ORACLE MAGAZINE

NOVEMBER/DECEMBER 2017

ORACLE AND EMERGING TECH

LET US TAKE YOU INTO THE FUTURE

LOOK WHAT'S TALKING

MEET THE NEXT-GENERATION SPARC

KEEP YOUR NODE.JS PROMISES

ORACLE®

Innovation arrives this Season with the Fujitsu SPARC M12

FUJITSU

Details

shaping tomorrow with you

CONTENTS

ORACLE MAGAZINE

28 Let Us Take You into the Future

38
Look What's Talking

8 MashUp

FEATURES

UP FRONT

CONTENTS

COMMUNITY

TECHNOLOGY

ORACLE MAGAZINE

EDITORIAL

Editor in Chief Tom Haunert

Managing Editor Jan Rogers

Editorial Director Robert Preston

Contributing Editors and Writers Blair Campbell, Leslie Steere

Copy Editors Claire Breen, Eva Langfeldt, Karen Perkins

DESIGN

Vice President, Brand Creative Francisco G Delgadillo

Design Director Richard Merchán

Senior Designer Arianna Pucherelli

Senior Production Manager Sheila Brennan

Designer Jaime Ferrand

Production Designer Kathy Cygnarowicz

PUBLISHING

Publisher and Audience Development Director Karin Kinnear

Audience Development Manager Jennifer Kurtz

ADVERTISING SALES

Western and Central US, LAD, and Canada
Tom Cometa +1.510.339.2403

Eastern US and EMEA/APAC Mark Makinney +1.805.709.4745

Mailing-List Rentals Contact your sales representative

SUBSCRIPTION INFORMATION
Subscriptions are complimentary for qualified individuals who complete the subscription form.

MAGAZINE CUSTOMER SERVICE
Omeda Communications

PRIVACY
Oracle Publishing allows sharing of its mailing list with selected third parties. If you prefer that your mailing address or e-mail address not be included in this program, contact Customer Service at oracle@omeda.com.

Data Quality Made Easy. Your Data, Your Way.

Melissa provides the full spectrum of data quality to ensure you have data you can trust.

We profile, standardize, verify, match and enrich global **People Data** – name, address, email, phone, and more.

Our data quality solutions are available on-premises and in the Cloud – fast, easy to use, and powerful developer tools and plugins for the **Oracle® Ecosystem**.

View Video

ORACLE PEOPLESOFT | ORACLE JD EDWARDS | ORACLE E-BUSINESS SUITE | Java | ORACLE Forms | ORACLE PL/SQL

Start Your Free Trial www.melissa.com/oracle

Melissa Data is now Melissa.
See What's New at www.Melissa.com

1-800-MELISSA

melissa™
GLOBAL INTELLIGENCE

Tom Haunert

Emerged. Emerging. Pervasive.

Emerging technologies may be tough to define, but you know them when you see them . . . and see them, and see them . . . .

Emerging technologies aren't just any old tech innovation. They're unique in that they underpin a radical new way of doing things, they're constantly evolving, and they're ever more pervasive.

One way to confirm that a tech innovation is also a true emerging technology is to watch industry activity and mainstream (nontech) media coverage of that tech. Both new and established businesses deliver emerging tech in the form of new products and services.

And while some emerging technologies, such as blockchain, are relatively new, others, such as artificial intelligence (AI), have been around for decades. That's because there's no expiration date after which a radical new technology becomes just a technology. In fact, it's a testimony to the technology's potential and value when an emerging tech (such as AI) stays top of mind and conversation decades after its introduction.

Perhaps the easiest way to see the opportunity in emerging technologies is to follow who is using them and how *today*. Take machine learning. If you've bought a product online in the last few years, chances are good that machine learning was involved in targeting ads, providing recommendations, and more. If online retailers are seeing the benefits today, look for them to expand their use of machine learning very soon.

EMERGING INFORMATION

If you attended Oracle OpenWorld 2017 in San Francisco, California, in October, you likely attended sessions and events that covered emerging technologies. For example, Oracle announced new Oracle Cloud Platform services for both AI (Oracle Artificial Intelligence Platform Cloud Service) and blockchain (Oracle Blockchain Cloud Service).

In this issue's *Oracle Magazine* cover story, "Let Us Take You into the Future", Michael Hickins explores some of the emerging technologies—including AI, blockchain, chatbots, and Internet of Things—supporting current and upcoming Oracle products, including Oracle Adaptive Intelligent Apps and Oracle Mobile Cloud Enterprise.

Chris Murphy's "Look What's Talking" story dives deeper into the power of Oracle Mobile Cloud Enterprise and how its new bot builder technology builds chatbots that leverage other emerging technologies.

There's far too much to say about far too many emerging technologies in Oracle products to fit in a single issue of *Oracle Magazine*, so look for expanded coverage in upcoming issues.

Tom Haunert,
Editor in Chief

Emails and posts received by *Oracle Magazine* and its staff may be edited for length and clarity and may be published in any medium. We consider any communications we receive publishable.

PHOTOGRAPH BY
BOB ADLER/THE VERBATIM AGENCY

NEXT STEPS

WATCH Oracle President Thomas Kurian discuss emerging technologies.

TRY Oracle Cloud services.

LEARN more about Artificial Intelligence.

Resolved

Learn about robots, predict AI, and test IoT.

Explore and Save Space

Explore and capture the night sky from the palm of your hand—Tiny1 is powerful enough to image deep sky objects but compact enough to fit in your pocket. The camera has a searchable augmented reality map that guides you to and identifies stars and constellations, and it connects with your smartphone to share photos and videos directly to social media. Tiny1 is available for preorder at US$479; delivery is expected December 2017. tinymos.com

PHOTOGRAPHS COURTESY OF TinyMOS

Future Benefits of Artificial Intelligence

How do you think AI will most substantially benefit humankind in the future?

It seems no opinion holds a majority, but many respondents anticipated advancements in science and medicine. Others predicted the automation of dangerous or tedious tasks, while the rest hoped for a cheaper, safer, and more convenient world.

37%
Advancements in areas that help humans, for example medicine

29%
Tedious or dangerous tasks being done by robots

19%
Lower business cost or increased capability leading to better service and lower prices for consumers

11%
Less chance of human accidents/mistakes

5%
More free time for humans

Source: ARM Artificial Intelligence Survey

DO YOU SPEAK TECH? QUIZ YOURSELF!

1. **Who coined the term *Internet of Things*?**

   A. Vinton Cerf
   B. Kevin Ashton
   C. Alan Turing
   D. Shiva Ayyadurai

2. **Which of the following was *not* a previous name for the Internet of Things?**

   A. Augmented Computation
   B. Pervasive Computing
   C. Ubicomp
   D. Ambient Intelligence

3. **What day is celebrated globally as Internet of Things Day?**

   A. June 24
   B. April 9
   C. February 11
   D. November 4

Answers: 1. B: Kevin Ashton; 2. A: Augmented Computation; 3. B: April 9

APPS: RESOLUTIONS

Every new year brings with it a raft of resolutions. Resolve to communicate more dynamically, read up on the things you love, and save smarter with these awesome apps.

Trove

Modernize your inbox with this artificially intelligent email app. Trove learns to prioritize the emails that matter most to you, contextualizes your contacts (by responsiveness, frequency of communication, and so on), and identifies unanswered questions within emails so important messages don't slip through the cracks. Compatible with Amazon Alexa and Microsoft Cortana, its voice control capacity makes for an effortless experience in true AI fashion. Free (Android, iOS)

Pocket

Remember that great story you just didn't have the time to read the other day? Now you can read it—not only whenever, but also wherever. Pocket saves content across all your devices in a format that you can access without the internet: simply pull an article or video from your browser, email, or apps such as Twitter and Flipboard for later consumption at your own convenience—offline. Free (Android, iOS)

Digit

Get those finances in order in 2018, without even having to think about it. Every day Digit analyzes your spending habits and income history, and then uses an algorithm to automatically move money from your checking account to your Digit account if you can afford it—a smart piggy bank of sorts. The moved money can be easily withdrawn at any time. Free (Android, iOS)

Bringing People and Technology Together to Support Strategic IT Initiatives

Fujitsu Builds Success Through Collaboration and Innovation

By Osamu Sekihata, Director of Global Strategic Alliances, Global Marketing

Osamu Sekihata, Director of Global Strategic Alliances, Global Marketing, Fujitsu

FUJITSU

For more information, visit
www.fujitsu.com/sparc/

Enterprise organizations in every industry are experiencing an accelerated change of pace and Fujitsu is committed to helping these organizations take advantage of the new opportunities that are emerging each day. The new Fujitsu SPARC M12 servers are just the latest milestone in an ongoing commitment to meeting modern enterprise computing demands. Built to support today's digital transformation initiatives, the Fujitsu SPARC M12 servers reflect Fujitsu's understanding of digital transformation, and its focus on supporting new IT priorities.

Not so long ago, IT focused primarily on "keeping the lights on" while controlling operational costs and risks. However, private cloud, big data, the Internet of Things (IoT), and other disruptive technologies are redefining IT expectations. These new technologies are empowering IT teams with the agility they need to support innovation and drive business growth. IT staff are embracing more strategic roles while managing new risks and an increase in data volume from connected people and things. Fujitsu is helping them achieve this.

How has Fujitsu delivered on these key requirements for forward-looking enterprises? It starts with a collaborative culture that values teamwork and innovation. This level of collaboration also extends to Fujitsu's rich relationships with other technology leaders such as Oracle.

COLLABORATION SPARKS INNOVATION INSIDE FUJITSU

Performance and reliability are key to driving faster innovation and business agility in the enterprise. To deliver these capabilities in the processor and beyond, Fujitsu's server developers are constantly exchanging ideas among their development teams across platforms. These developers build processors for mainframes, supercomputers, and UNIX® servers, bringing the best benefits from each platform to SPARC64™ XII processor development.

Fujitsu's firmware development teams work closely with hardware teams to enhance the processing power and reliability of its server systems. Different types of machines often prioritize different types of functions. For example, in supercomputers, the highest priority is placed on performance per unit of power. Reducing the processing power required to achieve performance goals is the most important requirement for the system because a supercomputer requires huge numbers of nodes.

However, for UNIX servers used in business applications, the highest priority is placed on performance and business continuity. The hardware is monitored and controlled primarily through the service processor (XSCF) and steps are taken to ensure that the system remains up and running. Fujitsu's culture encourages information sharing between teams from different platforms and different areas of the solution. The end result of this close collaboration is an enterprise server that provides not only the highest possible performance, but exceptional reliability.

As part of its solution development, Fujitsu has a dedicated team that focuses on the hardware platform that houses the core processors. Working closely with the processor developers, this platform's team drives innovation in the Fujitsu SPARC M12's cooling technologies. The new Vapor and Liquid Loop Cooling (VLLC) system offers twice the cooling performance over the Liquid Loop Cooling (LLC) system used in the Fujitsu M10 servers. VLLC achieves significantly higher cooling performance through the phase change of liquid to vapor, enabling superior heat absorption. Since VLLC technology minimizes the need for heatsinks, CPUs and memory can be packed more closely together. The outcome is reduced latency and a more powerful performance for key enterprise applications.

"The Fujitsu SPARC M12 server is the result of more than 30 years of close teamwork between these two tech giants, bringing Fujitsu's advanced system technology together with the Oracle Solaris 11 enterprise OS."

—Osamu Sekihata, Director of Global Strategic Alliances, Global Marketing

BUILDING ON A LONG-TERM ORACLE RELATIONSHIP

The collaborative spirit at Fujitsu extends well beyond its internal development groups. The company maintains a long-standing tradition of working with Oracle to produce SPARC systems designed to address the demanding, mission-critical requirements of enterprise infrastructures. The Fujitsu SPARC M12 server is the result of more than 30 years of close teamwork between these two tech giants, bringing Fujitsu's advanced system technology together with the Oracle Solaris 11 enterprise OS.

Private cloud management is highly accessible because OpenStack® is integrated in the robust Oracle Solaris operating system to deliver open standards-based private cloud. Using Oracle Enterprise Manager Cloud Control and Oracle Enterprise Manager Ops Center, IT can take advantage of a powerful array of software tools to build and manage a private cloud system using familiar Oracle technology.

The collaborative effort between Fujitsu and Oracle especially shined during the development of the Fujitsu SPARC M12 Software on Chip (SWoC). Software on Chip is a collection of technologies that accelerate the processing speed of specific processes that were previously handled by software. It achieves this high performance through specialized logic that resides directly in the processor itself. Implementing these functions required support from Fujitsu's processor, operating system, and the Oracle Database development teams.

Fujitsu gains strength—and ultimately results—from this close and face-to-face collaboration of hardware and software developers. The presence of the Fujitsu campus in Silicon Valley is also extremely beneficial, enabling Fujitsu developers to smoothly interact with their counterparts nearby at Oracle. Although remote meetings and videoconferencing are easier than ever, face-to-face collaborations are still the most effective way to achieve faster, better results through rapid feedback and deeper understanding of one another.

Fujitsu and Oracle will continue to work hand-in-hand with each other to continue to enhance Fujitsu SPARC technology. The two companies have a distinct vision for the future, and they have each charted their own course by developing their corresponding roadmaps. Customers can expect new capabilities, such as additional SWoC enhancements, in the years ahead.

THE COMMITMENT AND CULTURE FOR FUTURE SUCCESS

Collaboration and innovation are a deep part of Fujitsu's vision, and these qualities are deeply woven into the fabric of the company at all levels. It starts with inspiring executive leadership, and is executed every day by persistent, dedicated employees who are committed to staying ahead of the competition.

Ultimately, Fujitsu aims to empower people through technology, contribute to a Human Centric Intelligent Society, and further its vision for a safer, more prosperous and sustainable world. Fujitsu takes a bold step in this direction with the technology of the Fujitsu SPARC M12, which enables real-time connections between people and businesses. The close collaboration between Fujitsu and Oracle creates new value and supports customers.

Fujitsu is proud of more than 60 years of experience in computing technology development, and will continue to build on its rich tradition in the years ahead. The Fujitsu SPARC M12 servers are simply the latest chapter in an ongoing story that's continuing to unfold.

