

# #PTIK

PASS THE KNOWLEDGE

— INDEPENDENT —  
**UK OUG**  
UK ORACLE USER GROUP

[WWW.UKOUG.ORG](http://WWW.UKOUG.ORG)

SUMMER 2020 | ISSUE 73

**FORMERLY ORACLE SCENE**

AN INDEPENDENT  
PUBLICATION NOT  
AFFILIATED WITH  
THE ORACLE  
CORPORATION

+

## FLASHBACK REVISITED

Connor McDonald shows  
how to use it for faster  
application development

**TECH  
EDITION**

TURN OVER FOR  
BUSINESS  
APPS

The future of decision-making

with Oracle Blockchain

How a virtual council used Hyperledger – and a range  
of other technologies – to enhance democracy

+

## EXECUTION PLANS

Jonathan Lewis  
on speeding up  
database queries

+

## NINE OF THE WORST!

The most common  
DBA mistakes – and  
how to avoid them



# #PTK Tech Edition

## IN THIS ISSUE

### 10 **COVER STORY**

#### Enhancing democracy with Oracle Blockchain

The route to transparent and secure decision-making in a virtual municipality in the Netherlands

### 16

#### A fresh look at Flashback

How it can become a useful tool for streamlining application development

### 20

#### The worst DBA practices and their technical debt

Nine bad habits to avoid if you don't want to store up trouble for later

### 24

#### Learning from execution plans

Solving the curious mystery of an extra-long run time

### 28

#### Connectivity agent installation on Linux

How to download and install the Oracle agent on a host machine



CHECK OUT

BUSINESS APPS!

## PLUS

### 05

#### Me & the UKOUG

ESB's Simon Holt on boosting your career by volunteering

### 07

#### News Etc

How the UKOUG is adapting, enter our Partner of the Year Awards, and more

### 09

#### UKOUG Events

Find out more about our virtual gatherings for 2020

In the Business Apps half of this issue, you'll find an in-depth look at how Oracle HCM can play a key role in employee experience (page 14), two expert views on cloud security (page 30), plus lots more knowledge-sharing from the world of business.

## #PTKOnline

View the latest issue online and access the archive of #PTK and Oracle Scene editions here:  
[ukoug.org/ptk](https://ukoug.org/ptk)

# Welcome to the new issue of #PTK Pass The Knowledge



A lot has changed since we published the previous issue of #PTK at the end of last year, when we had no inkling of how the Covid-19 pandemic would transform almost everything in business and society alike – and is set to overshadow our lives for some time to come. One thing that became apparent at a very early stage of the crisis is the essential role played by digital technology in helping to keep

our organisations running, not to mention the way it will support our societies in the future. A glimpse at how that future might look is the virtual municipality of Vaardam, in the Netherlands, which is using **Oracle Blockchain** and a range of other technologies to enhance democratic decision-making. It doesn't take much of a leap of the imagination to see how this project can be extrapolated to all manner of scenarios in a wide range of organisations and communities around the world.

It's also clear that, because well-functioning IT is now more critical than ever for many organisations, it has never been a more important time to ensure you're getting the most out of your Oracle investments. One great way to do that is to avoid falling into common traps that are likely to create trouble for you and your colleagues in the future – or a “**technical debt**”, as it is termed in our article on page 20. Here, our writer lists nine of the most common bad habits he's observed among DBAs and other IT professionals, and how best to avoid them.

Elsewhere in this issue, we're fortunate to be able to share in the expertise of Oracle gurus Connor McDonald who (in the start of a new series) takes a fresh look at **Flashback**, and Jonathan Lewis who solves a mystery relating to **execution plans**.

Please also take a moment to look at our Business Apps issue. There's plenty there to grab your attention, such as our cover story which examines the key role **Oracle HCM** can play in helping to get the very best out of every organisation's most valuable resource – its employees (page 14).

As ever, this magazine relies on your valuable feedback, ideas and contributions to ensure it's the best it can be – so please do send your thoughts to [editor@ukoug.org](mailto:editor@ukoug.org).

Stay safe



**James Lawrence**  
Editor

#### About the editor

James Lawrence is a professional multi-media journalist and editor who has been covering business and technology for more than a decade.

#### #PTK: PASS THE KNOWLEDGE EDITORIAL TEAM

##### Editor

James Lawrence  
[james.lawrence@topdogcomms.co.uk](mailto:james.lawrence@topdogcomms.co.uk)

##### Art Director

Peter Allen  
[peter@peterallendesign.co.uk](mailto:peter@peterallendesign.co.uk)

##### Consulting Editor (Technology)

Martin Widlake

##### UKOUG contact

Deanna Bates  
[deanna.bates@ukoug.org](mailto:deanna.bates@ukoug.org)

##### Head of Programme Management

Anna Crellin  
[anna@ukoug.org](mailto:anna@ukoug.org)

##### UKOUG governance

A full listing of Board members, along with details of how the user group is governed, can be found at: [www.ukoug.org/about-us/governance](http://www.ukoug.org/about-us/governance)

##### UKOUG office

19-23 High St,  
Kingston upon Thames,  
London KT1 1LL  
Tel: +44 (0)20 8545 9670  
[info@ukoug.org](mailto:info@ukoug.org)  
[www.ukoug.org](http://www.ukoug.org)

##### Produced by Top Dog Communications

Tel: +44 (0)7913 045917  
[james.lawrence@topdogcomms.co.uk](mailto:james.lawrence@topdogcomms.co.uk)  
[www.topdogcomms.co.uk](http://www.topdogcomms.co.uk)

##### Designed by Peter Allen Design

[peter@peterallendesign.co.uk](mailto:peter@peterallendesign.co.uk)  
[www.peterallendesign.co.uk](http://www.peterallendesign.co.uk)

#PTK: Pass The Knowledge © UK Oracle User Group Ltd  
The views stated in #PTK are the views of the author and not those of the UK Oracle User Group Ltd. We do not make any warranty for the accuracy of any published information and the UK Oracle User Group will assume no responsibility or liability regarding the use of such information. All articles are published on the understanding that copyright remains with the individual authors. The UK Oracle User Group reserves the right, however, to reproduce an article, in whole or in part, in any other user group publication. The reproduction of this publication by any third party, in whole or in part, is strictly prohibited without the express written consent of the UK Oracle User Group.  
Oracle is a registered trademark of Oracle Corporation and/or its affiliates, used under licence. This publication is an independent publication, not affiliated or otherwise associated with Oracle Corporation. The opinions, statements, positions and views stated herein are those of the author(s) or publisher and are not intended to be the opinions, statements, positions, or views of Oracle Corporation.



# ME & the UKOUG

## Simon Holt from ESB on how he's benefitted from UKOUG membership and why volunteering can give your career a boost

**The UKOUG is a vast repository of knowledge** from people who are actually doing the job. Rather than getting the shiny, happy story that you get from Oracle about how it's so easy to implement feature X – where they say, “You just press a button and it works,” and you know very well that's not the case – you can talk to people who've actually gone through the pain of implementing a feature and said, “Yes, this is good,” or, “No, it isn't quite all it's cracked up to be.” When I first joined about 25 years ago, I immediately realised the value in this.

**Membership is about being able to validate your own views.** You might have an idea about how to go about doing something, but for whatever reason you might not have people around you that you can talk it through with. So being able to go to a conference and see others talking about that very thing is useful.

**There's a lot of value in connecting** with people in a peer environment. You're meeting people who you may come into contact with further down the line in your career – which has been very helpful to me over the years.

“We want to know what people have got to say and how they think we can improve things”



**I've found myself volunteering as the lead for the Irish conference** over the last few years. We've built that up from quite small beginnings to the point where we've got global recognition, and we've got people coming from all over the world to come and talk for us. It takes up most of my volunteering time and it's hugely enjoyable. It was a great shame when we had to cancel this year because of the Covid-19 pandemic, but it was the right thing to do in every respect.

**By volunteering, you can add your voice** to what's happening. So if you come along to an event and think, “I'd have liked to have seen some material on topic X,” or, “Why is the format at an event the way it is,” or anything else like that, you're able to influence it – and, indeed, I have done. I've instituted quite a few changes to the way the Irish conference works, for example. That's very satisfying.

**If you're unsure about whether to volunteer or not...** basically, just do it. There's nothing that's holding you back.

**We're an inclusive bunch of people,** and we want volunteers. We want people to come along and challenge what we're doing. We want to know what people have got to say, how they think we can improve things. It doesn't matter what level, either. If you're thinking that you've got to spend a load of time doing it, you absolutely don't. You can do as little or as much as you wish. And it doesn't always mean being involved with committees and events – it can be something like contributing a short piece for the magazine!

**Doing all of this can also have a career benefit for yourself** in terms of your CV and your professional development – you shouldn't ever overlook that fact.

▶ *Simon Holt is Oracle Systems Architect at the Irish utility ESB, and also a member of the UKOUG's Tech Senior Advisory Team.*

To learn more about volunteering for the UKOUG, go to: [ukoug.org/volunteersarea](https://ukoug.org/volunteersarea)



Out-of-the-box reports, dashboards, and analytics for the entire organisation and all Oracle Applications

## splashEBS

powered by splashBI®

Oracle EBS Reporting & Analytics  
1,400+ reports across all EBS applications and modules

## splashDM

powered by splashBI®

Oracle Discoverer migration utility.  
Why rip and replace when you can migrate?

## splashOC

powered by splashBI®

500+ Pre-built Reports for Oracle Cloud HCM, ERP, & SCM, and Ad-hoc Reporting

## splashGL

powered by splashBI®

Financial Reporting in Excel for Oracle Cloud and Oracle EBS

## splashHR

powered by splashBI®

SplashHR People Analytics (500+ KPIs) help organisations attract, develop & retain top talent

## splashCRM

powered by splashBI®

CRM Reporting & Analytics  
Salesforce, Marketo, Google Analytics, Microsoft Dynamics, etc.

## Free Pandemic Management Solution\*

Keep your employees safe and maintain business continuity



Pre-defined content supporting Covid-19 analysis



Hierarchical employee risk analysis



Remote work analysis



Health risk profile



Daily refresh of employee data



Covid-19 exposure

Visit [splashbi.com/covid-19-pandemic-solution](https://splashbi.com/covid-19-pandemic-solution) to get started

\*Cloud software subscription at no cost during the pandemic. Discounted implementation (at cost)



## ENTER NOW FOR THE UKOUG PARTNER OF THE YEAR AWARDS

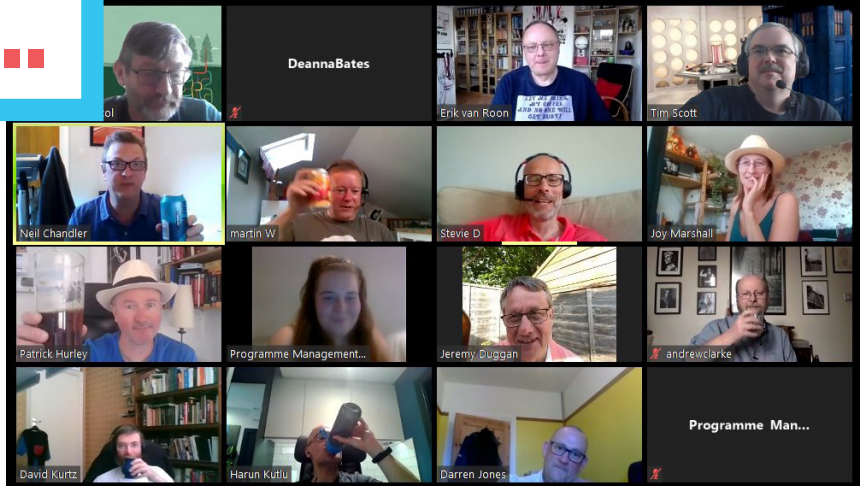
Nominations for one of our biggest events of the year are now open, so if you're an Oracle partner, make sure you get your entry in before our deadline of **26 June**.

We will be hosting the awards evening virtually, but that doesn't take anything away from its importance. Unique to these awards, the winners are decided exclusively by Oracle customers – which ensures they give a true reflection of the view of our community and recognise the genuine value that partners bring to end-users – and we'll still be celebrating in true UKOUG style with the winners this autumn.

Find out more about the awards, see the list of categories, apply to become an adjudicator, check out last year's winners, and submit your nominations here: [ukoug.org/pya](http://ukoug.org/pya)

## CHECK OUT BUSINESS APPS...

If you'd like to learn how Oracle HCM is helping organisations to deliver great employee experiences, take a look at the Business Apps half of this issue (page 14). You'll also find a range of other business-related topics that will help you get the most from your tech.



