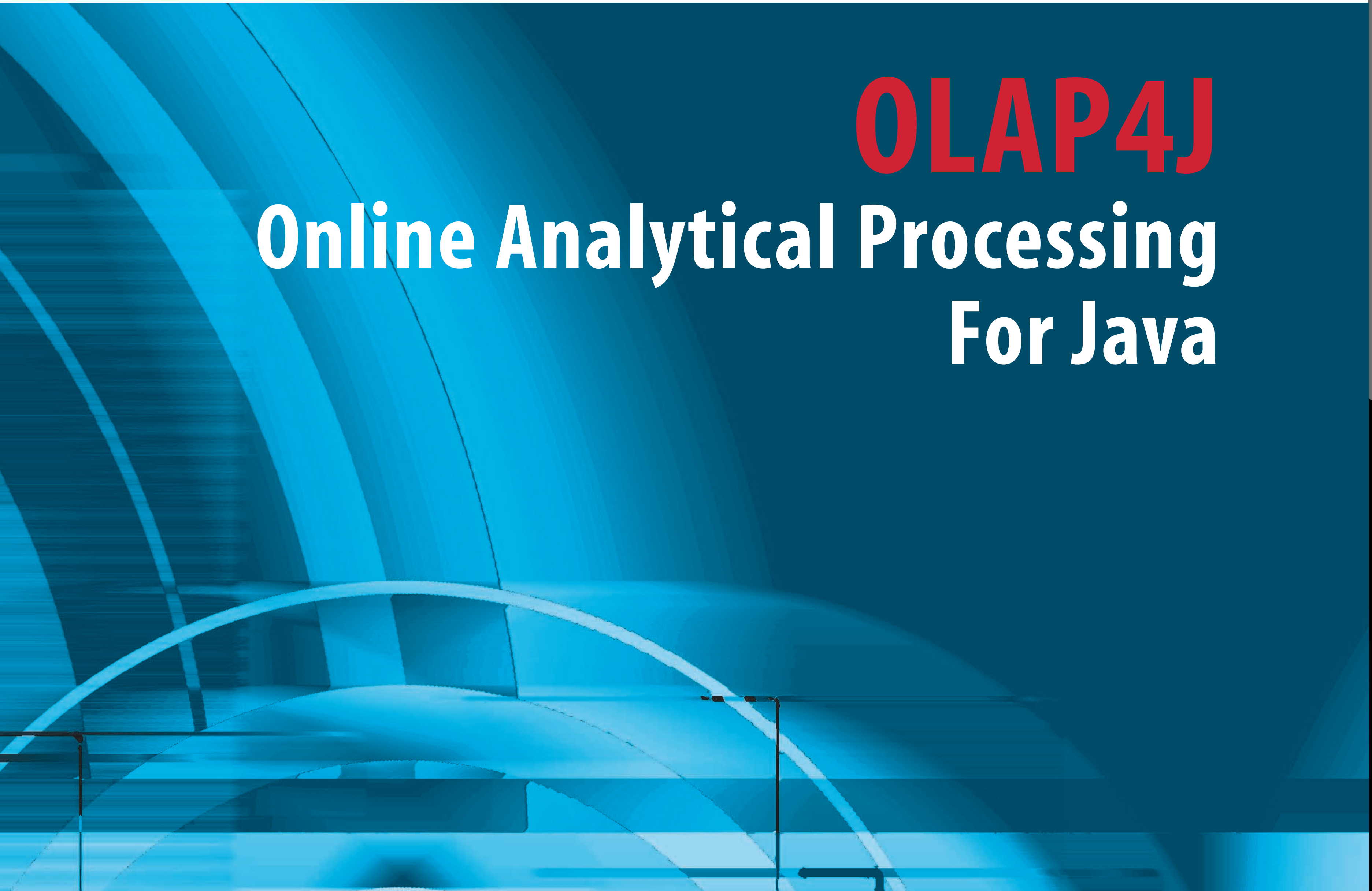
ALM ISSUE

Dr. Dobb's Journal

JULY 2011

Next



OLAP4J

Online Analytical Processing For Java

ALSO INSIDE

UBM TechWeb

www.drdobbs.com

Previous
Next

Dr. Dobb's Journal

CONTENTS

July 2011

COVER STORY

7 olap4j: Online Analytical Processing for Java

By Luc Boudreau

The olap4j API offers online analytical processing of relational data with Java. Specifically, olap4j extends core classes of JDBC specifications 3 and 4 in order to bring online analytical processing data sources to the Java platform.

5 Editorial

By Andrew Binstock

Immutability makes data items ideal for sharing between threads. It enables two threads to access a string simultaneously without the usual elaborate locking mechanisms

12 Making the Move to Maven 3

By Jason van Zyl

Maven 3 is a drop-in replacement for Maven 2 and fully backwards compatible. So aside from addressing duplicate dependencies and plugin declarations, you don't have to make any changes to your Project Object Models (POM) file.

16 From the Vault: The Rete Matching Algorithm

By Bruce Schneier

String comparison algorithms are plentiful in computer science. You have Boyer-Moore, Knuth-Morris-Pratt, Rabin-Karp, and even the old-fashioned manual comparisons. Expert systems also require extensive pattern matching during their execution, but it is a special case of pattern matching. The Rete matching algorithm was invented to speed up that pattern-matching process.

3 Letters

By you

Readers share their views on small classes and language tutorials.

22 Links

Snapshots of the most interesting articles on drdobbs.com, including a new series on creating music components with Java and a fast, high-quality parallel random number generator.

23 Editorial and Business Contacts

More on DrDobbs.com

Interview with Ken Thompson

The cocreator of UNIX was recently awarded the Japan Prize, one of the highest honors for outstanding contribution to science and technology. He discusses writing UNIX, the Go language, and collaborating with Dennis Ritchie.

http://drdobbs.com/open-source/229502480

In Praise of Small Classes

How can you avoid introducing complexity in your code? Andrew Binstock has a sugestion...

http://drdobbs.com/architecture-and-design/230300002

Soot: Analyze, Transform, and Optimize Java Bytecodes

This open source toolkit transforms source or bytcodes into SSA statements that can be analyzed, optimized, or used as a program analyzer.

http://drdobbs.com/java/229700038

A Plugin-Based Graphical App in C# Using MEF and Embedded IronPython

Graphr is a Windows Presentation Foundation-based GUI program implemented in C# that allows you to generate a dataset based on a rule. This article demonstrates Graphr, explains in detail how to embed IronPython, and explores its use in a plug-in architecture.

http://drdobbs.com/windows/229700040

Microsoft's Fickle APIs

Will Microsoft abandon Silverlight now that HTML5 is increasingly the center of the Rich Internet Application universe?

http://drdobbs.com/windows/229402780

Mailbag

Readers weigh in on small classes and lax language tutorials

Small Classes

In response to the editorial "In Praise of Small Classes" (see http://drdobbs.com/architecture-and-design/230300002), many letters arrived communicating the following:

Back in the day, I managed a development group. We wrote in COBOL. I taught and enforced "structured programming." One rule was that a paragraph should fit on one page of green bar printer paper; in other words, fifty lines. A reader (like me) didn't have to turn a page to see what it was doing. Some good ideas persist.

—Don Mackenzie
Software Architect
Cox Digital Solutions

Andrew Binstock responds: "As I point out in my current editorial (Making Large Classes Small, http://drdobbs.com/230600127), I was writing about classes, not paragraphs nor even functions. So the pressure to really keep code small and well refactored is even greater than this letter would suggest."

When you speak of a class being one page, is that the header, or the body as well? I can understand keeping the class declaration to one page, but keeping the class code to one page seems very restrictive, and I think I would find it very difficult to keep things to that size, even when obeying the one-class : one-responsibility rule.

—Simon Parker

Andrew Binstock responds: "I generally collapse the header in my IDE. Without that, I agree, it would be difficult to get very much done in the 50-60 lines, as copyright statements can be very long. What I propose is indeed a restrictive limitation, but precisely by forcing you to refactor aggressively as you approach the limit, it delivers the benefit of small, SRP-compliant classes. If you don't hew to the limit, it's easy to slip back into banging out the code in large, agglutinated classes. As I suggest, it requires a constant attention. However, over time, it changes the way you look at writing code and becomes part of design 'muscle memory.'"