INTERVIEW

"Software in Silicon capabilities are an answer to the question of how customers can improve business processes," says Marshall Choy, vice president of systems product management at Oracle.

Meet the Next-Generation SPARC

New SPARC M8 processing power and Software in Silicon features bring better performance and security. **BY TOM HAUNERT**

On premises or in the cloud, hardware and software typically align with different products and services. Compute power from a data center system processor or a public cloud service delivers processor core frequency, while security and programming language–specific operations are most often delivered through software.

SPARC M8 and its Software in Silicon technology bring high-performance processing and software features to the processor, and the new processor is at the core of updated SPARC M8 systems, including three different SPARC T8 servers, a SPARC M8 server, and the SPARC SuperCluster M8 engineered system. And SPARC M8 will be the processor power behind Oracle Cloud Infrastructure Dedicated Compute Classic - SPARC.

Oracle Magazine sat down with Marshall Choy, vice president of systems product management at Oracle, to talk about the new SPARC M8 processor and how it bridges the hardware/software divide.

Oracle Magazine: What are the key hardware improvements in SPARC M8?

Choy: SPARC M8 includes a new core design with 32 5.06 GHz processor cores, which effectively yield a 50 percent improvement in single-thread performance over SPARC M7. SPARC M8's memory bandwidth runs at up to 185 gigabits per second, which is a 16 percent improvement over SPARC M7. And SPARC M8 has reduced memory latency, compared to SPARC M7, by almost 10 percent.

Oracle Magazine: How does SPARC M8 performance compare to x86 and SPARC M7?

Choy: SPARC M8 delivers 2 times faster OLTP [online transaction processing] performance per core than x86, 1.4 times faster than SPARC M7 microprocessors, and up to 7 times faster database analytics than x86. SPARC M8 produces 2 times better Java performance than x86 and 1.3 times better than SPARC M7 microprocessors, as well as 8 times more efficient Java streams processing.

Oracle Magazine: Like SPARC M7, SPARC M8 includes Software in Silicon technology. What is it, and why is it important?

Choy: Software in Silicon capabilities are an answer to the question of how customers can improve business processes. With Software in Silicon, we looked at how we could optimize existing processes in silicon.

When we first started developing the SPARC M7 about seven years ago, some of the big problems we forecasted and were going to need to solve were things like analytics and security. Software in Silicon addresses these things—things that are either expensive or difficult to do in software—and implements them in the microprocessor with much greater efficiency.

The eighth-generation SPARC M8 includes Software in Silicon version 2. First launched in SPARC M7, Software in Silicon provides security protections and accelerates a variety of critical business tasks. SPARC M8 security includes always-on hardware-based memory protection and end-to-end encryption. Data Analytics Accelerators deliver breakthrough performance for running database analytics and Java streams processing.

Oracle Magazine: What has changed for Data Analytics

"With SPARC M8, we've doubled the security performance over SPARC M7," says Marshall Choy, vice president of systems product management at Oracle.

Accelerators [DAX] in Software in Silicon version 2 in SPARC M8?

Choy: We've increased the clock rate on the Data Analytics Accelerator cores and, therefore, the ability to process with Data Analytics Accelerators. We've also implemented open application programming interfaces [APIs] for DAX, so the use cases for Software in Silicon version 2 have expanded significantly from the initial release of the DAX in Software in Silicon in SPARC M7. This API feature in Software in Silicon version 2 enables the 8 times more efficient Java streams processing I mentioned earlier.

We've also added some new functionality called Oracle Numbers, which provides consistent precision floating-point calculations. We have Oracle Numbers accelerators on the SPARC M8 processor, which help to further accelerate database analytics performance.

Oracle Magazine: What's new in security for SPARC M8?

Choy: Software in Silicon version 2 on SPARC M8 continues to include silicon secured memory, which protects against situations such as invalid and stale references, buffer overflows, and buffer overreads. It protects against the approaches used in the Venom and Heartbleed exploits of a couple of years ago.

Software in Silicon version 2 on SPARC M8 also continues to include on-chip encryption that comes with minimal performance impact. We've enhanced the range of cryptography ciphers on the SPARC M8 processor. We support AES ciphers and hash functions such as SHA [Secure Hash Algorithm]—SPARC M8 is the only processor in the industry to natively embed SHA-3.

The performance that we can deliver for these types of security enhancements compared to SPARC M7 is a generational improvement. With SPARC M8, we've doubled the security performance over SPARC M7.

PHOTOGRAPHY BY
BOB ADLER/THE VERBATIM AGENCY

NEXT STEPS

LEARN more about Oracle's SPARC M8.

LEARN more about Oracle's SuperCluster M8 engineered system.

WATCH the Oracle's SPARC M8 launch webcast.

BY STEPHEN CHIN, ORACLE DEVELOPER COMMUNITY DIRECTOR

Name That Community

Developers drive change, and change, in turn, drives developers. A diverse population of software developers is driving change at Oracle, and Oracle has changed the Oracle Technology Network name to Oracle Developer Community and added developer-focused programs, content, social channels, and more. Changes to the new community are ongoing.

Update Your Addresses

Oracle Developer Community includes the new Oracle Developer Champion program, featured in the last issue, as well as the frequently updated Oracle Developer Portal and updated content and social media channels.

WATCH THE CODE

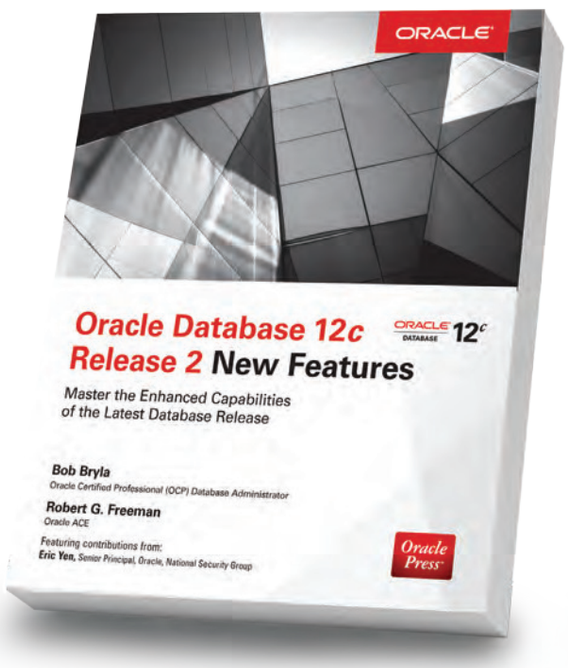
Oracle Code, a series of free international events for developers, added an event on October 3, 2017, colocated with Oracle OpenWorld and JavaOne. Didn't make it to any of the first 21 Oracle Code events in 2017? Watch select videos and live for the code.

ART BY WES ROWELL

Oracle Press™

Your Destination for Oracle and Java Expertise

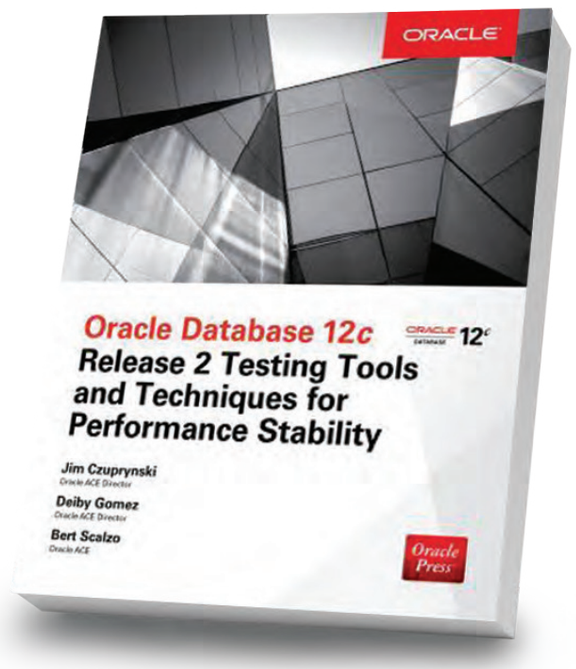
Written by leading technology professionals, Oracle Press books offer the most definitive, complete, and up-to-date coverage of Oracle products and technologies available.



Oracle Database 12c Release 2 New Features

Bob Bryla, Robert G. Freeman

Master the new and enhanced capabilities of the latest Oracle Database release.



Oracle Database 12c Release 2 Testing Tools and Techniques for Performance and Scalability

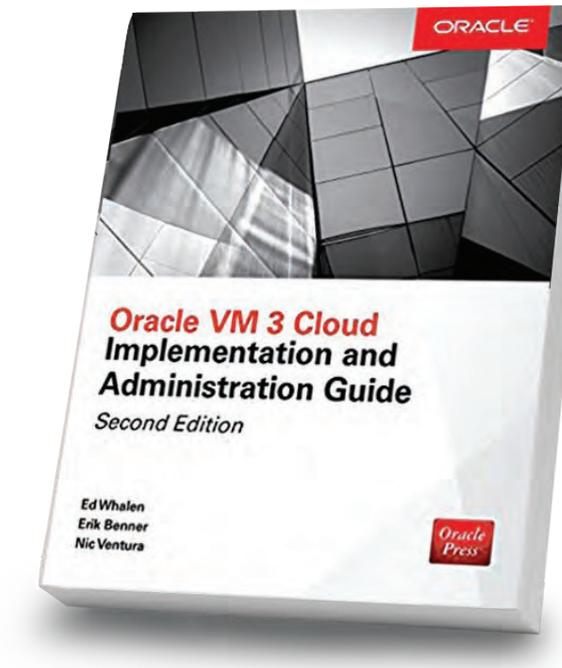
Jim Czuprynski, Deiby Gomez, Bert Scalzo

Seamlessly transition to Oracle Database 12*c* Release 2 and achieve peak performance using best practices in this step-by-step guide.

Oracle Mobile Cloud Service Developer's Guide

John Ray Thomas

Create modern, enterprise mobile apps with Oracle Mobile Cloud Service.



Oracle VM 3 Cloud Implementation and Administration Guide, Second Edition

Ed Whalen, Erik Benner, Nic Ventura

Master cloud building with Oracle VM 3 installation, configuration, and maintenance.

Available in print and ebook formats.

Oracle Press™

www.OraclePressBooks.com • @OraclePress

Getting Into Containers

Small packages. Big value.

By **Bob Rhubart**

In the continuing quest to examine the point at which trends in IT and architecture gain traction with reality, I thought it was about time to focus on containers and their use among members of the community. Are these standalone, executable packages of code, runtime, and system tools making a difference in the work of architects and developers?

Lucas Jellema, CTO at AMIS in the Netherlands, Oracle ACE Director, and Oracle Developer Champion, confesses to being strongly influenced by containers. "Containers have been a catalyst for my thinking about application and platform architectures," he says. "The importance of stateless applications in order to allow horizontal scalability and easy failover is a drastic shift from the heavy, session-oriented server-side web applications of the past 15 years."

On a personal, practical level, Jellema uses containers to make better use of the laptop he uses for R&D. "I used to quickly make a mess of my laptop with new tools, platforms, and frameworks I was experimenting with," he says. Virtual machines offered some help but were subject to his laptop's memory and hard disk limitations. "Containers are lightweight; can easily be shared with colleagues; and are simply available, ready to use for most popular technologies, saving me lots of time and anguish," he explains.

“Containers are lightweight; can easily be shared with colleagues; and are simply available, ready to use for most popular technologies.”

—Lucas Jellema, CTO, AMIS

Oracle ACE Associate Maarten Smeets, senior integration consultant with AMIS, also sees enormous value in the use of containers and distinct advantages over the alternatives, “especially large-scale application server implementations,” he notes. Containers offer relief from a number of issues, including difficulties in providing large-scale implementations when using application servers, problems with automatically provisioning and scaling stateful application servers, and the impact on availability when an entire system must be shut down to resolve certain issues.

Smeets appreciates that containers offer an additional benefit through their compatibility with Node.js, the wildly popular JavaScript runtime. “This makes JavaScript skills useful for different layers in the stack and also makes it possible to reduce the amount of translations and layers that have to be made to go from front end to back end.” It all adds up to reduced development effort, easier environment management, and optimized performance.

But despite the clear benefits, Smeets says his customers aren’t quite in step. “Most customers I work with implement application servers or database-bound integration platforms—not the most suitable match for containerization,” he says. But, he predicts, the limitations of those architectures and the benefits of using containers will force the foot-draggers to reconsider containers.

Dr. Frank Munz, Oracle ACE Director, Oracle Developer Champion, software architect, and cloud evangelist, reports that at least two of his customers had no reluctance at all to adopting containers. “Both were cloud-native projects. Both used Docker. None of them ever faced any container-specific issues,” he says. “In one project, the decision for Docker was made simply because it seemed to be part of a modern development stack. And in the

end, it turned out to be a great fit. In the other project, there was a major concern about the vendor lock-in with a particular cloud provider and therefore the decision for Docker was obvious."

Have containers become an obvious choice in your work? Join in the discussion and share your experience.

Oracle Developer Community Architect Community Manager Bob Rhubart is the host/engineer/producer of the Archbeat Podcast, produces the 2 Minute Tech Tip video series, and interviews technology experts in DevLIVE videos recorded at Oracle Code, Oracle OpenWorld, and other events.

PHOTOGRAPHY BY
MICHAEL MCELROY/THE VERBATIM AGENCY

NEXT STEPS

WATCH

No Pain, No Gain—Getting Started with Containers.

Basic Help for Docker Noobs.

Running WebLogic Applications on Docker Using Oracle Container Cloud Service.

Cloud, Emerging Tech, and the Expert

Emerging technologies such as IoT, blockchain, and machine learning require DBAs and database developers who understand a company's data.

Even as new autonomous database services take up tasks including tuning, patching, and backing up data, new technologies are emerging that scream for in-house data experts to guide the business in tackling them.

At Oracle OpenWorld 2017, *Oracle Magazine* spoke to attendees about how DBAs and database developers have always reacted when database tasks are automated and how that ethic will continue to make them indispensable as companies reach for emerging tech.