## Supporting our members through the pandemic

Like most other organisations, the UKOUG had to adapt the way it operates as soon as the UK's lockdown measures were announced.

The biggest change is that we have cancelled or postponed all of our physical events for the remainder of 2020. However, because supporting our members and helping them to get the best value from their Oracle investments continues to be paramount, we have been working hard to create a series of virtual events (see page 9 for more details) as well as uploading a wide range of virtual content onto our website.

Since making this shift, we have run

a Higher Education Forum virtually, our first multi-streamed event (Tech Summit, see below) and an evening of online networking with our President, Martin Widlake. Meanwhile, in our virtual content library, we are now hosting 700+ resources, including a wide range of videos, webinars and white papers, from both UKOUG members and Oracle itself.

Rest assured, we will continue to operate as an essential hub of information for all things Oracle – and we very much look forward to meeting you again in person as soon as we can.

▶ **SEE OUR VIRTUAL CONTENT LIBRARY:** [ukoug.org/virtualcontent](http://ukoug.org/virtualcontent)

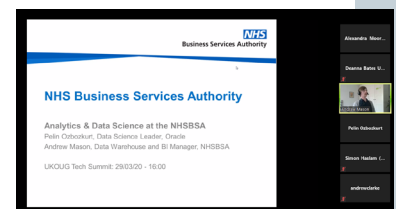
## Virtual Tech Summit was “brilliant”

The UKOUG's Tech Summit in April will go down in history as our **first ever multi-streamed event** – which meant that, as with our physical conferences, we hosted several presentations happening at the same time, so attendees could choose to view the ones most useful for them.

We received a lot of positive feedback, such as:  
 ▶ “Very good event – very well organised. I would be happy to attend other virtual conferences.”  
 ▶ “The virtual format was

a brilliant way to bring the Oracle community together for a day of learning which otherwise wouldn't have happened.”  
 ▶ “A great way to hear from experts including those who designed the software, and hear real experience stories, warts and all. You also get a chance to chat afterwards.”

The opening keynote was by Oracle's Andy Clark, who talked through Oracle's involvement in the World Bee Project, while two of the most popular streams were: Tim Hall on



The 7 Deadly Sins of SQL; and Andrew Mason (NHS) and Pelin Ozbozkurt (Oracle) on Analytics and Data Science in the NHS.

Members who weren't able to attend on the day can view all the sessions here: [ukoug.org/techsummitunoagenda](http://ukoug.org/techsummitunoagenda)



# 6 ways to get the most from your UKOUG membership

We've had to make a few changes in recent months, but the UKOUG is doing more than ever to help you make the most of your Oracle investments. Here are some good ways to ensure you're getting the best value from your membership fee:

Access our unrivalled virtual content, whether live or from our extensive library, from the largest independent Oracle user group in Europe



Add your colleagues to your membership so they too can network, share and learn



Volunteer and help us deliver useful content and influence other Oracle professionals



Save money with exclusive offers on training for members



Find an Oracle partner when you're in need of support, either from our Partner Directory, in person or online at our events



Sign up for our conferences – we're currently planning to run two of the largest European user group conferences in 2021, one for the Technology community and one for Business Applications



**TO FIND OUT MORE:** ▶ Visit our website which is full of information, ideas and knowledge-sharing from Oracle users like you: [www.ukoug.org](http://www.ukoug.org) ▶ Email us at: [membership@ukoug.org](mailto:membership@ukoug.org)

# UKOUG EVENTS

This year's schedule is looking very different to usual, due to the Covid-19 pandemic – but there will still be lots of great online events to help you get the most out of Oracle

**FIND  
OUT MORE**

For an up-to-date list of all the UKOUG's virtual events, and to book your place, visit our website: [ukoug.org/comingsoon](https://ukoug.org/comingsoon)



Because of the UK national lockdown, we have cancelled or postponed all our physical events for 2020. However, we're working harder than ever to ensure we can bring you the same broad range of Oracle expertise from the best and most experienced people in both our Business Apps and

Tech communities, as well as the opportunity to network and share knowledge with your peers.

We're rapidly reorganising everything, so are unable to publish a full schedule at the moment, but see below for one of the upcoming highlights. We will also be adding a

stream of **webinars** to the calendar, from both customers and partners, plus **virtual roundtables** for specific communities where participants will be able network and discuss particular topics.

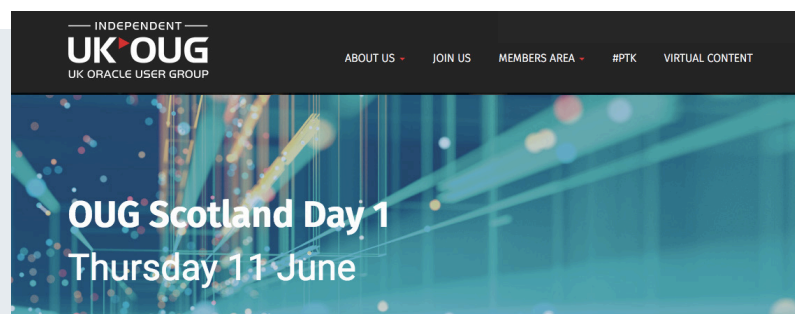
Keep checking our website for the latest info: [ukoug.org/comingsoon](https://ukoug.org/comingsoon)

## OUG Scotland 11 and 18 June

The agenda will be a mix of Technology and Business Applications topics across both days and all content will be relevant to Oracle users right across the UK and beyond.

### Highlights will include:

- ▶ Implementing Oracle Recruitment Cloud, with Patrick Haston from Scottish Natural Heritage
- ▶ Oracle Database 19c for Developers, with Chris Saxon from Oracle
- ▶ How we Developed Holly the Chatbot




at Hermes Parcelnet, with David Callaghan from Hermes

- ▶ Building a Highly Available and Scalable Logistics Platform with Oracle 19c and Goldengate 19c, with Nikitas Xenakis from The Co-op

There will also be an opportunity to join in our virtual networking drinks at the end of each day.

For more details of the agenda and to register, please go to: [ukoug.org/ougsotland2020](https://ukoug.org/ougsotland2020)



**Enhancing  
democracy  
with Oracle  
Blockchain**

Vaardam is the first virtual municipality in the Netherlands. Here's how it's exploiting a range of Oracle technologies to encourage citizen participation in government decision-making that is transparent, resistant to fraud and easy to use – and can be applied to many other business use cases ▶



### 10-SECOND SUMMARY

- ▶ To increase citizen participation, a group of scientists, businesses and non-profits created a virtual municipality that would be able to move fast and innovate without any of the traditional hurdles.
- ▶ A formal structure for decision-making was required – and they deployed blockchain technology on the Oracle platform to facilitate this.
- ▶ Various key design decisions were made along the way in order to ensure the decision-making could be from a trusted digital identity, anonymous and tamper-proof.

**T**raditionally, it has been hard for municipalities to reach out to their citizens. Participation is low and often the “usual suspects” show up for town hall meetings. Young people are hard to motivate, elderly people have mobility issues, people from different backgrounds might have language issues, and so on.

Another challenge for municipalities is that they are unable to innovate because of vendor lock-in, reluctance to be an early adopter, rules and regulations that are lagging and a lack of in-depth IT knowledge to manage complex new IT projects.

To tackle these issues, an initiative consisting of representatives from the science community, public sector and for-profit community worked together to create a virtual municipality that would be able to move fast and innovate without any of the traditional hurdles. Such an initiative needs several things in order to be successful:

- ▶ Citizens who can register as “a member or citizen of Vaardam” and try new things – a virtual municipality without citizens is meaningless
- ▶ A formal structure to make decisions
- ▶ Real software solutions that can be used by physical municipalities in the real world.

A formal structure to facilitate decision-making, including voting, needs to adhere to at least the following requirements:

- ▶ It should make use of a trusted digital identity
- ▶ Votes should be anonymous
- ▶ There should only be one vote per citizen
- ▶ It should be tamper-proof.

To facilitate the decision-making process and to offer a platform for “real organisations”, the initiative developed a solution that enables participation that is transparent, resistant to fraud and easy to use.

## DEVELOPING THE DECISION-MAKING PROCESS

The decision-making process that is supported involves five steps, as shown in Figure 1.

The first step is to decide what topics are important. For example, suppose a municipality wants to involve citizens in the decision to cut the budget. The first step is to ask for input. All eligible voters can enter one or more suggestions for items they feel should not be cut. You then group them,

to take out duplicates and merge similar suggestions until you end up with a list of about 15 to 20 topics. Then all eligible voters give three votes to the topics they feel are most important. This results in an order list, or agenda, of items that can be solved.

In step 2 a group (of citizens or professionals) will come up with a solution on how to cut the budget, by first investigating rules and regulations, conditions and stakeholders. Then a solution is defined in step 3. Experts can be called in and everyone can listen in on the conversation. This whole process can be followed from a mobile app. When a solution is ready, all eligible voters vote in favour or against the solution in step 4. When they vote against the solution, they must explain why. When 66% vote “Yes” the solution is accepted and will be executed in step 5. This is monitored to make sure the solution is realised according to the decision made.

This process can be applied to many different decision types and organisations: so not only what is important in the government of a country, but also to make decisions and involve employees, or decide on the agenda and topics in shareholder meetings, yearly union negotiations and many more.

## THE TECHNICAL SET-UP

The solution consists of the following logical components (see Figure 2).

- 1 Integration with itsme@ to verify the identity of the voter: this is based on OpenID Connect and is compliant with eIDAS level “high”, a European security standard
  - 2 Mobile apps based on Oracle JET to give access to the functionality on both iPhone and Android
  - 3 WordPress to publish content about the decision and the solutions
  - 4 Rocket.Chat to allow citizens and groups to video chat and comment on the process
  - 5 Oracle Blockchain Service to support voting
  - 6 Oracle Kubernetes Engine to support the business logic of voting, suggestions and results
  - 7 Oracle Autonomous DWH to offload the blockchain transactions
  - 8 Oracle Analytics Cloud to analyse trends in the results
- For Vaardam, the solution was integrated with Oracle Service Cloud as the source to check the eligibility of a citizen. Other organisations can do this based on a local membership database or on features that are shared with itsme@. No personal information about citizens is stored in the solution.

## WHAT IS BLOCKCHAIN?

A blockchain keeps track of transactions, in a so-called ledger. Data is immutable, and each transaction has a link to the previous transaction.

Before a transaction is put on the ledger, it is validated by peers, so there is no “man in the middle” and no single point of failure.

## BLOCKCHAIN DESIGN

To capture the votes, we used Oracle Blockchain. This is a service that is based on Hyperledger and has several advantages, as follows:



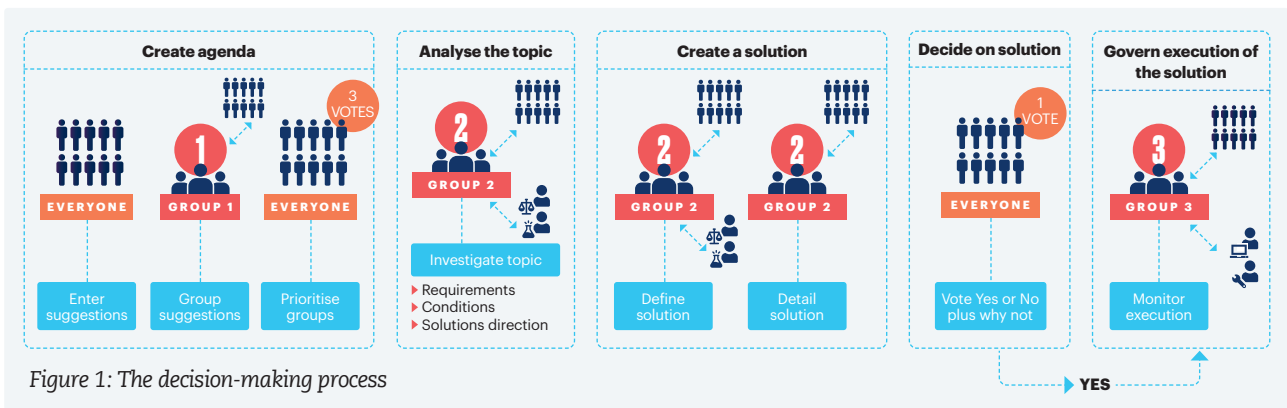


Figure 1: The decision-making process

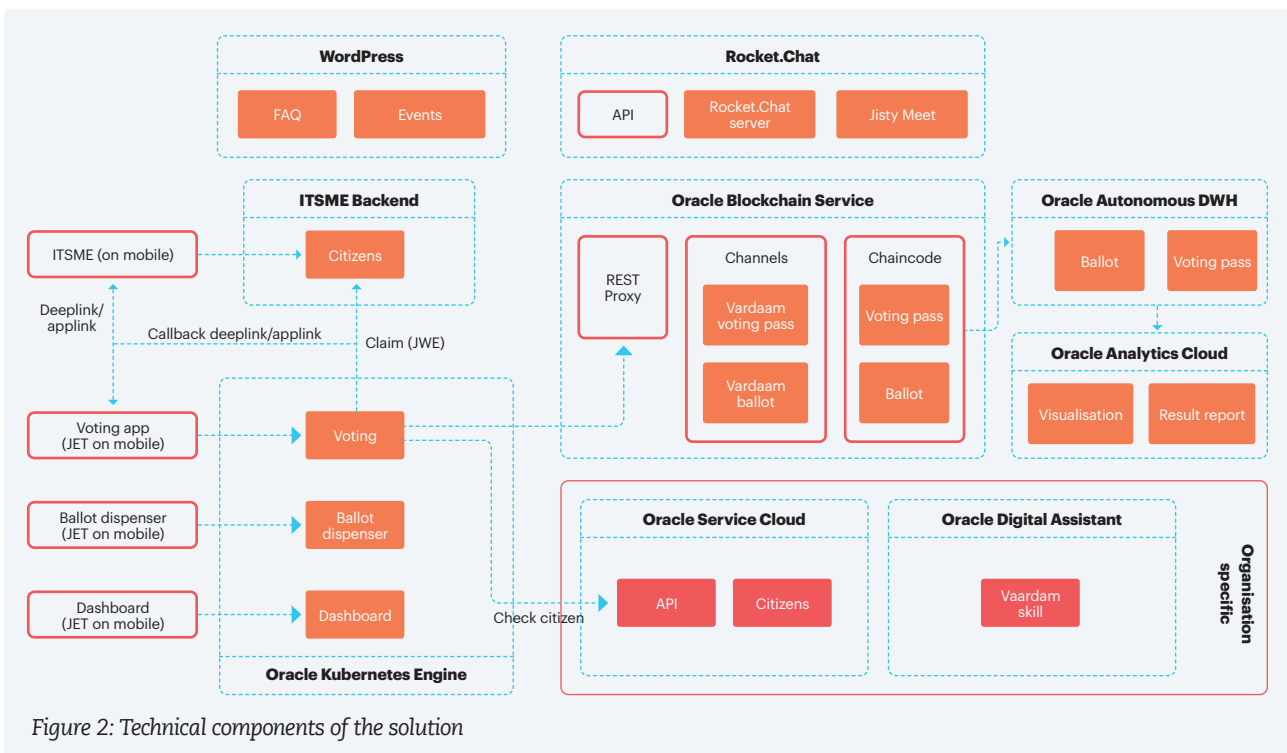


Figure 2: Technical components of the solution

- 1 It is a permissioned blockchain. This means it takes less processing power to validate new transactions and you can control access to the network.
- 2 It is available as open source and supported by multiple cloud vendors. This means participants can join the network with the provider of their choice: public cloud providers, or on-premise or private cloud solutions.
- 3 It supports multiple mainstream languages. This means that you can program your contract (chaincode) in the language of your choice. In our case that was Node.js.
- 4 Oracle has added features to make it simple to set up and communicate with a specific peer using REST APIs.

We designed the chaincode in such a way that there is strict separation between voting passes and ballots. Each eligible voter receives a voting pass. Each voter can cast a ballot, which is on a separate ledger.

## VOTING PASS DESIGN

Because voting needs to be anonymous, the voting pass chaincode is separate from the ballot chaincode. The

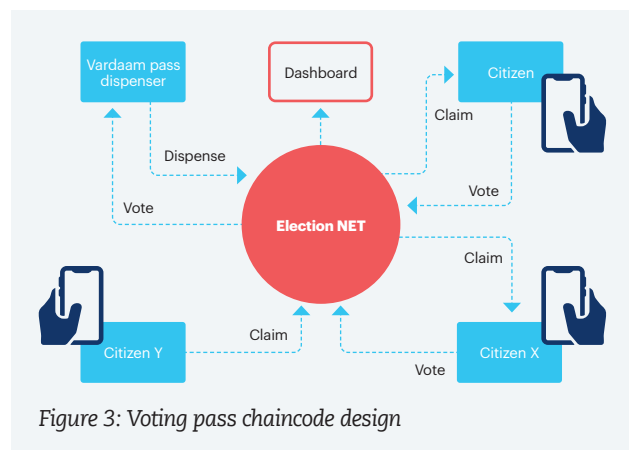


Figure 3: Voting pass chaincode design

voting transactions are shown in the design of the voting pass chaincode (see Figure 3).

Voting passes are dispensed (put on the blockchain) by the voting pass dispenser. Citizens can claim a voting pass (if they are eligible) and they can vote once. This ledger does not keep track of what the citizen voted for, only of the voting pass. ▶

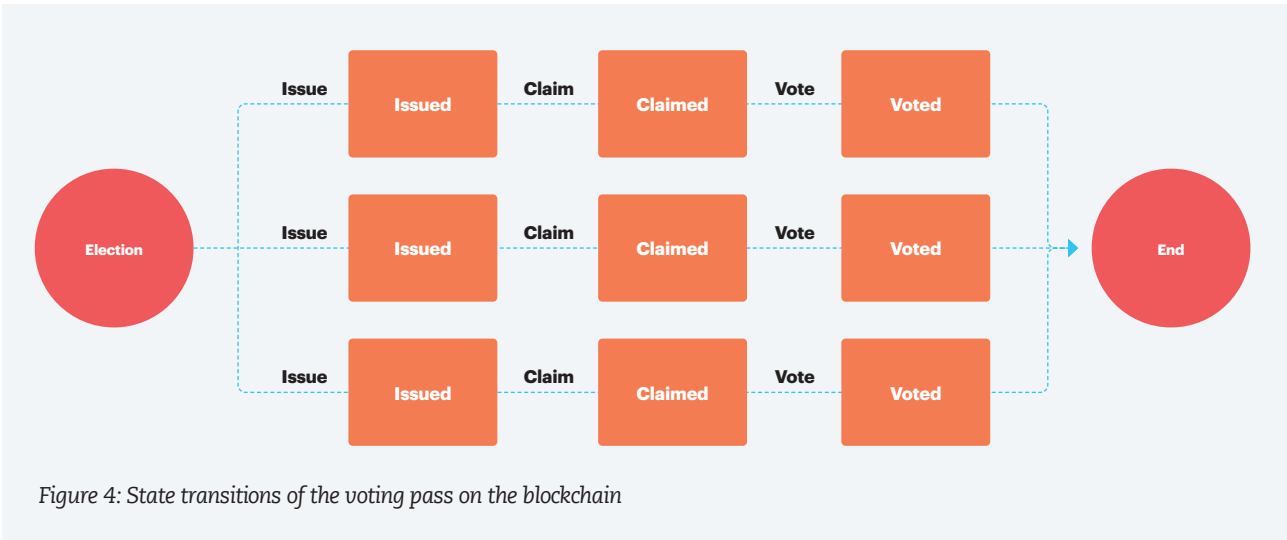


Figure 4: State transitions of the voting pass on the blockchain

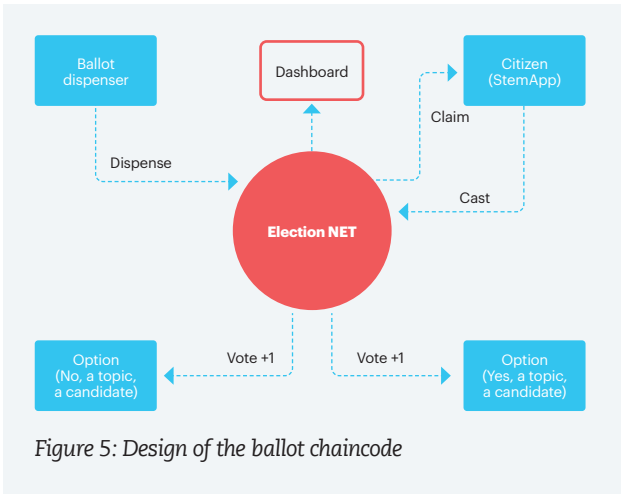


Figure 5: Design of the ballot chaincode

This means that in terms of transactions, the following states have been identified (see Figure 4).

- 1 Issued: A voting pass is issued. This is the empty starting state of a voting pass on the ledger.
- 2 Claimed: A citizen claimed the voting pass. This voting pass can now only be used by this citizen. The citizen is identified on the blockchain by a deterministic GUID that cannot be traced back to his or her identity.
- 3 Voted: The voting pass is used. This state can only be reached if the voting pass was claimed.

A vote can be for a topic (e.g. education), a solution (yes or no), or a candidate in the case of “traditional” elections

In the latest version we also added a state for “expired”, because voting passes are not valid indefinitely. We may also add a “delete” to be able to disqualify a voting pass. Each update, including delete, is validated and stored on the blockchain.

## BALLOT DESIGN

The design for the ballot looks similar, as you can see in Figure 5.

A vote can be for a topic (for example, education as a topic that can be voted for), a solution (yes or no), or a candidate in case of “traditional” elections of candidates for a governing body. The state transitions are similar as for the voting pass: issued, claimed, cast, expired, deleted (or invalidated).

## CONCLUSION AND NEXT STEPS

Integrating blockchain with microservices using APIs and Oracle Platform as a Service has created a very powerful framework. The blockchain technology facilitates the voting part of the decision process and ensures that votes can't be tampered with.

We have run several pilots in the Netherlands, both with municipalities and with a company. The next step will be to improve the user experience, test performance to ensure we can scale up, add functionality for sortition and to think about solutions to prevent family voting. ❌

## ABOUT THE AUTHOR

Lonneke Dikmans is a Partner and Global CTO at eProseed. Her specialities include architecture, integration and software development in the government, utilities and financial services sectors.



## MAKING THE KEY DESIGN DECISIONS

During this process, we had to make several important design decisions – some of these could be valid for other projects, while others might be revisited by us or are specific to the situation

### DECISION 1: Minimise code in the chaincode

**Description:** Chaincode is software that runs on the peer nodes and enables interaction on the network's shared ledger. We decided to minimise the business logic that is put in the chaincode.

**Rationale:** Chaincode is, like any other part of the blockchain, immutable. This means it can't be deleted. If there is an update, you must consider what to do with the transaction data already on the ledger. Do you want to start a new ledger? Keep the old values with the old version of the code, and new transactions that are interacted with using the new version? This will be hard to recognise when analysing the results. To avoid this problem, we decided to keep the chaincode as stable as possible: it only has a minimum of business logic, checking the state transition and required fields.

**Impact:** Because of this decision, business logic is created in a microservice for voting and ballots. These are easier to change than the chaincode and can be updated based on specific contexts. This backend must be secure and on the internal network of the platform to avoid man-in-the-middle attacks.

### DECISION 2: Define two independent chaincodes

**Description:** Each citizen receives a voting pass to make sure they can only cast their vote once (or select an option once). The ballot (the option or candidate the vote goes to)

is defined in a separate ledger and there are no links between the two.

**Rationale:** Voting should be anonymous, that is a ground rule in democracy. However, we do need to make sure there is no fraud by people voting more often than allowed. This calls for two different ledgers without any link between the transactions.

**Impact:** Since the system does not keep track of what people vote for, we can't show the individual vote after it has been cast in the app. To check if someone already voted, we need a mechanism that can identify a voting pass. We use a deterministic mechanism to generate a GUID. We also need a mechanism that can't be reversed, so the voting pass can't be parsed to determine the unique user.

### DECISION 3: Support multiple authentication and authorisation mechanisms

**Description:** In Vaardam we use itsme® as an authentication mechanism. We will support IDIN in the future and already support username/password for regular "inquiries" and two-factor authentication in cases where we want a higher security level.

**Rationale:** itsme® is a powerful mechanism, that gives full control to citizens over the data they exchange. However, there are two downsides:

► There are competing standards both from a public sector perspective and from a profit sector perspective. We want to be able to support different organisation types and they



have a free choice when it comes to authorisation and authentication means they want to acquire.

