ORACLE MAGAZINE

MARCH/APRIL 2017

Extend and Integrate

Oracle PaaS enhances and connects your SaaS applications and your business

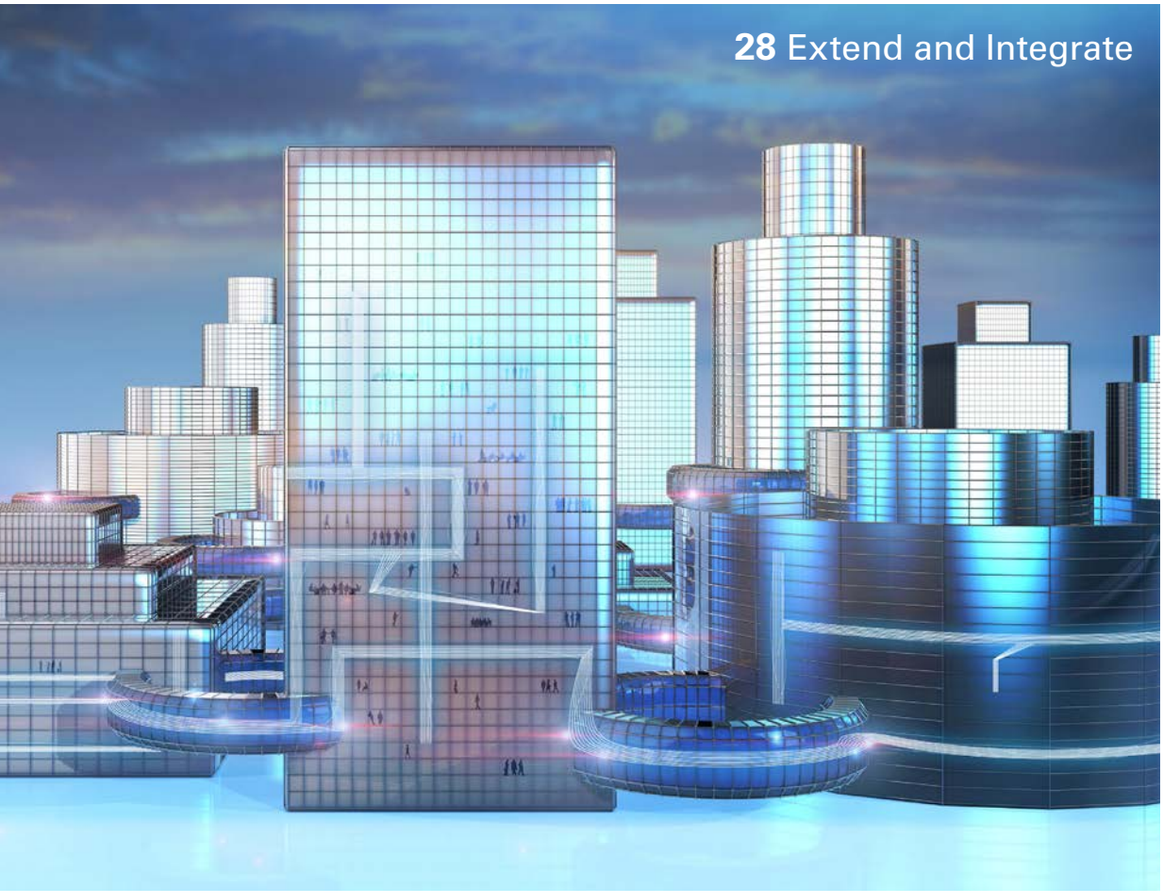
INTERVIEW>
FUTURE-FOCUSED DEVELOPMENT

LOW-CODE DEVELOPMENT>
ORACLE APPLICATION BUILDER CLOUD SERVICE

OPEN SOURCE DEVELOPMENT>
PYTHON

CONTENTS

ORACLE MAGAZINE



FEATURES

8 Interview

UP FRONT

CONTENTS

23 Peer-To-Peer

COMMUNITY

15 Community Bulletin

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, Blair Campbell, 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 Jennifer Hamilton +1.650.506.3794

Associate Publisher and Audience Development Director Karin Kinnear +1.650.506.1985

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.

Tom Haunert

Modules, Not Monoliths

Cloud-native development puts modular app development first.

Back in the 1990s, I attended a lot of developer-focused presentations and events that started with the premise that monolithic applications are *not* the wave of the future. One big emphasis at the time was that a monolithic, single-tier architecture of data access and UI code was a poor choice for an always-changing world.

Self-contained monolithic applications continue to challenge developers and businesses, but for different reasons. The good news in these modern times is that cloud-native development delivers the architectures and processes that make it easy to move from the monolithic past to the dynamic and modular future.

A knock against twenty-first-century one-piece monolithic applications is their independence. That's right—this is a case where independence can be a negative. A modern monolithic app may include everything it needs to function without having to call any other apps, services, or app modules, but it's also unlikely to open itself for use by other apps, services, and app modules. And the other side of that monolithic app independence is a complex internal interdependence of all its application components, making any change to the application a complicated process.

Technical innovations drive change, and of course those innovations drive app changes. Cloud apps are changing

more quickly than the on-premises apps they are replacing, and while monolithic apps can and do change, modular apps can be updated more quickly, with rapid changes only to the modules that are delivering the latest innovations.

Add to this mix the fact that today's modular cloud applications don't need to do *everything* themselves. If your modular app can use the latest payment and messaging services, for example, that's better for your modular app development processes and your app users. But those changes aren't easy with monolithic apps.

In this issue's interview, "Future-Focused Development," Amit Zavery, senior vice president of integration products at Oracle, talks about the benefits of modular development, APIs, and microservices. Beyond the agility that modular development provides, Zavery points to the importance of APIs and microservices in cloud-native development. APIs and microservices let you do more than consume application capabilities and make your own services available to other apps. "The API is really your contract with the rest of the developer community and the rest of the world," he says.

In "Oracle Code: What We Learned," Alexandra Weber Morales describes what went down at the inaugural Oracle Code event on March 1 in San Francisco, California. Oracle President Thomas Kurian's event-opening keynote set the tone, and the keynotes, hands-on labs, and other sessions that followed paid off the Kurian keynote. Read more on what Oracle delivered at the event, and what's next in the Oracle Code 20-city event series.

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

EXPLORE the Oracle Developer portal.

LEARN more about Oracle code.

Cloud Infrastructure For Digital Transformation: Fujitsu M10

FUJITSU

shaping tomorrow with you

OVER 60 YEARS OF HIGH PERFORMANCE & HIGH RELIABILITY TECHNOLOGY

ORACLE Diamond Partner

visit: www.fujitsu.com/sparc/

INTERVIEW

Amit Zavery, senior vice president of integration products at Oracle, explains that chatbot technology uses not just role-based algorithms but also dataset learning and intent-based user interactions.

Future-Focused Development

Cloud-native development shines a light on APIs, microservices, chatbots, and next-generation cloud services. **BY TOM HAUNERT**

Cloud-native development is changing the way developers work, from application architectures to delivery schedules. Modern architectures make it possible to develop modular applications more quickly, and they enable communication about architecture between developers—communication that becomes even more critical in the development of APIs, microservices, machine learning, chatbots, and more. *Oracle Magazine* caught up with Amit Zavery, senior vice president of integration products at Oracle, to discuss cloud-native architectures and how they are supporting cloud services, future development, and *futuristic* development.

Oracle Magazine: How is cloud-native development changing application delivery schedules and architectures?

Zavery: The big change is the move from a big bang release approach to a combination of Agile development and continuous integration and delivery.

Continuous integration and delivery is much more agile than a waterfall-based software development process, in which a developer writes a piece of code that gets handed off to QA for testing, and then handed off for system testing, followed by a handoff to release engineering before the change gets delivered as a big patch or a big release more than a year or a year-and-a-half later. With continuous integration and delivery, when a developer changes some application code it is tested as part of the development process and propagated into the internal and external versions of the application in an integrated process rather than through a waterfall release.

Cloud-native development also separates the development architecture from particular hardware platforms or systems. Development can move seamlessly between different development environments and systems. If a development project uses containers, for example, the developers know that containers enable them to move application workloads to other locations, but they don't need to know about the underlying infrastructure.

Oracle Magazine: What is the role of APIs in cloud-native development? And what about microservices?

Zavery: With modular software architecture, you can evolve components or modules independently, move and integrate them, and make

the components work together, but you can still make changes to a particular component without impacting the whole application. And historically, modules had limited sets of APIs,

> "The API is really your contract with the rest of the developer community and the rest of the world."

perhaps not very well defined. These APIs did provide an interface that developers could work through, but they were not used consistently.

With cloud-native development, new cloud services are dependent on a lot of different functionality coming from various different services. To support those new services, it is important to develop and build your own functionality as a *microservice*—a self-contained piece of software with well-defined APIs. The APIs provide the definition and interaction with a microservice and expose the microservice to users and developers, who can extract functionality through the APIs and evolve their overall application using multiple microservices.

And tomorrow, for example, if you want to use a different credit card processing engine or a different service for your shipping, you can switch to a different microservice with its well-defined APIs without having to rewrite your application for each change.

Oracle Magazine: How should developers prioritize the use of APIs in cloud-native development projects?

Zavery: If you're a developer doing cloud-native development of next-generation applications and systems but you *don't* use an API-first paradigm, I think you will be quite inefficient and make development very expensive. The API is really your contract with the rest of the developer community and the rest of the world. If you're defining your APIs right, other users are going to be able to easily take advantage of all the functionality you're building into your services.

So it's very, very important that developers think about this interface and prioritize how and what they're doing with APIs and how they build them. The lifecycle of APIs is critical in terms of how you design the APIs, how you publish them, how you share them, how you document

them, how you test them, how you analyze them, how you monetize them, and how you secure them. So you can't just say, "I'll design the right API, but I have no responsibility beyond that to manage the whole lifecycle."

Oracle Magazine: Mobile, machine learning, and chatbot technologies are getting a lot of attention and are part of a lot of development plans. What should developers be looking for when they use these technologies in app development?

Zavery: The combination of desktop and mobile applications and websites has fragmented the application user experience. It is creating "app fatigue," making and providing a fragmented user experience in which users have to continuously switch between applications for minor pieces of functionality. There needs to be a better way to interface with and do business with different organizations.

Cloud-native development can move seamlessly between different development environments and systems says Amit Zavery, senior vice president of integration products at Oracle.

This is where the evolution of chatbots comes in. People are already using a lot of messenger services, be it Facebook Messenger or WeChat or WhatsApp, and they are interacting or interfacing with other friends and family or users who are on the same communication channel.

But now businesses can also start to take advantage of these messenger services to provide specific information a user might be looking for. This could be from B2B; it could be from B2C. What chatbots allow developers to do is provide a new interface for their application data—data that already exists in their databases—without having to create a brand-new application interface or create a new user experience through a new mobile app.

Intelligent chatbots have a very good neural network–based and artificial intelligence–based underlying infrastructure and technology, but it doesn't need to be exposed to the end user or even the developer in many cases. The technology uses not just role-based algorithms but also dataset learning and intent-based user interactions to provide better navigation and context-based information to the end user as well.

Developers building mobile apps should use native mobile interfaces but also incorporate chatbot and newer-generation machine learning algorithms so that user interaction is simple and

"What chatbots allow developers to do is provide a new interface for their application data—data that already exists in their databases—without having to create a brand-new application interface."

so that applications can evolve very quickly with newer ways of interfacing with those application back ends.

Oracle Magazine: How does Oracle support API, mobile, and chatbot development?

Zavery: For developing APIs, we have Oracle API Platform Cloud Service. Developers can use it to start building an API and managing the lifecycle of an API. We recently acquired a company called Apiary, which offers API design and governance tooling. Developers can use the Apiary tools to design, develop, publish, and

test their APIs. Apiary tooling combined with Oracle API Platform Cloud Service automates complete API lifecycle management so developers can design and deliver APIs easily.

Once you've defined those APIs, you want to deploy them and integrate multiple applications together. For those tasks, we offer Oracle Integration Cloud Service.

Beyond that, we also have Oracle Mobile Cloud Service, which allows any customer to build native as well as hybrid mobile applications. It provides a mobile back end as a service, allowing you to take any of your existing application back ends, expose them, and create mobile front ends, which you can then deploy through native iOS and Android applications, or deliver as hybrid apps to a browser on a mobile device.

And associated with mobile apps, we've been developing neural network–based chatbot capabilities, where a developer can create a chatbot interface to a lot of popular messaging services and take data as well as information coming from multiple applications and allow users to interact with it.

PHOTOGRAPHY BY
BOB ADLER/THE VERBATIM AGENCY

NEXT STEPS

LEARN more about Oracle API Platform Cloud Service.

Data Quality Made Easy. Your Data, Your Way.

Melissa Data 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.MelissaData.com/oracle

Germany
www.MelissaData.de

United Kingdom
www.MelissaData.co.uk

India
www.MelissaData.in

Australia
www.MelissaData.com.au

MELISSA DATA®
Your Partner in Global Data Quality

www.MelissaData.com | 1-800-MELISSA

BY STEPHEN CHIN, DIRECTOR OF ORACLE TECHNOLOGY NETWORK COMMUNITY MANAGEMENT

Soaring Through the Integration

A team of Oracle ACE Directors accepted a challenge to combine as many Oracle platform-as-a-service (PaaS) products as they could to create a fully functional solution—and do it during a live demo at the Oracle Fusion Middleware Partner Forum in Valencia, Spain, in March 2016. This eight-part article series offers up the technical details of that solution.

New Group, New Architecture

What started as a conversation at Oracle OpenWorld 2016 became the Open Modern Enterprise Software Architecture (OMESA) group. In this podcast, OMESA members discuss lasagna architecture, why being a service-oriented architecture (SOA) architect isn't enough, JavaScript as a first-class citizen among languages, and why you need to step out of your comfort zone.

Open Source Gets Behind Java 9

According to Robert Scholte, chairman of the Apache Maven project, one of the key drivers for Java release adoption is getting build tools and IDEs upgraded to work with the latest release. Scholte has been working to upgrade Maven for the past year and deliver an official release that can be used to test against Java 9. Learn more in his video interview from the Devoxx conference.

PYTHON FOR PL/SQL DEVELOPERS

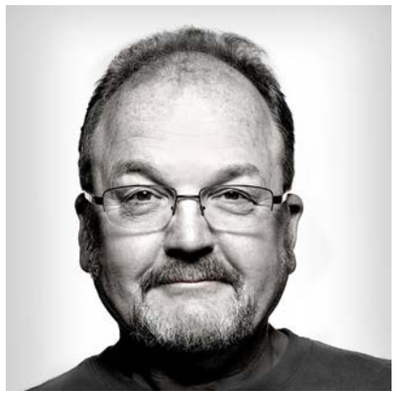
The Python language is worth learning, but who has the time to learn a new language from scratch? In his tutorial series "Learning Python for PL/SQL Developers," Oracle ACE Director Arup Nanda presents a quick way to learn Python—by comparing and contrasting it to PL/SQL. Each article offers a summary of important points and a short quiz to test your understanding. The quizzes are also available to download as PDF documents. Each article also includes a video with a detailed walk-through of the concepts presented.

TOP 10 TECH TIPS

2 Minute Tech Tips are short videos in which Oracle Technology Network community members offer up bite-size nuggets of technical insight on a wide variety of topics and technologies. During 2016, more eyeballs spent more time glued to these 10 tech tips. Meet the 2016 tech tip champions.

Open Source, Open Minds

Today's disruption is tomorrow's tool.



By **Bob Rhubart**

Based on entirely unscientific observation it seems that there has been a noticeable uptick in references to open source tools in blogs and tweets from Oracle Technology Network community members. Having sprung into action on far flimsier assumptions, this seems to me a perfect opportunity to check with community members about any recent changes in their use of open source software. As it turns out, open source has indeed been generating some additional heat of late.

In his role as a senior consultant for Capgemini, Oracle ACE Associate Phil Wilkins works with a variety of organizations. "My relationship with open source has ebbed and flowed, very much influenced by an organization's predisposition to open source," he says. "But it has never gone away. Who doesn't regularly encounter Tomcat or Jetty?"

Other open source products are showing up on Wilkins' radar, including server-side JavaScript environment Node.js, and Kafka, Apache's distributed streaming platform. Wilkins sees this as part of the impact of polyglot programming and microservices on the Oracle ecosystem in which he operates.

For Oracle ACE Director Luc Bors, managing partner and CTO at eProseed in the Netherlands, the past year saw a big change. While his use of open source products had been limited in the past, in 2016 he and his team completed

“Open source has taken on a much more relevant role for me.”

—Arturo Viveros, Oracle ACE

a project that exclusively used Oracle open source products NetBeans, Oracle JavaScript Extension Toolkit (Oracle JET), GlassFish, EclipseLink, and MySQL.

That project and those products left an impression. Late in 2016, Bors presented sessions focused on the project at the German Oracle User Group (DOAG) and UK Oracle User Group (UKOUG) conferences. And there’s more open source in his future. “I will continue to use Oracle JET and NetBeans for UI development as much as possible,” Bors says.

Last year also marked a change for Oracle ACE Arturo Viveros, principal architect at Sysco AS, based in Norway. “Open source has taken on a much more relevant role for me,” he says. “I attribute this in some measure to the influence cloud computing and digital transformation are having on the way we do things.” Viveros also cites the influence of open source tools for continuous integration and automated provisioning, such as Jenkins, Puppet, Ansible, Chef, and Vagrant.

“This kind of technology lets you keep the ‘infrastructure as code’ as well as automate and streamline the development and release lifecycle, improving time to market and reducing both overhead and the possibility of human error without losing a lot of flexibility,” Viveros explains. “Also, it stirs organizations closer to a true DevOps approach, which has become even more attractive and necessary with hybrid cloud integration.”

Viveros expects to continue working with open source, looking into “practical ways to leverage disruptive technologies such as Docker, Kubernetes, Elasticsearch, Kafka, and Blockstack.” He also plans to continue his involvement with the communities around the various open source products. “When you use these tools with a purpose, it becomes quite natural to help improve them, extend the available public resources, provide your own, and give as much feedback as possible,” he explains.

Oracle ACE Robert van Molken, senior integration and cloud specialist with

AMIS, based in the Netherlands, also saw his use of open source increase in 2016, thanks to his work on creating a pluggable Internet of Things (IoT) solution.

"For years I've used SoapUI for testing SOAP and REST services and for creating unit and integration test suites," he explains. "My favorite IDE currently is NetBeans, which I use to create Angular- and JavaScript-based web applications, and for Java development." For his IoT solution, van Molken's work makes what he describes as heavy use of Node.js, Python, and MQTT. "And I'm experimenting with disruptive technologies such as Docker, Kafka, and Blockchain."

Disruptive technologies, van Molken asserts, "tend to start as open source projects."

Are open source products disrupting your world or changing the way you work? Has your use of open source products increased? Join in the community discussion and share your perspective.

Bob Rhubart is the manager of the architect community on Oracle Technology Network, the host of the Oracle Technology Network ArchBeat podcast series, and the author of the ArchBeat blog.

PHOTOGRAPHY BY
MICHAEL MCELROY/THE VERBATIM AGENCY

NEXT STEPS

WATCH the video: *Implementing Node.js in the Enterprise*.

LISTEN to the podcast: "Docker and Virtualization."

VISIT
the NetBeans community space.

the Oracle JET community space.

the Containers, Docker, and Microservices community space.

MEETING THE DEMANDS OF TOMORROW AND DELIVERING HIGH-PERFORMANCE NOW

ORACLE Platinum Partner

As the leader in **Applications Performance Optimization**, Cybernoor can optimize your Oracle Environment to ensure your system performs well and scales. As the data footprint continues to increase along with the complexity of business requirements, delivering a high performance system is crucial to the success of the business. Contact Cybernoor to learn about our proven performance tuning solutions, and ensure your system is prepared for tomorrow's challenges.

Cybernoor Corporation | www.cybernoor.com | +1.925.924.0400

cybernoor™

Cloud and Business Intelligence

The cloud is changing business intelligence, data warehousing, and analytics, and people are talking.

The Business Intelligence, Data Warehousing, and Analytics (BIWA) special interest group (SIG) is part of the Independent Oracle Users Group (IOUG), and the annual BIWA SIG conference started as you might expect a SIG conference to start: small. But the conference has grown as data analytics has moved to the center of more business initiatives and processes—from recruiting to manufacturing to logistics. Now the annual BIWA SIG conference is a don't-miss event for companies that want to learn the latest in big data, spatial, Internet of Things (IoT), and other analytics technologies.

Oracle Magazine covered the 2017 BIWA SIG conference, spoke to three BIWA SIG conference veterans, and asked them this question: How has Oracle Cloud affected your business?

Arthur Dayton

SENIOR CONSULTANT, VLAMIS SOFTWARE SOLUTIONS

On using Oracle Ravello Cloud Service for teaching BI

"Here at the BIWA Summit, we're teaching 25 to 30 students how to deploy maps in Oracle Business Intelligence, and we have 25 VMs [virtual machines] in the lab for those students. In the past we had to do that on top of AWS and have several people involved to go through a bunch of iterations to get a VM up there that would work and that we could deploy. With Oracle Ravello Cloud Service, I get my VM how I want it and drag it into an upload interface and it goes up into the Oracle Cloud . . . and then I just push a button and start those 25 instances all by myself without needing to get anybody else involved."

GV Rao

PRODUCT STRATEGY,
L&T INFOTECH

On bringing cloud IoT to engineering

"Cloud allows us to converge the physical and digital world. For example, we brought in sensor devices onto the filters [on a large water filtration project], which are connected into the Oracle IoT Cloud [Oracle Internet of Things Cloud Service], which in turn pings into the Oracle Java Cloud Service and Oracle Mobile Cloud Service, so on our customers' device they can see what is happening in real time and respond to it. . . . IT connects into the ERP [enterprise resource planning] software, and every KPI [key performance indicator] is then available for you to build more and more intelligence on."

Mat Vranicar

BUSINESS INTELLIGENCE PRACTICE,
AST CORPORATION

On Oracle Business Intelligence Cloud Service changing customer engagements

"Before [the availability of Oracle Business Intelligence Cloud Service] we had to craft solutions that had multiple technologies, with software, hardware components, multiple servers, all the costs involved, all the effort of pricing it out, and configuring it for the initial implementation, but also for the next 5 or 10 years. Now with the cloud . . . we can design a working system, build a prototype, deploy it for a small set of users, only pay for a small set of users initially . . . and ramp up the production environment later. It gives us much more flexibility."

BY BLAIR CAMPBELL

Back in the Day

Computer rentals, love at first job, and the ease of getting into IT now versus then

Ric Van Dyke

Hotsos

Grapevine, Texas

ORACLE ACE

Company: Hotsos, a consulting firm helping clients and customers improve the performance of their Oracle systems

Job title/description: Education director, responsible for writing and delivery of Hotsos' Oracle education curricula

Oracle credentials: Oracle Certified DBA (Oracle 7, Oracle 8*i*, Oracle 9*i*), with 31 years of experience using Oracle products

What's the most common cause you see when IT projects go wrong? Time management. There's a tendency to think a project will go much faster than it does. People get sick, need time off for this or that, or need training on new technology—or a completely unexpected glitch is discovered. Too many times there's no slack in the schedule; one misstep and the whole project is thrown off. The best solution is to plan pessimistically. When I first started consulting, my boss at the time advised me, when estimating a project time, figure out how long you really think it will take, double that, and add 10 percent. That seems to still hold true. It's always better to come in under the deadline than over it.

What's the next big thing driving change in your industry? Of course the cloud is making a shift in how we do things. Interestingly, to some degree it's a shift back in time. In the town I grew up in, small companies used to rent time on a large company's computer system to do accounting-type work. And when I was taking classes in college, I would log in to the computer back at

school from home and do my course work. This was, in a sense, like cloud computing today. Of course it's much better than when I was logging in to a VAX cluster at Eastern Michigan University. But the concept is the same—just highly improved.

What's your go-to Oracle reference book? The Oracle Database Concepts manual. This one book can be the gateway to an incredible understanding of all things Oracle. I try to read it cover to cover every major release or so.

Ludovico Caldara

Trivadis

Lausanne, Switzerland

ORACLE ACE Director

Company: Trivadis, a provider of IT consultancy, system integration, solution engineering, and IT services in Central Europe

Job title/description: Senior database specialist, providing advanced database consulting services including license audits, high-availability and fault-tolerance audits, performance reviews, and training

Oracle credentials: Oracle Certified Professional DBA (Oracle Database 11*g*, Oracle Database 12*c*) and Oracle Certified Performance and Tuning Expert (Oracle Database 11*g*, Oracle Database 12*c*), with 17 years of experience working with Oracle products

How did you get started in IT? When I was a child I was addicted to video games. Later, as a teenager, I dreamed of becoming a game developer. I started developing not only at school but also in my spare time. Immediately after finishing my studies, I got a DBA job offer in a big data center and accepted it. I was 20 and in love with my new job—and I still love it so much, many years later.

What's your favorite tool on the job? Definitely the Data Guard feature of Oracle Database, Enterprise Edition. Not only because it's by far the best solution for Oracle Database disaster recovery, but also because it gives unique capabilities such as rolling upgrades or the capability of moving your

databases online from one server to another one. You can also leverage snapshot standbys to work on fresh copies of your production database every day, or use your standby databases as a master for storage-based snapshot copies—for example, using ACFS [Oracle Automatic Storage Management Cluster File System].

Which new features in Oracle Database are you currently finding most valuable? When Oracle Database 12*c* Release 1 was released, I was skeptical about the real use cases of Oracle Multitenant. Now I've tried the new features of Oracle Database 12*c* Release 2 such as improved resource management, I/O rate limits per pluggable database [PDB], near-zero downtime PDB relocation, lockdown profiles, and PDB flashback. And I finally see a real potential for consolidation and manageability of midsize applications. Databases with different characteristics, workloads, and recoverability needs can all be managed with a few SQL commands inside a container database instance—without the operational effort of maintaining several scripts and instances at the OS level.