Ross Smith

CHIEF ARCHITECT AT PITSS AMERICA

"Our job as data experts is to help the company express what our legacy applications do in new forward-looking technologies. Talk to your applications teams and show them alternatives like NoSQL and new cloud services."

Francis Mignault

CTO AT INSUM SOLUTIONS

"The DBA job has evolved to take on other things. And that's good, because DBAs can now focus on what the business is trying to do and what it needs."

Martin D'Souza

ORACLE ACE DIRECTOR

"Go learn about cloud and all the technologies around it. If you understand as a DBA how the data is stored, how it's indexed, and where it lives, you can help identify cloud services that can solve a business problem [cheaply]."

BY BLAIR CAMPBELL

Talking It Out

On a project, on social media, or quite literally in the field, communication is key.

Robert P. Lockard

Glen Burnie, Maryland

ORACLE ACE Director

Company: Oraclewizard

Job title: President

Length of time using Oracle products: 29 years

How did you get started in IT? After high school, I joined the US Navy and served as an aviation anti-submarine warfare technician. The Navy provided me with skills that I still use today to troubleshoot complex problems. When I left the military in 1982, I had quite a bit of difficulty, eventually becoming severely depressed. Because of my fascination with computers, my counselor and parents recommended that I start taking programming and computer science classes. So I went back to school and wound up working for Canon USA in PC support, and then as an IDMS programmer. One day my boss announced that my new career path would be with Oracle technology. He declared me the company's new Oracle wizard and handed me a stack of Oracle tapes.

What is your go-to Oracle reference book? That's easy: *Oracle PL/SQL Programming* by Steven Feuerstein [O'Reilly Media, sixth edition, 2014], which is sitting on my desk right now, dog-eared and marked up.

What's the most common cause you see when IT projects go wrong? Blaming the messenger. When things go wrong—and they will go wrong—some people blame others as opposed to focusing on the issue and correcting it. Large IT projects require communication, coordination, and respect across a team.

Rodrigo Radtke de Souza
Porto Alegre, Brazil

ORACLE ACE

Company: Dell Inc.
Job title: Principal software engineer
Oracle credentials: Oracle Data Integrator 11*g* Certified Implementation Specialist
Length of time using Oracle products: More than 10 years

What's your favorite tool on the job? I work heavily on data integration processes for EPM [enterprise performance management] tools, and there's no better tool for my job than Oracle Data Integrator. It's really a complete development suite where you can create your own code for very specific situations. It's also a complete execution platform where you can manage and orchestrate any kind of complex execution logic, and it fits well with any kind of project out there—from data warehousing and real-time data replication to data migration.

What advice do you have about how to get into database development and software architecture? If you want to find success as an ETL [extract, transform, and load] developer and architect specifically, I advise four steps. Pick an ETL tool and study it deeply; learn and master SQL, since it's the key to achieving performance in 99.99 percent of ETL projects; learn to code in a computer language, such as Java, which will allow you to extend your ETL tool's capability; and gain a clear understanding of how business data processes work.

How are you using social media in your work these days? Social media was a game-changer for me when I started to use it for work purposes. I'm pretty active on Twitter, on LinkedIn, and on my blog. The amount of knowledge out there is just amazing. I've made some great "virtual" friends over the years, and it was through social media that I also found out about ODTUG, got involved presenting technical sessions at ODTUG's Kscope conference, and learned about the Oracle ACE program.

Brian Peasland
Fargo, North Dakota

ORACLE ACE

Company: NAU Country

Job title: Principal database engineer

Length of time using Oracle products: 20 years

Which new capabilities in Oracle Database are you currently finding most valuable? One of my favorites is Oracle Database 12*c* Release 2's real-time materialized views. Since we started implementing real-time materialized views, we've gone from a small handful of materialized views—6 of them in production—to more than 30, and we're only just beginning.

What technology has most changed your life? Most recently, Oracle Exadata Multitenant is changing the way we do business to rapidly deploy clones of our production database. Prior to Oracle Exadata Multitenant, we ran 30 clones of our production database in virtual machines for our DevTest efforts. We always had to battle for more resources to host more clones. With Oracle Exadata Multitenant, we were able to stop duplicating the overhead of the OS and the database instance. We're now on track to host up to 100 clones of our production database without increasing the physical resources for our Oracle infrastructure.

How are you using mobile computing in your work these days? My company is a federal crop insurance provider, so our customers are farmers in the field. Our claims adjusters are with our customers in the field. Our agents are selling policies in the field. So clearly it's imperative that we have mobile applications for end users who are literally out in the field.

World's First "Self-Driving" Database

Oracle Autonomous Database

No Human Labor
Half the Cost

No Human Error
100x More Reliable

ORACLE®

LET US TAKE YOU INTO THE FUTURE

Artificial intelligence, autonomous databases, blockchain, chatbots, Internet of Things, and more: Oracle executives discuss emerging technologies.

BY MICHAEL HICKINS

Those of us who have been waiting for the future to arrive since, well, forever can finally rejoice. The future has arrived in the form of computer interfaces that feel human; self-learning software; self-healing autonomous databases; sensors that provide us with actionable, critical data from far afield; hyper-realistic visualization technology; and transaction systems that are simultaneously private and transparent.

What's most surprising, though, is that the experience of the future is incredibly easy for Oracle's customers to deploy. At this year's Oracle OpenWorld in San Francisco, California, Oracle leaders announced new products and discussed upcoming capabilities that put the power of several emerging technologies into the hands of Oracle customers.

Enterprise-Class Artificial Intelligence

Artificial intelligence (AI) is the most fundamentally important and pervasive of emerging technologies. Larry Ellison, executive chairman and CTO at Oracle, calls *machine learning*, a subset of artificial intelligence, "every bit as revolutionary as the internet."

Oracle is using AI, for instance, to power its autonomous database and chatbots (more on those to follow), and to turn the trillions of data points streaming from connected devices into actionable information.

"For years and years and years, artificial intelligence did not live up to its promise," Ellison said during his opening keynote address at Oracle OpenWorld on October 1, "but there is a new type of AI."

This new type is tuned specifically for business purposes. It is intended to replace manually coded business rules with AI-based "learning algorithms," noted Thomas Kurian, president of product development at Oracle, during his Oracle OpenWorld keynote. Kurian said Oracle is introducing both Oracle Artificial Intelligence Platform Cloud Service, an AI platform that lets customers build their own machine learning and AI algorithms, and domain-specific AI algorithms optimized for financial, HR, marketing, and other types of enterprise software.

For instance, Oracle Adaptive Intelligent Apps embed AI capabilities directly into Oracle Enterprise Resource Planning Cloud, Oracle

Human Capital Management Cloud, Oracle Supply Chain Management Cloud, and Oracle Customer Experience Cloud. The apps can react, learn, and adapt in real time based on historical and dynamic customer data, helping organizations improve business processes and user experiences.

In addition, third-party data feeds into Oracle Adaptive Intelligent Apps from Oracle Data Cloud, which is the largest third-party data marketplace in the world, collecting more than 5 billion global consumer and business names and more than 7.5 trillion data points monthly. Oracle customers can combine this third-party data with their own data, enabling them to make better real-time decisions about which offers to extend to customers.

Most important, Oracle Adaptive Intelligent Apps are designed to deliver immediate business outcomes based on real-world scenarios. This isn't AI in the abstract. And because the AI technology is embedded in applications that business users already know, app usage doesn't require extensive training—let alone the services of a data scientist, a business analyst, or some other technical specialist.

Business users can take advantage of the technology within their existing tools instead of having to export their data to an external AI application.

"AI has the power to be more transformative for the enterprise than any other technology in recent history," said Amit Zavery, senior vice president of product development for Oracle Cloud Platform, during his Oracle OpenWorld session on October 4.

Zavery noted that because Oracle has embedded AI into its platform-as-a-service (PaaS) offerings, customers have access to common AI libraries, machine learning frameworks, and development tools they can use to build AI code into their own custom applications.

"There aren't that many revolutionary new technologies, but this one is," said Ellison in his keynote. "And you can recognize it simply by the charisma of the applications it enables."

The "Self-Driving" Database

One of the most stunning applications of AI is Oracle's forthcoming "self-driving" database.

Oracle Database 18*c*'s self-patching and self-tuning capabilities, powered by machine

> **“We’re doing this to give you, our customers and developers, a canvas on which you can paint your vision and your ambitions and dreams, to use information technology in a new way—in a fundamentally new way—to transform your organization, your companies, and the world.”**
>
> —**Thomas Kurian**, President, Product Development, Oracle

learning, are developed to minimize human intervention and virtually eliminate human error, helping reduce security risks while freeing database managers to focus on higher-level work.

“This is a big deal, by the way. No one else does this,” Ellison said. “This is the most important thing we’ve done in a long, long time.”

Oracle Database 18*c*’s machine learning algorithms can automatically and continuously patch, tune, back up, and upgrade the system without manual intervention, all while the system is running. That construct minimizes the possibility for human error and malicious behavior.

Ellison noted that “the worst data thefts in history have occurred after a patch was available to prevent a theft. The patches just weren’t applied.” The self-patching capabilities of Oracle’s autonomous database turn that problem into an anachronism.

Oracle Database 18*c* also is developed to require no downtime window for provisioning, backup, patching, updating, and other maintenance. As such, Oracle service-level agreements will guarantee customers 99.995 percent availability, holding planned *and* unplanned downtime to an average of less than 2.5 minutes per month, or 30 minutes per year, Ellison said during his keynote.

"We have to automate our cyberdefenses. And you have to be able to defend yourself without having to take all your computer systems offline or shut down your databases," he said.

Oracle initially plans to release two cloud versions of Oracle Database 18*c*. A version for data warehouse and analytics workloads, called Oracle Autonomous Data Warehouse Cloud, should deliver two times faster in-memory column store and about 100 times faster query processing and in-memory analytics for external data. An online transaction processing (OLTP) version, called Oracle Autonomous Database Cloud, is developed to provide four times faster in-memory OLTP access. "It's [been] a four-year effort that is in beta test now," said Juan Loaiza, senior vice president of systems technology at Oracle, during a Wednesday session.

Loaiza noted that Oracle Database has already included many of the automated features of the forthcoming self-driving database—but the autonomous database requires complete automation of all parts of the database lifecycle. "If you put together your own system and configure the database yourself, we can't make it autonomous," he said. "The cloud allows Oracle to automate the entire system from disks to database, enabling a fully autonomous database."

If Oracle Database 18*c* customers choose to do so, they can hand over most generic tasks—such as configuring and tuning the database and provisioning backups and disaster recovery—to Oracle. Customers will still be able to pick specific maintenance windows so that routine maintenance doesn't slow down performance at peak times, Loaiza said.

The autonomous database will secure itself, applying the latest security updates, including off-cycle patches for "high-impact security vulnerabilities," he said. And the database will monitor itself using machine learning to avoid capacity limits and other bottlenecks.

Oracle plans to deliver all versions of its next-generation database in its public cloud as well as behind customers' firewalls under its Oracle Cloud at Customer program.

Loaiza also emphasized that the autonomous database will not put database administrators out of work. While it will automate generic tasks, DBAs will still be required to do data modeling, data lifecycle management, and applications-related tuning.

> **AI has the power to be more transformative for the enterprise than any other technology in recent history.**
>
> —**Amit Zavery**, Senior Vice President, Oracle Cloud Platform Product Development, Oracle

Blockchain Is the New Facilitator

There's a good reason *blockchain* is one of the most hyped technologies of the day. It promises to lower the costs associated with many types of transactions, while speeding their execution and making them more secure.

Blockchain is a data structure that enables multiple entities to share data on a shared ledger without a central authority. Each entity controls its assets using a private key and independently verifies all transactions. Transactions added to any block in the chain can be validated by multiple entities participating in that chain.

The way blockchain works, each participant in a given transaction or commercial relationship is represented by one or more nodes in a blockchain network. These nodes submit transactions on behalf of owners to the network. The blockchain network then validates and serializes the transactions and places the transactions into a block. Each block contains a cryptographic hash of the previous block, thus creating a chain of blocks. The information in a block can't be changed without detection, because the hashes stored in subsequent blocks wouldn't match.

"Blockchain can remove the need for intermediaries and replace it with cryptographically secure protocols," said Mark Rakhmilevich, senior director of product management and strategy at Oracle, during an Oracle OpenWorld blockchain session.

Blockchain addresses the problem of trust between organizations by providing a mechanism that allows multiple parties to independently validate a transaction and then record those validations and transactions in the ledger. Because each entity has its own copy of the ledger, they can independently verify that a validation was performed and by whom. And because each participant has a copy of the ledger, there is no need for reconciliation of the transactions.

In Oracle Blockchain Cloud Service, developed on the open source Hyperledger Fabric

project, the type of transaction specifies which entities need to validate the transaction, as well as how many validations are required to achieve consensus. The process typically requires only a subset of the organizations participating in the blockchain, but a sufficient number to satisfy the corroborations stipulated by the parties. This eliminates the need for every organization to ratify every transaction as is done in many other blockchain technologies.

Once a transaction has been validated, it is submitted to the ordering service that serializes the transactions and places them into a block linked to the previous block via a cryptographic hash. The transactions include not only what information was updated but also the signature of the submitter of the transaction and the signatures of all entities that validated the transaction.

Blockchain can manage common transactions, such as payment records, safety inspections, building permits, mortgage and loan records, purchase orders, and invoices. Members of a given supply chain can also use blockchain to manage their interactions with the various participants of the supply chain. For example, a food retailer can use it to check the provenance of its meats (and even see government inspection certificates). And the retailer, meat processor, and original seller can exchange fiduciary documents, validate shipments, and reconcile orders and invoices, all without needing a bank or other third party to validate these transactions.

Rakhmilevich noted that Oracle provides its cloud service as a managed PaaS, so that customers "don't have to stand up new instances for every use case." In this preassembled service, "all needed components have been provisioned," he said.