Lax Language Tutorials

In response to the editorial "Lax Language Tutorials" (see http://drdobbs.com/tools/229700183):

Until you've stood in front of 150 pairs of eyes for 15 weeks, semester after semester, year after year, and seen what works and what doesn't work, you are not going to write a good tutorial. Writing without feedback just doesn't work.

I've been lucky enough to been teaching various programming languages for over 25 years. It is impossible to convey the inner satisfaction one gets when you present an example and actually see an epiphany occur in a student's eyes when they "get it" because of your example.

— Jack Purdum

Andrew Binstock responds: "Jack was the driving force behind EcoSoft, one of the early vendors of C compilers for the x86 platform. For the last several decades, he's been teaching at Purdue. He has written a tutorial on C# (http://is.gd/eW1Fl8), which I have not seen, but look forward to reading in light of his remarks. "

I could not agree with you more. I have now been in the industry for 25 years (plus 6 in college), and have found K&R to be the most succinct reference for a language I've ever used. K&R (2nd edition) is a little enhanced, but does not lose the flavor of the original.

One thing that K&R does, that other books you've mentioned do not, is assume that the reader already knows how to program. Using K&R as an introductory textbook in a college course would be inappropriate (I've used Kelly & Pohl's "A Book on C" in that role), but for the professional, it is perfect.

Over the years, I've had to learn quite a few different languages, and the tutorials that give quick examples, with the correlation to something I already know, have made my life easiest.

—Jerrold Heyman
Technical Consultant
IBM

Thanks for the article about some of the good and bad language books. I share your observations about other language books. I have been working on learning Objective-C for the past eight months and I have been through a few Objective-C books. It is quite amazing — every book seems to take the memory management section from Apple's developer website, reword it, and put it in their book. Most of Apple's documentation in this section is not written all that well. A visit to devforums.apple.com reveals that most everyone has difficulties with memory management under Objective-C. I have yet to find a book, online video, or tutorial that completely explains Objective-C memory management.

—Paul Lohr

I appreciate your thoughts on the limitations of many language tutorials. I need to put in a word for a book that saved me back in my school days: The Little LISPer.

I think it is impossible to work through this and not come out the other end with a working knowledge of LISP and some understanding of how recursion really works.

— Matt Greising
Accenture Business Process Outsourcing

Have a correction or a thoughtful opinion on Dr. Dobb's content? Let us know! Write to Andrew Binstock at alb@drdobbs.com. Letters chosen for publication may be edited for clarity and brevity. All letters become property of *Dr. Dobb's*.

IN THIS ISSUE

The Need For Immutability

It makes data items ideal for sharing between threads

For many programmers, the emergence of data immutability as a desirable feature in programming languages is a curious development. Immutability — the capacity to create variables whose initial value cannot be changed — is suddenly the mode.

When the programming world was dominated by C and C++, most instructional materials barely touched on immutability. The entire conversation recognized the occasional need for constants and proscribed the use of a magic number, such as 3.14159. Eventually, a constant, PI, was suggested to help the good folks who'd have to maintain the code at some time in the future.

Except for the plaintive cry of academics whose fondness for functional languages was thoroughly ignored, the above was pretty much all you heard about data immutability. This situation changed with the advent of Java in 1995: The language implementation hid immutability behind the scenes. Strings, as well as other fundamental data types, were constants, rather than variables. This design was in part a reaction to the great difficulty of managing strings in C and C++. Because those languages treat strings simply as null-terminated arrays of characters, C strings are infinitely pliable, plastic entities that anyone with a copy can modify. Java, in counterpoint, views strings as their own fundamental and immutable data type.

Java expanded immutability in the release of Java 2 by adding immutable collections. These collections are quite useful in regular serial programming. For example, a getter that returns a collection should in most cases return an immutable collection. This step enforces data hiding and encapsulation: Objects that don't own the collection cannot change its values.

Immutability makes data items ideal for sharing between threads. It enables two threads to access a string simultaneously without the usual elaborate locking mechanisms.

With the wide adoption of x86 multicore processors, all programs have the possibility of useful parallelization, and so immutability is

Previous Next

IN THIS ISSUE
Editorial >>
Olap4j>>
Maven 3>>
Rete >>
Letters>>
Links>>
Table of Contents >>

moving inexorably to the fore. New languages, such as Scala, provide for it expressly (a one-letter change in a variable declaration creates an immutable constant). Languages derived from the functional world (Erlang and Clojure, for example) embrace immutability even further, making it the default behavior for variables. Languages, such as Groovy, that did not have immutability as a definable quality, first added it as an adjunct (in Groovy's case, as an annotation), and then as an integrated part of the language. C and C++ are laggards here — *const* correctness being the partial and somewhat cumbersome mechanism.

For developers not interested in multithreading, immutability still has a role to play as I've mentioned. But the full breadth of opportunity is much greater. It's safe to say that wherever possible, data items should be declared as immutable. The first benefit is performance. Compilers are very good at optimizing code when they know a data item won't change value.

Even if your code is fast enough, immutability has value. By specifying that an object is immutable, you can catch defects that might have been difficult to detect. For example, the long held practice of marking parameters to methods as *final* prevents you from mistakenly modifying a value that will be wiped away when the method returns.

COMING SOON TO DRDOBBS.COM

The next installment of our series on Cloud Programming

Fun with PHP

Herb Sutter Returns!

However, even within methods, there are many times when an object is returned from a function only for purposes of calling one of its methods. It too can be marked immutable. This prevents mechanical errors and provides greater readability. (As with all guidelines, this has to be tempered by the pragmatic realities. Code clutter, especially in languages that don't have simple immutability keywords, can be a drawback that more than offsets the readability benefits.)

My belief is that data immutability will become much more pervasive part of all programming languages — fully integrated at the syntactical and semantic levels. This, I expect, will presage the wider penetration of parallel programming into general-purpose languages.

— Andrew Binstock is Editor in Chief for Dr. Dobb's and can be contacted at alb@drdobbs.com.

olap4j: Online Analytical Processing For Java

Online analytical processing (OLAP) is capable of leveraging multiple processors, but it has long been hampered by the lack of standardized APIs...until now!

By Luc Boudreau

The advent of multicore and many-core processors on expensive desktops and servers has ushered in an era in which small companies can fairly easily perform data mining and analytical processing of large databases as part of an effort to optimize business performance.

Online analytical processing (OLAP) is capable of leveraging multiple processors, but it has long been hampered by the lack of standardized APIs, especially for Java. Not that nobody was working on them...

In fact, over the past two decades, many Java Community Process initiatives have tried to establish successful APIs, but all efforts have failed; in many cases because they relied on proprietary extensions. This article discusses a recently released API, supported by multiple vendors in the business intelligence (BI) sector, which makes it easy to leverage today's processors to glean insights into business data.

The API, known as olap4j (http://www.olap4j.org/), is the OLAP equivalent of Java database connectivity (JDBC) for relational data. Specifically, olap4j extends core classes of JDBC specifications 3 and 4 in order to bring OLAP data sources to the Java platform. Connections can be obtained by JDBC connection management facilities.

The API uses statements, which are provided over connections. The queries, formulated using the MDX language, can be sent textually to the connection. The olap4j API also includes a type system capable of representing any server-specific MDX grammar as a business object model.

The API makes heavy use of metadata, as most OLAP-related operations involve a user exploring the database and building queries interactively. Results of multi-dimensional queries are represented by a *CellSet* object. *CellSet* is to multi-dimensional data what *ResultSet* is to tabular data.

IN THIS ISSUE
Editorial >>
Olap4j >>
Maven 3>>
Rete >>
Letters >>
Links >>
Table of Contents >>

Getting Started

The olap4j API is distributed as four modules, which are shown in Table 1. For the purpose of this article, I will use both the core module and the XML/A driver. You can find those libraries at the olap4j website (http://www.olap4j.org/) and in the Resources section at the end of this article.