Luis Weir

Capgemini

London, England

ORACLE ACE Director

Company: Capgemini, an IT consulting, outsourcing, and professional services company

Job title/description: Cloud principal, focused on helping organizations define and implement cloud strategies and Oracle platform-as-a-service solutions

Oracle credentials: Oracle SOA Suite Certified Implementation Specialist and Oracle Business Process Management Suite Certified Implementation Specialist, with more than 15 years of experience using Oracle products

Which new features in Oracle Cloud Platform services are you currently finding most valuable? Right now I'm focusing heavily on application-programming interface [API] management and

have been beta-testing the Oracle API Platform Cloud Service for a few months now—so far I find it to be simple to use, yet very powerful. I've also implemented Oracle Mobile Cloud, which in my view is one of the most mature and stable of the cloud services. I'm in the process of implementing Oracle Integration Cloud, Oracle SOA Cloud Service, and Oracle Messaging Cloud.

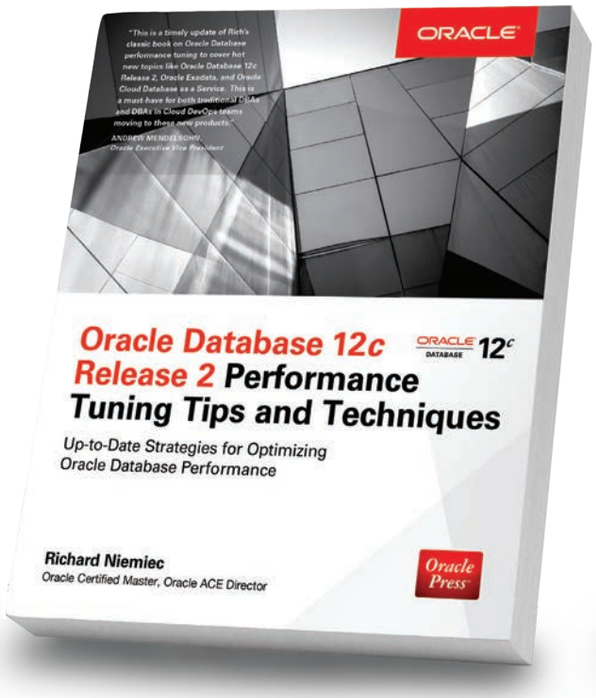
What advice do you have about getting into web development? At present it's a lot easier to get into programming than it was back in the day. There's so much information online that all it takes is a browser, Google, and of course the right appetite and attitude to learn. I find Java to be one of the most useful languages to start with, because it sets a solid foundation in object orientation and multipurpose programming. There are several resources online to help get started.

What green practices do you use in your development work? I use the green software wiki, which is a good source of inspiration for best practices when writing energy-efficient software. Of course, it's not always practical or even possible to adopt them all, but obvious ones such as "avoid data redundancy," "avoid polling," and "make proper use of virtualization" are now default practices when defining solutions.

Oracle Press™

Your Destination for Oracle and Java Expertise

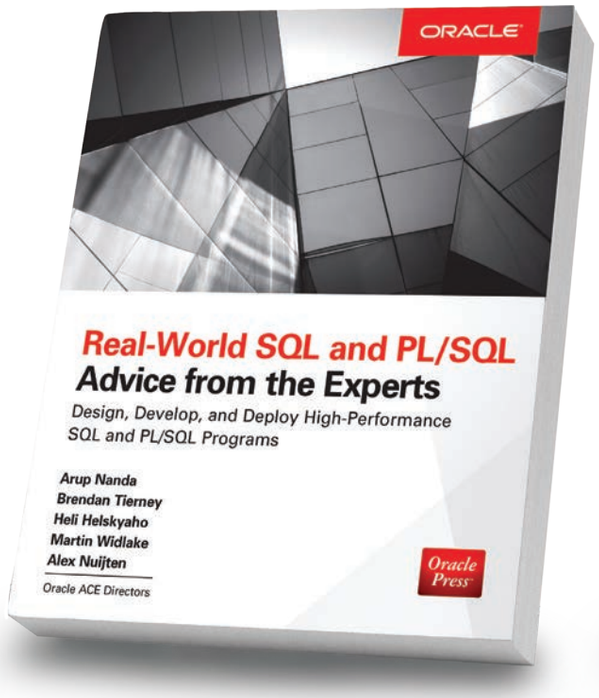
Written by leading technology professionals, Oracle Press books offer the most definitive, complete, and up-to-date coverage of Oracle products and technologies available.



Oracle Database 12c Release 2 Performance Tuning Tips & Techniques

Richard Niemiec

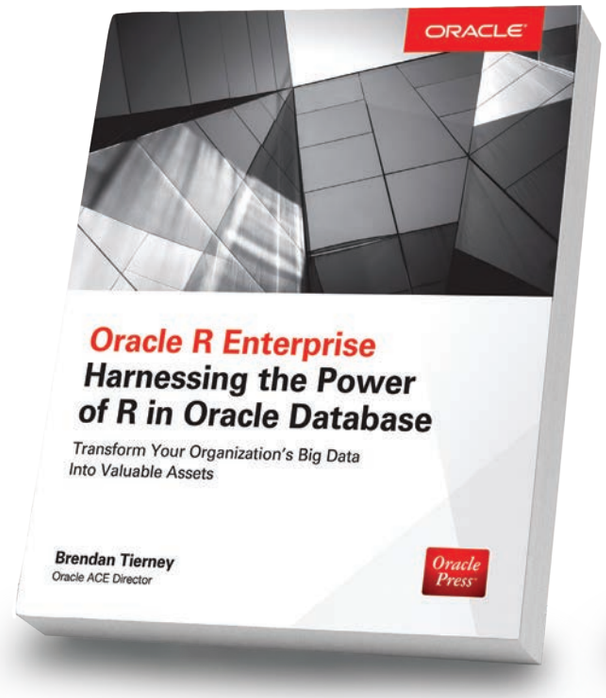
Proven database optimization solutions—fully updated for Oracle Database 12*c* Release 2.



Real World SQL and PL/SQL: Advice from the Experts

Arup Nanda, Brendan Tierney, Heli Helskyaho, Martin Widlake, Alex Nuitjen

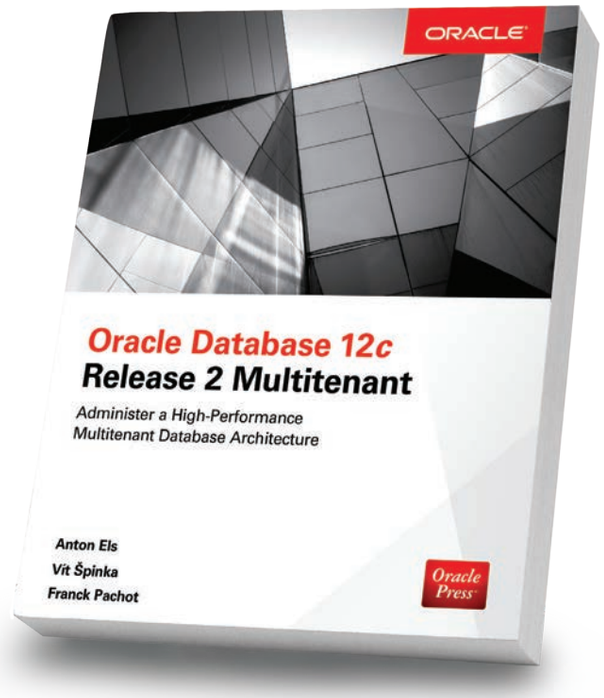
Master the underutilized advanced features of SQL and PL/SQL.



Oracle R Enterprise: Harnessing the Power of R in Oracle Database

Brendan Tierney

Effectively manage your enterprise's big data and keep complex processes running smoothly.



Oracle Database 12c Release 2 Multitenant

Anton Els, Vit Spinka, Franck Pachot

Master the powerful Multitenant features of Oracle Database 12*c*.

Available in print and eBook formats.

Oracle Press™

www.OraclePressBooks.com • @OraclePress

Extend and Integrate

Oracle PaaS enhances and connects your SaaS applications and your business.

BY DAVID BAUM

Software as a service (SaaS) delivers powerful standardized applications for everything from finance to sales force automation and marketing to human capital management. But when your organization's business processes require custom applications, how do you build new cloud services and keep your information and applications—including legacy on-premises apps—integrated? After all, these cloud-based software solutions can't operate in a vacuum. When the surrounding business processes change, you need to be able to keep your SaaS applications in sync, as well as develop new cloud services that complement and extend your existing ones.

"Too many companies make the mistake of adopting SaaS without thinking about how those applications will interoperate with their existing environment and future innovations," says Amit Zavery, senior vice president of integration products at Oracle. "And once they realize that the SaaS application must connect to existing on-premises enterprise resource planning (ERP) applications, they rely on yesterday's integration approaches: point-to-point integration, batch integration, or FTP file transfers that don't support real-time business. They hand-code integrations that are too brittle to keep up with the cloud's pace of change."

TREK BICYCLE CORPORATION
Waterloo, Wisconsin

BRANDS:
Trek, Bontrager, Electra, and Diamant

INDUSTRY:
Manufacturing

ORACLE PRODUCTS AND SERVICES:
Oracle Cloud Platform, including Oracle Mobile Cloud Service and Oracle SOA Cloud Service
JD Edwards applications
Oracle Consulting

When Trek Bicycle Corporation began its journey to the cloud, the company was well aware of those challenges. A world leader in the manufacture and distribution of bicycles and cycling products, Trek had large on-premises technology investments to consider, including an ERP system that handles inventory, purchasing, claim management, warehousing, and distribution. Senior managers at the Waterloo, Wisconsin–based company realized that cloud services would be the quickest way to modernize these entrenched legacy systems,

Trek's business case for modernizing legacy systems with cloud services was driven primarily by the needs of the company's more than 10,000 dealers around the world, says Global ERP Director Tom Spoke (right), pictured here with Global ERP Technical Manager Girish Washikar.

but the path forward had to include a solid integration strategy in which existing information systems could work hand in hand with new SaaS solutions.

"The business case was driven largely by our retailers, which include more than 10,000 dealers around the world, all of whom wanted a quicker, more efficient way to submit customer claims," says Tom Spoke, director, global ERP at Trek.

Trek sells its products through a network of independent bicycle dealers across North America, Europe, Asia Pacific, South America,

> **“We’re taking a lot of functionality that dealers can currently get only via their computers or by calling us on the telephone, and putting it on their handheld devices.”**
>
> —**Girish Washikar,** Technical Manager, Global ERP, Trek Bicycle Corporation

and Africa. Spoke and other Trek IT professionals envisioned a new set of SaaS apps that would streamline or replace several crucial business processes that these dealers depend on, such as enabling them to submit repair claims using their mobile phones. Currently, Trek dealers use a web-based business-to-business (B2B) system to enter and retrieve service claims. However, the way these dealers tend to interact with customers makes the system somewhat cumbersome.

“For each claim, a dealer has to take a picture of the customer’s bike, upload it to a PC, log in to our B2B system, create a claim, add the image, and then submit the claim—a process that takes six or seven minutes,” Spoke explains.

Trek found a solution with Oracle Cloud Platform, an evolving family of platform-as-a-service (PaaS) technologies that allows Trek to extend its existing applications and build new SaaS apps using versatile, cloud-based development and deployment services.

Extend Apps with Oracle Mobile Cloud Service

Trek is using Oracle Mobile Cloud Service, a key component of Oracle Cloud Platform, as a back end for mobile app development. “We decided to leverage Oracle Mobile Cloud Service to create a SaaS app that will offer a more efficient way of entering claims compared to our existing B2B system,” says Spoke. “Now dealers will be able to connect directly to our on-premises JD Edwards applications from Oracle through

Oracle Cloud Platform

Oracle Cloud Platform, Oracle's platform-as-a-service (PaaS) family of offerings, allows developers, IT professionals, and business leaders to develop, test, and deploy applications in the cloud in a secure, cost-effective way. With the industry's #1 database (Oracle Database) and #1 application server (Oracle WebLogic Server) at its core, Oracle Cloud Platform offers flexible cloud services that open the door to new business opportunities.

Oracle Cloud Platform includes services for application development, data management, business analytics, integration, content and collaboration, and mobile—among them Oracle Mobile Cloud Service and Oracle SOA Cloud Service.

Oracle Mobile Cloud Service enables companies to create and deploy scalable, robust, and secure mobile applications quickly and easily and provides support for continuous lifecycle management, API management, versioning, and security.

Oracle SOA Cloud Service provides a PaaS computing platform for running Oracle SOA Suite, Oracle Service Bus, and integration analytics in the cloud. This comprehensive, hot-pluggable software suite is ideal for building, deploying, and managing software integrations using service-oriented architecture.

Sign up for a free trial here.

their mobile devices and initiate the entire claims process."

Trek worked with Oracle Consulting to create the new claim submission mobile app. The solution included optical character recognition (OCR) technology that allows service professionals to call up customer service histories simply by scanning a bike's serial number. Dealers can use a mobile phone to photograph a bike, create a claim, and submit it to the JD Edwards system with a couple of taps on the screen.

"We're taking a lot of functionality that dealers can currently get only via their computers or by calling us on the telephone, and putting it on their handheld devices," explains Girish Washikar, technical manager, global ERP at Trek.

Oracle Mobile Cloud Service provided a cloud-based platform not only for building the front-end components of this new SaaS app but also for connecting it to Trek's back-end system. "It gave us a mechanism to build an API to connect to our ERP system through the JD Edwards EnterpriseOne application interface services server framework," Washikar continues. "Oracle Mobile Cloud Service handles all the actions of entering claims and uploading information into the ERP system."

> **Too many companies make the mistake of adopting SaaS without thinking about how those applications will interoperate with their existing environment and future innovations.**
>
> —**Amit Zavery,** Senior Vice President, Integration Products, Oracle

Spoke believes that this new app, set to enter production in the spring of 2017, will reduce the seven-minute claims-submission process to fewer than two minutes. "With the number of claims Trek assists with across our global retailer network, that adds up to a lot more time selling bicycles and serving customers," he notes.

Integrate Apps with Oracle SOA Cloud Service

In a related initiative, Trek is using Oracle SOA Cloud Service to establish electronic connections with third-party logistics providers around the globe, including opening a vast new revenue channel through Tmall, China's largest online marketplace. Oracle SOA Cloud Service simplifies the process of developing external integrations across numerous interfaces to retailers, using the electronic data interchange (EDI) and managed file transfer (MFT) protocols and tying them back to Trek's on-premises ERP system.

Washikar says Oracle SOA Cloud Service is more flexible than the on-premises SOA technology Trek used in the past, which makes it easier for Trek to develop unique SOA composites for each trading partner. Cloud-based development tools accelerate the process of onboarding new partners and setting up SOA services. "SOA development is a perfect function to migrate to the cloud," he emphasizes. "We are leveraging Oracle SOA Cloud Service to orchestrate new transactions."

Oracle's Zavery believes that Trek's use of Oracle SOA Cloud Service to create a hybrid cloud architecture represents a popular use case for PaaS, because it allows an on-premises service bus to connect traditional enterprise applications with cloud-based IT assets. "If you are using Oracle SOA Suite in conjunction with Oracle SOA Cloud Service or Oracle Integration Cloud Service, then the paradigm, the artifacts, the objects, and the integrations

"SOA development is a perfect function to migrate to the cloud," says Trek Global ERP Technical Manager Girish Washikar (right), pictured here with Global ERP Director Tom Spoke.

you've created will seamlessly move over [to the cloud]," he explains. "Because you can connect back to your existing on-premises services, APIs, agents, and gateway, these cloud services let you modernize without having to throw away existing investments, and you can evolve those investments without having to rewrite everything."

Developers "should be spending time either writing the code or extending the applications they have," says Zavery. "They shouldn't have to worry about all the underlying technologies required to build an application."

Developers should think about what language and framework they want to use, or what platform they want, who the user is for the application, and what the user interface should be, says Zavery. "Beyond that," he says, "all other capabilities should be provided from the platform provider in terms of automation, ease of use, elasticity, security, backup recovery, patching, and upgrades. And that's a service we provide as part of our platform, so developers can focus on writing that application."

Trek plans to use Oracle Cloud Platform to develop more SaaS assets in the future, including a mobile app that will provide bicycle

retailers additional functionality in other areas of their business and another mobile app that will be used internally in Trek's distribution centers. Over the long term, Trek is considering moving its JD Edwards applications off premises and into the cloud as well. "We want to get out of the business of owning, operating, maintaining, and upgrading on-premises information systems," Spoke says.

David Baum is a freelance business writer specializing in science and technology.

PHOTOGRAPHY BY **PAUL S. HOWELL**

NEXT STEPS

LEARN more about Oracle Mobile Cloud Service.

LEARN more about Oracle SOA Suite Cloud Service.

WATCH Amit Zavery discuss how PaaS is helping developers move into the future.

TRY Oracle Cloud.

Australian Finance Group CIO Jaime Vogel considered closing one data center but saw that the company could recognize true savings and benefits only by completely embracing the cloud.

WHY AFG WENT ALL-IN TO CLOUD

Australian Finance Group skips hybrid approach as it banks on Oracle Cloud to drive innovation. **BY MIKE FADEN**

About three years ago, Jaime Vogel, CIO at Australian Finance Group, determined that the company needed to overhaul its approach to IT amid intensifying competition and to support AFG's ambitious growth strategy. A variety of technology startups were starting to disrupt the financial sector, including the home-mortgage loan business that AFG operates via its network of 2,650 brokers. At the same time, AFG was diversifying into new products, such as car loans, commercial finance, and insurance.

The company needed to shift more of its IT resources into innovation—and to do so with a relatively flat IT budget. "At the time, we were spending only around 20 percent of our IT budget on innovation and the remainder on operational activities," Vogel says. "We knew we had to invert those numbers."

The route that Vogel and his team identified: progressively shift from on-premises computing to the cloud.

No Half Measures

AFG's IT environment included a range of Oracle business applications running on Oracle Exadata Database Machines in two fiber-connected data centers about 20 kilometers apart in Perth, Western Australia.

The company initially considered taking a hybrid cloud approach, in which it would continue to run applications in one data center but close the other data center and use the cloud for disaster recovery. But it quickly became apparent that a hybrid model wouldn't generate the required operational cost savings, Vogel says. Further analysis revealed that the greatest cost benefits would be delivered by switching from on-premises to cloud-based software-as-a-service (SaaS) applications.

"We could recognize the true savings and benefits only by completely embracing the cloud," he says. "No matter whether you are running one server, 50, or 150, there's a baseline of operational costs, and until you eliminate those servers you can't really gain satisfactorily from the transition to the cloud."

Another important factor played into the company's ultimate decision to move to a 100 percent cloud model and close both of its data centers. "The key breakthrough was our confidence in providers understanding our needs and providing the availability and reliability that AFG and our customers desire," Vogel says.

> **“We were spending only around 20 percent of our IT budget on innovation and the remainder on operational activities. We knew we had to invert those numbers.”**
>
> **—Jaime Vogel,** CIO,
> Australian Finance Group

Methodical Migration

Once AFG had made its decision, it formulated a strategy for migrating each application. “If there was a cost-effective SaaS solution that supported our business initiatives, then we’d adopt it,” he says. If not, the IT team moved the application to infrastructure as a service (IaaS), while rearchitecting the application to take advantage of cloud capabilities such as automated scalability and recovery.

AFG took a methodical approach, starting with lower-risk applications such as Microsoft Exchange and SharePoint, as well as development tools. With the confidence earned from those early successes, Vogel’s team then began moving AFG’s core business applications to their SaaS equivalents, including Oracle Fusion Incentive Compensation Cloud Service and Oracle Financials Cloud. In all, roughly 90 percent of AFG’s applications are SaaS today, Vogel says.

However, a different approach was needed for AFG’s on-premises Siebel Customer Relationship Management (Siebel CRM) applications from Oracle, which support the company’s all-important broker network. During 10 years of using the software, AFG had customized the Siebel applications extensively, and it determined that the same functionality wasn’t available in a SaaS alternative.

So AFG chose instead to migrate the Siebel applications to IaaS, while adapting them to

scale automatically to match the ebb and flow of demand. At night, the system runs on just a handful of virtual servers, minimizing operational cost. The next morning, it automatically adds servers to handle the increasing load as brokers log in. "We script and template-build these environments, to scale out when demand is high and down when the system is quieter," Vogel says. "We've effectively taken a traditional application and adapted it into a new-world cloud environment."

To ensure a smooth transition of the production Siebel system to the cloud, AFG prepared for about a year, first architecting and then progressively moving the development, test, and training environments, followed by considerable preproduction testing, says Holden Lai, IT manager–sales at AFG.

In all, AFG moved about 80 on-premises applications to its SaaS cloud, allowing the company to shut down its two underutilized data centers in May 2016. Vogel estimates that the shutdown saves the company about AU$500,000 (US$385,000) a year in operational costs, while the cloud applications and services bring a modern feature set, as well as modern APIs that improve integration and automation.

IT Budget Swings Toward Innovation

Meanwhile, Vogel has grown his team from 22 people four years ago to 36 today—on a relatively flat IT budget. Those 14 additional people are all developers focused on creating new products.

Today, Vogel says, AFG spends about 60 percent of its IT budget on new, innovative projects (as opposed to support and enhancements), compared with 20 percent when he joined the company.

Those innovations include the integration of Oracle Financials Cloud and Oracle Fusion Incentive Compensation Cloud Service with the Siebel CRM applications now running on IaaS. The integrated systems automatically collect data on daily mortgage sales transactions and commissions from lenders, calculating the required broker payments. As a result, AFG can now process its 600,000 monthly commission payments five times faster than it could before—in 2 hours instead of 10.

Oracle's platform-as-a-service (PaaS) products have also enabled innovation at AFG. The company built a new loan-origination system for its own home-mortgage business, using a combination of Oracle's PaaS products to develop

Australian Finance Group CIO Jaime Vogel says that confidence in providers understanding the company's needs was an important factor in the decision to close the on-premises data centers.

functionality that wasn't available in off-the-shelf software. AFG found that the off-the-shelf solutions weren't flexible enough to allow it to quickly adapt as the financial services sector gets disrupted by market entrants. "We think the market is going to change rapidly, and we want to be a leader in that space," says Andy McGee, IT manager–business services at AFG.

By combining Oracle Process Cloud Service, Oracle Documents Cloud Service, Oracle SOA Cloud Service, and Oracle Policy Automation Cloud Service, AFG created a solution that

> **If you just take what you've already got in your data centers and move it into the cloud, it may be a much easier project, but you're probably not going to get the benefits.**
>
> —**Holden Lai,** IT Manager–Sales, Australian Finance Group

manages the entire loan process, from receiving a loan application through the assessment process and obtaining additional information and services from third-party providers, to ultimate approval and funding. Despite the breadth and complexity of the process, AFG built the system in about 12 months, McGee says.

Lessons Learned

The cloud transition hasn't been seamless. Several times during the transition Vogel's team determined that the cloud products they were deploying lacked certain functionality, but the beauty of the cloud model is that the needed improvements tended to show up in a new release three months later, Vogel says.

Lai also emphasizes the need to allocate time for testing and tuning during a migration to ensure the same level of performance as with on-premises systems. And to get the most benefit from the cloud, he suggests looking at how to take full advantage of the features available via APIs and scripting. "If you just take what you've already got in your data centers and move it into the cloud, it may be a much easier project, but you're probably not going to get the benefits," Lai says.

Another early challenge was convincing AFG's IT staff that the cloud transition is the right move for them. "People naturally tend to be protective of their roles as system administrators and in managing infrastructure," Vogel says. But when AFG started explaining their future roles, it gave staff the understanding that the cloud transition was probably their best opportunity to obtain new skills in preparation for what the future may hold, says Vogel. "It didn't take long for people to get on board. Today, a lot of them are writing scripts and coding, and it's actually been a very welcome change for most of them," he adds.

One example of the reskilling: the team's former IT operations manager is now in a

continuous-improvement role, responsible for evaluating business processes with an eye toward boosting efficiency, reducing costs, and upping the quality of customer service.

Now that AFG's IT environment is running entirely in the cloud, Vogel continues to look for ways to direct an even bigger percentage of the company's IT spending—beyond the current 60 percent—into new, innovative projects. "We're continuing to evolve and improve," he says. "We like to aim high. Any incremental increase in IT budget is focused on innovation."

Mike Faden is a principal at Content Marketing Partners. He has covered business, technology, and science for more than 30 years as a writer, editor, consultant, and analyst. Faden is based in Portland, Oregon.

PHOTOGRAPHY BY
SABINE ALBERS/THE VERBATIM AGENCY

NEXT STEPS

LEARN more about Oracle's infrastructure as a service.

LEARN more about Oracle's platform as a service.

COLLABORATE 17

TECHNOLOGY AND APPLICATIONS FORUM
FOR THE ORACLE COMMUNITY

Discover all that COLLABORATE 17 has to offer and register today!

⟨IOUG⟩ OAUG™ Quest

#C17LV

attendcollaborate.com

April 2-6 2017
Las Vegas, NV

Mandalay Bay
Resort & Casino

Oracle Code: What We Learned

The Oracle Code series, launched in San Francisco on March 1, promises to be a must-attend event for cloud developers worldwide.

BY ALEXANDRA WEBER MORALES

Continuous delivery may still be aspirational for many software teams, but it's more attainable than ever, thanks to Oracle Cloud. The Oracle Code series of in-person events worldwide is showing cloud developers how to turn software development into a genuine supply chain—running on high-performance, self-maintaining, "invisible" infrastructure.

During his opening keynote on March 1 at Oracle Code in San Francisco, California—the first of the 20-city event series—Oracle President of Product Development Thomas Kurian offered Oracle's three-point vision for cloud developers.

1. Oracle Cloud Platform provides APIs or a GUI for developers to create a software-defined virtual data center and run it in the cloud within minutes by instantiating compute, storage, and network elements while bringing their own IP address ranges to the cloud to extend their applications and data centers seamlessly to the cloud.
2. Oracle is providing a complete development platform to automate the software supply chain and enable developers to focus on their code while the platform administers itself (installing, patching, configuring, backing up, encrypting the data, and maintaining) using software and machine learning. "We provision the stack, scale it, back it up, encrypt it—we do all those functions," Kurian said.
3. Finally, developers can bring their own modules of code, and Oracle Cloud Platform takes care of scaling the code for them and meeting required SLAs [service-level agreements] without developers' having to worry about scaling the underlying infrastructure. "Infrastructure as a service over time becomes invisible because you do not have to worry about the infrastructure your code is being deployed and scaled upon," Kurian said. "You should be able to deploy every minute or every hour."