Chatbots That Know More

Chatbots are another technology underpinned by Oracle's AI technology. These seemingly human agents interact with people, both consumers and employees, like extremely knowledgeable staff members. In large part because they are animated by AI rather than strict business rules, they can understand context and learn from experience.

For example, chatbots built using Oracle Mobile Cloud Enterprise can infer earlier parts of a buyer-seller conversation, so that consumers don't have to reiterate or contextualize their queries. Integrated APIs, such as a recommen-

> **“Blockchain can remove the need for intermediaries and replace it with cryptographically secure protocols.”**
>
> —**Mark Rakhmilevich**, Senior Director, Product Management and Strategy, Oracle

dation API, give chatbots access to customer purchase histories and preferences, improving a company’s chances of selling additional goods or services.

Another API can extract data from an image sent by a consumer; recognize that it’s an article of clothing; and recommend a similar, or even the exact same, article of clothing that’s in stock.

“We could have taken the approach that many other vendors have taken: ‘We’ll build an AI platform; now you figure out what to do with it,’” said Suhas Uliyar, vice president of mobile, bot, and AI strategy and product management at Oracle. “We took a different approach. We started with AI components that are chatbot-focused, but it doesn’t end there—that’s just a beginning that allows you now to expand into other utilities. If you want to use our NLU [natural language understanding] processing outside of chatbot, feel free to use it. It’s all built in a micro-services architecture in the first place.”

The Internet of Things That Matter

The Internet of Things (IoT) is another much-hyped technology that has not delivered a lot of success thus far, Oracle’s Zavery noted. But that’s because most IoT projects are focused on technology rather than business outcomes, and “data is not connected to business insights,” he said.

Oracle’s approach is different. Oracle is embedding IoT and data analytics technology into business applications that manage specific workflows, such as those of Oracle Customer Experience Cloud (which includes sales, marketing, and customer service applications) and Oracle Supply Chain Management Cloud (which addresses issues in factory maintenance).

Oracle Internet of Things Cloud Service supports new development and is the platform behind Oracle Cloud–based IoT asset monitoring, production monitoring, connected workers, and fleet monitoring applications.

This all means "you don't have to start from scratch" when it comes to connecting devices to your business applications, Zavery said.

In the case of the IoT capabilities embedded in Oracle Supply Chain Management Cloud, they let plant and operations managers remotely monitor production lines for potential outages and have them serviced before they go down. They also include support for augmented reality software that both plant managers and field technicians can use to review metadata about different assets, as well as real-time instructions on how to take those assets apart, execute fixes, and put the equipment back together. They can also "walk" the factory floor using virtual reality (more on that to follow).

Interfaces for Humans

The leap from green screen technology to more-graphical interfaces seemed like a revolution, but that was just a skirmish in the broader effort to make computing power available to all business users.

"Our vision for the human interface for applications is to become seamless for humans," said Oracle's Kurian. "No longer is it just web and mobile screens, but you could speak to the application. You can interact with it with messaging. You can take pictures and we can identify images, compare them with other things, and automate transactions."

In the case of a plant manager "walking" a factory floor, virtual reality in Oracle Internet of Things Cloud Service can illustrate a digital twin of a malfunctioning piece of equipment and allow the manager to try different fixes without disrupting the physical device. Once the solution is found, the manager can direct a technician to apply the fix to the actual device.

That technician can also use augmented reality to superimpose metadata on the device needing repair, making it easier to apply the fix.

"We're doing this to give you, our customers and developers, a canvas on which you can paint your vision and your ambitions and dreams, to use information technology in a new way—in a fundamentally new way—to transform your organization, your companies, and the world," Kurian said.

Easy Does It

The most remarkable thing about these remarkable technologies, however, isn't that they represent singular advances that will help businesses improve cybersecurity and boost business efficiency and revenues. It's that they're remarkably easy to implement and use.

Migration tools make it easy for IT pros to move applications onto Oracle's cloud infrastructure. Oracle's platform services make it easy for developers to deploy a variety of open source and Oracle machine learning algorithms to build AI-enhanced applications. Oracle's cloud applications are themselves easier for businesspeople to use and extract value from. This means that incorporating AI into existing Oracle Cloud applications; making chatbots part of the customer and employee experience; and taking advantage of IoT, machine learning, and blockchain capabilities to revolutionize your business is more than just aspirational. It's possible—now.

Michael Hickins was the founding editor of CIO Journal *and an editor at the* Wall Street Journal. *He is currently director of strategic communications at Oracle.*

[Follow *Oracle Magazine* in the coming months as it dives more deeply into these emerging technologies. —*Ed.*]

ILLUSTRATION BY **PEDRO MURTEIRA**

NEXT STEPS

WATCH Oracle OpenWorld 2017 keynotes on demand.

Try Oracle Cloud.

Rajesh Kumar Thakur, Principal Architect, Exelon

LOOK WHAT'S TALKING

Chatbots offer a next-generation link to customers—and employees. **BY CHRIS MURPHY**

Chatbots could soon become every bit as essential to engaging with customers as mobile apps and websites are now. Customers want to use their preferred messaging channel—Facebook Messenger, Slack, SMS, WeChat, Alexa, Google Home, and the like—to get instant answers. And thanks to advancements in artificial intelligence, deep learning, natural language processing, and back-end integration, the technology is here for companies to finally deliver smart, bot-driven conversations.

"What we're seeing is this huge transformation—messaging as the next browser," says Suhas Uliyar, vice president of mobile, bot, and AI strategy and product management at Oracle. "We saw the internet wave, then we saw the mobile wave, and what we're now seeing is the messaging wave—the conversational UI, backed by artificial intelligence."

With the intelligent bot-building capability now part of Oracle Mobile Cloud Enterprise, a multichannel development, operations, and analysis platform, companies can manage their bot, mobile, and web customer experiences as one strategy. Developers can use all their existing mobile-optimized APIs built for smartphone apps to integrate their chatbots with enterprise systems—such as building a chatbot that redirects to a mobile app to capture structured data when, for example, a chat about loan rates turns to filling out the form for a loan application. (See the sidebar, "Thinking About Chatbots? Keep These Five Factors in Mind.")

And businesses eager to improve connections with customers and employees are using Oracle Mobile Cloud Enterprise to prototype and test chatbots for everything from responding to customer billing questions to providing employees with precise maintenance and status information.

EXELON CORPORATION
Chicago, Illinois

INDUSTRY:
Energy

ORACLE PRODUCTS:
Oracle Mobile Cloud Enterprise

Exelon Adds Chatbots to "Channel Agnostic" Approach

A prime example of early chatbot adoption is Exelon, the United States' largest utility by customer count, working across the energy industry in generation, electric and gas delivery,

When launching a chatbot, access to analytics is important, says Rajesh Kumar Thakur, Exelon's principal architect for the chatbot project. "You can launch these things, but then you need to tell your business which channel is most popular or how this channel is reducing the call volume to your call center," he says.

and transmission. With six utilities including Atlantic City Electric, BGE, ComEd, Delmarva, PECO, and Pepco, Exelon delivers energy to about 10 million customers across the East Coast and in Illinois.

Exelon has built a working prototype of a chatbot that relies on natural language processing and artificial intelligence to understand conversations, with integration to back-end systems for the data needed to provide clear answers. Exelon expects to let customers use a variety of messaging platforms and digital assistants to ask questions.

Michael Menendez, vice president of IT for

> **When you develop a bot application, you need to have the microservices to feed it.**
>
> —**Rajesh Kumar Thakur**, Principal Architect, Exelon

BGE and Exelon Utilities Customer, describes the company's technology architecture as increasingly "channel agnostic," in which his teams can build once and quickly apply the capability across multiple channels, including new chat platforms and mobile and web apps. "Not only does this allow us to deliver solutions for emerging channels quickly, but it also ensures that our customers have a consistent experience however they choose to interact with us," Menendez says.

Exelon built its chatbot prototype in less than two weeks using the bot-building capability in Oracle Mobile Cloud Enterprise. By using the Oracle platform, Exelon was able to reuse the microservices developed for its mobile app and use the same APIs to securely provision the needed back-end services for this new channel. "When you develop a bot application, you need to have the microservices to feed it," says Rajesh Kumar Thakur, Exelon's principal architect for the chatbot project.

Integration is key to the chatbot being able to offer accurate answers. Exelon relies on numerous long-running, proprietary systems for billing and outage monitoring and reporting. Oracle Mobile Cloud Enterprise lets Exelon deliver data from those systems via microservices to its customer-facing channels. The company used the bot builder's dialog engine to craft the scripts for questions people might ask, and the machine learning capabilities help the bot refine responses over time, says Thakur.

Analytics Required

It's not enough just to build a chatbot, though. Having good analytics is just as essential as having a good dialog engine, because those analytics will tell marketing, operations, and customer service leaders what's working and

Thinking About Chatbots? Keep These Five Factors in Mind

IT and business-unit leaders thinking about the platform they'll need to develop, run, and analyze chatbots should consider the following five factors, according to Suhas Uliyar, vice president of mobile, bot, and AI strategy and product management at Oracle.

1. **Dealing with multiple channels.** Your tool should let you write a chatbot once that can run on many channels, such as WeChat, WhatsApp, and Facebook Messenger, as well as voice assistants such as those powered by Alexa and Siri.
2. **Having real conversations.** Developers need a dialog engine that uses natural language processing to help you write intelligent scripts at the start—that can understand and respond in conversational language.
3. **Using machine learning.** Your dialog engine also needs algorithms that can learn the many different ways customers ask questions, so it gets smarter over time.
4. **Integrating with enterprise systems.** A bot's intelligence comes from the data in systems such as customer relationship management, supply chain, enterprise resource planning, and more. Your bot needs to tap into a flexible, API-driven integration platform.
5. **Enabling bot-to-human-agent handoffs.** Your bot should be intelligent enough to know when it can't handle an end user's input and then hand off to a human agent, to ensure the best-quality experience. And the human agent should be able to hand off conversations to the bot when appropriate.

"At Oracle, we focus on making it very simple for developers to build bots, instead of having to build all these five things from the ground up. That's what we have built into our chatbot platform," Uliyar says. "I think the secret sauce of Oracle Mobile Cloud Enterprise is the sum of all its parts. The differentiator for the platform really is a packaging of all this into one cohesive solution."

> **“Not only does [a channel-agnostic approach] allow us to deliver solutions for emerging channels quickly, but it also ensures that our customers have a consistent experience however they choose to interact with us.”**
>
> —**Michael Menendez**, Vice President, Information Technology, Exelon

what isn’t with the chatbot experience.

Thakur says that access to analytics is important not only for improving the customer experience of a chatbot but also for proving the business value. “You can launch these things, but then you need to tell your business which channel is most popular or how this channel is reducing the call volume to your call center,” he says.

Oracle Mobile Cloud Enterprise includes customer experience analytics to help teams get the kind of deep insight needed to personalize engagement with end users. You want to be able to answer these very fundamental questions, Oracle’s Uliyar says: “Who is using my channel? How many users are coming through which channel? What are they doing in each channel?” And, he notes, that analysis needs to cross channels, so you know what people prefer to do on a mobile app versus a bot.

Calling on Instant Apps

Don’t expect to keep neat and tidy lines separating these bot, mobile, and web channels, however. In fact, Uliyar expects chatbots to accelerate a change in how mobile apps are built. Instead of a single mobile app, he talks about “instant apps” that can surface only one piece of an app’s capability. So, if you’re asking a chatbot about concert ticket options, and you pick one to buy, the bot can connect you to only that buying piece of the app, with your selection filled in, and not the full app with a seat selection map and a calendar of upcoming events.

Instant apps is an appropriate name for this idea, because the true driving force for chatbots circles back to the customer and employee desire for instant response. People text their friends because they want an answer *right now*, and they increasingly expect the same from companies.

Yet companies can't afford to hire enough people to provide such instant answers. And that's why demand is rising for this conversational UI that uses natural language and AI integrated with a back-end data system.

"I'm not advocating that human agents would go away by any means," Uliyar says. "There's a very good synergy between the bot and the human agent on the back end." But thanks to advances in AI, machine learning, and natural language processing, chatbots have become a practical solution to answering more of that first wave of questions.

Chris Murphy is Oracle director of cloud content.

PHOTOGRAPHY BY **EDWIN REMSBERG/ THE VERBATIM AGENCY**

NEXT STEPS

LEARN more about Oracle Mobile Cloud Enterprise.

Excellence Award Winners

Global 2017 Award-Winning Partners

accenture	CINTRA	Deloitte.	神州数码 Digital China	神州数码 Digital China	Evosys Passionate Evolutionary Friendly Meaningful
Specialized: HCM Cloud	Specialized: Database Appliance	Specialized: SaaS Cloud	Specialized: Database Appliance	Specialized: VAD	Specialized: ERP Cloud
The Hackett Group	Infosys	m-power solutions	NEXTGEN	TATA CONSULTANCY SERVICES	TIS TIS INTEC Group
Specialized: EPM Cloud	Specialized: CX Cloud	Specialized: Business Analytics Cloud	Specialized: VAD Cloud	Specialized: PaaS/IaaS Cloud	Specialized: SCM Cloud
		CALYPSO	Deloitte.		
		Oracle Cloud ISV	OPN Global Cloud Transformation		

ORACLE®

By Dan McGhan

Keep Your Node.js Promises

Here's Part 3 in a four-part series on asynchronous Node.js development.

In the previous article in this series, "Using the Callback Pattern and the Async Module", I covered using Node.js-style callbacks and the third-party Async module to execute a SQL query. While those patterns are effective, they are not the only ways of writing asynchronous code with Node.js. The `Promise` object, which became available with Node.js v0.12, provides a native means of reasoning about the success or failure of asynchronous operations.

It's important to learn about promise objects (promises) because many modules use them in their APIs. Also, they are integral to asynchronous functions, which I'll cover in the next part of this series. This article will cover the basics of promises and demonstrate how to use them to construct asynchronous applications. The last section, "Executing a Query with Promises," will show how to use promises to execute a SQL query.

To set up a test environment you can use to work though this article's examples, check out "Creating a Sandbox for Learning Node.js and Oracle Database" on the JavaScript and Oracle blog.