Connecting to an OLAP Server

Connecting to an OLAP server is done in the same way you connect to a relational database. As I mentioned earlier, olap4j leverages JDBC driver management and connection facilities. A sample is shown in Listing One.

Listing One: Connecting to a remote OLAP server.

```
Class.forName(
  "org.olap4j.driver.xmla.XmlaOlap4jDriver");
final Connection conn =
  DriverManager.getConnection(
    "jdbc:xmla:Server=http://example.com/xmla/msmdpump.dll;"
    + "Cache=org.olap4j.driver.xmla.cache.XmlaOlap4jNamedMemoryCache;"
    + "Cache.Mode=LFU;Cache.Timeout=600;Cache.Size=100");
final OlapConnection oConn =
  conn.unwrap(OlapConnection.class);
```

olap4j	This is the core API, compatible with JDBC 3 and 4.
olap4j-xmla	The XMLA module includes a generic XML for Analysis compliant driver. It is able to connect remotely to Jedox Palo, Pentaho Analysis, SAP BW, and SQL Server Analysis Services.
olap4j-jdk14	Olap4j is also available as a module that is backwards-compatible with Java 1.4. It includes both the core API and the XML/A module.
olap4j-tck	A test compatibility kit for olap4j implementors.

Table 1: olap4j API modules.

The code in Listing One will give you a connection object to a remote OLAP server using the XML/A driver. We use JDBC 4's connection unwrapping to obtain access to the underlying OLAP connection. Applications developed for Java 5 can cast the connection object directly.

Take note that the code in Listing One uses the XML/A driver's caching SPI. A few basic implementations of the cache are provided with olap4j, but integrators are encouraged to develop their own. In this case, the SOAP requests will be cached in memory and use a Least-Frequently-Used eviction policy, enforced for a maximum of 100 elements over 10 minutes.

More configuration options exist for both the XML/A driver or the cache SPI within the API documentation.

Exploring an OLAP Server

The metadata hierarchy of an OLAP server is different from what you might be accustomed to with relational databases. The information is organized according to the hierarchy illustrated in Figure 1.

The olap4j API exposes the metadata in two ways. It can be explored either through *DatabaseMetaData*, in the form of tabular data, or by exploring a hierarchy of business objects. Listing Two shows how this is done programmatically.

Listing Two: Exploring olap4j metadata.

```
final OlapDatabaseMetaData meta =
  oConn.getMetaData();
// We can explore the meta data in a tabular form
final ResultSet catalog = meta.getCatalogs();
// We can also explore using the object model
final Schema schema = oConn.getOlapSchema();
for (Cube cube : schema.getCubes()) {
  System.out.println(cube.getName());
}
```

Previous Next

IN THIS ISSUE
Editorial >>
Olap4j >>
Maven 3>>
Rete >>
Letters>>
Links>>
Table of Contents >>

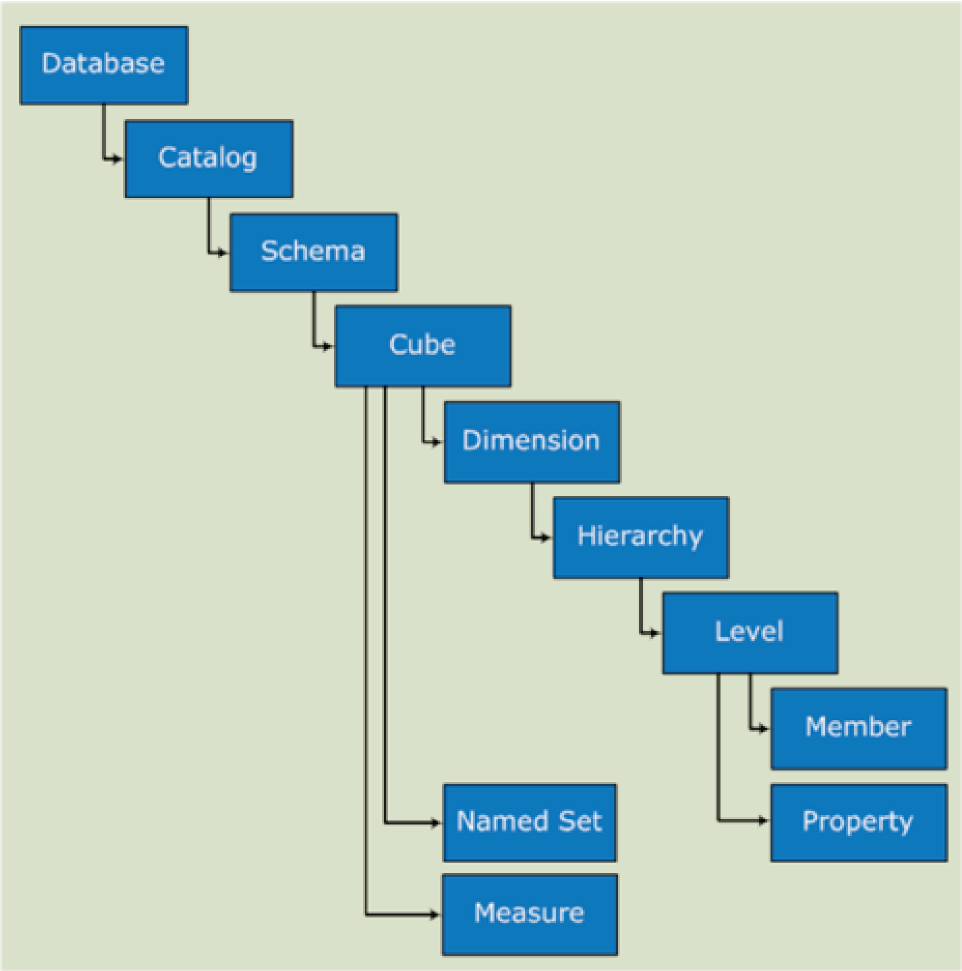


Figure 1: Layout of the olap4j metadata hierarchy.

By convention, OLAP connections are bound to a database, a catalog, and a schema. Although a user can modify the binding easily by simply invoking setters or passing JDBC arguments in the connection's URL, connection implementations are expected to bind automatically to the first elements of the hierarchy encountered, should none be specified. Users of olap4j should keep this in mind when using an OLAP connection's sugar accessors, such as *getOlapCatalog()*. Retrieving a list of *Catalogs* implies a prior binding to a *Database*. If you have not set a specific database to use, the connection will bind itself to the first one it encounters, then return all catalogs within this database.

Parsing and Validating Queries

We are now ready to query the server for data. Let's write a query and parse it (Listing Three). The parser will check it for correct grammar.

Listing Three: Querying the server.

```
final String myQuery =
  "SELECT " +
  "{[Drink].[Beverages].Children} " +
  "ON COLUMNS " +
  "FROM [Sales] " +
  "WHERE ([Time].[Cake Time!])";
final MdxParser parser =
  oConn
    .getParserFactory()
    .createMdxParser(oConn);
final SelectNode parsedObject =
  parser.parseSelect(myQuery);
System.out.print(
  parsedObject.toString());
```

A *SelectNode* object is a representation of your query, as interpreted by your connection's specific grammar. The API does not endorse any singular MDX grammar (so as to keep the project independent of OLAP vendors). Instead, the connection itself has to expose a parser that is capable of bridging the grammar with olap4j's type system.

Once the textual query is parsed, you are left with an object representation of it, but there is no telling if this query, aside from being grammatically correct, is valid. To determine if the query is executable and will return data, the query validator (Listing Four) is used.

Listing Four: The results of a query validation.

```
     oConn
  .getParserFactory()
  .createMdxValidator(oConn)
  .validateSelect(
    parsedObject);
->: Mondrian Error:MDX object '[Time].[Cake time!]'
    not found in cube 'Sales'
```

The validator is a helper object used to validate queries further. It looks up all members and hierarchy elements referenced in your query and makes sure that they exist on the target server. In our example, indeed, there is no such thing as cake time [*alas!* —*Ed*.].

