TechJournal

New York Oracle Users Group

Second Quarter 2013

Second Quarter General Meeting

Wednesday, June 5, 2013
St. John's University – Manhattan Campus
101 Murray Street

Free for Paid 2013 Members
Don't Miss It!

In This Issue –

Nothing hunts down Oracle performance issues like Confio Ignite™.

Over 50% of DBAs who try Ignite resolve a performance problem on the first day.

Start your **free trial at Confio.com/p-hog**

CONFIO SOFTWARE

NYOUG Officers / Chairpersons

ELECTED OFFICERS - 2012

President
Michael Olin
president@nyoug.org

Vice President
Mike La Magna
vicepresident@nyoug.org

Executive Director
Caryl Lee Fisher
execdir@nyoug.org

Treasurer
Robert Edwards
treasurer@nyoug.org

Secretary
Cathy Wang-Wender
secretary@nyoug.org

CHAIRPERSONS

Chairperson / WebMaster
Thomas Petite
info@nyoug.org

Chairperson / Technical Journal Editor
Melanie Caffrey
editor@nyoug.org

Chairperson / Member Services
Robert Edwards
membership@nyoug.org

Chairperson / Speaker Coordinator
Caryl Lee Fisher
speakers@nyoug.org

Chairperson / Vendor Relations
Caryl Lee Fisher
vendorcoordinator@nyoug.org

Chairperson / DBA SIG
Simay Alpoge
dbasig@nyoug.org

Chairperson / Data Warehousing SIG
Vikas Sawhney
dwsig@nyoug.org

Chairperson / Web SIG
Coleman Leviter
websig@nyoug.org

Chairperson / Long Island SIG
Simay Alpoge
lisig@nyoug.org

Director / Strategic Planning
Carl Esposito
planning@nyoug.org

CHAIRPERSON / VENUE COORDINATOR

Michael Medved
venuecoordinator@nyoug.org

EDITORS – TECH JOURNAL

Associate Editor
Jonathan F. Miller
jonathanfmiller@earthlink.net

Contributing Editor
Arup Nanda - DBA Corner

Contributing Editor
Jeff Bernknopf - Developers Corner

ORACLE LIAISON

Kim Marie Mancusi
Kim.Marie.Mancusi@oracle.com

PRESIDENTS EMERITUS OF NYOUG

Founder / President Emeritus
Moshe Tamir

President Emeritus
Tony Ziemba

Chairman / President Emeritus
Carl Esposito
cesposi@bers.nyc.gov

President Emeritus
Dr. Paul Dorsey

Table of Contents

New York Oracle Users Group, Inc.
#0208
67 Wall Street, 22nd floor
New York, NY 10005-3198
(212) 978-8890

Summer General Meeting – June 5, 2013

AGENDA

Time	Activity	Track/Room	Presenter
8:30-9:00	**REGISTRATION AND BREAKFAST**		
9:00-9:30	**Opening Remarks** **General Information**	(single session) Auditorium	Michael Olin NYOUG President
SESSION 1 9:30-10:30	**KEYNOTE: The Best Oracle Database 12c New Features – Part 1**	(single session) Auditorium	Rich Niemiec Rolta
10:30-10:45	**BREAK**		
SESSION 2 10:45 -11:30	**The Best Oracle Database 12c** **New Features – Part 2**	DBA Auditorium (single session)	Rich Niemiec Rolta
SESSION 3 11:30 -12:30	**Ask the Experts Panel**	(single session) Auditorium	Michael Olin Moderator
12:30 -1:30	**LUNCH - ROOM 123**		
SESSION 4 1:30-2:30	**Big Data Overview and** **Oracle's Big Data Solution**	DBA Auditorium	Rich Niemiec Rolta
	Bring Your iPads **(Because You're Gonna Build a Mobile Apex App in One Hour)**	Developer Room 118	Chris Ostrowski Avout
2:30-2:45	**BREAK**		
SESSION 5 2:45-3:45	**Top 10 Lessons Learned** **Implementing Exadata**	DBA Auditorium	Gary Bhandarkar & Mike LaMagna Merck
	Creating Custom PDF reports **with APEX 4.2.2**	Developer Room 118	Marc Sewtz Oracle Corporation
SESSION 6 3:45-4:45	**Bridging the Gap Between Privacy** **and Data Insight**	DBA Auditorium	Ulf Mattsson Protegrity
	Tagging, Encoding and Encrypting **with RMAN**	Developer Room 118	Anthony Noriega ADN/IBM

ABSTRACTS

9:30-10:30 AM KEYNOTE: The Best Oracle Database 12c New Features – Part 1

This presentation will discuss which Oracle 12c new features should be investigated for use. Most of the features covered will be related to the DBA, but there will also be a few outside that realm that focus on the developer. Simple examples (such as a quick example using pluggable databases) will be included to show the basic functionality of the new features, including:

- Invisible columns
- Multiple indexes on the same column
- Adaptive Execution Plans
- Runaway query management
- Change Table Compression at import time
- Creating views as tables
- Online Move Partition
- Partial Indexes for partitioned tables
- Pluggable databases
- Enhanced DDL Online
- Automatic Diagnostics Repository
- Enhanced Security Features

Rich Niemiec is a world renowned Oracle Expert. He is an Oracle Ace Director and was a co-founder and CEO of TUSC, a Chicago-based systems integrator of Oracle-based business solutions founded in 1988. Rich currently serves as an Executive Advisor to the Rolta International Board of Directors and has served as President of Rolta TUSC and Rolta EICT. TUSC was the Oracle Partner of the Year in 2002, 2004, 2007, 2008, 2010 (Rolta TUSC), & 2011 (Rolta). Rolta is an international market leader in IT-based geospatial solutions, and caters to industries as diverse as infrastructure, telecom, electric, airports, defense, homeland security, urban development, town planning and environmental protection. Rich is the past President of the International Oracle Users Group (IOUG) and the current President of the Midwest Oracle Users Group (MOUG). Rich won IOUG's Chris Wooldridge Award in 2012. Rich is one of six originally honored worldwide Oracle Certified Masters. In 2012, he authored the #1 Oracle bestseller, *Oracle11g Release 2 Performance Tuning Tips & Techniques*, and an update of his previous three Oracle best sellers on Oracle8i, Oracle9i, and Oracle10g Performance Tuning. Rich was inducted into the Entrepreneurship Hall of Fame in 1998.

10:45-11:30 AM DBA TRACK: The Best Oracle Database 12c New Features – Part 2

See presentation abstract and bio for Rich Niemiec above.

1:30-2:30 PM DBA Track: Big Data Overview and Oracle's Big Data Solution

This presentation will review many of the current Big Data Solutions and how Oracle's Big Data Solution is substantially better. Rich will review the role that both Google and Facebook have played in the Big Data Ecosystem and how other NoSQL databases like Cassandra and MongoDB stack up against the Oracle Solution. Topics to be covered include: Big Data Overview and How Much Data There Is in the World

- GFS and MapReduce early days
- Hadoop, HDFS, and MapReduce in Open Source
- A couple of the NoSQL Databases
- The Oracle Solution - Why Oracle should win this market
- Oracle Hadoop Data Loader and BerkeleyDB

See bio for Rich Niemiec above.

1:30-2:30 PM DEVELOPER TRACK: Bring Your Ipads (Because You're Gonna Build a Mobile APEX App in 1 Hour)

Building mobile applications has become a very hot topic in the IT world. Users are increasingly demanding access to corporate applications via their mobile devices. BYOD (Bring Your Own Device) articles have been increasingly featured in technology publications. New languages, development tools and skill sets along with the decisions of what mobile platforms to support can make the entry into mobile application development very difficult for many organizations. A new technology, however, makes decisions about mobile development much easier: JQuery Mobile. Oracle Application Express integrates very easily with JQuery Mobile and this combination can be used to create mobile applications that are supported on a wide range of mobile devices relatively simple. This presentation will walk attendees through the steps of building a mobile application using Oracle Application Express. Attendees are encouraged to bring an iPad (or other mobile device) to actively participate and see the final results of their work.

Chris Ostrowski is an Oracle Solution Architect Director for Avout in Colorado. He has worked with Oracle technologies for over 20 years as a Developer, DBA, Project Manager and Enterprise Architect. Recently, Chris has focused his efforts on Service Oriented Architecture technologies including Oracle JDeveloper, the Oracle SOA Suite and Enterprise Technologies including Oracle Fusion Applications and Oracle's Application Integration Architecture. He is the author of three books from Oracle Press, "Oracle Application Server 10g Web Development", "The Oracle Application Server Portal Handbook" and the upcoming "Migrating to Fusion Applications", is a certified Oracle SOA Implementation Champion and is a proud member of the Oracle ACE Program.

2:45-3:45 PM DBA TRACK: Top 10 Lessons Learned Implementing Exadata

After migrating many applications to Exadata, Mike and Gary will share their top 10 lessons learned when migrating applications to Exadata. Gary and Mike have over 6 years of experience on Oracle's Exadata Platform and more than a half a century of Oracle experience. This presentation will review the items you need to plan for when migrating an application to Exadata. Attendees will also learn what to watch out for to help ensure a successful migration to Exadata, thereby maximizing their investment in the Exadata platform.

Gary Bhandarkar has worked as a DBA and developer for the past 18 years and has worked with Exadata since 2010.

Mike La Magna, Vice President of NYOUG, is an Associate Director of Database Services at Merck and has been a developer and DBA working with Oracle since the early 1980's.

2:45-3:45 PM DEVELOPER TRACK: Creating Custom PDF reports with Oracle APEX 4.2.2

Oracle Application Express (APEX) provides several reporting options, allowing developers and users to quickly generate and view reports on data stored in the Oracle database. Users export their report data in PDF format, provided they have an external print-rendering engine configured with their APEX instance. With Oracle Application Express 4.2.2, the configuration has been simplified by building the PDF rendering functionality right into the APEX Listener. In this session we will show you how to configure PDF printing and demonstrate how you can create PDF reports with fully customized report layouts using standard XSL-FO layout tools, like Altova Stylevision and Stylus Studio.

Marc Sewtz is a Senior Software Development Manager at Oracle Corporation in New York. With over 16 years of industry experience, Marc held roles in Consulting, Sales and Product Development and now manages a global team of Software Developers and Product Managers in the Oracle APEX development group, part of Oracle Database Tools. Marc and his team are responsible for features such as the development of Mobile Web Applications with

APEX, Reporting and Charting, Tabular Forms, Oracle Forms to APEX conversion and integration with Oracle Business Intelligence Publisher. Marc has a Master's degree in computer science from the University of Applied Sciences in Wedel, Germany.

3:45-4:45 PM DBA TRACK: Bridging the Gap between Privacy and Data Insight Big Data

Learn how to bridge the gap between security regulations, privacy and compliance, yet still be able to provide powerful analysis and data insight to achieve the power behind a big data environment. Learn how to adapt to the ever-changing data security landscape, including compliance to PCI DSS, HIPAA/HITECH and US State laws.

Ulf Mattsson created the innovative architecture of the Protegrity Data Security Platform. He is commonly considered one of the founding fathers of tokenization and has been advising the industry's top analysts and stakeholders including PCI Security Standards Council, ISACA and Visa as they navigate the role of tokenization in payments security. Ulf is the inventor of more than 20 patents in the areas of encryption key management, policy driven data encryption, internal threat protection, data usage control and intrusion prevention. He also is a research member of the International Federation for Information Processing (IFIP) WG 11.3 Data and Application Security, ANSI X9, Information Systems Security Association (ISSA) and Information Systems Audit and Control Association (ISACA).

3:45-4:45 PM DEVELOPER TRACK: Tagging, Encoding, and Encrypting with RMAN

Tagging, encoding, and encrypting with RMAN allows Oracle DBAs not only to secure database backups, but also to customize them in order to provide an intelligent repository, with the ability to attain faster restores, comprehensive recovery, and optimizing throughput for all RMAN operations, while guaranteeing the privacy and confidentiality of the backup contents. This session will reveal top techniques to encode, tag and encrypt RMAN backups with a comparative benchmark on Linux, Windows, and Solaris systems through custom RMAN scripts supplied. The session will also demonstrate how tagging and encoding can optimize channel usage and minimize the Mean Time to Recover (MTTR). This session covers releases Oracle10g through Oracle12c.

Anthony D. Noriega is a Senior IT Consultant and computer scientist who has focused his efforts in Oracle database technology, network computing, software and network engineering, and object-oriented programming paradigms. Anthony spends most of his time as a database analyst, architect, and developer, and DBA. Anthony has earned an MBA from Montclair State University (2006), and MS in Computer Science from NJIT (1992 doctoral candidate through 1997) and a BS in Systems Engineering from University of the North (1987). He is an Oracle Certified Professional and Senior Oracle DBA/Developer and a member of IOUG, NJOUG, and NYOUG Oracle groups.

Message from the President's Desk

Michael Olin

Summer, 2013

Waiting for Oracle 12c

At our Summer General Meeting in early June, NYOUG members were treated to their second presentation about the forthcoming major release of the Oracle RDBMS. At our December 2012 meeting, Oracle's Charlie Garry first introduced us to the concept of "Pluggable Databases". In June, during his "Oracle Database 12c - New Features" keynote, Rich Niemiec covered as many topics as the Oracle Product and Legal teams would let him ("If it's on the slides, I can talk about it"), including Pluggable Databases. Rich also spent quite a bit of time speaking about the vast amounts of data that could be addressed, not only by Oracle 12c in particular, but by a 64-bit address space generally, and, anticipating the next logical development, a 128-bit address space. Rich went on to reference the work of inventor/entrepreneur/futurist Ray Kurzwiel, including a short video clip from the 2009 documentary "Transcendent Man" (http://www.amazon.com/movies-tv/dp/B0051Y6NUQ). I was happy to see Rich reference Kurzweil and his work. I have been following Kurzweil for almost 20 years, ever since he gave a keynote speech at the 1993 International Oracle Users Week conference in Orlando.

"The Singularity is Near"

Kurzweil's talk was entitled "The Sixty-Four Squares of the Chessboard". IOUW was not the only venue where he delivered this address, and I was able to find a transcript online in the January 1994 issue of "The Braille Monitor" (https://nfb.org/Images/nfb/Publications/bm/bm94/brlm9401.htm#2). Kurzweil begins by talking about the ability of information to transform society and explains how the increasing ability to leverage and increase the density of that information becomes evolutionary and leads to what he called in his 1992 book "The Age of Intelligent Machines" (http://www.amazon.com/Age-Intelligent-Machines-Ray-Kurzweil/dp/0262610795/).

A brief digression about inventing follows, with helpful hints regarding the process that had served Kurzweil well in several of his ventures. This process included the following steps:

- Designing the product brochure
- Sharing the brochure with potential customers
- Having those customers actually design and test the product

This struck many of the Oracle professionals in the audience as revolutionary. Kurzweil then segues into a discussion of Moore's Law and the corresponding graph that Rich showed in the clip from "Transcendent Man", relaying one version of the tale that is referenced in the title of his talk:

> The emperor of China is so impressed by the game of chess that he offers the inventor anything he would ask for as a reward. The inventor asks for a single grain of rice on the first square of the chessboard. When asked by the emperor if that was all he desired, the inventor then asked for two grains of rice on the second square, four on the third, eight on the fourth, and so on, leading to eighteen million trillion grains of rice in total.

Kurzweil's talk continues, using the grains of rice on the chessboard as a metaphor for the advances in computing power that we have already seen due to Moore's Law (about half of the chessboard), and he then speculates as to where things will go as we navigate the second half.

This is where Kurzweil begins to discuss the merging of biology and technology and things really start to get interesting. In 2040, he posits, "In accordance with Moore's law, your state-of-the-art personal computer will be able to simulate a

society of 10,000 human brains, each of which would be operating at a speed 10,000 times faster than a human brain." Rather than focus on Kurzweil's vision of the near-future, I'll simply recommend two of his subsequent books, "The Age of Spiritual Machines" (www.amazon.com/The-Age-Spiritual-Machines-Intelligence/dp/0140282025/) and "The Singularity is Near" (www.amazon.com/Singularity-Near-Humans-Transcend-Biology/dp/0143037889/), and navigate back to Oracle 12c.

Capacity is not Destiny

I think that where Kurzweil gets a bit ahead of himself is in assuming that simply having the capacity alone to compute on a scale that approaches or surpasses the human brain implies that evolutionary change will occur. While I have no doubts that this capacity will certainly facilitate great advances in artificial intelligence, we still need some people who can get the programming details right. We don't get from today to Star Trek just because we have the capacity to do nano-scale computing. The same can be said for the "Big Data" revolution that is expected to be unleashed with the availability of software like Oracle 12c. Now that the existence of the NSA's vast store of phone call metadata has been made public, it is only a matter of time before someone, with 20/20 hindsight, discovers that the information needed to discern that the Boston Marathon bombers were planning their attack has been sitting in a government database all along. The point is that even though the data may have been collected, the conclusion was not inevitable. Who knows what insight can be gleaned from a review of billions of Google searches, Facebook status updates, tweets on Twitter or YouTube videos? We have the capacity to store all of this information now. Perhaps with software such as Oracle 12c, we will have a practical way to wade through all of that data and start to make some important inferences. However, I am convinced that it is going to take more than just the available computing capacity to get to evolutionary change. It is going to take visionaries. People like Ray Kurzweil and, as Rich suggested in his keynote, Larry Ellison, who (along with many others) have imagined what the exponential growth of computing power promised by Moore's Law can lead to. I'm looking forward to hearing about the next generation of visionaries who figure out how to harness this capacity for something more transformative than internet search or movie recommendations.