► Some inquiries are less sensitive and should have a low threshold to enter so username/password is fit for those occasions (inquiries for events, for example).

**Impact:** Serving participants with a list of decisions they are eligible to vote for depends on how they are logged in. This means we need to provide a way to search for elections and present an alternative login if the election requires this. It complicates both the user interface and the business logic.

### DECISION 4: Hyperledger as blockchain technology

**Description:** As a blockchain platform we picked Hyperledger, a permissioned blockchain technology.

**Rationale:** Hyperledger supports permissioned blockchain. This is cheaper and more energy-efficient than public blockchains. Validation of transactions

can be done by the organising entities, the voting entities, or the publishing entities can have different privileges. Hyperledger is supported by multiple cloud vendors and can run both in the cloud and on premise. It is also open source and supports multiple coding languages.

**Impact:** We need to define our chaincode and permissions.

### DECISION 5: OCI Kubernetes Engine as container platform

**Description:** As a platform for running our backend logic we decided to use Kubernetes Engine.

**Rationale:** There were several reasons why: it is supported on different clouds so there is no vendor lock-in, it is a widely used container engine, it is scalable, it is lightweight, you can test the code locally and then deploy easily using automated build jobs and CI tooling.

**Impact:** We containerised our code and wrote YAML for deployment.

# A fresh look at Flashback *Part 1*

Flashback isn't just for emergencies. In the first of a series, we take a look at how it can become a useful tool for streamlining modern application development processes

By Connor McDonald



## 10-SECOND SUMMARY

- ▶ Developers using iterative processes, DevOps, and automated testing and deployment can all benefit from having a deeper understanding of Flashback technologies and how to get the most out of them.
- ▶ There are six Flashback technologies inside the Oracle database. Here, we examine how Flashback Query can enhance the development process.
- ▶ Future articles will look at Flashback Table, Drop, Database, Transaction and Data Archive technologies.

Many readers of this article, I would be willing to wager, are aware of the Flashback technologies in the Oracle database but have never had a reason to use them. This is by no means a criticism of the Oracle customer base, but more a reflection on how the technologies related to Flashback are often interpreted, and also the purpose for which they were originally built.

When “Flashback” is mentioned, the vision that comes to mind is often the red panel of glass on the side of the wall emblazoned with “In case of emergency, break glass” or perhaps oxygen masks dropping from the ceiling of the aircraft. Flashback is seen as solely a technology related to rectifying accidents or catastrophic emergencies that have occurred in the database.

Perhaps this was true when Flashback was first built back in Oracle 9 nearly two decades ago. In those days, the development of applications was a fairly static and methodical process. You designed your application; then once that phase was complete you coded the application, tested it in isolation from everything else, and finally (after countless meetings and the appropriate management sign-offs!) your application would be deployed to production. Each phase was a one-time affair, and then your application would be doomed in a potentially endless support and maintenance phase.

But times have changed and the entire mechanism via which applications are developed has also changed. Nowadays, the norm is a much more iterative process, and those iterations are also moving towards automation. We now have automated unit tests, automated integration tests and even automated deployment under the banner of DevOps. The traditional phases of development now repeat many times with increasing rapidity, and many of these phases are now routinely performed in an unattended fashion.

For this reason, it’s time for the modern application developer to revisit Flashback. Under these new development regimes, Flashback actually becomes a very useful tool in streamlining modern development processes.

The term “Flashback” is actually a banner for six disparate technologies inside the Oracle database.

These are:

- ▶ Flashback Query
- ▶ Flashback Table
- ▶ Flashback Drop
- ▶ Flashback Database
- ▶ Flashback Transaction
- ▶ Flashback Data Archive

In this series of articles, I will cover each in turn, with a description of the functionality and then a view of how it can improve your current application development processes. So let’s begin with the first of the technologies, the Flashback Query function.

## WHAT IS FLASHBACK QUERY?

In what came as a revelation for some Oracle practitioners, every single person that has ever written a query against the Oracle database has implicitly been using the Flashback Query function. By way of revision, one of the fantastic things about databases that observe the principles of ACID is that you can abandon (or “undo”) an uncommitted transaction before it causes any damage to your database.

The mechanism via which the Oracle database can achieve this is beyond the scope of this article but, in a nutshell, it takes a record of your changes along with some instructions internally as to how to reverse those changes should you issue a rollback command. It stores these instructions in *undo segments*.

Decades ago, back in Oracle version 4, an epiphany occurred, in that this same undo information could be used to allow applications that were querying the database to get what is now known as a “consistent read”. Using the undo information from transactions conducted in the database, a running query will, on the fly, reverse out database changes to give results that are consistent to the point in time at which the query commenced. You may have heard the term “*readers do not block writers, and writers do not block readers*”.

Thus, every query in the Oracle database presents a view of data at a nominated point in time, namely the moment the query commenced running, no matter how long that query takes to execute. This is the essence of Flashback Query. Flashback Query extends this concept so that the nominated point in time can be any time of your choosing, not only the time the query commenced.

Figure 1 shows the syntax for a Flashback query. The geeks among us can nominate a SCN, that is, a “system change number” but most people will usually use the `TIMESTAMP` clause to get data as of a point in time.

```
SQL> select * from DEPT AS OF SCN 995401;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

```
SQL> select * from DEPT
2 AS OF TIMESTAMP systimestamp -
3 interval '20:00' minute to second;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Figure 1: The syntax for a Flashback query

A table to which a Flashback Query clause is being applied is no different to any other table in a query. For example, Figure 2 (on the next page) shows that you ▶



can join tables as of the current moment in time with those involved in a Flashback expression.

## FINDING A USE CASE

At this point, curious readers may be pondering: “But what is the usefulness of knowing what the DEPT table looked like 15 minutes ago? Where is the use case?”

Consider a common unit testing requirement. The *current* state of data in one or more tables is often captured, then a test is run to achieve a desired *final* state. In order to validate the unit test, we often need to compare the original state of the data with the final state of the data.

Without Flashback Query, this requires more code – either taking a copy of the original data, or spooling it out to a file, or some other bespoke mechanism so that it’s possible to perform a difference calculation after the unit test has run. With Flashback Query, this becomes trivial.

Figure 3 shows an example of the DEPT table in its final state being joined to itself as of five minutes ago, in this case most likely to be before the unit test was run. Using a FULL OUTER JOIN we can easily identify deletions, insertions and updates, to track what the unit test did. Flashback Query becomes a much simpler mechanism of doing before and after analysis.

## DATA ANALYSIS

Flashback Query also offers more in the way of data analysis for your unit test results. Revisiting Figure 3 for DEPTNO=10, the original state of DNAME was “ACCOUNTING” and the final state was “UNKNOWN”. Does this mean that the unit test updated the Department name from accounting to unknown? Perhaps... perhaps not. There could have been *many* intermediate updates to this row during the test – it may have been one update; it may have been 10.

Flashback Query also offers a capability known as ROW VERSIONS, where the transactional history of changes to the row can be interrogated. Figure 4 shows the VERSIONS BETWEEN syntax for Flashback Query. Note that this is only a *single row* in the DEPT table, but with five *versions*.

The five rows in the result set are the five iterations of the value of DNAME for the DEPTNO=10. Flashback row versions reveals that five distinct updates were performed to department 10 to get from the original state of “ACCOUNTING” through to the final state of “UNKNOWN”.

Additionally, various pseudo-functions are available for use within a Flashback row versions query, to further interrogate the information pertaining to each transaction that occurred on this particular row. Figure 5 shows the VERSIONS\_STARTTIME, the VERSIONS\_XIS, and the VERSIONS\_OPERATION pseudo-functions to

```
SQL> select e.empno, e.ename, d.dname
2 from emp e,
3 dept AS OF TIMESTAMP sysdate-1/24 d
3 where d.deptno = e.deptno;
```

EMPNO	ENAME	DNAME
7782	CLARK	"PREVIOUS NAME"
7839	KING	"PREVIOUS NAME"
7934	MILLER	"PREVIOUS NAME"
7566	JONES	RESEARCH
7902	FORD	RESEARCH
7876	ADAMS	RESEARCH
7369	SMITH	RESEARCH
7788	SCOTT	RESEARCH

Figure 2: Joining tables as of the current moment in time

```
SQL> select case
2 when d1.deptno is null then 'DELETE'
3 when d2.deptno is null then 'INSERT'
4 end action,
5 d1.deptno, d2.deptno, d2.dname, d1.dname
6 from DEPT d1 FULL OUTER JOIN
7 DEPT AS OF TIMESTAMP sysdate-3/864 d2
8 on d1.deptno = d2.deptno;
```

ACTION	DEPTNO	DEPTNO	DNAME	DNAME
	10	10	ACCOUNTING	UNKNOWN
	20	20	RESEARCH	RESEARCH
	30	30	SALES	SALES
DELETE		40		
INSERT	50		MARKETING	MARKETING

Figure 3: An example of a table in its final state being joined to itself as of five minutes ago

```
SQL> SELECT deptno, dname
2 FROM dept
3 VERSIONS BETWEEN
4 TIMESTAMP SYSTIMESTAMP -
5 INTERVAL '20:00' MINUTE TO SECOND
6 AND SYSTIMESTAMP
7 WHERE deptno = 10;
```

DEPTNO	DNAME
10	ACCOUNTING
10	MONEY GRABBERS
10	FINANCE
10	BEAN COUNTERS
10	UNKNOWN

Figure 4: The versions between syntax for Flashback Query

reveal the moment in time that the version of this particular row came into existence, and the operation (I=INSERT, U=UPDATE, D=DELETE) that caused this change. (I will return to the transaction ID (XID) in a future article.)

Figure 6 shows a variant of the row versions syntax in order to mine as much transactional history information

```
SQL> SELECT deptno, dname,
2     VERSIONS_STARTTIME
3     , VERSIONS_XID
4     , VERSIONS_OPERATION
5 FROM dept
6 VERSIONS BETWEEN TIMESTAMP
7     SYSTIMESTAMP - INTERVAL '20:00' MINUTE TO SECOND
8     AND SYSTIMESTAMP
9 WHERE deptno = 10;
```

DEPTNO	DNAME	VERSIONS_STARTTIME	VERSIONS_XID	V
10	UNKNOWN	16-SEP-19 11.53.45 PM	0200100060040000	U
10	MONEY GRABBERS	16-SEP-19 11.53.36 PM	0600050065040000	U
10	FINANCE	16-SEP-19 11.53.24 PM	09000D001D050000	U
10	BEAN COUNTERS	16-SEP-19 11.53.12 PM	01001A00EA030000	U
10	ACCOUNTING			

Figure 5: The moment in time that the version of each particular row came into existence

```
SQL> SELECT deptno, dname,
2     VERSIONS_STARTTIME
3     , VERSIONS_XID
4     , VERSIONS_OPERATION
5 FROM dept VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE;
```

DEPTNO	DNAME	VERSIONS_STARTTIME	VERSIONS_XID	V
50	UNKNOWN	16-SEP-19 11.08.15 PM	04000700EA030000	U
30	UNKNOWN	16-SEP-19 11.08.15 PM	04000700EA030000	U
20	NERDS	16-SEP-19 11.07.57 PM	090016001D050000	U
20	R&D	16-SEP-19 11.07.48 PM	05000B0074040000	U
...				

Figure 6: A variant of the row versions syntax

as possible, by ranging between a system change number of MINVALUE and MAXVALUE.

## FURTHER CAPABILITIES

Another nice capability of Flashback Query is that, if you already have a suite of SQL scripts that you use to interrogate tables impacted by unit or integration testing, you do not have to go into those scripts and manually edit every single SELECT statement to add an AS OF clause in order to activate a query. The nominated moment in time for which you wish to run your Flashback Queries can also be specified at the *session* level. The DBMS\_FLASHBACK package shown below allows enabling Flashback Query for an entire session, at which point all queries will run as of that nominated point in time.

Thus you could run a suite of existing queries as of the current moment in time, then set the nominated point in time to before the unit test was run, and then rerun the same suite of scripts to get before and after results.

```
SQL> exec dbms_flashback.enable_at_
time(systimestamp-1/24);
```

One note of caution. Remember that any database query, Flashback or otherwise, is in effect undoing any transactional changes that have occurred to the data after the nominated moment in time.

If you're performing a Flashback Query back over many minutes of time, and many changes have occurred to the data within that time span, then you are potentially undoing hundreds, thousands or even millions of transactions, in order to return the data consistent with a point in time in the past. A Flashback Query against a table that is aggressively being changed could be a very *expensive* query to run.