Validating the query might seem like overhead, as executing the query straight away will indeed tell you whether the query is valid or not. But when a query is validated, only the metadata is evaluated against the server, and no data is computed or returned. This means that the server won't have to compute all the tuples of members and associated data. Validating a query is a very fast operation compared with executing it. If you are developing a GUI where a user sits in front of the screen until an answer is given, you should keep that in mind.

Executing Queries

Executing a query against an OLAP server is similar to what you would do for a relational database. Queries can consist of a query parsed into a *SelectNode* object, as in Listing Five(a) or executed using a *String* of MDX, as in Listing Five(b).

Listing Five(a): Executing a query with SelectNode.

```
// We can execute the parsed SelectNode itself...
CellSet data =
  oConn
    .createStatement()
      .executeOlapQuery(parsedObject);
```

IN THIS ISSUE
Editorial >>
Olap4j >>
Maven 3>>
Rete >>
Letters >>
Links >>
Table of Contents >>

Listing Five(b): Executing a query with MDX.

```
// ...or use the MDX query directly.
data =
  oConn
    .createStatement()
      .executeOlapQuery(myQuery);
```

Reading the Results of an OLAP Query

Once the query is executed, you can explore the returned data. A *CellSet* object is different than a *ResultSet*. You must think of CellSets as data organized on any number of axes. On each of those axes, there are a series of positions. Each position describes a set of tuples, formed by cross-joining all the members of the current axis.

Listing Six: Iterating over the results of an olap4j query.

```
// Iteration over a two-axis query
for (
  Position axis_0
  : data.getAxes().get( Axis.ROWS.axisOrdinal() ).getPositions())
{
  for (
    Position axis_1
    : data.getAxes().get(Axis.COLUMNS.axisOrdinal()).getPositions())
  {
    Object value =
      data.getCell(axis_0, axis_1)
        .getValue();
  }
}
```

The code in Listing Six demonstrates how to iterate over the results of a query that included only two axes; columns and rows. If your query uses more than two axes, you will need to iterate over each of them. In our simple two-dimensional use case, olap4j includes a very handy helper package to help display the results of the query:

```
// We use the utility formatter.
RectangularCellSetFormatter formatter =
  new RectangularCellSetFormatter(false);

// Print out.
PrintWriter writer = new PrintWriter(System.out);
formatter.format(cellSet, writer);
writer.flush();
```

Going Further

With the information presented so far, you should now be able to connect, execute queries against an OLAP server, and obtain meaningful results. The olap4j project can get you far beyond this point. The Resources section contains links to materials that showcase the most advanced features, including how to build a user interface to explore OLAP data sources using a programmatic query model, statistical simulations, and real-time updates.

Resources

olap4j.org: Home of the olap4j project: http://olap4j.org/
olap4j.org/api: The latest API reference: http://olap4j.org/api
olap4j download resources:. http://is.gd/3Z7kyf

—Luc Boudreau is a comanager of the olap4j project and Mondrian, an open source OLAP engine. He works at Pentaho Corporation as a Senior Software Engineer and is an active member of many community projects related to business intelligence.

Comment

Making the Move to Maven 3

You won't lose any time in upgrading to Maven 3

By Jason van Zyl

Most Java programs use either Ant or Maven for the build step. While Ant is infinitely configurable, Maven employs convention instead of configuration. Conventions include location of files, sequence of build steps, etc. For the last few years, Maven 2 has been the standard release.

This article discusses Maven 3, which is now available. Java developers who have been relying on Maven 2.x for project build and reporting management may wonder why they should switch to Maven 3. There are many reasons.

Before I get into those, you should know that you won't lose any time in upgrading to Maven 3. It is a drop-in replacement for Maven 2 and fully backwards compatible. So aside from addressing duplicate dependencies and plugin declarations, you don't have to make any changes to your Project Object Models (POM) files, because the command line is fully compatible between Maven 2 and 3.

Performance and Memory Use

Maven 3 is faster and has a smaller memory footprint than Maven 2. While your mileage may vary depending on the specific structure of your project, benchmark testing using a dual-core CPU with 4GB RAM, running Microsoft Windows XP with JDK 1.5 showed the following improvements:

"mvn package" on Maven SCM Trunk (32 modules)

Maven Version	*Time*	*Max Memory*
Maven 2.2.1	3:20	99M
Maven 3.0.2	3:15	51M

"mvn package" on a Corporate Project (11 modules)

Maven Version	*Time*	*Max Memory*
Maven 2.2.1	1:04	48/87M
Maven 3.0.2	0:54	15/35M

This increase in speed is even greater for parallel builds, which is a new feature supported by the Maven 3 core. This feature requires that plugins used during parallel builds declare themselves as thread safe. Here are the benchmark results for parallel builds using Maven 3:

“mvn package” on Maven SCM trunk (32 modules)

Maven Version	*Time*	*Max Memory*
Maven 3.0.2	3:15	21/57M
Maven 3.0.2, 4 threads in parallel	2:26	28/62M

“mvn package” on a Corporate Project (11 modules)

Maven Version	*Time*	*Max Memory*
Maven 3.0.2	0:54	15/35M
Maven 3.0.2, 4 threads in parallel	0:40	16/42M

**Notice also the decrease in memory consumption.*

Better POM and Dependency Management

This release of Maven is much stricter than Maven 2 in validating POM files. It quickly complains about obvious errors and omissions. For instance, Maven 3 warns users if they haven’t specified the plugin versions to use or if they have duplicated dependencies that could lead to unpredictable build results. Unlike Maven 2, which made developers entirely responsible for learning and incorporating best practices into builds, Maven 3 provides clear direction regarding what should be happening in the POMs and points developers toward more concise and complete POM configuration.

Maven 3 also separates project dependencies from plugin dependencies, which is particularly helpful in large projects. In Maven 2, plugin dependencies could be retrieved from any Maven repository. Because plugin dependencies could intermingle with regular project dependencies, users managing complex projects were forced to meticulously track hundreds of dependencies. In Maven 3, the project dependencies are retrieved from declared “repositories” while plugin dependencies are retrieved from “pluginRepsitories.” This distinction significantly reduces procurement complexity and simplifies repository management.

Integration with Other Products

Maven 3 features new extension points that optimize IDE use, allowing seamless integration with the Hudson CI system as well as support tools, including Polyglot Maven and Maven Shell.

The Maven Shell is embedded in a long-lived shell process that can shorten build times significantly. It caches POMs and eliminates startup costs associated with repeated use of Maven. The Maven Shell includes a built-in help system and provides Growl support on Mac OS X. It also supports project workflow, and support for Hudson, Tycho, and Polyglot Maven.

No XML: Polyglot Maven

Developers dissatisfied with Maven’s original XML format can use Polyglot Maven, which is easily integrated with Maven 3 via an extension point. Polyglot Maven supports dynamic languages, including YAML, and aims to provide first-class POM-mapped Domain-Specific Language (DSL) support for Clojure, Groovy, Ruby, Scala, and Xtext.

Eclipse Support

Embedding changes in Maven 3 significantly improve its performance inside m2eclipse (the first Maven integration plugin for the Eclipse IDE). In fact, while running in this customized plugin environment,

IN THIS ISSUE
Editorial >>
Olap4j >>
Maven 3>>
Rete >>
Letters >>
Links >>
Table of Contents >>

Maven 3 is capable of up to a 300% performance boost. M2eclipse provides additional XML metadata in the Maven POM that is recognized by m2eclipse and enables high build performance. M2eclipse also downloads all sources automatically and has a single-click new project creation feature for any of your dependencies. Of course, if you're using m2eclipse now, you are already using Maven 3. Switching to Maven 3 for command-line executions will align Maven versions in the tools and add consistency between CLI and your IDE.