Michael Olin
President, NYOUG

A Hitchhiker's Guide Integrating Oracle XML DB 11gR2 and SQL Developer 3.2.2

Coleman Leviter, Arrow Electronics

Preface

For the past several years, we have been involved in a wide range of development projects using Oracle's XML DB. The types of projects include integrating a Transportation Management System[1] (TMS), transmitting an XML Manifest Document using a B2B[2] portal, and finally, communicating with Oracle's E-Business Suite[3] (EBS). The focal point of the aforementioned three projects is a Warehouse Management System[4] (WMS).

In spite of the diversity of the three projects, all are in production and for the most part, perform flawlessly. With any new technology stack, at times issues arise and alternate solutions must be developed or you find that you are unable to proceed. Consequently, the project falls behind schedule.

Some examples of issues and their resolution:

- We encountered an issue developing XML Document communications to EBS. Our Oracle XML DB technology stack was based upon Oracle 10.2.0.1.0. The EBS developers were using Oracle (on 10.2.0.4) XQuery and requested that we (WMS) sync up with their technology stack. We ran into issues (ORA-19114: error during parsing the XQuery expression) using XQuery. Consequently, we abandoned XQuery and used XPath for XML Document shredding. EBS continues to use 10.2.0.4. There are no current issues with communications.
- A bug was encountered using the DBMS_XMLGEN.CONVERT[5]. DBMS_XMLGEN.CONVERT is used to convert an escaped version to an unescaped version (or visa versa). As an example, the escaped form of the character ">" (without the " characters) is ">" (without the " characters). Oracle's suggested "workaround" was to 1) append a space (CHR(32) to the end of the XPath string, then 2) DBMS_XMLGEN.CONVERT the string, finally 3) TRIM the string. The workaround resolved the issue.

With the projects mentioned above as a starting point, we would like to delve into the following XML topics: XML basics, XML document construction, XML message efficiency vs. Binary data messages, XSD[6] development, Warehouse Management System (WMS) communications to Oracle's Service Oriented architecture (SOA), XML namespace and finally dbms_xmldom.

Introduction

XML DB was introduced in Oracle 8i. In Oracle 11g, XML DB reached a new maturity level, providing high-performance with native XML storage and retrieval technology. The W3C[7] XML data model is fully immersed into the Oracle Database.

How is XML DB used in a project? What must one know when using XML DB? In this discussion, we will address these issues and the design criteria one might consider when using XML in a project.

XML or Extensible Markup Language means the language can grow as required. You may include an XML declaration line for the first line (<?xml version="1.0"?>). You may define your own elements or tag fields in the body of an XML document (<tagfield>data</tagfield>). As long as the sender and receiver agree on the format of an XML Document, there is complete flexibility for its construct.

Throughout this paper, we will present several examples of XML constructs on XML documents. We hope the readers will become familiar with XML by reviewing the examples, modifying them and using them in their own database, whether that is 9i, 10g or 11g. Some examples demonstrate different capabilities of SQLX (SQL XML). The WEB contains a great amount of XML material. For those beginning an XML project, this establishes a good starting point. Finally, the last section presents the reader with an example using SQLX (XML Document creation) and XML Document shredding (XML Xpath). Many examples demonstrate the use of XML Documents with namespace. The examples are

presented using table reference as well as in line code, which the reader may simply copy and paste in their own environment and view the results. The examples are constructed so that the reader may use them in their own Oracle database.

Project Overview

The current method of communications between the mainframe computer and the WMS are fixed length messages. Recently, the company embarked on a project retiring the mainframe computer and replacing it with Oracle's eBusiness suite (EBS). The inter-computer method of communications is XML. Therefore, the project uses Oracle 10g XML for communications between the WMS and Oracle's EBS. The WMS' pathway into eBusiness Suite is through the Service Oriented Architecture (SOA).

Using XML Messaging between EBS and WMS will accommodate a changing business model. Changing business rules may easily be incorporated into XML Messaging.

Presented is the WMS-EBS/SOA communications data flow:

WMS Appl. Step 1)

1. WMS Appl. issues an XML Request Message (XML Document) to MQ Write Queue
2. MQ Responds with a Successful Status indicating guaranteed XML Message delivery
3. EBS/SOA receives XML Request Message and processes the message

EBS/SOA Step 1)

1. EBS/SOA issues an XML Request Msg (XML Document) to MQ Read Queue
2. WMS Appl. reads XML Request Msg (XML Document) from MQ Read Queue
3. WMS Appl. Responds to MQ with Successful MQ Status indicating receipt of XML Request Msg

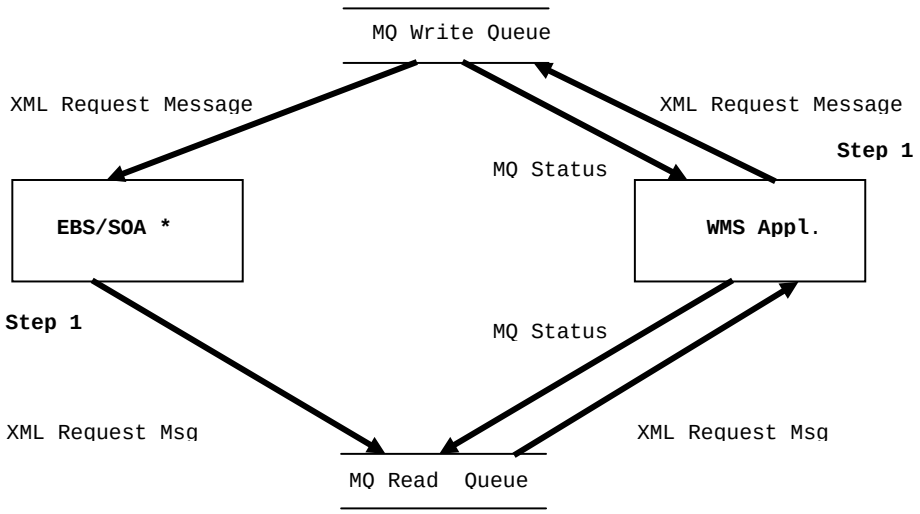


Figure 1: EBS/SOA - WMSAppl. XML Document Data Flow

* Note: SOA is not used here. It has been replaced by DTI or Direct Memory Interface

Message Comparison

Why use XML for messaging? To answer that question, let us explore an alternate method of data transfer.
One predominant method of transferring messages between two computers (or processes) is Binary Messaging. This method continues to prevail with many legacy computers and embedded firmware systems.
Data may be compressed, so encoding and reassembly must all work in concert. If a transmitted data steam contains 256 bits, the receiving side must also be in alignment and decode those same 256 bits. All data is position dependent.
When implementing this method, maintenance costs may run very high because of the volume of software that must be managed on the sender and receiver sides. Troubleshooting adds to the cost as well.
Describing a system's efficiency is the ratio of output to input. Therefore, the efficiency of messages between two computers is shown as:

$$\text{Efficiency (\%)} = \frac{\text{Output}}{\text{Input}} \times 100 \quad \text{or} \quad \frac{\text{Message Data}}{\text{Message Data + Overhead Data}} \times 100$$

Sales Complete Message (SCP)

Field Name	Field Type	Size	Bit Position
Overhead Data	Alphanumeric	32	1 - 32
Message Code	'SCP* '	4	33-36
Entering Location	Alphanumeric	3	37-39
Sales Number	Alphanumeric	6	40-45
Version Number	Alphanumeric	2	46-47
Cartons for Shipping	Numeric	2	48-49
Shipping Charges	Alphanumeric	9/2	50-58
Date (YYMMDD)	Alphanumeric	6	59-64
Carrier	Alphanumeric	16	65-80
End	Alphanumeric	27	81-107

Table 1: Typical Binary Interface Data Layout

Using the sample binary data layout from Table I we have:

$$\frac{\text{75 bits (pos 33- 107)}}{\text{(Message data) 75 bits + 32 bits (overhead data)}} \times 100 = 70\% \text{ efficient}$$

Let us look at a simple XML Document using namespace (437 bytes):

```
<ns1:object_group xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"
xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
  <obj1:object>
    <obj1:thing>ball</obj1:thing>
    <obj1:thing>key</obj1:thing>
    <obj1:thing>table</obj1:thing>
  </obj1:object> ^-----------------------  typical data
  <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
    <obj2:thing>frisbee</obj2:thing>
```

```
    <obj2:thing>bbq</obj2:thing>
    <obj2:thing>switch</obj2:thing>
  </obj2:object>
</ns1:object_group>
```

Figure 2: Simple XML Message

Here, the meaningful data consists of “ball”, “key”, “table”, “frisbee”, “bbq” and “switch” or 28 bytes. The overhead data is everything else or 437 bytes - 28 bytes = 409 bytes of metadata. (We address a bit more on metadata later on.) Using the Efficiency Formula from above we have:

$$\frac{\text{28 Bytes Message Data (Figure 3)}}{\text{28 Bytes Message Data + 409 Bytes XML Element Definition \& namespace (= 437 total bytes)}} \times 100$$

= 6 % efficient

Therefore, using an XML data message with namespace results in 6% efficiency, while using binary data transfer results in 70% efficiency. It is obvious that the messages are quite different, but an important observation is the overhead ratio required to send data. XML messages far exceed binary data messages, mainly due to the verbose nature or metadata of the element names and namespaces in the XML Document.
If the efficiency of an XML data message does not fare well compared to a binary data message, then why use XML data messaging? Here are arguments for and against:

XML Arguments for Usage:

- It is platform and system independent i.e. it can work on any computer.
- It allows us to define our own tags thus making your data content understood.
- XML has adopted a standard, ISO 10646 also known as Unicode, which is a framework to encode characters. It will support most languages, thus not forcing people to use English for coding.
- Software can be developed to increase efficiency, that is, encode the element tags on the transmission as well on the receiving side. This must be balanced between easily understood tags and tags that are hard to follow.
- The code is easy to understand even for those people who do not have any prior knowledge.
- XML DB has been available with Oracle 8i and up
- XML is self-describing. For example it is obvious from <lastname>Doe</lastname> that this represents a name.
- Bandwidth issues can become non existent using data compression techniques

XML Arguments against Usage:

- It requires a wide amount of bandwidth.
- It may require extensive processing time to decode. The host computers must be capable of processing the messages.
- Only newer software will be able to read and understand XML. It may become costly and time consuming to retrofit legacy code with XML

A good design decision rests with one’s ability to analyze a problem and choose the proper tools. The same applies when selecting XML or other data communications methods. In the previously described application, the new EBS/SOA system already used XML. Our system (WMS) used Oracle 10g in which XML DB was available. In our case, it was a perfect fit to use XML for data communications. Although the XML duty cycle is low, if the messaging frequency is low, XML is certainly a viable option. For example, if the application environment is limited to human interaction with a computer and waiting for an answer (i.e. credit card verification), XML inefficiencies would not appear to present a problem. If XML messages were used to guide a shuttlecraft into its docking station, perhaps too many messages would

rapidly use up the bandwidth. Another example where XML data transfer might not work: a central station (monitoring burglar alarms) receiving video transmissions of an alarm of a potential intruder. With samplings of several seconds and several frames of a possible intruder in the area, data transmission with compression may exceed 250 Kbytes. The real time video example may not be a candidate for XML unless binary compression techniques become available.
To summarize, many real time or mission critical applications may not be candidates for XML data communications. But there are many applications that are using XML for data communications. It is important to assess the project requirements and use the proper architecture. That will ensure project success.

XML Communications

Our message queuing system for XML communications is IBM's WebSphere MQ Message Queuing, which is in use with many enterprise applications.
When we view XML Messages on MQ, we use a tool called IBM Tivoli CandleNet Portal. Here is an extract of a sample message using the tool:

```
Hexadecimal Data                    Character Data
3C3F786D 6C207665 7273696F 6E3D2231 *<?xml version=1*"
2E302220 3F3E3C69 6D70313A 574D535F *.0 ?><imp1:WMS_*"
4D51456E 76656C6F 70652078 6D6C6E73 *MQEnvelope xmlns*
3A696D70 313D2268 7474703A 2F2F7777 *:imp1=http://ww*"
772E6172 726F772E 636F6D2F 574D535F *w.arrow.com/WMS_*
4D51456E 76656C6F 70655F76 315F305F *MQEnvelope_v1_0_*
3030223E 0A202020 3C696E70 313A574D *00>.   <inp1:WM*"
535F5072 6F636573 73496E76 656E746F *S_ProcessInvento*
72794D6F 76656D65 6E742078 6D6C6E73 *ryMovement xmlns*
3A696E70 313D2268 7474703A 2F2F7777 *:inp1=http://ww*"
772E6172 726F772E 636F6D2F 574D532F *w.arrow.com/WMS/*
50726F63 65737349 6E76656E 746F7279 *ProcessInventory*
4D6F7665 6D656E74 5F76315F 305F3030 *Movement_v1_0_00*
223E0A20 20202020 203C6E73 313A5374 *>.      <ns1:St*"
616E6461 72644865 61646572 20786D6C *andardHeader xml*
6E733A6E 73313D22 68747470 3A2F2F77 *ns:ns1=http://w*"
77772E61 72726F77 2E636F6D 2F574D53 *ww.arrow.com/WMS*
2F537461 6E646172 64486561 6465725F */StandardHeader_*
76315F30 5F303022 3E0A2020 20202020 *v1_0_00>.      *"
```

Figure 3: Partial XML Document Viewed Using IBM Tivoli CandleNet Portal

The left side of Figure 3 shows the XML message in hexadecimal; the right side shows the XML message. The asterisks are not part of the data. Viewing XML data in this manner aids in isolating problems when the XML message is on its way to the receiver or arriving from sender. Additionally, observe that the namespace definition is part of the payload.

Background: XML Schema Definition (XSD)

An enterprise project may begin with an XML Schema Definition or XSD. The XSD can be used to express a set of rules to which an XML document must conform in order to be considered 'valid' according to that schema. There are several graphical user interface (GUI) tools on the market that allow one to design an XSD. Displayed in Figure 4 below is an example of an XSD using a GUI tool. The schema is designed as well as the XML Document data types.

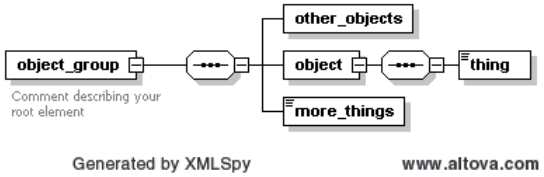


Figure 4: XSD Example

After the XSD is designed, one may generate the associated XML that describes the XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2008 rel. 2 sp1 (http://www.altova.com)  -->
<xs:schema xmlns=" =" ="http://www.w3.org/2001/XMLSchema/sample_namespace_1 "
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xs:element name="object_group">
            <xs:annotation>
                  <xs:documentation>Comment describing your root
element</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="other_objects">
                              <xs:complexType/>
                        </xs:element>
                        <xs:element name="object">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="thing"/>
                                    </xs:sequence>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="more_things"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema
```

Using the above XML Schema Definition, one may use it as a template to generate the XML Document. Note the “ns:” or namespace notion. We will present several examples using this notation throughout this paper.

XML Document Construction

Properly formed XML documents contain (except where noted) the following components:
An optional declaration as the first line (generally, it is good practice to include the declaration, but it is not mandatory):

```
<?xml version="1.0" encoding="UTF-8"?>
```

where the mandatory first attribute identifies the XML version number and the optional second attribute is the encoding attribute which specifies to the XML parser what character encoding the text is in for translation into Unicode (Unicode is an industry standard allowing computers to consistently represent and manipulate text expressed in any of the world's writing systems).
An optional comments section:

```
 <!--this is a sample comment-->
```

A mandatory start tag and end tag as the root node:

START TAG

```
<ns1:object_group xmlns:ns1=http://www.w3.org/2001/XMLSchema/sample_namespace_1
                  xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
```

END TAG

```
</ns1:object_group>
```

And finally, a mandatory data element:

Data Element

```
<obj1:thing>ball</obj1:thing>
```

So, at a minimum, one root node (start tag and end tag) and one data element constitute a properly formed XML document.
Putting it all together we have:

```
<ns1:object_group xmlns:ns1=http://www.w3.org/2001/XMLSchema/sample_namespace_1
                  xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
      <obj1:thing>ball</obj1:thing>
</ns1:object_group>
```

A more meaningful XML Document follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:object_group xmlns:ns1=http://www.w3.org/2001/XMLSchema/sample_namespace_1
xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
  <!—- XML Document with namespace example-->
  <obj1:object>
    <obj1:thing>ball</obj1:thing>
    <obj1:thing>key</obj1:thing>
    <obj1:thing>table</obj1:thing>
  </obj1:object>
  <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
    <obj2:thing>frisbee</obj2:thing>
    <obj2:thing>bbq</obj2:thing>
    <obj2:thing>switch</obj2:thing>
  </obj2:object>
</ns1:object_group>
```

Shortly, we will explain the following syntax: “xmlns:ns1”, “xmlns:obj1”, “obj1”, “obj2”.
The XML document fits a hierarchical model as presented in Figure 5:

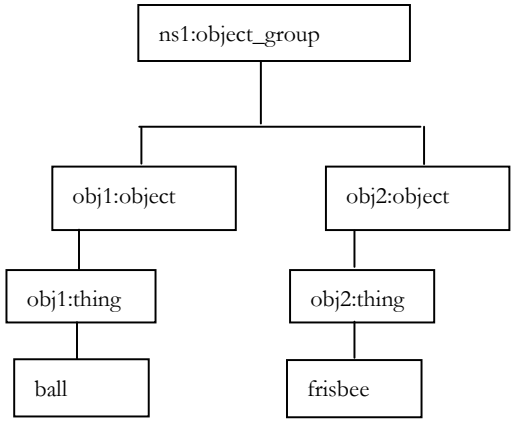


Figure 5: Hierarchical Model of an XML Document

where ns1:object_group is the root element and obj1:object and obj2:object represent the data sections. Note: data items "key", "table", "bbq" and "switch" have been omitted from Figure 5 but are part of obj1:object and obj2:object.