It is for this reason that the ability to run a Flashback Query is a privilege that must be granted via the Database Administrator. If you cannot run a Flashback Query, speak to your DBA to be granted the FLASHBACK privilege.

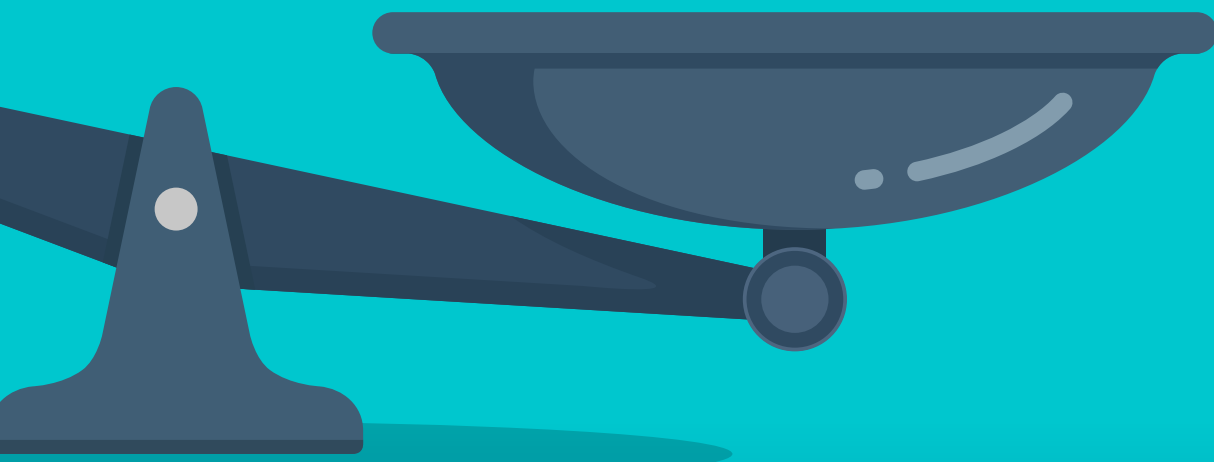
This covers just one of the six flashback technologies that have existed in the Oracle database since version 9. Hopefully you can now see the opportunities for using Flashback Query to complement your current unit testing strategies. In the next article I will look at more of the Flashback technologies and how they can streamline your application development processes. ☒



### ABOUT THE AUTHOR

Connor McDonald is a Database Advocate for Oracle. He loves talking tech and blogs at: [connor-mcdonald.com](http://connor-mcdonald.com) and [youtube.com/ConnorMcDonaldOracle](https://youtube.com/ConnorMcDonaldOracle)

# The worst DBA practices and their TECHNICAL DEBT



If you want to avoid storing up trouble for you and your colleagues in the future, here are nine common mistakes to steer clear of

By Franck Pachot



## 10-SECOND SUMMARY

- ▶ “Technical debt” is when an IT solution – often in software – is provided, but ends up causing more problems or complexity further down the line.
- ▶ One good way of avoiding these is to look at some of the worst IT practices to understand why they are likely to leave a technical debt.
- ▶ Such practices include: going for a quick fix rather than a better quality long-term solution; taking pride in large databases; poor naming conventions; and using too many one-off patches.
- ▶ Part 2 of this article, in the next issue of #PTK, will focus on the more technical errors that are frequently made.

I’m always cautious when being asked for “best practices” because it depends on the context. I can encourage some “default practices” for when you have no good reason to do differently – but people often ask for simple, generic rules that they can apply everywhere. However, for general recommendations, I prefer to list the things that should always be avoided, so I’ll cover off a few “worst practices”.

As a consultant, I have seen environments with a substantial “technical debt”. That is, decisions made in the past, due to a lack of knowledge or just by laziness, may be annoying in the present but also compromise future operations, making everything complex and difficult to maintain and evolve. This is not a comprehensive list but, for all of them, I’ve experienced the consequences.

As it’s so long, I’ve split the list into two – this one will focus more on “architectural” worst practices and the next one (to be published in the next issue of #PTK) will be more about the technical side of things.



## 1 JUDGING SUCCESS AS “IT WORKS” RATHER THAN WORK QUALITY

This one is the most important and therefore top of the list. You should always think about the future for yourself and your colleagues and try to build something that is sustainable.

As an example, a user asks for a schema copy with Data Pump and you do it quickly, happy to close the ticket. But did you consider that the same request will probably happen again soon? Take the time to document what you did (the `expdp/impdp` commands in this example) so that next time it will be an easier job. Documenting for colleagues is also like asking for a code review. The colleague may tell you that you forgot a `“flashback_time”` and that what you did in haste ends up inconsistent. And after a few occurrences where this documented procedure is followed, it will be just a small additional step to script it. And one day, your user may even have a portal to run it by themselves. This chain of documenting, scripting and automating will improve your quality and efficiency. Sustainable productivity is not about doing things quickly but about reducing the technical debt.

In the same way, when you script some actions, the quality of your code will be crucial in making it usable by others: naming and documenting, verifying return codes to trap errors, making procedures re-runnable, and cleaning the state at the end. For example, I have seen a `login.sql` that formatted a few columns with `“noprnt”`. You or a colleague will waste a lot of time one day when running a script with a column that has the same name and which will just disappear. How long will you waste before tracing this to the `login.sql`? Each time you use `COLUMN` in `sqlplus`, you should `COLUMN CLEAR` at the end. In IT as everywhere else, it is smart to leave the place in the same state you found it.

## 2 BEING PROUD OF LARGE DATABASES

I like autoextending datafiles (and bigfile tablespaces for databases in terabytes) but letting datafiles grow automatically doesn't mean that they remain managed. You should set the `maxsize` to what you expect, with a little margin, and set an alert when it reaches a threshold. And then, rather than increasing the `maxsize` lazily, take the time to check if this growth is normal or not. This decision is made by the application owners for the user tablespaces, or the DBA for other tablespaces.

I have seen a `SYSAUX` tablespace that reaches 1TB. This is not normal and is possibly due to bugs in an AWR purge that have not been fixed. And even when the bug is fixed, once `SYSAUX` is very large, the purge procedures will fail, and it will continue to grow. The technical debt will be paid when you need to upgrade the database. The upgrade procedure may have to change some AWR tables, or when you have to restore any tablespace because `SYSTEM` and `SYSAUX` must be fully restored as well. And to repair that,

## WHAT IS THE “TECHNICAL DEBT”?

The best illustration I've seen for this is “Technical Debt Is like a Tetris Game” by Jonathan Boccara. When you play Tetris, you can be very efficient at the beginning of the game. Just press the space bar quickly without thinking too much. But soon, your game will be completely fragmented, and it will be difficult to get back to a manageable situation. You can take the time to design the bricks in the best way to avoid any holes, or keep holes that you manage, like when you leave space for the vertical bar.

The same in IT. You can do things very quickly in software, with apparently good productivity. But this productivity is not sustainable and may have

bad consequences in the future. You, or your future colleagues, will regret your quick “solution”. When I was at university, the first programming project we had was a good eye-opener for this. We delivered a program that worked, and even did more than was required. But we all had a big surprise when we got the result: a very bad mark. Because “it compiles” or “it works” is not the most important goal. The program we wrote the night before the deadline was completely unreadable and unmaintainable.

The lesson learned: providing a solution is not enough. It must be understandable and maintainable in the future.

you have to run some unsupported actions like the one I mentioned in this blog post: Truncate AWR tables (unsupported): (see [tinyurl.com/PTK-TruncateAWRtables](https://tinyurl.com/PTK-TruncateAWRtables)).

Also, not purging or archiving the audit trail will be a problem later. Everything that grows must have some housekeeping. The same goes for the traces in `DIAG`. Oracle manages some of them with `ADRCI` but not all. The day you will need to look at the trace, to troubleshoot a critical issue, you don't want to encounter an `“ls”` in a directory with millions of files, or with a `“vi”` on a background process trace in gigabytes.

I have seen a 500MB controlfile. The controlfile is not optimised to be this size. You pay the debt when you duplicate the database, each “switch datafile” taking minutes if your storage is not optimal.

## 3 ADDING YET-ANOTHER-MASTER-REPOSITORY TO DESCRIBE THE INFRASTRUCTURE

You can be tempted to describe all your infrastructure (servers, databases, versions...) in an infrastructure database and then generate everything (provisioning scripts, backup schedules...) from there. But everywhere I've seen this done, it was adding a single-point dependency, adding complexity to any operation procedure, limiting the automation possibilities and agility.

First, you don't need it because there are already a lot ▶

of those repositories, with a dedicated purpose. From a list of servers you can get the databases from oratab or Grid Infrastructure, and take all the info from there. From the RMAN catalog you get all the info for files and backups. From Data Guard Broker you get the primary/standbys configuration (yes, not using the broker is another of the worst practices...).

Enterprise Manager Cloud Control gathers information from the targets, and you can add the small amount of missing information (department, line of business, lifecycle, contact, cost centre) with a GUI or command line.

Keep things simple. Rather than building from a “master-meta-database”, it is more efficient to implement and use standards and then get information from configuration repositories. And avoid a single point of dependency between backups, monitoring, auditing, provisioning... you can then evolve one without being dependent on the other.

For more about this, I recommend Ludovico Caldara’s publications and presentations about convention over configuration and why you should not “manage a zoo”.

#### 4 BEING LAZY WITH NAMING CONVENTIONS

We are all weak with documentation because it’s not easy to keep up to date. That’s why we should always favour auto-documented actions, and the easiest way to do this is having a consistent naming convention. When you add a ‘P’, ‘T’ or ‘D’ to the database name for Production, Test, or Development, then you don’t need to maintain a table of database names and environments. Your eyes and your fingers immediately know when you connect to production. Of course, a red prompt is even better and that is easy to set when you can rely on a letter.

You should define naming conventions in your team for database names, database unique names, file paths and service names. Services can have a domain name, like hostnames. This is perfect for distinguishing production from test. Then when you copy production to test you just need to change the default domain name and you are sure that all database links go to the right environment. When you duplicate to a temporary auxiliary instance, for point-in-time-recovery, you define a temporary db\_domain and you are sure it will not register with a listener with an existing service.

I said that service domain names are like hostnames, but I recommend that you don’t use the same ones. Because when you enable EZCONNECT and the service name is not defined, it will attempt to find a host with this name. And if there is a host name resolution for it you will connect to the wrong one or simply wait for TCP timeout. That’s another reason for good naming conventions: isolate the namespaces. For example, have a different pattern for a database unique name, a PDB and an application service. They are all registered as services to the listener, but you must be sure to use only the right one.



#### 5 BEING OVERZEALOUS WITH NAMING CONVENTIONS

I said that naming conventions are good as a replacement to document properties that will not change. But they are bad if you put more information in than that. For example, I have seen databases where the version is part of the name, like DB12 for 12c. One day you will upgrade this database and you don’t want to have to rename it. Also, it adds no value at all as it is easy to query the version from a database. This is the same when mentioning “\_STANDBY” for the standby database. One day, you will switch over and the roles will change.

Instead, maybe add the physical location like “GE” for Geneva if you are sure

that if you move to another data centre you will add a new standby rather than physically moving the servers. But do not put the name of the rack where the servers are, or the name of the storage where the database is, because you may want to change that (online in RAC by adding nodes and/or ASM disks) without renaming everything. When a user connects to a service, they should know the database they connect to and the role (application workload, read/write, read-only...) but not where it runs physically. That’s the goal of HA services, TAF, Application continuity, connection manager, SCAN listener and so on.

## 6 CONFIGURING THE CART BEFORE THE HORSE

I have seen many inefficient configurations because the structure has been defined in the wrong order or in the wrong way, such as buying the hardware first and then thinking about how to configure the RAC cluster, and then not having enough network cards to ensure reliable interconnect. Or configuring huge pages as a percentage of the available RAM rather than from the SGA sizes required. Or using the in-memory option for a data warehouse without thinking first about partitioning and parallel query.

In this case inter-instance parallel query was even disabled with `parallel_force_local=true` and the tables were populated with `distribute`: any query had only half the rows in the in-memory column store and the others had to come via a full table scan from the row store.

## 7 NOT KEEPING CONTROL, OR BEING TOO SMART

Those features that have consequences on the overall system complexity or the overall operation effort should remain under control. Database links are a great feature when you have to share data or database calls between two databases. But this needs to be designed and controlled properly (loosely coupling, no cyclical dependency, documented, correctly updated when the test is refreshed from production...).

