why SQLJ?
  - 10.1.3 The Oracle SQLJ Translator

- 10.1.4 The Oracle SQLJ Runtime
- 10.1.5 Environment Setup
- 10.1.6 SQLJ Primer
- 10.2 SQLJ in the Database
- 11 The SQLJ Language and Oracle Extensions
  - 11.1 Declaration Statements
    - 11.1.1 Import Statements
    - 11.1.2 Connection Contexts
    - 11.1.3 Execution Contexts
    - 11.1.4 Iterators
    - 11.1.5 IMPLEMENTS Clause in Context Declarations
    - 11.1.6 WITH Clause in Context Declarations
  - 11.2 Executable Statements
    - 11.2.1 Statement Clauses
    - 11.2.2 Assignment Clauses
    - 11.2.3 Dynamic SQL
  - 11.3 Expressions in SQLJ
    - 11.3.1 Context and Result Expressions
    - 11.3.2 Expressions Evaluation
  - 11.4 Interoperability: Using SQLJ and JDBC Together
    - 11.4.1 JDBC to SQLJ Interoperability
    - 11.4.2 SQLJ to JDBC Interoperability
  - 11.5 Conclusion
- 12 SQL Data Access and Best Practices
  - 12.1 Manipulating Oracle SQL and PL/SQL Data Types with SQLJ
    - 12.1.1 Oracle SQLJ Type-Mapping Summary
    - 12.1.2 Column Definitions
    - 12.1.3 Manipulating SQL Null Data with SQLJ
    - 12.1.4 Manipulating Character Data Types with SQLJ
    - 12.1.5 Oracle SQLJ Support for Number Data Types
    - 12.1.6 SQLJ Streams, LONG, and RAW Data Types
    - 12.1.7 SQLJ Support for SQL Datetime Data Types
    - 12.1.8 SQLJ Support for SQL LOB Data Types
    - 12.1.9 SQLJ Support for Oracle SQL ROWID
    - 12.1.10 SQLJ Support for OPAQUE Types
    - 12.1.11 SQLJ Support for SQL Object Types and SQL References Types
    - 12.1.12 Serialized Java Objects
    - 12.1.13 SQLJ Support for User-Defined SQL Collections
    - 12.1.14 PL/SQL Associative Array
    - 12.1.15 Unsupported Types
  - 12.2 SQLJ Best Practices
    - 12.2.1 Row Prefetch
    - 12.2.2 Statement Caching
    - 12.2.3 Update Batching
  - 12.3 Conclusion
- Part IV: Oracle Database Programming with JPublisher
- 13 Abridged Oracle JPublisher
  - 13.1 Why JPublisher?
  - 13.2 Overview
    - 13.2.1 Environment Requirements
    - 13.2.2 JPublisher Options
  - 13.3 JPublisher In Action
    - 13.3.1 User-Defined SQL Object Types

- 13.3.2 SQL Object Reference Types (REF types)
- 13.3.3 REF Cursor Types and Subclassing
- 13.3.4 User-Defined SQL Collection Types
- 13.3.5 User-Defined OPAQUE Types
- 13.3.6 XMLType
- 13.3.7 PL/SQL Conversion Functions
- 13.3.8 PL/SQL RECORD Types
- 13.3.9 PL/SQL Table or Scalar Index-by-Table
- 13.3.10 Oracle Streams AQ
- 13.4 Conclusion

## Part V: Programming the Oracle Database with Web Services

### 14 Web Services and SOA for DBA, Data Architects, and Others

- 14.1 Web Services 101
  - 14.1.1 Core Web Services Technologies
- 14.2 Service-Oriented Architecture (SOA): The Bigger Picture
- 14.3 Conclusion

### 15 Database as Web Services Provider Service

- 15.1 Rationales for Database as Web Services Provider
- 15.2 How Does Database as Web Services Provider Work?
  - 15.2.1 Implementation and Packaging
  - 15.2.2 How Does Oracle Database as Web Services Provider Work?
  - 15.2.3 Web Services and SOA Features in Oracle Application Server 10.1.3
- 15.3 Turning Oracle Database Functionality into Web Services
  - 15.3.1 Type Conversions and Result Set Representation
  - 15.3.2 Setting up the Oracle AS OC4J for Database as Web Services Provider
  - 15.3.3 Assembling PL/SQL Web Services Using JDeveloper Wizard
- 15.4 Assembling Database Web Services Using the Command-Line Tool
  - 15.4.1 Assembling PL/SQL Web Services Using Web Services Assembler
  - 15.4.2 Assembling Java in the Database as a Web Service
  - 15.4.3 Assembling SQL Queries or SQL DML Statements as Web Services
  - 15.4.4 Assembling Oracle Streams AQ as Web Services
- 15.5 Data Type Restrictions
- 15.6 Conclusion

### 16 Database as Web Services Consumer

- 16.1 Rationales for Database as Web Services Consumer
- 16.2 How Database as Web Services Consumer Works
  - 16.2.1 The Software Pieces
  - 16.2.2 The Required Steps
- 16.3 Turning Your Oracle Database 10g into a Web Service Consumer
  - 16.3.1 Ensure That Java Is Installed in the Database
  - 16.3.2 Installing JPublisher on the Client Machine
  - 16.3.3 Installing the Web Services Call-Out Utility in Your Database
- 16.4 Database Web Services Call-Out Samples
  - 16.4.1 Calling Out Google Search Web Service
  - 16.4.2 Calling Out the Phone Verifier Web Service
- 16.5 Conclusion

## Part VI: Putting Everything Together

### 17 360-Degree Programming the Oracle Database

- 17.1 TECSIS Systems: Custom Enterprise Integration Framework
  - 17.1.1 About the Company

- 17.1.2 About the Application
- 17.1.3 Our Business and Technical Requirements
- 17.1.4 The Architecture of the Integration Framework
- 17.1.5 The Complete Picture
- 17.1.6 Conclusion
- 17.2 Oracle interMedia
  - 17.2.1 What Is Oracle interMedia?
  - 17.2.2 How Does It Work?
  - 17.2.3 Rationales for Storing Media Data in the Database
  - 17.2.4 interMedia Powered by the Oracle Database Extensibility Framework
  - 17.2.5 interMedia Powered by Java in the Database
  - 17.2.6 Developing Feature-Rich Multimedia Applications Using interMedia
- 17.3 British Columbia: Online Corporate Registration
  - 17.3.1 Corporate Online: Background
  - 17.3.2 How It Works
  - 17.3.3 Architecture: Requirements and Design
  - 17.3.4 Messaging across Tiers
  - 17.3.5 Future Work
  - 17.3.6 Conclusion
- 17.4 Information Retrieval Using Oracle Text
  - 17.4.1 What Is Oracle Text?
  - 17.4.2 Why Java in the Database?
  - 17.4.3 Technical Features
  - 17.4.4 Benefits of an Integrated Search Capability
  - 17.4.5 Yapa
  - 17.4.6 Conclusion
- 17.5 Database-Driven Content Management System (DBPrism CMS)
  - 17.5.1 DBPRISM CMS: Key Features and Benefits
  - 17.5.2 The Architecture of DBPrism CMS
  - 17.5.3 DBPrism CMS Internals
  - 17.5.4 Extended Capabilities
  - 17.5.5 Text Searching
  - 17.5.6 Installing DBPrism CMS
  - 17.5.7 Future Work
- 17.6 Conclusion