XMLTYPE Column

To create a sample table containing an XMLTYPE column, use the following CREATE TABLE DDL:

```
CREATE TABLE my_table (id number, xmlcol XMLTYPE, clobcol CLOB);
```

The underlying type, XMLTYPE is a CLOB, which enables storage of up to 4GB of data. Additionally, you may perform XPath queries on the XML Documents residing in the column. By simply defining the column (xmlcol) as a CLOB, XPath expression queries are not possible. When storing XML documents into an XMLTYPE column, Oracle will raise an exception if the XML document is not properly formed. If you want to store the improperly formed document for later evaluation, depending upon its length, it may be stored in a CLOB type column, which in the above sample table, is identified by "clobcol".
When using XMLTYPE columns in tables, it may be helpful to use the pragma AUTONOMOUS_TRANSACTION. When a subprogram is marked with this pragma, it is possible to perform rollbacks or commits without affecting operations in the parent transaction. Basically, this pragma works the same way as a sequence object.

XML Examples

In this section we will explore several XPath and SQLX (or SQL/XML) examples. SQLX is used for constructing XML documents. XPath is used for shredding or extracting data from the XML document. You may use the code in the following examples in your own Oracle environment. All my examples work in Oracle 10g Enterprise Edition Release 10.2.0.1.0. I cannot guarantee the results if you use an earlier Oracle edition, especially with namespaces. XPath or XML Path Language is a language for selecting parts (shredding) of an XML document and computing values (strings, numbers, or Boolean values) based on the content of an XML document. We will produce (construct) several XML fragments using functions XMLElement(), XMLAgg() and XMLForest(). We will demonstrate document shredding using the XMLSequence() function and the EXTRACT method. At the end of each respective section, the SQLX and XPath section, the reader will view other XML examples using namespace notion. To wrap it up, we will present two different types of SQLX and XPath examples: one that is using XML documents in-line, the other using Oracle table references. Finally, at the end, we provide one example using the Oracle function UPDATEXML() demonstrating an update to a single data element residing in an XMLTYPE column.

SQLX Document Construction

The following examples presented in this section, show how a few XML functions are used as building blocks to build complex XML documents.
Some of the examples reference the following relational table:

```
SQL>  select * from myobject;

THINGS       QUANTITY PARENT
------------ -------- ------
BALL         2        1
KEY          3        1
TABLE        1        1
FRISBEE      4        2
BBQ          1        2
SWITCH       6        2
```

Table 2: "myobject" Relational Table

XMLELEMENT()

The simplest SQLX Query uses the XMLElement() function, which returns an XMLTYPE expression (XML fragment):
First, we will view the results of the query using SQL:

```
SELECT  'JONES' EMPLOYEE FROM DUAL;

EMPLOYEE
---------------
JONES
```

Now, let us view the SQLX version of the query:

```
SELECT XMLELEMENT("Emp", 'jones')  employee FROM DUAL;

EMPLOYEE
----------------
<Emp>jones</Emp>
```

Oracle's SQLX provides the tag fields <Emp></Emp> for the query results. The "/" notation in the last tag field signifies the element terminator.

XMLForest()

This example, which uses XMLForest(), produces a non qualified XML document: (the reason that it is not qualified is that the result set is missing the parent node):
First, we will view the results of the query using SQL:

```
select  quantity, things from myobject;

QUANTITY THINGS
-------- --------------------
2        BALL
3        KEY
1        TABLE
4        FRISBEE
1        BBQ
6        SWITCH
```

Now, let us view the SQLX version of the query:

```
SELECT XMLFOREST(obj.quantity, obj.things) "Object_list"
FROM myobject obj;

Object_list
--------------------
<QUANTITY>2</QUANTITY><THINGS>BALL</THINGS>
<QUANTITY>3</QUANTITY><THINGS>KEY</THINGS>
<QUANTITY>1</QUANTITY><THINGS>TABLE</THINGS>
<QUANTITY>4</QUANTITY><THINGS>FRISBEE</THINGS>
<QUANTITY>1</QUANTITY><THINGS>BBQ</THINGS>
<QUANTITY>6</QUANTITY><THINGS>SWITCH</THINGS>
```

The XMLForest() function produces an XML fragment that contains a set of XML elements.

XMLAGG()

Let us view an example using the XMLAgg() function. First, we will view the results of the query using SQL:

```
select things, quantity from myobject;

THINGS               QUANTITY
-------------------- --------
BALL                 2
KEY                  3
TABLE                1
FRISBEE              4
BBQ                  1
SWITCH               6
```

The first example returns an ordered set using the XMLAgg() function:

```
SELECT XMLELEMENT("OBJECT", XMLAGG(XMLELEMENT("Things",obj.things ||' '||
obj.quantity )
ORDER BY obj.things)) AS "Object_list"
FROM myobject obj;

Object_list
--------------------
<OBJECT>
<Things>BALL 2</Things>
<Things>BBQ 1</Things>
<Things>FRISBEE 4</Things>
<Things>KEY 3</Things>
<Things>SWITCH 6</Things>
<Things>TABLE 1</Things>
</OBJECT>
```

The XMLAgg() function returns an XML fragment in an XMLTYPE by assembling XML fragments, with the option of XML element sorting. The XMLAgg() function assembles all the XML elements into one XML document fragment. The outer XMLElement() function, incorporates the XML document fragment into its "OBJECT" element (parent) as child elements. As a result of using the XMLELEMENT() function, we have essentially created a fully qualified XML Document, one that has a parent node and child nodes.

XMLELEMENT - Namespace

This is an SQLX Query (anonymous block) namespace example using XMLELEMENT(), XMLAGG(), XMLTYPE.getclobval() and XMLATTRIBUTES() . In our example, XMLATTRIBUTES() is used to define the namespaces associated with the elements.
(reference Table 2: "myobject" relational table)

```
DECLARE
lcl_obj1    CLOB;
lcl_obj2    CLOB;
lcl_full_xml   CLOB;
```

```
BEGIN
    SELECT XMLTYPE.getclobval(XMLELEMENT("obj1:object",
                      xmlagg(xmlelement("obj1:thing",obj.things) ))  )
    INTO lcl_obj1
    FROM  myobject obj
    WHERE obj.parent = '1';

    SELECT XMLTYPE.getclobval(XMLELEMENT("obj2:object",
           XMLATTRIBUTES ('http://www.w3.org/2001/XMLSchema/obj_2' AS "xmlns:obj2"),
           XMLAGG(xmlelement("obj2:thing",obj.things) ) ) )
    INTO lcl_obj2
    FROM  myobject obj
    WHERE obj.parent = '2';

    SELECT ( '<?xml version="1.0" encoding="UTF-8"?>' ||
          '<ns1:object_group
                 xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"
                 xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">'||
          lcl_obj1|| lcl_obj2|| '</ns1:object_group>' )
    INTO lcl_full_xml
    FROM dual;
    dbms_output.put_line(lcl_full_xml);
END;
```

The results are:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:object_group xmlns:ns1=http://www.w3.org/2001/XMLSchema/sample_namespace_1
                       xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
<obj1:object>
            <obj1:thing>ball</obj1:thing>
               <obj1:thing>key</obj1:thing>
                  <obj1:thing>table</obj1:thing>
</obj1:object>
<obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
<obj2:thing>frisbee</obj2:thing>
<obj2:thing>bbq</obj2:thing>
<obj2:thing>switch</obj2:thing>
</obj2:object>
</ns1:object_group>
```

Note: The above example demonstrates the flexibility of constructing an XML document. As in pl/sql, you may construct almost any type of query. In the above example, we have constructed an XML document using several different queries and finally constructing the XML document by concatenating the individual sections.

XPATH Document Shredding

XPath expressions are used to shred XML Documents. Shredding an XML document enables you to store data elements into a relational database.

XMLSEQUENCE()

The XMLSequence() function returns a collection of XMLTYPE. This function in a TABLE clause can be used to decompose the collection values into multiple rows. This can be further processed in a standard SQL query.

```
================================
Example: XML Document

<objects>
   <thing>ball</thing>
   <thing>key</thing>
   <thing>table</thing>
</objects>
-------------------------------------------------------------------------
XPath Query

SELECT  value(tab).extract('/*').getStringVal() "This Column"
FROM  table ( XMLSequence(extract ( XMLTYPE('
<objects>
   <thing>ball</thing>
   <thing>key</thing>
   <thing>table</thing>
</objects>
'),'/objects/*')  )   ) tab;
-------------------------------------------------------------------------
Result Set

This Column
--------------------
<thing>ball</thing>
<thing>key</thing>
<thing>table</thing>
```

The format clause ('/objects/*') indicates which child to return. Here, we request all the children under the 'object' parent. The "extract('/*')" clause in the SELECT statement requests everything from the "format" clause.

```
=======================================
Example:    XML Document Deux

<objects>
   <thing>ball</thing>
   <thing>key</thing>   □ extract this data item
   <thing>table</thing>
</objects>
=======================================
XPath Query

SELECT VALUE(tab).extract('/objects/thing[2]/text()').getStringVal() "This Column"
FROM   TABLE ( XMLSequence(extract (XMLTYPE('
<objects>
   <thing>ball</thing>
   <thing>key</thing>
   <thing>table</thing></objects>
'),'*')    )   ) tab;

Result Set
This Column
-----------------
key
```

The format clause ('*') indicates return everything. The “extract('/objects/thing[2]/text()')” method in the select statement requests return the data from the second node or ‘key’. The next example demonstrates, though the use of a cursor, how to traverse the nodes to obtain all the data items.

PUTTING IT ALL TOGETHER – Shredding Example with Cursor, No Table, Namespace

(Note: this is a self contained example. An Oracle table is not used. Namespace is introduced in this example, which is presented with the “xmlns:” notation. Except for the namespace notion denoted by the prefix on each element and the namespace definition, the focus of this example is the use of a cursor to traverse the nodes to obtain each data item from its respective element.
The namespace declaration uses the following syntax. xmlns:prefix="URI". “URI” or Uniform Resource Identifier can be any Internet resource or simply a string. Its purpose is to distinguish two elements with the same name. Namespace notation is optionally the last argument in the EXTRACT method:

```
extract (XMLTYPE_Instance>, <XPath_string>, <namespace_string>)

-- Anonymous Block
DECLARE
        -- Cursor for parsing object_group XML
        CURSOR obj_cur
        IS  SELECT EXTRACT (VALUE (entire_things),'//obj1:thing/text()',
                  'xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"').getstringval()
AS lcl_thing
        FROM   table ( XMLSequence(extract  (XMLTYPE('<?xml version="1.0" encoding="UTF-
8"?>
<ns1:object_group  xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"
    xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
<obj1:object>
<obj1:thing>ball</obj1:thing>
<obj1:thing>key</obj1:thing>
<obj1:thing>table</obj1:thing>
</obj1:object>
<obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
                        <obj2:thing>frisbee</obj2:thing>
<obj2:thing>bbq</obj2:thing>
                  <obj2:thing>switch</obj2:thing>
</obj2:object>

</ns1:object_group>'),'//obj1:thing','xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1"
' )  )  )
                               entire_things;
  BEGIN
      FOR obj_row IN obj_cur LOOP
dbms_output.put_line('each element thing  '|| obj_row.lcl_thing );
      END LOOP;
 END anonymous_block ;

Result Set
SQL> set serveroutput on
SQL> /
each element thing  ball
each element thing  key
each element thing  table
PL/SQL procedure successfully completed.
```

Note: Once the data items are “shred” from the XML Document, they can be stored into a relational table.

UPDATEXML() Example

We thought the reader might be interested in the following example. It does not fall within the realm of XML Document Shredding or XML Document Query. It is used to update a data element within an XML Document. We will demonstrate the Oracle function UPDATEXML. We will be "upper casing" the data item "frisbee".
Consider the following table:

```
SQL> describe my_xml_table;

 Name         Null?      Type
 ------------ --------   --------
 REF_ID                  NUMBER
 XMLCOL                  XMLTYPE

 select xmlcol from my_xml_table where ref_id = 1;

XMLCOL
--------------------------------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<ns1:object_group  xmlns:ns1=http://www.w3.org/2001/XMLSchema/sample_namespace_1
                xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
      <obj1:object>
            <obj1:thing>ball</obj1:thing>
            <obj1:thing>key</obj1:thing>
            <obj1:thing>table</obj1:thing>
      </obj1:object>
      <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
            <obj2:thing>frisbee</obj2:thing>
            <obj2:thing>bbq</obj2:thing>
            <obj2:thing>switch</obj2:thing>
      </obj2:object>
</ns1:object_group>


SQL> UPDATE my_xml_table mxt
  2  SET mxt.xmlcol = UPDATEXML(mxt.xmlcol,
  3  '//obj2:thing[1]/text()','FRISBEE',
     'xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2"')
  4  WHERE mxt.ref_id = 1;

1 row updated.

select xmlcol from my_xml_table where ref_id = 1;

XMLCOL
--------------------------------------------------------------------------------
<?xml version="1.0" encoding="UTF-8"?>
<ns1:object_group xmlns:ns1="http://www.w3.org/2001/XMLSchema/sample_namespace_1"
xmlns:obj1="http://www.w3.org/2001/XMLSchema/obj_1">
  <obj1:object>
    <obj1:thing>ball</obj1:thing>
    <obj1:thing>key</obj1:thing>
    <obj1:thing>table</obj1:thing>
  </obj1:object>
  <obj2:object xmlns:obj2="http://www.w3.org/2001/XMLSchema/obj_2">
    <obj2:thing>FRISBEE</obj2:thing>
```

```
    <obj2:thing>bbq</obj2:thing>
    <obj2:thing>switch</obj2:thing>
  </obj2:object>
</ns1:object_group>
```

Back to MetaData

As more projects utilize XML, we would like to add a final comment about metadata. We can present the same XML Document in two different ways:

1. Parent-child nodes containing the intelligence (for brevity, we have omitted several “room” nodes).

```
<?xml version="1.0" encoding="UTF-8"?>
<StandardHeader>
  <DateTime>12/08/2010 16:03:46</DateTime>
  <PacketNumber>1</PacketNumber>
  <AnyMorePackets>Y</AnyMorePackets>
  <TotalPackets>4</TotalPackets>
  <house>
    <room>
      <room_name name="kitchen"/>
      <room_item item="sink"/>
    </room>
            |
            |
            |
    <room>
      <room_name name="kitchen"/>
      <room_item item="table"/>
    </room>
<house>
</StandardHeader>
```

2. Attributes containing the intelligence (no “room” nodes have been omitted).