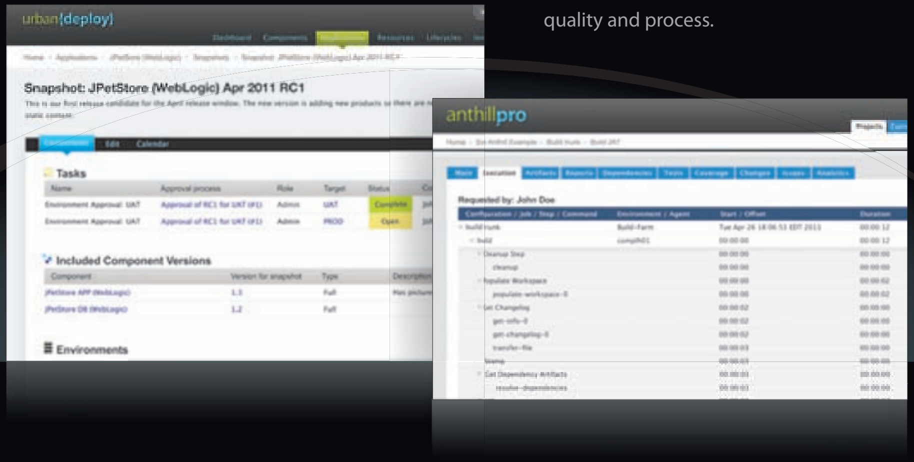
Hudson Integration

Maven 3's revamped internals support more efficient embedding. Hudson and Sonatype's multi-year investment in bringing both JSR-330 support and GWT UI integration to Hudson CI has resulted in a continuous integration platform that supports Maven 3 particularly well.

To ensure that Hudson users have the flexibility they need to create quality enterprise builds, Sonatype has taken Maven 3 support in a new direction. In a complex build environment, Maven is often not the only build tool involved in a process. So instead of binding users to a single Maven-specific project type, there is a Maven builder that can be used as part of a larger freestyle build. Maven 3 build processes can contain two separate calls to *release:prepare* and *release:perform* to better support real-world Maven builds that need to prepare and perform releases. Developers can also call out to supporting scripts or other build parts that fall outside of the scope of a Maven build.

Fixes to Maven 2

Today, focus has shifted to Maven 3 and few committers are paying attention to Maven 2. While Maven 2.x defects do eventually get fixed, there is no longer a sustained effort to test and push new Maven 2 releases. In contrast, Maven 3 has a six-week release cycle.



Maven 3 has improved artifact/dependency resolution. Previous artifact resolution deficiencies in reactor builds (that is, builds in which a goal is executed on a specific set of modules) led to problems when Maven 2.x users tried to build multi-module projects unless they executed "mvn install." This was caused by the fact that artifacts built by the modules were not resolved properly from the reactor. This defect led to an extra step of configuring the *preparationGoals* parameter of a maven-release-plugin to "clean install" instead of the default "clean verify." Maven 3 corrects this problem and resolves artifacts within the reactor, which allows you to use the default values of the *preparationGoals* parameter to streamline the build process.

Maven 3 also responds directly to the request to provide more intelligent class loading for multi-module builds. In Maven 2, the first execution of a plugin determined the classpath for all subsequent executions of that plugin. This often led to complications when a plugin required the use of a different classpath in its second invocation during a build. For example, Maven 2 users who needed to execute the Antrun plugin more than once in a multi-module build with two different classpaths were simply out of luck. This issue was fixed in Maven 3 by making sure that every execution gets its own, isolated classpath.

Interestingly, the Maven 3 codebase is two-thirds the size of the Maven 2 codebase. This is due in part to Maven 3's use of Google Guice for dependency injection and Sisu to extend Guice's OSGi capabilities. This release moves the entire dependency resolution mechanism into a standalone product, Sonatype Aether, for which Maven 3 is a client.

Continuous Evolution

Maven 3 is continuously evolving and is currently at version 3.03. Looking ahead, Maven 3.1 will include a security manager with the settings.xml file specifying the default. Maven 3.1 will also introduce POM mixins, which are a type of POM composition that enables parameterized POM fragments to be injected into the current POM with a simple reference. POM mixins make configuration more portable and maintainable, solving the Maven 2 problem where sharing configuration could only be done via inheritance.

Maven 3 is an Apache project and can be downloaded as open-source software at http://maven.apache.org.

—Jason van Zyl is CTO and Founder of Sonatype, and the founder of the Apache Maven project, the Plexus IoC framework, and the Apache Velocity project. He goes by @jvanzyl on Twitter.

Comment

From the Vault

The Rete Matching Algorithm

The Rete algorithm was invented to speed up the pattern-matching process. Back in 1992, Bruce Schneier introduced *DDJ*'s "AI Expert Newsletter" readers to its benefits.

—DDJ

By Bruce Schneier

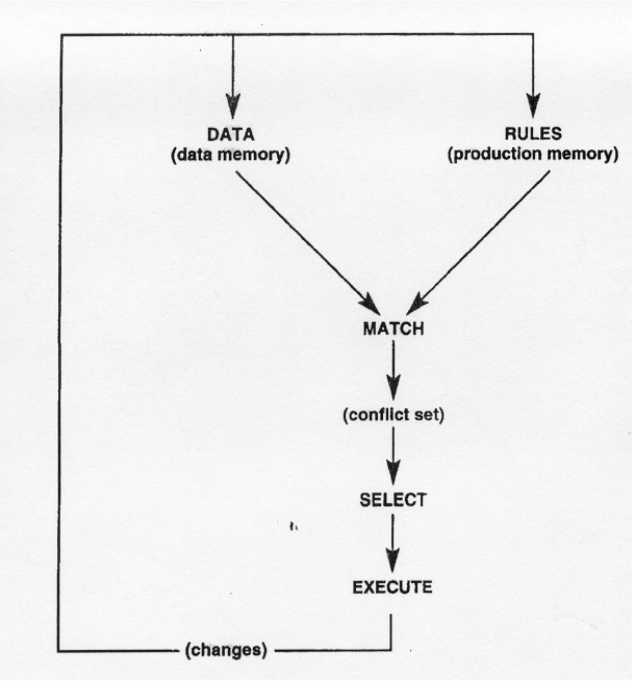
String comparison algorithms are plentiful in computer science. You have Boyer-Moore, Knuth-Morris-Pratt, Rabin-Karp, and even the old-fashioned manual comparisons. Expert systems also require extensive pattern matching during their execution, but it is a special case of pattern matching. The Rete matching algorithm was invented to speed up that pattern-matching process.

One class of expert systems is known as production systems, which contain a database of production rules that govern behavior. Production systems such as OPS5 are straightforward; see Figure 1.

Everything is orchestrated by the interpreter. A set of rules sits in production memory, and a set of facts, or assertions, sits in a database called data memory. First, a pattern matcher looks at both memories to see which rules have their conditions satisfied by the facts. After the matcher generates a list of rules whose conditions have been satisfied, called the conflict set, it calls sort of conflict-resolution system.

This system looks at everything in the conflict set and, according to some algorithm, determines which particular rule will fire. The rule makes some changes in either data memory or production memory, and the cycle starts again.

Matching algorithms compare all the data elements in data memory with all the rules in production memory to determine which rules have their conditions satisfied and should be put in



nbol → is the

Figure 1: Production systems such as OPS5.

the conflict set and sent to the conflict resolver. For example, an expert system for a detective might have the following rules in production memory:

- If a person *x* did something illegal, then person *x* is a criminal.
- If fingerprints belonging to person *x* are on object *y* then person *x* touched y at some point in the past.
- If person *x* shot person *y* then person *x* did something illegal.
- If person *x* is dead, then person *x* should not be invited over for dinner.

And it might have the following data element in data memory:

- Person Fred shot person Sam.
- Person Sam is dead.

The Rete matching algorithm product is a conflict set containing two rules: If person *x* shot person *y*, then person *x* did something illegal, and If person *x* is dead, then person *x* should not be invited over for dinner.

All this takes a lot of pattern matching. Each time the inference engine cycles, the pattern matcher must compare the data stored in data memory with the rules stored in production memory to see which rules had all their conditions satisfied. This comparison process can be very lengthy if the production system has hundreds of data items and thousands of rules, as many real systems do.

Charles Forgy, inventor of the OPS5 rule-based language, devised the Rete algorithm in the late 1970s to speed up the comparisons. He found that old systems spent as much as 90% of their performing pattern matching. They would iterate through the process, taking each rule in turn and looking through data memory to see if the conditions for the rule were satisfied and then proceeding to the next rule. Some people found ways to index data elements and rule conditions. This speeds up program execution, but still requires iterating through the series of rules and data elements. The Rete algorithm eliminates the iterative step and, hence, is a substantial improvement over competing algorithms.