Kurian's keynote was packed with demos of an array of infrastructure, platform, and application development services—including "low code" declarative development with Oracle Application Builder Cloud Service and Oracle MySQL Cloud Service, as well as DevOps and monitoring/management.



From left: Oracle President of Product Development Thomas Kurian gives the opening keynote, attendees take in a breakout session, and Douglas Crockford discusses technology after JavaScript.

Supply Chain in the Sky

Kurian's demonstration of a Docker-based deployment workflow included a variety of prepackaged multitenant services for container orchestration and discovery. Describing how developers on Oracle Cloud can build microservices in Java, Ruby, Node.js, Python, or Scala, he explained how Oracle fills in the gaps: "Using Kubernetes, we'll deliver the orchestration and scheduling of your containers for you. If you use Docker today, you have to write your own discovery service. We do that for you. If you need caching, today you have to write and manage your own Redis distributed caching environment. We will give you a managed Redis environment," he said.

Networking in the Flesh

Standing proudly among the partner booths in the Oracle Code Lounge, Stephen Chin, director of Oracle Technology Network, explained the logic

behind Raspberry Pi– and cloud-driven devices, using a 3-D printer, a coffee station, a CNC router, and a Pac-Man–style game. The printer and the router were sending gcode commands to turn designs into prototypes.

"This actually is running in a browser," Chin said, pointing to a CAD-like interface. "It's a JavaFX application written by Michael Hoffer using JPro, which is technology that lets you take any JavaFX application and then run it all on the server side, so you just open the browser and don't need any applets or plugins. You can run this in Oracle Cloud Marketplace on Oracle Application Container Cloud." The result was a piece of a drone that Hoffer flew at the JavaOne conference last October in San Francisco, Chin explained.

Nearby, Hans-Henry Sandbaek, CEO of German ISV JPro.io, explained to attendee Jim Wu, a software engineer at Loyal 3 in San Francisco, how Java's original "write once, run anywhere" promise was finally coming to fruition

From left: Stephen Chin, Director of Oracle Technology Network, talks Java and coffee; attendees get hands-on in lab session; an explorer gets virtual in the Oracle Code Lounge.

From left: Technology gets physical with 3-D printing and a CNC router in the Oracle Code Lounge; Baruch Sadogursky, developer advocate at JFrog, gives a two thumbs up status report; a hands-on lab offers questions and answers.

with his tool's interpretation of JavaFX, the GUI library replacing Swing in Java Platform, Standard Edition.

Discovering DevOps Maturity

In his morning talk titled "DevOps at Scale: A Greek Tragedy in 3 Acts," Baruch Sadogursky gave a humorous overview of the maturity stages of DevOps—from giddy three-person startup to thriving enterprise. Sadogursky is a developer advocate at JFrog, an Oracle partner and maker of a binary repository tool that he included in his descriptions of typical DevOps tool-chains that reflect each level of company experience and customer demand.

After his talk, Sadogursky praised Oracle's support for DevOps tooling in the cloud. While he called DevOps tools fairly interchangeable, the same could be said for cloud stacks, he posited.

"With cloud as the deployment target, Oracle Cloud is emerging as one of the

> **Infrastructure as a service over time becomes invisible because you do not have to worry about the infrastructure your code is being deployed and scaled upon.**
>
> **—Thomas Kurian,** President of Product Development, Oracle

very good alternatives," Sadogursky said. "It gives you this versatility that you can select the right cloud for the job. Selection of the cloud makes a lot of sense as well. If I want my stuff on the cloud, can I get all the parts from the cloud provider? That's important because locality matters, both for performance and for money, because intracloud transfers don't cost money. Having these toolchains out of the box on one cloud, with great uptime and everything, is great."

Ultimately, Oracle Code participants made it clear that developers are eager to embrace continuous delivery but are aware that they must choose wisely. As Sadogursky put it, "There is no such thing as 'serverless.' All that means is there is a server somewhere, just not here. The quality of the server and the quality of the people taking care of the server are still very important."

Alexandra Weber Morales, principal at World Wind Writing, is the former editor in chief of Software Development *magazine and has more than 15 years of experience as a technology content strategist and journalist.*

PHOTOGRAPHY BY **ORACLE**

NEXT STEPS

LEARN more about Oracle Code.

EXPLORE the Oracle Developer portal.

ORACLE CODE

Register Now

New One-Day, Free Event | 20 Cities Globally

Explore the Latest Developer Trends:

- DevOps, Containers, Microservices & APIs
- MySQL, NoSQL, Oracle & Open Source Databases
- Development Tools & Low Code Platforms
- Open Source Technologies
- Machine Learning, Chatbots & AI

Live for the Code

Find an event near you:

developer.oracle.com/code

ORACLE®

ORACLE DATABASE

Open for Development

By Anthony Tuininga

Use Python and cx_Oracle to build Oracle Database applications.

Python is a general-purpose open source language. It is interpreted, and it is object-oriented with class-based inheritance. Everything in Python is an object, including classes, functions, and modules, so anything can have methods and attributes and can be passed as arguments to functions. Python is *dynamically typed*, which means that the types of variables do not need to be declared and can be changed easily. On the other hand, Python is *strongly typed*, which means that the type of an object cannot be changed once it is created.

Python is supported on all major operating systems, including embedded systems, and implementations beyond the reference C implementation enable integration with other ecosystems. The C (primary) implementation of Python can be extended easily in C to provide access to other libraries written in C or to improve performance, which enables Python to excel at integration tasks. The cx_Oracle

driver for Oracle Database is one example of a C extension. It enables fast, full-featured, and efficient access to Oracle Database. This article introduces Python and the cx_Oracle driver and demonstrates how to use them to write applications for Oracle Database.

Python 3 was released in 2008, and over the last eight years, it has been enhanced considerably. The examples shown in this article were written with Python 3.5 but can be used with Python 2.7 with minimal changes.

INSTALL PYTHON

The official source of the Python language is the Python website. The site includes binary packages for Windows and macOS and source packages for use on other platforms. (Most Linux distributions come with at least one version of Python pre-installed.) The Python site also offers excellent documentation on the language, an extensive standard library, and tutorials to help you get started. If the standard library doesn't have what you need, chances are very good that one of the tens of thousands of packages available on PyPI (the Python Package Index) will satisfy your requirements. You can download and install these packages directly with the built-in package manager, "pip," if you know the name of the package you want to install, or you can browse or search the list of packages. Mailing lists are available for Python user questions, and there is a large Python community on Stack Overflow for questions and answers. In addition, Python conferences are held on every continent except Antarctica.

Included in the many packages on PyPI is support for all the major database platforms available today. To keep the modules for all the database platforms behaving as similarly as possible, a database API specification was developed. cx_Oracle, the

module developed to support the Oracle Database platform, fully implements this database API specification, and it includes support for properties and methods that enable you to take advantage of additional Oracle Database–specific features.

Before installing and using cx_Oracle, install an Oracle client. The Oracle client software enables the database and the application to reside on different machines (even completely different platforms), and it seamlessly handles network traffic between the two. The easiest Oracle client package to install is called Oracle Instant Client, and you can download it for free from Oracle Technology Network for any of the platforms supported by Oracle. Installation instructions are included at the bottom of the download page. Several packages are available, but using cx_Oracle requires only the Basic or Basic Lite package. The SDK package is also required, however, if you plan to build cx_Oracle yourself.

With an Oracle client installed, install cx_Oracle. On Windows, the simplest approach is to download an executable installer directly from PyPI. On Oracle Linux or similar Linux distributions, RPM package files are available for download from the same location. For all other platforms, use the pip package manager to perform the download, compilation, and installation of cx_Oracle with one command:

```
pip install cx_Oracle
```

By default, pip is installed in Python 2 starting at Python 2.7.9 and in Python 3 starting at Python 3.4. If you are using an older version of Python that doesn't have pip installed, you can also download the source package for cx_Oracle directly from PyPI, extract the files from the package, and then run the following commands:

```
python setup.py build
```

```
python setup.py install
```

TEST AND EXPLORE

After you complete the Python, Oracle client, and cx_Oracle installation steps without errors, test the installations by importing the cx_Oracle module into the Python interpreter and connecting to an Oracle Database instance, as in the following script:

```
import cx_Oracle

connection = cx_Oracle.connect("user/password@server/ServiceName")
cursor = connection.cursor()
cursor.execute("select sysdate from dual")
today, = cursor.fetchone()
print("The current date is", today)
```

Note that the connection string passed to the cx_Oracle.connect() function is the same one you use when connecting with SQL*Plus to Oracle Database. Knowing this can be useful for troubleshooting installation issues.

The test query returns an Oracle date, and cx_Oracle maps that data to a Python datetime object from the built-in datetime module. **Table 1** shows the mappings of Oracle, cx_Oracle, and Python types.

When cx_Oracle executes a query, it examines the metadata Oracle Database makes available and creates a variable of the type indicated in the second column of **Table 1** for each column that is fetched. Although these types can be changed with

Table 1

Oracle Type	cx_Oracle Variable Type	Python Type
VARCHAR2	cx_Oracle.STRING	str
NVARCHAR2	cx_Oracle.NCHAR	str
CHAR	cx_Oracle.FIXED_CHAR	str
NCHAR	cx_Oracle.FIXED_NCHAR	str
ROWID	cx_Oracle.ROWID	str
RAW	cx_Oracle.BINARY	bytes
LONG	cx_Oracle.LONG_STRING	str
LONG RAW	cx_Oracle.LONG_BINARY	bytes
NUMBER	cx_Oracle.NUMBER	int or float
DATE	cx_Oracle.DATETIME	datetime.datetime
TIMESTAMP	cx_Oracle.TIMESTAMP	datetime.datetime
TIMESTAMP WITH TIME ZONE	cx_Oracle.TIMESTAMP	datetime.datetime
TIMESTAMP WITH LOCAL TIME ZONE	cx_Oracle.TIMESTAMP	datetime.datetime
CLOB	cx_Oracle.CLOB	cx_Oracle.LOB
BLOB	cx_Oracle.BLOB	cx_Oracle.LOB
NCLOB	cx_Oracle.NCLOB	cx_Oracle.LOB
BFILE	cx_Oracle.BFILE	cx_Oracle.LOB
CURSOR	cx_Oracle.CURSOR	cx_Oracle.Cursor

advanced cx_Oracle features (input and output type handlers), the default types are usually more than adequate. As rows are fetched, a tuple is created for each row, as shown in the script in **Listing 1**.

Code Listing 1: Script that creates tuple plus output

```
# drop table if it exists
try:
    cursor.execute("drop table TestQuery purge")
except:
    pass

# create table to demonstrate queries
cursor.execute("""
    create table TestQuery (
        IntCol number(9) not null,
        StrCol varchar2(30) not null,
        DateCol date not null
    )""")

# populate table with a few sample rows
for i in range(3):
    cursor.execute("""
        insert into TestQuery (IntCol, StrCol, DateCol)
        values (:intCol, :strCol, :dateCol)""",
        intCol = i + 1,
```

```
            strCol = "Test String %d" % (i + 1),
            dateCol = datetime.date(2017, i + 1, 1))

# perform query
cursor.execute("select IntCol, StrCol, DateCol from TestQuery")
for row in cursor:
    print("Row:", row)
```

```
Row: (1, 'Test String 1', datetime.datetime(2017, 1, 1, 0, 0))
Row: (2, 'Test String 2', datetime.datetime(2017, 2, 1, 0, 0))
Row: (3, 'Test String 3', datetime.datetime(2017, 3, 1, 0, 0))
```

cx_Oracle uses array fetching internally to reduce the number of times the Oracle client must communicate with Oracle Database, which is particularly important when the database and the client are not together on a fast network. The default array size is 50, which is adequate for most queries. You can change this easily, however, by modifying the arraysize property of the cursor before executing the query, as in the following:

```
cursor.arraysize = 200
```

Inserts, as shown in the script in **Listing 1**, can use an array instead of executing the insert statement once for each row. The following script performs the same inserts as the script in **Listing 1** but inserts them all at once, instead of one at a time,

thereby improving performance.

```
data = [(i + 1, "Test String %d" % (i + 1),
                datetime.date(2017, i + 1, 1)) \
        for i in range(3)]
cursor.executemany("""
        insert into TestQuery (IntCol, StrCol, DateCol)
        values (:intCol, :strCol, :dateCol)""", data)
```

One other type that deserves further mention is the large object (LOB). You can use LOBs to store images, videos, text documents, or anything else that requires more space than what is supported in the basic string and raw types. One other use for LOBs is JSON data. Oracle Database 12*c* Release 1 (Oracle Database 12.1) added support for JSON, which enables the database to store, index, and query JSON data—without the need for a schema that defines the data. **Listing 2** demonstrates cx_Oracle support for JSON in Oracle Database 12*c* Release 2 (Oracle Database 12.2).

Code Listing 2: cx_Oracle accessing JSON in Oracle Database 12.2 via character large object (CLOB)

```
# drop table if it exists
try:
    cursor.execute("drop table TestJSON purge")
except:
    pass
```

```
# create table to demonstrate use of LOBs and JSON
cursor.execute("""
    create table TestJSON (
        JSONCol clob not null,
        constraint TestJSON_ck check (TestJSON is JSON)
    )""")

# populate table with a few rows
inputData = [
    dict(x = 1, y = 2, z = 3),
    dict(x = 4, y = 5, z = 6),
    dict(x = 7, y = 8, z = 9)
]
for row in inputData:
    cursor.execute("insert into TestJSON (JsonCol) values (:jsonData)",
            jsonData = json.dumps(row))

# perform query; output should match input
cursor.execute("select JsonCol from TestJSON")
outputData = [json.loads(lob.read()) for lob, in cursor]
print("Data matches?", outputData == data)
```

As you can see from this brief introduction, you can use Python with the cx_Oracle module to quickly and easily connect to Oracle Database, perform queries, and insert data. For more information on cx_Oracle, consult the documentation and additional sample code in the source samples directory.

Anthony Tuininga is a member of the data access group at Oracle. He has more than 25 years of experience with Oracle Database and is the creator and maintainer of the cx_Oracle driver.

NEXT STEPS

TRY Oracle Database Cloud services.

LEARN more about Oracle Database support for Python.

LEARN more about cx_Oracle.

ORACLE APPLICATION BUILDER CLOUD SERVICE

High on Power, Low on Code

Get started with Oracle Application Builder Cloud Service.

By Shay Shmeltzer

Oracle Application Builder Cloud Service provides a browser-based visual development environment for creating web and mobile applications quickly. It also includes a runtime platform for publishing applications and storing data in a built-in Oracle Database instance. Business users and developers can use Oracle Application Builder Cloud Service to create bespoke applications and applications that get their data from external apps using REST interfaces.

Oracle Application Builder Cloud Service excels at extending Oracle software as a service (SaaS) apps. To support that capability, it includes a built-in service catalog that provides easy access to services exposed by Oracle SaaS, enabling developers to build applications that incorporate data from those apps along with custom data objects they create. Oracle Application Builder Cloud Service also includes out-of-the-box support for the Oracle Applications simplified look and feel, so applications can look and behave like Oracle SaaS apps.

In this article, I'll show you how to create a basic Oracle Application Builder Cloud Service application without writing a single line of code. In addition, this article's demonstration video shows the steps for building the application and the Oracle Application Builder Cloud Service features described in the article.

To walk through the article steps, sign up for a free trial account for Oracle Application Builder Cloud Service. Get your account at Oracle Cloud.

CREATING A BASIC APPLICATION

With your trial account created, log in to Oracle Application Builder Cloud Service. When you log in, you'll be prompted with a list of applications you and your team have created. (Note that users can collaborate simultaneously on the development of an application in Oracle Application Builder Cloud Service).

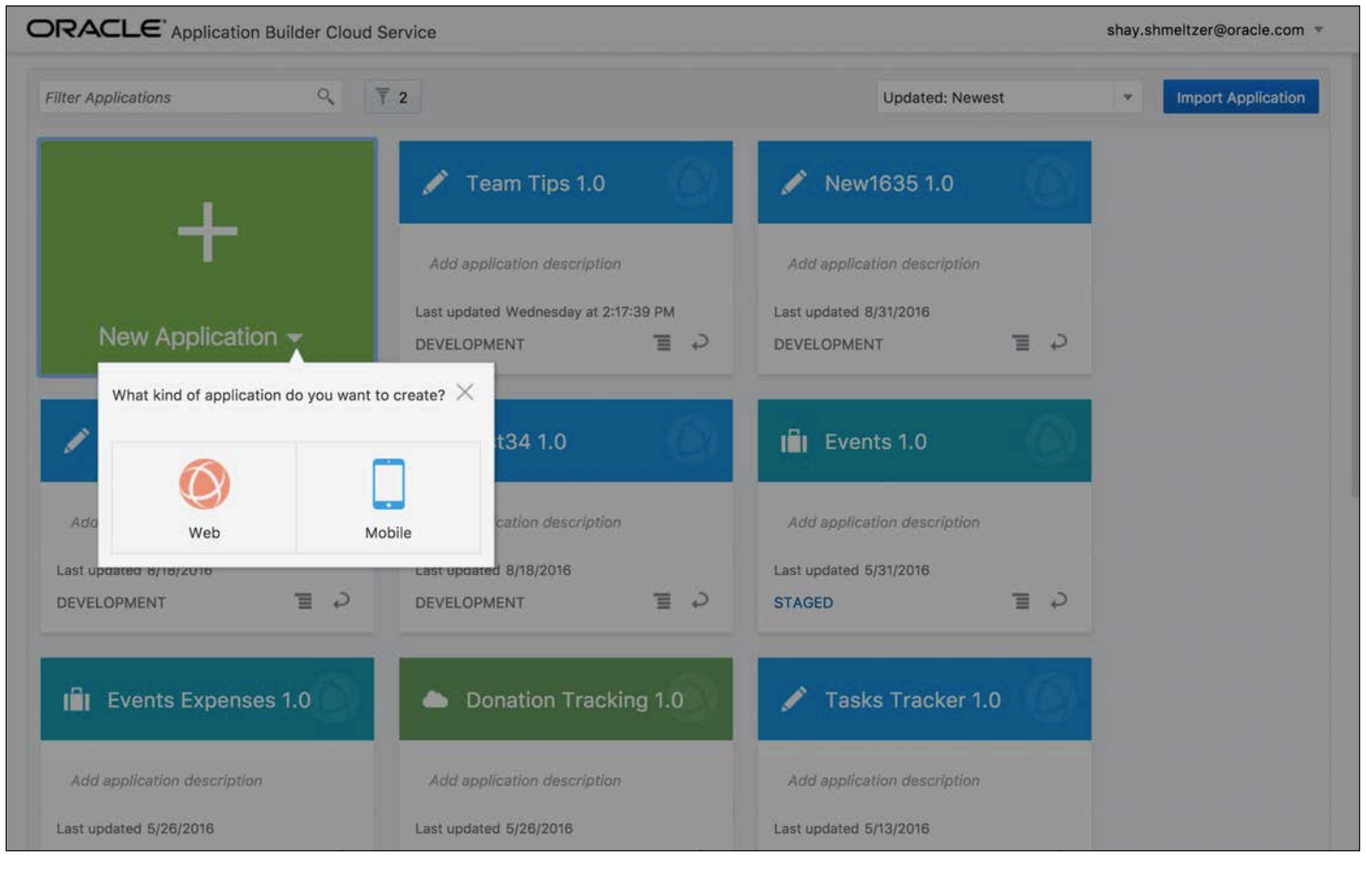


Figure 1: Oracle Application Builder Cloud Service home screen

Logged into Oracle Application Builder Cloud Service, as shown in **Figure 1**, click the big green **+** icon to create a new web application. The mobile option enables you to create an app that deploys and runs on a mobile device directly. The web option creates an application that is web-based, with responsive adaptation to various screen sizes, including mobile devices.

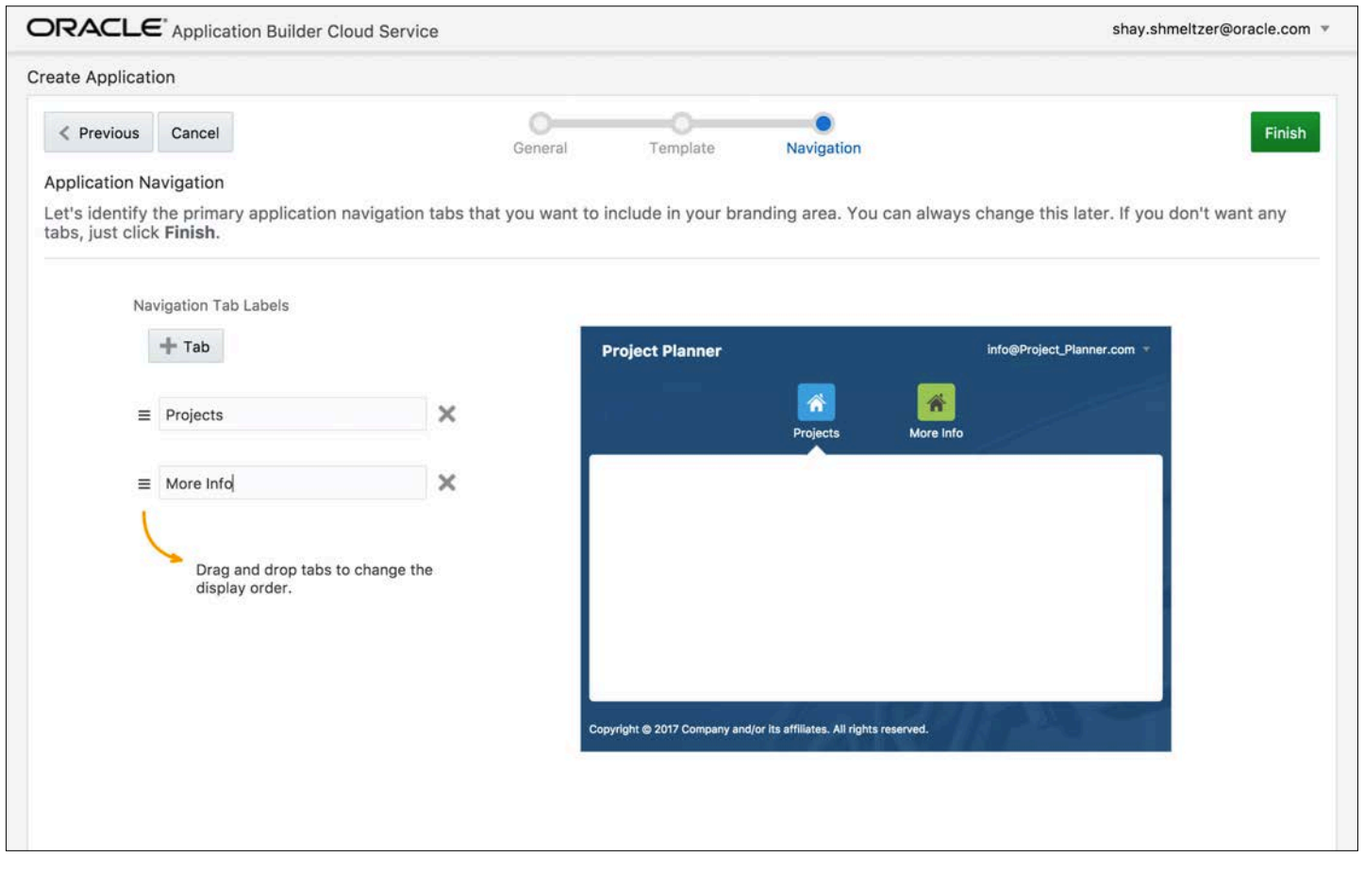
Choose **Web** for the application type. A three-step Create Application wizard will then guide you through

- Providing a name for the application
- Specifying the look and feel of the app (which you can choose from a set of built-in templates, such as Oracle Alta UI or Oracle Applications Cloud Look and Feel, or from additional themes you can develop and add to the list)
- Defining the main area of the app that will represent the top-level menu

In this case, you'll create a new budget planning application, entering the name (I used Project Planner for my application), choosing **Oracle Applications Cloud**, and creating two top-level menu options—Projects and More info—as shown in **Figure 2**.

Click **Finish** to create the application. When the application is created, you'll be directed to the visual page editor. Oracle Application Builder Cloud Service uses the familiar concepts of a component palette on the left, a visual what-you-see-is-what-you-get layout editor in the middle, and a property inspector on the right.

Figure 2: Create the Oracle Application Builder Cloud Service application



To design your page, drag components from the component palette (on the left) onto the visual editor (in the middle) and set their properties in the property inspector (on the right). You can click the page title and set a title and a logo for the application, as shown in **Figure 3**.

