1. Data Management Introduction

1.1 Oracle data management strategy

This is the Oracle University module on Oracle Data Management strategy. I'm going to explain Oracle's Data Management strategy to you. We're going to talk about what it means to be simply complete and completely simple. It's a fun play on words, right? So I'm not going to waste any of your time. Let's dive right in.

App development paradigms are in a rapid state of transformation. Modern app development is simplifying and accelerating how you deploy applications. Also simplifying how data models and data analytics are used. Oracle data management embraces modern app development and transformations that go beyond technology changes. It presents a simply complete solution that is completely simple. Immediately you can see benefits of the easiest and most productive platform for developing and running modern app and analytics.

Let's start by looking at how Oracle Data Management is easy and most productive for development of our modern applications. Oracle Database is a converged database that provides best of breed support for all different data models and workloads that you need. When you have converged support for application development, you eliminate data fragmentation. You can perform unique queries and transactions that span any data and create value across all data types and build into your applications. This also includes structured and unstructured data. The Oracle converged database has the best of breed for JSON, graph, and text while including other data types, relations, blockchain, spatial, and others.

Now that we have the ability to access any data type, we have various workloads and converge data management that supports all modern transactional and analytical workloads. We have the unique ability to run any combination of workloads on any combination of data. Simply complete for analytics means the ability to include all of the transactions, including key value, IoT, or Internet of Things, along with operational data warehouse and lake and machine learning.

Oracle's decentralized database architecture makes decentralized apps simple to deploy and operate. This architecture makes it simple to use decentralized app development techniques like coding events, data events, API driven development, low code, and geo distribution. Autonomous Database or ADB now supports the Mongo database API adding more tools for architectural support.

Autonomous Database or ADB has a set of automated tools to manage, provision, tune, and patch. It provides solutions for difficult database engineering with auto indexing and partitioning and is elastic. You can automatically scale up or down based on the workload. Autonomous Database is also very productive. It allows for focus on the data for solving business problems. ADB has self-service tools for analytics, data access, and it simplifies these difficult data engineering architectures.

Now that we discovered how development is easier and most productive, let's look at running modern apps and analytics. Running applications means thinking about all the operational concerns and solving how to support mission critical applications. Traditionally this is where Oracle excels with high availability, security, operational solutions that have been proven over the years.

Now having developer tools and the ability to scale and reduce risk simplifies the development process without having to use complex sharding and data protection. Mission critical capabilities that are needed for the applications are already provided in the functionality of the Oracle Data Management architecture. Disaster recovery, replication, backups, and security are all part of the Oracle Autonomous Database.

Even complex business critical applications are supported by the operational security and availability of Oracle ADB. Transparently, it provides automated solutions for minimizing risk, dealing with complexity, and availability for all applications. Oracle's big picture data management strategy is simply complete and completely simple with the converged database, data management tools, and the best platform.

It is focused on providing a platform that allows for modern app development across all data types, workloads, and development styles. It is completely scalable, available, and secure, leveraging the database technologies developed over several years. And it's available consistently across the environment. It is the simplest to use because of the available tools and running completely mission critical applications.

Simply complete and completely simple. Easy to remember and easy to incorporate into your existing architectures. Thanks for watching and don't forget to get your free Cloud account if you haven't signed up yet. Take care.

1.2 Oracle Database Offerings

Hey there. Kay Malcom here. I'm going to talk about the different Oracle Database offerings we have at our back and the ways you can actually use them, if you have ever wondered about on-premise, DBCS, ADP, Exadata, or third-party cloud. So let's talk about this a little bit more.

Let's compare Autonomous Database to how you ran the database on premise. How you ran the database on the cloud using our earlier Cloud Services, Database Cloud Services, and Oracle Exadata Cloud Service. The key thing to understand is Autonomous Database, or ADB, is a fully managed service. We fully manage the infrastructure. We fully manage the database for you.

In on premise, you manage everything-- the infrastructure, the database, everything. We also have a service in between that that we call a co-managed service. Here we manage the infrastructure, and you manage the database. That service is important for customers who are not yet up to 19c. Or they might be running a packaged application like E-Business Suite. But for the rest of you, ADB is really the place you want to go.

And why do you want to do that? Well, because it's fully managed and, because it's fully managed, is a much, much lower cost way to go. So when you talk to your boss about why he wants to move to ADB, they often care about the bottom line. They want to know like, am I going to lower my costs?

And with ADB, because we take care of a lot of the tedious chores that DBAs normally have to do and because we take care of best practices, configurations, we can do things at a really low cost. We have some analyst reports now that have looked at this cost of ownership around ADB.

In particular, on this slide, we have Wikibon Research. And they analyzed the costs of running Autonomous Transaction Processing versus running it on prem, running Oracle Database on Amazon using their RDS-managed service, and then ATP. What they found was very interesting. You've heard a lot of noise about-- yeah, you know. Amazon is actually pretty expensive. And this research sort of confirms it.

If you run an Oracle database on RDS for Amazon, they found that the cost was either about the same as on prem or even higher. If you want to use disaster recovery, any kind of configuration like that, multizone configuration, the cost was actually about 50% higher to use Amazon versus on prem.

But with ATP, they found the cost was almost half of what you could do on prem. And so it's a really nice report. I recommend you go take a look at it.

Now, you know, you may have heard that moving to Autonomous Database requires some extra work. And there is some element of truth there. So what did we do? We've got a tool that helps you look at your current database on prem. This tool will analyze what features you're using and let you know, hey, you know you're doing something that's not supported for ADB, for example.

If you're running some release before 19c, we don't support it. If you're doing stuff like putting database tables in the system or sys schema, we don't support it. There are a few things that very few customers do that we don't support. And this tool will flag those for you.

And then the next step, it's pretty simple. You just use our Data Pump import/export tool to move your data out of your database on prem into the object store on the Cloud. And then you simply import-- you know how to use Data Pump to import-- the data off the file and the object store into the database. Then you're done. Pretty simple process.

If you've already tuned this application on prem for years and years, there's no need to use the autotuning capabilities of ADB, if you don't want to. Your DBAs can sort of preserve all the existing tuning they did-- all the existing indexes on the database. DBAs have a really important role here to make sure your database continues to run well, your existing database.

Now, we more recently have come out with a new service on our Cloud called the Database Migration Service. This service automates those steps we just talked about on the previous slide. With Autonomous Database Migration Service, you can just point us at your source database on prem or even on some other cloud. Whatever it is, we will take care of everything from there and move that, go through all the steps I talked about on the previous slide for you, and move your database to ADB on the Cloud.

Even better, we now are working with our Applications customers to make it really easy for them to move their packaged applications to Autonomous Database. The Oracle development teams that built JD Edwards, PeopleSoft, Siebel have now all certified that those packaged applications can run with Autonomous Database no problem. Our EBS team is working on it. And that'll be coming soon, sometime next year.

And once you do that, we have a fully managed service available on our Cloud that lets you take your entire application stack on the middle tier and the database tier, move it to our Cloud. Move the database part to Autonomous Database. And they will also manage your middle tier for you.

And then we're working with third parties also to certify their applications with Autonomous Database. We list a few here-- MESTEC, MineSense, NEC, Zebra. And then we're also working, of course, with all the tools' vendors who have tools that run against the Oracle Database, people like Tableau, BusinessObjects, and more. And we've got all of these guys and gals certified. And if you have more questions about whether your tech stack is certified on ADB, there's a link at the bottom of the slide that lets you look them up there.

And then, of course, we have lots of real customers who have moved on-prem databases to ADB. TaylorMade is a pretty well-known golf club manufacturer. They had a big Oracle data warehouse on prem. They have successfully moved that to Autonomous Data Warehouse a couple of years ago. They are getting much lower cost of ownership around doing this. They're getting much better performance. And basically, the vision of Autonomous Data Warehouse is playing out very well for them.

I mentioned earlier that we have this great tool called APEX, or Application Express. We have a version of Autonomous Database just for any APEX application. And just to review, what is APEX again?

Well, APEX is a low-code tool. It is our low-code tool that lets you rapidly build data-driven applications where the data is in the Oracle Database, really easy and really rapidly. We estimate at least 10 times faster than doing traditional coding to build your applications. What we're seeing is much, much higher productivity than that. Sometimes 40, even 50 times faster coding.

Out of the box, it comes with really nice tools for building things-- your classical forms and reporting kinds of workloads. It gives you things like faceted search and capabilities to do things like see on an e-commerce website where you get to choose things like dimensions, like I want a product where the cost is in this range. And it might have some other attributes. And it can very quickly filter that data for you and return the best results.

And it's a really nice tool for iterating. Now, if your user interface doesn't look quite right, it's very easy to tweak colors and backgrounds and themes. Another reason it's so productive is that the whole middle tier part of your application is fully automated for you. You don't have to do anything about connection management or state management. You don't have to worry about mapping data types from some other 3GL programming language to data types. All of that is done for you.

And we've got millions of APEX apps, including live labs out there in the market. 3,000 or so are being built pretty much every single day. So the bottom line is if you're still doing traditional coding to build your database applications, you really should rethink things. That should be the exception. Do something quick. Do something fast. You should be using a low-code tool like APEX for building pretty much all your database-related apps.

And here's an example of a customer. Wilson Truck Lines, they're using a combination of ADB and APEX to build an application for their business. And their story is really kind of interesting. They actually originally built an application on Amazon. And they used traditional coding methods, recommended by Amazon, which is like lots of code.

They were able to redevelop an application that took them three months to code on Amazon in two days using APEX. This is that 45 times productivity I was talking about. And like I said, this is actually pretty typical with APEX. And so this application gave them great performance and great interactive response times. And they're very happy with it. The combination of ADB and APEX really rocks.

Here I just want to show you a little bigger picture of what we're doing here. We're not just talking about a data warehouse on the cloud that does massively parallel query really fast. This is what the original first generation data warehouse on the cloud did. It's much more than that.

So beyond this, we've built a collection of self-service tools that would make your analysts and your data scientists really productive. I mean, you don't need an engineer to get data from your data sources and transform it. You don't need them to build the OLAP models. We've got self-service tools that do the data loading for you in this drag-and-drop fashion.

We have ETL transformation tools. Again, they let you specify transformations in a drag-and-drop fashion on the screen. We have all sorts of other tools and, in the service, the full power of the converged analytic technologies, things like graph analytics, spatial analytics, machine learning. All of this is built into this new platform.

Now, a big, new capability around machine learning is something that we call AutoML. That lets any data scientists give us a data set, tell us what the key feature is that they want to analyze, and what the predictions are. And we will come up with a machine learning model for them out of the box. Really that easy. Plus, we have the low-code tool APEX that I mentioned earlier.

So this environment is really powerful for doing more than traditional data warehouses. We can build data lakes. We are integrated with the object stores on Oracle Cloud and also on other clouds. And we can do massively parallel querying of data in the core database itself and the data lake.

And of course, we support all third-party popular analytic tools. And here's an example. Seattle Sounders Football Club is a customer using Autonomous Data Warehouse, or ADW. They were able to build a very powerful analytic environment to analyze their sports statistics. And as you ladies and gents know, sports are becoming really focused on analytics, all different kinds of sports.

They were able to build an ML model quickly to detect things around various models to help them figure out, you guessed it, how to score more goals. That's the objective in soccer, right? And of course, the pay-per-use model of the cloud lets them lower cost quite a bit. And beyond the Seattle Sounders, we're also working with the English Premier League. They are using Autonomous Data Warehouse technology for their environment.

Now, just to start closing things out, beyond the database technology, there is the business side. We want to make your path to ADB really easy from a business standpoint, a decision-making standpoint as well.

So if you're an existing Oracle customer, you have an existing Oracle Database license you're using on prem, we have something called BYOL, Bring Your Own License, to OCI. We have the Cloud Lift Service. This huge cloud engineering team across all regions of the world will help you move your existing on-prem database to ADB for free.

And then, finally, we announced fairly recently something called the Support Rewards Program. This is something our customers are really excited about. It lets them translate their spending on OCI to a reduction in their support bill. So if you're a customer using OCI, you get a $0.25 to $0.33 reward for every dollar you spend on Oracle's Cloud.

You can then take that money from your rewards and apply it to your bill for customer support, for your technology support even, like the database. And this is exactly what customers want as they move their investment to the cloud. They want to lower the costs of paying for their on-prem support.

And so just to summarize what we talked about today, my big focus here was ADB. And we first talked about our converged database architecture for software that, of course, we deliver with ADB.

And the big thing is ADB and the fact that it's a fully managed cloud service. It manages the Exadata infrastructure. It manages fully the database software on top of it. It gives you "out of the box," best-practice configurations for security, for availability, and for performance. You avoid all the problems you might run into on prem when you miss one of those best practices. It automatically patches and tunes the database for you as well.

Now, we've talked about money. This lowers costs greatly. So ADB has lots of value. But the big thing I think to think about is really that it lowers costs. It lowers that cost via automation, higher productivity, less downtime, all sorts of areas. We showed you some TCO studies from Wikibon Research and from IDC to support those claims. We're not making them up.

And finally, we talked a lot about how it's really quite easy now, as most of you have now moved a lot of your databases up to 19c. ADB is a great place to go. Take those existing Oracle Databases you have. Move and modernize them to a modern cloud infrastructure that's going to give you all the benefits of cloud, including agility and lower cost.

And finally, we really want to make it easy for you to adopt this technology. So on our Cloud, we have something called the Always Free Autonomous Database Service. This service lets you get your hands on ADB. Try it out for yourself. You don't have to believe what we claim about how great this technology is.

And we have other technologies like Live Labs that you can find on developer.oracle.com/livelabs that lets you do all kinds of exercises on this Always Free ADB infrastructure. Really get your hands dirty. And see for yourself how productive it can be. Thanks for watching.

1.3 Multicloud and Hybrid Cloud

While enterprises are making significant cloud investments, the reality is that less than 20% of existing mission critical enterprise workloads have already been migrated to the cloud. This number is rapidly changing. Currently almost 70% of enterprises are already moving or planning to move on premise workloads to the cloud.

So what happens when an organization has already chosen a strategic cloud provider yet has mission critical databases like Oracle Engineered Systems on premise? Well, there are many solutions which have fallen into stalled categories. They're stuck on premise with no path forward to migrate to the Cloud and unable to receive the modernization, the innovation and economic benefits that the Oracle Cloud offers. So what do you do?

Well, in response to this dilemma, as Gartner reported last year, 81% of organizations are now working with two or more public cloud providers. A multicloud strategy gives companies the freedom to use the best possible cloud for each workload with the added cost saving advantages. Let's take a look at what a multicloud strategy actually is.

Multicloud is the use of multiple cloud computing and storage services in a single heterogeneous architecture. This also refers to the distribution of cloud assets, software, applications, and more. It's across several cloud hosting environments, and this multicloud environment aims to eliminate the reliance on any single cloud strategy provider. It is a recognition of the fact that in the majority of cases, not one cloud provider can be everything to one organization. A multicloud strategy not only provides more flexibility for which cloud services and enterprise can choose to use. It also reduces dependence on a single cloud hosting provider.

Here are some additional reasons for adopting a multicloud platform. Best of breed innovation for specific workloads. Reduced risk and increased redundancy happens when an application is deployed across two different cloud providers. Reduced latency in which dispersed organizations can reduce latency by choosing local public cloud vendors based on each facility location. Compliance and governance. Here some organizations may need to use multiple Cloud storage providers to adhere to various government regulations and data sovereignty laws.

Multicloud economics. Most organizations subscribing to multicloud capabilities use the public cloud infrastructure, avoiding the need to build and maintain their own data center by building a virtual data center environment in the cloud without needing a single piece of physical hardware. Reduced vendor lock in occurs when an organization is not invested in only one cloud service provider and spreads the risk across multiple clouds.

In order to connect the clouds, options need to provide performance, reliability, and security. Private connectivity that does not open connections over the public internet will allow for predictable performance and a secure solution. This also factors in that the circuit can be redundant and take advantage of multiple cloud providers' economics to reduce cost and deployment time. Customers can use third party connectivity providers for which they can enable multicloud connectivity between OCI virtual cloud-- between OCI Virtual Cloud networks and Amazon Virtual Private Cloud, for example.

These are called Megaport Cloud Routers or MCR. Oracle Cloud Infrastructure FastConnect is an example of one of these types of connections. It forms a connectivity directly between the customer and Oracle. A VXC is essentially a private point to point ethernet connection without the need for any physical infrastructure to connect to services on the Megaport network.

Let's take a look at Oracle's approach, which enables new hybrid models including dedicated regions, edge computing, data transfer, and high performance database appliances, all cloud controlled but secure and performant. We have well over 30 public regions and our architecture allows us to automate the process of building new regions allowing globally consistent scale in services and regional disaster recovery capabilities.

We've increased resiliency at the individual node level to support more resilient, smaller deployments, and simpler availability for enterprise applications. We reach cloud scale with a much smaller footprint. This allows us to offer an industry first dedicated Cloud region installed and operated in your customer data center to satisfy requirements for data sovereignty or low latency.

A dedicated region has the full power of our public cloud regions and is installed, operated, and upgraded by Oracle Cloud, just as we do our own public regions. We also offer our high performance Exadata and Autonomous Database solutions on customer premises and delivered as a service. Here again we install and manage the infrastructure and the services. We've also enabled blazing fast hybrid cloud and multicloud architectures with our Microsoft Azure and Oracle FastConnect services.