Keeping things under control means the DBA creates and manages them. I have seen databases where the developers could create a database link as soon as they had to share some data and I had to troubleshoot bugs with distributed transactions over three databases of different versions. Allowing such things is not being smart, but lazy. The debt is paid later by having to migrate or upgrade some databases.

It gets even worse when materialised views is the quick solution for any performance issue. I have seen a “BI” database generating a terabyte of redo log per day just because of those frequent refreshes. Again, replication and aggregates are valid features when designed and controlled. But this has consequences on the infrastructure. It must be designed by Dev and Ops together.

And finally, granting powerful privileges is OK for sandboxes and development – not production. Dropping a table by mistake in production should never happen because what runs in production should have been run in dev, test and preprod before, with the same automated scripts, and with a validation check between each.

### ABOUT THE AUTHOR

Franck Pachot is Principal Consultant and Database Evangelist at dbi services, Switzerland. He is an Oracle Certified Master, Oracle ACE Director and a member of the Oak Table.



## 8 GOING HEADLONG WITH ONE-OFF PATCHES

Oracle is complex software and we encounter issues (bugs). There are usually two “solutions”: fixes (like a patch) and workarounds (like disabling a feature).

Of course, the fix is the long-term solution when implemented in a Release Update. But in the short term, applying a one-off patch is probably the worst idea. You will start to pay the debt when you will have to ask for a merge, and then you will increase the debt further. I have seen databases with more than 100 one-off patches. That prevents keeping up to date with Release Updates, or you have to roll back the one-off patches, apply the RU, and ask for a merge for the previous fixes that you still need.

Always prioritise workarounds. It is better to ask for the fix to be included in the next RU. I’ve written more about this before in *Oracle Scene* (the previous title for #PTK): “Why you must run on the latest release update” (see [tinyurl.com/PTK-LatestReleaseUpdate](https://tinyurl.com/PTK-LatestReleaseUpdate)).

## 9 DISABLING THE OPTIMIZER STATISTICS

The optimizer’s goal is to find the execution plan for the fastest response time. But there’s something more important for the users: the stability of the execution plan and the predictability of this response time. The right approach for plan stability is SQL Plan Management (SQL Plan Baselines).

But this is about worst practices... Of course, forcing the use of the rule optimizer is a very bad one. Tweaking “optimizer\_index\_cost\_adj” parameter is also a bad one. And stopping the statistics gathering is the worst one because of the technical debt. You may be happy with the results in the first months, but you pay the price after a while.

I was contacted by a user with a critical performance issue where they were not gathering new statistics. How can I “tune” a query when the statistics provided to the optimizer are completely wrong? I can’t. When I asked for the reason, I got: “As far as I remember the statistics were disabled a

couple of years ago for performance reasons.”

That’s a completely wrong approach. In this example, the partitions that were filled at the time they stopped the gathering job have their stats from 10 years ago. The partitions that were pre-created at that time still have `numrows=0` but now contain millions of rows. And the new partitions created after this decision had no statistics, so that dynamic sampling gets the current cardinality. Stale statistics are bad, but a mix of stale, zero and no statistics is even worse.

To add to that technical debt, a lot of `/*+ INDEX() */` hints had been added to the queries, probably to work around the lack of fresh statistics at some point, thus creating another technical debt (all the hints) to maintain.

I’ve also seen many databases with incorrect system statistics. The worst you can do is gather these without verifying them and today the recommendation is to use the defaults. ❌



# Learning from execution plans

The curious case of an SQL statement that had a very long run time – and how a possible solution was found

By Jonathan Lewis

One of the most common requests made to the UKOUG is for more “real life” examples. It’s often challenging to do this, because no one wants to broadcast their failures and very few want to reveal the secrets of their success. Fortunately, you can find some interesting cases on the Oracle Developer Community forum – though you don’t always get to see the outcome of your suggested solutions. Here is such a case, where a simple query presented a surprise and the subsequent conversation produced a nice example of how to use run-time execution plans to solve problems.

## WHAT’S IN AN EXECUTION PLAN?

The question we are usually trying to answer when we review an execution plan is “How can I make this query more efficient?” and the execution plan can tell you three things that help. First, the order in which you visit the tables in your query; second, the method used to visit each table; and finally, the statistics about those visits – how many times something happened, how many rows (or rowids) were acquired and how many were subsequently used or discarded.

From these “mechanical” features of execution plans you can then derive three pieces of “design” information. First, you can get a good idea of how the optimizer has



## 10-SECOND SUMMARY

- ▶ In the Oracle Developer Community forum, the author came across a query where an SQL statement was taking many hours to run without any immediately obvious reason why.
- ▶ After investigation, it became clear that the optimizer had introduced a massive extra load through complex view merging.
- ▶ The probable solution was to tell the optimizer to unnest, but without complex view merging.
- ▶ Although the original poster never responded to say if this worked, it’s likely to be the right starting point in the search for an answer.

transformed your query in the early stages of optimisation; second, you can see where most of the work and most of the time went as the query executed; and finally, you get some ideas about how you can manipulate the query, or correct the object statistics, or change the indexing strategy, to do less work and get a better response time.

Here, we will examine a query, the problem posed by its owner, and the execution plans that told us all we needed to know to come up with a possible solution.

## THE PROBLEM

In the initial statement of the problem we were told that the following SQL statement (running in 10.2.0.5, even though it was reported in March 2020) was taking 50

seconds to return a count of 157,000 rows:

```
select count(*)
from (
  select num_telefono, ind_baja
  from pga_abonos
  where num_telefono < '606000000'
) abo
where abo.ind_baja = 'N'
and num_telefono in (
  select num_telefono
  from pfa_contabon
  where cta_facturac in (
    select
      cta_facturac
    from pfa_contabon
    group by cta_facturac
    having count(*) = 1
  )
)
```

You'll notice the predicate `num_telefono < '606000000'`. This may suggest a classic modelling issue of storing something that appears to be a number as if it were a character string, but we seem to be looking at telephone numbers here so this apparent design flaw may be necessary to avoid losing leading zeros.

The problem is this: when the owner changed that predicate “a little” to read: `num_telefono < '607000000'` the count doubled to 314,000 rows but the run time jumped to 3 hours, 42 minutes and 35 seconds – while the execution plan stayed the same.

### INITIAL OBSERVATIONS FROM THE SQL

The fact that the `count()` doubles doesn't mean we only had to do twice the amount of work: it's possible that the volume of data where `num_telefono` is between '606000000' and '607000000' is much larger than the volume of data where `num_telefono` is less than

'606000000' even though the final result is only a little larger because the (possibly expensive) subquery eliminates most of the extra data.

We see an “IN” subquery, with its own “IN” subquery, and the inner subquery is an aggregate with no filter predicates. I wonder how much data there is in the `pfa_contabon` table and how many distinct values of `cta_facturac` there are and if there are any special values of `cta_facturac` that cover a huge number of `num_telefono` in the increased range and therefore remove them from the final count.

Another thing we might think of at this point is that the optimizer will probably do something to convert an “IN” subquery to an “EXISTS” subquery, in fact it might manage to transform both subqueries, and it might change existence subqueries into semi-joins. Another possibility is that the optimizer could unnest the inner subquery to produce a join with an aggregate view, and then it might use complex view merging to transform from the “aggregate then join” construct into a “join then aggregate” construct and, again, it could finish off by turning the resulting “EXISTS” subquery into a semi-join.

It's useful to be aware of options like this before looking at the plan as that may help us to recognise what the plan is telling us.

### FIRST SIGHT OF THE PLAN

At first we were shown the execution plan that resulted from running the query from SQL\*Plus with autotrace enabled, but that doesn't show us where the work happened or where the time was spent – and it's always possible for autotrace to lie about the real plan. So in the follow-up conversation we got hold of plans from memory (`dbms_xplan.display_cursor()`) after enabling rowsource execution statistics, and this is the plan for the faster query (with the `omem` and `lmem` columns removed):

```
PLAN_TABLE_OUTPUT
-----
SQL_ID dggzstf4a786t, child number 0
select count(*) FROM (select num_telefono, ind_baja FROM PGA_ABONOS where num_telefono < '606000000') ABO where ABO.IND_BAJA = 'N' and
NUM_TELEFONO IN ( (select num_telefono from pfa_contabon where cta_facturac in (SELECT CTA_FACTURAC FROM PFA_CONTABON
GROUP BY CTA_FACTURAC HAVING COUNT(*) = 1) ) )
Plan hash value: 1329905074
-----
| Id | Operation | Name | Starts | E-Rows | Cost (%CPU) | A-Rows | A-Time | Buffers | Reads | Used-Mem |
-----
| 1 | SORT AGGREGATE | | 1 | 1 | | 1 | 00:00:11.56 | 52006 | 39358 | |
|* 2 | HASH JOIN SEMI | | 1 | 24857 | 22M (4) | 20364 | 00:00:11.55 | 52006 | 39358 | 1935K (0) |
|* 3 | TABLE ACCESS FULL | PGA_ABONOS_1 | 1 | 24857 | 9097 (3) | 22237 | 00:00:05.76 | 40290 | 39358 | |
| 4 | VIEW | VW_NSO_2 | 1 | 28M | 22M (4) | 28038 | 00:00:05.61 | 11716 | 0 | |
| 5 | FILTER | | 1 | | | 28038 | 00:00:05.59 | 11716 | 0 | |
| 6 | HASH GROUP BY | | 1 | 28M | 22M (4) | 30123 | 00:00:05.56 | 11716 | 0 | |
|* 7 | HASH JOIN | | 1 | 2867M | 57758 (92) | 133K | 00:00:04.07 | 11716 | 0 | 6087K (0) |
|* 8 | INDEX FAST FULL SCAN | PK_CONTABON | 1 | 526K | 1299 (4) | 30123 | 00:00:00.84 | 5858 | 0 | |
| 9 | INDEX FAST FULL SCAN | PK_CONTABON | 1 | 1196K | 1289 (3) | 1196K | 00:00:00.01 | 5858 | 0 | |
-----
Predicate Information (identified by operation id):
-----
 2 - access("NUM_TELEFONO"=$sno_col_1)
 3 - filter(("NUM_TELEFONO"<'606000000' AND "IND_BAJA"='N'))
 5 - filter(COUNT(*)=1)
 7 - access("CTA_FACTURAC"="CTA_FACTURAC")
 8 - filter("NUM_TELEFONO"<'606000000')
```

Figure 1



The data sizes in this run are smaller than the case described in the original posting – the 11.56 seconds (A-time) and 20,364 (A-rows) at operation 2 correspond to the 50 seconds for 157,000 rows in the initial posting.

Looking at the body of the plan we can see the name VW\_NSO\_2 at operation 4 – the form of the name tells us we have a non-mergeable view generated by Oracle as it unnested a subquery; and we can see that it's the second child of operation 2 which is a "hash join semi" that has a full tablescan of PGA\_ABONOS\_1 as its first child. So pga\_abonos\_1 is the build table, vw\_nso\_2 is the probe table – and the "semi" tells us that it probably represents the first IN subquery that has been transformed into an existence subquery and then into a semi-join.

There are many internally generated viewnames that Oracle uses, almost all of them start with the vw\_ prefix and end with a number, and a common theme to them is that they represent non-mergeable views, which means they represent the opening line of a self-contained query block with a subplan which is essentially independent of the main plan.