```
<?xml version="1.0" encoding="WINDOWS-1252"?>
<StandardHeader>
   <DateTime>12/08/2010 16:03:46</DateTime>
   <PacketNumber>1</PacketNumber>
   <AnyMorePackets>Y</AnyMorePackets>
   <TotalPackets>4</TotalPackets>
  <house>
    <room room_name="kitchen" room_item="sink"/>
    <room room_name="kitchen" room_item="table"/>
    <room room_name="kitchen" room_item="counter"/>
    <room room_name="kitchen" room_item="microwave"/>
    <room room_name="kitchen" room_item="oven"/>
    <room room_name="kitchen" room_item="range"/>
    <room room_name="kitchen" room_item="refrigerator"/>
    <room room_name="living_room" room_item="pictures"/>
    <room room_name="living_room" room_item="chair"/>
    <room room_name="living_room" room_item="pictures"/>
    <room room_name="living_room" room_item="HDTV"/>
    <room room_name="living_room" room_item="couch"/>
    <room room_name="living_room" room_item="chandelier"/>
```

```
    <room room_name="den" room_item="surround_sound"/>
    <room room_name="den" room_item="fireplace"/>
    <room room_name="den" room_item="table"/>
    <room room_name="den" room_item="chair"/>
    <room room_name="den" room_item="lamp"/>
    <room room_name="den" room_item="étagère"/>
   <room room_name="den" room_item="HDTV"/>
  </house>
</StandardHeader>
```

If we compare the byte count of 1) vs. 2) we have 2153 vs.1307 or a nearly 40% in reduction in the size of the XML Document using attributes to contain the intelligence of the payload. Over large XML Documents, that is a significant savings.
Observe the two different ways of shredding the intelligence:

1. Using XPATH:

```
SELECT extractValue(p.XMLtext,'/StandardHeader/DateTime') date_time,
       extractValue(p.XMLtext,'/StandardHeader/PacketNumber') packet_number,
       extractValue(value(t1),'/room/room_name/@name') room_name,
       extractValue(value(t1),'/room/room_item/@item') room_item
  FROM testclob p,
       table(xmlsequence(extract(p.XMLtext,'/StandardHeader/house/room'))) t1
 WHERE p.id = 6;
```

2. Using dbms_xmldom[8]:

```
DECLARE

    dDoc           DBMS_XMLDOM.DOMDocument;
    nlNodeList     DBMS_XMLDOM.DOMNodeList;
    nNode          DBMS_XMLDOM.DOMNode;
    nNode2         DBMS_XMLDOM.DOMNode;
    nmNodeMap   DBMS_XMLDOM.DOMNamedNodeMap;
    aAttr          DBMS_XMLDOM.DOMAttr;
    cText          CLOB;
    i              NUMBER;
    j              NUMBER;
    strRoom        VARCHAR2(50);
    strItem        VARCHAR2(50);


BEGIN

    -- Get the Clob from the table
    SELECT XMLTYPE.getclobval(XMLText)
    INTO cText
    FROM testClob
    WHERE Id = 5;
    -- Create the xml document from the Clob
    dDoc := DBMS_XMLDOM.NEWDOMDOCUMENT(cText);

    -- Get the nodes corresponding the the 'room' tag
    nlNodeList := DBMS_XMLDOM.GETELEMENTSBYTAGNAME(dDoc, 'room');
```

```
    -- Loop through the 'room' nodes
    FOR i IN 0..DBMS_XMLDOM.GETLENGTH(nlNodeList)-1 LOOP

        -- Get the ith node
        nNode := DBMS_XMLDOM.ITEM(nlNodeList, i);

        -- Get the attributes of this node
        nmNodeMap := DBMS_XMLDOM.GETATTRIBUTES(nNode);

        -- Find the room_name attribute
        nNode2 := DBMS_XMLDOM.GETNAMEDITEM(nmNodeMap,'room_name');
        aAttr := DBMS_XMLDOM.MAKEATTR(nNode2);
        strRoom := DBMS_XMLDOM.GETVALUE(aAttr);
        DBMS_OUTPUT.PUT('Room_Name = '||RPAD(strRoom,15)||CHR(9)||CHR(9));

        -- Find the room_item attribute
        nNode2 := DBMS_XMLDOM.GETNAMEDITEM(nmNodeMap,'room_item');
        aAttr := DBMS_XMLDOM.MAKEATTR(nNode2);
        strItem := DBMS_XMLDOM.GETVALUE(aAttr);
        DBMS_OUTPUT.PUT_LINE('Room_Item = '||strItem);

    END LOOP;

    -- Free ressources
    DBMS_XMLDOM.FREENODE(nNode);
    DBMS_XMLDOM.FREENODE(nNode2);
    DBMS_XMLDOM.FREEDOCUMENT(dDoc);