Because of the Oracle and Microsoft partnership, we can offer an interconnected multicloud solution to support your multicloud environments. Azure and Oracle Cloud are interconnected today. So you can migrate and run mission critical enterprise workloads across clouds. FastConnect and express route direct connection with sub two millisecond latency and no intermediate service provider required. We have unified identity and access management via single sign on with automated user provisioning to easily manage resources across clouds. There's collaborative support of workloads across clouds. So for example, Oracle applications on Azure with Oracle Database Cloud Services. We connect best in class services across clouds.

Thanks for listening to Oracle's approach to partner with our other Cloud vendors. For more information about OCI, please be sure to take the free OCI foundation specialist exam. Thanks for watching.

2. Converged Database

2.1 How to Simplify Application Development?

Welcome to Oracle University's lesson on how to simplify application development, part of the Oracle converged database series. My name is Zach Shallcross, and I'll walk you through today's course. Once upon a time, not so long ago, App Dev for business purposes was relatively simple. A small team of developers could use a single development tool and database to quickly develop a business application.

But as time had gone on, enterprises have been under increasing pressure to stay competitive, App Dev has become more and more complex. Developers face a whole new set of requirements for enterprise apps. Apps not only had to capture information on what customers were doing, but they had to also anticipate customers' needs, and allow the business to create value or insights from data in real time.

These modern or data-driven apps operate on a diverse set of data-- spatial, documents, sensor, transactional, pulled from multiple sources, often in real-time, and they create value from that data in very different ways compared to traditional applications.

For example, they may use machine learning to make real-time recommendations to customers or detect fraudulent transactions. They may also use graph analytics to identify influencers in a community, and target them with specific promotions, or use spatial data to track deliveries.

Data-driven apps are often built using new data-driven development methodologies or paradigms to help accelerate the development process. These include microservers, then processing, API-driven development, or low code development.

These apps also need to be developed and enhanced using continuous delivery as they're often 24/7 applications. They can't afford to take any downtime for patching. This has led to a very different approach to App Dev, where each aspect of the business is implemented as a separate service, with its own single purpose database.

With the approach, developers are able to quickly begin a new project as the single purpose databases offers us convenient model that fits the project's purpose. And it's easy to adopt APIs that seem natural for that data model.

At first glance, the single-purpose database approach appears to be a good option. The developers are happy because they get exactly what they need to begin the project. But as so often happens, the development requires change mid project when an unforeseen business need crops up, and we are asked to run complex analytics on the data stored in the document store.

In fact, even the simplest of projects require multiple data types and workloads, so you will need multiple single-purpose databases, increasing the complexity and fragmenting the data across multiple single-purpose databases.

There may also be a steep learning curve with these single-purpose databases. Each database will require app developers to use their proprietary APIs or languages and transactional models rather than a standard like SQL. Plus, these proprietary APIs can often lead to apps getting logged in to a single-purpose database or Cloud provider, as no other database or Cloud provider supports these APIs.

This approach also requires a more complex application code to be written, and maintained to propagate data from one store to another, and to keep them in sync. Each single-purpose database requires ongoing tuning, securing, scaling, troubleshooting, patching, requiring specialized skills and unique management tools.

You may also need specialized application code and/or skills in your org to create and maintain a secure, scalable, and highly available solution, which is going to be custom and potentially costly and complex to maintain. Unfortunately, your organization must bear the brunt of the integration work required to make a single-purpose database approach feasible on a large scale.

You will need personnel who are knowledgeable about the operational aspects of each single-purpose database. Your security policies will have to be re-implemented in every database, and your apps will become more complex to deal with, propagating data from one database to another.

In fact, integrating single-purpose databases has the potential to become a job that never ends. So how can you deal with this level of complexity? At Oracle, we believe there's an easier way. By providing synergistic data technology for each of the new development methodology, we've made it simpler to take advantage of the approaches to develop data-driven apps.

We want to eliminate the data fragmentation introduced by multiple single-purpose databases by offering support for all data types and data models in a single converged database. And we also want to make it easy for developers to take advantage of any data type or data model by providing a uniform API in the form of SQL and REST for all data types and workloads.

A converged database is a database that has native support for all modern data types, and the latest development paradigms built into a single product. But not only are they available in single converged database, they have been engineered to work together, allowing developers to seamlessly mix and match data types and models.

For example, a recommendation query can span three different models. First, find a customer's friend within a three-hop friendship, which is graph, or two, identify those who have watched similar movies, JSON, and those who have provided feedback with a five star rating, key value.

Don't let the diagram on the left fool you, though. We're not asking you to go back to use a monolithic database for all of your database applications. A converged database still allows each application service to be modularized or containerized with its own independent data store or database container. This means the data types or workloads you are working with won't dictate your architecture.

So how does a converged Oracle database simplify the latest development paradigms or methodologies? As I mentioned earlier in this presentation, each modern development methodology or paradigm dramatically accelerates data-driven apps development. But they do have the potential to complicate data architecture.

A converged Oracle database can eliminate this complexity by pairing each development methodology with a synergistic data technology, making it much easier to take advantage of them. Let's take a look at this in action. One of the most popular development paradigms is microservices. The microservices architecture provides each developer with great agility and independence, as each aspect of the business or application is implemented as a separate service.

Oracle simplifies microservice architecture by allowing each service to store its data in logically separate data containers called pluggable database or PDB, providing all of the isolation and independence needed in a microservices architecture. Just as you would deploy your applications or services using Kubernetes containers, you can deploy the database for those services as pluggable database, making microservices simple.

So let's briefly summarize. Oracle makes it simple to build data-driven apps by providing synergistic data technologies, one converged database for all data types and models engineered to work together. and lastly, one converged database greatly simplifies development and operations. So this concludes our lesson today. Thank you.

2.2 Oracle Autonomous JSON Database

We map collections to tables. A collection can be thought of as a logical abstraction over a table. At the second line, when we create the collection "employees," it is actually creating a table within a single JSON column that will store the documents. This table will be created in the admin schema.

The third line inserts a JSON document into the collection, calling the "insertOne" function on the collection. This causes a row to be inserted into the backing table as shown on the left. The document will be inserted into the collection using a SQL insert statement. The JSON document will be bound to the query as OSON, our binary JSON storage format.

The last line of this program calls the find method on the collection. It passes a filter expression to the function that selects documents where the job attribute is equal to programmer. This is sometimes called a query by example. It selects all documents that look like the value passed in.

These filter expressions can get more complex. We will see a few more examples during the demo. But on the left side, it shows that this "find" is translated to a SQL query. The query shown here is just meant to convey the concept and is not the actual one we generate. It's a bit more complex. I will show you how to view the actual query when we do the demo.

The idea here is that the table backing the collection is still accessible using SQL when you need it. You get the best of the SQL and NoSQL worlds with AJD. If you're building a microservice or application where you simply want to persist your data, using SQL may make the application code a bit more verbose than it really needs to be. And for some developers, it may even be a barrier to entry.

The API allows you to very simply put, get, update the JSON objects in the database. It is actually creating a table with a JSON column in the background. This backing table is still accessible using SQL. And at a deep level in the database kernel, we have added extensions to SQL to allow it to process JSON data, so you can use SQL when it's needed.

If you are an app developer that likes MongoDB, you can continue to leverage your skills, use your DevTools, and have access to drivers, libraries, and frameworks. At the same time, if you are a developer, analyst, or data scientist that likes SQL, you can write analytic queries and reports using SQL directly over JSON. You can query different types of data at once, and you can use all the existing SQL-based tools.

One very practical advantage AJD has is that you can connect to it using Oracle drivers, even old versions of Oracle drivers. Many third-party data tools like Power BI and Tableau already know how to connect Oracle. It's one of the most widely used databases.

So you can connect these tools to AJD and have them directly access your JSON collections through relational views of the data. And I will show you how to create a relational view over JSON during the demo.

In conclusion, we've seen that JSON and the Oracle Autonomous Database gives you the best of both worlds. You get all of the advantages of a document database and all of the advantages of a full relational database as well. That concludes today's lesson. My name is Sara. Thanks for hanging out with me.

2.3 Developing on Oracle Autonomous Database - Using Graph

Welcome to Oracle University's lesson on developing on Oracle Autonomous Database using Graph. My name is Kamryn. Let's get started. The objectives are to understand the graph feature of the Autonomous Database and how to create, query, analyze, and visualize graphs. Graph databases explicitly store relationships between data entities. Hence, they are helpful for discovering and understanding relationships.

When you're modeling your data as a graph, the entities, such as account, owners become vertices and the connection or activity, such as a cash transfer that connects them becomes an edge. Let's look at a simple example of modeling an accounts and transaction tables as a graph. The Bank Accounts tables, contains account IDs and possibly some other related columns. The rows of this table become vertex with an ID and some properties which correspond to the values in the other columns of the table.

The bank transactions table contains rows with details on a cash transfer between two accounts. So these rows become edges between the vertices, representing those accounts, and the amount, date, and any other details become the properties of the edge. Each edge is assigned a unique ID, too. It may have a direction. In the example above edge 1,000 represents the transfer of $1,000 from account one to account 672.

What can you do with the graph model and structure? You can query it and visualize direct or indirect relationships and often expose hidden connections and patterns. For example, the visualization on the top left shows a known pattern called distribution and accumulation, which indicates potential fraud. Money is distributed across multiple accounts before being accumulated in a destination account.

This is sometimes used to circumvent regulations such as maximum amount in an online betting or political contributions. Similarly, you can run graph analytics to detect communities. That is, groups which have more interactions among themselves than with other elements of the network. The other two visualizations show examples of communities and their interactions. Now, let's look at examples of business problems which benefit from graph analytics.

Here are some business problems which benefit from graph analytics. Financial institutions use it to improve the effectiveness and efficiency of their anti-money laundering and fraud detection operations. Money moving in unusual patterns often merits investigation. Cycles are a particularly strong indicator. That is, money originating at an account and passing through multiple layers of intermediate accounts before finally landing back in the original account. In Graph, algorithms are great at detecting cycles.

In manufacturing or network management, you can model your bill of materials or network as a graph and perform what-if analysis. Which assemblies and product lines are affected if we change the design of a component? What will be impacted if a switch or router in the network goes down? Graph algorithms enable path analysis that helps determine the impact on elements that are directly and indirectly connected to the affected component or network element. It helps detect a downstream impact, which is not immediately obvious without such a path analysis.

Similarly, Graph algorithms are good at identifying clusters. This helps in finding similarities among customers and, hence, in product recommendation. A product catalog often has different and varying attributes which describe a product. Graph data models have a flexible schema where each vertex or edge can have different properties. So as new products are added or existing ones are updated with new attributes, these are easily accommodated in the model.

Oracle Graph provides all of the functionality and flexibility we've discussed so far. It is free and included in all Oracle database editions. It consists of a means to model and store your graph data, a means to query it and to analyze it. The query language is called Property Graph Query Language. It's like SQL but extends it to enable you to specify graph patterns such as cycles, which are then evaluated against the graph. There are over 60 built-in graph algorithms to perform graph analytics, such as finding paths, detecting communities and employers or ranking vertices by importance.

And lastly, a means to visualize and interact with query or analysis results. Graph analytics are operations on all or parts of the graph. Some are iterative. Some are recursive. But most are computationally expensive. Oracle Graph has over 60 built in operations implemented as efficient parallel in-memory algorithms. Algorithms for detecting communities, evaluating structures or ranking vertices may operate on the complete graph, while path-finding, such as finding the shortest route between a start and end node, usually operates on portions of a graph.

A graph analytics driven insight helped one of our customers reduce false positives in their fraud detection processes. They have discovered that cycles of up to six or seven hops merit further investigation. They also knew that such cycles usually involve accounts, which done a lot of transfers. That is, money flowing into or out from that account. In Graph parlance, those vertices or accounts had a high in-degree incoming transfers plus out-degree outgoing transfers. So those accounts were important.

The insight, however, was in using a different notion of importance, namely PageRank. PageRank includes both the number of connections and the rank or importance of connected vertices when computing the rank of a vertices. They achieved a reduction in false positives when they used PageRank and identifying cycles and, hence, accounts and transactions to investigate further. In this lesson, we use the Graph feature of Autonomous Database and learn to create, query, analyze, and visualize graph.

2.4 Developing on Oracle Autonomous Database - Using Spatial

Welcome to Oracle University's lesson on developing an Oracle Autonomous Database using Spatial. My name is Kamryn. Let's get started. In this lesson, we'll learn how Autonomous Database supports spatial data management, see how to develop spatial applications with ADB using developer languages and APIs, and see how to use spatial features of ADB using Spatial Studio no-code tools. The value of spatial data is that it helps you identify patterns, determine relationships, and understand correlations.

For example, do a majority of your in-store customers come from neighborhoods with similar socioeconomic characteristics? Can we provide fiber network services to a new subdivision in town, or do we need to install additional infrastructure? The Oracle database contains hundreds of functions to perform the necessary spatial analyzes to answer those questions. Let's look at why spatial data is so important and how you can use the Oracle database to make sense of this kind of information.

Spatial data specifies a real world location and, hence, relates disparate entities, such as people, events, activities that are or occur at that location. Virtually, any enterprise data includes some aspect of location, assets, incidents, activities, and transactions. They'll all happen somewhere. Oracle allows you to manage location as just another attribute in your enterprise data and answer these types of questions.

Location analytics is integrated with the rest of Oracle's data management capabilities, making it simple for developers and analysts to use. This allows you to deal with all kinds of geospatial data making use of the multi-model capabilities of our converged database. From the locations of stores and customers to street networks to satellite images and 3D city models from laser scans, you can store, manage and analyze it all right in the database together with your other business data.

Locations and their relationships offer the ability to address a wide range of business problems. Is this a duplicate incident report for an accident? Who was the nearest first responder? Does this ship to location and have a sales tax? How many competitors are near a planned franchise location? Which assets are in a flood zone? Does a plan route cross any restricted areas? All these questions relate to the location relationships and can be answered in many forms, like maps, dashboard reports, and notifications.

Spatial analysis and visualization are used in almost any industry, but specifically, in defense, public sector, utilities, energy, retail, and finance. Our spatial offering actually consists of three parts. The first is, of course, the core spatial functionality in the database to store, query, and analyze spatial data via SQL. We provide hundreds of operators and functions to search, categorize, filter, aggregate, enrich, or other words, modify spatial data.

The second part is a number of components, APIs and services that make it simpler for developers to build spatial applications. This enables programmers to use their language or framework of choice, whether that's Java, Python, Javascript, RESTful services, to do things like visualize maps or perform advanced analytics such as route calculations and so on. And the third part is a self-service tool called Spatial Studio, which makes it easier than ever for non-experts to get started with spatial.

Spatial Studio has an intuitive browser-based, self-service, drag-and-drop user interface. It's designed to enable analysts and non-technical users to get started with creating maps and performing spatial analysis without writing a single line of code. It provides no code access to the hundreds of spatial operators and functions we have in the database. It renders the result as different kinds of maps so that analysts can put their data into context, and it can hand off the data to Oracle Analytics Cloud for further analysis.

With Spatial Studio, you can also convert address data to coordinates so that you can plot the locations on a map. This operation, called geocoding, leverages the Oracle Maps Cloud Service in the background so that the user doesn't need to manage the reference data locally in their database. Note that Spatial Studio also helps developers because it provides a graphical user interface for data preparation for index creation and for data publishing. But most importantly, it supports the developer in creating spatial analysis or analytic workflows, which would otherwise be too difficult to code.

Spatial is included with Autonomous Database without any additional cost but requires additional compute resources. It is available as a deployable JEE application and on the Oracle Cloud Marketplace. Step-by-step directions at Oracle LiveLabs and search for spatial. Spatial Studio provides a no-code interface to the vast array of spatial analysis operators in the database.

These operators let you ask questions and perform operations such as, which is the nearest warehouse to this facility? Which has the following items in stock? Are these recent spate of drop call complaints clustered in a localized area? Which is the nearest cell tower to these drop call complaints? Do the service areas of our current retail banking centers and those of a recent acquisition overlap? Can we close some outlets and still serve the same customer base?

And then, visualize the results of those analyzes on an inviting, intuitive, and interactive map display, in this case, showing customers colored according to the closest warehouse and size according to the distance, which can then be published and shared with stakeholders or included in another web application. Any data such as street addresses, place names, postal codes, or GPS coordinates is spatial data.

The Oracle Autonomous Database includes features to store, query and analyze such data using the language or development framework of your choice. And Spatial Studio provides no-code access to this functionality. Since it is native functionality, you can use it with other features such as JSON, Graph, or ORDS. In summary, the location aspect of your enterprise data is easy to harness in your applications.

3. Exadata and Base Database Service

3.1 Oracle Base Database Service Overview - Part 1

Hello, and welcome. My name is Eddie Ambler. Today, we're going to take a look at the Oracle Base Database Service, which runs in Oracle's public cloud. The Oracle Base Database Service enables you to maintain absolute control over your data while using the combined capabilities of the Oracle Database and Oracle Cloud Infrastructure. Over the next few minutes, I will introduce you to this service, ensuring to highlight some of its key features and capabilities.