CREATING PROMISES

The `Promise` object in JavaScript is a constructor function that returns new promise instances. Each promise instance has two important properties: state and value. When a new promise is created, the constructor function accepts a "resolver" function with two formal parameters: `resolve` and `reject`. These parameters will become functions that can transition the state of the promise instance and provide a value. Promises start off in the pending state, and they are typically resolved when the resolver successfully finishes or rejected if an error occurs.

Here's an example demo script that shows several promises with different states and values resulting from the use of the `resolve` and `reject` functions.

```
const promise1 = new Promise(function(resolve, reject) {
  // noop
});

console.log(promise1);

const promise2 = new Promise(function(resolve, reject) {
  resolve('woohoo!');
});

console.log(promise2);
```

```
const promise3 = new Promise(function(resolve, reject) {
  reject(new Error('ouch!'));
});

console.log(promise3);
```

If you run the script above in either a web browser or Node.js, you should see output similar to the following. I used Chrome and then modified the output to fit better on the page. For now, ignore any errors related to uncaught or unhandled promise rejections. (There's more on uncaught or unhandled promise rejections later.)

```
Promise {[[Status]]: "pending", [[Value]]: undefined}
Promise {[[Status]]: "resolved", [[Value]]: "woohoo!"}
Promise {[[Status]]: "rejected", [[Value]]: Error: ouch! at …}
```

As you can see, the first promise has a status of "pending" and no value, because neither the `resolve` nor `reject` function was used. The second promise, which was resolved with the `resolve` function, is in the "resolved" state, and its value is "woohoo!"—the value passed to the `resolve` function. The last promise's state is "rejected" because it was rejected with the `reject` function, and its value is the error that was passed to `reject`.

Note that this demo script is completely synchronous. Typically, promises are resolved or rejected depending on the result of invoking one or more asynchronous APIs. Once a promise has been resolved or rejected, its state and value become immutable.

RESPONDING TO STATE CHANGES

To specify what should happen when the state of a promise instance changes, promises have two methods: `then` and `catch`. Both methods accept callback functions that may be invoked asynchronously at some point in the future.

The `then` method is typically used to specify what should happen when the promise is resolved, although it can accept a second function to handle rejections too. The `catch` method is explicitly used to handle rejections.

The callback functions passed to `then` and `catch` will receive the value passed when the `resolve` or `reject` function is invoked in the resolver. The value passed through the `reject` function should always be an instance of `Error`, but that's not enforced.

Here's an example script that uses `resolve` and `reject` asynchronously. The `then` and `catch` methods are used to define what should happen when the promise's state changes.

```
1  function getRandomNumber() {
2    return new Promise(function(resolve, reject) {
3      setTimeout(function() {
4        const randomValue = Math.random();
5        const error = randomValue > .8 ? true : false;
6
7        if (error) {
8          reject(new Error('Ooops, something broke!'));
9        } else {
10         resolve(randomValue);
11       }
```

```
12      }, 2000);
13    });
14  }
15
16  getRandomNumber()
17    .then(function(value) {
18      console.log('Async success!', value);
19    })
20    .catch(function(err) {
21      console.log('Caught an error!', err);
22    });
```

Here's what's going on in the script:

- Lines 1–14: This section of code defines a function named `getRandomNumber`, which returns a new promise instance. The resolver function uses `setTimeout` to simulate an asynchronous API call, which returns a random number. To demonstrate how error handling works, some random numbers will throw exceptions.
- Line 16: The `getRandomNumber` function is invoked and immediately returns a promise in the pending state.
- Lines 17–19: The `then` method of the promise is passed a callback that logs a success message with the resolved value.
- Lines 20–22: The `catch` method of the promise is passed a callback that logs an error message with the rejected error.

If you run this script several times, you should see success messages and the occasional error message. Did you notice how the `catch` call flows from the `then` call? That technique is called promise chaining, and I'll discuss that in more detail next.

PROMISE CHAINING

Calls to `then` and `catch` return new promise instances. Of course, these promises also have `then` and `catch` methods, which allow calls to be chained together as needed. Because these then- and catch-created promises are not created with the constructor function, they are not resolved or rejected with a resolver function.

Instead, if the function passed into `then` or `catch` finishes without error, the promise will be resolved. If the function returns a value, it will set the promise's value and be passed to the next then handler in the chain. If an error is thrown and goes unhandled, the promise will be rejected and the error will be passed to the next error handler in the chain.

Consider the following script:

```
const myPromise = new Promise(function(resolve, reject) {
  resolve(42);
});

myPromise
  .then(function(value) {
    console.log('Got a value!', value);

    throw new Error('Error on the main thread');
  })
  .catch(function(err) {
    console.log('Caught the error without standard try/catch');
    console.log(err);
```

```
    return 'woohoo!';
  })
  .then(function(value) {
    console.log('Cool, we can simulate try/catch/finally!');
    console.log(value);
  });
```

If you run this script with Node.js or in a browser, you should see output similar to the following (I used Node.js here):

```
Got a value! 42
Caught the error without standard try/catch
Error: Error on the main thread
    ...
Cool, we can simulate try/catch/finally!
woohoo!
```

As you can see, errors thrown and values returned are routed to the next appropriate handler in the chain. This fact allows Node.js developers to simulate a try/catch/finally block.

Things get more interesting when the value returned is a promise. When this happens, the next handler in the chain will not be invoked until that promise is resolved or rejected.

Here's an example:

```
function getRandomNumber() {
```

```
return new Promise(function(resolve, reject) {
  setTimeout(function() {
    const randomValue = Math.random();
    const error = randomValue > .8 ? true : false;

    if (error) {
      reject(new Error('Ooops, something broke!'));
    } else {
      resolve(randomValue);
    }
  }, 2000);
});
}

getRandomNumber()
  .then(function(value) {
    console.log('Value 1:', value);
    return getRandomNumber();
  })
  .then(function(value) {
    console.log('Value 2:', value);
    return getRandomNumber();
  })
  .then(function(value) {
    console.log('Value 3:', value);
  })
```

```
  .catch(function(err) {
    console.log('Caught an error!', err);
  });
```

With a little luck, when you run this script you should see three random values printed to the console every two seconds. If an error occurs at any point in the chain, the remaining `then` functions will be skipped and the error will be passed to the next error handler in the chain. As you can see, promise chaining is a great way to run asynchronous operations in series without running into callback hell. But what about more-complex flows?

For running operations in parallel, the `Promise` constructor function includes the `all` and `race` methods. These methods return promises that are resolved or rejected a little differently: `all` waits for all the promises passed in to be resolved or rejected and `race` waits only for the first to be resolved or rejected.

Asynchronous iteration of collections, something that's quite trivial with Async, is not so easy with promises. I'm not showing the technique here because there's a much simpler way to do this now with asynchronous functions: just use a loop!

ERROR HANDLING

I've already covered some error handling basics in the section on promise chaining. In this section, I want to explain something that often trips up folks who are new to promises.

Consider this example:

```
const myPromise = new Promise(function(resolve, reject) {
  resolve(42);
```

```
});

myPromise
  .then(function(value) {
    console.log('Got a value!', value);

    throw new Error('Error on the main thread');
  })
  .catch(function(err) {
    console.log('Caught the error without standard try/catch');
    console.log(err);
  })
  .then(function() {
    console.log('Cool, we can simulate try/catch/finally!');

    throw new Error('Ouch, another error!');
  });
```

Running this updated script should give you output similar to the following:

```
Got a value! 42
Caught the error without standard try/catch
Error: Error on the main thread
    ...
Cool, we can simulate try/catch/finally!
(node:11780) UnhandledPromiseRejectionWarning:
```

```
Unhandled promise rejection
(rejection id: 1): Error: Ouch, another error!
(node:11780) [DEP0018] DeprecationWarning:
Unhandled promise rejections are deprecated.
In the future, promise rejections that are not
handled will terminate the Node.js process with
a non-zero exit code.
```

Note that unhandled errors thrown in functions passed to `then` and `catch` are swallowed up and treated like rejections. This means an error will be passed to the next error handler in the chain. But what happens if there are no more error handlers?

The updated script above throws two errors on the main thread. The first error is handled properly by the subsequent catch handler. However, there are no error handlers after the second error is thrown. This resulted in an unhandled rejection and the warnings in the console.

Typically, when code throws errors in the main thread outside of a try/catch block, the process is killed. In the case of an unhandled rejection in a promise chain, Node.js creates an `unhandledRejection` event on the process object. If there's no handler for that event, you'll see the `UnhandledPromiseRejectionWarning` text in the output. According to the deprecation warning, unhandled promise rejections will kill the process in the future.

The solution is simple enough: be sure to handle those rejections!

EXECUTING A QUERY WITH PROMISES

Here's a practical example that demonstrates how promises can be used to execute a query. As in my previous article, this is a three-step process that must be done serially.

To use promises to execute a query, first obtain a connection to the database, then use the connection to execute the query, and finally close the connection. Each step is an asynchronous operation that must include error handling logic.

All the asynchronous methods of the driver have been configured to return a promise if the last parameter passed in is not a callback function. This makes it easy to choose the pattern you prefer.

```
const oracledb = require('oracledb');
const dbConfig = {
  user: 'hr',
  password: 'oracle',
  connectString: 'localhost:1521/orcl'
};
let conn; // Declared here for scoping purposes

oracledb.getConnection(dbConfig)
  .then(function(c) {
    console.log('Connected to database');

    conn = c;

    return conn.execute(
      'select *
      from employees',
      [], // no binds
      {
```

```
      outFormat: oracledb.OBJECT
    }
  );
})
.then(result => {
  console.log('Query executed');
  console.log(result.rows);
})
.catch(err => {
  console.log('Error in processing', err);
})
.then(() => {
  if (conn) { // conn assignment worked, need to close
    return conn.close();
  }
})
.then(function() {
  console.log('Connection closed');
})
.catch(err => {
  console.log('Error closing connection', err);
});
```

Copy the code block above to a new file, name the file `promises.js`, and run the script with Node.js. The output should match that of the previous script versions from my last article, written with Node.js-style callbacks and the Async module.

For many folks, this promise version of the script is easier to read than the previous versions. I believe that's debatable and largely dependent on each reader's understanding of promises. However, I think everyone will agree that the best option is the `async` function (`async`/`await`) pattern. I'll cover that pattern in the next (and last) article in this series.

Dan McGhan is the Oracle developer advocate for JavaScript and HTML5. He enjoys sharing the passion he's developed for JavaScript and HTML5 with others.

PHOTOGRAPHY BY
CAREY KIRKELLA/THE VERBATIM AGENCY

NEXT STEPS

LEARN more about
JavaScript and Oracle.

By Steven Feuerstein

ORACLE DATABASE

Using Dynamic SQL for Multirow Queries

Explore three dynamic SQL solutions to understand which is best for your program requirements.

Most of the SQL statements you will write in PL/SQL are *static*, which means that they are parsed when you compile the block in which you wrote the SQL statements. But sometimes you don't have all the information needed at compile time to parse your SQL statements. Perhaps the name of a column or a WHERE clause is constructed only when the program is executed. In this case, you must write *dynamic* SQL.

It's very easy to write static SQL in PL/SQL program units, and that's one of the best things about PL/SQL. It's also quite easy to implement dynamic SQL requirements in PL/SQL, especially compared to doing the same in nondatabase languages such as Java and JavaScript.

In this article, I will explore how to use PL/SQL features to execute dynamic SELECT statements that return *multiple* rows. It turns out that there are three different ways to do this:

- EXECUTE IMMEDIATE with BULK COLLECT
- OPEN FOR either with BULK COLLECT or with a loop
- DBMS_SQL either using arrays (with its own approach to bulk processing) or with a loop

Before diving into the details, however, it is incumbent upon me to offer several warnings regarding dynamic SQL and to urge you to make sure that you use the dynamic SQL features of PL/SQL only when you need them.

There are several good reasons to *avoid* unnecessary dynamic SQL:

- **Security**. Dynamic SQL opens up the door to SQL injection, which can lead to data corruption and the leaking of sensitive data.
- **Performance**. While the overhead of executing dynamic SQL has gone way down over the years, it is certainly still faster to use static SQL.
- **Maintainability**. The code you write to support dynamic SQL is *more*—literally more code—and harder to understand and maintain.

Sometimes the misuse of dynamic SQL is obvious. Consider the following:

```
FUNCTION name_from_id (id_in IN INTEGER)
   RETURN VARCHAR2
   AUTHID DEFINER
IS
   l_the_name   the_table.the_name%TYPE;
BEGIN
   EXECUTE IMMEDIATE 'select the_name
```

```
                          from the_table
                         where id = ' || id_in
      INTO l_the_name;

   RETURN l_the_name;
END;
```

The developer apparently believed that because the value of the ID *can* change, it needs to be concatenated to the SQL statement, so it has to be done dynamically. Ha! There's nothing dynamic about this query. It should be rewritten to the following:

```
FUNCTION name_from_id (id_in IN INTEGER)
   RETURN VARCHAR2
   AUTHID DEFINER
IS
   l_the_name   the_table.the_name%TYPE;
BEGIN
   SELECT the_name INTO l_the_name
     FROM the_table
    WHERE id = id_in;

   RETURN l_the_name;
END;
```

This is, of course, a simplistic example. In the real world, it can be harder to detect scenarios in which dynamic SQL can be made static. A more thorough examination

of "false" dynamic SQL will be the subject of my next article.

Once you determine that you *must* switch to dynamic SQL, you should study SQL injection and protect your database by reducing "attack surfaces." The most important steps to take are

- Whenever possible, bind values into your SQL string, rather than using concatenation.
- If you must concatenate, never trust text entered by a user. Check it for inappropriate content, such as JavaScript code or common SQL hacks such as `"; DROP TABLE...."`). Use the DBMS_ASSERT package to do some of the checking for you.

OK, so you have determined that you need dynamic SQL, and you are going to use it appropriately and safely. Let's explore options for dynamic multirow querying.