END;
/
```

XPath requires less pl/sql to extract the intelligence than dbms_xmldom. DBMS_XMLDOM traverses the nodes of the XML Document. XPath requires path definitions. Each has its own merits and shortcomings.

Conclusion

XML development began on this project a number of months ago. We recently completed the majority of the SQLX (XML document creation) and XPath (document shredding) programming and began testing. Except for adjusting the SQLX queries and changes to XPath (parsing out similar tag fields under one parent), we made no major revisions. As far as the project is concerned, Oracle's XML DB is a stable environment and meets all the standards according to W3C. However, programming in XML DB does require a bit of some effort. The constructor XMLTYPE creates an instance of an XML object. We used it to convert a CLOB into a properly qualified XML document for storage into a table with an XMLTYPE column. When we shred XML documents we made extensive use of the function EXTRACT (with methods getstringval and getnumval). Once we developed the primitives for XML fragment creation and XML document shredding, we found ourselves using the same type of code over and over again. The iterative process for developing a project using Oracle's XML DB is quite similar and an extension to PL/SQL programming. We hope the information presented here establishes a good starting point for those embarking on an XML project.

About the Author

Coleman Leviter, OCP is employed as an IT Software Systems Engineer at Arrow Electronics. He has presented at IOUG's Collaborate and at Oracle Open World. He was the Collaborate Conference Chair. He is the WEB SIG chair and sits on the steering committee at the NY Oracle Users' Group (www.nyoug.org). His articles have been published in Select Journal, IOUG Tips and Best Practices and OTDUG Journal. He has worked in the financial services industry and

the aerospace industry where he developed Navigation, Flight Control and Reconnaissance software for the F-14D Tomcat. Coleman was recently elected to the IOUG Board or Directors. He may be contacted at cleviter@ieee.org .

Appendices

[1] Wikipedia, http://en.wikipedia.org/wiki/Transportation_management_system
[2] Wikipedia, http://en.wikipedia.org/wiki/B2b
[3] Wikipedia, http://en.wikipedia.org/wiki/Oracle_E-Business_Suite
[4] Wikipedia, http://en.wikipedia.org/wiki/Warehouse_management_system
[5] Oracle® Database PL/SQL Packages and Types Reference 11*g* Release 1 (11.1), Part Number B28419-03
[6] Wikipedia, http://en.wikipedia.org/wiki/XML_schema
[7] Wikipedia, http://en.wikipedia.org/wiki/XML_Schema_(W3C)
[8] http://docs.oracle.com/cd/B28359_01/appdev.111/b28419/d_xmldom.htm

Creating an Operational Data Store Using Schema Integration

Angelo R. Bobak, Atos

In today's modern business environment, corporate entities are constantly merging or splitting, internal divisions are sold to different companies, and new business lines are created in order to meet the challenges of difficult economic times. Business data integration is a complex problem that must be solved when organizations change or enhance their internal structures. New IT departments must be merged with old ones, and transactional, operational, and master data must be integrated in order to be managed efficiently, if the business is expected to grow and be profitable.

The goal of this presentation is to present a simple yet thorough process that describes the challenges of business data integration and the solutions to these challenges. It will show you how the application of a technique called "schema integration" addresses these challenges.

Schema integration is both a theory and process that was pioneered by experts in the field of data management. We will discuss the techniques of two of these pioneers, M. Tamer Ozsu and Patrick Valduriez in the design of an Operational Data Store (ODS) for a small business.

M. Tamer Ozsu and Patrick Valduriez also discussed distributed database architectures and related topics such as distributed transaction processing and federated database architectures in their books and papers.

For our discussions, we will utilize some small examples that are vendor agnostic.

The examples will be simple, but have enough complexity to identify and resolve the key issues and challenges that surface when integrating data from multiple source operational databases.

This discussion is both theoretical (mildly) and practical. It is mildly theoretical in that it instructs and guides the audience through design steps based on the theory. It is practical in that it also presents a case study of a classic business data integration problem, the creation of an Operational Data Store (ODS).

What Exactly Is An ODS?

An Operational Data Store (ODS) is a key component in data integration and data warehouse (DW) architectures. Its role is two-fold: to integrate data from operational systems so as to provide a single view of the enterprise data for operational reports; and for delivery to data warehouse platforms, enabling production of advanced Business Intelligence (BI) solutions.

The design of the data model for an ODS should be based on a schema integration technique pioneered by experts and academics in the field. This technique derives a unified data model based on the integration of all source database schemas that will feed the ODS.

There are several flavors of schema integration. The technique that we will use is called binary schema integration and its variations.

In our presentation, we will discuss not only how to use schema integration, but also show how solid data modeling techniques, from logical modeling, physical database reverse engineering, and physical database modeling are used to create the ODS.

Key Topics

FoundationAL Concepts

What is an Operational Data Store? This discussion provides some basic foundational concepts of the ODS: what it is, how it is used, and its role in a data warehouse architecture and data integration project. It also identifies some of the challenges faced when designing this data integration model. Specifically, we will address the various layers of the architecture:

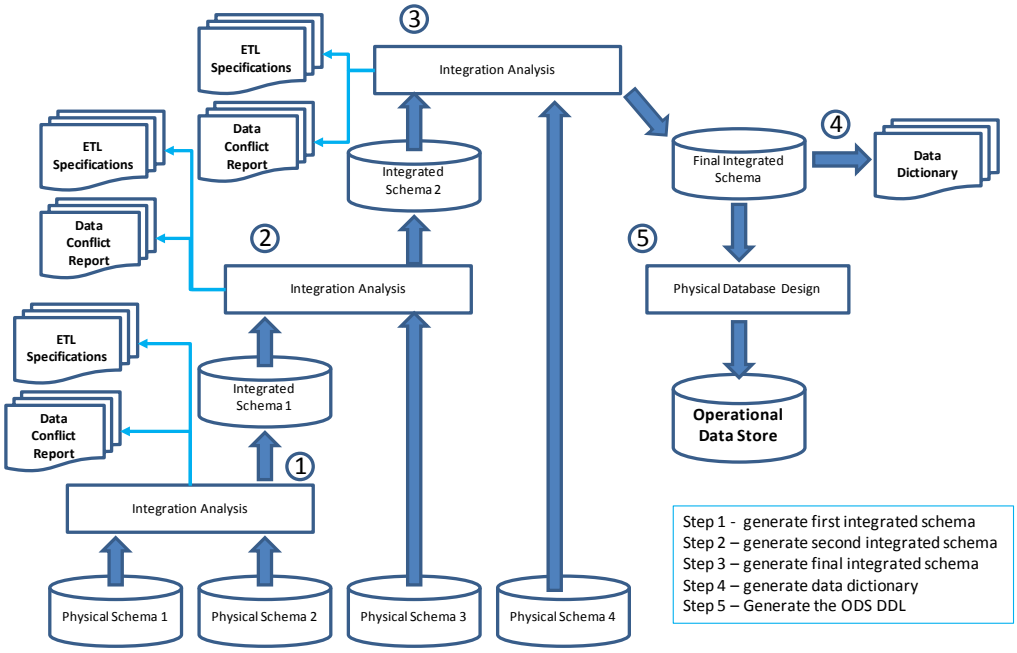
- The Interface Layer
- The Data Staging Layer
- The Data Profiling Layer
- The Data Cleansing Layer
- The Data Integration Layer
- The Export Layer

Each layer is separated by one or more ETL processes that loads, transfers, measures, cleanses and delivers the data of interest.
We will delve into not only the architecture, but also into some simple examples of the various main processing components that stage, profile, cleanse, and deliver the data.

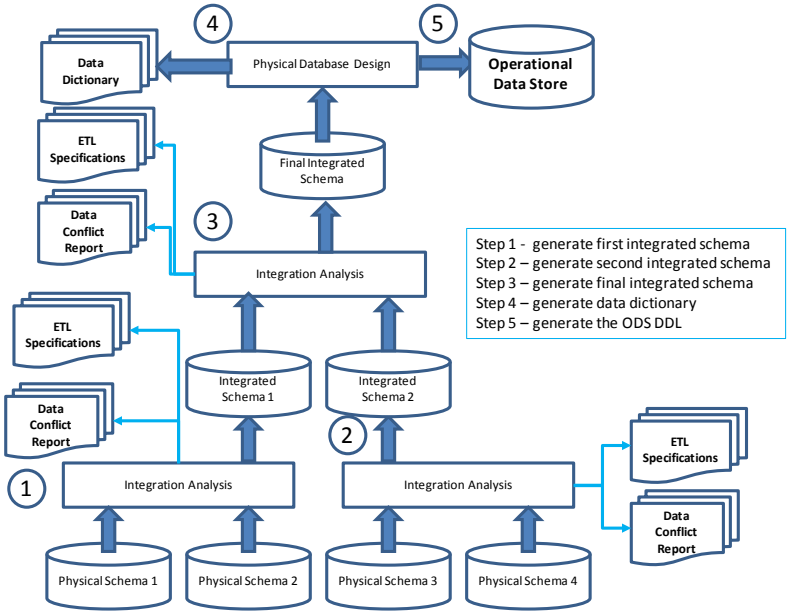
What is Schema Integration?

This discussion introduces the process of schema integration. It identifies the three types of data conflicts that have to be resolved to integrate physical database schema, specifically naming, data type, and data structure conflicts. Additionally, it shows how to generate specifications for the ETL (Extraction, Transformation, and Load) processes that are used in the data conflicts resolution steps.
Schema Integration, or more specifically Binary Schema Integration (BSI) by its very name implies taking two source schemas at a time and combining them to generate one integrated schema. The Figure below depicts this process for merging 4 schema.

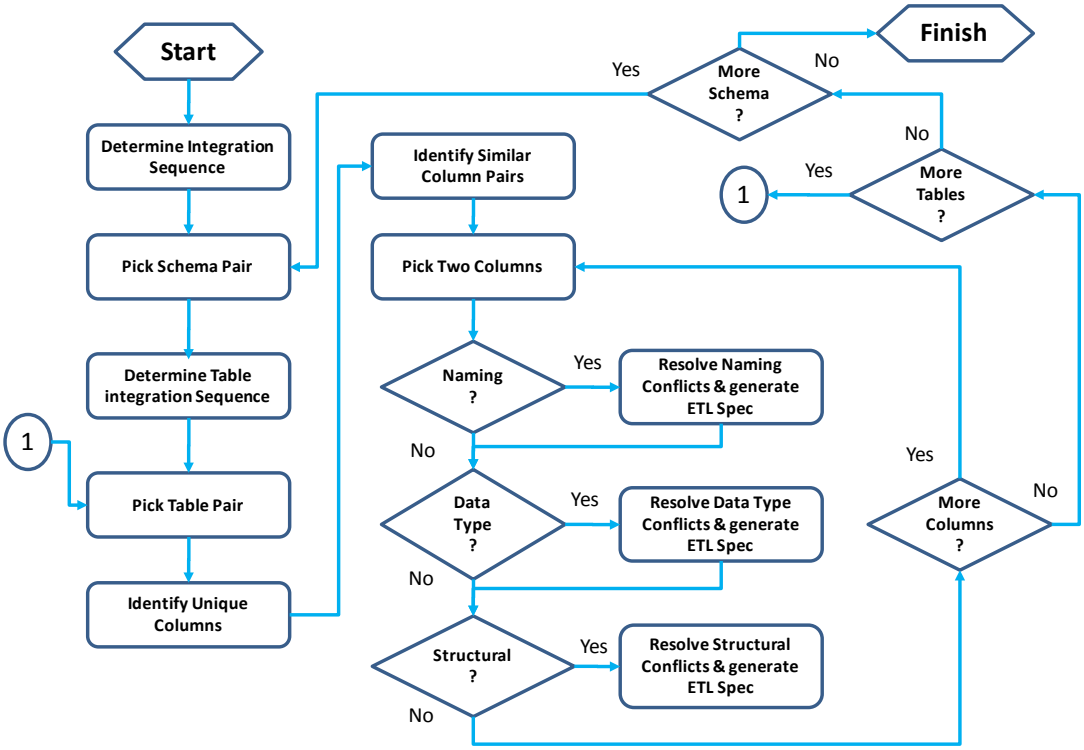
Binary Schema Integration – Technique 1



Notice how the first 2 schema are merged to form an intermediate schema. Then a third schema is merged with the first intermediate schema and the process continues until all schema are integrated.
Variations of this technique are illustrated below:

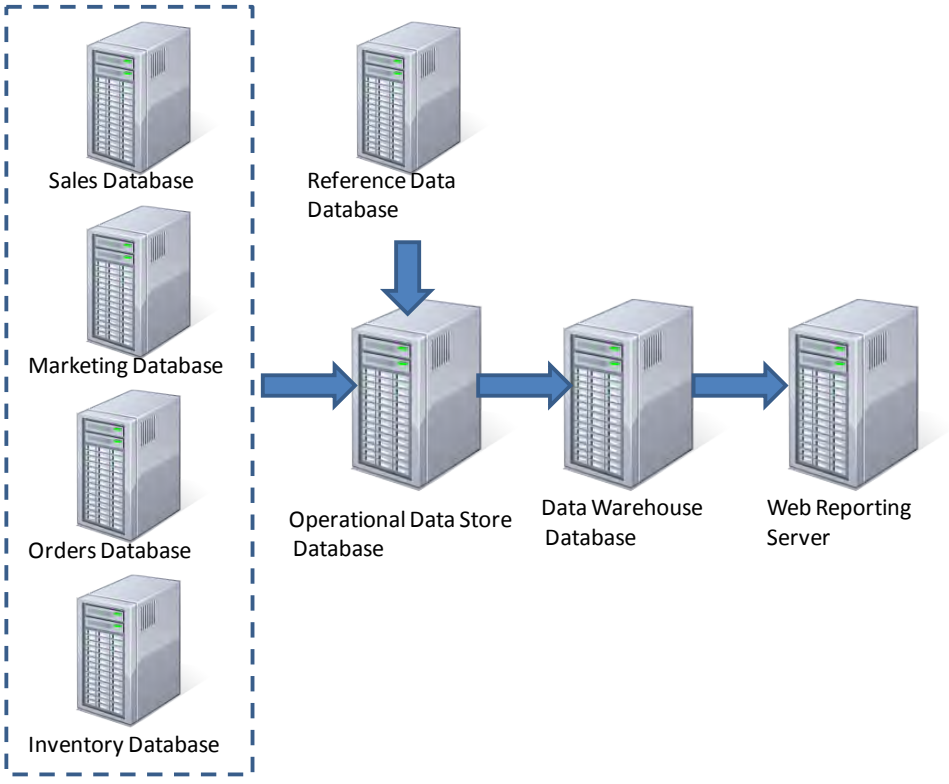


Notice how we integrate 2 pair of schema at a time and then integrate the results.
This process allows us to combine related schema pair like products with products, projects with projects, customers with customers, etc. The process pairs up and merges the schema pair to form a first layer of intermediate schema. Then the intermediate schema are integrated to create a second layer of intermediate schema and the process continues until all pairs are merged into the final ODS model.
A flow chart for the integration process is depicted below



The process begins be determining an integration sequence. What we mean by this is we identify schema to merge that satisfy some condition like a business requirement or merging databases that have the highest level of common data like merge inventory and product databases together or customer and location databases. A business requirement might be to merge all customer schema first as an integrated customers master is required to support the customer base.
Once the sequence is determined, we begin the next step of merging the first two schema. We pick the first two tables to merge out of a list of table pairs (that we prepared) so we can combine them. From this pair of tables we identify the unique columns to each schema and the common columns. The unique columns we can pretty much leave alone. The common columns are what we are interested in.
For each common column we look to identify any data conflicts, specifically data type, data length or structure and data naming conflicts. As each conflict is identified we resolve it by creating the logic flow and then documenting it in an ETL specification for our developers.
This process continues in a loop until all the columns for all the tables and databases are merged and all the conflict resolution steps are documented in the ETL specifications. One all the schema are combined you are done!

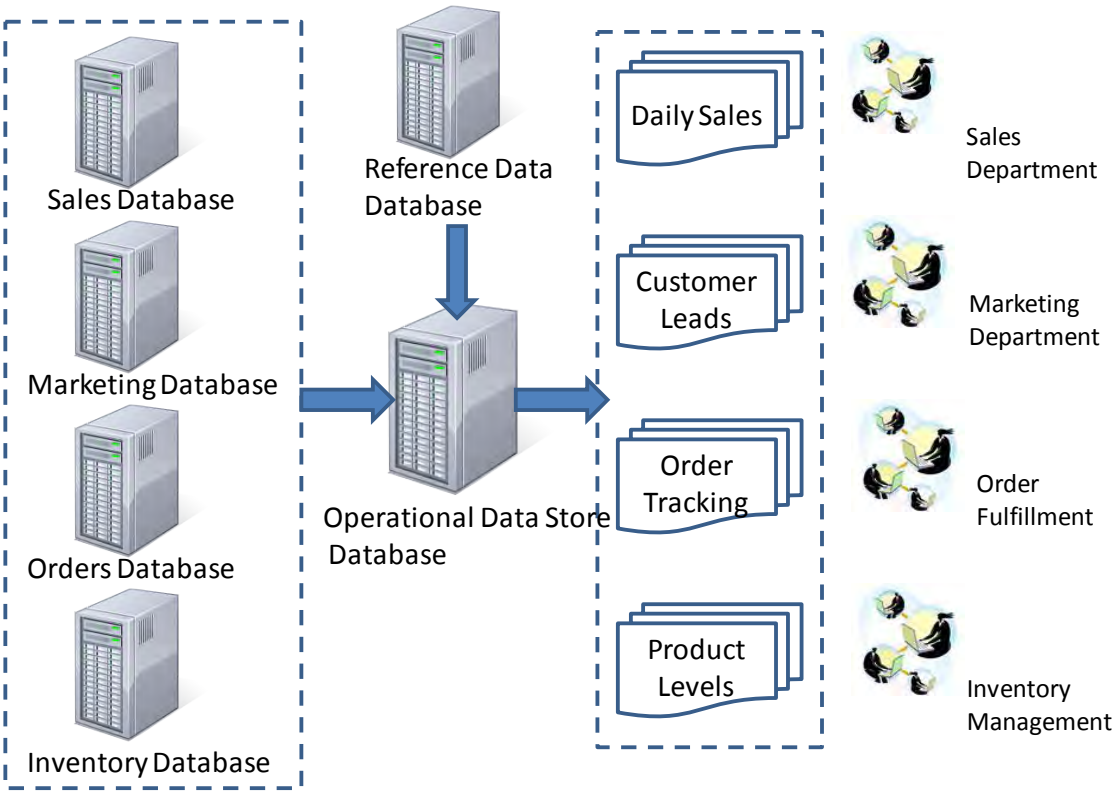
The Role of the ODS within DW Architectures. Next, this discussion delves into greater detail on the role of the ODS within a data warehouse and operational reporting architecture. It describes the component layers of the ODS, together with the de-normalized tables and SQL VIEW architecture required to support operational reporting and also data preparation for delivery to downstream data warehouses and data marts. The export layer can also be implemented as a series of web services to deliver the data downstream users and applications need.
The diagram below shows a typical data warehouse architecture:



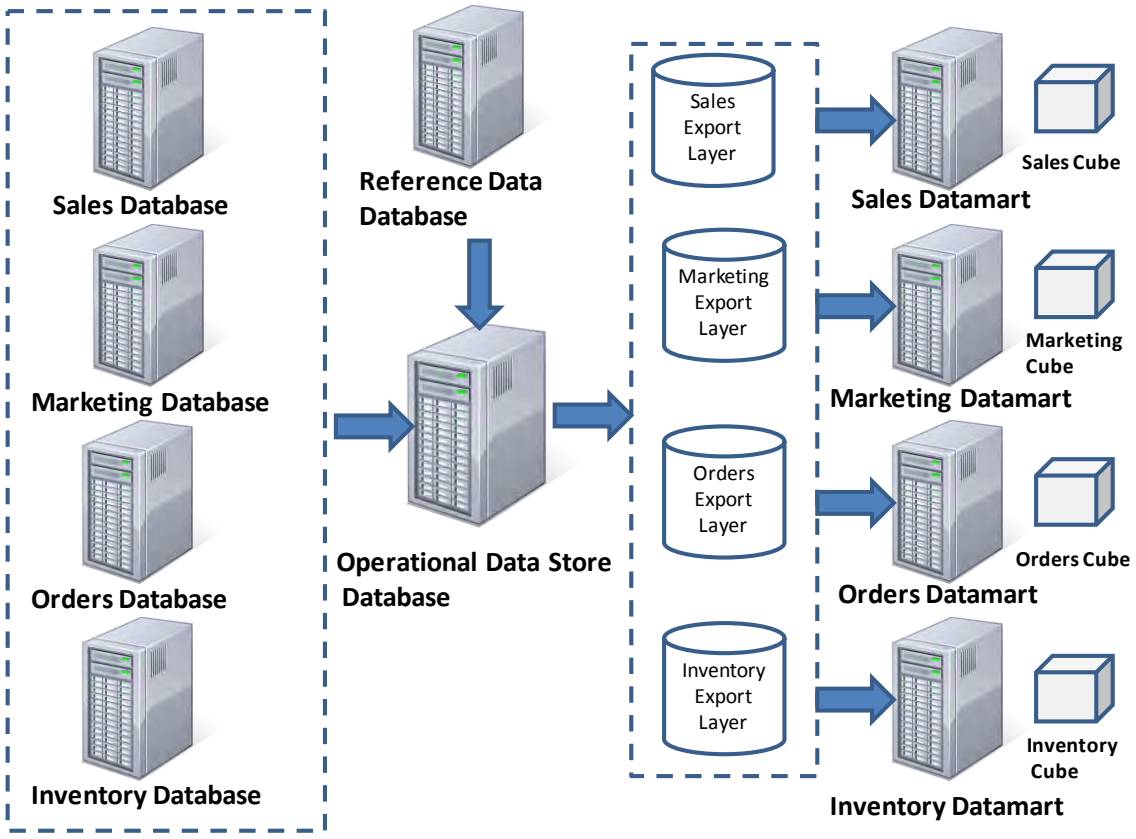
Source databases include sales, marketing, orders and inventory databases. Not shown but also very important are the customer and product price databases. These are staged in the ODS and merged into the ODS core set of tables that were designed with the schema integration processes.

At this layer of the architecture we also perform data quality profiling and cleansing and data enrichment. The reference database provides the necessary master data to fill in missing values like state codes, postal codes or de-duplication of address data.
The data is then presented to the data warehouse via an export layer of web services, SQL views and tables to the downstream data warehouse for loading into a STAR or SNOWFLAKE schema. Connections are made via the usual connection protocols like ODBC, OLEDB, etc.
Another role of the ODS is as a provider of data for operational reporting as the figure below depicts:

As a Source for Operational Reporting



The staged and merged transaction and reference data is a good source of operational reports for the various sales, marketing, order and fulfillment departments. Data in these types of reports is typically at a very low level and is only 1 day to at most 3 months old. By low level we mean that the granularity of the data is at the lowest transactional level. Any historical reporting is left to the data warehouse or the data mart.
The next diagram depicts the ODS providing data for a set of data marts. Data marts are similar to data warehouses in that they use STAR or SNOWFLAKE schema for the underlying databases. They are design to store large amounts of historical data which can span years.
They differ from data warehouses in that each data mart is dedicated to a specific business of the enterprise whereas the data warehouse covers all aspects of the business. For example, one can have a data mart dedicated to orders and sales or just marketing leads.
The figure below depicts a generic data mart architecture:



Notice the delivery of the data via the export layer to the data marts. Each data mart has at least one multi-dimensional cube that supports OLAP type analysis.

Let's take a check point. We have gone over the basics of what an ODS architecture is, what schema integration is and what the process behind schema integration is. We have also looked at a few examples of reporting architecture that use an ODS in order to deliver high quality, cleansed and integrated transactional data from multiple transactional systems. Lastly, we briefly discussed the importance of master data and data quality processes such as data profiling and data cleansing.

These architectures will deliver various reports and dashboards to report on data quality. Below is a mock up of a simple data quality scorecard:

DATE: 10/10/2012
DATA PROFILE REPORT

TBL NAME	COL NAME	NULLS	DUPLICATES	OUT OF RANGE	MAX VALUE	MIN VALUE	DISTINCT VALUES
State	State Key	0	0	0	1	50	51
	State Code	1	2	2	AL	??	49
	State Name	0	2	0	ALABAMA	WYOMING	50
	Country Code	2	0	1	US	ZZ	49

ROWS SAMPLED 51		NULLS		DUPLICATES		Out Of Range		DISTINCT VALUES	
Statistics Score	State Key	0.00%	Passed	0.00%	Passed	0.00%	Passed	100.00%	Passed
	State Code	1.96%		3.92%		3.92%		96.08%	
	State Name	0.00%		3.92%		0.00%		98.04%	
	Country Code	3.92%		0.00%		1.96%		96.08%	

DATE: 10/12/2012
DATA PROFILE REPORT

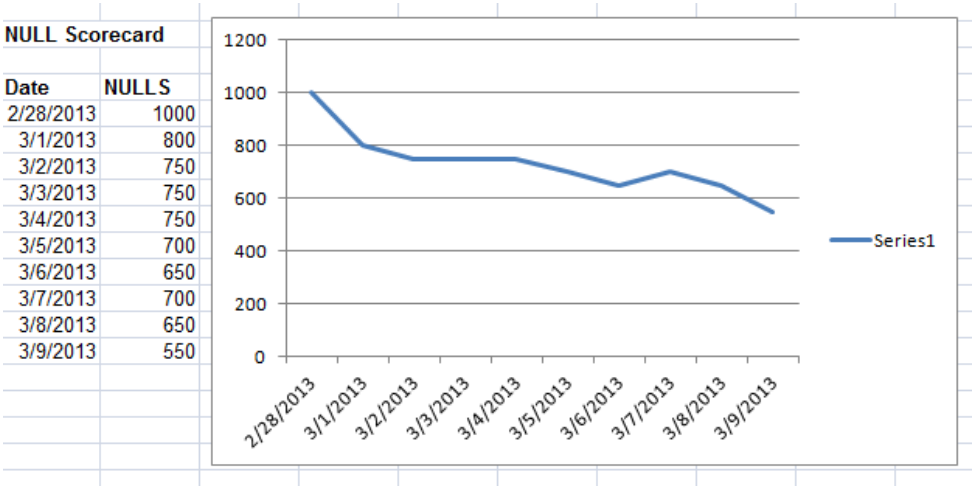
TBL NAME	COL NAME	NULLS	DUPLICATES	OUT OF RANGE	MAX VALUE	MIN VALUE	DISTINCT VALUES
State	State Key	0	0	0	1	50	50
	State Code	0	0	0	AL	WYOMING	50
	State Name	0	0	0	ALABAMA	WYOMING	50
	Country Code	0	0	0	US	US	50

ROWS SAMPLED 50		NULLS		DUPLICATES		Out Of Range		DISTINCT VALUES	
Statistics Score	State Key	0.00%	Passed	0.00%	Passed	0.00%	Passed	100.00%	Passed
	State Code	0.00%	Passed	0.00%	Passed	0.00%	Passed	100.00%	Passed
	State Name	0.00%	Passed	0.00%	Passed	0.00%	Passed	100.00%	Passed
	Country Code	0.00%	Passed	0.00%	Passed	0.00%	Passed	100.00%	Passed

Notice the basic checks:

- Checking for NULL values
- Checking for duplicate values
- Check maximum to minimum value range
- Check number of distinct values

These checks provide the bare minimum information to deliver a profile of the quality of the data being examined. A before and after snapshot is taken to see if the data cleansing processes we implement are working.
Additionally, we may graphically represent the quality as in the scorecard below:

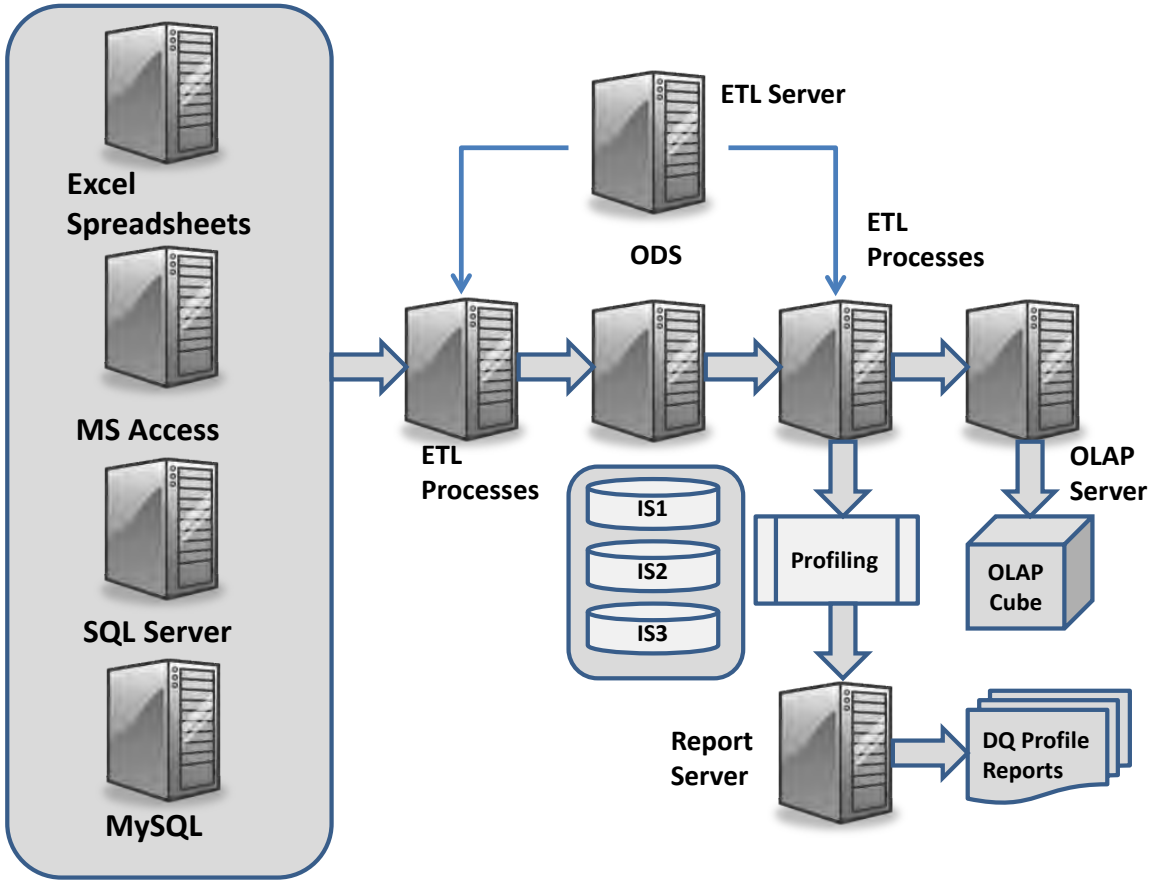


This is an excellent method to present data stewards with the levels of data quality so they can identify issues, recommend processes for cleansing and monitor results.

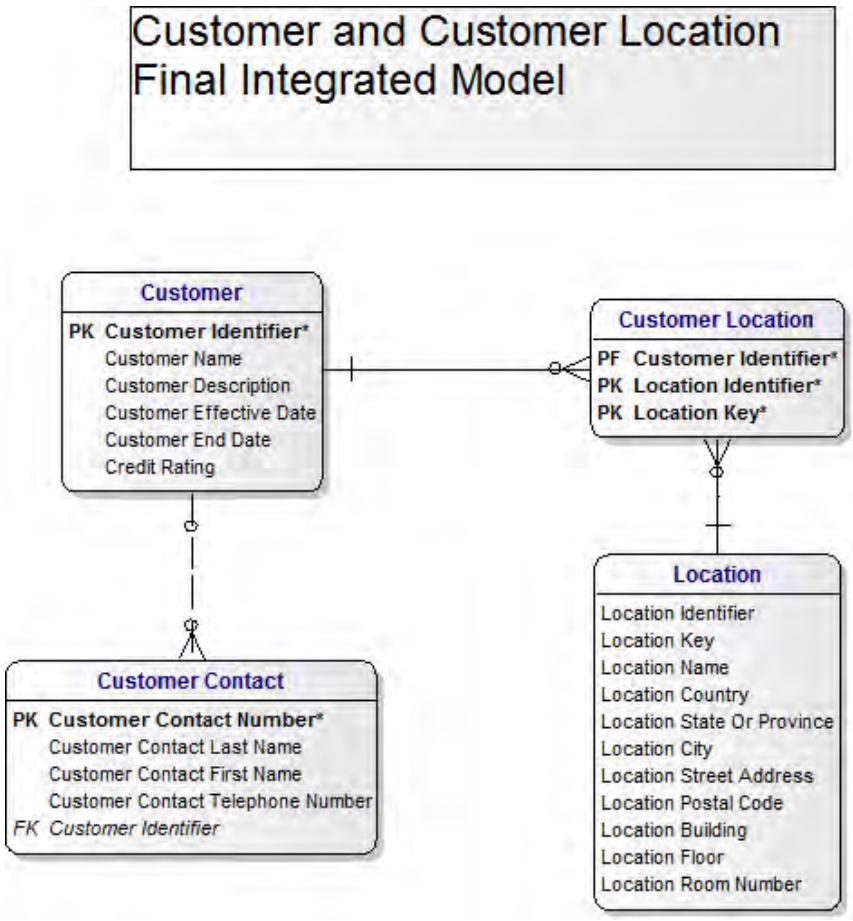
This is also a valuable tool in the schema integration processes as it allows us to check the quality of the incoming data as we load it into the new ODS.

PREPARATION and Design

Let us now introduce our case study, a small, fictitious international coffee product distribution company that has acquired some small, independent coffee roasters and coffee equipment vendors. The diagram below depicts the simple architecture and data flow for the various data repositories uses by the small companies that were acquired:

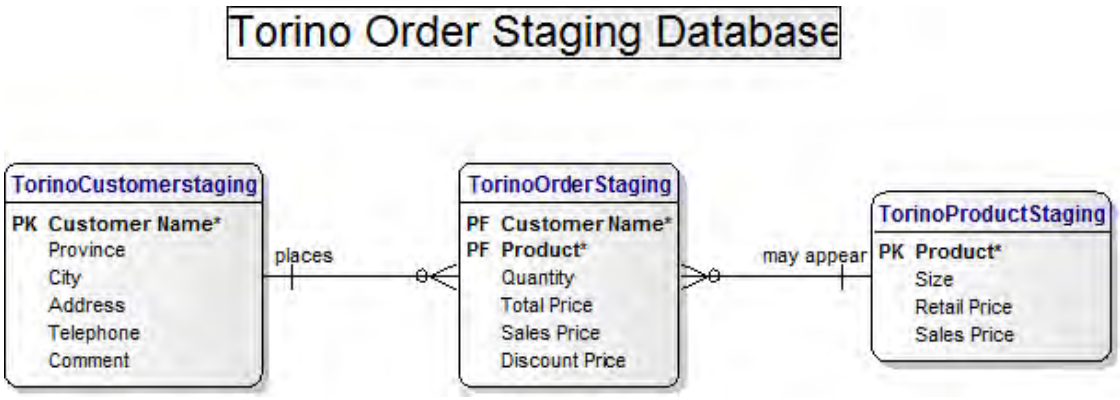


As can be seen, we have 4 source databases, each implemented with popular database software.
The example vendor tools above are just to illustrate how it is necessary in these types of projects to pull data from multiple, multi vendor sources. Notice that the data is pulled from the source systems with an instance of an ETL Server running two sets of processes. The first is to pull and stage the data, the second is to perform data quality checks and data cleansing activities.
The ETL tool stages the data and then merges it into the three integrated databases on the ODS. The data is then fed to a set of profiling ETL process so that various data quality statistics reports can be generated. These could be presented to a web server, so that data stewards can view them on a report server.
The diagram also shows how another set of ETL process is used to extract data and feed it to a small data mart composed of one fact table and several dimensions so that we can generate an order cube with any of the popular OLAP (Online Analytical Processing) tools available on the market. These tools are used to create multi-dimensional structures called cubes which are similar to spreadsheet pivot tables.
Below we present a partial view of the final integrated data model for an ODS. The customer subject area appears in the diagram below:



This is a simple model but it supports the basic information one would require to deliver integrated customer data.
The customer entity contains attributes such as customer identifier, name and description. Also the effective and end dates for a customer together with the credit rating is included.
Next, a customer contact entity contains basic name and contact telephone information for the customer. In this model a customer can have one or more contacts.
Third is our customer location entity. A customer can have one or more locations. This entity simply acts as a bridge table between customer and physical sites.
Our fourth and final entity is the location entity. This is where the physical location address attributes are stored. These include street address, country, state, postal code and even attributes like suite, building floor or room numbers.

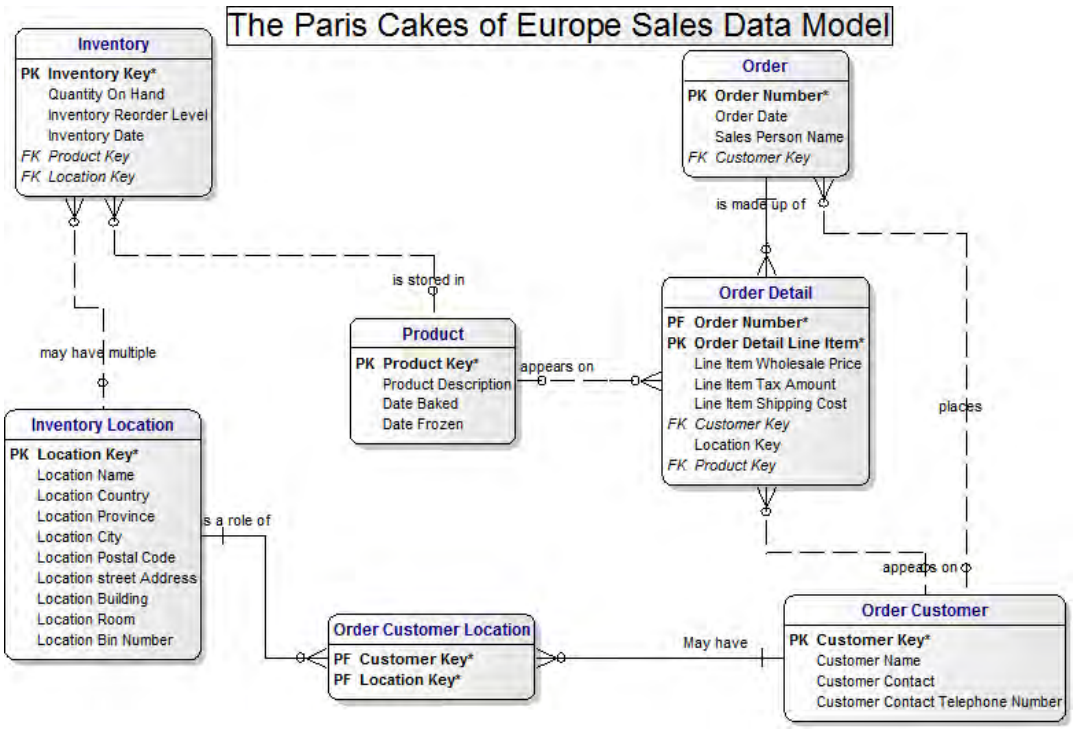
Reverse Engineering the Source Schema. Next we will discuss the need to reverse engineer each the different data repository sources used in our process so we can apply schema integration techniques to create our ODS. This is an important step that needs to be performed prior to the integration steps as it provides us with the necessary data dictionaries that will aid us in merging schema and identifying data conflicts. Without these design documents it will be impossible to successfully merge multiple database schema together.
Another important task is to prepare the logical and physical data models for the operational data sources. Below is a physical model for a company order database located in Torino Italy:



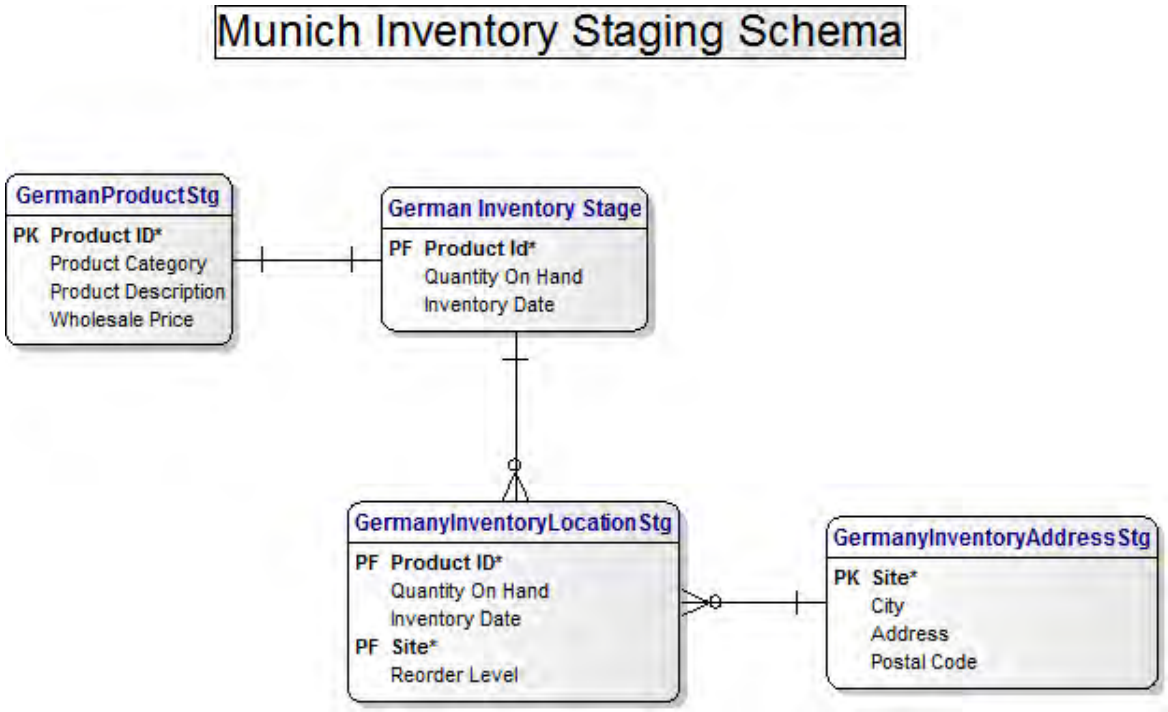
This model could be an interpretation of the underlying model in a set of spreadsheet tabs. Our example company could use a set of spreadsheets to manage orders in one of its divisions or sub companies. Another company might use a relational database or an elaborate set of files.
To further complicate things, each of the stores in this company might use the same set of spreadsheets but with minor alterations and variations. It is important to capture at least the most important physical implementation details such as:

- Primary to foreign key relationships between tables
- The primary keys of each table
- The foreign keys found in each table
- The data types of each column
- Detailed descriptions of what the tables store