The Oracle Base Database Service provides you with the ability to deploy full featured Oracle Databases in Oracle's public cloud. The Oracle Base Database Service offers database systems on virtual machines that are available around the world in Oracle Cloud Infrastructure regions. The service allows you to run Oracle Database Standard Edition and Enterprise workloads on flexible virtual machine shapes. It leverages a multi-level security model that helps to protect databases with features like always on database encryption.

The Base Database Service supports licensing the database server compute cores with a License Included, or Bring Your Own License model. The service currently supports the deployment of Oracle databases with version 12c, 19c, 21c, and 23ai, using virtual machines, which are referred to as DB systems. With the Oracle Base Database Service, you can choose to run and manage workloads on a single instance virtual machine DB system or a 2-node rack virtual machine DB system.

The Base Database Service is a co-managed service in which Oracle manages the infrastructure, and you manage the contents of the database VM. To simplify user managed tasks, the service provides you with a rich set of cloud automation functions that allows you to conduct tasks on your schedule.

Now, let's dig into the powerful set of cloud automation functions and tools that the Base Database Service provides to allow you to easily conduct database lifecycle operational tasks on your schedule. Having automated database lifecycle management capabilities in the cloud helps IT teams to avoid configuration and maintenance errors, and reduces the required database administration workload, and the time required to deploy new database environments.

Database lifecycle management in the cloud automates time-consuming tasks such as database provisioning, resource scaling, patching, backup and recovery, HA and DR deployments with Data Guard, all on your schedule, with security built in at all layers. Cloud automation frees your DBA team from these tasks, allowing them to focus on delivering value-added functions to their line of business.

When you run the Oracle Database in the cloud, you have a choice of two cost-effective compute core licensing models that you can leverage to best meet your business needs. The License Included model includes all of the Oracle Database Enterprise Edition options, plus the Oracle Database Enterprise Manager packs.

This subscription model is ideal for customers without sufficient existing Oracle Database licenses, seeking to build new applications, or customers seeking to use Oracle Database features beyond what they are currently licensed to use. The cost efficiency of license included model and elastic OCPU metering usually drives innovation with Oracle Database features that might have been out of reach due to cost with the on-premise licensing model.

The Bring Your Own License model is designed to minimize costs when migrating existing workloads to the cloud. In a BYOL model, customers can deploy their existing database and database option licenses to their Oracle Cloud Database Service. When a customer brings an Oracle Database Enterprise Edition license to the Oracle Cloud with the BYOL model, they are granted the rights to use transparent data encryption, diagnostic pack, tuning pack, data masking and subsetting pack, and real application testing without needing to bring those licenses to the cloud.

BYOL customers can also save on their existing Oracle Database software support costs by leveraging the Oracle Support Rewards program, which provides them with 25% credit of their cloud spend that can be applied towards their on-premise support bill.

When leveraging Oracle Base Database Service with the License Included model, you can choose to license your compute cores using one of four licensing tiers. Each cloud database licensing tier provides access to additional database options, and also includes all of the database options from the lower tier.

The first tier is Standard Edition 2, which includes the license for multitenant for up to three or less pluggable databases per container, Machine Learning, Spatial and Graph. The second tier is Enterprise Edition, which adds database features such as Data Guard and the Enterprise Management Packs for Data Masking and Subsetting, and Tuning and Diagnostics.

The third tier is Enterprise Edition High Performance, which adds a Life Cycle and Cloud Management Packs, as well as partitioning, advanced compression, and advanced security. And if you need multitenant with more than three PDBs, you'll need Enterprise Edition High Performance as well.

And the fourth tier is Enterprise Edition Extreme Performance, which has all of the previously discussed features, plus Active Data Guard, Real Application Clusters, known as RAC, and Database In-Memory. Since security is at the top of Oracle's Cloud mission, note that all of the licensing tiers include Oracle's Database Transparent Data Encryption.

If you're licensing the Base Database Service with RAC or the Exadata Database Service with License Included, note that both will require the use of the Enterprise Edition Extreme Performance CPU licensing tier.

The Oracle Base Database Service enables customers to build, scale, and manage full featured Oracle databases on virtual machines. The key benefits of running databases on VMs are ease of getting started, durable and scalable storage, and the ability to run real application clusters to improve availability. With this service, you are the only one who has root access to your database VM.

When launching a virtual machine DB system, you select the Oracle Database Edition and version that you want to be created on the virtual machine DB system. A single container database is then created in the VM with the database version you selected. You can then choose the desired number of OCPUs and corresponding memory size that meet your workload requirements. With Base Database Service, OCPU usage is billed by the second, based on the license tier selection you make.

Oracle Base Database Service users block volumes that are attached to the VM for its data storage. For the database storage, simply specify the amount of storage that you want for data, and the cloud automation will do the rest. For backups, the Base Database Service has a choice of using object storage or the recovery service, which is the default option for the backup destination.

The recovery service offers more backup and recovery features, and the ability to achieve a better recovery time and recovery point objective at the same price point as using object storage for the backup destination. When creating the Base Database Service, you will have a choice of Ampere, AMD, and Intel flexible shapes and high performance OCI block volume storage to choose from that can satisfy a broad range of application and business requirements.

The maximum number of OCPUs, memory, and storage for your database on a virtual machine depends on the shape you choose. Note that the amount of memory allowed is based on the number of OCPUs selected, and the ratio of memory to OCPUs depends on the shape that you select.

After creating your virtual machine database system, you can scale the usable block volume storage online up to 80 terabytes for data and 20 terabytes for RECO. Note that the available I/O performance will increase as you scale up the amount of allocated storage. Although Base Database Service supports single instance and 2-node RAC deployments, you cannot scale your existing VM from a single node VM to a 2-node VM.

Deploying the Base Database Service as a 2-node RAC requires the use of the Extreme Performance Edition license. When you're scaling the OCPU with a 2-node RAC virtual machine DB system, the CPU change is done in a rolling manner on one virtual machine at a time.

Let's review the four different types of compute shapes that are available for the Base Database Service DB system shape. These shapes range in CPU configurations from 1 to 64 cores, to support customers with small to large-sized database workloads. There are three flexible VM shape types to choose from. The Ampere flexible shape is the most economical shape, but it is only supported on logical volume manager and on single node DB systems.

Also, Oracle Database Standard Edition is not supported on Ampere A1 shape Base DB systems. For Ampere flex VM shapes, the OCPUs can be assigned from 1 to 57 in increments of 1, and the memory is allocated at 8 gigabytes per CPU. As you increase the allocated OCPUs, the available network throughput for your VM configuration will be increased by 1 gigabit per OCPU, all the way up to 40 gigabits per second.

The Intel and AMD flexible shapes both provide you with the ability to run single instance or 2-node RAC database systems, and both support Standard and Enterprise Edition databases. For the Intel flex VM shapes, the OCPUs can be assigned from 1 to 32 in increments of 1, and the memory is allocated at 16 gigabytes per OCPU. As you increase the allocated OCPUs, the available network throughput for your VM configuration will be increased by 1 gigabyte per second per OCPU, up to 32 gigabytes per second.

For the AMD flex VM shapes, the OCPUs can be assigned from 1 to 64 in increments of 1, and the memory is allocated at 16 gigabytes per OCPU. As you increase the allocated OCPUs, the available network throughput for your VM configuration will be increased by 1 gigabit per second per OCPU, all the way up to 40 gigabits per second.

The fourth and final shape option is the older standard fixed shapes. Note that fixed VM shapes are available from 1 to 24 OCPUs, and that the memory is allocated at 15 gigabytes per OCPU. And that to scale a fixed shape DB system, you must change to another shape, size. Remember when planning for OCPU scaling, that changing the shape from a single instance VM DB system to a 2-node RAC VM DB system is not supported.

3.2 Oracle Base Database Service Overview - Part 2

Now, let's take a look at the available storage architecture options for creating the Base Database Service VM DB system. When creating your Base Database Service instance, there is a choice of Logical Volume Manager or Grid Infrastructure Storage Management. For single node virtual machine deployment, you can select Logical Volume Manager or ASM for your storage management architecture.

When deploying a single node Base Database Service DB system, you can select a fast provisioning option that allows you to create your database VM system with block storage, using Logical Volume Manager as the storage management software. The default storage architecture for the Base Database Service is Oracle Automatic Storage Management, known as ASM.

When you select ASM Storage Management for the virtual machine database system, ASM uses DATA and RECO Disk Groups by default, when you create a virtual machine DB system. ASM Disk Group allocation will be 80% for data and 20% for the RECO storage. Block storage provides triple mirroring of the data. The use of ASM for the storage management layer is required for 2-node RAC DB system deployment.

Note that DB System clones are also supported for both LVM and ASM storage management. Since a virtual machine DB system uses Oracle Cloud Infrastructure block storage, you will specify your desired storage size when you launch the system. Once the DB system has been created, you can scale up the storage online as needed at any time.

Before we proceed further, let's take a look at the Oracle Cloud concepts of regions, availability domains, and fault domains. In Oracle Cloud, a region is a single localized geographical area, where Oracle has deployed Oracle Cloud Infrastructure. Each region is wholly independent of other regions and can be thousands of miles apart from other regions.

An availability domain consists of a set of data centers within an Oracle Cloud Infrastructure region. The availability domains within a region are interconnected via a low latency network. A region can have multiple isolated availability domains with separate power and cooling. A fault domain is a grouping of hardware and infrastructure within an availability domain. Each availability domain contains three fault domains. Fault domains are used to reduce the impact of hardware failures.

Now that we've covered what fault domains, availability domains, and regions are, let's take a look at how they are used to deliver high availability and disaster recovery protection for the Base

Database Service. When you create your Base Database Service as a 2-node RAC deployment, you select an availability domain. And to improve availability, the individual virtual machines in the configuration are deployed on separate physical servers in separate fault domains.

Placing the database servers on separate fault domains isolates each RAC database instance, so they both won't be impacted by the same network failure, power failure, or outages from the same infrastructure maintenance window. Additional availability can be added to the Base Database Service by adding a standby Data Guard instance in another availability domain to protect against availability domain outages.

You can also create a standby Data Guard instance that is deployed in another region to provide disaster recovery protection. A key feature of Base Database Service is that the Oracle best practices are built in. You no longer need to comb over technical briefs and documentations to figure out how to deploy your database for best performance, availability, and security. Just deploy using the cloud automation, and your system will be optimally configured.

You have the option to deploy Oracle RAC to provide a scalable, highly available database. Oracle RAC protects from unplanned failures by spreading work across multiple database instances. In addition to RAC, system and database updates are deployed in a rolling manner to maintain system availability.

To provide additional availability protection, automatic backups and replication with Data Guard can be easily configured for local HA or disaster recovery using the cloud automation. In fact, the Base Database Service supports all of the Oracle maximum availability architecture technologies, which form the high availability blueprint for Oracle databases in the cloud.

This table is a quick reference to the MAA components and the four license included tiers available for the Base Database Service. The components used to deliver and enhance availability for the Base Database Service are Flashback, Backup & Recovery, Multitenant, RAC, Data Guard, and Application Continuity.

Flashback, as well as Backup & Recovery are available with all license included editions. Multitenant is available in all database editions with up to three PDB without a license. For database instances licensed with the High Performance or Extreme Performance licensing tier, the PDB limit is lifted, and you can deploy up to 4,098 PDBs.

Data Guard is available with all of the Enterprise Editions' license tiers. But if you need the benefits of active Data Guard, then you must use the Enterprise Edition Extreme Performance licensing tier. 2-node database deployments with RAC also require the use of the Enterprise Edition Extreme Performance license.

Finally, Application Continuity is only available with environments license for Enterprise Edition Extreme Performance because it requires active Data Guard and/or RAC licenses. Ensure that the license included edition that you select when provisioning the Base Database Service DB system has the features that aligns with your availability requirements.

At the core of every layer in the Oracle Cloud is security. Oracle strategy to security is Defense in Depth, which is designed to permit authorized work and prevent, detect, and respond to unauthorized work. Defense in Depth works by implementing controls throughout the Oracle stack to strike the prudent balance of risk mitigation and operational efficiency.

The bubble drawing shows the concentric circles that protects data expanded out, so that you can see how each ring adds to the Defense in Depth posture.

The Base Database Service security posture starts by securing the data in the Oracle Database. Oracle Database Security features lead the industry in helping to prevent unauthorized data access, and includes the following. Transparent data encryption to encrypt user data at rest, data redaction, masking, and subsetting to permit users to see only the relevant data to do their job.

Key Vault to separate the control of TDE keys from control of the Base Database Service infrastructure. Customer VMs and databases. Database Vault to control DBA access so that DBAs are prevented from accessing user data with SQL queries. Database Firewall to control what SQL statements can be executed against a database. Data Safe to automatically detect sensitive data and assess risk so that you can better secure your database.

The Virtual Machine DB systems are built from the hardened operating system image based on Oracle Linux 7. It secures the core operating environment by restricting the installation image to only the required software packages, disabling unnecessary services, and implementing secure configuration parameters throughout the system.

The networking is implemented with OCI Virtual Cloud networks or VLANs to isolate access to your databases. To further protect database user and application connections, the Base Database Service provides Oracle Native Network Encryption to encrypt connections from the database clients and applications to the database.

The Base Database Service also constitutes a complete deployment and service and is subject to industry standard external audits, such as PCI, HIPAA, and ISO 27001. These external audit requirements impose additional value-added service features, such as antivirus scanning, automated alerting for unexpected changes to the system, and vulnerability scans for all Oracle managed infrastructure systems in the fleet.

By integrating the security features throughout the stack, Oracle's approach to Defense in Depth security provides you the control you need to govern how instances of your cloud services can be created, accessed, used, maintained, and destroyed.

3.3 Exadata Database Service Overview

now. That we've talked about the Exadata platform, let's turn our focus to running the Exadata Database Service, which runs on the Exadata platform in the Cloud.

The Oracle Exadata Database Service enables you to leverage the power of the Oracle Exadata Database machine in conjunction with the efficiency of Oracle Cloud. With the Exadata Database Service, you get an Oracle Database with all features and options. Each Exadata Database Service deployment is provisioned with a complete Oracle Database installation that includes all of the Oracle Database Enterprise features, plus entitlements to all of the Database Enterprise management packs, and all of the Enterprise options such as Oracle Real Application Cluster, known as RAC, Oracle Database In-Memory and Oracle Multitenant.

The foundation for the Exadata Database Service is the Exadata Database Machine, which has been deployed at thousands of sites around the world. The Exadata platform is the highest-performing and most available platform for running an Oracle Database, with a fault-tolerant architecture featuring scale-out database servers and scale-out intelligent storage servers connected with a fully redundant high-performance RoCE network fabric.

Exadata is an ideal cloud platform. The Exadata Database Service delivers all of the advanced features of Exadata, including SQL offload, smart flash cache, storage index, and hybrid columnar compression. In the Oracle Cloud, customers pay a monthly subscription fee for the Exadata Database Service. The subscription fee is based on the shape of the Exadata Cloud Infrastructure configuration and the number of enabled Oracle Cloud CPUs in the system configuration, which are known as OCPUs, where one OCPU equals one enabled database server CPU core. There is no initial capital cost and no additional data center charges.

All of the supporting infrastructure for the Exadata Database Service is deployed, maintained, and managed by the Oracle operations team, including the data center networking, the private Exadata RoCE Networks, physical Exadata Database servers and storage servers, the firmware, and Exadata storage server software.

The Exadata Database Service includes easy to use, web-based wizards to quickly provision an Exadata system and its associated database. deployments. The Exadata Database Service provides complete service isolation with no overprovisioning of hardware to ensure that response times and throughput are predictable for critical business processes. This contrasts with some cloud service delivery models that silently overprovision hardware and consequently, may not be able to deliver the expected resources during the busy periods.

The Exadata Database service is 100% compatible with on-premise Oracle databases and all existing applications. Exadata Database Service enables existing on-premise Exadata customers to easily embark on a journey to the Cloud without compromising the database performance and availability levels they enjoy with on-premise Exadata deployments. Existing Oracle Database customers who have not yet experienced Exadata can also easily start enjoying the performance, availability, and scalability benefits of the Exadata without compromising any of the database functionality that they rely on.

Exadata in the Oracle Cloud provides the most complete Database Service available. Exadata in the Oracle Cloud provides you the ability to leverage the Oracle Cloud automation functions to simplify operational and database lifecycle tasks, and to leverage all of the Oracle Database innovations and all of the Exadata Database Davis machine innovations across all of its deployment options. This consistency of available features across all of the Exadata deployment options allows for easy portability of your databases, from on-premise to the cloud.

On the left are some of the key features available in the Oracle Database, inclusive of important features like Multitenant, Oracle RAC, Active Data Guard, Partitioning, and others. On the right are some of the Exadata innovations available with the Exadata platform, including recent additions like XRMEM data accelerators and the 100-gigabit per second RDMA overconverged ethernet RoCE internal network fabric. When the Exadata Database Service is deployed with a license included OCPU model, all of these key capabilities and innovations are included.

For the most control over your Cloud Database Service, Oracle offers the Exadata Database Service, which is a co-managed service jointly managed by you and Oracle, in which Oracle manages the infrastructure and you manage the contents of the database. VM leveraging the provided Cloud automation tooling. With this service, customers have root access to the database VM to be able to customize their database environments.

It is supported to install third party agents to meet your operational needs. The Exadata Database Service provides a powerful set of customer-controlled cloud automation designed to simplify conducting operational and database lifecycle tasks on their schedule. With the cloud automation provided for this service, you are in full control of the database VM patching.