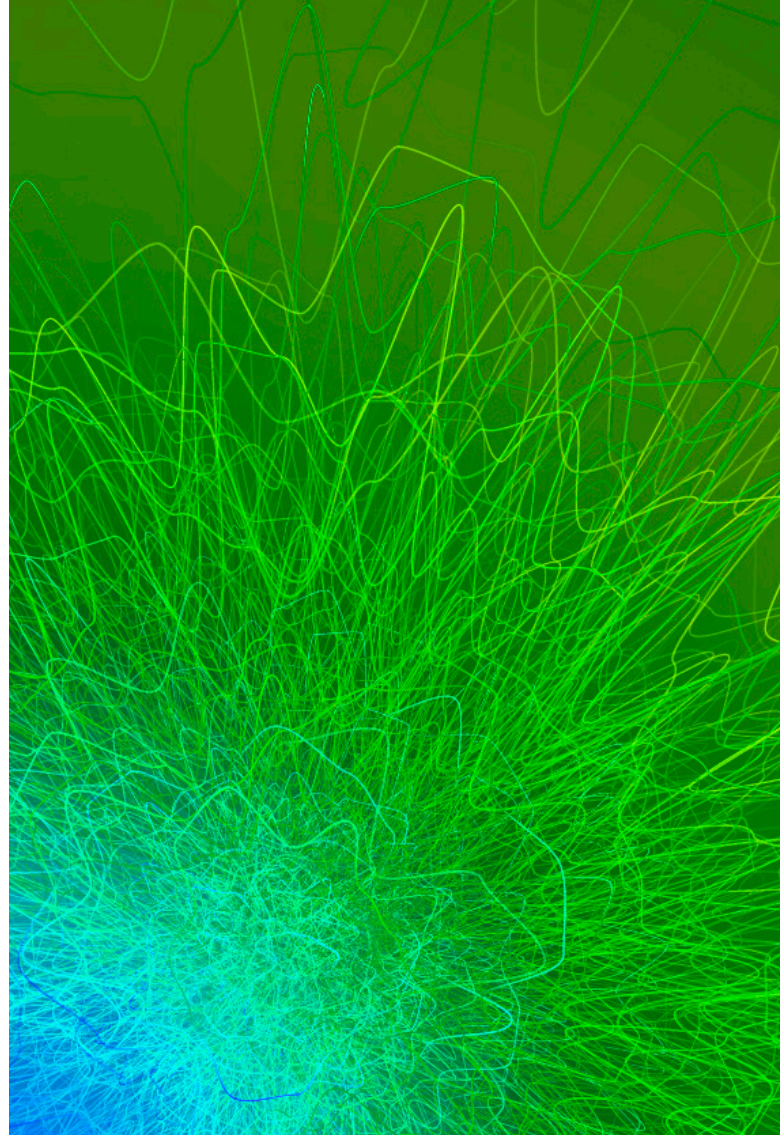
So let's see what's going on inside this view. Running through the basic "first child first, recursive descent" mechanism we find that the view contains a hash join between two "tables" that, judging by their names, are actually the primary key index on the pfa\_contabon table.

So it looks as if the second subquery in the original query has also been transformed away (in this case being replaced by a join rather than semi-join). But we also notice that after the hash join (operation 6) Oracle then does a hash group by (operation 5) followed by a filter (operation 4). How does this structure compare with the actual appearance of the second subquery?

The code has the structure:

```
cta_facturac in (
  select cta_facturac ...
  group by cta_facturac ...
)
```

So we can infer that Oracle has unnested this subquery to produce an aggregate view to join to the other occurrence of pfa\_contabon – but has then used "complex view merging" to transform the plan from "aggregate then join" into "join then aggregate".



Checking the Predicate Information we can see that the optimizer has been able to copy the predicate num\_telefono < '606000000' to operation 8 using transitive closure. We can also see that the filter at operation 5 is the count(\*) = 1 that appeared in the original second subquery.

Amazingly the predicted number of rows (E-rows) from the hash join in this view is 2.867 billion rows, and the optimizer then expects to aggregate this down to "only" 28 million rows (at a cost of 22M – the most significant cost in the entire plan). Luckily at run-time the join produced (A-rows) only 133 thousand rows which could be processed in a few seconds.

Now we compare this with the plan we get with the "tiny" change to the num\_telefono predicate (see below):

Id	Operation	Name	Starts	E-Rows	Cost (%CPU)	A-Rows	A-Time	Buffers	Reads	Used-Mem
1	SORT AGGREGATE		1	1		1	100:42:40.00	52005	40197	
* 2	HASH JOIN SEMI		1	29516	22M (4)	20964	100:42:39.99	52005	40197	2185K (0)
* 3	TABLE ACCESS FULL	PGA_ABONOS_1	1	29516	9097 (3)	28917	100:00:04.31	40289	40197	
4	VIEW	VW_NSO_2	1	28M	22M (4)	28876	100:42:35.49	11716	0	
* 5	FILTER		1			28876	100:42:35.46	11716	0	
6	HASH GROUP BY		1	28M	22M (4)	37088	100:42:35.40	11716	0	
* 7	HASH JOIN		1	2868M	57766 (92)	458M	100:07:39.74	11716	0	6859K (0)
* 8	INDEX FAST FULL SCAN	PK_CONTABON	1	526K	1299 (4)	37088	100:00:00.82	5858	0	
9	INDEX FAST FULL SCAN	PK_CONTABON	1	1196K	1289 (3)	1196K	100:00:01.20	5858	0	

Figure 2

The shape of the plan and the placement of the predicates (which I haven't repeated) doesn't change, the only critical changes are in the A-Rows and A-Time. The number of rows acquired from `pfa_contabon` at operation 8 goes up from 30,000 to 37,000 thanks to the change in predicate for `num_telefono` – at the same time the number of rows produced by the hash join goes up from 133,000 to 458 million.

To generate an extra 458 million rows in the hash join, every one of the extra rows in the build table must, on average, correspond to 65,400 rows in the probe table ( $458M / 7,000 = 65,400$ ). We might like to run some queries against just those 7,000 rows to see how many distinct values they hold for `cta_facturac` and what the pattern of rows per `cta_facturac` is in that subset and in the whole table.

Whatever the fine detail might be, though, the key point is that it takes the optimizer 7 minutes 39 seconds (A-Time) to generate the 458M rows, and a further 35 minutes (42:35 – 7:35) to aggregate then back down to the 37,000 rows we started with. The (most significant) performance problem seems to be in the complex view merge.

## THE SOLUTION

At this point we might consider whether there's a way to express the requirement differently to avoid such a bad plan. On the other hand there's a very obvious defect in the optimizer's strategy that we might be able to address very easily, and sometimes it's a lot safer to tweak the plan than it is to change the code, since an error in the rewrite could result in a piece of code that isn't logically equivalent.

In this case we've seen that the optimizer has introduced a massive extra load through complex view merging. We've identified that point because we can see the numbers and the time, and we recognise the patterns that say "subquery unnest (vw\_nso)" and "complex view merge (join then aggregate)" and in those patterns we can see that it's the late aggregation that has added almost all the time. Conversely the A-rows drops back to a relatively small number before we go into the following hash semi-join, which makes it look as if the preliminary unnesting strategy was a reasonable choice.

So let's just tell the optimizer we want the unnest, but we don't want the complex view merging. Since we're running on 10.2.0.5 this could be quite easy – just edit the innermost subquery to read:

```
select /*+ unnest no_merge */
      cta_facturac
from   pfa_contabon
group by
      cta_facturac
having count(*) = 1
```

With those hints in place (and the unnest is probably unnecessary) I believe the plan will aggregate the rows

from this copy of `pfa_contabon` before joining to the other copy of `pfa_contabon` and the massive explosion in volume won't occur. Of course, it's possible that with just these two hints in place the optimizer may decide it has to change the entire shape of the plan – hinting isn't easy, you have to be thorough – but as a starting point it's the minimum test we should try before we worry about doing anything more complicated like a rewrite.

## HAPPILY EVER AFTER?

It would be nice at this point to say that the owner of the problem tested this suggestion and reported back that the effect was amazing and the query was nearly as quick as the fast query. Unfortunately silence fell and the OP was never heard from again – it's really rather disappointing when someone asks for help, goes the extra step to supply the information when requested, and then you don't hear whether your explanation and suggested solution has worked.

On the plus side, we've been given an example of a query that's clearly a problem, and a good example of the strategic approach needed when you have to work outwards from the database without necessarily having any business-oriented information about what the query is trying to achieve and what the data means.

As a final thought, in this example the owner of the problem was able to supply execution plans after enabling rowsource execution statistics and re-running the queries. In a production system there are two alternative methods for acquiring the same information provided you're running 11g or later:

- ▶ Enable `sql_trace` by `SQL_ID` for every execution of the query – if it's run more than once. The plan dumped into the trace file will contain almost all the information you need (though the "Starts" column doesn't appear until 12.2.0.1).
- ▶ If you're licensed for the diagnostic and performance packs you can look at the output from `dbms_sql_monitor` (`dbms_sqltune` until 12c) – if you can execute the `report_sql_monitor()` procedure within an hour or so of the query running. This, too, will give you lots of run-time stats about the query – though it won't report the predicate information. Any query taking more than five seconds, or running parallel, will automatically be monitored.

For further notes on these two options see the following on my blog ([jonathanlewis.wordpress.com](http://jonathanlewis.wordpress.com)): `sql_trace` (dated May 2014), and `SQL Monitor` (dated April 2018). ❌

## ABOUT THE AUTHOR

Jonathan Lewis has more than 30 years' experience using Oracle software. He has published three books about Oracle, the most recent being *Oracle Core*. He is semi-retired but still does some online consulting.



# Connectivity agent installation on Linux

How to download and install the Oracle connectivity agent on a host machine – and how to leverage the agent group to create a connection with an on-premise Oracle DB.

By Ankur Jain



## 10-SECOND SUMMARY

- ▶ **Connectivity agents can be used to integrate on-premise applications to Oracle Integration Cloud (OIC).**
- ▶ **This article covers the steps needed to install and run a connectivity agent.**
- ▶ **It also covers key areas such as: how to monitor the agent's health, how to run the agent in high availability, and how to delete an Agent Group, as well as how upgrades are managed.**

**C**onnectivity agents help to integrate on-premise applications with Oracle Integration Cloud (OIC). The agent is required for OIC to exchange messages with on-premises applications, for example: Database, E-Business Suite, REST/SOAP API etc.

## PRE-REQUISITES

The Oracle connectivity agent is certified with Oracle JDK 8+ and on the following operating systems:

- ▶ Oracle Enterprise Linux 6.x
- ▶ Oracle Enterprise Linux 7.2
- ▶ Oracle Enterprise Linux 7.5
- ▶ RedHat Enterprise Linux 6.6
- ▶ RedHat Enterprise Linux 7.2
- ▶ RedHat Enterprise Linux 7.5
- ▶ Suse Linux Enterprise Edition 12 SP2
- ▶ Windows Standard Edition 2016

It also requires:

- ▶ 8GB memory with 4GB heap size dedicated for agent JVM
- ▶ Either Internet connectivity or the OIC host name whitelisted on the host machine

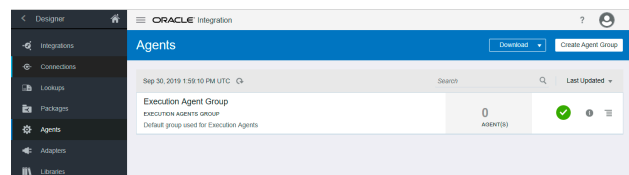
Note: Open JDK and others are not supported.

## DOWNLOAD & RUN THE CONNECTIVITY AGENT

### Create an Agent Group

Before we start agent installation, it's necessary to create an Agent Group. The reference of the agent group is provided during the connectivity agent installation. In order to create the Agent Group, you'll need to perform the following steps:

- ▶ Log in into the OIC console and navigate to the "Integrations" navigation
- ▶ From the "Designer" menu, click "Agent". You will land on the "Agents" page
- ▶ On top of the page, look for the "Create Agent Group" button

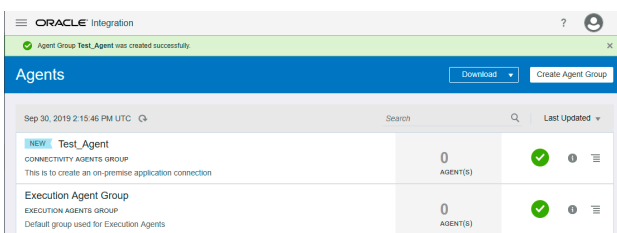


- ▶ Click on the "Create Agent Group" button – the New Agent Group dialog is displayed
- ▶ Enter your information as follows and click on the "Create" button:



Field	Description
<b>Agent Group Name</b>	Enter a meaningful name. The name must be unique among all agent names. Below is the criteria to enter the name: <ul style="list-style-type: none"> <li>▶ Letters (A-Z, a-z)</li> <li>▶ Numbers (0-9)</li> <li>▶ Spaces ( )</li> <li>▶ Special characters ( _ - )</li> </ul>
<b>Identifier</b>	The identifier would be picked up automatically based on the name you enter, but in upper case – but you can edit it. Note: Once the agent group is created it cannot be updated, but you can delete it
<b>Agent Type</b>	By default, Connectivity Agent is displayed and cannot be updated
<b>Description</b>	Enter a meaningful description

▶ The moment you click on the “Create” button, the agent group will be shown on the “Agents” page. A success message will be shown on the top. Initially the count will be shown as 0. Once connectivity is installed using the agent group identifier, the count will increase automatically. (We’ll see more on this later in the article.)