- Detailed descriptions of the column names

These minimal set of meta-data and diagrams will be of great value when merging the schema. In the interest of space the diagrams do not show the column data types.
Next we examine another schema that would be integrated into our target ODS. This time it is a model of an inventory control database in Paris France. We notice that the customer information is minimal, only one table. This is a big difference from the prior model
This diagram is depicted below:



Here we also see typical order header and order line entities linked to the customer entity.
Order Header contains high level order information such as order data, order number and the sales person that took the order. A foreign key exists that links the order header to the customer.
We also see that the order entity is linked to the order detail table were all the important attributes of an order are kept. These include line item number, or price, product and miscellaneous amounts such as tax amounts and shipping costs. This entity is also linked to the customer entity and also the product entity.
Additionally, on the left side of the model we see the inventory related entities. We see the inventory entity linked to the product entity. We also see that an inventory is composed of multiple locations. This is supported in the inventory location table. Notice how the inventory location table is linked to the order customer location link table. It seems that the customer entity uses this link table so it can store customer location data in the inventory location table. This is a bit confusing. Maybe a more generic design to support all types of locations is needed to make the design clearer.
This is also another aspect of schema integration. As your merge schema you look for opportunities to improve the underlying design. Also, data models help the designer identify the class of data conflicts called structural conflicts. These involve different relationships between related entities in the various schema in addition to data objects implemented as tables in one schema and only attributes in the second schema.
Our last model is a similar inventory model as the prior model but for a company in Munich Germany. This model is depicted in the figure below:



The subject area is similar to the prior model we discussed. This also is for inventory and product. Notice also that there is no customer information. We are only interested in supporting inventory data in this model.
I think we can see by now how these various models are not only different but related and at least conceptually, we can start to visualize some sort on integration sequence. Our key areas are customer, locations, orders, inventory and product.

In our integration process for a real production project we would start to build lists to identify potential candidates for schema pair and subject area pairs in case the schema are complex and have multiple subject areas.
Lastly, I cannot stress how a minimal set of data dictionaries, object lists, data models and schema lists are an important set of tools that will aid us in our integration process.
We now wrap up this abstract by identifying the design steps we will study in the presentation.

Designing the Interim Schema. We discuss schema integration techniques to merge the models of the various data sources.

Preparing the ETL Specifications. We discuss how to create the ETL specifications required to resolve the data conflicts identified during the schema integration process. The specifications will be a set of spreadsheets that describe the ETL logic and a set of data flow, process hierarchy and process dependency diagrams that will be used to create the ETL processes required to stage, transform, enrich and load the data into the ODS.

Designing the Physical ODS Database Model. We will translate the final interim schema into a physical database model and also present the final DDL statements used to create the ODS.
Of course, our examples will be small so as to be discussed in the brief time slot allotted for our discussion.

Physical Implementation

In this section of the discussion, we will deal with the physical aspects of our case study. Topics included are:

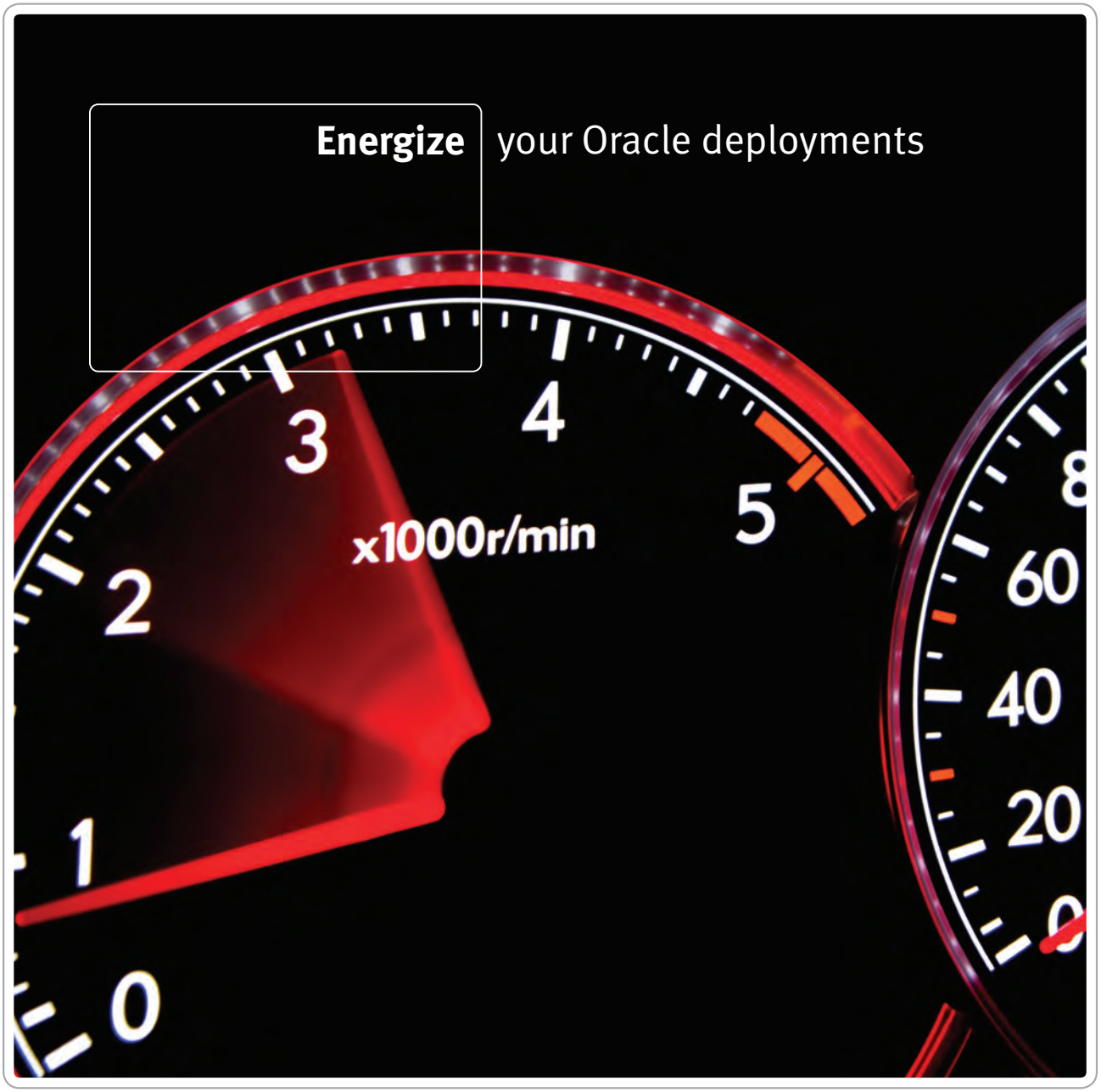
Designing Our ETL processes with SSIS. In this discussion we identify one example of the necessary diagrams and specifications for our ETL processes.

Data Quality profiling. Lastly we discuss the importance of data profiling, data cleansing and data quality. We will discuss how to collect statistics for data profiling such as counting column NULL values, maximum/minimum values, patterns and other statistics. We will show you how to collect these statistics into relational tables so that you can create simple web reports for data stewards to use.

Summary

To summarize, in this short hour we will cover the theory, steps and tools required to create a successful ODS, using the theory of schema integration. We will examine various ODS architectures and their use and also discuss the importance of high quality data.
This abstract was adopted from portions of several chapters of my book: Connecting the Data published by Technics Publication.

EMC²
where information lives®



Reduce risk and accelerate your application deployments by drawing on the power of our Oracle® expertise.

With EMC® Proven® Solutions, your information infrastructure accelerates towards greater productivity. Learn more at www.EMC.com/oraclesolutions.

EMC² PROVEN SOLUTIONS

Top 5 Issues That Cannot Be Resolved by DBAs
(other than missed bind variables)

Michael Rosenblum, Dulcian, Inc.

Common knowledge says that if there is something wrong with the database, blame DBAs. Fortunately for database administrators, about 95% of such cases have nothing to do with their job responsibilities. Unfortunately, it takes a lot of time to explain this to any manager. It takes even longer for the organization as a whole to look for a real solutions to its IT problems rather than asking their DBAs to look for "_run_faster=TRUE". All too often, quick-fixes are sought because of time constraints, running production environments, and/or somebody else's code written 10 years ago. This eventually results in a situation where nobody can even explain why your system behaves in such-and-such a way because of layers upon layers of patches and tweaks.

The goal of this paper is to illustrate the most common "pseudo-DBA" issues and suggest proper ways resolving them. The following are the top 5 issues identified that cannot be resolved by DBAs:

1. "Smart" columns - Storing multiple data elements in the same attribute looks like a good idea until you need to run a report, or until you need to adjust the whole structure.
2. "STUFF" table – It is very tempting to create a table called PARTY to store objects of all kinds ranging from the company main office building to the name of your great-grandfather from the security clearance form. This means that all child tables with foreign keys pointing to PARTY cannot easily differentiate between these party types. This creates a data integrity nightmare.
3. Improper datatypes – If the date field is stored as text, you can forget about effective range scans and cardinalities because, for Oracle between 20123001 and 20130101, there are 7100 numbers and not one day! Another issue is implicit datatype transformation. There is not much that can be done by DBAs if front-end developers pass timestamp values into date variables (which in turn invalidates index lookup)
4. Lots of user-defined functions in SQL – Unfortunately, it is very hard to tell upfront how many times a user-defined function from the WHERE clause will be executed because this number depends upon the execution plan. Execution plans change! Also, it is very hard for DBAs to guess the "weight" of any function and optimize the entire statement.
5. Inefficient hierarchical structures – It is true that many things can be stored hierarchically. However, managing a lot of hierarchical objects is very expensive and resource-intensive. Unless developers are willing to work with some kind of denormalized structures, the performance "black hole" will be deeper and deeper with the growth of total data volume.

Target Audience

This paper is targeted at developers and software architects. They should be able to properly focus development efforts to solve real problems instead of just throwing everything to the DBAs.

Executive Summary

This paper explains the benefits of close collaboration among all teams (DBAs/Developers/Architects), even at the early stages of the project lifecycle. From a practical standpoint, the top five development traps (including ways to get out) will be covered in detail.

Background

Too many issues in the contemporary IT system are being resolved by DBAs. The main reason for this situation is that managers are afraid to touch production software. Maybe the original architect is no longer available, or the

documentation is not complete. This paper works as an "early warning system." What should be checked while developing a new system so that the well-being of the whole company will not be left to miracles performed by DBAs? It may be too late for existing systems. However, let's assume that we had time machine and could influence the original design decisions. The article discusses the five most common mistakes and illustrates why seemingly smart solutions later lead to major disasters.

Introduction

The fact that software managers are afraid to touch production code is normal. What is not normal is that this fear leads conclusion that since we cannot change a single line of production code and problems have to be solved no matter what, it is the DBA's job to come up with a solution. Although in many cases, the overall qualifications of DBAs in the US are good enough to resolve just about any problem, the complexity of the IT environment places more and more burden onto DBAs without this extra load.

The goal of this paper is to provide IT specialists with a tool to help them convince management that fixing real problems is more effective than inventing system bandages, especially in the long-run.

Architect's Mistakes

The most dangerous issues for the long-term survivability of the IT system are usually linked to decisions made by software architects at the inception of the overall solution. These decisions are not always well thought out from a total system strategy standpoint and are sometimes influenced by current trends.

"Smart" Columns

The idea of storing multiple logical data elements as a single structural data element is not new. Currently, the "ultimate" answer is XML, but for the last 20 years, the notion of smart columns lures a lot of system architects into a serious trap. The idea is pretty simple. If multiple "things" are being used for the same purpose, why not store once and parse them as needed? From my experience, the two most popular cases for this implementation are as follows:

- Organizational rollup – For example, a pipe-delimited combination of Region/State/City/Zip
- Questionnaires, especially ones with Y/N possible values – The whole answer becomes a single text line where some number of characters maps to a question number

I personally have seen both of these cases in real systems, and both caused significant problems. In the first case (organizational rollup), the whole system fell apart when there was a sudden need to add an extra organizational level in the middle. Also architects recognized that developers were actively using counts of separators (if none – region, if one – state, etc.) for decision-making. This logic is completely flawed. The resulting cost of implementation was an order of magnitude more than originally estimated and involved almost 100% code review of the whole system.

The questionnaire answers problem comes from a different angle. Due to the nature of the business, sets of questions were very volatile, but there was an explicit requirement to keep historical answers of previous incarnations of the same set. As a result, the meaning of any character in the answer string was dependent not only on the question number, but also on the version that was defined by start/end dates (i.e. derived from the date when the answer was created). Therefore, it is no surprise that efficient bulk reporting from such structure became an impossible task.

Suggestions: In both of these cases, the solution is in the separation of the physical and logical representations. However, the data access patterns are different. In the case of organizational rollup, the meaning of a sub-element is higher so the resulting changes to the implementation should be different:

- Organizational rollup
  - Split smart column into real data elements
  - Aggregate them back using either virtual columns or views
- Questionnaires
  - High-quality version control to prevent data corruption
  - Materialized views to transform historical data into the "current" format to improve reporting
  - A lot of function-based indexes on top to optimize querying

"STUFF" Table

As previously mentioned, system changes are costly. Well before the NoSQL days, people started to use highly generic solutions to feel more protected against future system changes. Over the last 20-30 years, such approaches followed various trends, but still were never completely off the radar.

Throughout my entire IT career, I have been a very strong proponent of generic solutions in theory. However, in real life, I found that, all too often, they mask an incomplete (at best) or erroneous (at worst) understanding of subject areas by senior architects. Also, very often generic solutions that are highly efficient in one aspect of the IT system are a complete disaster in others and their overall impact is negative. This comes under the heading of looking for a silver bullet that would hit all targets at once. Sadly, there is no such thing. A balance between multiple simultaneous goals is always necessary. The model shown in Figure 1 is one of the most reoccurring data modeling traps:

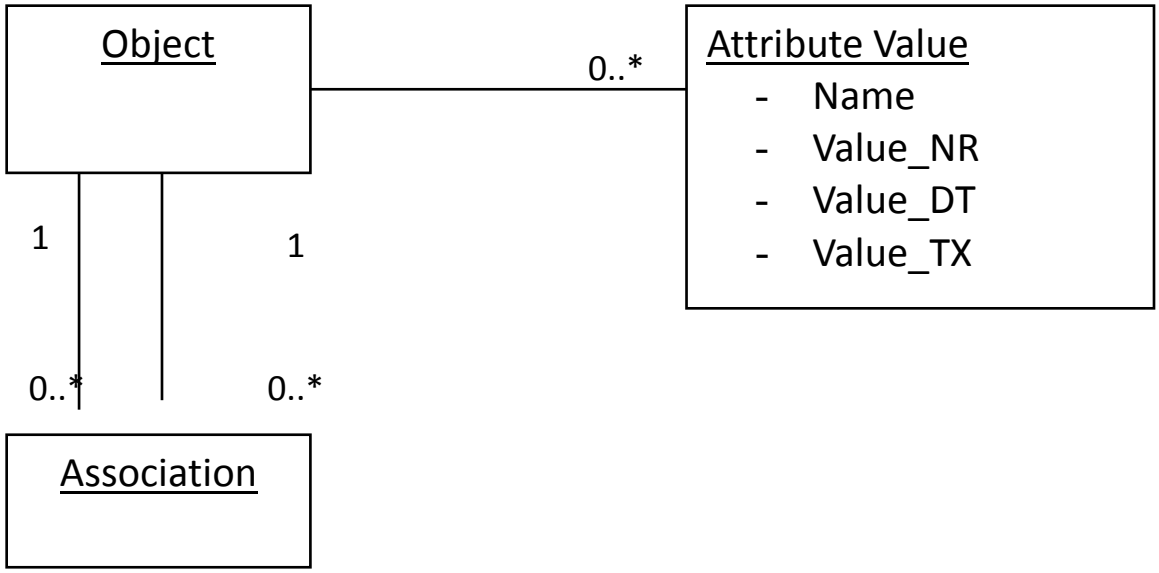


Figure 1: Typical Generic Data Modeling Trap

This type of model is a trap because it usually works perfectly on a small scope of data, especially in a testing environment. Howver, when deployed to production systems and populated with production volumes, all bets are off! Problems start to crop up everywhere:

- Data entry
  - Many operations used to retrieve a single object, which means a lot of wasted resources.
  - Data quality deteriorates very quickly because rules are hard or expensive to enforce.
- Data retrieval
  - Because of mutating structures, indexes are useless.
  - There is no real metadata, so the CBO goes crazy.
  - Performance is sporadic and does not follow any meaningful logic.
- Reporting
  - Reporting is almost impossible, especially if any meaningful aggregation is required.

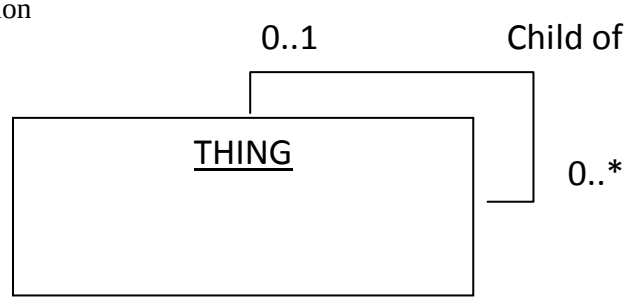
There are cases when key/value stores are perfect, but only when they are appropriate (i.e. in environments with high data quality or when heavy reporting should be handled separately.

Suggestions: Storage is reasonably cheap now, so the creation of duplicate structures that would look like real tables could be justified. It directly solves the reporting issue and helps OLTP (if non-generic storage could be modified directly). Of course, having two images of data may cause data inconsistency, but this is the lesser evil in the current situation.

Insufficient Hierarchical Structures

There are many valid reasons for using recursive data structures. Recursion is a powerful modeling technique to represent linked lists (for example, contract versions) or tree structures (organizational hierarchies). However, there is a big gap between the architect's vision of a recursive structure and its proper representation:

- Real recursion



- "Kind-of" recursion

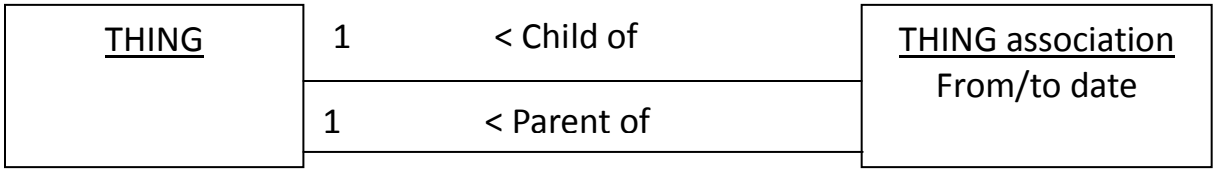


Figure 2: Typical Generic Data Modeling Trap

The reason why many people consider the second case a good choice is because it allows storage of hierarchical structures and time-based versioning. But this causes a lot of extra headaches on the maintenance and data quality fronts for the following reasons:

- Since hierarchical consistency is not enforced, it could easily allow the creation of cycles, dead branches, etc.
- Since FROM/TO dates are associated with each individual association, answering the question about how the tree looked during a certain period of time (not as of some date!) becomes a major challenge to even properly articulate. Also, because there is no single point of time to be referenced, it is very easy to accidentally drop the whole branch of the tree by setting time-frames.

Recommendations: Unfortunately, there is no simple way of implementing hierarchical structures (For Dulcian's attempt to do it, see the presentation "Looping the Loop: Different Ways of Working with Recursive Structures" by Michael Rosenblum & Dr. Paul Dorsey from Collab'12). Draconian enforcement of data quality is the best strategy for simplifying data access. For example, to achieve the second goal it is strongly recommended that you create a lot of denormalized tables and/or materialized views which are accessible by developers for day-to-day needs. Real recursive structures should not be touched unless there is clear need.

Developer's Mistakes

High-level software architects are not the only people responsible for issues that eventually trickle down to DBAs. There are some purely technical decisions which lead to the hunt for someone who fix a messed up IT system.

Datatypes Misuse

Many IT professionals overlook the importance of properly configured datatypes, both for attributes and variables. Many decisions made in the RDBMS environment are made using very granular information, and the slightest misrepresentation can cause significant problems. These problems don't even include pure data corruption when the wrong kind of data is being stored. Some problems are much more difficult to detect.

One of the most common bad attribute practices is storing dates as text (for example, in the format YYYYMMDD). The biggest challenge, obviously, will be data quality, but let's assume that the system enforces data quality pretty well and there are no values similar to "PRESENT" or "NO IDEA" in storage, although about 99% of date-as-text columns I've seen have at least some "strange values". From the DBA's point of view the main headache is very sporadic performance, because we are lying to the CBO for the following reasons:

- Date range {December 31, 2012; January 1, 2013} consist of only two distinct date values
- Textual range {'20121231'; '20130101'} is huge! Since the date is stored as text, starting with the 4th character, there could be any valid character in the current character set.

One of the results of such datatype misuse is that histograms being built for this dataset would be completely wrong. It means that there is a good chance that any report spanning two calendar years would perform significantly slower than expected. The following is an example of an index being ignored if a textual column is being used :