With the Exadata Database Service, customers only pay for the compute resources they allocate and can scale those allocated resources as needed. The Exadata Database Service is ideal for migrating any supported on-premise Oracle Database Enterprise version to the Exadata Cloud without requiring any changes. For the simplest data management, you can also choose to run the Autonomous Database, which is a fully managed database service with complete end-to-end automation.

With this service, the database is self-managing and Oracle manages the infrastructure, which eliminates the cost of database and infrastructure management to provide the best total cost of ownership, and frees the DBAs and developers to focus on value added tasks and innovation. With the Autonomous Database, customers only pay for the compute resources used.

To achieve optimal cost savings, auto scaling can be used for workloads with varying workload demands to automatically scale the compute resources up and down, as required to align with the workload demand. The Autonomous Database also has automatic optimizations that improve the performance for data warehousing, transaction processing, and JSON workloads. The focus of this module is the Exadata Database Service, so we'll restrict the rest of our discussion to that service.

One of the greatest differentiators of the Exadata platform versus other database platforms is the flexibility it gives you to choose the deployment location. You can run the Exadata Database machine on premise with a traditional CapEx deployment model, where you own and manage the complete system and have total flexibility as to how you will manage it.

You can also run Exadata in the public Cloud with Exadata Cloud Infrastructure, where Oracle manages the infrastructure and you have automation to simplify your management tasks and can leverage a pay-per-use subscription model that can dramatically reduce your costs.

Oracle understands that you might have requirements that do not allow you to move your data into the public cloud, so it has developed a hybrid cloud offering called Exadata Cloud@Customer, which allows you to experience all of the benefits of a cloud deployment while keeping all of your data safely inside of your own data center.

The Exadata Database Service in a multi-cloud environment can be connected to your application via something like a FastConnect network connection, or be utilized natively in Azure with the Oracle Database@Azure offering. Because it's the same Exadata technology in the cloud as on premise, we can run it everywhere with 100% compatibility and provide a zero downtime migration experience.

Let's explore the Exadata Database Service on dedicated infrastructure, which runs on the Exadata Cloud Infrastructure in the Oracle public cloud. For those of you able to run in the public cloud, this is the ideal solution for running enterprise class database as a service in the cloud. You get all of the power and functionality of the Oracle Database and the performance, availability, security, and scale of the Exadata Database machine with the simplicity of the Oracle Cloud.

With the Exadata Database Service, the Exadata Cloud Infrastructure is dedicated to you, but still provides all of the cloud benefits of Oracle managing the infrastructure, the extensive cloud automation that is available, and the pay-per-use cloud economics. You are able to purchase your service with a choice of license included or the bring your own license model for licensing your database compute CPUs.

The Exadata Database Service is 100% compatible with existing applications running the Oracle Database, which makes it simple to move your Oracle Database workloads to the Exadata Database Service. With the Exadata Database Service, the burden of infrastructure management is eliminated as the Oracle operations team manages the infrastructure, which includes the hardware, the physical database compute and storage servers, and any internal networking elements.

You only manage the database virtual machines and the database instances in those VMs. To simplify that management, Exadata Database Service includes a powerful set of user-controlled automation that allows you to initiate life cycle operations such as provisioning, updates, backups, and more on your schedule through easy to use web-based user interfaces and programmable APIs.

The Exadata Database Service provides you with elastic resource scaling capabilities, which allows you to only license the OCPUs that your VMs are using and the ability to elastically scale those OCPUs up and down at will to align with your workload demand, enabling you to only pay for the resources you use. With a short minimum 48-term for provisioning the service, there's virtually no financial commitment to a deployment.

That same Exadata Database Service is also available on Exadata Cloud@Customer. It provides the same management model, cloud automation and APIs, and flexible subscription-based economics. The difference is that with Exadata Cloud@Customer, all of your data is in your own data center. With Exadata Database Service on Cloud@Customer, you can easily combine a Cloud Database Service with your existing on-premise application without the disruption of needing to migrate some or all of it to a public cloud data center.

Your data stays in your data center next to your applications, satisfying any data residency compliance requirements that might apply to your business. With Exadata Cloud@Customer, databases are provisioned and subscribed to as a service. And Oracle manages the Exadata Cloud@Customer Infrastructure. The control plane is deployed on the chosen public Cloud OCI region to provide the best performance when leveraging other Cloud Services.

Exadata Cloud@Customer provides a consistent Cloud experience equivalent to the public Cloud. It has public Cloud UI/API-driven database provisioning and management. It has the same financial model as public cloud, in which you are able to subscribe to infrastructure and compute cores. And it allows for the use of the pay-per-use subscription model.

Exadata Cloud@Customer has a four-year minimum subscription term for the infrastructure element. Exadata Cloud@Customer is one of the most efficient cloud adoption strategies. It allows you to leverage your existing data center investment. It also allows you to keep your data next to your applications. And it allows you to maintain your existing data security standards and data residency compliance.

3.4 Cloud Management Responsibilities

Let's take a quick look at the cloud management responsibilities for the Exadata Database Service. The Exadata Database Service follows a simple cloud management model. Oracle owns and manages the infrastructure up to the hypervisor. This includes the database servers, storage servers, and the internal fabric network, and the hypervisor.

Oracle will perform infrastructure patching, security scans, and security updates. Although Oracle is managing the infrastructure, the customers can schedule the maintenance windows for Oracle to perform the infrastructure maintenance that aligns best with their business needs.

Customers are not authorized to access Oracle infrastructure. When you subscribe to the Exadata Database Service, you are responsible for managing everything that runs above the hypervisor in the database VM. The customer manages everything above the hypervisor using cloud automation via the UI or REST APIs that are provided.

The guest VM includes the database software, grid infrastructure, there are data, schema, and encryption keys. The Exadata Database Service provides customers with powerful cloud automation tools that give them complete control over the components that they need to manage.

Cloud Automation UI and APIs enable customers to manage VMs and databases. Cloud automation functions enable customers to create, delete, patch, backup, and scale up and down resources along with other tasks. The guest VM runs all supported Oracle Database versions up to 23ai.

Customers can install and manage additional software in the guest VM. Customers also control who has access to the guest VM. The Oracle staff is not authorized to access the guest VM. With the Exadata Database Service, the customer owns everything inside the database, including data, schema, and encryption keys.

To simplify operational and database lifecycle tasks, the Exadata Database Service includes a powerful set of cloud automation functions and tools that allows you to easily conduct these tasks on your schedule. Database lifecycle management in the cloud automates time-consuming tasks, such as database provisioning, resource scaling, patching, backup and recovery, HA and DR deployments with Data Guard all on your schedule with security built in at all layers.

Having automated database lifecycle management capabilities in the cloud helps IT teams to avoid configuration, and maintenance errors, and reduces the required database administration workload, and the time required to deploy new database environments. Cloud automation frees your DBA team from these tasks, allowing them to focus on delivering value-added functions to their line of business.

Here's an illustration of the Exadata Cloud Infrastructure architecture. Just like the Exadata Database machine on premise, the Exadata Cloud Infrastructure consists of a collection of database and storage servers connected by an internal high-performance, secure RoCE network fabric.

Each deployment of the Exadata Cloud Infrastructure uses a scale out architecture that cooperatively runs the database services on both the database and storage servers. The Exadata Database and storage servers can be independently scaled horizontally to meet your business requirements.

Each physical database server hosts one or more database virtual machines, which contains the Grid Infrastructure software, the database home software, and the cloud automation tools. The VMs provide consistent high-performance and isolation between the virtual database servers and the infrastructure.

The database VMs and storage servers communicate over the secure RoCE internal network that provides ultra high-speed compute-to-compute and compute-to-storage networking. The client and backup networks provide 50 gigabit per second connectivity for high bandwidth use cases, such as application connectivity, backup, data loading, and disaster protection using Data Guard.

They connect directly to your subnets in your virtual cloud network, known as a VCN. With Exadata in the cloud, The Oracle Cloud operations team manages the infrastructure via secure and isolated cloud management network. The customer admin manages the contents of the database

VMs by connecting to the Oracle Cloud infrastructure control plane to perform operational and lifecycle management tasks, utilizing the powerful set of provided cloud automation functions via the cloud, console, or APIs.

Oracle realized that while public cloud works well for many customers, there are still many customers who cannot move their workloads to the public cloud. Sometimes regulations or policies require that the data remains under your control in your data center.

Often there is many to many relationship between applications and databases, making it impossible to move a few to the public cloud and leave the others on-prem. Those left behind will see unacceptable latencies when connecting to those in the public cloud, and it's way too difficult to move everything at once.

Lastly, there are many companies that are just not ready for the public cloud yet. They want the control over their data and are not yet comfortable outsourcing that to someone else. For those reasons, Oracle is able to meet you where you are at in your cloud journey and let you run the cloud in your data center with Exadata Cloud@Customer.

Let's now look at the Exadata Cloud@Customer architecture, which allows you to run the Exadata Database Service in your data center to see how it differs from running the Exadata Database Service in the public cloud. In your local data center, we deploy a full-featured Exadata system in one or more RACs, similar to what you would get with an on-premise deployment.

We add two local control plane servers that talk to the Oracle Cloud Infrastructure control plane. The local control plane servers create a secure tunnel from the Exadata Cloud@Customer platform in your data center to the Oracle Cloud Infrastructure in the public cloud.

Over that tunnel, Oracle sends telemetry to monitor the health of the system as well as accept REST API calls from Oracle Cloud Infrastructure to manage the system. Your administrator connects to the Oracle Cloud Infrastructure control plane via a web UI or API calls to monitor and manage the database VMs and database instances running on the Exadata Cloud@Customer.

End users and their applications connect over your internal corporate network directly to the Exadata Cloud@Customer as if it were any other on-prem database server. Exadata Cloud@Customer gives you the same experience as you would get with Exadata Cloud@Customer Infrastructure in Oracle's public cloud, but all of your data stays inside of your data center.

Just like in the Oracle public cloud, the Oracle operations team manages the infrastructure, which includes the physical compute, and storage servers, and any internal networking components. The Oracle operations team, also known as Cloud ops, can access the Exadata Cloud@Customer Infrastructure layer in your data center should the infrastructure experience issues that requires their attention.

Oracle provides customer, and Oracle auditing, and access controls so that the customer can be in full compliance and control of their data. We will discuss more about the access control for the Oracle Cloud operations team in upcoming slides.

3.5 Billing and Licensing
The Exadata Database Service is available with a choice of two cost-effective licensing models that customers can leverage to best meet their business needs. The license included option allows users to subscribe to Oracle Database software licenses as part of the Exadata Database Service.

The license-included subscription model includes all of the features of Oracle Database enterprise edition, plus all of the Oracle Database enterprise manager packs, and all enterprise edition database options. This subscription model is ideal for customers without existing Oracle Database licenses or customers seeking to use Oracle's database features beyond what they are currently licensed to use.

The Bring Your Own License model, sometimes called BYOL, should be selected if your organization already owns the Oracle Database software licenses that you want to use on the VM cluster. The Bring Your Own License model is designed to minimize costs when migrating to the cloud by maximizing the value of your existing investments.

In a BYOL model, customers can deploy their existing Oracle Database enterprise edition licenses and their database option licenses on the Exadata Database Service. When a customer brings a database enterprise edition license to the Exadata Database Service, they are granted the rights to use the following database options without needing on-premise license entitlements.

These would include transparent data encryption, data safe, diagnostics pack, tuning pack, data masking and subsetting pack in real application testing. The Exadata storage server software is also included in a BYOL subscription. So BYOL customers do not have to bring a license entitlement for the Exadata storage software. Note that the Oracle Standard Edition is not a supported version for use with the Exadata Database Service.

Let's take a minute to look at the importance of being able to vertically scale your virtual machines. Licensing costs are based on the number of OCPUs or cores in use. Scaling the cores in use by your VMs as your workload demands scales up and down is important to reducing your costs and is one of the greatest advantages of cloud over on-premise deployments.

When you deploy an on-premise system, you must size it to handle the peak workload as an on-prem deployment does not support scaling CPUs up and down. In fact, you probably oversize it a little as it is difficult to scale an on-premise system and you want to ensure that you have the capacity for the workload spikes.

However, during most normal operations, the system's CPU is normally underutilized and you are paying licensing fees on processor cores that are not in use. Contrast that with the Exadata Database Service deployment, where you can scale the service up and down to adjust to the workload demand.

If you have a workload spike, simply scale up the OCPUs to accommodate. If the demand drops, scale the OCPUs down. Unlike most cloud database services, the Exadata Database Service allows you to scale your OCPUs online with no disruption to the database service.

Defense in depth is Oracle's strategy to control IT systems to permit authorized work and prevent, detect, and respond to unauthorized work. Defense in depth works by implementing controls throughout the Oracle stack to strike the prudent balance of risk mitigation and operational efficiency.

The Exadata Database Service security posture starts by securing the data in the Oracle Database. Oracle Database industry-leading security features help to prevent unauthorized data access and includes the following. Transparent data encryption to encrypt user data at rest. Data redaction, masking, and subsetting to permit users to see only the relevant data to do their job.

Key vault to separate the control of TDE keys for the control of the Exadata infrastructure, customer VMs, and databases. Database vault to control DBA access so that DBAs are prevented from accessing user data with SQL queries. Database firewall to control what SQL statements can be executed against a database. Data safe to automatically detect sensitive data and assess risks so that you can better secure your database.

The Exadata Database Service compute resources are based on the Exadata Database machine and include secure technical implementation guidelines compliant deployments to minimize software configuration risk, minimum packages to minimize software security risk, and token-based SSH access for secure login to service accounts.

The networking is implemented with OCI virtual cloud networks, or VLANs, to isolate access to your databases. To further protect database user and application connections, the Exadata Database Service provides Oracle native network encryption to encrypt connections from the database clients and applications to the Oracle Database. It also provides dedicated client and backup networks for traffic isolation.

The Exadata Database Service infrastructure is used across various industries with demanding security and availability requirements and is subject to stringent compliance testings that include FedRAMP, HIPAA, PCI, and more. The OCI tenancy has several layers of security and identity management.

It offers identity and access management, support for federated identity providers, multi-factor authentication, policy-based access controls, users, groups, and compartments for resource, organizations, and isolation, along with instance principles and resource principles.

By integrating the security features throughout the stack, Oracle's engineered systems approach to defense in depth security provides you the control you need to govern how your cloud services can be created, accessed, used, maintained, and destroyed. The bubble drawing shows the concentric circles that protect data expanded out so that you can see how each ring adds to the defense in depth posture.

One part of security that deserves special attention is how Oracle limits infrastructure access by Oracle operations using operator access control. Many organizations using Exadata Database Service are attracted to the cloud economics and operational benefits, but are required by regulations to retain the control and visibility of their data in any infrastructure in their data center.

With operator access control enabled, cloud operations staff members must request access to your system. Customers can control access to the Exadata Cloud@Customer infrastructure by the Oracle operations team by specifying operator limits. Customers can grant a designated Oracle operator access to a specific component for a specific period of time and with a specific set of privileges.

You can observe and record the Oracle operator commands and keystrokes that the Oracle staff runs in real-time. And if you see something you don't like, you can terminate the operator's access, session, connections, and processes. This gives anyone, especially customers that need to regulate remote access to their infrastructure, the confidence to use the Exadata Database Service.

Since we introduced this feature, we have seen a tremendous interest from customers in regulated industries who see this as a feature that finally enables them to take advantage of the many benefits of cloud. We hope you found this overview of the Exadata Database Service interesting and educational. Thank you.

3.6 Exadata Database Lifecycle Management - Administering Exadata
Hello, and welcome. My name is Eddie Ambler from Oracle Product Management. Over the next few minutes, we're going to discuss the Exadata Database Service and highlight the core database lifecycle management functionalities that are built into the service.

After completing this lesson, you should be able to create custom database and grid infrastructure images, create a Database Home, create a database, perform pluggable database management, enable Data Guard, and perform user-managed maintenance updates and upgrades. You can use the console to perform most of these activities. And of course, you can also use the API, CLI, or SDKs to perform these operations as well.

Let's take a look at how we can create custom database and grid infrastructure software images to establish standard-based solutions that can easily be incorporated into the cloud automation workflows and Terraform-based deployments.

Being able to create a custom database and grid infrastructure software image makes it easy for your organization to create an approved gold image for developers and database administrators to use. Custom software images give you the ability to create customized Oracle Database software configurations that include your chosen grid infrastructure and database versions, release updates, one-off patches, and/or contents of existing Database Home patch inventories.

Custom database and grid infrastructure software images are resources within your tenancy that you create prior to provisioning or patching the Exadata Database Service instance, your Grid Infrastructure Home, or Database Home. You can create custom software images for different types, so ensure to select the correct image type and service type before creating the software image.

You can create custom software images for your Exadata Database Service instance using the console or API. Custom software images are automatically stored in Oracle-managed object storage and can be viewed and managed in the Oracle Cloud Infrastructure Console.

Let's take a quick look at how to create a custom database software image that can be used with the Exadata Database Service. Now that you have your image type and service type selected, click on the Create Software Image button to launch the Create software image input screen.

At the top of the screen, you're able to choose if you will be creating a database or grid infrastructure software image. Once you have made the image type selection, provide a friendly display name for the custom software image.