"The Rete matching algorithm avoids iterating through the data elements by storing the current contents of the conflict set in memory, only adding and deleting items from it as data elements are added and deleted from memory"

The Rete matching algorithm avoids iterating through the data elements by storing the current contents of the conflict set in memory, only adding and deleting items from it as data elements are added and deleted from memory.

For example, assume that the expert system for a detective has the following rule in its conflict set: *If someone is discharging a firearm at me, then duck*. This means that somewhere in data memory is the fact that someone is discharging a firearm at me.

The conflict set might contain other rules: *If someone is discharging a firearm, then he is committing an illegal act; if someone is wearing a sym-*

bol of an inverted badger, then he is a member of the Dreaded Legion of Inverted Badgers, and so on. When it comes time to update memory, either as a result of new input or of as a result of rule firing, the Rete algorithm would update the conflict set. Let's say the expert system decided to have our hero duck (a prudent move). That would delete the fact that someone is discharging a firearm at our hero; hence the rule, *If someone is discharging a firearm, then he is committing an illegal act,* would not be deleted from the conflict set.

> **"The pattern matching algorithm never has to look at data memory, because the algorithm keeps a record of how data memory affects the rules and the conflict set"**

On the other hand, the conflict resolver might decide to fire a different rule. As a result, this might add the fact that the person shooting him is a member of the Dreaded Legion of Inverted Badgers. Another rule might have its conditions satisfied, and it would be added to the conflict set. For example: *If a member of the Dreaded Legion of Inverted Badgers discharges a firearm at me, then it is futile to duck because the firearm is equipped with special Inverted Badger Homing Bullets*...and so on.

The point of all this is that the interpreter doesn't have to iterate through the entirety of the data memory to see if a given pattern matches any of the data memory elements. A list of elements that match is stored with each pattern. When a new element is added to data memory, then the interpreter finds all the matches it can, and adds them to the lists. When an element is deleted from data memory, the interpreter finds all the patterns that matched it, and deletes them from the lists. The pattern matching algorithm never has to look at data memory, because the algorithm keeps a record of how data memory affects the rules and the conflict set.

This alone would still force the pattern matching to iterate through production memory, looking for rules that must be updated because of changes in data memory. However, there is another half that iteration as well.

The Rete algorithm stores the conditions of all the production rules in a tree-structured sorting network. The Rete algorithm complies the network from the list of production rules and then keeps the network current as rules are added and deleted from the lists.

To see how that works, let's look at some rules in a candy company's expert system. Two production rules in that system are:

- Rule *Red__Round__Ones*: If the goal is to identify a piece of candy, and if the candy sample is red, and if a candy sample is round, then hypothesize it is a jellybean.
- Rule *Red__Cylindrical__Ones*: If the goal is to identify a piece of candy, and if the candy sample is red, and if the candy sample is cylindrical, then hypothesize it is a licorice stick.

In OPS5 talk:

```
(P Red__Round__Ones
     (Goal | Type Identify | Object {N})
     (Candy | Name {N} | Color Red | Shape Round | Company {X})
->
     then clause, which is not important in this context)
->
(P Red__Cylindrical__Ones
     (Goal | Type Identify? Object {N})
     (Candy | Name{N} | Color Red | Shape Cylindrical | Company {X})
->
   then clauses
```

OPS5 notation is easy to understand. *P* is the symbol for the production rule; it is followed by the name of the rule. The two lines after the name are the two elements that make up the statement that consists of class, type, element, and a list of attributes. The symbol | is OPS5's way of distinguishing attributes (*Shape*, for example) from values (*Cylindrical*, for example). *Data* memory might look like:

```
(Goal | Type identify | Object Sample7)
(Candy | Name Sample7 | Color Red  | Shape Round | Company FerraraPan)
```

To determine if the conditions for the first rule were satisfied, the pattern matcher would first check whether there was a data element in data memory of class *Goal*, with the *Type* attribute of *Identify* and some *Object* variable. Then it would check whether there were some data element of class *Candy*, with some *Name* variable, a *Color* attribute of *Red*, a *Shape* attribute of *Round*, and a *Company* attribute of some unknown variable. Finally, it would compare the value of the *Object* variable of the *Goal* with the value of the *Name* variable of the *Candy*. If all those subconditions were satisfied, the pattern matcher would add the rule *Red__Round__Ones* to the conflict set.

Then, if it were a conventional iterative pattern matcher, it would step down to the rule *Red__Cylindrical__Ones* and perform almost the exact same steps. Those two production rules are almost identical.

Rete's tree-structured sorting network takes advantage of these redundancies. The pattern complier builds a network of individual subconditions. It first looks at each element of a production rule individually, building a chain of nodes that tests for each attribute individually. Then it looks at comparisons between elements (comparing the *Object* variable of the *Goal* with the *Name* variable of the *Candy*, for example) and connects chains with new nodes. Finally, terminator nodes are added to signal that all the conditions for the production rule have been satisfied. The tree structure for *Red__Round__Ones* is shown in Figure 2.

Additional production rules are grafted onto the same network. If they have no test in common, they don't interact at all.

How does all this work? The root node starts the process. Whenever a new element is added to data memory, it distributes it down the tree. Single-input nodes do their tests. If the element passes the test, then the node passes it downstream; if the element fails the test, then the node drops it into the bit bucket. Two-input nodes behave the same way, except that if they get only one input, they save it until the second input comes along. If, sometime later in the expert system's churning, the second element entered data memory, the pattern matcher would not have to process the first element all over again. It would just process the new pre-existing input would perform the test and either send the elements downstream or save them for next attempt.

Let's watch this in action. Assume data memory starts empty and then has the following data element added: (*Goals* | *Type Identify* | *Object Sample7*). The root distributes the element. It is not a *Candy*, so the right tree stops there. It is a *Goal*, so the left node passes it down. The value of the *Type* is *Identify*, so it passes down yet again to two different nodes. These nodes, both two input nodes, receive the element. They have only one of their two inputs satisfied, so they both save the fact that their *Goal* element was satisfied with *Object* equal to *Sample7*.