In this application, you'll want to track information about various projects, so start by dragging a table component from the component palette (under the Collection label) and dropping it onto the visual editor page. You'll be prompted to

Figure 3: Oracle Application Builder Cloud Service visual editor

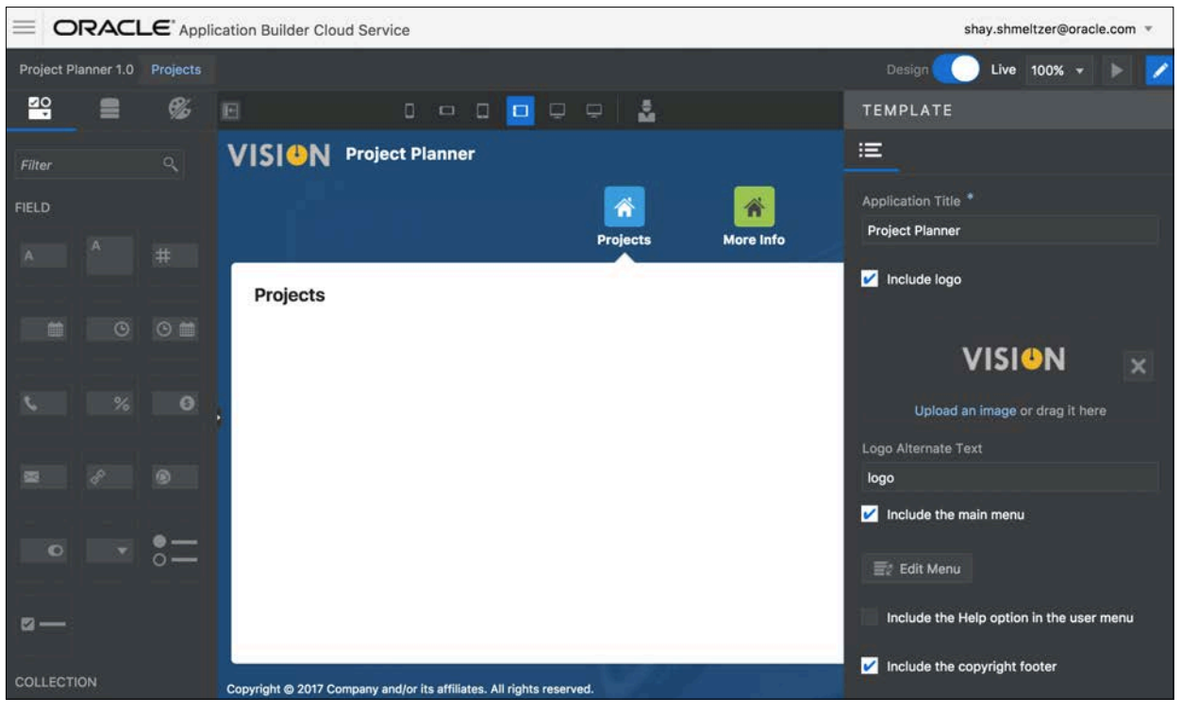
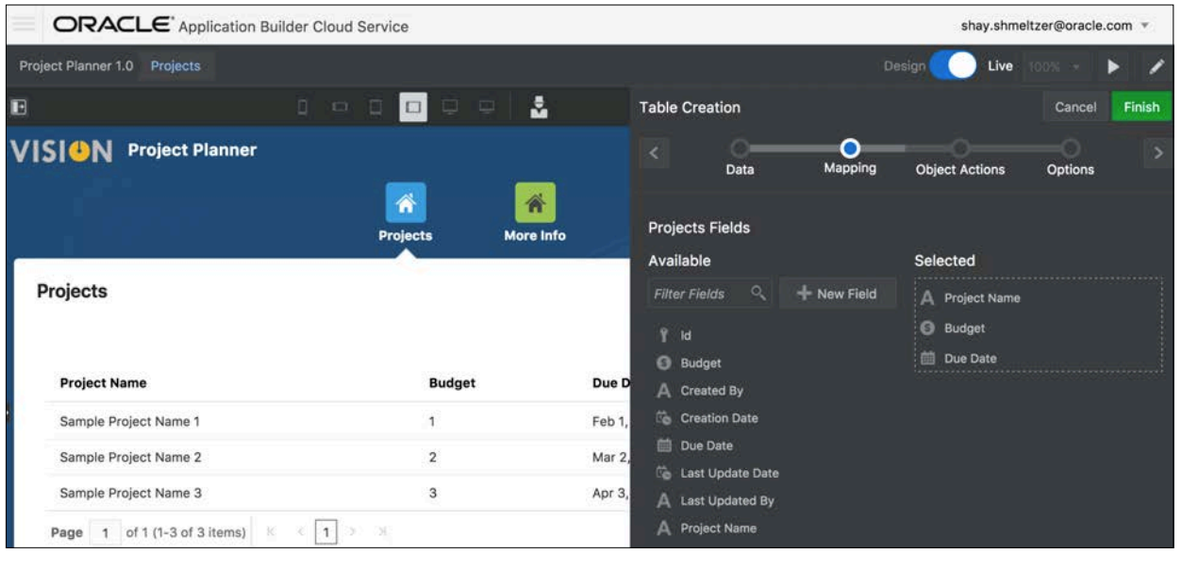


Figure 4: Table Creation wizard



connect this table to a source of data and, because you don't already have one, create a new business object. In the Table Creation wizard on the right, click **New Business Object** and enter `Project` as the object name.

Next, add the specific fields to track. Fields can be of various types: text, numeric, currency, date, email, URL, and so on. Create each of the following fields by clicking **New Field**, entering the name in the **Default Display Label** field, choosing the type button, and clicking the check mark button:

- `Project Name` is a character field.
- `Budget` is a currency field.
- `Due Date` is a date field.

Figure 4 shows that as you add fields, the visual editor immediately reflects the changes in the UI, showing you a preview of how your page is going to look.

Click the "next" arrow (**>**) to continue through the steps of the Table Creation wizard. Indicate which operations—Create, Edit, Delete, Details—you would like to do on the new object. These are all **On** by default, and those are the settings you want for this application. Next, indicate that clicking the name of a project will take you to the page that will enable you to edit its details. From the **Field to link on**

menu, select the **Project Name** field.

Click the "next" arrow (**>**) to add filters to the table to enable users to search through the data. For **Basic Filter**, click the **On** button. **Project Name** is the default value for the **Filter Field** and the value you want for this application.

Click **Finish** to close the Table Creation wizard, and click the **Run** button (right-arrow icon at the upper right of the screen) to run the application. Because the application is completely HTML/JavaScript–based (leveraging the advanced Oracle JavaScript Extension Toolkit [Oracle JET] UI technologies), no additional compile, build, and deploy steps are needed—the application is immediately functioning in the browser.

You can add records and update the data as needed. By default, a full set of create, read, update, and delete (CRUD) operations and pages is provided for your new custom object. Add a first project to the application to see how the application presents data at this point. Click the **Create** button, and enter data for the project. For **Project Name**, enter `Holiday Party`; for **Budget**, enter `5000`; and for **Due Date**, choose **February 14, 2017**. Click **Save and Close** to save the project data. The application should now look similar to **Figure 5**.

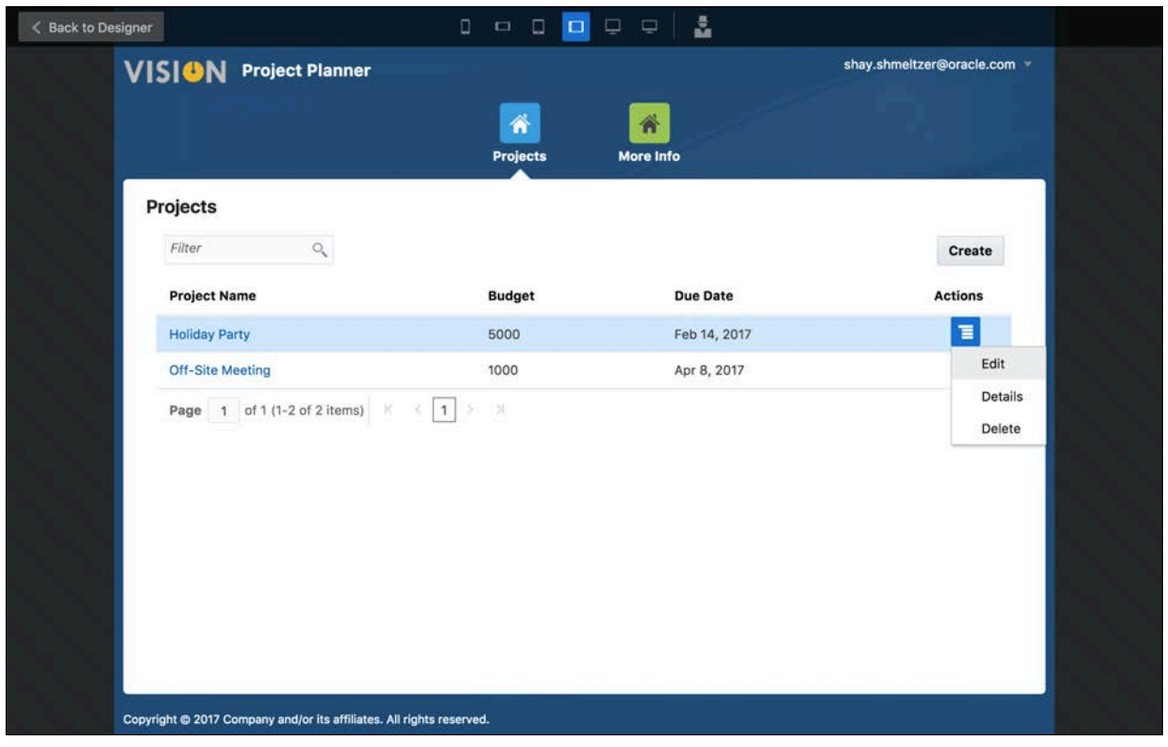


Figure 5: Running the Oracle Application Builder Cloud Service application

Modifying the page layout and adding another field are also quite easy. While you are previewing your application, click **Back to Designer** to navigate to the project editing page.

Now change the page layout to use two columns for the current project data and add a remarks field. Drag a **Two-Column Layout** component from the component palette (under the Layout label), and drop it onto the visual editor page. Then drag the

WHAT IS IT?

Oracle Application Builder Cloud Service provides an easy way for business users to create their own web and mobile applications and host them in a cloud environment. Through an intuitive visual development interface and with no need for prior development experience, creating engaging applications and reducing IT backlogs have never been easier.

Try Oracle Application Builder Cloud Service.

Project Name, **Budget**, and **Due Date** fields into the new layout component, keeping their original order from top to bottom. Now drag a **Text Area** component from the component palette (under the Field label), and drop it onto the visual editor to the right of the two columns you just defined. For **Label**, enter `Remarks` and click **OK**. In the Text Area property inspector pane on the right, change **Label Position** so the Remarks label is left-aligned and above the text entry field. Click **Run** to see the new two-column layout and remarks area in the application runtime.

Now add master/detail (or parent/child) relationships to the application. Click **Back to Designer**, and drag a **Tabs** component from the component palette (under the Layout label) and drop it onto the visual editor page, below the current page content. In the visual editor, click **Tab 1** and name it `Tasks` in the Tabs property inspector pane (on the right). Click **Tab 2**, and use the same process to change the name of that tab to `Costs`. Click **Tab 3**, and click **Delete** in the Tabs property inspector pane to remove the tab from the application.

Onto the first tab, drop a new table component. In the Table Creation wizard, use the same general process you used to create the Project business object earlier in this article. Click **New Business Object**, name the object `Tasks`, and add the following fields:

- `Task`, a character field
- `Cost`, a currency field
- `Owner`, a lookup

To create the **Owner** lookup, enter `owners` in the **Create New** field, and click **Define Lookup Values**. In the next dialog box, click **Add Lookup Value** and enter three values: `David`, `Jane`, and `Jack`. Click **Add Lookup Value** before adding each new value. Click **OK**. Click **Finish**, and click **Run**.

The application runtime shows the Edit Projects page with the **Tasks** and **Costs** tabs and a couple of additional tasks, as shown in **Figure 6**. Refer to the article's

Figure 6: Edit Projects, with Tasks and Costs tabs

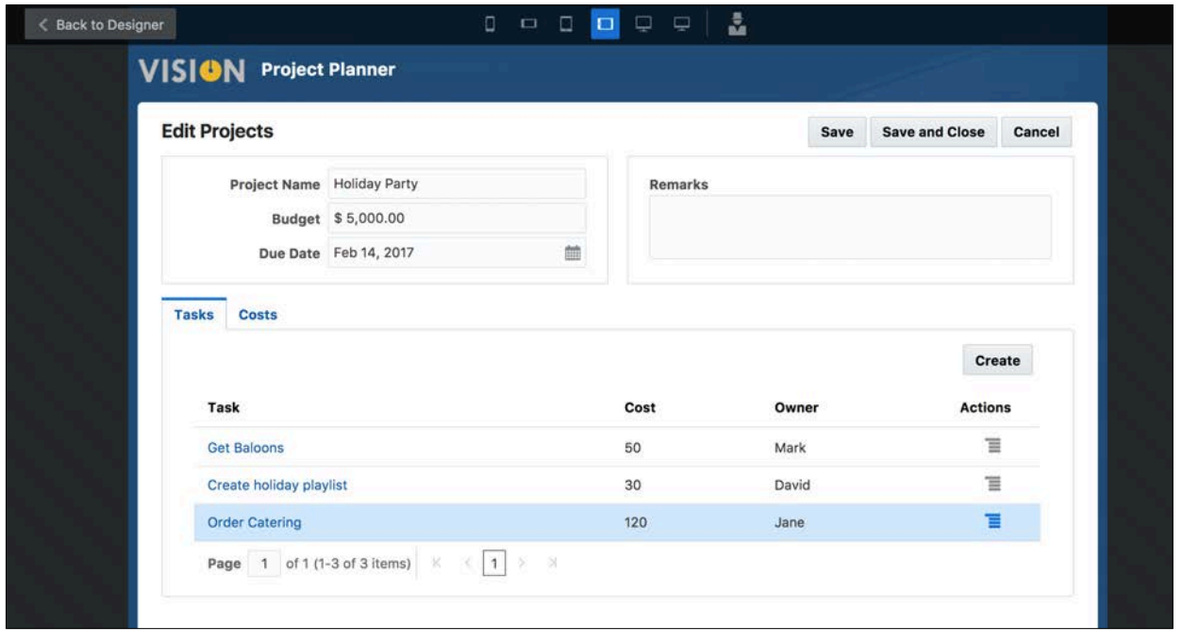
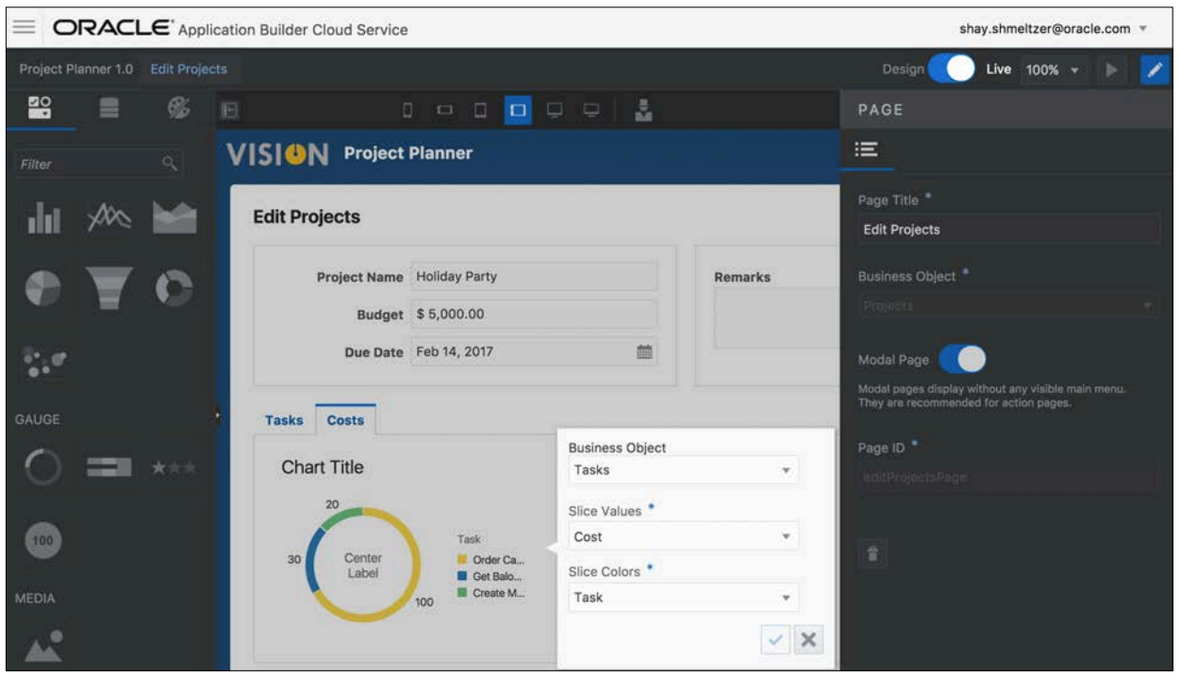


Figure 7: Dragging and dropping a chart



video for instructions for adding the tasks.

Oracle Application Builder Cloud Service offers a collection of data visualization components, including various charts and gauges that help represent data in meaningful ways. The article's video shows how to add and configure a doughnut chart to represent tasks and their costs, as shown in **Figure 7**.

Although you have used a UI-first approach to building the complete application, if you were to dig deeper into what has been created, you would discover a set of tables in an Oracle Database instance, a set of REST services providing full CRUD access to the data, and an Oracle JET–based set of web pages.

POWERFUL DATA DESIGNER

A UI-first approach is just one of the approaches Oracle Application Builder Cloud Service supports. Oracle Application Builder Cloud Service also includes a powerful data designer component.

Navigate to the data designer from the main (hamburger) menu of Oracle Application Builder Cloud Service. You'll see that you can manage both custom objects you created as well as new external REST services you can add to your application as

Figure 8: Configuring the Expenses field to aggregate the costs of tasks

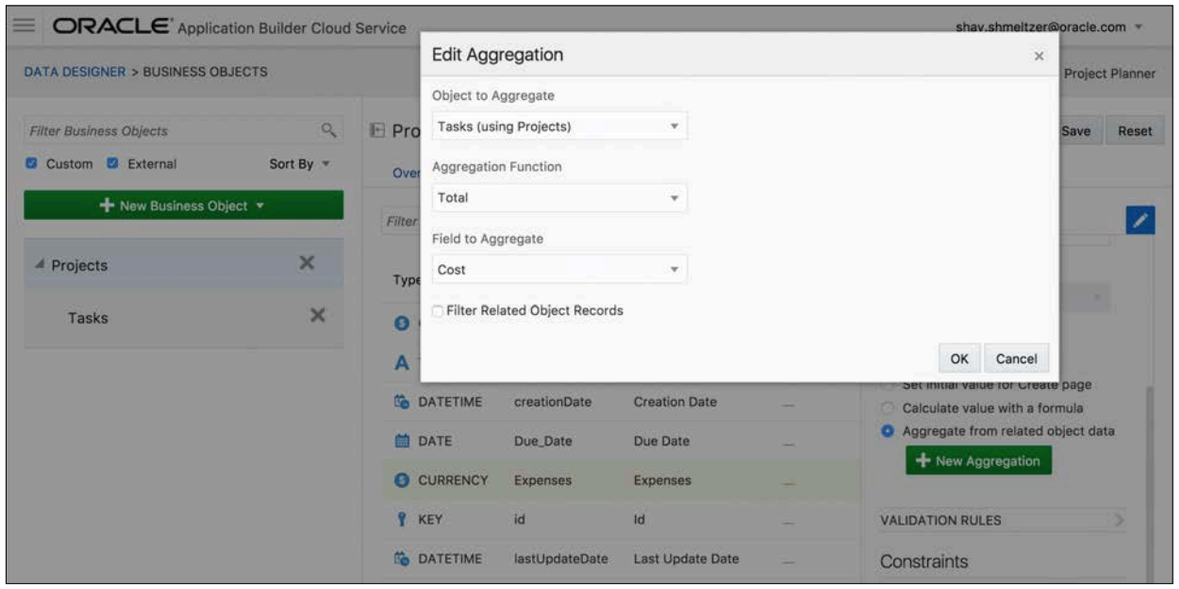
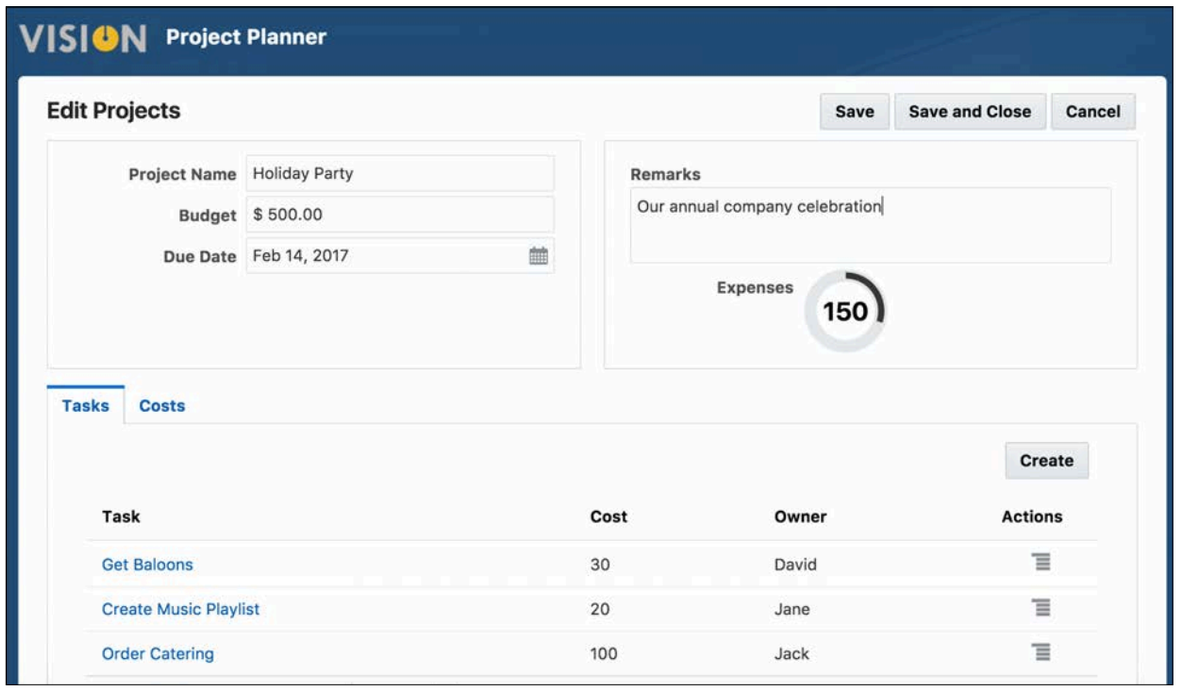


Figure 9: The completed application



extra datasources. If your Oracle Application Builder Cloud Service instance is located in the same cloud instance as an Oracle SaaS application, you will see the REST services those apps expose in a service catalog—for easy addition to your application.

For this application, focus on editing the custom components you've created. The article's video shows how to

- Use the data designer to add an **Expenses** field (of type Currency)
- Configure the **Expenses** field to aggregate the cost of tasks (as shown in **Figure 8**)
- Add a circular status meter (under the Gauge label) object to the application. Configure that object to show expenses from 0 to the allocated budget

The finished application looks similar to the application in **Figure 9**.

SHARING YOUR APPLICATION WITH THE WORLD

Oracle Application Builder Cloud Service is not just about easy development; it also simplifies the process of rolling out your application to production.

The stage option in the main menu of Oracle Application Builder Cloud Service creates a new instance of your application with a URL you can share with team members who you would like to test your

application. This instance of the application has its own copy of the database, which can be blank or include data from your development environment.

When you are happy with the test results, you can publish the application—creating another instance that will be your production application.

After an application is published, you can manage further changes to the application through Oracle Application Builder Cloud Service's automatic version management features. Working on a new version doesn't break the existing code, and upgrading to the new version is seamless, because Oracle Application Builder Cloud Service takes care of advanced rollout functionality, including updating existing database tables and columns with the latest changes.

WHAT'S NEXT?

This article only scratched the surface of Oracle Application Builder Cloud Service. The environment enables you to create much more complex and advanced applications, manage security, import data from external sources, create mobile apps, and more. Future articles will explore these features.

Shay Shmeltzer is director of product management for Oracle's development tools and frameworks. He is focused on helping developers simplify and streamline their work by leveraging the right tools and technologies.

NEXT STEPS

TRY Oracle Application Builder Cloud Service.

LEARN more about Oracle Application Builder Cloud Service.

EXPLORE the Oracle Application Builder Cloud Service Channel.

By Chris Muir

ORACLE MOBILE CLOUD SERVICE

Extending Mobile Apps

Add maps, cameras, and browser support to your Oracle Mobile Application Accelerator app.

In "Empowering the Mobile Citizen Developer," I covered how to create a basic phone book application for your staff in Oracle's citizen development tool: Oracle Mobile Application Accelerator. The result was a very simple app listing all company employees and their phone numbers, and it didn't require me to write a single line of code. The app was rudimentary, with two pages: one a list view of employees and the other a detail page for a selected employee, which then enabled me to phone or email them.

This article explores how easy it is for everyday business users to use Oracle Mobile Application Accelerator to add support for maps and the camera and the ability to launch an in-app browser to view web pages. In the context of my existing phone book app, this will enable me to show the location of employees, take photos of employees to update their photographs, and open websites with other content about the employee.

PREREQUISITES

Much as in the prerequisites section of the previous Oracle Mobile Application Accelerator article, to follow along with the steps in this article, you will need access to Oracle Mobile Cloud Service, which you can obtain by signing up for a free trial on the Oracle Mobile Cloud Service home page. After signing up for the trial and receiving approval, watch and follow the instructions in the video on how to set up and provision your Oracle Mobile Cloud Service instance.

Also, much as you did in the previous article's prerequisites section, you need to set up two team members, *Jeff* the *service developer*, who will work with the Oracle Mobile Cloud Service APIs used by the Oracle Mobile Application Accelerator application, and *Bob* the *business developer*, who will build the Oracle Mobile Application Accelerator application based on the Oracle Mobile Cloud Service APIs built by Jeff. Finally, you will need to create *Mary* the *mobile user,* who will actually use the Oracle Mobile Application Accelerator application built by Bob.

Now I can pick up the app where I left off in the last Oracle Mobile Application Accelerator article. From a tutorial perspective, because I can't guarantee that you completely implemented the previous article's API and application and because I've made some changes to the underlying employee API the app relies on in this article, you'll reimport these. Watch the following video, where I demonstrate how to import a new version of the API, quickly demonstrate how I built the app in the previous article, and finally demonstrate an export of the app for you to import. Before watching the video, download the exported API and exported Oracle MAX app source code. Then follow the steps in video 01.

WATCH THE VIDEO

ADDING A MAP

Inherently, mobile devices are, well, mobile, because users roam the world during their daily lives. As a result, maps and locations are a core feature of mobile applications. Map and location features include showing the user's current location, the location of another person or place, or even a range of locations plotted as markers so you can see their relative distances from each other.

The implementation of maps is somewhat more complex than a marker on a map image, though. A location can be identified by a latitude and a longitude, which are relatively easy to plot, but there are also more-complex use cases, such as where only a human-readable address is available.

WATCH THE VIDEO

Both of these use cases—that is, the ability to plot one or more locations with latitude and longitude *or* addresses—are supported by Oracle Mobile Application Accelerator. Check out video 02 for details on how to do this.