Then select the compartment to store the software image in. You can choose a different compartment from the compartment you are working in to store the software image. This can be handy if you are creating a software image for development that uses a future standard for the database or grid infrastructure versions that you only want visible from the dev compartment.

Now let's enter the information required to configure the database software image. Select the desired database release and choose a release update, proactive bundle patch, or a one-off patch to include. You can also upload the Oracle home inventory to use as the required patches to include in your software image.

If you had selected to create a grid infrastructure image, you would select the desired grid infrastructure release and choose a release update, a proactive bundle patch, or one-off patches to include in your software image. You can also upload an Oracle Grid Infrastructure home inventory to use as the required patches to include in your software image.

You can create your custom software images with any Oracle Database or Grid Infrastructure software version and update that is supported in the Oracle Cloud. Click on the Create Software Image to complete the image creation.

Once the image has been successfully created, you will see your custom images listed on the software images screen. From here, you can easily identify the image type, the service the image is for, and the image version.

At this point in our journey, we should have created the Exadata infrastructure resource and the VM cluster resource. We can now continue on to create Database Homes and databases. To create a Database Home, navigate to the Exadata VM cluster where you will be adding the database phone.

Under Resources, click Database Homes. Then click Create Database Home to launch the Create Database Home configuration screen. In the Create Database Home screen, enter the Database Home display name. And select a database image.

Note that you can change the image from the default by selecting the Change Database Image button. For the database image, you can select an Oracle-provided database software image or use a custom database software image. Click on the Create button to proceed.

Now let's take a look at how to create a database on the Exadata Database Service. Now that we have reviewed how to create a custom software image and Database Home, let's move on to creating a database.

From the Exadata VM Cluster details page, click on the Create Database button. This will launch the Create Database screen where we can enter the basic information for the database, such as the database name and the desired database version. We can even provide an optional pluggable database name.

Next, we will specify the Database Home that should be used to create our database with. Note that you can select an existing Database Home, such as the one we just created, or create a new Database Home.

Provide the administrator credentials for the CIS user, which will be created with the password you supply. Next, you will configure the details for the database backups, such as selecting the backup destination and defining the desired backup schedule.

In the advanced options section, you can choose if you want to use Oracle-managed keys or customer-managed keys. With Oracle-managed keys, data is encrypted with an encryption key that Oracle maintains. With customer-managed keys, data is encrypted using the master encryption key from the vault service you select.

Now, simply click on the Create button to start the database creation workflow. If you are creating a database on the Exadata Database, service on Cloud@Customer, you can integrate your on-premise Oracle Key Vault, known as OKV, with the Exadata Cloud@Customer to secure your critical data on-premises.

Oracle Key Vault integration enables you to take complete control of your encryption keys and store them securely on an external, centralized key management device. OKV is optimized for Oracle wallets, Java keystores, and Oracle Advanced Security, which includes Transparent Data Encryption master keys. For more information on Oracle Key Vault, refer to the Oracle Cloud Infrastructure training courses and documentation.

Now that we have created databases, let's take a look at how to manage the pluggable databases, known as PDBs, that reside within the container database. With the Exadata Database Service, when you create a database, a single container database, known as a CDB, with a single pluggable database, known as a PDB, is created by default.

After the database is created, you can use the console or APIs to conduct pluggable database lifecycle management tasks. From within the existing container database, you can create additional pluggable databases.

To create another pluggable database in an existing container database, from the Database Details page, click on the Create Pluggable Database button, which will launch the Create Pluggable Database screen.

In the Create Pluggable Database screen that is displayed, enter your desired PDB name and the admin password. And click on the Create Pluggable Database button to proceed. After the new PDB is provisioned, it will be displayed in the list of available PDBs as shown.

From the Pluggable Database Details page, you can connect to individual PDBs directly via separate connection strings using either Easy Connect or Long Connect strings. You can also open the Performance Hub to check on the performance of your application cycles.

If you need another copy of your database, the Exadata Database Service makes it very easy to clone your pluggable database. If you click on the Clone tab on the Pluggable Database Details page, it will launch the Clone Pluggable Database page where you can enter the information required to build the clone you need.

From the Clone Pluggable Database page, you are presented with options for the type of clone that you can create to meet your requirements. You can create a local pluggable database clone, which creates a pluggable database within the same container database as the source PDB.

You can also create a remote pluggable database cloud, which creates a pluggable database in a different container database than the source pluggable database resides in. Remote clouds can be created across VM clusters in the same availability domain, enabling the creation of replicas of PDBs on-demand.

You can also create a refreshable PDB clone, which creates a PDB in a different container database than the source PDB, whose data can easily be refreshed using the refresh function found under the More Actions tab.

Once you have selected the type of clone you need, select the clone destination. And enter the remaining information requested to configure the new pluggable database. And click on the Clone Pluggable Database button to proceed with creating the clone.

The More Actions tab allows you to be able to conduct additional PDB, lifecycle, and operational tasks. From the More Actions tab, you can conduct tasks like start and stop of a PDB. Clicking on Start or stop from the More Actions tab launches the dialog box to confirm that you want to proceed with the action or cancel the operation requested.

You can also relocate and restore a pluggable database. To relocate a pluggable database, in the More Actions tab, choose Relocate, which will launch the Relocate Pluggable Database page. In the Relocate Pluggable Database page, enter the name of the destination DB system and the new PDB name and password. And click on Relocate Pluggable Database to proceed. Note that you can use the same PDB name or change it as part of the relocate PDB process.

To restore a PDB, in the More Actions tab of the PDB, you want to restore, choose restore. In the restored PDB dialog box that is displayed, choose if you want to restore the PDB to the latest backup or to a specific point in time. And then click Restore to proceed.

Deleting a PDB is also a simple operation that can be done from the More Actions tab by just choosing delete. And in the dialog box displayed, confirm that you want to proceed with the action or cancel the delete pluggable database requests.

Let's take a look at how to enable Data Guard protection for your Exadata Database Service. To ensure business continuity in the event of a disaster, you want to ensure that you implement a disaster recovery strategy.

Data Guard is a key database feature that provides real-time disaster protection. Should you lose your primary database or data center, you can fail your workloads over to a standby site maintained automatically by Data Guard.

With Data Guard, you can perform plan maintenance, database upgrades, and apply patches with minimal service interruption, allowing for the lowest downtime maintenance choice. Setting up Data Guard is a critical component of any maximum availability architecture solution.

Exadata Database Service makes it simple to enable Data Guard to support your standby or disaster recovery needs with a single API call or a few mouse clicks in the UI using the cloud automation. Simply select the pure VM cluster where your Data Guard will reside. The VM cluster can be in another region or availability domain.

Then select the Data Guard type that you need. Active Data Guard extends Oracle Data Guard capabilities by providing advanced features for data protection and availability, as well as offloading read-only workloads and fast incremental backups from a production database. Active Data Guard is included in the Oracle Exadata Database Service with a license included option.

When used in a hybrid configuration, active Data Guard must also be licensed for the on-premise system. For the Database Home, choose if you want to use an existing database phone or create a new Database Home.

For the standby, provide a database unique name and the standby database admin password, which must match the primary database admin password. And you're done. Click on the Enable Data Guard button to complete the workflow.

Data Guard enables you to change the role of a database. Oracle Data Guard supports two role-transition operations. The switchover operation is used for planned role reversal and enables you to switch the role of the primary database to one of the available standby databases.

The chosen standby database becomes the primary database. And the original primary database then becomes a standby database. This feature is used to reduce downtime from planned operations, such as OS or hardware maintenance and database software patching.

The failover operation is used for unplanned role reversal. You invoke a failover operation when a catastrophic failure occurs on the primary database. During a failover operation, the failed primary database is removed from the Data Guard environment. And a standby database assumes the primary database role.

You invoke the failover operation on the standby database that you want to failover to the primary role. You can also manually configure fast start failover, which allows Data Guard to automatically and quickly failover to a previously chosen synchronized standby database. The reinstate operation is used to return a failed database into service as a standby in the Oracle Data Guard Association after correcting the cause of the failure.

Here are some requirements that you should be aware of before enabling Data Guard. Both of the DB systems must be in the same compartment. If your primary and standby databases are in the same region, then both must use the same virtual cloud network.

If your primary and standby databases are in different regions, then you must peer the virtual cloud networks for each database. The database versions must be the same. Each database in a Data Guard association must have a unique name. This is set by the DB unique name value.

However, the primary and standby database can use the same database name value. Configure the security list, ingress, and egress rules for the subnets of both the DB systems in the Oracle Data Guard association to enable TCP traffic to be able to move between the applicable ports. Ensure that the rules you create are stateful, which is the default.

The minimum requirement for Oracle Data Guard to work is to enable egress for TCP traffic only for the SCAN listener port, which has a default value of 1521.

3.7 Exadata Database Lifecycle Management Use Managed Maintenance Tasks - Part 1

Let's dig a little bit more into the management responsibility matrix to understand what the user-managed maintenance tasks are for the Exadata Database Service. With the Exadata Database Service, managing maintenance updates is divided into two sections-- Oracle Managed Infrastructure Maintenance updates and User Managed Maintenance updates.

Let's take a look at the Oracle Managed Infrastructure Maintenance updates. Oracle is responsible for applying the quarterly infrastructure updates to all of the infrastructure components, including the physical compute servers and the root VM, the Exadata storage servers, the internal switches in the Exadata secure RDMA network fabric, the PDUs, ILOM interfaces, firmware, and any control plane components. This is referred to as Infrastructure Maintenance.

Maintaining a secure Exadata Database Service instance in the best working order requires that you perform the following user managed tasks regularly, such as patching the Oracle Grid Infrastructure software, patching the Oracle Database software on the Exadata Database VM, and updating the operating system on the Exadata Database VM. These tasks should be done quarterly, but not to exceed 180 days from the last applied patch.

To perform User Managed Maintenance operations on the Oracle Cloud infrastructure, you must be granted security access with the appropriate IAM policies. This access is required whether you're using the web console or the REST APIs with an SDK, command line interface, or other tools. If you get a message that you don't have permission or are unauthorized, verify your access with a tenancy administrator.

In this section of the course, we will be focusing on the User Managed Maintenance tasks. Before proceeding with applying updates or patches to the Exadata Database Service, let's take a look at the best practices that you should follow to avoid patching failures. You should always ensure to backup your database before you apply any updates to the DB Home.

You should consider moving the database to a new Database Home if possible, instead of updating the existing Database Home. Since the Grid Infrastructure patch level determines the highest database level that the cluster can support, in short, to update the Grid Infrastructure before updating or creating a Database Home with a new version.

To ensure that your environment is ready for the planned update operation, it is suggested to run the precheck operation prior to the maintenance window to identify and resolve any issues that would prevent a successful update. You should also ensure that all servers and database instances are up and running, and that /u01, which contains the Grid Infrastructure Home directory, has at least 15 gigabytes of free space before conducting a Grid Infrastructure update.

You should also ensure that /u02, which contains the Database Home directory, has at least 15 gigabytes of free space before conducting a Database Home update. Oracle recommends that you use the OCI management interfaces to perform the update and patching operations.

In this section, we will review how to identify the current patch level. And how to find the available updates for your VM cluster components and Database Homes. We will also explore the views that allow us to view the history of the patch operations that have been applied to the Exadata Database Service VM cluster components and the Database Homes.

To find the current patch level of your Exadata Cloud VM cluster, navigate to the VM Cluster Details page. In the Version section of the page, you are able to see the details about your current patch levels, for the Exadata image and the Grid Infrastructure.

In the Database section of the page, you can see the version of the Database Home in use for each database that has been created. Next, to Updates Available in the Version section, you can click on the View Updates link to go to the Updates page, where you can see the available patch updates that can be applied to your VM cluster components.

The Updates page allows you to get a sense for the type of information that you can find about the available updates for the Exadata VM cluster components. The Updates page is organized into three sections reflecting the VM cluster components of the operating system, the Grid Infrastructure, and the Database Home.

In the Exadata VM cluster OS section of the Updates page, you can find the available updates listed for the Exadata image components, which contain the OS updates. Each update listed contains an Update Description field, which provides you with a high-level description of the available update, and a Version field that specifies the exact patch version. You are also provided with a Release Date field, which shows the date timestamp of when the update was released, and a timestamp of the last successful precheck that has been run for the patch.

In the Exadata VM cluster Grid Infrastructure section of the Updates page, you can find the available updates listed for the Grid Infrastructure software components. Each update listed contains similar information to what was described for the operating system updates.

For the Grid Infrastructure updates, the Type field tells you if the update is a patch or an upgrade. In the Database Home section of the Updates page, you can find the standard Oracle provided database software images, and your organization's custom database software images.

Each update listed contains a patch description field, which provides you with a high-level description of the available update, along with an update version fields, and a release date field, which shows the date timestamp of when the update was released. Note, if you have multiple Database Homes, you can use the scope selector on the left rail to list the patches that are specific to that Database Home.

The Exadata Database Service also provides an update history view that allows you to review the previous patching operations that occurred on your VM cluster components and the database column. From the Update History page, you will be able to see what patches have been prechecked or applied, and the timestamp of each action.

Each patch history entry represents an attempted patch operation and indicates whether the operation was successful or failed. You can retry a failed patch operation. Note that repeating an operation will cause a new patch history entry to be created. Also note that the patch history views in the console do not show updates that were applied by using command line tools such as dbaascli.

Let's take a look at how to update the guest VM OS image on your Exadata VM cluster. The Exadata image update allows you to update the guest VM OS image on your Exadata VM cluster nodes in an automated manner from the OCI Console and APIs. This automated feature simplifies and speeds up VM cluster patching, making patching less error prone and eliminates the need to use Patch Manager.

The Exadata image contains the virtual machine OS updates. The OS update process provides a precheck function to ensure that the virtual machines are ready for you to apply the OS update. The update process also provides an option to roll back in case any issues are encountered during the update process.

Minor and major version updates to operating system image versions can be done from the UI and the REST APIs. To ensure that the Exadata Database Service remains secure, only the four latest Exadata image minor version updates are made available in the console.

Let's take a look at what it takes to run the precheck and apply process for the Exadata image update. You can use the Updates page in the OCI Console or the REST API to find and apply the available Exadata VM OS image updates. From the list of available updates, choose the desired OS update version you would like to apply.

From the Actions icon located on the right of the row of the patch you want to apply, you can select to run a Precheck or Apply the Exadata OS image update. The Run Precheck operation is used to validate system readiness to accept the update. The Apply Exadata OS image update operation also performs a precheck and then applies the update if the precheck passes.

Oracle highly recommends running the precheck operation in advance of your maintenance window to identify and resolve any issues that would prevent a successful update and to validate system readiness for the patching. If the precheck is successful, then you're ready to proceed with the Apply Exadata OS image update process, which brings up the Apply Exadata Image Update Confirmation screen.

Confirm the name of the VM cluster you're trying to update the OS for is correct, and also confirm that the correct target version for the OS update is shown. To start the OS version update process, click the Apply Exadata OS Image Update button.

In this section, we will explore finding and applying available updates for the Oracle Grid Infrastructure software on a VM cluster in the Exadata Database Service. Let's look at how to patch the Oracle Grid Infrastructure on a VM cluster. You can use the Updates page in the OCI Console or the REST API to find and apply the available GI patches.

From the list of available updates, choose the desired Grid Infrastructure version to apply. From the Actions icon located on the right of the row of the patch you want to apply, you can select to run a precheck or to apply the Grid Infrastructure patch. The run precheck operation is used to validate system readiness to accept the update. Again, the Apply Grid Infrastructure Patch operation also performs a precheck and then applies the update if the precheck passes.

Oracle recommends running the precheck operation in advance of your maintenance window to identify and resolve any issues that would prevent a successful update, and to validate system readiness for patching. The GI update is performed in a rolling fashion, with only one node being updated at a time. Note that the database instance in the VM undergoing Grid Infrastructure update will not be available while that node goes through the update cycle.

3.8 Exadata Database Lifecycle Management Use Managed Maintenance Tasks - Part 2
In this section, we will discuss updating the Oracle Database Home software in an Exadata VM cluster on the Exadata Database Service. To update an Oracle Database Home, customers can use an Oracle provided database software image or a custom database software image.

To update an Oracle Database Home, there are two paths to choose from. Option one, update an existing Database Home to the desired patch level, which updates all of the databases using the Database Home. Option two, create a new Database Home with the desired patch level and move a database to the new Database Home. Note that using the move to another home function is the quickest way to patch a database and reduce downtime.

You can use the Updates page in the OCI Console or the REST API to find and apply the available Database Home updates. From the list of available updates, choose the desired Oracle Database Home patch version from an Oracle standard or custom database software image you would like to apply. From the Actions icon located on the right of the row of the patch you want to apply, you can select to Run a Precheck or to Apply the Database Home patch. The Run Precheck operation is used to validate system readiness to accept the update.

The Apply operation also performs a precheck and then applies the update if the precheck passes. Oracle highly recommends running the precheck operation in advance of your maintenance window to identify and resolve any issues that would prevent a successful update, and to validate system readiness for patching.

Database Home updates are delivered in a rolling manner across the RAC database instances in the VM cluster. This patching procedure updates the Oracle Database software for all databases located in the Database Home. To patch an individual database, you can move it to another Database Home that uses the desired Oracle Database image.

Moving a database to a new DB Home with the desired patch level is the preferred approach for patching your database, and is quicker and less disruptive, since you're only updating a single database instead of all the databases in a Database Home.