### Download connectivity agent

- ▶ In the left navigation pane, click “Integrations”, then click “Agents”
- ▶ Click “Download > Connectivity Agent”



- ▶ A zip file `oic_connectivity_agent.zip` will be downloaded

### Run connectivity agent

Now you’ll need to follow these steps in order to install the

connectivity agent:

- ▶ Move the agent installer on the host machine and unzip the file `oic_connectivity_agent.zip`
- ▶ Find the “InstallerProfile.cfg” and input the following information:

```
# Required Parameters
# oic_URL format should be https://hostname:ssl
Port oic_URL=
agent_GROUP_IDENTIFIER=
#Optional Parameters
oic_USER=
oic_PASSWORD=
# Proxy Parameters
proxy_HOST= proxy_PORT=
proxy_USER=
proxy_PASSWORD=
proxy_NON_PROXY_HOSTS=
```

Property	Description	Sample Value
<code>oic_URL</code>	HTTPS URL for the Oracle Integration host. The port is 443.	<code>https://xxxx.integration.ocp.oraclecloud.com:443</code>
<code>agent_GROUP_IDENTIFIER</code>	This is the identifier for the connectivity agent group created in Oracle Integration. The identifier name is case sensitive.	<code>TEST_AGENT</code>
<code>oic_USER</code>	Oracle Integration username. When the agent runs for the first time, this field, if provided, is encrypted in the properties file. If this field is not provided, you are prompted to enter the username at agent startup and it is not persisted with.	<code>ankur</code>
<code>oic_PASSWORD</code>	Oracle Integration password. When the agent runs for the first time, this field, if provided, is encrypted in the properties file. If this field is not provided, you are prompted to enter the password at agent startup and it is not persisted with.	<code>welcome@123</code>
<code>proxy_HOST</code>	These parameters are required only if the connectivity agent is used with a proxy.	<code>12.11.44.11</code>
<code>proxy_PORT</code>		<code>1211</code>
<code>proxy_USER</code>		<code>User</code>
<code>proxy_PASSWORD</code>		<code>Password</code>
<code>proxy_NON_PROXY_HOSTS</code>		<code>example.com</code>

## Sample file

```
# Required Parameters
# oic_URL format should be https://hostname:sslPort
oic_URL=https://xxx.integration.ocp.oraclecloud.
com:443
agent_GROUP_IDENTIFIER=TEST_AGENT
```

```
# Proxy Parameter
proxy_HOST=
proxy_PORT=
proxy_USER=
proxy_PASSWORD=
proxy_NON_PROXY_HOSTS=
```

- ▶ Set the “JAVA\_HOME” property to the location of the JDK installation
- ▶ Set the “PATH” property

```
export JAVA_HOME=/home/opc/jdk/jdk1.8.0_221
export PATH=$JAVA_HOME/bin:$PATH
```

```
opc@instance
[opc@instance]$ export JAVA_HOME=/home/opc/jdk/jdk1.8.0_221
[opc@instance]$ export PATH=$JAVA_HOME/bin:$PATH
```

- ▶ Run the connectivity agent installer from the command prompt: “java -jar connectivityagent.jar”
- ▶ Enter OIC username and password when prompted

```
[opc@instance]$ java -jar connectivityagent.jar
Proceeding to install a new agent ...
Enter your OIC username : ankurjain.jain26@gmail.com
Enter password for ankurjain.jain26@gmail.com :
```

- ▶ Once it is installed, a success message will appear:

```
Done with Agent installation & configuration... Starting
Agent for message processing.
```

```
Agent started successfully... Now available for new
messages...
```

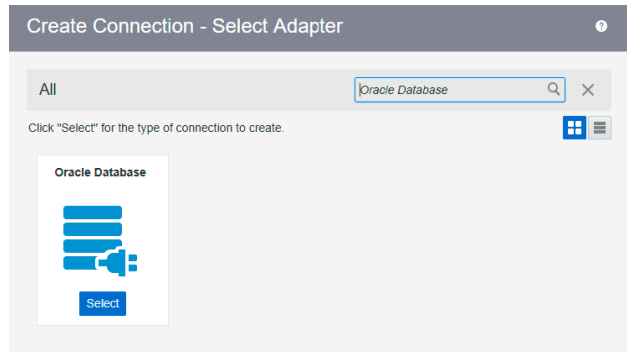
```
Done with Agent installation & configuration... Starting Agent for message processing.
Agent started successfully...Now available for new messages...
```

## CREATE A CONNECTION WITH AN AGENT GROUP

After the successful installation of the connectivity agent, we can create a connection with the on-premises application. Only Agent Groups whose status is green in the monitoring tab can be used to build a connection.

In order to create a connection, take the following steps:

- ▶ From the Oracle Integration home page, click “Integrations > Connections”
- ▶ Click on the “Create” button and select the Adapter. In this case, select “Oracle Database”

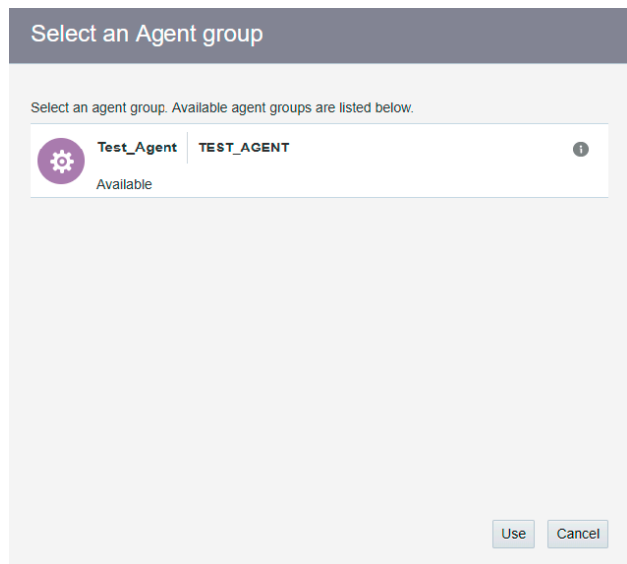


- ▶ From the connection dialog box, enter the following information and click on the “Create” button:
  - **Name:** Enter a meaningful name
  - **Identifier:** Enter the unique identifier
  - **Role:** Select role either Invoke or Trigger or Trigger and Invoke
  - **Description:** Enter the description (optional)

- ▶ Click on the “Configure Connectivity” button, enter the following information and click the “OK” button:
  - **Host:** Enter database hostname or IP address
  - **Port:** Enter database port
  - **SID:** Enter Service ID

Click on the “Configure Security” button, enter the following information and click the “OK” button:

- **Username:** Enter database username
- **Password:** Enter database password
- **Confirm Password:** Confirm database password
- Click on the “Configure Agent” button, choose the available agent and click the “Use” button



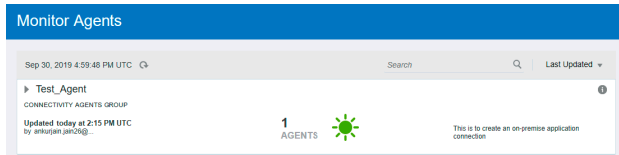
- ▶ Click on the “TEST” button from the top right corner. This test executes the ping command on the on-premises instance when the connection is associated with an agent. The test should be performed successfully.

## MONITORING THE AGENT

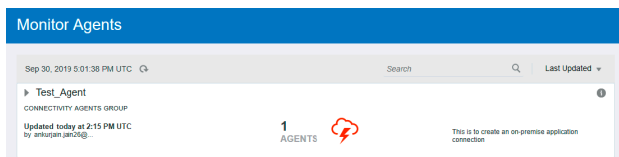
You need to check the health of the connectivity agent from time to time to see if it is up and running or not. In order to do

this, take the following steps:

- ▶ From the Oracle Integration home page, navigate to “Monitoring > Agents”
- ▶ Find the Agent – the Green icon means the agent is ready to serve messages



- ▶ Red means the agent is down and can't serve messages



### CONNECTIVITY AGENT IN HIGH AVAILABILITY

The connectivity agent can be installed in high availability by installing the agent twice on different hosts. During the installation of the agent, you should use the same agent group identifier.

Some points to keep in mind:

- ▶ The file adapter doesn't support high availability environments.
- ▶ Only two connectivity instances can be installed per agent group.
- ▶ Both hosts on which the agent is installed must have the same network set-up.
- ▶ You can install the agent on the same host machine but it is not recommended. Install the agent on different hosts to utilise high availability.
- ▶ Ensure both the agent instances can access the same endpoints.

### RESTARTING THE AGENT

We can start the connectivity agent whenever required.

- 1 Stop the agent in either of the following ways:
  - Enter Ctrl + C on the host where the agent is running
  - Search for the connectivity agent process and kill it

2 Restart the agent

```
java -jar connectivityagent.jar
```

### UPGRADING THE CONNECTIVITY AGENT

When a new version is available for the connectivity agent, your host is automatically upgraded without any intervention required. When Oracle Integration is upgraded, the agent is upgraded within a four-hour window. This is online activity and there should be no downtime or interruption of the services that are running.

The connectivity agent upgrade occurs as follows:

- 1 A check is made of the version of the agent installed on your on-premises host.
- 2 If the agent version on your host is older than the latest available version, the new version is downloaded to your host.
- 3 The downloaded ZIP file is unzipped.

- 4 A back-up directory is created.
- 5 A back-up copy is made of your existing installation in the new back-up directory.
- 6 Older artefacts are replaced in the agent home directory of your installation.
- 7 The endpoints are quiesced.
- 8 The agent is shut down and restarted.
- 9 You are notified of the upgrade success.

### DELETING AN AGENT GROUP

The Agent Group can only be deleted if it is not associated with any connection and if the agent is not running. To delete an Agent Group, first stop the connectivity agent and then remove the Agent Group reference from the connection wherever the Agent Group is being used.

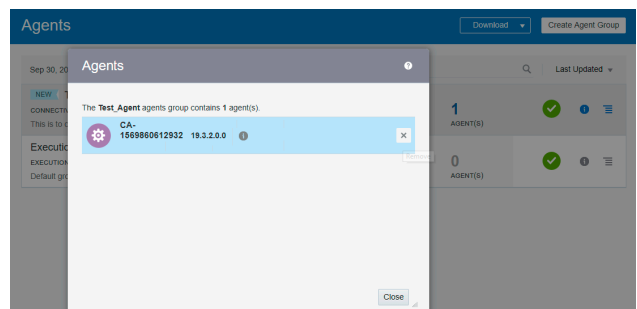
In order to delete an agent group, perform the following steps:

- ▶ Enter Ctrl + C or kill the connectivity agent process using this command:

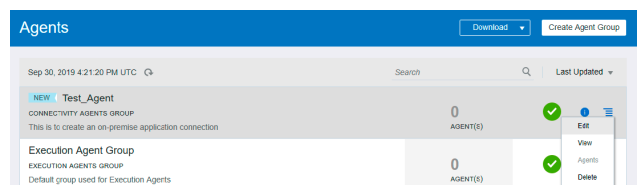
```
kill -9 agent_PID_number
```

The “agent\_PID\_number” can be found in the “AGENT\_INSTALL\_DIRECTORY/pid” file

- ▶ From the Oracle Integration home page, click “Integrations > Agents”
- ▶ Find the agent group to delete. If the number count for the agent group is zero, you can delete the agent. Otherwise, you must first click the number (it can be 1 or 2). Delete those agent instances first. If any connections are using the agent, you cannot delete the agent instance.



- ▶ From the hamburger menu, select “Delete”.



- ▶ Finally, select “Yes” when prompted to confirm. ❌

### ABOUT THE AUTHOR

Ankur Jain is an Oracle ACE Associate and General Manager at Nile Technologies where he manages the integration delivery for one of the organisation's prestigious clients. You can find his blogs at: [www.techsupper.com](http://www.techsupper.com) and [youtube.com/TechSupper](https://youtube.com/TechSupper)



# Check out the Business Apps Edition



As usual, we've divided #PTK into two halves – one for UKOUG's Business Apps community and one for our Tech community.

They're both related, as are our two communities, and we're sure everyone will find each half useful.

**Now take a look at Business Apps to see what we mean.**

We'd love to hear what you think of #PTK, what you like and dislike about this issue, and any ideas you have for future editions – after all, this is *your* magazine and a key benefit of UKOUG membership. So please send your comments and suggestions to:  
**[editor@ukoug.org](mailto:editor@ukoug.org)**

## #PTKOnline

You can view this latest issue online and access the archive of #PTK and Oracle Scene editions here:  
**[ukoug.org/ptk](http://ukoug.org/ptk)**