ADDING CAMERA SUPPORT

My app also shows pictures of all employees, a handy feature that enables users to put faces to names. But, of course, employees change over time: they grow a few gray whiskers, change their hair, or grow two heads. Given these kinds of changes, it would be good to give users of my app the ability to upload new photos to keep their pictures up to date.

WATCH THE VIDEO

To add support for updating an employee's picture, the underlying REST API needs to support an explicit POST API for updating employee photos. In the API you've uploaded for this article, I've provided this for you as POST employees/:id/updatePhoto. Watch video 03 to see how I add camera support to update an employee's image.

WATCH THE VIDEO

ADD SUPPORT TO LAUNCH IN-APP BROWSER

Employees also have their own home pages and other relevant links, such as Twitter and Facebook. It would be handy to be able to open links to these and other web-sites in my application, in a separate browser session inside the app.

Adding support for hyperlinks is as easy as having URIs in the data returned from the APIs and mapping the resulting fields to a "Link" field type. Watch video 04 to see how.

CONCLUSION

From the perspective of a "traditional" mobile developer, adding maps, camera, and in-app browser support to mobile apps takes a considerable amount of coding and configuration. From the perspective of an Oracle Mobile Application Accelerator business developer, adding this functionality to your mobile apps is *easy*. As you've seen in this article's videos, you can expand your app's functionality with just a few clicks.

Chris Muir is a senior principal product manager for mobility, cloud, and development tools at Oracle.

PHOTOGRAPHY BY
SABINE ALBERS/THE VERBATIM AGENCY

NEXT STEPS

READ more about Oracle Mobile Cloud Service.

WATCH Oracle Mobile Cloud Service YouTube training.

JOIN the Oracle Mobile LinkedIn community.

TRY Oracle Mobile Cloud Service.

DOWNLOAD
the custom API for this article.

the exported Oracle Mobile Application Accelerator code.

ORACLE APPLICATION EXPRESS

See Better Results

Build better data visualizations with Oracle Application Express 5.1 charts.

By Joel Kallman

The data visualization engine of the recently released Oracle Application Express 5.1 is now powered by Oracle JavaScript Extension Toolkit (Oracle JET), a modular open source toolkit based on modern JavaScript, CSS3, and HTML5 design and development principles. The charts in Oracle Application Express 5.1 are powered by Oracle JET and fully HTML5 capable, and they work in any modern browser, regardless of platform or screen size. These charts provide numerous ways to visualize a data set, including bar, line, area, range, combination, scatter, bubble, polar, radar, pie, funnel, and stock charts.

In this *Oracle Magazine* article, you're going to build a web application on top of US flight arrival data, and using the newly integrated charts from Oracle JET, you will easily create several data visualizations to better interpret the results.

This article's sample application is built in Oracle Application Express 5.1. If you're not already running Oracle Application Express 5.1 or later locally, you can request

a free workspace at https://apex.oracle.com. Alternatively, you can download the Database App Development Virtual Machine from the Oracle Technology Network, which includes a preconfigured Oracle Database 12*c* Release 1 Enterprise Edition database, Oracle Application Express 5.1, Oracle REST Data Services, Oracle SQL Developer, and Oracle SQL Developer Data Modeler. You will also need to download and unzip the comma-separated value (CSV) file for this article to load the sample data.

ABOUT ORACLE APPLICATION EXPRESS

Oracle Application Express is a high-productivity, low-code platform for creating modern, responsive, and accessible web applications. A no-cost feature of Oracle Database, it is a compelling application development platform available in all Oracle Database Cloud services.

CREATING THE APPLICATION

Begin your Oracle Application Express 5.1 data visualization exploration by creating the initial application:

1. In a web browser, log in to Oracle Application Express and click the **App Builder** icon.
2. Click the **Create** icon, and then click the **From a spreadsheet** link.
3. Select **Upload file**, and click **Next**.
4. Choose the supplied **airline_delay_causes_102016.csv** file, enter `"` for **Optionally Enclosed By**, and click **Next**.
5. Enter `AIRLINE_DELAYS` for **Table Name**, and click **Next**.
6. Change **Application Name** to `Airline Delays`, choose **Interactive Grid** for **Report Type**, choose **Single Page** for **Page Type**, and click **Create Application**.

You've now created a table in Oracle Database; inserted your spreadsheet data as rows into your table; and created a fully functional, multiuser, editable grid on this table, accessible from a web browser.

Click the **Run Application** icon, and log in with your workspace credentials. Your application should look similar to **Figure 1**.

INTERACTIVE GRID DATA VISUALIZATIONS

The interactive grid, a new component in Oracle Application Express 5.1, enables you to easily manipulate data simply by clicking on a cell and editing its value. The interactive grid includes many features for powerful reporting, including fixed headers, frozen columns, scroll pagination, multiple filters, sorting, aggregates, computations, and more. A future *Oracle Magazine* article will discuss the interactive grid in greater depth. Right now, let's move on to data visualizations!

The Oracle Application Express 5.1 charting engine for interactive grids has been directly integrated with Oracle JET charts, which results in greatly improved visualizations and also a wider array of built-in chart types (such as bubble, polar, radar, range, and scatter). Charts can be easily created in interactive grids and saved as custom views. End users can also create and save their own custom charts on the data.

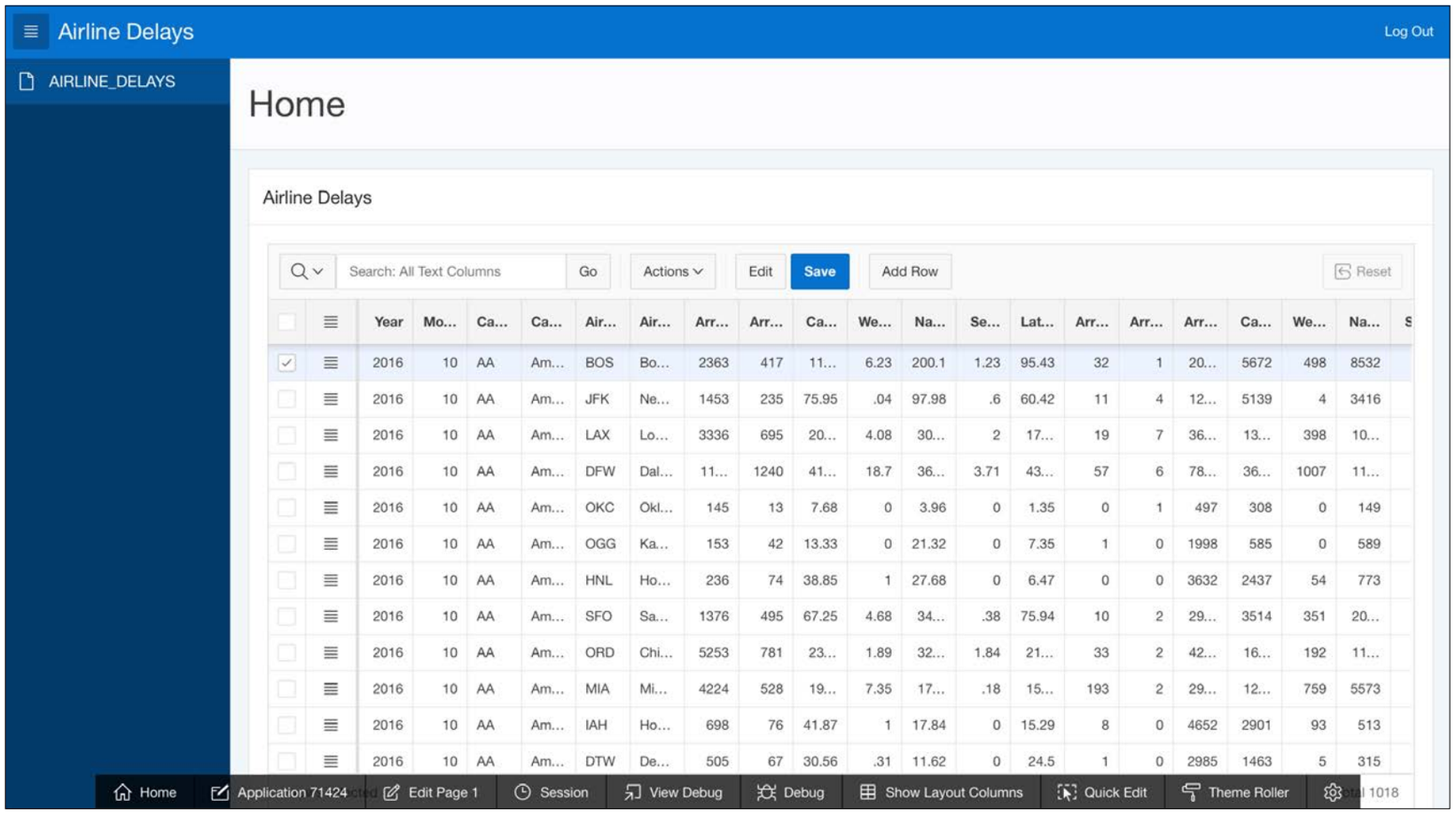


Figure 1: Application home page with interactive grid

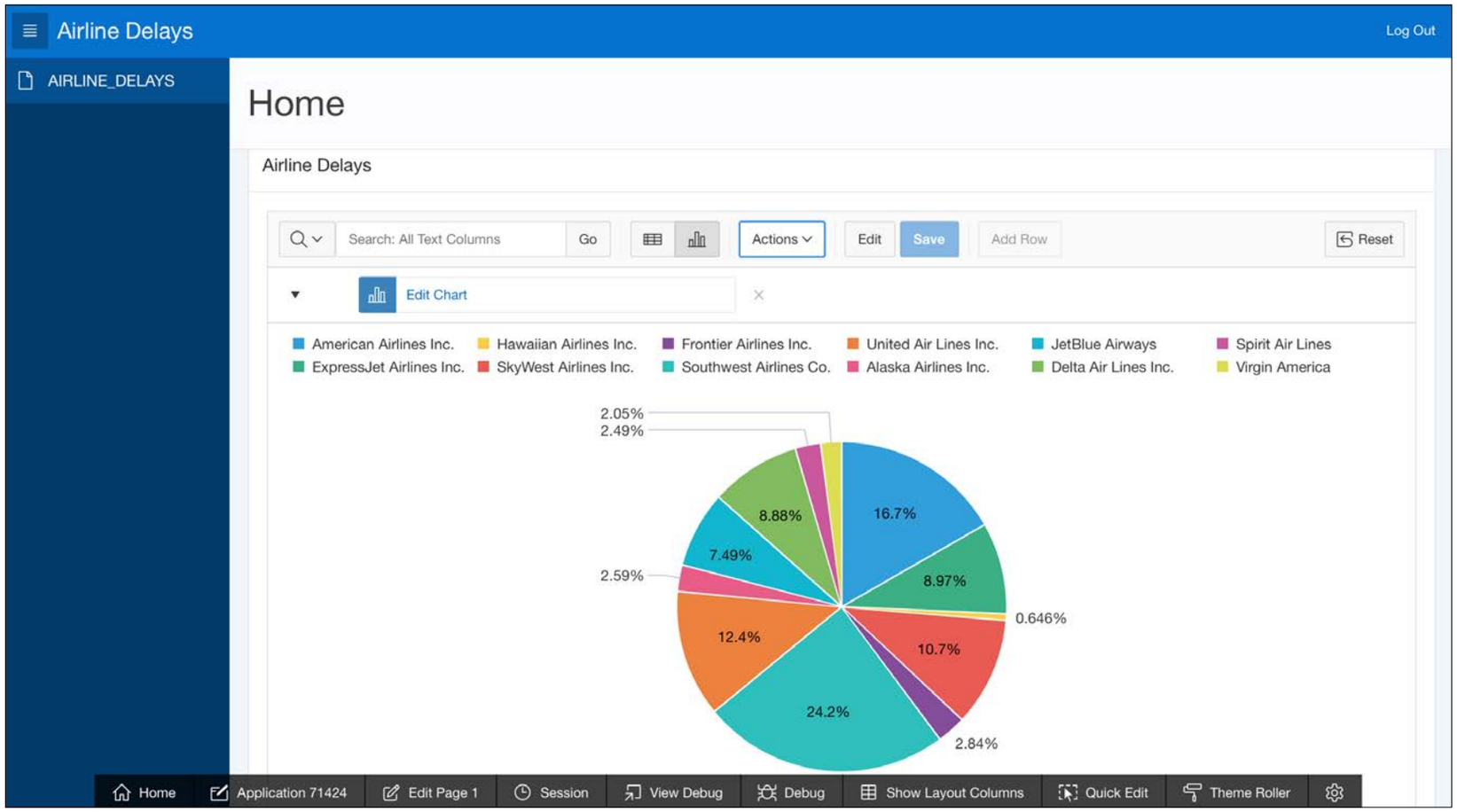
Now add a pie chart to your Oracle Application Express 5.1 application:

1. Ensure that you're running page 1 of your application, the report on the AIRLINE_DELAYS table
2. Click the **Actions** button at the top of the report, and click **Chart**.

3. Click the **Pie** icon.
4. Choose **Carrier Name** for **Label**, **Arr Del15** for **Value**, and **Sum** for **Aggregation**, and then click the **Save** button.

Your application should look similar to **Figure 2.** You have created a pie chart showing, on a percentage basis, the total number of flight delays 15 minutes or greater in the US for October 2016. Hover over the slices of the pie chart to get greater detail about this particular data point. Hover over each of the airline names in the legend, observing the highlighting of the respective slice. Click an airline name in the legend to remove that portion of data from the pie chart. These are all examples of the out-of-the box functionality present in Oracle JET charts.

Figure 2: Pie chart showing flight delays on a percentage basis



You can save your chart as a custom view of the data in the interactive grid. Click the **Actions** button at the top of the report, click **Report,** and then click **Save**. At the top of the report, you'll now notice an icon, enabling you to toggle between the grid view and the chart view.

CUSTOM DATA VISUALIZATIONS

The formatting of charts in interactive grids is concise and limited, intended to

cover the majority of basic charting needs. Oracle Application Express 5.1 also supports chart regions, which offer a greater array of customization options and showcase the power of Oracle JET charts and the native declarative integration with Oracle Application Express.

Add a page to the application with a new chart region to show the aggregate counts of airline delays by carrier:

1. Navigate in your web browser to the browser tab running App Builder (and not the browser tab running your application).
2. Click the **Create Page** button.
3. Click the **Chart** icon.
4. Click the **Bar** icon.
5. Enter `Delays by Carrier` for **Page Name**, and click the **Next** button.
6. Click **Create a new navigation menu entry** for **Navigation Preference**, and click **Next**.
7. Ensure that **SQL Query** is selected for **Source Type**, and enter the following SQL query:

```
select sum(carrier_ct) value, carrier_name, 'Carrier' as series from airline_delays
group by carrier_name
union all
select sum(weather_ct) value, carrier_name, 'Weather' series from airline_delays
group by carrier_name
union all
select sum(nas_ct) value, carrier_name, 'National Aviation System' as series
from airline_delays group by carrier_name
union all
```

```
select sum(late_aircraft_ct) value, carrier_name, 'Late Aircraft' as series
from airline_delays group by carrier_name
order by carrier_name
```

The full power of Oracle SQL can be used in Oracle Application Express to create beautiful data visualizations.

8. Click **Next**.
9. Choose **CARRIER_NAME** for **Label Column**, choose **VALUE** for **Value Column**, and click **Create**.
10. Click the blue **Run** button in the upper right to run your page.

You have now created a vertical bar chart, showing the sum of all delays for each airline and specific type of delay.

At this point, the chart probably looks jumbled and confusing to you, but with the adjustment of a few small attributes, you can easily improve it:

1. Click **Edit Page 2** in the developer toolbar at the bottom of the page. This will bring you back to the tab running App Builder, ready to edit your newly created page and chart.
2. In the tree pane on the left side of the page, click **Series 1**. The properties for the chart series will be displayed in Property Editor, on the right side of Page Designer.
3. Scroll down the list of properties, and select **SERIES** for **Series Name**.
4. Click the **Run** button in the upper right to save your changes and run your page again.

The chart is looking better, now that you have distinguished the separate series in the chart. The data labels in the chart, however, simply add clutter and should be removed. The information will be easier to consume if the chart orientation is changed too:

1. Navigate in your web browser to the browser tab running App Builder and Page Designer (and not the browser tab running your application).
2. In the tree pane on the left side of the page, click **Attributes** under **Delays by Carrier**. These are the overall attributes associated with the chart.
3. In Property Editor on the right side of Page Designer, set **Orientation** to **Horizontal**. Also change **Stack** to **Yes**.
4. In the tree pane on the left side of the page, click **Series 1**.
5. In the list of series properties in Property Editor on the right side of the page, in the **Label** region, change **Show** to **No**.
6. Click the **Run** button in the upper right to save your changes and run your page again.

Figure 3: Stacked bar chart showing flight delays

By adjusting a few attributes, and without writing any additional code, you've created a very attractive horizontal stacked bar chart, comparing different delays across different airlines. Your chart page should now look similar to **Figure 3.**

Click one segment of the bar chart, and then use the arrow keys on your keyboard to navigate around the various segments of the chart. Resize your browser

window until it approximates the dimensions of a smartphone. Another great benefit of Oracle Application Express applications and the new Oracle JET charts is that they are responsive out of the box. Your applications will display appropriately across desktop and mobile devices.

CONCLUSION

The new charting engine in Oracle Application Express 5.1, based on Oracle JET, is a great advancement in the breadth and quality of data visualizations possible with Oracle Database and Oracle Database Cloud, and the low-code nature of Oracle Application Express provides a high level of developer productivity. As you've seen in this article, you can quickly and easily enhance a bar chart without writing a lot of code, and no separate deployment steps are required between development and execution.

Oracle Senior Director of Software Development Joel Kallman is responsible for the development and product management of Oracle Application Express. He is also a contributing author of several books on Oracle technology.

NEXT STEPS

DOWNLOAD

Database App Development VM.

Oracle Application Express 5.1.

the spreadsheet file for this article.

READ

more about what's new in Oracle Application Express 5.1.

more about Oracle JET.

REQUEST a free Oracle Application Express workspace.

TRY Oracle Database Cloud services.

By Steven Feuerstein

TAKE THE CHALLENGE

ORACLE DATABASE 12*c* RELEASE 2

Get Up to Speed with DBMS_SQL

Explore the latest DBMS_SQL features in Oracle Database 12*c* Release 2.

DBMS_SQL offers a broad and deep API for executing dynamic SQL and dynamic PL/SQL blocks from within PL/SQL. Most of the common and simpler dynamic SQL requirements are handled through native dynamic SQL and the EXECUTE IMMEDIATE statement. Yet DBMS_SQL is still quite handy for some of the more challenging tasks, including dynamic SQL method four, which lets your program execute dynamic SQL statements that contain a varying number of bind variables.

So it is important to keep up with DBMS_SQL and its latest capabilities for working with dynamic SQL in PL/SQL. In Oracle Database 12*c* Release 1 (Oracle Database 12.1), the EXECUTE IMMEDIATE statement was extended to support user-defined PL/SQL types, allowing you to, for example, bind a Boolean value in the USING clause. Now, in Oracle Database 12*c* Release 2 (Oracle Database 12.2), those same capabilities have been extended to DBMS_SQL, giving you more flexibility

when handling complex dynamic PL/SQL requirements, and enabling you to come up with a simpler solution.

First, let's take a look at the problems you would encounter in Oracle Database 12.1 and earlier releases if you tried to use DBMS_SQL with PL/SQL–specific and user-defined types.

Suppose I have this simple procedure defined in my schema:

```
PROCEDURE pass_boolean (b IN BOOLEAN) AUTHID DEFINER
IS
   local_b   BOOLEAN;
BEGIN
   local_b := b;
END;
```

Using a release prior to Oracle Database 12.2, if I try to bind a Boolean variable with DBMS_SQL, I see this error:

```
DECLARE
   cur NUMBER;
BEGIN
   cur := DBMS_SQL.open_cursor ();
   DBMS_SQL.parse (cur
      , 'BEGIN pass_boolean (:my_Boolean); END;'
      , DBMS_SQL.native);
   DBMS_SQL.bind_variable (cur, 'my_boolean', TRUE);
```

```
   dummy := DBMS_SQL.execute (cur);
   DBMS_SQL.close_cursor (cur);
END;
/

ORA-06550: line 8, column 4:
PLS-00306: wrong number or types of arguments in call to 'BIND_VARIABLE'
```

The same thing happens when I try to bind to a user-defined type, as shown in **Listing 1**.

Code Listing 1: Trying to bind a user-defined type before Oracle Database 12.2

```
CREATE OR REPLACE PACKAGE my_pkg AUTHID DEFINER
IS
   TYPE r IS RECORD (n NUMBER);
END;
/

CREATE OR REPLACE PROCEDURE pass_record (r_in IN my_pkg.r)
   AUTHID DEFINER
IS
BEGIN
   NULL;
END;
/
```

```
DECLARE
   cur   NUMBER;
BEGIN
   cur := DBMS_SQL.open_cursor ();
   DBMS_SQL.parse (cur,
                   'BEGIN pass_record (:my_record); END;',
                   DBMS_SQL.native);
   DBMS_SQL.bind_variable (cur, 'my_record', TRUE);
   dummy := DBMS_SQL.execute (cur);
   DBMS_SQL.close_cursor (cur);
END;
/
```

```
ORA-06550: line 8, column 4:
PLS-00306: wrong number or types of arguments in call to 'BIND_VARIABLE'
```

Now let's see how things work in Oracle Database 12.2.

BIND TO BOOLEANS

Booleans, famously, are a native datatype in PL/SQL, but they are not supported natively in SQL. This is no longer a "blocking factor" when it comes to executing dynamic PL/SQL blocks that need to bind a Boolean.

Listing 2 demonstrates an example of constructing and executing a dynamic PL/SQL block, binding a Boolean as an IN value.

Code Listing 2: Binding a Boolean in Oracle Database 12.2

```
DECLARE
   c_with_boolean   CONSTANT VARCHAR2 (2000)
      := q'[
    DECLARE
       x BOOLEAN := :my_boolean;
    BEGIN
       IF (x) then
          DBMS_OUTPUT.PUT_LINE ('value of x = ' || 'true');
       ELSE
          DBMS_OUTPUT.PUT_LINE ('value of x = ' || 'false');
       END IF;
    END;]' ;

   dummy                    NUMBER;
   cur                      NUMBER;
BEGIN
   cur := DBMS_SQL.open_cursor ();
   DBMS_SQL.parse (cur, c_with_boolean, DBMS_SQL.native);
   DBMS_SQL.bind_variable (cur, 'my_boolean', TRUE);
   dummy := DBMS_SQL.execute (cur);
   DBMS_SQL.close_cursor (cur);
END;
/

value of x = true
```

As you can see from this example, the Oracle Database 12.2 DBMS_SQL.BIND_VARIABLE procedure now includes an overloading for Booleans.

You can also *extract* Boolean values from a dynamic block through a Boolean overloading of the DBMS_SQL.VARIABLE_VALUE procedure.

To see this functionality in action, first I create a table:

```
CREATE TABLE tab (c1 VARCHAR2 (30));
```

Then I create the dyn_sql_bool_tab_out procedure, as shown in **Listing 3**, that will insert rows (first dynamically and then with static SQL inserts) into the table and, along the way, change the value of the variable.

Code Listing 3: Inserting rows and changing variable values

```
CREATE OR REPLACE PROCEDURE dyn_sql_bool_tab_out (b_in IN BOOLEAN)
   AUTHID DEFINER
AS
   local_b   BOOLEAN := b_in;
   str       VARCHAR2 (3000);
   dummy     NUMBER;
   cur       NUMBER;
BEGIN
   str := '
BEGIN
   IF :v1
   THEN
      :v1 := FALSE;
```

```
      INSERT INTO tab VALUES (''first true'');
   ELSE
      :v1 := TRUE;

      INSERT INTO tab VALUES (''first false'');
   END IF;
END;';

   cur := DBMS_SQL.open_cursor ();
   DBMS_SQL.parse (cur, str, DBMS_SQL.native);

   /* Bind the Boolean to pass the value in. */
   DBMS_SQL.bind_variable (cur, 'v1', local_b);
   dummy := DBMS_SQL.execute (cur);

   /* Extract the value back into the variable */
   DBMS_SQL.variable_value (cur, 'v1', local_b);

   DBMS_SQL.close_cursor (cur);

   IF local_b
   THEN
      INSERT INTO tab
           VALUES ('then true');
   ELSE
      INSERT INTO tab
```

```
            VALUES ('then false');
   END IF;
END;
/
```

I then execute the dyn_sql_bool_tab_out procedure and see the associated output:

```
BEGIN
   dyn_sql_bool_tab_out (TRUE);

   FOR rec IN (SELECT * FROM tab)
   LOOP
      DBMS_OUTPUT.put_line (rec.c1);
   END LOOP;

   ROLLBACK;

   dyn_sql_bool_tab_out (FALSE);

   FOR rec IN (SELECT * FROM tab)
   LOOP
      DBMS_OUTPUT.put_line (rec.c1);
   END LOOP;
END;
/
```

```
first true
then false
first false
then true
```

BIND TO RECORDS

Records are a very handy type of data in PL/SQL. They allow you to group together related variables under a single name. Suppose in my program I need to manipulate a user's city, favorite color, and age. I could declare three variables, as follows:

```
DECLARE
   l_user_city VARCHAR2(100);
   l_user_fav_color VARCHAR2(100);
   l_user_age NUMBER;
```

But I could also work with a record instead:

```
DECLARE
   TYPE user_rt IS RECORD (
      city VARCHAR2(100),
      fav_color VARCHAR2(100),
      age NUMBER);

   l_user user_rt;
```

And the best place to define user-defined types is in a package specification, so they can be used across your application code base, as in the following:

```
CREATE OR REPLACE PACKAGE my_pkg
   AUTHID DEFINER
AS
   TYPE user_rt IS RECORD (
      city VARCHAR2(100),
      fav_color VARCHAR2(100),
      age NUMBER);
END ;
/
```

If, however, in Oracle Database 12.1 or earlier releases, I tried to bind and work with a record of this type inside a dynamic PL/SQL block, I would get another PLS-00306 error, as shown in **Listing 4**.