To update a database using a custom database software image, there are two paths to choose from. Option one, create a Database Home with custom database software image and move the database individually to the new Database Home. Option two, update the Database Home with custom database software image. This option causes all of the databases using the Database Home to be updated at the same time.

Let's take a look at how to move an existing database to a new Database Home. Moving an existing database to a new Database Home is the easiest way to patch a database. From the Database Details page of the database you want to move to a new home, click on the Move to Another Database Home tab on the Database Details page. This will open the Move Database screen.

For the target Database Home, select the target Database Home that contains the desired patch level, and click on the Move Database button to proceed. Note that while the database is being moved, the database and the source and target Database Home status displays as updating. If the operation is unsuccessful, the status of the database is displayed as failed, and the Database Home field provides information about the reason for the failure.

Wherever possible, Grid Infrastructure and Database Home maintenance is done in a manner that preserves the service availability throughout the patching process. Let's review the impact of performing update operations on the Grid Infrastructure and Database Home. During the update operation on the Grid Infrastructure and Database Home, you can achieve zero database service downtime with RAC database rolling updates.

During the update operation, the maximum database compute performance and throughput are temporarily reduced during the window of time that the RAC database instances are being restarted. To achieve zero downtime applications, follow the Exadata Cloud maximum availability architecture best practices documentation for achieving continuous availability for applications.

Now that we have reviewed how to patch the Grid Infrastructure and Database Home, let's review how to upgrade the Grid Infrastructure. This topic describes how to upgrade the Oracle Grid Infrastructure on an Exadata Cloud VM cluster using the Oracle Cloud Infrastructure Console, or APIs.

Upgrading the GI allows you to provision the Oracle Database Homes and databases that use the most current Oracle Database software version. Upgrading the Oracle Grid Infrastructure on a VM cluster involves upgrading the GI software on all of the compute nodes in the VM cluster. The GI upgrade is performed in a rolling fashion, with only one node being upgraded at a time.

Database instances in the VM undergoing Grid Infrastructure upgrade will not be available for workloads. You can monitor the progress of the GI upgrade operation by viewing the associated work requests. Oracle recommends running an upgrade precheck to identify and resolve any issues that would prevent a successful upgrade prior to the maintenance window.

Note that the GI upgrade feature is not available if you have an Exadata infrastructure maintenance operation that is scheduled to start within 24 hours of the upgrade operation. Also note that the following Data Guard operations are not allowed on the VM cluster that is undergoing a GI upgrade.

You cannot enable Data Guard, conduct a switchover, failover to the database using the VM cluster undergoing maintenance, perform management operations such as starting, stopping, or rebooting nodes, scaling CPUs, provisioning, or managing Database Homes or databases, restoring a database, or editing the IORM settings.

Now that we've reviewed how to upgrade our Grid Infrastructure, let's review how to upgrade the database. To prepare for upgrading a database, we should start by backing up the database and testing the new database software version on a test system before you upgrade your production database. You should then run an upgrade precheck operation for the database before the upgrade maintenance window, so that you can discover and fix any issues before the time you have planned to perform the database upgrade.

Before starting the database upgrade process, you must create an Oracle Database Home that contains the target database software version to be used for the database upgrade. You can use Oracle published software images or a custom database software image based on your patching requirements to create the Database Home. You also must ensure that all of the pluggable databases in the container database that is being upgraded can be opened.

Pluggable databases that cannot be opened during the upgrade process can cause an upgrade failure. Since an upgrade operation cannot take place while an automatic backup operation is running, Oracle recommends that you disable automatic backups and perform an on-demand full backup before you kick off the upgrade operation.

Note that after the database upgrade, you cannot use automatic backups that were taken prior to the upgrade to restore the database to an earlier point in time. During the database upgrade process, the following steps are automatically performed. An automatic precheck is conducted. This allows the system to identify issues needing mitigation and to stop the upgrade operation.

A guaranteed restore point is set, enabling it to perform a flashback in the event of an upgrade failure. The database is moved to a user specified Oracle Database Home that uses the desired target software version. The Database Upgrade Assistant software is then utilized to perform the upgrade.

Let's take a quick look at how we conduct the database upgrade process. To upgrade a database, start by navigating to the Database Details page for the database you want to upgrade. On the Database Details page in the database version section, you can see the current Database Home and database version.

To upgrade the database, select the More Actions tab and choose the upgrade option. After selecting the Upgrade Database action, enter the following information into the Upgrade Database screen. Select the Oracle Database version that you want to upgrade to. Then select the target Database Home that you want to use for the upgrade. Then run the precheck function and troubleshoot any issues. Once the precheck passes, click on the Upgrade Database button to proceed.

If your database uses Data Guard, you can upgrade the primary or the standby first. Standby first patching and upgrades are recommended. Upgrading a primary or standby will disable redo apply during the upgrade operation. Oracle recommends checking the redo apply and open mode configuration after the upgrade process.

In this lesson, you should have learned how to create custom database and Grid Infrastructure images, create a Database Home, create a database, perform pluggable database management, enable Data Guard, and perform user managed maintenance updates and upgrades. Thank you for watching.

4. Autonomous Database & Tools

4.1 Describe Autonomous Database Architecture and Integration

Welcome to Oracle University's course on getting started with Autonomous Database. My name is Kamryn Vinson. In this lesson, we are going to focus on architectural components and key features. After our time together is up, you should be able to define the architectural components of Autonomous Database and its key features.

So let's dive right in. So what does Oracle Autonomous Database, or ADP, do to help our customers? The biggest benefit is the massive decrease in time spent managing databases. ADB eliminates the complexity of operating and securing Oracle Database while giving customers the highest level of performance, scalability, and availability. availability.

Administering a mission-critical database is traditionally very expensive because it requires manual provisioning, securing, monitoring, patching, backing up, upgrading and recovering, troubleshooting, testing, and tuning of a complex, highly available scale-out deployment with disaster recovery protection.

The extensive automation provided by the Autonomous Database dramatically simplifies these tasks, reducing administration costs by up to 80%. New application development offer suffers for many months of delays waiting for database provisioning, testing, and tuning. With the Autonomous Database, new applications don't wait at all, saving tens of thousands of dollars per application and enabling much faster innovation.

Incremental changes around the edges aren't enough to answer these challenges. Even moving the entire enterprise computing estate to the cloud is only a partial solution, because this just shifts many of the same management difficulties to a different infrastructure. Instead, IT leaders need to strike at the heart of the matter, rethink data management for a cloud-native world. They need data management that can manage on its own the core activities of performance, security, and availability.

This new approach to data management has to make data appear as if it were stored in a shape and made available at a scale ideal for the current workload, without having to predict it ahead of time. To deliver data management as a service like this, IT leaders need a new kind of database.

Downtime for critical applications causes severe financial and reputational damage. IT leaders must be sure that the cloud architecture they adopt handles every threat to availability, from software and hardware failures and maintenance to natural disasters. The Autonomous Database is more reliable than a manually operated database. At startup, it automatically establishes a triple mirrored scale-out configuration in one regional cloud data center, with an optional full standby copy in another region.

The Autonomous Database automatically recovers from any physical failures, whether at the server or data center level. The Autonomous Databases fault-tolerant scale-out cluster that works transparently for both OLTP and analytic workloads makes it uniquely capable of running mission-critical workloads.

We've touched on ADB features, let's talk about them a bit more. Like an autonomous car. Oracle Autonomous Database provides a level of performance and reliability that manually-managed databases can't deliver. It can provision, secure, and manage in a way manual databases can't. Let's look at provision.

Oracle quickly deploys a mission-critical database on our Exadata Infrastructure Cloud, a platform optimized for Oracle Database. The database is created as a rack cluster, so it can scale out seamlessly, but also protect the database in case of a server failure. In order to protect the system against a complete failure, we also create a standby system using Autonomous Data Guard in the same region or a different region in the cloud to protect against regional failures.

Once provisioned, the system needs to be secured by default. We encrypt all the data in the database, but encrypting the data at REST is not enough. All connections to the database are fully encrypted too. To ensure systems are not exposed to any security vulnerabilities, we also apply all security patches as soon as they are available.

Manage the system. So Oracle applies all the OS, storage, VM, and database patches, plus we diagnose all problems. We use machine learning to help us detect and diagnose errors. ADB detects and protects from system failures and user errors automatically, and provides failover to standby databases with 0 data loss. It provisions highly available databases, configures and tunes for specific workloads, and scales compute resources when needed, all done automatically to enable true pay-per-use.

Without human detection. ADB uses indexes, optimizer statistics, and plans, and optimize data formats to run the workloads and achieve the best performance. ADB ushers in a new era in data management, radically changing the way companies deal with database and data management in general. This new class of databases is based on the world's most secure, available, performant, and reliable database, Exadata. This new era is available both in the public cloud and on premises. The largest enterprises and governments already run their mission-critical workloads with Oracle Database on Exadata. ADB offers them the safest transition to the cloud.

4.2 Describe the different ADB offerings and license types

Welcome to Oracle University's course on Autonomous Database. My name is Kamryn Vinson. In this lesson, we are going to focus on Autonomous Database licensing.

Oracle is offering an ECPU based pricing metric for the Autonomous Database. What are ECPUs? They're an abstract compute metric of cloud resources that is performance-based and are elastically allocated from ADB's shared pool of Exadata Database and storage servers. CPUs are not based on physical cores or threads, and is the default compute model for all new ADB instances.

What are the benefits of ECPUs? ECPUs provide a durable pricing metric that is not tied to the exact make, model, and clock speed of the underlying processor, which avoids pricing complexity as hardware systems change. They provide finer granularity system sizing and auto scaling for ADW. Customers also benefit from reduced storage costs for ADW instances using ECPU.

Here are some key points when discussing Autonomous Database pricing metrics. ECPU pricing is available for ADW and ATP with a minimum configuration of two ECPUs. Beyond two ECPUs, scaling is available in single step increments for maximum cost efficiency.

Licensing for ECPUs can be Pay-As-You-Go, BYOL, which is Bring-Your-Own-License for existing Oracle Database customers. You also choose an Oracle Database Edition and the edition you select is based on the licenses you bring to Autonomous Database and changes the maximum value that you can select for the number of ECPUs.

When purchasing ADW or ATP, there are two pricing models. Your customer can use the universal credit model or UCM, which includes all database options. Or if the customer owns their Oracle Database licenses, they can use Bring-Your-Own-License or BYOL to reduce their cost.

Advantages. When your customer obtains ADB licenses through the universal credit model, they can access all capabilities with standard pricing. If your customer leverages their existing Oracle licenses, they can pay a fraction of the UCM standard pricing.

Licensing. The standard UCM pricing does not require the customer to separately own Oracle licenses. BYOL pricing for provisioning instances with more than 64 ECPUs requires existing real application clusters licenses, in addition to the Oracle Enterprise Edition licenses.

BYOL pricing for standard edition restricts the number of ECPUs you can use to a maximum of 32 ECPUs, with or without compute autoscaling enabled.

99.995% service level agreement. If your customer requires a 99.995% SLA and is using Bring-Your-Own-License model, then they would need. Oracle Enterprise Edition, Real Application Clusters, and Active Data Guard to achieve that level of service.

Oracle universal credit pricing is based on two distinct payment structures. The first is a Pay-As-You-Go unit cost structure that allows customers to quickly provision services and only pay for what they use. Unit costs are based on the type of service provided and includes the use of certain licenses, such as Oracle Database or database options, for example.

The second payment structure is called annual universal credits. It's like a prepaid gift card, and the credits represent the amount of money that a customer is committed to spend per year on OCI. It's a one year minimum commitment that enables customers to have the flexibility to use any OCI and platform services at any time in any region.

Customers can commit to an amount of Oracle annual universal credits that can be applied towards the future usage of eligible Oracle IaaS and PaaS cloud services. Customers can also receive a field discount depending on how much they commit to. It's a very straightforward way to do things and we don't touch anything that's running.

You can convert from BYOL to license required and also from license required to BYOL. Here's an example of a database that's running with Bring-Your-Own-License. Let's see how to toggle the license type from Bring-Your-Own-License to license included.

Now click the More Actions dropdown list. Then select Update License and Oracle Database edition. We're setting it to be subscribe to an Enterprise edition license included. Once we click Save.

We're setting it to be subscribed to a regular Pay-As-You-Go license included billing model by toggling the radio button to disable BYOL. Once we click Save. The license type changes to license included, and this information is removed from the console. So it's possible to toggle between both types of licenses dynamically and on the fly without incurring any downtime.

Thanks for watching this lesson on Oracle Autonomous Database licensing.

4.3 Create Autonomous Database Serverless Instances - Auto Scaling

Welcome to Oracle University's lesson on Auto Scaling with Autonomous Database. My name is Kamryn Vinson. Let's get started.

In this session, we're going to have a quick demonstration on scaling the Autonomous Database. Now, before we get started, just recall that the Autonomous Database allows you to scale up and scale down both ECPUs or compute as well as storage. It allows you to do this independently of the other. So you can scale up your CPUs without touching your storage and scale it back down. And you can do the same with your storage.

In addition to that, you can also set up auto scaling. So the database will automatically, when it detects the need, scale up to three times the base level number of CPUs that you have allocated or provisioned for the Autonomous Database. Auto scaling is not available for an always-free database, but it is enabled by default for other tiered environments. Changing the setting does not require downtime, so this can also be set dynamically.

So let's take a quick demonstration on how to do this. So you will see on your database console page the first thing you want to do is choose the database that you wish to scale. So we're going to pick ATP demo.

And when we navigate to ATP demo, you will see that the database has an ECPU count set at 2 at present. And you'll see our storage is set at 1 terabyte. In addition to that, we can also see that compute autoscaling is enabled for this database. So if it hits our requirement or workload requirement that requires additional capacity, it can increase the number of CPUs up to 3 or anywhere in between. So we can click the Manage Resource allocation button on the top of our screen.

And when we click that, it will bring up a pop-up, and you can see right away that we have compute auto scaling already enabled here. And we could disable it if we wish to turn that off. We can also see our ECPU count is set at 2 and our storage is set at 1 terabyte. We can go ahead and change our settings for either one of those independently of the other. So once you increase the ECPU count, we've set it up to 6 here, we simply click Apply.

And when we navigate back to our database console page, we'll see that the tile now says it's scaling in progress. Now, remember, even though we have done this through the manual process, the database is still open, it's still running, and it's still allowing users and sessions to connect and perform their normal workload. You'll see that our life cycle state is listed as scaling in progress. So if you're using a REST API or something to check the status of the database, that's the messaging you'll get. And you'll see the ECPU count is still set at two while this process is taking place.

When this has changed, when it has now increased our ECPU count to 6, you'll see the tile now changes to say that it's now available. Our life cycle state says it's available. And you can see that our ECPU count has now been increased to 6. So it's a very simple operation.

So to wrap things up, the Autonomous Database allows you to very easily scale both ECPU and storage independently of one another. You can also automatically set the system up or set the system up to automatically scale up to three times your base limit. So in our case, when we increased our database ECPU count to 6, having auto scale enabled, means that it can indeed go up to 18 ECPUs should the need arise. I hope you enjoyed this demonstration. Thank you.

4.4 Create Autonomous Database Serverless Instances - Provision ADB

Welcome to Oracle University's lesson on provisioning an Oracle Autonomous Database. My name is Kamryn Vinson. Let's get started.

In this session, we're going to take a look at provisioning an Autonomous Database Serverless service. So the Oracle Cloud automates the process of provisioning an Autonomous Database, and it automatically provisions for you a highly scalable, a highly secure, and a highly available database simply out of the box.

Provisioning the database involves very few steps, but it's important to understand the components underneath that are being provisioned. For example, in the background, a database 19C pluggable database is being added to a container database that's part of your cloud infrastructure.

When you provisioned an Autonomous Database Serverless service, you're actually provisioning a database on a private software defined network that you do not have to recreate, and your provisioning a fully encrypted database, a database that's running on a scalable RAC cluster that's also running on top of exadata. And processes such as backing up is already pre-configured for you. So there are no additional steps to go ahead and set up your backups. You simply choose the number of ECPUs that you need for your database, and then you choose the size and increments of terabytes.

So let's take a look. When you log into your Cloud Console on the main landing pad, you'll see that there's a tile there to automatically provision either a data warehouse or a transaction processing database. Or you can navigate by clicking the hamburger menu in the top left-hand corner of the screen. After clicking the hamburger menu, click Oracle Database. The various Autonomous Databases that are available to you appear. In our case, we can click either Autonomous Data Warehouse or Autonomous Transaction Processing.

And it will bring us to a screen where we can create our new Autonomous Database or provision an Autonomous Database. It will allow us to see existing Autonomous databases that we've already created. So when we get to this screen, you'll see the first thing there is a button that says Create Autonomous Database.

On the left side of the screen, you'll see that our compartment is listed. We could choose to change compartments if we wish to see any services that we've created in a different compartment. We can also see the workload type. This is where we would see either data warehouse Autonomous Database or transaction processing or Autonomous JSON Database and so on.

And below that, we can filter the databases that are listed for us. In our case, we don't have any databases listed, so we're not going to do anything there. However, if we had a large number of databases provisioned, we could filter up to 20 different metrics to filter out our databases that are being listed. So we could see databases that are available, databases that have stopped, databases that are being backed up. We could filter only those databases.