EXECUTE IMMEDIATE WITH BULK COLLECT

EXECUTE IMMEDIATE is the most popular path to dynamic SQL in PL/SQL. With it, you can execute data definition language (DDL) statements (for example, drop a table), queries, nonquery data manipulation language (DML) statements such as inserts and deletes, and even dynamically constructed PL/SQL blocks.

From the standpoint of queries, EXECUTE IMMEDIATE works much like SELECT-INTO (implicit query). With static SQL and a single row fetch, you would write SELECT-INTO code like this:

```
BEGIN
   SELECT t.my_column INTO my_variable
     FROM my_table t
    WHERE single_row_condition;
END;
```

ORACLE LIVE SQL

Oracle's Live SQL provides developers a free and easy online way to test and share SQL and PL/SQL application development concepts.

If you are fetching more than one row with SELECT-INTO, you must use BULK COLLECT to populate an array:

```
BEGIN
   SELECT t.my_column BULK COLLECT INTO my_array
     FROM my_table t
    WHERE multi_row_condition;
END;
```

And it is much the same with EXECUTE IMMEDIATE.

Note that in the following dynamic SQL code samples, I will use strings that are *not*, in fact, dynamic, in order to keep the code as simple as possible. Please heed my earlier dynamic SQL advice, which includes avoiding unnecessary usage of dynamic SQL, studying SQL injection, reducing "attack surfaces," binding values, never trusting text entered by a user, using DBMS_ASSERT, and so on.

Also note that sample code for this article is available on Oracle's Live SQL website.

Here is a sample block that uses EXECUTE IMMEDIATE with SELECT-INTO and BULK COLLECT:

```
1 DECLARE
2    TYPE name_salary_rt IS RECORD (
3       name     VARCHAR2 (1000),
4       salary   NUMBER
5    );
6
```

```
 7    TYPE name_salary_aat IS TABLE OF name_salary_rt
 8       INDEX BY PLS_INTEGER;
 9
10    l_employees   name_salary_aat;
11 BEGIN
12    EXECUTE IMMEDIATE
13       q'[select first_name || ' ' || last_name, salary
14            from hr.employees
15           order by salary desc]'
16       BULK COLLECT INTO l_employees;
17
18    FOR indx IN 1 .. l_employees.COUNT
19    LOOP
20       DBMS_OUTPUT.put_line (l_employees (indx).name);
21    END LOOP;
22 END;
```

Here is an explanation of this block, which features EXECUTE IMMEDIATE:

Line(s)	Description
2–5	Declare a custom record type for the two column values I will be retrieving.
7–8	Declare an associative array type of those record types.
10	Declare an associative array that will hold all the rows retrieved by my dynamic query.

12–16	Use EXECUTE IMMEDIATE to dynamically parse (if necessary) and execute the query. Note my use of "q" to escape single quotes within my string to keep the string itself more readable. Because I might be fetching more than one row, I use BULK COLLECT INTO to fill up my array.
18–21	Iterate through all the elements in the array and take the appropriate action (in this case, simply display the name).

OPEN FOR

The OPEN FOR syntax takes advantage of the cursor variable feature of PL/SQL. You do not, in fact, have to use OPEN FOR with dynamic SQL, as in the following:

```
FUNCTION names_for (department_id_in   IN INTEGER,
                    min_salary_in   IN NUMBER)
   RETURN SYS_REFCURSOR
IS
   l_cursor   SYS_REFCURSOR;
BEGIN
   IF department_id_in IS NOT NULL
   THEN
      OPEN l_cursor FOR
         SELECT last_name
           FROM hr.employees
          WHERE department_id = department_id_in;
   ELSE
      OPEN l_cursor FOR
         SELECT last_name
```

```
         FROM hr.employees
        WHERE salary >= min_salary_in;
   END IF;

   RETURN l_cursor;
END;
```

I take advantage of OPEN FOR in this case to simply allow myself to use conditional logic at runtime to determine which static SELECT statement should be associated with the cursor variable. I can then pass this variable back to the calling program, which is often not a PL/SQL program, such as a Java method.

But OPEN FOR is most commonly used with dynamically constructed queries. Let's see how I can use OPEN FOR this way by rewriting the previous PL/SQL block that used EXECUTE IMMEDIATE.

```
1  DECLARE
2     TYPE name_salary_rt IS RECORD (
3        name     VARCHAR2 (1000),
4        salary   NUMBER
5     );
6
7     TYPE name_salary_aat IS TABLE OF name_salary_rt
8        INDEX BY PLS_INTEGER;
9
10    l_employees   name_salary_aat;
11
```

```
12    l_cursor   SYS_REFCURSOR;
13 BEGIN
14    OPEN l_cursor FOR
15       q'[select first_name || ' ' || last_name, salary
16            from hr.employees
17           order by salary desc]';
18
19    FETCH l_cursor BULK COLLECT INTO l_employees;
20
21    CLOSE l_cursor;
22
23    FOR indx IN 1 .. l_employees.COUNT
24    LOOP
25       DBMS_OUTPUT.put_line (l_employees (indx).name);
26    END LOOP;
27 END;
```

Here is an explanation of the block, which features OPEN FOR with BULK COLLECT:

Line(s)	Description
2–10	This code is unchanged from the EXECUTE IMMEDIATE block: declare a custom record type, an associative array type of those record types, and an associative array that will hold all the rows retrieved by my dynamic query.
12	Declare a cursor variable that will hold the pointer to the dynamically constructed and opened dataset.

14–17	Instead of EXECUTE IMMEDIATE, use OPEN FOR to associate the dynamically constructed SQL statement with the l_cursor cursor variable. This cursor is also then opened.
19	With a single FETCH BULK COLLECT, retrieve all rows identified by the cursor into my array.
21	Don't forget to close the cursor! Actually, PL/SQL will automatically close the cursor when the block terminates, but I suggest you include the explicit CLOSE anyway. It shows you are paying attention—and that you are done with the cursor from this point on in your program.
23–26	Use the same loop as with EXECUTE IMMEDIATE to process the contents of the array.

There's not much of a reason to use OPEN FOR instead of EXECUTE IMMEDIATE if you are going the BULK COLLECT route. In both cases, you need to declare the array. But with OPEN FOR, you need to follow up your OPEN statement with FETCH and CLOSE. All of that is done for you automatically with EXECUTE IMMEDIATE.

With OPEN FOR, however, you have another option: you do not *need* to use BULK COLLECT to fetch all rows in a single round-trip to the SQL engine. Instead, you can do something more akin to a cursor FOR loop:

```
1 DECLARE
2    TYPE name_salary_rt IS RECORD (
3       name     VARCHAR2 (1000),
4       salary   NUMBER
5    );
6
7    l_record   name_salary_rt;
8
```

```
 9    l_cursor   SYS_REFCURSOR;
10 BEGIN
11    OPEN l_cursor FOR
12       q'[select first_name || ' ' || last_name, salary
13            from hr.employees
14           order by salary desc]';
15
16    LOOP
17       FETCH l_cursor INTO l_record;
18
19       EXIT WHEN l_cursor%NOTFOUND;
20
21       DBMS_OUTPUT.put_line (l_employees (indx).name);
22    END LOOP;
23
24    CLOSE l_cursor;
25 END;
```

Line(s)	Description
2–9	Declare a record type, a record variable for that type, and a cursor variable to point to the dynamic result set. Note that you no longer need a collection.
11–14	Open the cursor for this dynamic SELECT and assign the cursor variable the pointer to the dataset.

16–22	Loop through the result set using the same FETCH and EXIT WHEN statements you would use with explicit cursors.
24	Close the cursor now that you are done with it.

DBMS_SQL AND MULTIROW QUERYING

DBMS_SQL is a package supplied by Oracle Database to perform dynamic SQL operations. Up until Oracle8*i*, it was the *only* way to execute dynamic SQL in PL/SQL. When native dynamic SQL commands (EXECUTE IMMEDIATE and OPEN FOR) were added in Oracle8*i*, DBMS_SQL became a method of last resort for dynamic SQL. This means that it is used now only for the most complex dynamic SQL scenarios, generally categorized as "method 4" dynamic SQL, which is when, at the time of compilation, you don't know either the number of columns in your SELECT list or the number of bind variables in your statement.

DBMS_SQL is a last-resort option, because you must write lots more code to use it. For simple operations, such as those shown in the earlier examples, DBMS_SQL is overkill. For "method 4" dynamic SQL, it is a perfect fit.

You can use DBMS_SQL to fetch multiple rows either one at a time or in bulk. In this article, I will show you only the bulk approach.

```
1 DECLARE
2    l_cursor        PLS_INTEGER := DBMS_SQL.open_cursor;
3    l_names         DBMS_SQL.varchar2_table;
4    l_salaries      DBMS_SQL.number_table;
5    l_fetch_count   PLS_INTEGER;
6 BEGIN
7    DBMS_SQL.parse (l_cursor,
```

```
    q'[select first_name || ' ' || last_name, salary
         from hr.employees
        order by salary desc]',
    DBMS_SQL.native);

DBMS_SQL.define_array (l_cursor, 1, l_names, 10, 1);
DBMS_SQL.define_array (l_cursor, 2, l_salaries, 10, 1);

l_fetch_count := DBMS_SQL.execute (l_cursor);

LOOP
   l_fetch_count := DBMS_SQL.fetch_rows (l_cursor);
   DBMS_SQL.COLUMN_VALUE (l_cursor, 1, l_names);
   DBMS_SQL.COLUMN_VALUE (l_cursor, 2, l_salaries);

   EXIT WHEN l_fetch_count != 10;
END LOOP;

FOR indx IN 1 .. l_names.COUNT
LOOP
   DBMS_OUTPUT.put_line (l_names (indx));
END LOOP;

dbms_sql.close_cursor (l_cursor);
END;
```

Line(s)	Description
2	Declare a DBMS_SQL cursor pointer. This pointer is used in each call to a DBMS_SQL subprogram.
3–4	Declare the two collections I will use to hold the names and salaries. I can declare my own collection types or I can use predefined DBMS_SQL types.
7–11	Parse the dynamic SELECT statement, associating it with the cursor.
13–14	Associate each expression in the SELECT list with an array variable. Note that I have to specify the position of the expression in the list (1 and then 2), the number of rows to retrieve with each fetch (10), and the starting location in the array (1).
16	Execute the SELECT statement. This identifies the result set. Now I can fetch the data.
19	Call the FETCH_ROWS procedure to fetch the next 10 rows (that's what I specified in my call to DEFINE_ARRAY).
20–21	Use COLUMN_VALUE calls to write those rows of data to the collections.
23	Stop fetching if the last fetch retrieved fewer than 10 rows. This must be done!
26–29	Display the contents of the collections.
31	Close the cursor and release the memory.

See what I mean about the volume and complexity of code you must write with DBMS_SQL? And this is a much-simplified example, because I did not have to deal with a variable number of expressions in the SELECT list or the number of bind variables.

By the way, one nice advantage of DBMS_SQL with bulk operations is that the elements are *appended* to the collection. With static SQL BULK COLLECT, the contents of the collection are completely replaced by the rows from the latest fetch.

SUMMARY

The PL/SQL language makes it easy to implement dynamic SQL requirements. When it comes to querying multiple rows of data from a dynamic query, you can choose between EXECUTE IMMEDIATE, OPEN FOR, and DBMS_SQL. DBMS_SQL should be used only for the most dynamic situations, such as when you don't know at compile time how many columns you are selecting. OPEN FOR makes the most sense when you are fetching a small number of rows, because you can avoid the extra coding involved with using arrays. EXECUTE IMMEDIATE is the simplest construct, but you must use arrays and BULK COLLECT if you might fetch more than one row.

Steven Feuerstein is a developer advocate for Oracle, specializing in PL/SQL. Feuerstein's books, including Oracle PL/SQL Programming*, videos, and more than 1,500 quizzes at the Oracle Dev Gym (devgym.oracle.com) provide in-depth resources for Oracle Database developers.*

PHOTOGRAPHY BY **ANDREA MANDEL**

NEXT STEPS

LEARN more about dynamic SQL in PL/SQL.

EXPLORE dynamic SQL scripts on the Live SQL website.

VIEW and **RUN** this article's code on the Live SQL website.

By Connor McDonald

ORACLE DATABASE 12*c* RELEASE 2

Unintended Side Effects

Ensure that the code you write does not create problems elsewhere in your applications.

Two or more features you utilize to build your database applications might seem totally unrelated, and thus you might not even consider that one may have a detrimental impact on the other. But many a developer has been lulled into a false sense of security by assuming a functional separation rather than validating it via integration testing. I've been traveling a lot recently, and recent observations I've made at airports serve as a nice metaphor for reinforcing the importance of integration-testing your applications.

It's common now for many airlines to charge a fee for each checked bag and no fee for carry-on bags. Passengers who carry all their bags on, therefore, save money and time getting out of the airport, because there's no more waiting at the baggage carousel. And for passengers with checked bags, their additional investment means that it takes less time to collect their bags, because there are fewer bags to deliver from the aircraft to the carousel. Another change, or perhaps innovation, I've seen

more recently is a move to achieve faster boarding via more rigid definition and control of "boarding zones" before flights, with passengers for different zones queuing in different waiting areas before a flight starts boarding.

In *isolation*, both of these air travel innovations would appear to be sensible and beneficial for all concerned. However, the *combination* of these strategies has created a problem—one I observed every time I flew with an airline over the past few months. To avoid paying the extra fee for checked luggage, passengers would bring with them the maximum allowed amount of carry-on luggage, a totally understandable position to take. Why pay more if you can avoid it? The consequence was that the aircraft cabin could not accommodate the volume of luggage, so once the cabin luggage space was filled, all remaining baggage had to be checked into the cargo hold anyway. In trying to avoid this, passengers would begin queuing in their respective boarding zones as early as possible. I often saw these queues start well over an hour before a flight was due to depart, but airport departure gates are not designed for this eventuality. Gates' walkways were totally clogged with queuing passengers fervently hoping to avoid having to check their luggage.