Code Listing 4: Attempting to bind and work with a record (before Oracle Database 12.2)

```
DECLARE
   l_block CONSTANT VARCHAR2 (2000) := q'[
DECLARE
   l_user   my_pkg.user_rt;
BEGIN
   l_user := :my_user;
END;]';
```

```
   dummy  NUMBER;
   cur    NUMBER;
   l_user my_pkg.user_rt;
BEGIN
   cur := DBMS_SQL.open_cursor ();
   DBMS_SQL.parse (cur, l_block, DBMS_SQL.native);
   DBMS_SQL.bind_variable (cur, 'my_user', l_user);
   dummy := DBMS_SQL.execute (cur);
   DBMS_SQL.close_cursor (cur);
END;
/
```

```
ORA-06550: line 17, column 4:
PLS-00306: wrong number or types of arguments in call to 'BIND_VARIABLE'
```

In Oracle Database 12.2, you can now bind to user-defined record types, using a brand-new procedure, BIND_VARIABLE_PKG, which has the following signature:

```
DBMS_SQL.BIND_VARIABLE_PKG (
   c              IN INTEGER,
   name           IN VARCHAR2,
   value          IN <datatype>);
```

<datatype> can be any of the following:

- RECORD
- VARRAY

- NESTED TABLE
- INDEX BY PLS_INTEGER TABLE
- INDEX BY BINARY_INTEGER TABLE

You might be wondering: what's the "_PKG" suffix for? That's just another way of telling you that the datatype of the value passed to the procedure must be declared in a *package specification*.

Because your dynamic SQL statement is executing in the SQL engine, it must be able to resolve references to your types. If a type is declared in the declaration section of a procedure, function, or anonymous block, the SQL engine cannot find it.

So using that same package-based record type for a user, and changing the name of the DBMS_SQL procedure call to BIND_VARIABLE_PKG, I can now work with user-defined records, as shown in **Listing 5**.

Code Listing 5: Binding and working with a record (in Oracle Database 12.2)

```
DECLARE
   l_block CONSTANT VARCHAR2 (2000) := q'[
DECLARE
   l_user   my_pkg.user_rt;
BEGIN
   l_user := :my_user;
   DBMS_OUTPUT.PUT_LINE (
      l_user.city || ' - ' || l_user.fav_color);
END;]';

   dummy  NUMBER;
   cur    NUMBER;
```

```
   l_user my_pkg.user_rt;
BEGIN
   l_user.city := 'Chicago';
   l_user.fav_color := 'Green';

   cur := DBMS_SQL.open_cursor ();
   DBMS_SQL.parse (cur, l_block, DBMS_SQL.native);
   DBMS_SQL.bind_variable_pkg (cur, 'my_user', l_user);
   dummy := DBMS_SQL.execute (cur);
   DBMS_SQL.close_cursor (cur);
END;
/
```

```
Chicago - Green
```

BIND TO INTEGER-INDEXED ASSOCIATIVE ARRAYS

Associative arrays come in two flavors: INDEX BY BINARY_INTEGER (and any of its subtypes, such as the more common and preferred PLS_INTEGER) and INDEX BY VARCHAR2.

The INDEX BY BINARY INTEGER associative array, also referred to as an *ibbi*, was the first type of collection ever added to PL/SQL. This was back in Oracle7 (Oracle Database 7.3), and this type was referred to as a *PL/SQL Table*. String-indexed arrays were added in Oracle Database 11*g* and offer lots of interesting flexibility. They cannot, however, be bound to dynamic PL/SQL blocks, either with DBMS_SQL or EXECUTE IMMEDIATE.

With Oracle Database 12.2, though, you *can* bind integer-indexed associative arrays—even arrays of records—as long as they are declared in a package specification. **Listing 6** demonstrates integer-indexed associative array binding.

Code Listing 6: Binding to an integer-indexed associative array

```
CREATE OR REPLACE PACKAGE my_pkg
   AUTHID DEFINER
AS
   TYPE user_rt IS RECORD
   (
      city        VARCHAR2 (100),
      fav_color   VARCHAR2 (100),
      age         NUMBER
   );

   TYPE users_t IS TABLE OF user_rt
      INDEX BY PLS_INTEGER;
END;
/

DECLARE
   dummy     NUMBER;
   cur       NUMBER;
   l_users   my_pkg.users_t;
   str       VARCHAR2 (3000)
```

```
      := q'[
DECLARE
   dyn_users my_pkg.users_t;
BEGIN
   dyn_users := :my_users;

   DBMS_OUTPUT.put_line (dyn_users (dyn_users.first).city);
   DBMS_OUTPUT.put_line (dyn_users (dyn_users.first).fav_color);
END;]';
BEGIN
   l_users (100).city := 'Chicago';
   l_users (100).fav_color := 'Green';

   cur := DBMS_SQL.open_cursor ();
   DBMS_SQL.parse (cur, str, DBMS_SQL.native);
   DBMS_SQL.bind_variable_pkg (cur, 'my_users', l_users);
   dummy := DBMS_SQL.execute (cur);
   DBMS_SQL.close_cursor (cur);
END;
/

Chicago
Green
```

But if I change to a string-indexed collection in the package specification, as follows, and run the anonymous block again, I will get a PLS-00306 error.

```
TYPE users_t IS TABLE OF user_rt
   INDEX BY VARCHAR2(100);
```

STILL RUNNING INTO THE PLS-00306 ERROR?

When you first start working with this Oracle Database 12.2 extended version of DBMS_SQL, you might find yourself encountering the PLS-00306 error that you expected to avoid.

Here are some of the reasons you might be hitting that error:

- You are using BIND_VARIABLE instead of BIND_VARIABLE_PKG (or vice versa). Use BIND_VARIABLE for Booleans, but use BIND_VARIABLE_PKG for all user-defined types.
- Your type is declared locally. User-defined types must be declared in the specification of a package in order for them to be available to DBMS_SQL. That package name must be included as a prefix to the type, both in the "outer" static block and in the dynamic PL/SQL block executed via DBMS_SQL.EXECUTE.
- You are trying to bind to an associative array that is indexed by VARCHAR2. This is not supported for either DBMS_SQL or EXECUTE IMMEDIATE.

DBMS_SQL: STILL GOING STRONG

There was a time when the *only* way you could implement dynamic SQL inside PL/SQL was through the use of DBMS_SQL. Then native dynamic SQL, with the simple EXECUTE IMMEDIATE statement, was introduced. This left DBMS_SQL to tackle only the most complex dynamic SQL requirements.

So while you might not run into a need for DBMS_SQL very often, when you do, make the best possible use of it. In Oracle Database 12.2, DBMS_SQL is now able to

work with an expanded set of PL/SQL datatypes, greatly improving the scope of the problems it can solve.

TESTYOUR NEW DBMS_SQL KNOWLEDGE

SEE ANSWERS BELOW

QUESTION

Which of the following blocks will execute without an error?

A

```
DECLARE
      c_with_boolean   CONSTANT VARCHAR2 (2000)
         := q'[
       DECLARE
          x BOOLEAN := :my_boolean;
       BEGIN
          NULL;
       END;]' ;

   dummy NUMBER;
   cur   NUMBER;
BEGIN
   cur := DBMS_SQL.open_cursor ();
   DBMS_SQL.parse (cur, c_with_boolean, DBMS_SQL.native);
   DBMS_SQL.bind_variable (cur, 'my_boolean', TRUE);
   dummy := DBMS_SQL.execute (cur);
   DBMS_SQL.close_cursor (cur);
```

```
END;
/
```

B

```
DECLARE
   c_with_boolean   CONSTANT VARCHAR2 (2000)
      := q'[
    DECLARE
       x BOOLEAN := :my_boolean;
    BEGIN
       NULL;
    END;]' ;
   dummy NUMBER;
   cur   NUMBER;
BEGIN
   cur := DBMS_SQL.open_cursor ();
   DBMS_SQL.parse (cur, c_with_boolean, DBMS_SQL.native);
   DBMS_SQL.bind_variable_pkg (cur, 'my_boolean', TRUE);
   dummy := DBMS_SQL.execute (cur);
   DBMS_SQL.close_cursor (cur);
END;
/
```

C

```
DECLARE
   TYPE info_rt IS RECORD (info VARCHAR2(100));
```

```
l_info info_rt;

my_block CONSTANT VARCHAR2 (2000)
   := q'[
 DECLARE
    x info_rt := :my_record;
 BEGIN
    NULL;
 END;]' ;

dummy NUMBER;
cur   NUMBER;
BEGIN
   cur := DBMS_SQL.open_cursor ();
   DBMS_SQL.parse (cur, my_block, DBMS_SQL.native);
   DBMS_SQL.bind_variable_pkg (cur, 'my_record', TRUE);
   dummy := DBMS_SQL.execute (cur);
   DBMS_SQL.close_cursor (cur);
END;
/
```

ANSWERS

A | CORRECT

I bind a Boolean variable using the new overloading of DBMS_SQL.bind_variable.

B | INCORRECT
I call the wrong bind procedure: DBMS_SQL.bind_variable_pkg.

C | INCORRECT
I call the right bind procedure, DBMS_SQL.bind_variable_pkg, but the record type is defined locally.

Steven Feuerstein is Oracle's Developer Advocate for PL/SQL, offering guidance on the PL/SQL language through books, quizzes, videos, and more.

PHOTOGRAPHY BY **ANDREA MANDEL**

NEXT STEPS

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

READ *Oracle Database 12c Release 2 PL/SQL Language Reference*.

WATCH Feuerstein's *Practically Perfect PL/SQL* videos.

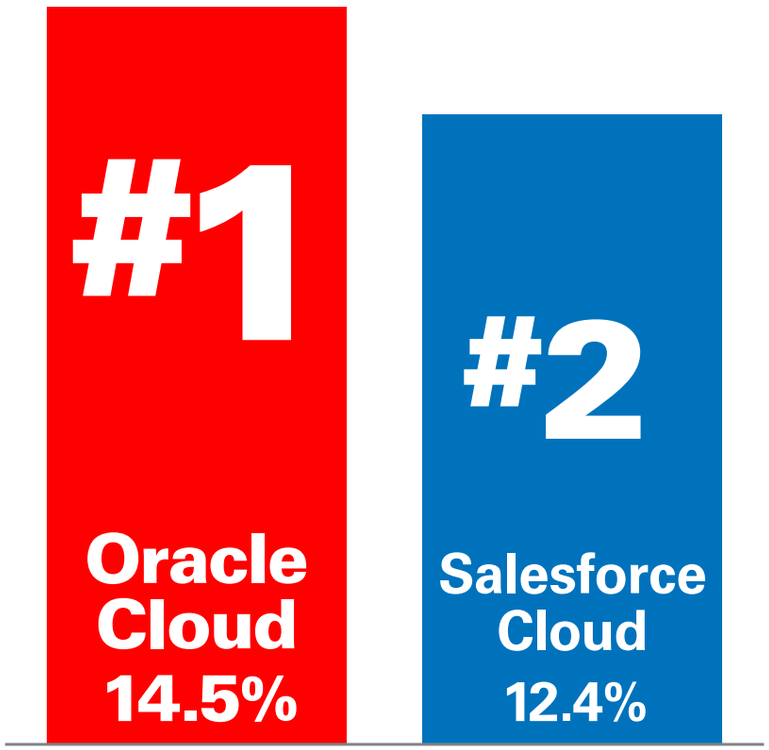
READ more about DBMS_SQL.

MEET the Oracle Developer Advocates team.

CHALLENGE yourself with quizzes on PL/SQL, SQL, database design, and more at the brand-new Oracle Dev Gym.

Oracle #1 SaaS Enterprise Applications Revenue

1,000+ Employees Segment, 2015



Source: IDC "Worldwide SaaS Enterprise Applications Market Shares, 2015: The Top 15 by Buyer Size," doc #US41913816, Dec. 2016; Table 4. For the purposes of this report, SaaS enterprise applications include the following application markets: CRM, engineering, ERP, operations and manufacturing, and SCM.

ORACLE®

ORACLE SQL DEVELOPER

Automatic REST

REST-enable your Oracle Database tables and views with Oracle REST Data Services and Oracle SQL Developer.

By Jeff Smith

Representational State Transfer (REST) is today's dominant software architectural style for creating modern, scalable web services. JavaScript Object Notation (JSON) is the most popular data interchange format for RESTful web services (which use the REST architecture).

Oracle REST Data Services accepts RESTful web service uniform resource identifiers (URIs) and directs them to the appropriate SQL statement or PL/SQL block, returning the output in either JSON or comma-separated values (CSV) format. Oracle REST Data Services 3.0 is easy to install and configure, empowers auto-generated REST endpoints for tables and views, and is supported in Oracle SQL Developer 4.1.

This article will show you how to REST-enable a table and how to query, insert, and update records in it and delete records from it via REST requests—all without writing any custom SQL.

To follow along, you will need to install and configure Oracle REST Data Services with your Oracle Database instance. You will also need a browser and a RESTful client capable of making REST calls. All of this is available via the Oracle Technology Network Developer Day Database Virtual Box Appliance—it includes Oracle Database 12*c* Release 1, Oracle REST Data Services 3.0, and Oracle SQL Developer 4.1 configured and running. The installed appliance appears as a virtual desktop, as shown in **Figure 1**.

For those who have never worked with an Oracle VM VirtualBox appliance before, I've written this quick introduction for getting started with virtual appliances and connecting to Oracle Database 12*c*.

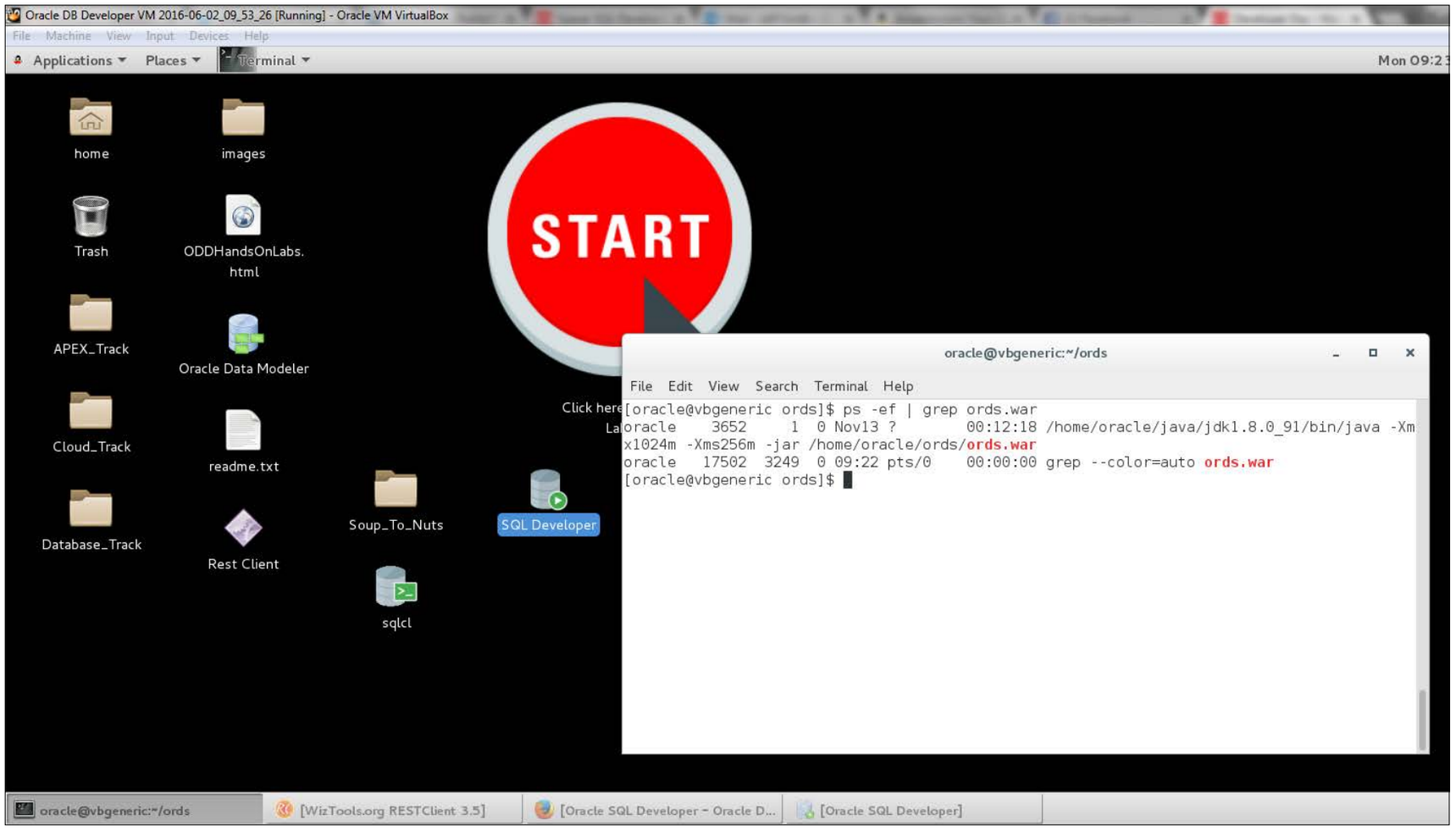


Figure 1: Oracle Technology Network Developer Day Database Virtual Box Appliance

CONFIRMING THAT ORACLE REST DATA SERVICES IS RUNNING

In addition to marshaling REST requests to the database and returning results as JSON to the requester, Oracle REST Data Services can also run as a standalone web server.

To see the web server in action, in the appliance, open a browser and

request the following URL from Oracle REST Data Services:

```
http://localhost:8080/ords/hr/
```

Oracle REST Data Services is listening on port 8080, and you're asking for a RESTful endpoint on the HR schema. Because you have not REST-enabled HR yet, the link should return a 404 error, with a message indicating that the resource, "hr," could not be mapped to a database object.

If you instead get a request timeout, Oracle REST Data Services is not running. If you have just started up the appliance, you may need to wait a few more moments for the database and Oracle REST Data Services to finish their startup processes. You can also start Oracle REST Data Services in the appliance with this command:

```
/home/oracle/java/jdk1.8.0_91/bin/java -Xmx1024m -Xms256m -jar
/home/oracle/ords/ords.war
```

REST-ENABLING A SCHEMA

Before making a table or a view available via REST, you must first enable and map its schema. Use Oracle SQL Developer to REST-enable the HR schema, which is already available and unlocked in the appliance database.

To REST-enable the HR schema, first create a connection in Oracle SQL Developer as HR, with a password of `oracle` to the `orcl` service on `locahost:1521`.

After your connection has been created, right-click the connection in the tree and select **REST Services** -> **Enable REST Services**.

You do not want to expose the Oracle HR username via the REST URI, so you will

be creating the alias "peeps" for HR, as shown in **Figure 2**. Enter `peeps` for **Schema alias**, check **Enable schema**, and click **Next**. For the sake of simplicity and the space constraints of this article, you will not be securing this schema or the REST-enabled table, but it is highly advisable always to do so in your own environments. On the next wizard screen, click **Finish**.

Oracle SQL Developer provides a complete integrated development environment (IDE) for managing your RESTful services and REST-enabling your database objects, and Oracle REST Data Services also provides a complete PL/SQL API. So the same operation in the RESTful Services wizard that REST-enabled the HR schema can also be accomplished via this code block:

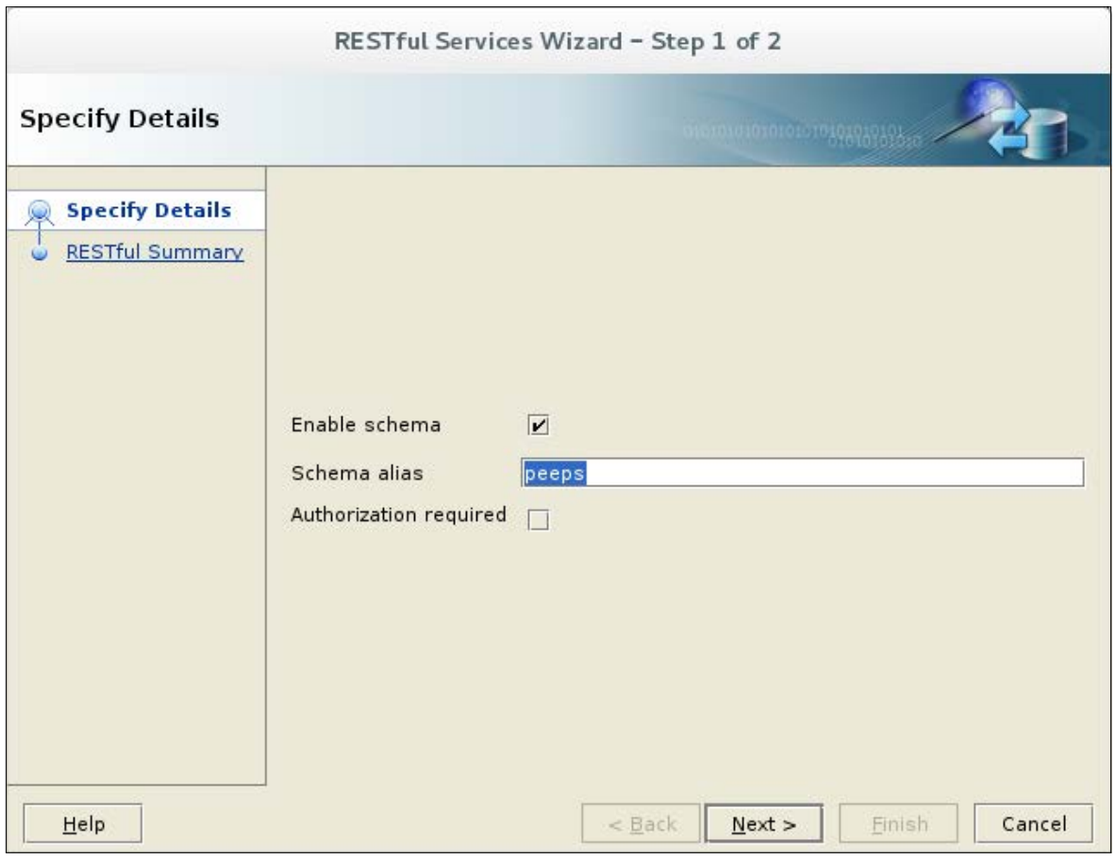
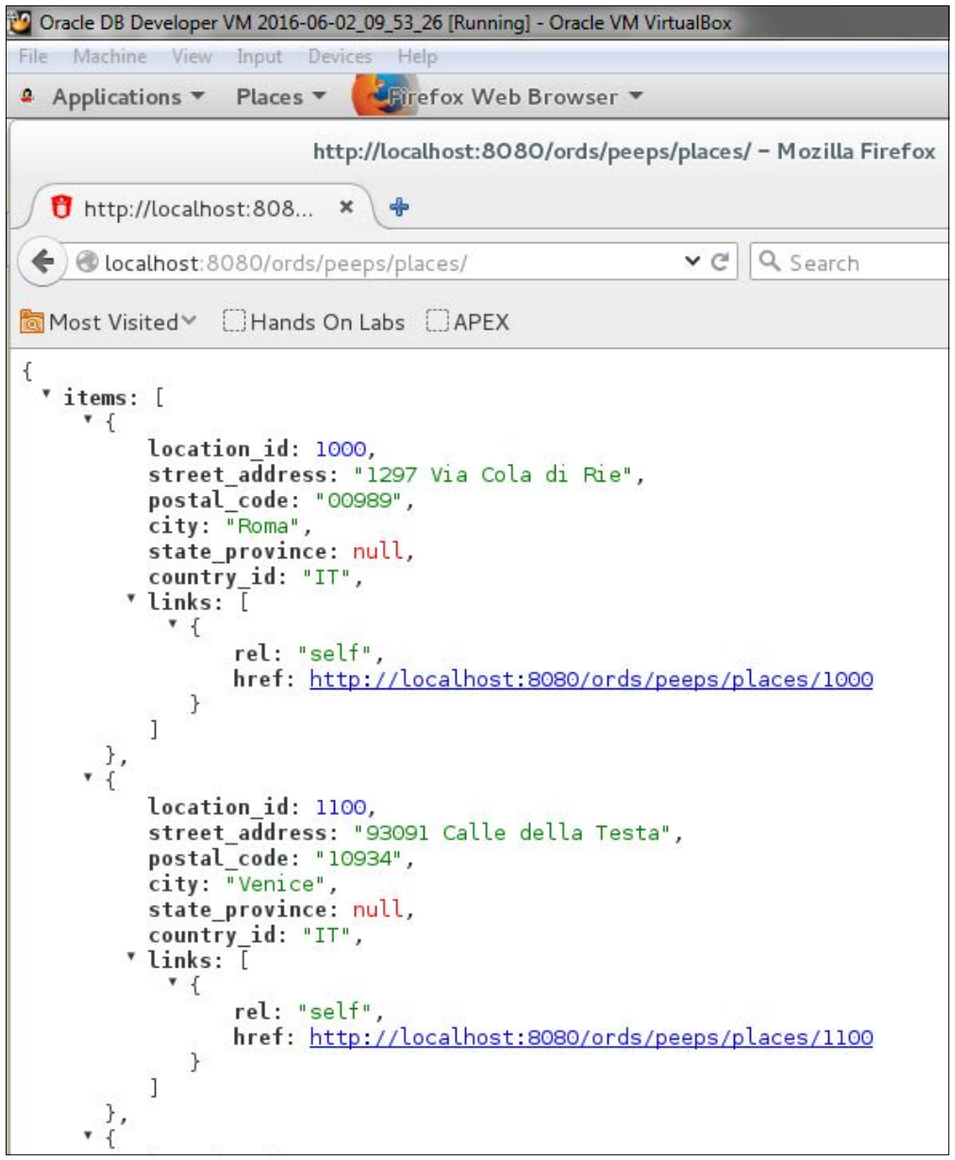


Figure 2: REST-enabling a schema

```
DECLARE
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
   ORDS.ENABLE_SCHEMA(p_enabled => TRUE,
                      p_schema => 'HR',
                      p_url_mapping_type =>
                      'BASE_PATH',
                       p_url_mapping_pattern => 'peeps',
                       p_auto_rest_auth => FALSE);
    commit;
END;
```

With the HR schema REST-enabled, you can start REST-enabling its tables and views. For the remainder of this article, you will be working with the

Figure 3: Generated automatic REST endpoints, accessible via /ORDS/<schema>/<table>/



LOCATIONS table.