In this case, we've just got it listed as any type. So if we click that button to click create an Autonomous Database. That will go ahead and start the provisioning process. The first things we'll see on the screen is the compartment in which we're going to provision our database, and it will allow us to change our compartment if we wish to put our Autonomous Database in a different location. It will also let us set the display name.

So this is a simple name that we can recognize and that we may choose to name our Autonomous Database Service. We could set this to something that maps to our application. In addition to that, we see our database name listed. And our database name is a little bit more unique. It's provided as a default out of the box, but there's nothing preventing you from setting that to be the same as your display name.

Once we've entered those values and we scroll down, we'll see we can choose the workload type. We could use data warehouse, transaction processing, and Autonomous JSON, an APEX database, and so on. In this case, we're just going to choose transaction processing. And then we're choosing Serverless. And then we can choose the database version 19c in this case. And we can choose the number of ECPUs and storage size in terabytes.

You'll notice that autoscaling is already pre checked. The check box has a check mark in it, so autoscaling will be enabled by default. In the case of an always free account, that would not be pre-enabled, and that is something you would have to look at changing yourself.

Now, as we scroll down, your data is automatically backed up. Specify the backup retention policy. We then provide the password, an Oracle secure password for our admin account, which is the account to manage the instance that we will be creating. And then we choose our network type. We can choose a publicly routable IP address, which is secure access from everywhere, or we could choose to provision an Autonomous Database on a private subnet somewhere.

And we could choose that in either one of these options. This is not fixed. If you wish to change this later, you can always relocate your Autonomous Database to a different resource and compartment that could be mapped to have a private network to deploy your Autonomous Database. So changing this is not a problem after the fact.

Also, you'll see that we can configure access control rules. And this is where we may choose to have a publicly routable access from anywhere setting for our Autonomous Database. But we want to restrict the IP addresses that are allowed to access it that are coming in to access our Autonomous Database.

Now, this setting can also be changed after the fact if you wish to add or remove IP addresses or CIDR routes and so on from accessing your Autonomous Database. Finally, you'll see that there's an option to choose the licensing type. If you're bringing a license from on-premise and applying it to your cloud footprint, you could choose the BYOL license model, or you could choose license included, which is where you would pay for the services as they're being consumed.

Once we've got all of that set up, we simply click Create Autonomous Database. Now, at this point, you will be taken back to the Autonomous Database details page. And the database will appear in a provisioning state. Once the database has been provisioned, it's a couple of minutes later, we'll see that the database is listed in an available state. And at this point, the database is provisioned and running and open and ready to be connected to and accessed by us.

So to wrap things up, you have seen that provisioning an Autonomous Database is extremely fast and very, very easy. There is no need to specify any tablespaces to specify any file system requirements or operating system requirements or initialization parameters and so on. It was a very simple process to follow. And in doing so, we've created a self-contained, self-secure, self-managing, scalable Autonomous Database that we're ready to connect to and load our data. Thanks for watching.

4.5 Create Autonomous Database Serverless Instances - Start and Stop ADB
Welcome to Oracle University's lesson on how to start and stop an Autonomous Database. My name is Kamryn Vinson. Let's get started.

In this lesson, we're going to take a look at starting and stopping an Autonomous Database Instance. The Autonomous Database allows you to start your instance very rapidly on demand. It also allows you to stop your instance on demand as well, to conserve resources and to pause billing.

Do be aware that when you do pause billing, you will not be charged for any CPU cycles because your instance will be stopped. However, you will still be incurring your monthly billing for your storage.

So when we navigate to our database Cloud console, we can see here that our database has been stopped. Now remember the database is available in that the data is still there. Everything is still persistent, the instance itself is just paused. At this point in time, we can go ahead and click the More Actions dropdown.

And we will see that there is a start option there. We'll click the Start button, and that will bring up a pop up. And then we confirm that we indeed wish to start this instance. We click the Start button. And the database will then change its status from stopped to starting, and we'll also see this during our lifecycle state as well.

So when the instance has been started, the tile will now change to a green color and it will say available. And it's now available for sessions to connect and run their workloads on the instance. Now that the instance is actually up and running, we can click our More Actions button.

And we'll see that the first option that's displayed now is to actually stop the database. So the dropdown has changed and it now gives us the option to go ahead and pause or stop our database instance. So we click the Stop button.

Once again, it asks for confirmation to confirm that we do wish to indeed stop our database instance. And in doing so we click that button. And we'll see that our tile, when we return back to our database console, has now turned orange again and the status is listed as stopping. And we can also see the tile is listed as stopping.

At this point in time, if we navigate back to our higher level of our compartment, we can see that our database is listed with a state as stopped. You also define a schedule to automatically start and stop your Autonomous Database Instance. This allows you to reduce cost by scheduling shutdown periods for times when a system is not in use.

On the More Actions menu, select the AutoStart/Stop schedule. For each day you want to schedule, select the start time checkbox and the stop time checkbox, and select a start time and a stop time. The dropdown list allows you to schedule start times and stop times hourly, on the hour, specified in coordinated universal time UTC.

Selecting both a start time and a stop time is not required. You can select just a start time or just a stop time. Click Apply to apply the schedule. Then in the confirmation dialog, click Apply.

So to wrap things up, it's very easy to start and stop your database instance using the Database Cloud console. We provide the ability to just point and click to do this very effectively. Remember that you can also invoke REST services to perform those operations as well, should you wish to script or automate those operations.

Pausing a database instance from running can pause your billing for those CPU cycles, and it's just a case of restarting it to make that database available. So thank you for attending this session. I hope you enjoyed it.

4.6 Describe Autonomous Database Tools

Welcome to the Oracle University lesson on Oracle Autonomous Database tools. My name is Hope Fisher. Let's get started. In this lesson, we'll describe Oracle Autonomous Database tools. An Autonomous Database leverages AI and machine learning to provide full end-to-end automation for provisioning, security, updates, availability, performance, change management, and error prevention. In this respect, an Autonomous Database has specific characteristics.

It is self-driving. All database and infrastructure management, monitoring, and tuning processes are automated. DBA's can now focus on more important tasks, including data aggregation, modeling, processing, governance strategies, and helping developers use in-database features and functions with minimal changes to their application code. It's also self-securing. Built-in capabilities protect against both external attacks and malicious internal users. This helps eliminate concerns about cyber attacks on unpatched or unencrypted databases. And it's also self-repairing. This can prevent downtime, including unplanned maintenance. An Autonomous Database can require fewer than 2.5 minutes to downtime per month, including patching.

Database actions provides all of the tools in one location with easy access to SQL worksheet, a data modeler, APEX machine learning, REST, JSON, and others. These are just the development tools. Additional tabs provide a workspace for Data Studio, administration, downloads, monitoring, and other related services. Developer tools are available to you out of the box with Autonomous. This includes SQL Developer web, APEX, Oracle REST Data Services, and others. Let's dive in to learn more.

SQL Developer is built into Autonomous Database tools to run SQL, load data, and manage your database. APEX is the low-code environment that's built into the Oracle Database. And from database tools, you can get started with developing an APEX or access workspaces that you already have created.

Enabling REST on your data allows for using APIs and PL/SQL and application code here. In the REST tool, you can create custom REST services and provide database Management REST APIs. You want to work with JSON as JSON in your Oracle Database, you can create collections, load JSON, and browse documents in the JSON toolset.

SQLcl is a modern command line interface in your Oracle Database. This tool, as part of the Autonomous Database, has auto complete SQL syntax with command history when working with JSON, CSV, HTML, XML, and other actions. This has Liquibase schema lifecycle integration along with OCI and OSS integration.

The Data Studio is another set of tools for loading data, gaining data insights and analysis, provides a data catalog, and allows you to configure data sharing and see your data shares. The catalog lists your entities with details of each entity that you can drill into. You can search for Insights in your database on the entities in your catalog. Autonomous Database has AutoML, and we already heard about that in an earlier lesson. Here, you can work with machine learning notebooks.

This is a tool, part of the Database Actions adding to the analytics and insights in working with your data. Of course, you can do standalone machine learning on the data in Autonomous Database. Graph can be created and queried in the Oracle Database. And part of the Autonomous Database toolset is a way to visualize the graphs and see the connections between vertices in Graph Studio.

And since we can store spatial data with our relational data in the database, Spatial Studio, again, allows for visualization of this data.

Thanks for watching this lesson on Oracle Autonomous Database tools.

5. AI Innovations

5.1 Introduction to AI

Human intelligence is the intellectual capability of humans that allows us to learn new skills through observation and mental digestion, to think through and understand abstract concepts and apply reasoning, to communicate using a language and understand the nonverbal cues, such as facial recognition, tone variation, and body language. You can handle objections in real time, even in a complex setting. You can plan for short and long-term situations or projects. And of course, you can create music and art or invent something new like an original idea.

If you can replicate any of these human capabilities in machines, this is Artificial General Intelligence or AGI. So in other words, AGI can mimic human sensory and motor skills, performance, learning, and intelligence, and use these abilities to carry out complicated tasks without human intervention. When we apply AGI to solve problems with specific and narrow objectives, we call it Artificial Intelligence, or AI. So let's look at some examples.

AI is all around us. And you've probably interacted with AI even if you didn't realize it. Some examples of AI can be viewing an image or an object and identifying if that is an Apple or an orange. It could be examining an email and classifying it spam or not. It could be writing computer language code or predicting the price of an older car. So let's get into some more specifics of AI tasks in the nature of related data.

Machine learning, deep learning, and data science are all associated with AI, and it can be confusing to distinguish. So let's look at why we need AI and how it's important. AI is vital in today's world. And with the amount of data that's generated, it far exceeds the human ability to absorb, interpret, and actually make decisions based on that data. That's where AI comes in handy by enhancing the speed and effectiveness of human efforts.

So here are two major reasons why we need AI. Number one, we want to eliminate or reduce the amount of routine tasks. And businesses have a lot of routine tasks that need to be done in large numbers. So things like approving a credit card or a bank loan, processing an insurance claim, recommending products to customers are just some example of routine tasks that can be handled.

And second, we as humans need a smart friend, too, who can create stories and poems, designs, create code and music, and have humor just like us. So let's break it down further to explore some more benefits. Here are some AI domains and their examples.

We have language for language translation, vision like image classification, speech like text-to-speech, product recommendations that can help you cross sell products, anomaly detection, like detecting fraudulent transactions, learning by reward like self-driven cars. You have forecasting with weather forecasting and, of course, generating content like image from text. So we're going to be exploring some of these further in the next lesson. Thank you.

5.2 Select AI

Now we use Select AI because we use the natural language to generate and get the result sets. But here, we're using a feature of APEX to be able to work with the data and see the different filters we have, and being able to also look at it as a chart and pull back that information and see it in a different format. It's quick and easy to use. And we didn't even have to know the table names or the columns that we were looking for. Now let's show the SQL plan, where we can actually see the SQL that was generated with Select AI show SQL. And we see the tables and the joins that are part of the statement that have returned that result set.

Since we are demonstrating the app we created an APEX, you can also simply take it on the move. The same Select AI capabilities used the large language models to return the result sets or the SQL query about your data. As you can see from the questions that were being asked, Select AI takes the question in natural language and translates into SQL language. Using AI large language models, it breaks up the question, infers the details being asked, and formulates the SQL query.

Large language models are different, and there are models better at writing SQL by inferring intent and being loaded with the information about the objects in the database. Now you can review the query to verify the results. Developing apps with Select AI is simple. So you can use generative AI capabilities with existing applications and build new. It is future enabled because you can choose from an array of large language models and pick the best suited for your business.

There are also new models being trained. And you can even leverage models that you have fine tuned. Using Oracle Database, the security protects your data. Using Autonomous Select AI with OCI Generative AI, your data stays in the tenancy and not sent to another LLM provider or seen by other customers. Let's take a quick look under the covers to see how this works and how simple it is to query the database with natural language. Using a tool like SQL Developer to query the database, you see the statements that run against the database.

It is a standard SELECT statement followed by AI, and then your question. Instead of the from and where clause, it processes the Select AI statement just like other SQL statements. You can also use the command select AI showsql, and you will see the query that is created to return the results. You can keep refining your questions to get more details or other answers that you need from the data. One of the major design principles for Select AI is pluggability. And this is encapsulated in an AI profile.

Depending on your use case, you may want to use OCI Generative AI, Cohere, or OpenAI models, or even Llama. You can choose which provider and model to use. Then you specify schemas and tables, or views, to participate in the process. To configure the database, you can use one or more AI profiles with a large language model that best fits with your business. Specific schemas, tables, and/or views are to be included in the processing. Use dbms_cloud_ai PL/SQL package that is provided in the database.

To wrap up, let's put it all together. To generate SQL query, you can simply ask your database using Select AI. Find the top customer segments for George Clooney movies. The AI profile uses the large language models defined within the database metadata to produce the prompt from the question asked, and generates the SQL using the model and returns the result sets or SQL to a SQL tool, such as SQL Developer, or to an app. We have shown you how easy it is to build AI into your apps and use Oracle Autonomous Database with Select AI to get answers from your data using large language models.

This is secure at every layer to protect your data, and future enabled to use new available models or fine tune models that meet your business needs. I hope you enjoyed this lesson on Select AI and are looking forward to building AI-powered apps with the Oracle Database.

5.3 Vector Search

Hi, welcome to this lesson on AI Vector Search in Oracle Database 23ai. My name is Michelle Malcher. I'm a Director in Database Product Management. And I'm excited to walk you through how Oracle Database 23ai has built-in Vector Search capabilities. Let's dive in.

AI Vector Search is built into the Oracle Database 23ai. In this lesson, we're going to take a look at the components available in the database. In this course, we have been talking about large language models and generative AI capabilities and OCI Gen AI services. Now we're going to drill down into the Oracle Database 23ai with AI Vector Search and see how it powers Gen AI pipelines with your data and apps.

There are single-purpose vector databases, but Oracle Database 23ai converged database approach supports JSON, XML, graph, spatial, text, relational, and vectors for data-driven workloads. AI Vector Search is an innovative technology which can greatly enhance information retrieval by enabling efficient searches based on semantics across structured and unstructured data, as well as a critical component of the modern Gen AI systems.

The database has SQL support for database-native vector generation, VECTOR datatype for storing vector embeddings, SQL syntax and function to perform similarity search with ease, and approximate search index packaged and tuned for high performance and quality.

Oracle Database 23ai allows you to use models with API calls or load an ONNX model into the database. In this example, resnet_50 has been loaded into the database. Then you can pass the model you want to use into the VECTOR_EMBEDDING function along with the query image.

The function returns a vector that you can pass into your database. The database has already stored other embeddings of other images. And using the same VECTOR_EMBEDDING function and model, AI Vector Search can perform similarity searches to return the top matches of the images that look like the query image.

The VECTOR datatype has been added to create a table with vector columns, even with relational data. The dimension format can be included or you can use a simpler VECTOR specification without the dimension format. So why is this useful? Because as your models evolve and newer machine learning embedding models become available, your schema can stay the same.

Now, to do similarity searches, the VECTOR_DISTANCE function is used to determine the distance between the vectors. The smaller the distance between the vectors shows which entities are more similar. Distance metric can be stated to change from the default cosine and should be chosen based on the model that was used to generate the vector embeddings.

Now that we have tables with the vector embeddings and distance function, Vector Search is used in SQL statements to find the closest matches to a given query. In this example, we are looking to find the top 10 positions matching a candidate's resume that are in their preferred cities.

It combines applicant data, job data, and AI Vector Search in five lines of SQL. It is a single integrated solution, all data fully consistent, that is simple to implement. Vector Indexes are not just for performance but can also control the accuracy. Creating a Vector Index is the same as creating an index on other tables.

There are additional options with organization and distance. The organization of a vector is based if an index will fit in-memory or not using INMEMEORY NEIGHBOR GRAPH for in-memory, otherwise, use NEIGHBOR PARTITIONS. Just like VECTOR_DISTANCE function, the

DISTANCE clause is optional with the default of COSINE and should be used based on the model that was used to generate the embeddings.

TARGET ACCURACY clause is used to indicate the default accuracy the index should use for similarity search queries. You can also specify the target accuracy in a query and the indexes that are used with the approximate searches.

We can specify APPROXIMATE keyword in the FETCH clause, which indicates a similarity search using a Vector Index. When returning the top five customers, the accuracy, either defined in the index or query, provides the quality of the approximate results.

So if a target accuracy was set at 80, the approximate results of the top five has four out of five correct matches. Accuracy of 80%. If there is no Vector Index with this query, an exact search might also be used, or if the SQL Optimizer might also decide that an index access plan would be too expensive and uses exact.

We had a previous example when we were looking at the vector, search, and SQL. But similarity search over joins is a differentiator with Oracle Database 23ai. It's essential to be able to do these joins because the enterprise data is usually normalized and available to provide more data attributes.

The cost-based optimizer decides on the join choice and the vector index used for the performance of the queries. This query joins three tables-- Author, Books, and Pages. Each page in the Pages table has individually embedding and stored in the vector column.

The qVec is a text embedding that is being searched on the pages and the query, returning the top five books containing text similar to this query text, where the genre of the book is fiction and the author of the book comes from India.

The capability of AI Vector Search is that it can power a Gen AI pipeline. You have your data sources, which can be pretty much anything. It can be a database, a CSV file, even social media handle. You usually use a document loader to bring those documents in. Then you can transform those documents.