```
-- setup a case
create table misha_date01
as
select owner, object_name,
       to_char(created,'YYYYMMDD') created_tx,
       created created_dt
from dba_objects

create index misha_date_tx_idx on misha_date01(created_tx);
create index misha_date_dt_idx on misha_date01(created_dt);

begin
   dbms_stats.gather_table_stats(user,'MISHA_DATE01');
end;
-- run query against text and date columns
SQL> explain plan for
  2  select *
  3  from misha_date01
  4  where created_tx between '20121231' and '20130101';
Explained.
SQL> select * from table(dbms_xplan.display());
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              | 48100 |  2113K|   299   (1)| 00:00:04 |
|*  1 |  TABLE ACCESS FULL| MISHA_DATE01 | 48100 |  2113K|   299   (1)| 00:00:04 |
--------------------------------------------------------------------------------

SQL> explain plan for
  2  select *
  3  from misha_date01
  4  where created_dt between to_date('20121231','YYYYMMDD') and
to_date('20130101','YYYYMMDD');
Explained.
SQL> select * from table(dbms_xplan.display());
```

```
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                    |   212 |  9540 |    11
|   1 |  TABLE ACCESS BY INDEX ROWID| MISHA_DATE01       |   212 |  9540 |    11
|*  2 |   INDEX RANGE SCAN          | MISHA_DATE_DT_IDX  |   212 |       |     3
-------------------------------------------------------------------------------
```

Recommendations: Usually, virtual columns (TO_DATE) and function-based indexes can help as long as data quality is strictly enforced and developers slowly move to proper data representation.
From a coding point of view, the majority of bad practices involve implicit datatype conversion. Personally, I would like to see Oracle behaving more strictly. The commonly detected issues are as follows:

- Code injections via NLS settings
- Calling wrong overloaded functions/procedures
- Wrong execution plans because of bind variable datatype mismatch

The last case is the most common, which can be easily demonstrated using the same coding example:

```
SQL> explain plan for
  2  select *
  3  from misha_date01
  4  where created_tx = 20121231;
SQL> select * from table(dbms_xplan.display());
-------------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              |   573 | 25785 |   300   (1)| 00:00:04 |
|*  1 |  TABLE ACCESS FULL| MISHA_DATE01 |   573 | 25785 |   300   (1)| 00:00:04 |
-------------------------------------------------------------------------------------
SQL> explain plan for
  2  select *
  3  from misha_date01
  4  where created_tx = '20121231';
Explained.
SQL> select * from table(dbms_xplan.display());
-------------------------------------------------------------------------------
| Id  | Operation                   | Name              | Rows  | Bytes | Cost
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                   |   573 | 25785 |    14
|   1 |  TABLE ACCESS BY INDEX ROWID| MISHA_DATE01      |   573 | 25785 |    14
|*  2 |   INDEX RANGE SCAN          | MISHA_DATE_TX_IDX |   573 |       |     4
-------------------------------------------------------------------------------
```

As you can see, if I pass 20121231 as a number, the index is not being used, but the query runs without any syntactical issues.

Misuse of User-defined Functions

The fight about user-defined functions inside SQL statements started from the moment this option became available. On the upside, they allow for a significantly increased functional range of valid queries, but on the downside, they introduce a huge “black hole” that can consume all of the available hardware resources for no good reason. The key things to remember are as follows:

- The cost of switching between SQL and PL/SQL is not zero. Therefore, in addition to the cost of the processing, there is a cost to calling functions themselves.

- SQL is a set language with multiple possible processing path patterns. This means that, depending upon the execution plan, the same function could be fired a different number of times.

The conclusion is simple. We should decrease the total number of function calls, preferably to zero. And if something is possible to do in SQL, do it (this is especially true for different aggregations and analytics).
Too often, systems are constantly firing hundreds upon hundreds of calls and the IT folks wonder why performance suffers. One of my favorite anecdotes is a case when for a 100-attribute table, every insert consisted of 1 insert + 99 updates, one attribute at a time! It seemed to me that developers were trying to replicate object-oriented style getters and setters without even considering whether this practice was appropriate on the database side.
Fortunately, in an Oracle environment, there are mechanisms to better manage user-defined functions for cases when you just cannot avoid them, both in SELECT and in WHERE-clauses. Unfortunately, all of these approaches require changes to queries and/or PL/SQL so DBAs must have direct access to development teams and the original source code.
Recommendations: Talk to the DBAs and let them see the original code because it is very hard to reverse-engineer processes from the bottom up. The results may surprise you. The following sections describe a number of cases when the impact of these adjustments is very significant.

Managing Execution Orders

Assume for an example that there are two different function calls in the WHERE clause and the first function is calculation-heavy while the second one is calculation-light. Because Oracle uses short-circuit evaluation of conditions, it always makes sense to first check the light function and continue only when it returns a needed value. The key is how to tell the CBO about the statistics associated with each function. By default, Oracle uses the following assumptions:

- Selectivity – 1% (0.01)
- CPU cost – 3000
- I/O cost – 0
- Network cost - 0

All of these parameters could be adjusted using a special command "ASSOCIATE STATISTICS WITH FUNCTIONS". These associated statistics could either be hardcoded or calculated with extremely challenging mechanisms (That topic is outside of the scope of this paper). The following example shows how to use the ASSOCIATE STATISTICS command:

```
associate statistics with functions f_misha_light_tx
default selectivity 0.1
default cost (0,0,0); -- light

associate statistics with functions f_misha_heavy_tx
default selectivity 0.1
default cost (99999,99999,99999); -- heavy
```

Next, run a query using both of these functions in the same condition. Using standard Oracle patterns, the earlier function is checked first. In this example the heavy function is referenced before the light one, which is not very efficient:

```
select /*+ gather_plan_statistics */*
from emp
where f_misha_heavy_nr(empno) = 1
and f_misha_light_nr (empno) = 0

-------------------------------------------------------------------------------
| Id  | Operation         | Name | E-Rows | A-Rows |   A-Time   | Buffers |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |        |     14 |00:00:00.01 |      33 |
|*  1 |  TABLE ACCESS FULL| EMP  |      1 |     14 |00:00:00.01 |      33 |
-------------------------------------------------------------------------------
```

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(("F_MISHA_LIGHT_TX"("EMPNO")=0 AND
              "F_MISHA_HEAVY_TX"("EMPNO")=1))
```

Because of the associated statistics the execution order has been changed as expected. Therefore, it is possible to impact the order of WHERE clause evaluation using statistical methods. Unfortunately, this approach is somewhat limited because package-based functions cannot include statistics. In the worst case scenario, it is possible to create standalone wrappers and reference them instead of using the original ones from the package.

Number of Calls in SELECT Clause

There are multiple ways of ensuring that if a function is referenced in the SELECT clause, it is not fired more often than needed. Unfortunately, few developers are even aware of this problem. My recommendation is to include the following set of examples in any PL/SQL class. This explicitly illustrates the difference between real understanding and guessing. First, set up a basic environment to count total number of calls: a package variable to store the counter, a simple function, and a checker to display/reset the counter:

```
create package misha_pkg is
    v_nr number:=0;
end;

create or replace function f_change_tx (i_tx varchar2)
return varchar2 is
begin
    misha_pkg.v_nr:=misha_pkg.v_nr+1;
    return lower(i_tx);
end;

Create or replace procedure p_check is
begin
     dbms_output.put_line('Fired:'||misha_pkg.v_nr);
     misha_pkg.v_nr:=0;
end;
```

Second, run a very simple query against table EMP, where the function above will be applied against EMP.JOB. And let us keep in mind that there are 14 total rows in the table EMP:

```
SQL> select empno, ename, f_change_tx(job) job_change_tx
  2  from emp;
   ...
14 rows selected.

SQL> exec p_check
Fired:14
PL/SQL procedure successfully completed.
```

If you just use the function, it will be fired for every row. But we know that there are only 5 distinct JOB values, so we should try to decrease the number of calls. In Oracle 11gR2, there is a very interesting internal operation called “scalar sub-query caching” being used while processing SQL queries. It allows Oracle to internally reuse previously calculated results on SELECT statements if they are called multiple times in the same query. The following example tests to see if using this operation helps:

```
SQL> select empno, ename, (select f_change_tx(job) from dual)
  2  from emp;
  ...
14 rows selected.
SQL> exec p_check
Fired:5
PL/SQL procedure successfully completed.
SQL>
```

The result shows that it did help. Now, only five distinct calls are registered, which isexactly as needed. Although, since we are discussing cache, why not use it explicitly? There is another very powerful feature called "PL/SQL function result cache." The following example enables it on the function while the same query is run two times:

```
create or replace function f_change_tx (i_tx varchar2)
return varchar2 result_cache is
begin
    misha_pkg.v_nr:=misha_pkg.v_nr+1;
    return lower(i_tx);
end;

SQL> select empno, ename, f_change_tx(job) from emp;
...
14 rows selected.
SQL> exec p_check
Fired:5
SQL> select empno, ename, f_change_tx(job) from emp;
...
14 rows selected.
SQL> exec p_check
Fired:0
```

The result is impressive! If the first call matches the sub-query caching, the second call is a fantastic example of great performance tuning – everything works as needed, but nothing is being done (actually, this is not 100% true, since the cache should be retrieved anyway, but for practical purposes it is a very simple PK lookup).

Sub-queries with User-defined Collections

Another case of "mystical" Oracle defaults is linked to functions that return object collections. These collections could be converted into regular SQL sets using the TABLE clause. However, there is a minor problem in that Oracle is not perfect in guessing how many objects are in the result set. The following example shows a table with the primary key and a sub-query, represented as a collection of objects:

```
-- create required objects
create table misha_demo_inlist as
select object_id, created
from dba_objects
where owner = 'MISHA'
and object_id is not null;

alter table misha_demo_inlist add constraint misha_demo_inlist_pk primary key (object_id)
using index;

begin
dbms_stats.gather_table_stats(user,'MISHA_DEMO_INLIST');
```

```
end;

-- create object collection
create type id_tt is table of number;

-- run the query
select /*+ gather_plan_statistics */ max(created) from
misha_demo_inlist where object_id in (
select t.column_value
from table(id_tt(227011,227415)) t)

-------------------------------------------------------------------------------------------
| Id  | Operation                                | Name              | E-Rows | A-Row
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                   |        |   1
|   1 |  SORT AGGREGATE                          |                   |      1 |   1
|*  2 |   HASH JOIN                              |                   |   8168 |   2
|   3 |    COLLECTION ITERATOR CONSTRUCTOR FETCH|                   |   8168 |   2
|   4 |    TABLE ACCESS FULL                     | MISHA_DEMO_INLIST |  29885 | 29885
-------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("OBJECT_ID"=VALUE(KOKBF$))
```

For some reason, Oracle assumes that collection will contain 8168 distinct values. Because of this, the estimation uses a full table scan, which is obviously wrong. There are a couple of ways to tell Oracle that it may be wrong using this default, either directly with a CARDINALITY hint or using a DYNAMIC_SAMPLING hint. Either way, the resulting execution plan will be the same. Oracle will suddenly recognize the index and run the query as expected as shown here:

```
select /*+ gather_plan_statistics */ max(created)
from misha_demo_inlist
where object_id in (
    select /*+ cardinality (t 2) */t.column_value
    from table(id_tt(227011,227415)) t
    )

select /*+ gather_plan_statistics */ max(created)
from misha_demo_inlist
where object_id in (
    select /*+ dynamic_sampling (t 4) */t.column_value
    from table(id_tt(227011,227415)) t
    )

-------------------------------------------------------------------------------------------
| Id  | Operation                                | Name                 |E-Rows |A-Rows
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                      |       |      1
|   1 |  SORT AGGREGATE                          |                      |     1 |      1
|   2 |   NESTED LOOPS                           |                      |       |      2
|   3 |    NESTED LOOPS                          |                      |     2 |      2
|   4 |     COLLECTION ITERATOR CONSTRUCTOR FETCH|                      |     2 |      2
|*  5 |     INDEX UNIQUE SCAN                    | MISHA_DEMO_INLIST_PK |     1 |      2
|   6 |    TABLE ACCESS BY INDEX ROWID           | MISHA_DEMO_INLIST    |     1 |      2
-------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
```

```
5 - access("OBJECT_ID"=VALUE(KOKBF$))
```

Summary

This paper covers only a small number of the issues that normally fall DBAs to resolve. The biggest problem is that they are also expected to be fixed by these DBAs. This is an extremely unrealistic expectation, because nine out of ten times, the real solutions lie elsewhere, either in the insufficient code or in the misused architecture. My goal was to illustrate that proper fixes indeed exist, but they require all areas of an organization's IT infrastructure to be involved, hopefully, from the very beginning of the development process at the global level. Keep in mind that a tactical approach to strategic goals has never been efficient since the dawn of human civilization, so let us not repeat this mistake over and over again.

About the Author

Michael Rosenblum is a Software Architect/Development DBA at Dulcian, Inc. where he is responsible for system tuning and application architecture. Michael supports Dulcian developers by writing complex PL/SQL routines and researching new features. He is the co-author of *PL/SQL for Dummies* (Wiley Press, 2006), contributing author of *Expert PL/SQL Practices* (APress, 2011), and author of a number of database-related articles (IOUG Select Journal, ODTUG Tech Journal) and conference papers. Michael is an Oracle ACE, a frequent presenter at various Oracle user group conferences (Oracle OpenWorld, ODTUG, IOUG Collaborate, RMOUG, NYOUG, etc), and winner of the ODTUG Kaleidoscope 2009 Best Speaker Award. In his native Ukraine, he received the scholarship of the president of Ukraine, a Master of Science degree in Information Systems, and a diploma with honors from the Kiev National University of Economics.

Data Distribution and Consolidation Using Database Replication

Sujith Kumar, Technologist and Jeffrey Surretsky, Solutions Architect, Dell Software

In today's fast-paced mobile age, data continues to accrue by leaps and bounds. To support strategic, operational and tactical business decisions, organizations need effective data management that enables them to both consolidate data from multiple sources and distribute data to multiple targets in real time. For example, a department store chain could consolidate and analyze sales data from geographically dispersed stores to provide valuable insight for inventory management, and it could use data distribution to send selective data updates based on demographics to individual stores in order to increase sales.

Of course, adding these data distribution and consolidation capabilities should not impose a financial burden on enterprises or a strain on IT organizations. The key is to enable enterprises to distribute or consolidate data cheaply and efficiently, while making it easy for IT departments to implement and manage the systems.

There are a few database replication technologies that deliver efficient, low-latency data replication that enables the real-time data distribution and data consolidation organizations need today.

Data Consolidation

Enterprises are faced with an explosion of data from multiple applications, departments and domains across the enterprise, and even from business units or departments spread around the globe. All too often, that data comes in a variety of formats that lack interoperability. Data consolidation enables enterprises to establish a central database that can store data in an application-neutral format for use by employees worldwide. Organizations can also consolidate data from multiple geographically dispersed satellite databases into a central database or repository that can be used for real-time reporting, business intelligence and business analytics. For example, a well-known online meeting hosting company based in California uses database replication to consolidate data from multiple databases located in different data centers across North America so that customer billing information can be accessed from one database.

In addition, data consolidation can help organizations improve efficiency and reduce IT costs. Databases are a major component of the operational IT budget, requiring both staff (database administrators or system administrators) and hardware (such as database servers and storage).

Data consolidation offers additional important benefits, including the following:

- Reduction in total cost of ownership by increasing database server utilization
- Enhanced availability, data security and compliance with company policies
- Improved application performance and data visibility
- Centralized data backup and archiving
- Better globalization of data

With a database replication tool, you can replicate data from multiple databases to a single database in a hub-and-spoke configuration, with no limit to the number of source databases. Look for a tool with the versatility that allows replication of entire tables and/or selected columns or rows within a table to the hub or central database. In addition, the data being replicated can also be transformed en-route prior to being applied to the central database.

Data Distribution

In an age where businesses are trying to find new ways to reach customers and customers crave instant gratification, having data available locally in real time can have a huge impact on revenue. Hence the need for data distribution – it's all about getting the right data into the hands of the right users, in the right place, at the right time.

Distributed database architecture typically consists of a single publisher and multiple subscribers. Data is collected in a holistically normalized manner in a central database and is selectively de-normalized and distributed in real time to multiple target databases that may be geographically dispersed. The distribution of data results in improved performance and availability at the local sites.

For example, a major U.S.-based cellular phone company uses a database replication solution to replicate user account and profile data from a central database to multiple satellite databases or subscribers located in different data centers. This allows users and vendors to access localized data, resulting in improved availability and performance and also enabling the company to target its marketing campaigns based on the customer demographics.

Data distribution provides additional key benefits as it:

- Enables localization or compartmentalization of data
- Simplifies growth or expansion due to targeted data distribution
- Minimizes IT infrastructure requirements
- Improves reliability and performance through data localization
- Mitigates risk of data breaches since the data is highly compartmentalized

You should be able to configure your database replication solution to replicate data from a single central database or publisher to multiple target databases or subscribers at the lowest possible cost. In addition, send targeted data to each subscriber. Once the data is selectively provisioned and sent to each subscriber, any new updates to the central database are selectively sent to all the subscribers in real time. Your database replication solution should be capable of instantiating new subscribers or refreshing existing subscribers in case of failures without any impact on the central database.

Conclusion

Business leaders often talk about data streams. In the coming years, these streams will become rivers. As data continues to accrue by leaps and bounds, our ability to navigate the data will be a far more important differentiator from competitors than virtually any other factor.

In particular, when organizations invest in high-availability databases, optimizing for performance and transactional processing, the data they store must be simultaneously available for decision support. Downtime is no longer an option. Making systems that improve fault tolerance and concurrent access is as critical as the databases themselves. Ensuring continuous high availability is possible with powerful data replication solutions and innovative approaches to data management. Choosing the right database replication solution for your business is as important as the company behind it.

About the Authors

Sujith Kumar is a chief technologist, with more than 20 years of experience in a variety of database technologies and has been working on SharePlex for the last 15 years. Sujith was involved in architecting custom data management solutions for Fortune 500 companies. As a CTO he was involved in acquisitions and mergers of other software companies. Sujith holds a M.S. in Engineering from Texas A&M University and a B.S. in Engineering from Bangalore University.

Jeffrey Surretsky formerly a DBA, has been working at Quest Software, now a part of Dell, for over 13 years as a Strategic Systems Consultant and Solutions architect where he has focused on their database solutions. His expertise encompasses technologies around high availability, migrations, and database performance of relational database management systems (RDBMS) such as Oracle, SQL Server, DB2 and Sybase.

About Dell

Dell Inc. (NASDAQ: DELL) listens to customers and delivers worldwide innovative technology, business solutions and services they trust and value. Dell offers a variety of IT management software products, including SharePlex® for Oracle. SharePlex for Oracle is a mature, high-performance, high-availability database replication technology that offers a low cost alternative to other Oracle database replication tools. Unlike other solutions, SharePlex provides data compare and repair, in-flight data integrity, plus monitoring and alerting functionalities – all in a comprehensive packaged solution. For more information, visit www.dell.com and www.quest.com/shareplex.
If you have any questions regarding your potential use of this material, contact:

Dell Software
5 Polaris Way
Aliso Viejo, CA 92656
www.dell.com

SIGS, SIGS and more SIGS!

The following Special Interest Groups (SIG) hold meetings throughout the year for the benefit of NYOUG members:

DBA SIG – Database Administration
Data Warehouse SIG – Business Intelligence
Web SIG – Web / XML / Java / Weblogic / APEX / Fusion
Long Island SIG – Nassau/Suffolk area - All topics

ORACLE ACCELERATION

YEAH, IT'S KIND OF LIKE THAT FAST!

- Accelerate Oracle databases up to 10x
- Deploy with minimal disruption to operations
- Extend the life of your IT infrastructure

Put Your Big Data on the Fast Track.

The GridIron Systems TurboCharger™ data acceleration appliance seamlessly integrates into your existing IT environment without changes to applications, databases, servers, storage or operational processes.

Learn more at **www.gridironsystems.com/oracle**.

GRIDIRON
Big Data. Fast

NYOUG 2013 Sponsors

The New York Oracle Users Group wishes to thank the following companies for their generous support.

datAvail (www.datavail.com)
Dell/Quest Software (www.quest.com)
Oracle (www.oracle.com)

Contact Caryl Lee Fisher for vendor information, sponsorship, and benefits