Back in Oracle SQL Developer, find the LOCATIONS table in your connection tree. Right-click it, and again select **REST Services** -> **Enable REST Services**.

In the dialog box, check **Enable object**, enter `places` as an **alias**, and leave the **authorization required** item unchecked. This directs Oracle REST Data Services to create the alias "places" for the LOCATIONS table.

Click **Next**, and observe the PL/SQL required to REST-enable the LOCATIONS table in the following dialog box. Click **Finish**.

Now that both the HR schema and the LOCATIONS table have been REST-enabled, you can start making REST requests.

GETS/SELECTS

The automatic REST enablement feature in Oracle REST Data Services provides a complete create, retrieve, update, delete (CRUD) API on your REST-enabled objects.

To see the API in action, issue an HTTP GET request on the "places" endpoint to return a JSON document that delivers the results of your SELECT query on the LOCATIONS table.

You can use a RESTful client or CURL (a command-line tool) to make this request, but you can also point to the resource in your browser, which defaults to GET:

```
http://localhost:8080/ords/peeps/places
```

Making this request in Firefox running in the appliance to the LOCATIONS table looks like **Figure 3**.

Because you requested /places/, you are asking to do a GET (SELECT) of *all* the records in the LOCATIONS table. Note that one of the requirements for a table to be REST-enabled is that it have a primary key defined. In **Figure 3**, observe how each record is addressable by its primary key (location_id) value.

Automatic REST points are resources that can handle all REST requests (GET, POST, PUT, DELETE) automatically. And automatic REST points on your tables do more than just return all records or a specific record. You can also apply sorts and predicates to your automatically generated SELECT statements. These amendments to your GET request are passed along via a `?q=<FilterClause>` appended to the table URI, where *FilterClause* is a JSON document describing the WHERE and/or ORDER BY clause.

The complete syntax rules for these amendments are listed in the *REST Data Services Installation, Configuration, and Development Guide*. To see an example of a predicate with an ORDER BY clause, request LOCATIONS with a LOCATION_ID of "US" ordered by STATE_PROVINCE in ascending order.

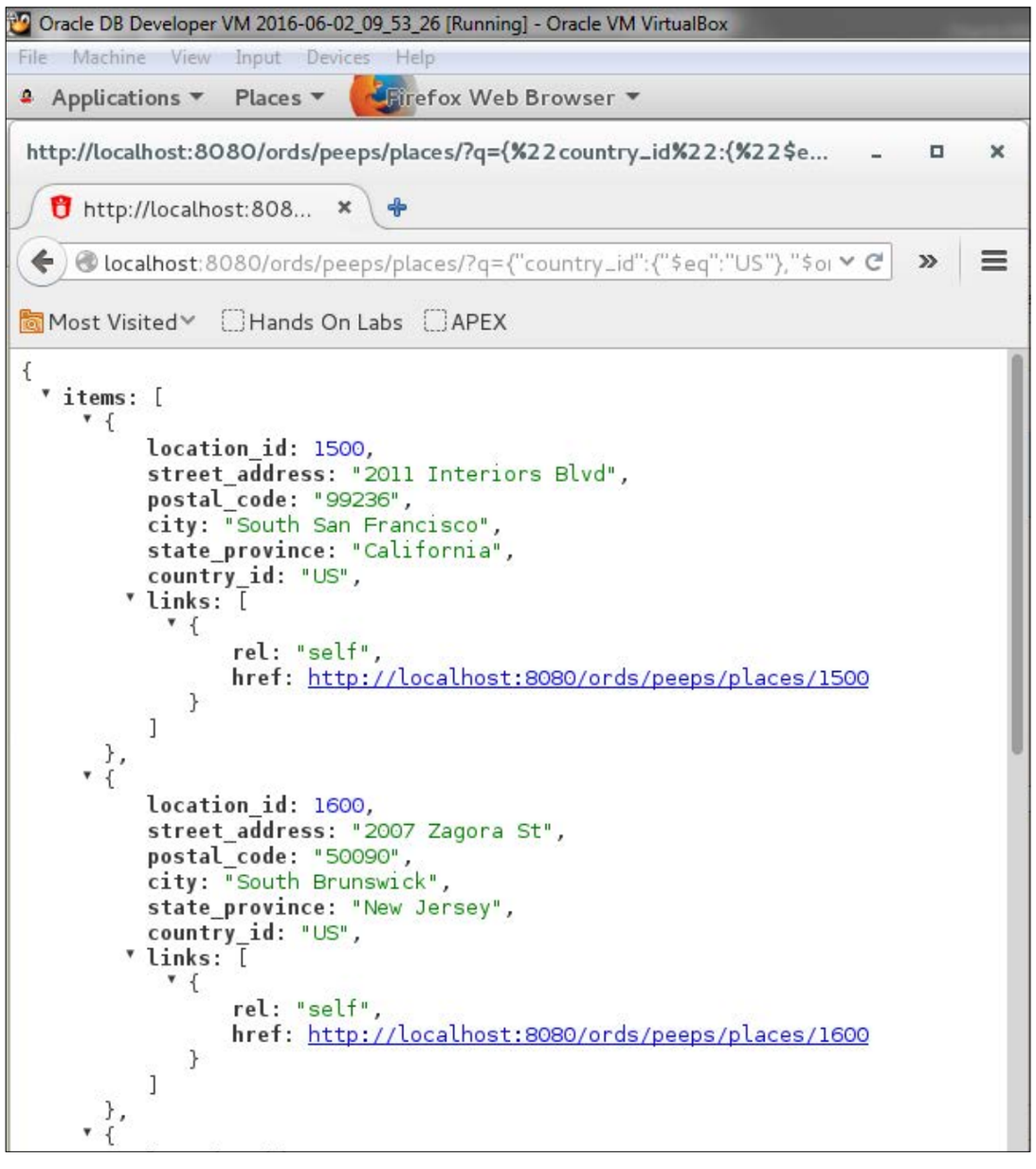
Again in the Firefox browser, ask for this URI to localhost:8080/ords:

```
/peeps/places/?q={"country_id":{"$eq":"US"},"$orderby": {"STATE_PROVINCE":"ASC"}}
```

The records satisfying the predicate are returned in the expected order, as shown in **Figure 4**.

This URI is mapped by Oracle REST Data Services to the HR schema and the LOCATIONS table, and the WHERE and ORDER BY SQL clauses are generated automatically. The application developer needs to know only the resources available and the expected syntax of the JSON describing the query filters.

Figure 4: Locations in the US ordered by STATE_PROVINCE



POSTS/INSERTS

Issuing POST requests requires a RESTful client. The appliance includes one conveniently located on the desktop, "Rest Client." Open it.

To insert a record, issue a POST to /ords/*schema*/*table*/ with a content type of "application/json" and put the record to be inserted into the body of the request in JSON format.

The record in JSON format is simply the list of column names and values to be inserted. Inserting the address of the world's most famous detective would look like this:

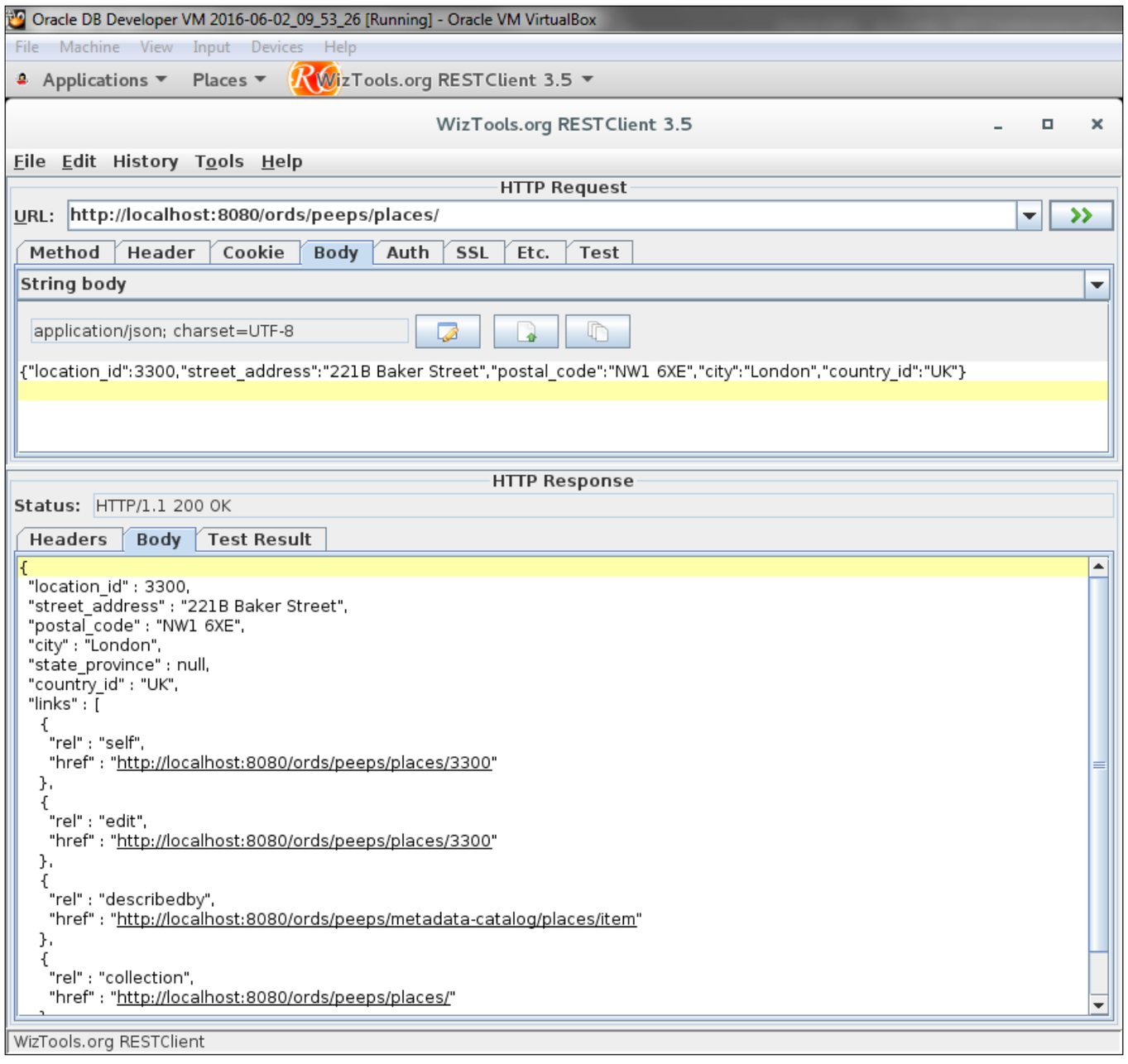
```
{"location_id":3300,"street_address":"221B Baker Street",
"postal_code":"NW1 6XE","city":"London","country_id":"UK"}
```

Issuing this POST to /ords/peeps/places/ will insert a new record whose primary key is 3300.

In the RESTful client, set METHOD to "POST"; paste the JSON into the body, remembering to set the content type to "application/json"; and, for the URL, provide the RESTful endpoint for the LOCATIONS table, `http://localhost:8080/ords/peeps/places/`.

Click the double-green-arrow button to submit the request. Observe the Body panel in the output, as shown in **Figure 5**. (Note that I have used the RESTful client's JSON Formatter feature to make the JSON easier to read. Oracle REST Data Services does not format the

Figure 5: The response to the POST request



JSON responses.)

The response is a 200 (all is good!) followed by the inserted record and how to access it via REST.

The documentation includes the complete process for inserting records via the automatic REST endpoints generated by Oracle REST Data Services when you REST-enable a table.

PUTS/UPDATES

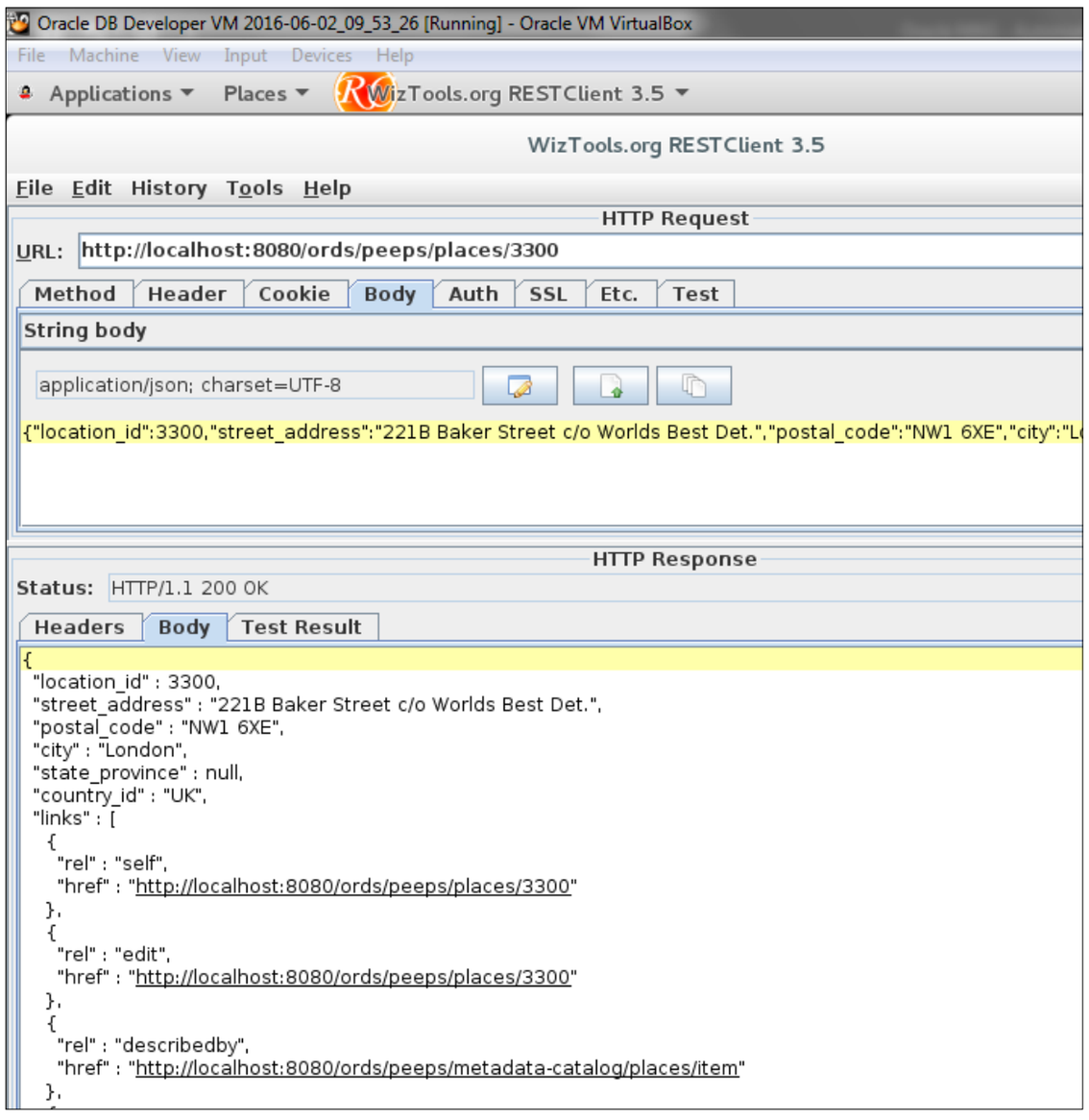
The process for updating a record is much like the process for an INSERT. The two key differences are that you do a PUT versus a POST and that you specify the record to be updated in the URI—for example, `ords/<SchemaAlias>/<ObjectAlias>/<KeyValues>`.

So again using the RESTful client, switch the method to PUT. You will update the record you just inserted with a different street address. Do a PUT of the following to `/ords/peeps/places/3300`:

```
{"location_id":3300,"street_
address":"221B Baker Street c/o
Worlds Best Det.","postal_code":"NW1
6XE","city":"London","country_id":"UK"}
```

Figure 6 shows the result. (Note that I have again used the RESTful client's JSON Formatter feature to make the JSON easier to read. Oracle REST Data Services does not format the JSON responses.)

Figure 6: The response to the PUT—the new record along with the URI to access it



DELS/DELETES

The process for a delete is extremely simple: issue a DEL on the record URI.

Before you delete the record, let's confirm that it exists in the table. Run

```
SELECT * FROM HR.LOCATIONS WHERE LOCATION_ID = 3300;
```

You should see the record you previously inserted and updated.

Now let's issue the DEL, as shown in **Figure 7**.

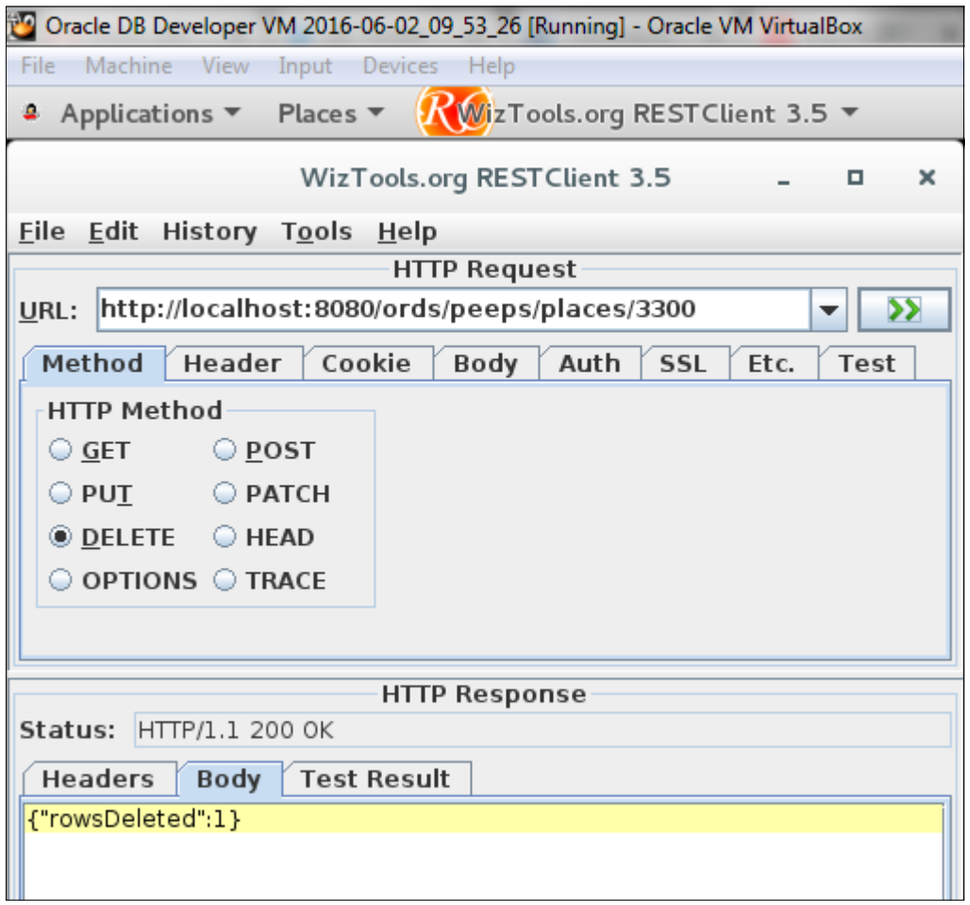
Because the row is gone, the response has no URI by which to reference the now-deleted record. If you rerun the earlier SELECT query in Oracle SQL Developer, you should get no rows returned for LOCATION_ID 3300.

REST is stateless—there is no transaction model. Your INSERTs/UPDATEs/DELETEs are autocommitted.

SUMMARY

In this article, you REST-enabled a schema and a table. You also queried a table and inserted, updated, and deleted a record.

Figure 7: The response confirming that one row was deleted



Note that REST-enabling a resource in the database without protecting it via authentication is not recommended; everyone with access to the URI can now do anything they want to your table or view. The "Securing Your RESTful Service in Oracle REST Data Services" lab, included with the virtual appliance, shows how to secure your REST-enabled objects with third-party Initiative for Open Authentication (OATH) 2.0 authentication.

The automatic REST enablement feature of Oracle REST Data Services 3.0 provides a REST API to your tables and views, but this may not be sufficient for your application. You can also write and publish custom RESTful services by using Oracle SQL Developer and Oracle REST Data Services to run any SQL or PL/SQL block via REST. The REST developer hands-on lab, included with the virtual appliance, demonstrates how to develop and deploy a custom RESTful service.

Jeff Smith, a senior principal product manager in Oracle's Database Development Tools group, is responsible for Oracle SQL Developer and Oracle SQL Developer Data Modeler.

NEXT STEPS

DOWNLOAD Oracle Technology Network Developer Day Database Virtual Box Appliance.

GET a quick introduction to Oracle Technology Network Virtual Box appliances.

READ the *REST Data Services Installation, Configuration, and Development Guide*.

By Connor McDonald

ORACLE DATABASE 12*c* RELEASE 2

Better Tools for Better Data

New functions in Oracle Database 12*c* Release 2 solve data validation challenges.

In the next few columns, I'll spend some time looking at new features in Oracle Database 12*c* Release 2. These features come from the "12 Things About Oracle Database 12*c*" presentation series that Chris Saxon and I, the Ask Tom team, gave at Oracle OpenWorld 2016 in San Francisco. (See asktom.oracle.com, under the **Resources** tab). In this article, I'll take a look at a suite of Oracle Database 12*c* Release 2 improvements for validating data that make loading or querying data from unreliable datasources easier.

BACKGROUND

At the UK Oracle User Group conference (December 2016), during my presentation on Oracle Database 12*c* Release 2 features, I projected a slide containing the acronym QTFWC. I could see many attendees in the audience nodding, acknowledging that this was the next acronym to be digested in an information technology

industry already filled to the brim with acronyms.

The acronym was fictional; there's no such product or feature as QTFWC. It was merely a humorous reflection on a real issue that faces all developers dealing with data whose quality and correctness are unknown: producing Queries That Finish Without Crashing.

In times gone by, data that wasn't in your database was in someone else's database. But datasources now vary widely, and equally variable are the quality and correctness of that data. Developers tasked with querying, cleansing, and loading this data face a chicken-and-egg problem when it comes to datatypes: the only way to know if a string can be converted to, say, a valid number is to attempt the conversion. But to attempt the conversion is to also run the risk of the conversion's being invalid and having the entire query crash.

Consider a load process that must load data from a table called STAGING_SALES, representing sales data staged from several external sources. The task seems simple enough: create an INSERT statement to transfer the raw data into the target table ANNUAL_SALES.

```
SQL>  insert into ANNUAL_SALES
  2   select *
  3   from   STAGING_SALES;
```

A developer may patiently watch this statement execute for minutes, or possibly hours, eagerly waiting for its completion, only to see the following:

```
Elapsed: 06:12:34.00
```

```
ERROR at line 1:
ORA-01847: day of month must be between 1 and last day of month
```

The error suggests a problem with a DATE conversion, but there are no clues as to what data caused it. Adding to the developer's frustration is that the data conversion error probably occurred some three hours into the total elapsed time of six hours, with the last three hours representing the effort of undoing all the changes applied so far. (See the "Instead of Waiting" sidebar for information on how to determine whether a data manipulation language [DML] statement has commenced a rollback before it completes.)

An examination of some of the data in the STAGING_TABLE reveals the conversion issue. A column named CREATED_DATE contains string data, some of which cannot be correctly converted to a date in the equivalent ANNUAL_SALES table.

```
SQL> select CREATED_DATE
  2  from   STAGING_SALES;

CREATED_DATE
____________________

01-FEB-2016
12-MAR-2012
54-AUG-2013
09-SEP-2014
23-OCT-2012
...
```

I am trivializing the true troubleshooting effort here. Analyzing source data for correctness, especially if the source data is millions of rows over dozens of columns, is an arduous task.

HISTORICAL SOLUTIONS

In the past, intercepting an error in data conversion typically required a PL/SQL function to act as a wrapper around standard facilities. In the insert-into-ANNUAL_SALES example, to check for a valid date in the CREATED_DATE column and prevent the statement from failing, I first create a date_checker PL/SQL wrapper function:

```
SQL> create or replace
  2  function date_checker(p_str varchar2) return date is
  3    l_dte date;
  4  begin
  5    l_dte := to_date(p_str,'dd-mon-yyyy');
  6    return l_dte;
  7  exception
  8    when others then return null;
  9  end;
 10  /

Function created.
```

The date_checker function returns the source data as a DATE datatype if the conversion can be performed and returns NULL otherwise. In Oracle Database 12*c* Release 1, the PL/SQL code can be folded directly into the SQL statement itself to

avoid cluttering the data dictionary. For more information on using PL/SQL functions within a WITH clause, refer to the documentation.

```
SQL> insert /*+ with_plsql */ into ANNUAL_SALES
  2  with
  3    function date_checker(p_str varchar2) return date is
  4      dte date;
  5    begin
  6      dte := to_date(p_str,'dd-mon-yyyy');
  7      return dte;
  8    exception
  9      when others then return null;
 10    end;
 11  select date_checker(created_date) valid_date, ...
 12  from   staging_sales;
```

Although this solves the data conversion problem, there is increased complexity in the SQL code as well as the performance overhead of calling a PL/SQL function potentially millions of times.

MORE VALIDATION CONTROL WITH ORACLE DATABASE 12*c* RELEASE 2

Oracle Database 12*c* Release 2 adds attempt-to-convert-and-catch-errors functionality natively to the database via the new VALIDATE_CONVERSION function and the existing CAST and TO_*datatype* suite of conversion functions.

Returning to the data loading example, here's how the new VALIDATE_CONVERSION can be used in the INSERT statement:

```
SQL> insert into ANNUAL_SALES
  2  select to_date(created_date, 'dd-mon-yyyy'), ...
  3  from   STAGING_SALES
  4  where  validate_conversion(
  5           created_date as date, 'dd-mon-yyyy'
  6         ) = 1;
```

Instead of returning an error because of string data that could not be converted to a DATE datatype, the new VALIDATE_CONVERSION predicate in the WHERE clause picks up data only where a conversion of the CREATED_DATE column to a DATE datatype with the supplied format ‘dd-mon-yyyy’ mask is successful. Success is indicated by a return value of 1. (If the conversion would not have succeded, the function returns 0.) Because only the rows that *could* be converted are returned, I can now apply the TO_DATE function in the SELECT portion of the INSERT with the assurance that it will *not* cause the statement to fail.