Two innovations, lower-cost flights and streamlined boarding, are seemingly unrelated, yet together they have created congestion chaos at airport boarding gates, and not just at the boarding gates of the airline concerned. Even other airlines that do not adopt the newer boarding practice are now experiencing issues with late-boarding passengers, because their own boarding gates are rendered nearly inaccessible by large queues in the walkways.

RETURNING TO THE DATABASE

Here is a similar example, this time with *database* innovations. Consider an application built on a simple CUSTOMERS table structure.

```
SQL> CREATE TABLE CUSTOMERS
  2  (
  3    COUNTRY   VARCHAR2(128),
  4    CREATED   DATE,
  5    CUST_NAME VARCHAR2(150)
  6  );

Table created.

SQL> create index cust_ix on customer ( cust_name );

Index created.
```

The application contains the following query for selecting customers with a specific surname. After the table has been populated with the expected data volumes, the code can be tested with a known customer name:

```
SQL> select *
  2  from   customers
  3  where  cust_name = 'ADAMS';

COUNTRY      CREATED   CUST_NAME
____________ _________ _______________

AUS          08-NOV-16 ADAMS
```

The query returns an expected row and is using an appropriate index on the CUST_

NAME column to find rows efficiently. All looks as it should until—when a subsequent query is being run—the realization that there might be an underlying issue:

```
SQL> select *
  2  from   customers
  3  where  upper(cust_name) = 'ADAMS';

COUNTRY      CREATED   CUST_NAME
____________ _________ _______________

AUS          07-NOV-16 Adams
AUS          08-NOV-16 ADAMS
AUS          09-NOV-16 adams
```

This second query suggests a possible flaw in application logic with respect to the database design. Due to the way the application has been coded, it's possible that incomplete results are being returned to the consumers of that first query, because the rows with the surname ADAMS in mixed case are being dropped. If case insensitivity is indeed the desired goal for this application, a potentially large refactoring exercise is going to be required, because the application code will need to be revisited to recast any SQL statements with the predicate CUST_NAME = <value> to include the UPPER expression applied to the CUST_NAME column.

UNINTENDED SIDE EFFECTS

Not only do the two queries to the CUSTOMER table return different results but they also have differing performance characteristics, as you can see in the execution plans for each:

```
SQL> select *
  2  from   customers
  3  where  cust_name = 'ADAMS';

-----------------------------------------------------------------------------
| Id  | Operation                           | Name      | Rows  |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |           |     1 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| CUSTOMERS |     1 |
|*  2 |   INDEX RANGE SCAN                  | CUST_IX   |     1 |
-----------------------------------------------------------------------------
```

```
SQL> select *
  2  from   customers
  3  where  upper(cust_name) = 'ADAMS';

-----------------------------------------------------------------------
| Id  | Operation         | Name      | Rows  | Bytes |
-----------------------------------------------------------------------
|   0 | SELECT STATEMENT  |           |     1 |   152 |
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |     1 |   152 |
-----------------------------------------------------------------------
```

A side effect of wanting case-insensitive data is that refactoring the code to include UPPER creates a performance issue. Use of an expression prohibits the use of the

existing index, and application performance is degraded. A new index is required:

```
SQL> create index cust_ix2
  2    on customers ( upper(cust_name) );

Index created.
```

The ability to create indexes on expressions has existed since Oracle8*i* Database and is commonly used to implement case-insensitive queries. But unless the refactoring of the application code can be done *en masse*, it is likely that both indexes will need to coexist until all the code can be revisited. Although it is trivial to add an index to a table, you should never add indexes without understanding the repercussions of doing so. Each index

- Consumes space, which, in turn, may mean higher storage costs or more management effort for administrators.
- Increases the resource cost of performing transactions on the underlying table. The index has to be maintained, and this may also increase redo and undo consumption.
- May increase contention for resources in high-throughput environments, especially if the index is on a monotonically increasing value.

Obviously, in an application that may already have hundreds of indexes across many tables, there may be the perception that "just" one more index will hardly have an impact. But similarly, an airline thought that "lower-cost flights" would *not* create airport boarding gate chaos! The unintended side effects are not limited to just requiring a new index. Once that new index is created, consider what happens when a DBA attempts to perform an online space reclamation within the CUSTOMERS table.

```
SQL> alter table customers shrink space;
*
ERROR at line 1:
ORA-10631: SHRINK clause should not be specified for this object
```

As in the airline metaphor, two seemingly unrelated features (function-based indexes and TABLE SHRINK SPACE) clash when combined. In this case, the side effect is a restriction on the SHRINK SPACE functionality: it cannot be performed if a function-based index is in place. Whenever you are using any feature in Oracle Database, it is critical that you perform integration testing across your entire application, *including* the maintenance activities of the DBA.

COLUMN COLLATION IN ORACLE DATABASE 12*c* RELEASE 2

Integration testing would have revealed the conflict between function-based indexes and SHRINK SPACE, but that does not solve the underlying question: how to handle mixed-case data in a performant way without resorting to additional indexes or a large refactoring effort. Oracle Database 12*c* Release 2 includes the new column-level collation feature to help developers tackle this issue. With column-level collation, it is now possible to specify table columns as case-insensitive.

Returning to the example, the CUSTOMERS table is now re-created with the CUST_NAME column using the new COLLATE clause and with a single index on the CUST_NAME column.

```
SQL> CREATE TABLE CUSTOMERS
  2  (
  3    COUNTRY   VARCHAR2(128),
```

```
  4    CREATED   DATE,
  5    CUST_NAME VARCHAR2(150) COLLATE BINARY_CI
  6  );

Table created.

SQL> create index cust_ix on customer ( cust_name );

Index created.
```

The BINARY_CI specification (CI = case insensitive) indicates that case is not a differentiator between values in the CUST_NAME column. Note that the index is on the CUST_NAME column only. The basic query on ADAMS customer names now retrieves all relevant rows independent of case *and* can take advantage of the standard index on the CUST_NAME column.

```
SQL> select * from customer
  2  where cust_name = 'ADAMS';

        ID CUST_NAME
__________ ____________________

        37 ADAMS
       237 Adams
       447 adams
```

```
-------------------------------------------------------------------------
| Id  | Operation                              | Name     | Rows  |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |          |     1 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| CUSTOMER |     1 |
|*  2 |   INDEX RANGE SCAN                     | CUST_IX  |     1 |
-------------------------------------------------------------------------
```

Note that this is a true collation feature, not just a means of obtaining case-insensitive data. For example, aggregation of the data also ignores the case differences between the values:

```
SQL> select cust_name, count(*)
  2  from   customer
  3  where  cust_name = 'ADAMS'
  4  group  by cust_name;

CUST_NAME    COUNT(*)
____________ _________

ADAMS              3
```

A variant on the case insensitivity of data is the handling of accented data. The collation BINARY_AI (AI = accent insensitive) can be specified to also include accented data in the collation. The CUSTOMERS table will be re-created with the CUST_NAME column specified as accent insensitive:

```
SQL> CREATE TABLE CUSTOMER (
  2    ID INT,
  3    CUST_NAME VARCHAR2(40) COLLATE BINARY_AI
  4    );

Table created.

SQL> create index cust_ix on customer ( cust_name );
Index created.
```

After sample data is added to the table, note the effect of BINARY_AI. A fourth row is retrieved, and the third character in the customer name is ă (a with a breve).

```
SQL> select * from customer
  2  where cust_name = 'ADAMS';

        ID CUST_NAME
__________ ____________________

        37 ADAMS
       237 Adams
       447 adams
       647 adăms
```

CHECKING FOR SIDE EFFECTS

The new collation feature allows for case-insensitive queries, without the need to add expressions to existing queries. Digging into the details, however, reveals one

lingering side effect. When the index on CUST_NAME is created, the collation of BINARY_CI on the column makes an adjustment to the underlying definition of the index. Although the data definition language (DDL) text specified the creation of an index on column CUST_NAME, the actual column definition is something else. A query of the data dictionary shows the index definition:

```
SQL> select column_expression
  2  from   user_ind_expressions
  3  where  index_name = 'CUST_IX';

COLUMN_EXPRESSION
_____________________________________________________
NLSSORT("CUST_NAME",'nls_sort=''BINARY_CI''')
```

An *expression* is used to define the index, and hence this is still a function-based index, so there will be limitations on the use of SHRINK SPACE.

To use the new collation features in Oracle Database 12*c* Release 2, the database must have been modified to use a feature that became available in Oracle Database 12*c* Release 1: larger data type sizes for string data. MAX_STRING_SIZE must be changed to EXTENDED from the default of STANDARD to enable VARCHAR2 columns to be sized up to 32,767 bytes instead of the default, 4,000.

If the change to the database string size has not been made, attempts to use the new collation features will result in an error, such as

```
SQL> alter table CUSTOMER default collation binary_ai;
*
```

```
ERROR at line 1:
ORA-43929: Collation cannot be specified if parameter MAX_STRING_SIZE=STANDARD
```

The steps for activating extended strings in the database are documented in *Oracle Database Reference*. Consult *Oracle Database Globalization Support Guide* for more information on the new collation features in Oracle Database 12*c* Release 2.

SUMMARY

With every release of Oracle Database, there are a myriad of new features available to developers and database administrators alike. Just like the features you build into your applications, the database features you exploit should always be tested extensively to ensure that you do not run into any unintended side effects. In this article's particular example of case insensitivity, the collation features in Oracle Database 12*c* Release 2 can help reduce the effort of code refactoring or the need for additional indexes. However, always consult the documentation to get a good understanding of the feature and perform thorough integration tests to make sure you get the maximum benefit from it.

Connor McDonald is an Oracle developer advocate for SQL. His passions are database design, SQL, and PL/SQL, and he can answer your database questions on AskTom.

PHOTOGRAPHY BY
SABINE ALBERS/THE VERBATIM AGENCY

NEXT STEPS

LEARN more about Oracle Database 12*c* Release 2.

DOWNLOAD Oracle Database 12*c*.

By Melanie Caffrey

ORACLE DATABASE

Becoming Privileged and Creating Synonymously

Part 12 in a second series on the basics of the relational database and SQL

This article is the 12th in a series that helps you build on the fundamentals you learned in the 12-part SQL 101 series in *Oracle Magazine*. The previous Beyond SQL 101 article, "Setting Parameters for Dynamic Productivity," taught you how to pass parameters to database script files. You learned how to save the results from a database script execution to an .lst or .txt file with the SQL*Plus SPOOL command. Additionally, you saw how to save database script execution results to an .html file by using the SQL*Plus SET MARKUP HTML command functionality. You also learned how to control aspects of the SQL*Plus execution environment with various common SET commands. Last, you explored how to generate SQL statements at runtime with dynamic SQL and how to document such scripts with comments.

In this article you will

- Learn how to create, alter, and drop users and schemas
- Discover the difference between system privileges and object privileges
- See how to GRANT and REVOKE privileges
- Get an introduction to roles

To try out the examples in this series, you need access to an Oracle Database instance. If necessary, download and install an Oracle Database edition for your operating system. I recommend installing Oracle Database, Enterprise Edition 12*c* Release 2 (12.2.0.1.0). If you install the Oracle Database software, choose the installation option that enables you to create and configure a database. A new database, including sample user accounts and their associated schemas, will be created for you. (Note that SQL_201 is the user account to use for the examples in this series; it's also the schema in which you'll create database tables and other objects.

When the installation process prompts you to specify schema passwords, enter and confirm passwords for the SYS and SYSTEM users and make a note of them. Finally—whether you installed the database software from scratch or have access to an existing Oracle Database instance—download, unzip, and execute the SQL script to create the tables for the SQL_201 schema used for this article's examples. (View the script in a text editor for execution instructions.)

A NAMED COLLECTION

For the examples in this series's articles, you log in to the Oracle Database instance as a database *user*, SQL_201, to create objects (such as tables, views, indexes, and sequences) and then populate or manipulate these objects. The collection of objects a user creates and maintains is known as a *schema*. This article's SQL_201 user owns the SQL_201 schema—the user and schema name are the same. Because each

schema is owned by a single Oracle Database user account, the terms *schema* and *user* are often used interchangeably.

Listing 1 illustrates the relationship between the SQL_201 user and its associated schema. The query of the USER_OBJECTS data dictionary view in **Listing 1** returns schema object information for the currently logged-in database user. All of the objects returned in the result set belong to the SQL_201 schema. Recall from the script you use to create the tables for the SQL_201 schema—used for this article's examples—that you create a database user with the CREATE USER data definition language (DDL) command. Recall also from the same creation script that you were logged in as a user other than SQL_201 to create the SQL_201 user.

Listing 1: Query the USER_OBJECTS data dictionary to return schema object information for the currently logged-in database user.

```
SQL> select object_type, object_name
  2    from user_objects
  3  order by object_type, object_name;

OBJECT_TYPE          OBJECT_NAME
_____________        ______________________________

INDEX                DEPARTMENT_NAME_LOCATION_UK
INDEX                DEPARTMENT_PK
INDEX                DEP_WAGE_INCREASE_I
INDEX                EMPLOYEE_PK
INDEX                EMPLOYEE_WAGE_INC_WORTH_BMI
INDEX                EMP_DEPT_FK
INDEX                EMP_HIRE_DATE_I
```

```
INDEX                SYS_C0010551
SEQUENCE             EMPLOYEE_ID_SEQ
SEQUENCE             ISEQ$$_94754
TABLE                ANNUAL_REVIEW
TABLE                DEPARTMENT
TABLE                EMPLOYEE
TABLE                EMPLOYEE_CTAS
TABLE                EMPLOYEE_EXTRA
TABLE                EMPLOYEE_IDENTITY
TABLE                EMPLOYEE_SUBSET
VIEW                 EMP_MANAGER_OVERVIEW
VIEW                 EMP_WAGE_INCREASE_IT_VW
```

A PRIVILEGED EXISTENCE

To create a new user, the current user must have the CREATE USER *system privilege*. (*Privileges* will be discussed later in this article.) To see what privileges are currently associated with the SQL_201 user, you can run the query in **Listing 2** as the SQL_201 user. Currently the only privilege explicitly granted to the SQL_201 user is CREATE VIEW. If you are not the administrator of the database you are using to run this series articles' examples, you may need your database administrator's assistance to run the first five statements in **Listing 3**.