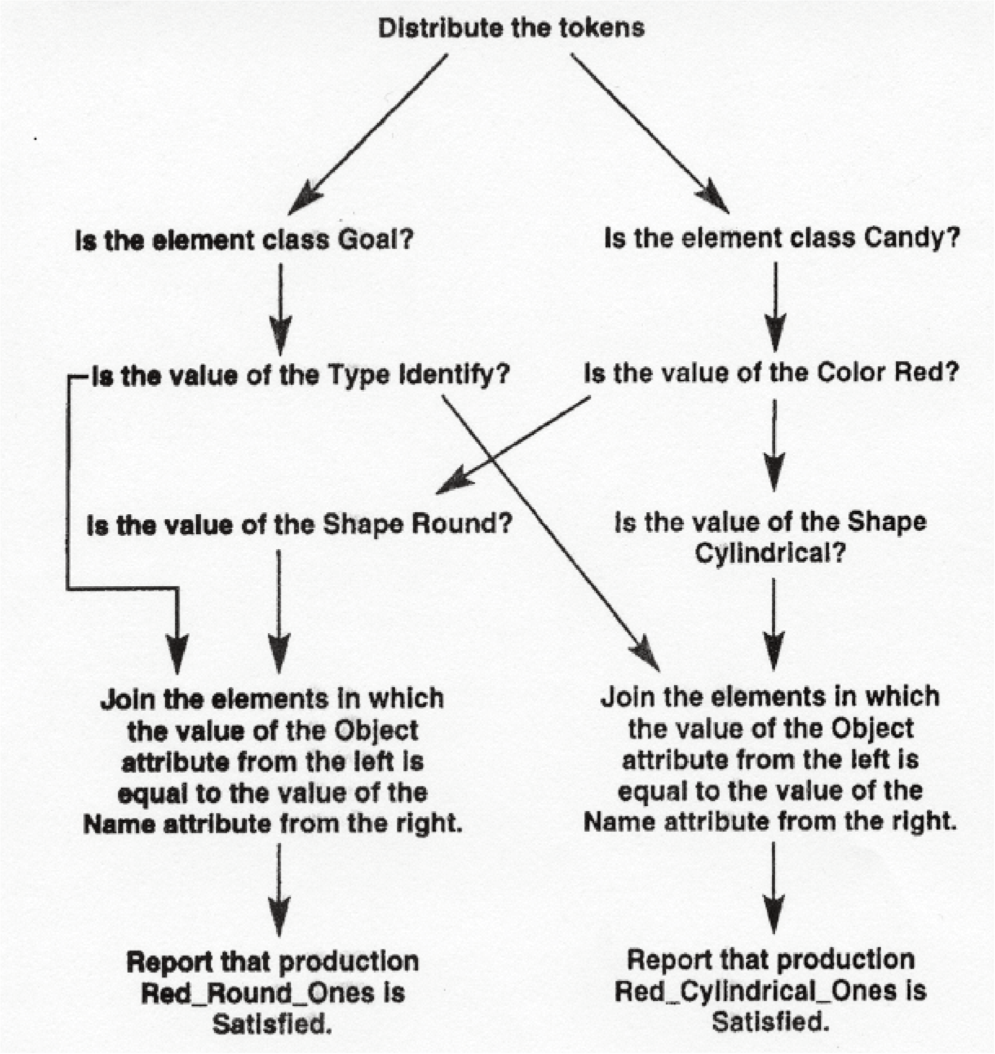


Figure 2: The tree structure for Red__Round__Ones.

Sometime later in the program's execution, the following data element is added to data memory: (*Candy* | *Name Sample8* | *Color Red* | *Shape Round* | *Company FerraraPan*). Now it is processed through the network. It is not a *Goal*, but it is a *Candy*. The value of *Color* is *Red*. Now it is passed to two different nodes. the value of *Shape* is not *Cylindrical*, but it is *Round*. Now one of the joins has both of its inputs satisfied (remember, it was saving the *Goal* element), so it looks for cases where the *Name* of the *Candy* equals the *Object* of the *Goal*. There are none, so it saves both elements. The other join doesn't do anything.

Even later, a third data element appears in data memory: (*Goal* | *Type Identify* | *Object Sample8*). It is processed like the previous *Goal* where the *Object* equals *Sample8* as well as having a *Name* that equals *Sample8*. That fact is sent downstream: The final node indicates that all the conditions for the rule *Red__Round__Ones* have been satisfied.

Deleting data elements from the network is similar. The element is processed the same way and deleted from any two-input node in which it is stored. Thus, the network is kept current and constantly mirrors the contents of data memory.

The Rete algorithm has applications in all expert system applications that involve these production rules. It is especially suited for this many object/many pattern matching problem, in which many data memory elements are compared with many production memory rules. The only

Previous Next

IN THIS ISSUE

caveat is that the elements must be relatively consistent between iterations. Since the Rete algorithm maintains itself between iterations, applications in which most of the data changes between iterations will be slowed down with this algorithm.

What the Rete algorithm basically says is that production systems are best implemented as a series of sophisticated tree walks and that everything else is nothing more than syntactic sugar. Considerable mental leverage can be gained by thinking in terms of rules and conditions, but when it comes time for implementation, that should all be thrown away. The Rete algorithm does not claim to make any conceptual changes to the production system model — it is simply a way to implement that model efficiently on a computer.

Suggested Reading

Brownston, L., R. Farrell, E. Kant, and N. Martin. *Programming Expert Systems in OPS5*, Addison Wesley, 1985 (http://is.gd/h7L9OE).

Forgy, Charles L. *On the Efficient Implementation of Production Systems*, Ph.D. Thesis, Carnegie-Mellon University, 1979.

Forgy, Charles L. "Rete: A Fast Algorithm for the Many Pattern/Many Object Match Problem," *Artificial Intelligence*, (19)1, Sept. 1982, pp. 17-37.

*— Bruce Schneier has a B.S. in physics and an M.S. in computer science. At the time this article was originally published (*AI Expert*, December 1992), he was the president of Counterpane Systems, a computer consulting firm. He has since become a well-known security expert and is author of the bestselling* **Applied Cryptography** *(http://is.gd/qOMb0l) among other books.*

Comment

This Month on DrDobbs.com

Interesting items posted on www.drdobbs.com over the past month that you may have missed

LINUS AND C++

Mark Nelson reopens a proverbial can of worms when he considers Linus Torvalds' famed assertion that "C++ leads to really really bad design choices."

http://drdobbs.com/blogs/cpp/229700143

CREATING MUSIC COMPONENTS IN JAVA

In the first entry in this series, author Craig Lindley discusses the background necessary to understand how computer based music synthesis works and providse a basic platform for experimentation. In the second article, he presents code for a multi-waveform oscillator (a staple of electronic music) and for an envelope generator, which is also an important electronic music component.

Part I: http://drdobbs.com/java/229700113
Part II: http://drdobbs.com/java/230500178

FAST, HIGH-QUALITY, PARALLEL RANDOM NUMBER GENERATORS

Very fast, parallel random number generation is possible on modern PCs, but it requires new algorithms to meet the most stringent randomness tests.

http://drdobbs.com/tools/229625477

THE FUTURE OF MICROPROCESSORS AND THE POTENTIAL TRICK BAG FOR DEVELOPERS

Shekhar Borkar and Andrew Chien recently wrote an article on the future of microprocessors for the *Communications of the ACM*. Cameron and Tracey Hughes think you should check it out, and want you to know why they see it as a fine piece of prophecy.

http://drdobbs.com/go-parallel/blogs/architecture-and-design/229700215

ALTERED STATES

Al Williams believes state machines can make embedded systems — at least some of them — much simpler to create and maintain. Here, he walks you through a small state machine library that he wrote and often employs while using C++ (though the same ideas are easily adapted to C or Java).

http://drdobbs.com/blogs/embedded-systems/229900157

ALICE 3: VISUALIZE CONCURRENT CODE IN ACTION

A new version of this innovative 3D programming environment makes it easy to understand (and to teach) the importance of concurrency.

http://drdobbs.com/go-parallel/blogs/tools/230500053

Previous Next

IN THIS ISSUE

Dr.Dobb's

Andrew Binstock Editor in Chief, Dr. Dobb's
alb@drdobbs.com

Deirdre Blake Managing Editor, Dr. Dobb's
dblake@techweb.com

Amy Stephens Copyeditor, Dr. Dobb's
astephens@techweb.com

Sean Coady Webmaster, Dr. Dobb's
scoady@techweb.com

Jon Erickson Editor in Chief Emeritus , Dr. Dobb's

CONTRIBUTING EDITORS
Mike Riley
Herb Sutter
Scott Ambler

DR DOBB'S
UBM TECHWEB
303 Second Street,
Suite 900, South Tower
San Francisco, CA 94107
1-415-947-6000

INFORMATIONWEEK
Rob Preston VP and Editor In Chief, InformationWeek
rpreston@techweb.com 516-562-5692

John Foley Editor, InformationWeek
jpfoley@techweb.com 516-562-7189

Chris Murphy Editor, InformationWeek
cjmurphy@techweb.com 414-906-5331

Art Wittmann VP and Director, Analytics, InformationWeek
awittmann@techweb.com 408-416-3227

Alexander Wolfe Editor In Chief, InformationWeek.com
awolfe@techweb.com 516-562-7821

Stacey Peterson Executive Editor, Quality, InformationWeek speterson@techweb.com 516-562-5933

Lorna Garey Executive Editor, Analytics, InformationWeek
lgarey@techweb.com 978-694-1681

Stephanie Stahl Executive Editor, InformationWeek
sstahl@techweb.com 703-266-6030

Fritz Nelson VP and Editorial Director
fnelson@techweb.com 949-223-3608

David Berlind Chief Content Officer, TechWeb
dberlind@techweb.com 978-462-5315

REPORTERS
Charles Babcock Editor At Large
Open source, infrastructure, virtualization
cbabcock@techweb.com 415-947-6133

Thomas Claburn Editor At Large
Security, search, Web applications
tclaburn@techweb.com 415-947-6820