Using VALIDATE_CONVERSION as a predicate ensures that the statement will not crash, but it also keeps complete rows from the source data from being loaded into the target table. What about other requirements for row handling and values? What if you must replace erroneous data with a default value but retain the row, so that before and after row counts are consistent in the load process?

To address that requirement, the TO_*datatype* conversion functions have been extended in Oracle Database 12*c* Release 2 to optionally return a default value if the data conversion fails.

The SALES_AMT column in the STAGING_SALES table also contains string data, but that data should be loaded as number values into the ANNUAL_SALES table. A

sample of the SALES_AMT data shows that one of the rows has an erroneous comma, which would typically cause an error for the standard TO_NUMBER function call.

```
SQL> select SALES_AMT,
  2  from   STAGING_SALES;

SALES_AMT
_________

120000
172125
128000
125,000
99500
...
```

By using the new extended syntax for TO_NUMBER, however, you can nominate a default value for use whenever the TO_NUMBER function fails:

```
SQL> select SALES_AMT,
  2         TO_NUMBER(SALES_AMT
  3                   DEFAULT -1 ON CONVERSION ERROR) conv_sale
  4  from   STAGING_SALES;

SALES_AMT  CONV_SALE
_________ __________

120000         120000
```

```
172125          172125
128000          128000
125,000             -1
99500            99500
...
```

The other TO_*datatype* functions support the same extended functionality. Similarly, the CAST function supports the same extension for casting from one datatype to another.

SUMMARY

With Oracle Database 12*c* Release 2, extensions to data conversion functions and the new VALIDATE_CONVERSION function make data validation via SQL a breeze for database developers. SQL statements can self-validate data without the need for additional PL/SQL wrappers or nondatabase code to guard against conversion errors.

BONUS CONTENT

Connor McDonald is an Oracle Developer Advocate for SQL. His passions are database design, SQL, and PL/SQL. He can answer your database questions on Ask Tom.

PHOTOGRAPHY BY
SABINE ALBERS/THE VERBATIM AGENCY

NEXT STEPS

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

DOWNLOAD Oracle Database 12*c* Release 2.

INSTEAD OF WAITING

If you are running a large DML statement (INSERT, UPDATE, DELETE) and it fails during execution, the statement must roll back (or undo) all the changes completed from the commencement of the statement up to the point of failure. Hence a statement that gives the appearance of still executing may in fact already have failed, with control not yet returned to the requesting program or user because the undo phase is still in progress. However, you can detect whether a statement has already failed by examining the V$TRANSACTION view. The USED_UREC column in that view provides the number of undo records consumed by this transaction. Note that this is *not* necessarily going to be the same as the number of *rows* updated, because a single row change might change several indexes, large object (LOB) contents, and so on, each of which will have its own undo information. However, you can observe the direction of *change* in this column to determine the state of a transaction. For example, in one session, I create a table with approximately 100,000 records and then delete them.

Session 1

```
SQL> create table t as select * from dba_objects;

Table created.

SQL> delete from t;

104074 rows deleted.

SQL> select sys_context('USERENV','SID') from dual;
```

```
SYS_CONTEXT('USERENV','SID')
____________________________________________________

590
```

In this example, the delete has already completed, but even if it were still executing, I could monitor the progress of the active transaction from another session by joining V$SESSION to V$TRANSACTION.

Session 2

```
SQL> select t.used_urec
  2  from   v$transaction t,
  3         v$session s
  4  where  s.sid  = 590   -- from above
  5  and    t.addr = s.taddr;

 USED_UREC
__________

    104074
```

If that DELETE statement were still executing and then failed, the changes would commence rolling back. I can simulate that by issuing an explicit rollback command and rerunning my transaction monitor query while the rollback takes place.

Session 1

```
SQL> rollback;
```

```
[executing...]
```

Session 2

Running the query against V$TRANSACTION a few times in quick succession shows the progress of the rollback operation in Session 1.

```
SQL> select t.used_urec
  2  from   v$transaction t,
  3         v$session s
  4  where  s.sid = 590
  5  and    t.addr = s.taddr;

 USED_UREC
----------
     94872
SQL> /

 USED_UREC
----------
     32897

SQL> /

no rows selected
```

The last result—"no rows selected"—indicates that the rollback is complete, and that there is no longer an active transaction in Session 1.

By Melanie Caffrey

ORACLE DATABASE

Rapid Retrieval of Rows in Small Data Sets

Part 8 in a second series on the basics of the relational database and SQL

This article is the eighth 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, "Sequential Additions and Different Points of View," illustrated more about data definition language (DDL), including how to drop and rename tables and how to recover tables by using the recycle bin. You also learned how purging syntax affects table recovery and found out about the difference between truncating and dropping a table. You discovered how to use virtual columns to help simplify query writing. You also discovered how sequences and IDENTITY columns can be used for generating surrogate key values. Finally, you were introduced to views and how they can assist with query writing and data hiding.

In this article, you'll learn the following about indexes:

- How to create a balanced tree (B-tree) index
- The difference between individual indexes and composite indexes
- The relationship between indexes and constraints
- The relationship between indexes and NULL values
- How to create a function-based index

You'll also be introduced to altering and dropping indexes. Last, you'll discover how you can employ various individually tailored indexing strategies for your different application needs.

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 1 (12.1.0.2.0). 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 schemas used for this article's examples. (View the script in a text editor for execution instructions.)

BOOSTING YOUR PERFORMANCE

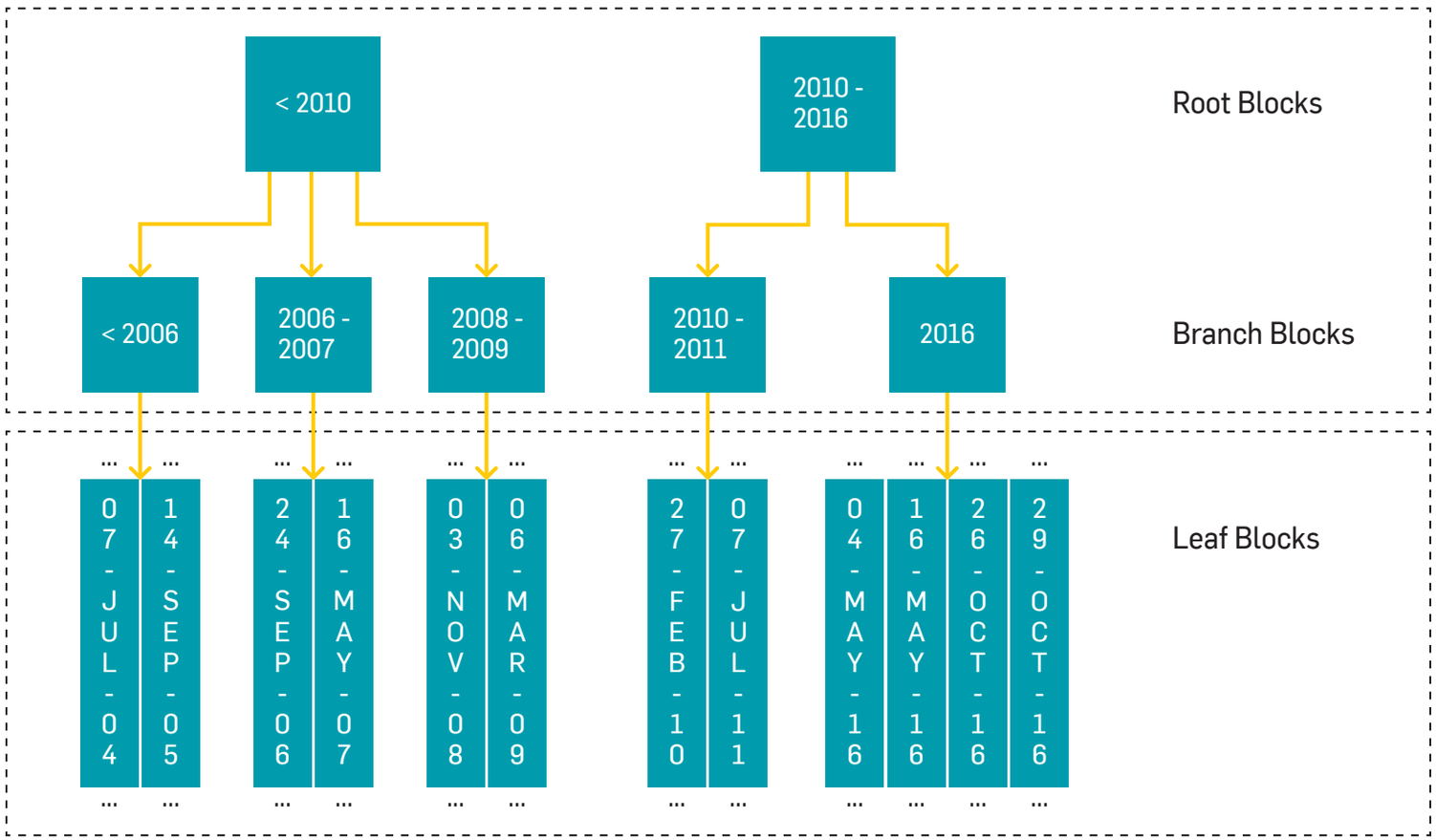
As your tables grow with each additional row inserted into them, your users' ability to quickly retrieve information can degrade, particularly if users perform changes (inserts, updates, deletes) on your tables frequently, as often happens with highly transactional tables. To help speed up data retrieval, Oracle Database sometimes uses *indexes*. Oracle indexes work in a manner similar to the index you might find in the back of a book. Just as you use a book index to determine the fastest way to obtain a particular piece of information, you can employ an Oracle index to help your users quickly obtain the relevant row(s) containing their desired query results. If an index that can assist with quickly selecting requested rows does not exist on a table, Oracle Database must examine every row to determine its eligibility for inclusion in the returned result set.

The act of examining every row is called a *full table scan*. If your query warrants the return of many row values from your table, it might be more efficient to scan the entire table to return the row values than to look up and obtain row values from an index. For each change you issue against a table that changes a value in an indexed column, the corresponding index must also be changed. Therefore, your indexing strategies will require some planning and analysis, because it can take a significant amount of time and storage space to build and maintain each index.

The Oracle Optimizer feature of Oracle Database uses statistics to determine whether to use a full table scan or an index for queries where an index is available. The next article in this series will introduce you to the Oracle Optimizer and statistics, but these two topics are beyond the scope of the current article.

STRIKING A BALANCE

The most common type of Oracle index is the *balanced tree*, or *B-tree*, index. It is often used to increase performance in situations where there are many distinct values in a single column or set of columns. Columns that have many distinct values are referred to as having *high selectivity*. Columns that contain primary key values and dates are examples of columns that have many distinct values and have, therefore, high selectivity. You can use B-tree indexes when you want to perform an exact match to retrieve a particular row value or when you want to perform a *range search* (for example, you'd like to retrieve all employees hired between 01-JANUARY-2010 and 15-JULY-2011). The structure of a B-tree index looks similar to **Figure 1**.



The ellipsis (...) above and below each leaf block indicates that there could be more than one value in each leaf block.

Figure 1: Structure of a B-tree index

The structure looks like an inverted tree and consists of two types of blocks: *root/branch blocks* and *leaf blocks*. *Root* and/or *branch* blocks (parent blocks) store the index key (such as a range of date values), along with a pointer to a *leaf* block (child block) containing the key value (such as a date value) along with the *ROWID* for any row that matches the key value. Every row has a ROWID. A ROWID is a unique address to a particular row that indicates the row's physical storage location in a table. A ROWID is a pseudocolumn—much like the

ROWNUM or SYSDATE pseudocolumns—in that it is not an actual column but can be referenced like one.

The query in **Listing 1** illustrates ROWID values returned in a query from the EMPLOYEE table. Note how each ROWID value returned in the result set is distinct from every other ROWID value. A ROWID is always unique, and using it is the fastest way to access a row. You can also update or delete a row by using the row's ROWID in the WHERE clause, thereby avoiding having to make Oracle Database search an index or perform a full table scan.

Code Listing 1: Selecting ROWID values from the EMPLOYEE table

```
SQL> set lines 10000
SQL> select ROWID, first_name||' '|| last_name employee_name
  2    from employee
  3  order by last_name, first_name;

ROWID              EMPLOYEE_NAME
__________________ ____________________________________________________

AAAXFUAAJAAAEB9AAK Lori Dovichi
AAAXFUAAJAAAEB9AAA Emily Eckhardt
AAAXFUAAJAAAEB9AAE Roger Friedli
AAAXFUAAJAAAEB9AAF Betsy James
AAAXFUAAJAAAEB9AAI Thomas Jeffrey
AAAXFUAAJAAAEB8AAB Sasha Meyer
AAAXFUAAJAAAEB9AAD Matthew Michaels
AAAXFUAAJAAAEB9AAC Donald Newton
```

```
AAAXFUAAJAAAEB9AAB Frances Newton
AAAXFUAAJAAAEB9AAN Don Rose
AAAXFUAAJAAAEB8AAC Gerald Sowell

ROWID              EMPLOYEE_NAME
__________________ ____________________________________________________

AAAXFUAAJAAAEB9AAL Mary Streicher
AAAXFUAAJAAAEB8AAA Marcy Tamra
AAAXFUAAJAAAEB9AAJ Theresa Wong
AAAXFUAAJAAAEB9AAH Mark Leblanc
AAAXFUAAJAAAEB9AAG Michael Peterson

16 rows selected.
```

Using **Figure 1** as an example, suppose you want to query the EMPLOYEE table for those employees hired between 01-JANUARY-2010 and 15-JULY-2011 and you have created a B-tree index on the HIRE_DATE column. Your search through the B-tree index would begin with the root block. A WHERE clause similar to

```
WHERE HIRE_DATE >= TO_DATE('01-JAN-2010', 'DD-MON-YYYY')
  AND HIRE_DATE <  TO_DATE('16-JULY-2011', 'DD-MON-YYYY')
```

would yield search input values that would be compared with the rightmost root block, which has pointers to 2010–2016. The next step would be to search the leftmost branch block, which has pointers to 2010–2011. Finally, the leaf blocks

would be scanned to find the exact HIRE_DATE index values—and their associated ROWIDs—that satisfy the search criteria. Leaf blocks also contain links to the next and previous leaf blocks, which enable the index to be scanned for ranges. (Note that the WHERE clause in this example searches inclusively for all dates between 12:00 a.m., January 1, 2010, and 11:59 p.m., July 15, 2011.)

PLANTING TREES

The following SQL

```
SQL> CREATE INDEX emp_hire_date_i
  2  ON employee(hire_date);
Index created.
```

illustrates the simplest method for creating a B-tree index on the HIRE_DATE column of the EMPLOYEE table. A query to find every employee hired on, for example, May 4, 2016, can take advantage of the EMP_HIRE_DATE_I index, because Oracle Database can look up the value in the index.

The query in **Listing 2** shows how many employees were hired on May 4, 2016. Only two employees, Sasha Meyer and Marcy Tamra, are included in the returned result set. With so few rows returned, using an index to retrieve the relevant rows is faster than reading every row in the table, especially if the table has many rows.

Code Listing 2: Using a B-tree index to speed up the retrieval of a few rows

```
SQL> select first_name||' '||last_name employee_name
  2    from employee
```

```
  3   where hire_date >= to_date('04-MAY-2016', 'DD-MON-YYYY')
  4     and hire_date <  to_date('05-MAY-2016', 'DD-MON-YYYY')
  5  order by last_name;

EMPLOYEE
--------------------------------------------------------------------------------
Sasha Meyer
Marcy Tamra

2 rows selected.
```

It may sometimes be useful to build indexes using multiple columns, particularly on individual columns with low selectivity. An index composed of multiple columns is referred to as a *composite index* or a *concatenated index*. The combination of one or more columns may make an index more selective than it would otherwise be with single-column B-tree indexes on individual columns.

The following SQL

```
SQL> CREATE INDEX dep_wage_increase_i
  2  ON employee (department_id, wage_increase_worthiness);

Index created.
```

demonstrates a simple method for creating a composite B-tree index on the DEPARTMENT_ID and WAGE_INCREASE_WORTHINESS columns of the EMPLOYEE

table. It also illustrates how virtual columns, such as WAGE_INCREASE_WORTHINESS, can be used in indexes.

And **Listing 3** shows a query that selects all employees who work in the accounting department (DEPARTMENT_ID = 10) and are eligible for a cost-of-living pay raise (WAGE_INCREASE_WORTHINESS = 'Cost of Living Increase Eligible'). Columns used together frequently in a WHERE clause, combined with the AND operator, are often good candidates for use in a composite index.

Code Listing 3: Using the DEP_WAGE_INCREASE_I index to retrieve rows

```
SQL> select emp_full_name employee_name
  2    from employee
  3   where department_id = 10
  4     and wage_increase_worthiness = 'Cost of Living Increase Eligible'
  5  order by last_name;

EMPLOYEE
_____________________________________________________________________________

Roger Friedli
Betsy James
Matthew Michaels
Frances Newton
Donald Newton

5 rows selected.
```

The order of the listed columns in a composite index can affect performance. When creating composite indexes, it is a good idea to list the column(s) you use with equality searches in the WHERE clause first and the column(s) you use with range searches (such as those that use the LIKE operator or > and < operators, for example) second. For instance, if you find that your application uses queries with a WHERE clause similar to the following:

```
WHERE salary    = 70000
  AND hire_date > TO_DATE('01-JAN-2010', 'DD-MON-YYYY')
```

then an index on EMPLOYEE (SALARY, HIRE_DATE) would be beneficial. If you tend to use equality searches on all columns, the order in which you list the columns in the composite index is not relevant, unless you plan to *compress* the index, in which case you should list the columns in the index in order of lowest to highest selectivity. The topic of index compression will be covered in subsequent articles in this series. Meanwhile, if you want to learn about index compression, refer to the index compression documentation.

A CONSTRAINED RELATIONSHIP

Recall that in the Beyond SQL 101 article "Defining, Constraining, and Manipulating Your Entities," you learned about different types of table and column constraints. Whenever you create a primary key constraint or a unique key constraint, Oracle Database creates a corresponding index automatically, unless an index for the column(s) of the constraint already exists. **Listing 4** demonstrates how adding a primary key constraint, EMPLOYEE_PK, to the EMPLOYEE table results in the automatic creation of the corresponding EMPLOYEE_PK index.

Code Listing 4: Adding a primary key constraint, which automatically creates an index

```
SQL> select index_name
  2    from user_indexes
  3   where index_name = 'EMPLOYEE_PK';

no rows selected

SQL> alter table employee
  2  add constraint employee_pk primary key (employee_id);

Table altered.

SQL> select index_name
  2    from user_indexes
  3   where index_name = 'EMPLOYEE_PK';

INDEX_NAME
------------------------------------------------------------------------------

EMPLOYEE_PK

1 row selected.

SQL> select column_name
  2    from user_ind_columns
  3   where index_name = 'EMPLOYEE_PK';
```

```
COLUMN_NAME
_______________________________________________________________________

EMPLOYEE_ID

1 row selected.
```

If columns upon which you have foreign key constraints defined are used frequently in WHERE clauses as *access methods*, they should be indexed. Access methods will be covered more fully in subsequent articles in this series. Meanwhile, to learn more about when it makes the most sense to index foreign-key-constrained columns, refer to the documentation for foreign key constraints; begin with the subsection named "Indexes and Foreign Keys." Additionally, if your database application code allows primary keys referenced by foreign keys to be updated or rows in the parent table that includes the primary keys to be deleted, the entire child table that includes the unindexed foreign key columns will be *locked* while such an update or delete action takes place. This means that any subsequent insertions, updates, or deletions on the child table will be prevented until the action being performed on the parent table is completed.

For this update/delete example, note that without foreign key indexes in place, for each row in the parent table that is modified and also referenced by foreign key values in the child table, a full table scan of the child table will be performed. Therefore, the update/delete action could take a very long time, due to one or more full table scans' being performed against the likely much larger child table. **Listing 5** illustrates an alternative query for obtaining the same information as that obtained in **Listing 3**. The main differences are that an index is created for the foreign-key-

constrained DEPARTMENT_ID column of the EMPLOYEE table and a subquery is performed against the DEPARTMENT table to obtain the DEPARTMENT_ID associated with the accounting department.

Code Listing 5: Using a foreign key index and a subquery to retrieve information from a parent table

```
SQL> create index emp_dept_fk
  2  on employee (department_id);

Index created.

SQL> select emp_full_name employee
  2   from employee
  3   where department_id = (select department_id
                             from department
                            where name = 'Accounting')
  4    and wage_increase_worthiness = 'Cost of Living Increase Eligible'
  5  order by last_name;

EMPLOYEE
_____________________________________________________________________

Roger Friedli
Betsy James
Matthew Michaels
Frances Newton
```

```
Donald Newton


5 rows selected.
```

WHEN YOUR WHERE WON'T WORK (AND OTHER CONSIDERATIONS)

There are several circumstances in which Oracle Database might not use an index, even though you've defined an index on the column used in your WHERE clause. Entirely NULL index key values are not stored in a B-tree index, but if *any* of the columns in a composite B-tree index has a non-NULL value, the index key will be stored. However, if all the index key columns for a given row are NULL, such an index key will not be stored. The following SQL

```
SQL> select emp_full_name employee_name
  2    from employee
  3   where hire_date IS NULL;

no rows selected
```

outlines a query on the EMPLOYEE table that will not use an index. Although there is an index, EMP_HIRE_DATE_I, defined on the HIRE_DATE column of the EMPLOYEE table, it does not include NULL HIRE_DATE values. Therefore, the query must perform a full table scan to determine that no rows in the EMPLOYEE table have a NULL value in the HIRE_DATE column.

Additionally, be aware that if you use a function on a column of a WHERE clause, you'll be altering it in such a way that an exact match value cannot be compared

with or retrieved from the index. The query in **Listing 6** illustrates the type of query that would not be able to use an index for function-changed WHERE clause columns. Although there is an index, DEP_WAGE_INCREASE_I, defined on both the DEPARTMENT_ID and WAGE_INCREASE_WORTHINESS columns of the EMPLOYEE table, only the EMP_DEP_FK index can be used, because the UPPER function is applied to the WAGE_INCREASE_WORTHINESS column in the WHERE clause.

Code Listing 6: Inability of WHERE clause columns with functions to make use of indexes

```
SQL> select emp_full_name employee_name
  2    from employee
  3   where department_id = (select department_id
                             from department
                             where name = 'Accounting')
  4     and UPPER(wage_increase_worthiness) = 'COST OF LIVING INCREASE ELIGIBLE'
  5  order by last_name;

EMPLOYEE_NAME
_____________________________________________________________________________

Roger Friedli
Betsy James
Matthew Michaels
Frances Newton
Donald Newton

5 rows selected.
```

If you plan to apply a function to a particular column of a WHERE clause regularly, you can create a *function-based index*, as demonstrated in

```
SQL> CREATE INDEX dep_name_upper_i
  2  ON department (UPPER(name));

Index created.
```

However, be aware that virtual columns, which are function-derived, are not allowed in functional index expressions. However, virtual columns can be B-tree-indexed.

The following SQL

```
SQL> ALTER INDEX dep_name_upper_i
  2  RENAME TO dep_upper_name_i;

Index altered.
```

demonstrates how you can rename an existing index by using the ALTER INDEX command. You can also use the ALTER INDEX command to change the storage characteristics of an index, *rebuild* it (a topic to be introduced in subsequent articles in this series), or mark the index as *visible* or *invisible*. If an index is marked as invisible, the program that determines the fastest way to retrieve data (the Oracle Optimizer, to be more fully covered in subsequent articles in this series) will ignore the index unless a special session setting is enabled. This feature might come in handy, for

example, when you are considering removing a possibly unused index but would like to mark it as invisible before doing so, to ensure that there is no negative impact on your application.

The following SQL

```
SQL> ALTER INDEX dep_upper_name_i INVISIBLE;

Index altered.
```

shows how you can mark an index as invisible. Once you've marked an index as invisible, it will not be used to access rows and you can determine whether application performance suffers or remains the same as before you marked the index as invisible.

Marking indexes as invisible before making your final decision to remove them can save you the work of having to re-create them if necessary. The following SQL

```
SQL> DROP INDEX dep_upper_name_i;

Index dropped.
```

demonstrates how to remove (drop) an index.

SHOULD YOU OR SHOULDN'T YOU?

It's important to look at the different ways Oracle Database can help you achieve good performance for both data retrieval and data changes. You should consider

indexing columns that are frequently used in SQL statement WHERE clauses and in foreign key column access methods. A B-tree index may be useful if

- Your column contains highly selective data, is accessed frequently, and is accessed by queries that typically return a few rows
- Your columns contain many NULL values and are accessed frequently by queries that typically return a few rows, and you are searching for non-NULL values

A B-tree index will most likely not be useful if

- A full table scan is determined by the Oracle Optimizer to be a faster method of access for row retrieval
- Your indexed column is included in a WHERE clause that applies a function to it (In such cases, a function-based index might be more suitable for your SQL statement needs.)

Be aware that adding indexes may increase the time required for INSERT, UPDATE, and DELETE operations to complete. However, if you query the table more than you change its data, the disadvantage of the additional time required may be offset by the advantage gained in increased query performance.

SUMMARY

This article introduced you to B-tree indexes, both single-column and composite. You learned how they can be used to increase performance when retrieving rows. You discovered how to create them and what kind of relationship they have with constraints. You also discovered how to alter them, how to drop them, and when they cannot be used. Last, you got a few guidelines for when B-tree indexes are most useful, when they are not useful, and when a function-based index can be helpful. The next article in this series will introduce you to bitmap indexes, statistics, and the cost-based Oracle Optimizer.

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

PHOTOGRAPHY BY **RAY NG**

NEXT STEPS

READ more Beyond SQL 101.

READ SQL 101, Parts 1–12.

LEARN more about relational database design and concepts.

DOWNLOAD the sample script for this article.