Listing 2: See what system privileges have been granted to the SQL_201 user.

```
SQL> select privilege
  2    from user_sys_privs;
```

```
PRIVILEGE
_____________________________________________

CREATE VIEW
```

Listing 3: A privileged database user grants system privileges to other database users.

```
--Connect as a privileged user in order to grant necessary privileges to SQL_201
SQL> connect / as sysdba
Connected.

--This step is only necessary if you are using pluggable databases (also known as
containers)
SQL> alter session set container=bynd;

Session altered.

SQL> select user
  2    from dual;

USER
______________________________

SYS

SQL> grant create user to sql_201;

Grant succeeded.
```

```
SQL> grant select any dictionary to sql_201;

Grant succeeded.

--Reconnect as SQL_201 to continue this article's exercises
SQL> connect sql_201@bynd
Enter password:
Connected.

SQL> select privilege
  2    from user_sys_privs
  3  order by privilege;

PRIVILEGE
----------------------------------------

CREATE USER
CREATE VIEW
SELECT ANY DICTIONARY
```

To grant the CREATE USER system privilege to the SQL_201 user, you must be logged in to the database as a privileged user with the ability to grant this privilege to another user. **Listing 3** demonstrates that the SYS user has the ability to grant that privilege to the SQL_201 user.

Using the command

```
connect / as sysdba
```

is a shorthand method of connecting to the database as the SYS user. The SYS user is the most privileged Oracle Database user. It is a default database user created at the time an Oracle database instance is created, and it not only is automatically granted the SYSDBA system privilege but also owns the data dictionary.

In the example in **Listing 3**, the SYS user grants not only the CREATE USER system privilege to the SQL_201 user but also the SELECT ANY DICTIONARY system privilege. The SELECT ANY DICTIONARY system privilege enables the SQL_201 user to query many of the DBA_ and V$ data dictionary views as well as the USER_ and ALL_ views. For more information about which data dictionary views are excluded by default from the SELECT ANY DICTIONARY system privilege, see the *Oracle Database Security Guide*. Note that only trusted users who are to be tasked with database administration activities requiring them to create users should be given the CREATE USER system privilege. (It is not a privilege generally granted to a database developer.) Similarly, the SELECT ANY DICTIONARY system privilege should be granted with discretion as well. In most situations, a database administrator should grant SELECT access to only those data dictionary views that developers need beyond those they already have access to, such as USER_ and ALL_. Grants given to developers should be doled out on a case-by-case basis as determined by the database administrator.

The query in **Listing 4** demonstrates a useful query of the DBA_USERS data dictionary view. The information obtained from the query in **Listing 4** is used to help construct the input to the keywords for the statement in **Listing 5**. The statement in **Listing 5** demonstrates DDL commands for creating an Oracle Database user. Every Oracle Database user must have a username and a password.

Listing 4: Obtain assigned tablespace information for users from the DBA_USERS data dictionary view.

```
SQL> select username, default_tablespace, temporary_tablespace
  2    from dba_users
  3   where username = 'SQL_201';

USERNAME     DEFAULT_TABLESPACE     TEMPORARY_TABLESPACE
________     __________________     ____________________

SQL_201      USERS                  TEMP
```

Listing 5: Create a new user, and assign tablespaces and quota.

```
SQL> CREATE USER beyond_101
  2  IDENTIFIED BY 201
  3  DEFAULT TABLESPACE users
  4  TEMPORARY TABLESPACE temp
  5  QUOTA 100M ON USERS;

User created.
```

(Note that there are exceptions to this database password requirement. You may also create a user that connects and is authenticated either via the operating system, a directory service such as Oracle Internet Directory, or a trusted certificate, among other authentication methods. However, these methods of database authentication should be administered with great care and are beyond the scope of this article. To learn more about these methods of authentication and others, see the *Oracle Database Security Guide*.)

Additionally, if the user is to create objects (tables, indexes, views, and so on), it should be assigned a *default tablespace*, where such objects are then stored. A space usage quota is assigned with the keywords

```
QUOTA {integer[K|M] | UNLIMITED} ON <tablespace name>
```

The *temporary tablespace* assignment determines where data sorting that cannot be performed in memory is done. (A temporary tablespace is also used to store information including, but not limited to, the rows of temporary tables, hash aggregations used for GROUP BY and UNIQUE operations, and hash joins to join larger datasets. To learn about any of these additional uses for a temporary tablespace, search for the relevant keywords in the database documentation.) The statement in **Listing 5** illustrates a new user, BEYOND_101, being created with the password 201. The keywords IDENTIFIED BY, followed by a password, assign the password to the user being created. The tablespace, USERS, is assigned as the default tablespace, and the temporary tablespace, TEMP, is assigned as the temporary tablespace. Finally, the user is granted a quota of 100 megabytes of storage space for its objects in the USERS tablespace.

THE CIRCLE OF LIFE

Although the BEYOND_101 user has been created successfully, it lacks the necessary system privilege to log in to the database. **Listing 6** demonstrates what happens if the user attempts to log in to the database without appropriate permissions in place. **Listing 7** shows which *grant* is necessary to give the BEYOND_101 user the ability to create a session within the database. The CREATE SESSION system privilege is

granted to the BEYOND_101 user via the GRANT <privilege> TO <username> syntax. Additionally, the CREATE TABLE system privilege is granted to the BEYOND_101 user, demonstrating that multiple system privileges can be conferred in a single grant statement. **Listing 7** also demonstrates how the BEYOND_101 user is now able to successfully log in to the database and create and populate a table stored in the USERS tablespace.

Listing 6: User must be granted a system privilege to log in to the database.

```
SQL> connect beyond_101@bynd
Enter password:
ERROR:
ORA-01045: user BEYOND_101 lacks CREATE SESSION privilege; logon denied
```

Listing 7: User can now log in to the database and create a table in its schema.

```
SQL> connect / as sysdba
Connected.
SQL> alter session set container=bynd;

Session altered.

SQL> GRANT CREATE SESSION, create table TO beyond_101;

Grant succeeded.

SQL> connect beyond_101@bynd
```

```
Enter password:
Connected.
SQL> create table test_me (id number, name varchar2(10));

Table created.

SQL> insert into test_me (id, name) values (1, 'Try it');

1 row created.

SQL> commit;

Commit complete.

SQL> select * from test_me;

        ID NAME
__________ __________
         1 Try it
```

Listing 8 outlines how a user can be altered by a change in its quota and password. Note how the act of changing the quota to zero kilobytes enables the user to create a table in the USERS tablespace but results in an error when the user tries to insert data into the table. A user can create a table in its default tablespace even when it has no assigned quota, because, by default, Oracle Database does not look to reduce quota usage for the user until a table *segment* has been created

for a table. (A segment is a logical storage structure that stores a schema's object data—in this case, table data—within the schema's default tablespace.) Users are able to create a table within the USERS tablespace without error because of Oracle Database's default *deferred segment creation* behavior. To learn more about deferred segment creation, see the *Oracle Database Administrator's Guide*.

Listing 8: Alter a user with the ALTER USER data definition language command.

```
SQL> connect / as sysdba
Connected.
SQL> alter session set container=bynd;

Session altered.

SQL> ALTER USER beyond_101 quota 0K on users;

User altered.

SQL> connect beyond_101@bynd
Enter password:
Connected.
SQL> create table test_again (id number);

Table created.

SQL> insert into test_again (id) values (2);
insert into test_again (id) values (2)
```

```
                *
ERROR at line 1:
ORA-01536: space quota exceeded for tablespace 'USERS'
```

Listing 9 demonstrates what happens if you attempt to drop a user that owns schema objects. The error returned means that schema objects were found for the user named in the DROP USER statement. **Listing 10** shows the statement that should be used when you want to drop a user along with its schema objects. Required use of the CASCADE option protects you from accidentally dropping a user whose schema objects are valuable to your database. The DROP USER command without the CASCADE option drops the user only if that user does not own any objects. Note also that once you drop the user, you are no longer able to retrieve information about that user or its schema objects from the data dictionary.

Listing 9: Trying to drop a user that owns schema objects results in an error.

```
SQL> connect / as sysdba
Connected.
SQL> alter session set container=bynd;

Session altered.

SQL> select object_type, object_name
  2    from dba_objects
  3   where owner = 'BEYOND_101'
  4  order by object_type, object_name;
```

```
OBJECT_TYPE          OBJECT_NAME
_____________        ____________________

TABLE                TEST_AGAIN
TABLE                TEST_ME
```

```
SQL> DROP USER beyond_101;
DROP USER beyond_101
*
ERROR at line 1:
ORA-01922: CASCADE must be specified to drop 'BEYOND_101'
```

Listing 10: Use the CASCADE option when dropping a user that owns schema objects.

```
SQL> connect / as sysdba
Connected.
SQL> alter session set container=bynd;

Session altered.

SQL> DROP USER beyond_101 CASCADE;

User dropped.

SQL> select object_type, object_name
  2    from dba_objects
  3   where owner = 'BEYOND_101'
```

```
  4  order by object_type, object_name;

no rows selected

SQL> select *
  2    from dba_users
  3   where username = 'BEYOND_101';

no rows selected
```

RIGHTS OF EXECUTION

A *privilege* is a right to execute a certain type of SQL statement. The two types of privileges are *system* (introduced in the first half of this article) and *object*. The right to create a table or an index is an example of a system privilege. System privileges are often rights to execute DDL commands but can also be rights to execute data manipulation language (DML) commands, such as SELECT ANY DICTIONARY or DELETE ANY TABLE. The rights to query a particular table not owned by the current database user, to perform DML commands on such a table, or to access an integer from a sequence (again, not owned by the current database user) are all examples of object privileges.

Recall that in **Listing 7** it was necessary to grant the CREATE SESSION system privilege to the BEYOND_101 user before it could log in to the database. Two individual system privileges were granted to the BEYOND_101 user: CREATE SESSION and CREATE TABLE. Although it is possible to individually grant each system privilege to each user, it may be simpler and easier for a database administrator if such privileges are instead granted through *roles*. A role is a collection of privileges that

can be granted to a user via GRANT keyword syntax similar to that used for granting privileges. Consider the example in **Listing 11**.

Listing 11: Collect system privileges into a role, and grant the role to users.

```
SQL> connect / as sysdba
Connected.
SQL> alter session set container=bynd;

Session altered.

SQL> CREATE ROLE system_privs_for_201;

Role created.

SQL> grant create user, select any dictionary to system_privs_for_201;

Grant succeeded.

SQL> grant system_privs_for_201 to sql_201;

Grant succeeded.

SQL> select grantee, granted_role
  2    from dba_role_privs
  3   where grantee = 'SQL_201';
```

```
GRANTEE          GRANTED_ROLE
_____________    ____________________________

SQL_201          SYSTEM_PRIVS_FOR_201
SQL_201          RESOURCE
SQL_201          CONNECT
```

In **Listing 11**, because the SQL_201 user has not been granted the CREATE ROLE system privilege, a privileged user must be used to create the SYSTEM_PRIVS_FOR_201 role. Next, the two system privileges already granted to the SQL_201 user are granted to the new role. Finally, the new role is granted to the SQL_201 user. The query of DBA_ROLE_PRIVS confirms that the SYSTEM_PRIVS_FOR_201 role has been granted to the SQL_201 user. This query also illustrates that the SQL_201 user has been granted two additional roles: RESOURCE and CONNECT. The queries in **Listing 12** show which privileges have been granted to the RESOURCE and CONNECT roles. Recall from the script you used to create the tables for the SQL_201 schema used for this article's examples that one of the statements executed was

```
grant connect, resource to sql_201;
```

Listing 12: Display the system privileges granted to the CONNECT and RESOURCE roles.

```
SQL> select grantee, privilege
  2    from dba_sys_privs
  3   where grantee in ('CONNECT', 'RESOURCE')
  4  order by grantee, privilege;
```

```
GRANTEE                        PRIVILEGE
______________________         ____________________

CONNECT                        CREATE SESSION
CONNECT                        SET CONTAINER
RESOURCE                       CREATE CLUSTER
RESOURCE                       CREATE INDEXTYPE
RESOURCE                       CREATE OPERATOR
RESOURCE                       CREATE PROCEDURE
RESOURCE                       CREATE SEQUENCE
RESOURCE                       CREATE TABLE
RESOURCE                       CREATE TRIGGER
RESOURCE                       CREATE TYPE
```

When you executed the above statement, because the SQL_201 user was granted the CONNECT and RESOURCE roles, the user received the ability to perform many of the actions it has performed in this series's articles, including CREATE SESSION, CREATE TABLE, and CREATE SEQUENCE. **Listing 13** demonstrates how to revoke a granted privilege from a user. This command can also be used to revoke a privilege or set of privileges from a role.

Listing 13: Revoke a privilege or a set of privileges from a user or a role.

```
SQL> connect / as sysdba
Connected.
SQL> alter session set container=bynd;

Session altered.
```

```
SQL> REVOKE create user from sql_201;

Revoke succeeded.
```

CONCLUSION

This article taught you how to create, alter, and drop users and how users relate to schemas. You learned what system privileges and object privileges are and how privileges are granted and revoked. Additionally, you saw that roles are collections of privileges that can be granted and revoked in similar fashion to individual privileges. Last, you witnessed how privileges and roles should be granted on a discretionary as-needed basis for security and ease-of-administration purposes.

In the next article in this series, you'll learn more about object privileges and how to create and alter profiles and synonyms.

Melanie Caffrey is a senior development manager at Oracle. She is a coauthor of Beginning Oracle SQL for Oracle Database 12*c (Apress, 2014),* Expert PL/SQL Practices for Oracle Developers and DBAs *(Apress, 2011), and* Expert Oracle Practices: Oracle Database Administration from the Oak Table *(Apress, 2010).*

PHOTOGRAPHY BY **RAY NG**

NEXT STEPS

READ SQL 101, Parts 1–12.

READ more Beyond SQL 101.

DOWNLOAD the sample script for this article.