Paul McDougall Editor At Large
Software, IT services, outsourcing
pmcdougall@techweb.com

Marianne Kolbasuk McGee Senior Writer IT management and careers
mmcgee@techweb.com 508-697-0083

J. Nicholas Hoover Senior Editor
Desktop software, Enterprise 2.0, collaboration
nhoover@techweb.com 516-562-5032

Andrew Conry-Murray New Products and Business Editor
Information and content management
acmurray@techweb.com 724-266-1310

W. David Gardner News Writer
Networking, telecom
wdavidg@earthlink.net

Antone Gonsalves News Writer
Processors, PCs, servers
antoneg@pacbell.net

Eric Zeman
Mobile and Wireless
eric@zemanmedia.com

CONTRIBUTORS
Michael Biddick mbiddick@nwc.com
Michael A. Davis mdavis@nwc.com
Jonathan Feldman jfeldman@nwc.com
Randy George rgeorge@nwc.com
Michael Healey mhealey@nwc.com

EDITORS
Jim Donahue Chief Copy Editor
jdonahue@techweb.com

ART/DESIGN
Mary Ellen Forte Senior Art Director
mforte@techweb.com

Sek Leung Senior Designer
sleung@techweb.com

INFORMATIONWEEK ANALYTICS
analytics.informationweek.com

Art Wittmann VP and Director
awittmann@techweb.com 408-416-3227

Lorna Garey Executive Editor, Analytics
lgarey@techweb.com 978-694-1681

Heather Vallis Managing Editor, Research
hvallis@techweb.com 508-416-1101

INFORMATIONWEEK.COM
Benjamin Tomkins Managing Editor
btomkins@techweb.com 516-562-5336

Roma Nowak Senior Director,
Online Operations and Production
rnowak@techweb.com 516-562-5274

Tom LaSusa Managing Editor,
Newsletters
tlasusa@techweb.com

Jeanette Hafke Web Production Manager
jhafke@techweb.com

Joy Culbertson Web Producer
jculbertson@techweb.com

Nevin Berger Senior Director,
User Experience
nberger@techweb.com

Steve Gilliard Senior Director,
Web Development
sgilliard@techweb.com

UBM TECHWEB
Tony L. Uphoff CEO

John Dennehy CFO

David Michael CIO

Bob Evans Sr. VP
and Global CIO Director

Joseph Braue Sr. VP,
Light Reading
Communications Network

Scott Vaughan CMO

Ed Grossman Executive Vice President, InformationWeek Business Technology Network

John Ecke VP and Group Publisher, Financial Technology Network, InformationWeek Government, and InformationWeek Healthcare

Martha Schwartz EVP, Group Sales, InformationWeek Business Technology Network

Beth Rivera Senior VP,
Human Resources

David Berlind
Chief Content Officer, TechWeb, and Editor in Chief, TechWeb.com

Fritz Nelson VP and Editorial Director, InformationWeek Business Technology Network, and Executive Producer, TechWeb TV

UNITED BUSINESS MEDIA LLC
Pat Nohilly Sr. VP, Strategic Development
and Business Administration

Marie Myers Sr. VP,
Manufacturing

INFORMATIONWEEK VIDEO
informationweek.com/tv

Fritz Nelson Executive Producer
fnelson@techweb.com

INFORMATIONWEEK BUSINESS TECHNOLOGY NETWORK

DarkReading.com
Security
Tim Wilson, Site Editor
wilson@darkreading.com

IntelligentEnterprise.com
App Architecture
Doug Henschen,
Editor in Chief
dhenschen@techweb.com

NetworkComputing.com
Networking , Communications, and Storage
Mike Fratto, Site Editor
mfratto@techweb.com

PlugIntoTheCloud.com
Cloud Computing
John Foley, Site Editor
jpfoley@techweb.com

InformationWeek SMB
Technology for Small and Midsize Business
Benjamin Tomkins,
Site Editor
btomkins@techweb.com

Dr. Dobb's
The World of Software Development
Andrew Binstock
Executive Editor
alb@drdobbs.com

Previous Next

IN THIS ISSUE

Dr.Dobb's Business Contacts

DR. DOBB'S

Sales Director, Michele Hurabiell
(415) 378-3540, mhurabiell@techweb.com

Account Executive, Shaina Guttman
(212) 600-3106, sguttman@techweb.com

INFORMATIONWEEK BUSINESS TECHNOLOGY NETWORK

CMO, Scott Vaughan
(949) 223-3662, svaughan@techweb.com

VP of Group Sales, InformationWeek Business Technology Network, Martha Schwartz
(212) 600-3015, mschwartz@techweb.com

Sales Assistant, Group Sales, Kelly Glass
(212) 600-3327, kglass@techweb.com

Publisher's Assistant, Esther Rodriguez
(949) 223-3656, erodriguez@techweb.com

SALES CONTACTS—WEST

Western U.S. (Pacific and Mountain states) and Western Canada (British Columbia, Alberta)

Inside Sales Manager, Vesna Beso
(415) 947-6104, vbeso@techweb.com

Sales Assistant, Ian Doyle
(415) 947-6105, idoyle@techweb.com

Strategic Accounts

Account Director, Sandra Kupiec
(415) 947-6922, skupiec@techweb.com

Account Manager, Shoshana Freisinger
(415) 947-6349, sfreisinger@techweb.com

Sales Assistant, Matthew Cohen-Meyer
(415) 947-6214, mmeyer@techweb.com

SALES CONTACTS—EAST

Midwest, South, Northeast U.S. and Eastern Canada (Saskatchewan, Ontario, Quebec, New Brunswick)

District Manager, Jenny Hanna
(516) 562-5116, jhanna@techweb.com

District Manager, Michael Greenhut
(516) 562-5044, mgreenhut@techweb.com

Account Manager, Cori Gordon
(516) 562-5181, cgordon@techweb.com

Inside Sales Manager East, Ray Capitelli
(212) 600-3045, rcapitelli@techweb.com

Sales Assistant, Elyse Cowen
(212) 600-3051, ecowen@techweb.com

Strategic Accounts

District Manager, Mary Hyland
(516) 562-5120, mhyland@techweb.com

Account Manager, Tara Bradeen
(212) 600-3387, tbradeen@techweb.com

Account Manager, Jennifer Gambino
(516) 562-5651, jgambino@techweb.com

Sales Assistant, Kathleen Jurina
(212) 600-3170, kjurina@techweb.com

Dr. Dobb's

Sales Director, Michele Hurabiell
(415) 378-3540, mhurabiell@techweb.com

Account Executive, Shaina Guttman
(212) 600 3106, sguttman@techweb.com

MARKETING

VP, Marketing, Winnie Ng-Schuchman
(631) 406-6507, wng@techweb.com

Marketing Manager, Monique Luttrell
(949) 223-3609, mluttrell@techweb.com

AUDIENCE DEVELOPMENT

Director, Karen McAleer
(516) 562-7833, kmcaleer@techweb.com

BUSINESS OFFICE

General Manager, Marian Dujmovits

United Business Media LLC
600 Community Drive
Manhasset, N.Y. 11030 (516) 562-5000

UBM TECHWEB

Tony L. Uphoff CEO

John Dennehy CFO

David Michael CIO

Bob Evans Sr. VP and Global CIO Director

Joseph Braue Sr. VP, Light Reading Communications Network

Scott Vaughan CMO

Ed Grossman Executive Vice President, InformationWeek Business Technology Network

John Ecke VP and Group Publisher, Financial Technology Network, InformationWeek Government, InformationWeek Healthcare

Martha Schwartz VP, Group Sales, InformationWeek Business Technology Network

Beth Rivera Senior VP, Human Resources

David Berlind Chief Content Officer, TechWeb, and Editor in Chief, TechWeb.com

Fritz Nelson VP, Editorial Director, InformationWeek Business Technology Network, and Executive Producer, TechWeb TV

UNITED BUSINESS MEDIA LLC

Pat Nohilly Sr. VP, Strategic Development and Business Admin.

Marie Myers Sr. VP, Manufacturing

UBM TechWeb