For example, you can summarize them, you can split them, you can generate vectors for them using an embedding model stored in the vector database. And then the user can do similarity searches on that or use a RAG design pattern to interact with the LLMs.

Using Oracle Database 23ai and AI Vector Search allows you to fully process the Gen AI pipeline. We also provide tight integration with third-party frameworks like LangChain and LamaIndex to enable developers to build more sophisticated applications.

AI Vector Search is seamlessly integrated with the Oracle Database 23ai for enterprise-grade performance and reliability. There is converged SQL processing to perform AI vector power similarity searches directly on your business data.

In a single query, it is simple to combine relational, document, spatial, graph, machine learning, and AI Vector Search. And there is efficient orchestration of Gen AI pipelines natively in the database or through integrated third-party frameworks. Thanks for joining me for this lesson and learning about Oracle AI vector search.

6. Data Lake, Data Warehouse & ML

6.1 Data Lakehouse on OCI

Hi, and welcome to Oracle University's module on understanding the data Lakehouse on OCI. My name is Greg. And I'll be walking you through today's lesson. Now, traditionally, when deciding how to best increase their data productivity and liquidity, companies often find themselves having to make a choice between leveraging a data lake or a data warehouse, each of them having their own benefits and drawbacks.

Now, companies no longer need to make that choice. Instead, they can look to a broader strategy that offers highly accurate machine learning capabilities, the flexibility of using open source services, and the superior data and analytics capabilities of the best-in-class Oracle database and data warehouse.

These capabilities are integrated with common identity, data integration, orchestration, and catalogued into a unified architecture, the Oracle Lakehouse. Oracle Lakehouse facilitates ease of data reuse and recombination, maximizing insights from your data and generating several other benefits, including pure cost savings, as well as improving the agility of your current data warehouse by easily extending with new metrics details or attributes, which help me better understand your customers, your processes, or your risks, all while using your existing applications.

Also, for those of you companies who haven't yet adopted Oracle Cloud Infrastructure, but instead have existing data lakes on AWS or Azure, if you still want to make that data available to the Oracle Autonomous Database, you can reach out to these data lakes using Oracle SQL. Here at OCI, we feel your experience would not be a productive one if you weren't allowed to use your choice of tools and applications, such as Analytics Cloud, Tableau, Looker, Notebook, REST, Python, and more.

This is the Oracle Data Lakehouse. It combines current data warehouse and data lake components with capabilities to also include external or third party data. This effectively eliminates data silos or having to manually move data between data warehouses and data lakes if you leverage both currently.

The five key elements of the Oracle Lakehouse are the data warehouse, the data lake for raw data, normally used for loading and staging data, managed open source services to support Spark, Hadoop, and Redis, data integration, moving data depending on use case, and data catalog, which maintains a complete view of the available data for discovery and governance.

With these elements, you can write the data once with any engine and analyze or even build machine learning modules from any of your current data. Using all data to innovate, this is the challenge, to include all of your data and use it to drive better more profitable business decisions. Some data is easy to access. But accessing all of your data and then correlating that data in a way that helps make decisions and drive better outcomes isn't easy.

So the opportunity we've identified here is harnessing the power of all that data and creating a competitive advantage from it. The challenge, again, is how. How can you run and maintain what you've got today efficiently, quickly, and most importantly, securely? And also extract all of that additional value that we all know your data has hidden away inside, especially with Oracle on Oracle technology, but not required, here's how you can best make use of that data within the Oracle Cloud infrastructure.

We have functions that move data from sources to outcomes. The process is taking the source, going through integrations, and connecting to different data. Once we've done this, traditionally, we looked at persistence, processing the data and storing it somewhere to pass along for analysis. This has connected and curated the data for outcomes.

The Oracle Lakehouse is a solution leveraging multiple tools and products to get the desired outcomes from this process. You can use existing data warehouses to start. And a data warehouse, especially the Converged Autonomous Database, allows for storing all types of data. This is for the relational structured data to store in an Oracle Autonomous Database or warehouse.

The Autonomous Data Warehouse is self managed, with better performance and efficiencies to help focus on the analysis and the outcomes of the data. The unstructured or raw data can be persisted in any data type in its current format within object storage. This can be within an existing data lake, for example. Object storage is an efficient manner to land data where it's needed.

Now that we have an understanding of the start of the Data Lakehouse, let's talk about data integration and analysis. Lakehouse provides for an all encompassing orchestration of integration and is allowing your choice of tools to keep your source of truth and compliance for your data, whether you decide to deploy Oracle Golden Gate, the premier data integration tool, Oracle Data Integration, helping you move data within the lake, or even an open source or third party tool, Lakehouse is, by design, flexible and meant to fit your specific needs.

Oracle Analytics Cloud is used to perform predictive analytics. And other third party tools can read into the data from the database APIs or using SQL. Oracle AI Service has machine learning models that will continue to work with the transactional systems and bring in other data types as well. OCI Data Science can harness all of the data for better business outcomes and fills in the tools for integration and analysis for the Oracle data Lakehouse.

Within the Autonomous Data Warehouse, we have transactional and dimensional query capabilities. But in our Lakehouse story, we're also very lucky to have products like MySQL HeatWave, the blazing fast in-memory query accelerator which increases MySQL performance by orders of magnitude for analytics and mixed workloads, all without any changes to your existing applications.

Really, no other cloud provider is going to give you that much choice in the data warehouse bucket and managed open source components. Not only are the options for all data types a part of the Lakehouse, but also the understanding and metadata management of the data sources themselves. The OCI data catalog captures whether you're building a schema, building a query from ADW, or building a table that you want to query from a Spark job. And all that data definition goes into the OCI data catalog. So wherever this data goes, you'll be able to access it.

The data catalog is the source of truth for Object Store metadata and can regularly harvest the information from the data sources. It also manages the business glossary, providing consistent terms and tags for your data. Discovery of data is a powerful search feature to discover new data sets entirely.

Even with all these capabilities, there are still more being added or enhanced over time. For example, now with OCI Data Flow, you have a serverless Spark service. You can build a Spark job that makes sense from some unstructured data and include it as a part of the Oracle Lakehouse. Enterprises are moving to Data Flow, because you can write, decode, and execute code, and focus on the application, because the challenging part of where this is running is handled through the service components of the Oracle Lakehouse.

And then, of course, as you put everything together into this architecture, the key thing is that you want to be able to write data once and then combine it with other previously written data, move it around, combine it here and there, and analyze it. So at the center of this entire picture, we have a way to store both structured and unstructured data. You have the Object Store for unstructured data and write your structured data to a relational database, perhaps MySQL or Oracle database.

And you can then leverage the Oracle data catalog to have a single way to understand and tag your data. Oracle Data Lakehouse is an open and collaborative approach. It stores all data in an order that's easy to understand and analyze through a variety of services, as well as AI tools. OCI can accelerate your solution development for your most common Data Lakehouse workloads.

You can easily get started from where you are today, and often, without writing any new code whatsoever. Within each path, we can work with you at Oracle to highlight the investments we've made that will help accelerate your own Lakehouse transformation. The Oracle Data Lakehouse is the premier solution for transforming data into better more profitable business decisions.

Remember, it's not just your architecture that's powerful. With Oracle Lakehouse, you can help combine the architecture, data sets, services, and tools across your entire technical landscape into something more valuable than just the sum of its parts. Thanks so much for joining me today on Oracle's module on understanding the Oracle Data Lakehouse within OCI. I hope you enjoyed it and learned something and look forward to seeing you again soon.

6.2 Oracle Machine Learning Overview

Welcome to Oracle University's course on Oracle machine learning with Autonomous Database. My name is Cameron, and we're so glad that you're here. In this course, you will learn about Oracle machine learning, also known as OML on Autonomous Database.

You will also learn about AutoML with Oracle machine learning for Python and the AutoML UI. Creating, running, and scheduling notebooks, as well as exploring the use of built-in visualizations using R and Python to prepare and explore database data, as well as build in database ML models, and deploy R in Python-based solutions, along with augmenting database functionality using third-party R and Python packages.

Monitoring your data and in-database models using OML's no code data monitoring and model monitoring UIs. You will also learn about administering OML. You can use skill checks included within each model of this course to validate your knowledge.

With the help of this course, you can take the associated using Oracle machine learning with Autonomous Database certification exam with confidence. Now let's get started.

To better understand the role Oracle machine learning plays in creating effective business solutions and how we can apply Oracle machine learning, let's take a look at a use case together.

El Tronics is an online e-commerce company specializing in selling electronic products with a customer base spread out around the globe. The company uses on-premises database servers to store data pertaining to their businesses in terms of its staff, its inventory, purchases, sales, marketing, and so on.

El Tronics management is looking for a Cloud-based solution for analytics to improve targeted sales and run promotional campaigns that are region-specific. Steve, the chief data scientist, plans to adopt Oracle machine learning and hires Jake to be their Cloud technology architect.

Jake and his team will work on leveraging the power of Oracle machine learning to help intelligently predict, analyze, and report on buying patterns based on regional and demographic features.

So what is Oracle machine learning? How about a brief overview? Machine learning is focused on enabling data science teams to add ML-based intelligence to applications and dashboards.

With AI and machine learning, we can use powerful algorithms and models to support our data analysis process. Our team needs to collaborate to obtain data from potentially many sources, devise creative solutions to business problems, and work with developers to deploy solutions in production applications, dashboards, and reports.

So as you can see, Jake is saying machine learning and AI are key to analytics in every industry. and Oracle machine learning provides a reliable, AI-driven environment that truly encapsulates the power of machine learning.

Let's take a deeper look at Oracle machine learning. At a high level, the machine learning process is fairly straightforward. You define your project, get your data into a form that's useful for building a machine learning model, build that model, and then you use it. That can mean generating predictions or looking at the patterns that the model found.

Finally, deploy your solution in some application or environment where you or your users can take advantage of it. In practice, the flow is not so linear. Problems arise. Perhaps the data isn't as good as you thought, or the algorithm you chose isn't giving good predictions. We'll go into this in more detail in the next section.

Enhanced performance and scalability is achieved in part by eliminating data movement for data-based data and providing algorithms that have been redesigned specifically for scalability and performance.

Next is simpler solution architecture and management, where we want to avoid requiring separately maintain analytical engines or tools to enhance data and model Governance. In database, machine learning also offers flexible architectures that support separate development, test, and production environments spanning the Cloud on-premises and hybrid environments.

And because of its SQL and REST interfaces, it's easy to integrate with the broader Oracle stack. Now, the third is that OML empowers a broader range of users with machine learning. It's readily available in the database from multiple language interfaces, including SQL, Python, and R, along with no code user interfaces.

Application developers will appreciate OML's REST APIs. Python and R users can also leverage third-party packages. To make even non-experts productive with machine learning, OML supports AutoML from a Python API in a no-code user interface.

OML supports data and model monitoring using REST endpoints in no-code user interfaces. The in-database algorithms also have built-in automation features like automatic data preparation, integrated text mining, and partition models.

All of these are not only productivity aids for data scientists, but make machine learning more accessible to a broader range of users. And lastly, it's worth noting the simple pricing structure. Machine learning capabilities are included in the core product at no additional cost with Autonomous Database.

The OML components on ADB are pre-provisioned and ready for use. For on-premises database users, available Oracle machine learning components are included with the database license.

The SQL component is immediately available, and the R and Python components require additional installation. So overall, the takeaway is that OML helps reduce cost and complexity while increasing productivity and access.

We often think of industry-specific vertical use cases, but there are surprisingly many use cases that cut across industries, what we might call horizontal use cases. Most enterprises have customers, products, equipment, and employees.

Here we see a range of use cases that we could apply to customers, from customer segmentation to fraud detection. For products, what products should we recommend to our customers? Perhaps we want to forecast demand, inventory, and revenue.

On the equipment front, predictive maintenance is a big one. If we service equipment too soon, we're wasting money. But if we service it too late, we could incur more expensive repair costs or even needing to replace the equipment.

Finally, we have HR related use cases involving employees. What are the attributes of a best employee? Can we predict which employees are likely to leave? Another way to look at use cases involves the machine learning techniques that enable them.

We'll start with classification. A machine learning technique that we can use, for example, to predict customers with high lifetime value, or who are likely to prepay their loan or perhaps default on their loan.

Other use cases may include predicting which vehicles need maintenance, or if a customer is likely to churn, or which customers we should prioritize with marketing campaigns.

A second technique is regression, which could also be applied to some of these classification use cases, but adapted to predict a numeric outcome. Other regression use cases include forecasting, product demand, sales, or revenue.

Of course, time series provides another class of algorithms that could be applied to these same use cases. Clustering helps with customer segmentation and document classification, which overlap with the classification technique, but also for grouping cell types or just exploratory data analysis for finding similar instances.

Association rules helps to identify cross-sell and upsell opportunities. An anomaly detection is useful for transaction or claims fraud detection, or just identifying unusual cases which may signify the need for closer inspection of your data. In these examples, just scratch the surface of what's possible.

Oracle machine learning can support these use cases and more using powerful algorithms in the database. Machine learning can be leveraged in multiple use cases. Banks and other businesses in the financial industry use machine learning for risk management, such as to prevent fraud or assess credit risk, address customer retention, as well as profitability and revenue forecasting.

Machine learning is a fast growing trend in the health care industry. The technology can help medical experts analyze data to identify trends, help with clinical decision support, or chronic disease management that may lead to improved diagnosis and treatment.

Forecasting energy demand, finding new energy sources, analyzing minerals in the ground, predicting refinery equipment failure, streamlining oil distribution to make it more efficient and cost effective.

The number of machine learning use cases for this industry is vast and expanding. Analyzing data to identify patterns and trends is key to the transportation industry, which relies on making routes more efficient and predicting potential problems to increase profitability.

The data analysis and modeling aspects of machine learning are important tools to delivery companies, public transportation, and other transportation organizations. Websites recommending items you might like based on previous purchases and demographic and psychographic data are using machine learning to analyze your buying preferences and promote other items you might be interested in.

Some of the most successful retailers capture data and analyze it to personalize a shopping experience or implement marketing campaigns. Government agencies such as public safety and utilities have a particular need for machine learning because they have multiple sources of data that can be mined for insights.

Analyzing sensor data, for example, identifies ways to increase efficiency and save money. Machine learning can also help detect fraud and minimize identity theft. Retail industries can use machine learning to recognize customer spending patterns for targeted marketing or optimize supply chain logistics by recognizing outliers or anomalies in their data.

All that a data scientist needs to do is to identify the problem domain, such as transportation, find the data, and let Oracle machine learning take care of the rest. In this lesson, we've covered the reasons machine learning is so popular, and where you might consider using machine learning.

6.3 Data Mesh Architecture

Hey, there. Welcome back to Oracle University. My name is Greg, and I'm going to be walking you through today's lesson, which is all about Understanding the Data Mesh Architecture on OCI. Data Mesh is an emerging hot topic for enterprise software that puts a focus on new ways of thinking about your data. Data Mesh aims to improve business outcomes by driving more data-centric solutions. Much like what we've been talking about lately and also encourages our customers to take a more modern approach to their data architecture.

Oracle provides Thought Leadership in data strategies, as well as architectures. Our focus on the Data Mesh has been in providing a platform that can address emerging technology requirements, including tools for data products, decentralized event-driven architectures, and streaming patterns for data in motion. Oracle's approach is to think of data as a product and align it across operational and analytic data domains. And data in motion is a key indicator of success. Data in motion is an enabler for distributed and decentralized data architecture.

These innovative approaches to data management find a means to gain competitive advantage and accelerate successful business transformation goals. A mindset shift is the most important first step toward using Data Mesh. The willingness to embrace the learned practices of innovation is the springboard toward successful modernization of your data architecture. These learned practice areas include Design thinking for solving wicked bad problems, Jobs to be Done Theory, customer-focused innovation, and the outcome-driven innovation process.

Design Thinking methodologies bring proven techniques that help break down the organizational silos frequently blocking cross-functional innovation. The Jobs to be Done Theory is the critical foundation for designing data products that fulfill specific end customer goals or jobs to be done. It defines the product's purpose. The Data Product approach initially emerged from the data science community but is now going mainstream being applied to all aspects of the data management discipline. It keeps the focus on the business outcomes. The data consumers all rather than the IT and technology.

Data product thinking can be applied to other data architectures, but it's an essential part of the Data Mesh. With an estimated 25% of a firm's value being in digital capital, investing in a Data Mesh can yield impressive benefits, including total clarity into data's value chain through applied data product thinking best practices. Operational data availability using microservices-based data pipelines. Faster innovation cycles, shifting away from ETL to continuous transformation and loading or CTO. And reduction in data engineering, no code, and self-serve data pipeline tooling, and agile development.

This hidden data economy has a supply-side where application sensors and devices originate data in a wide variety of shapes and structures. And it has a demand side, where analytics and AI try to use that data in a wide variety of shapes and models. Let me show you what this looks like. The data economy in your organization most likely started as a supply-driven command data economy. The purpose of enterprise applications was to execute business processes.

They created data as a side effect. Some of that data would be packaged up for predetermined questions in the form of reports and dashboards. At the same time, new sources of supply emerged as company websites, and mobile applications proliferated. Providing the informal data economy with a diversified supply of data enabling them to create new value outside the command data economy. Meanwhile, out on the internet, something completely different happened. A market data economy arose.

This market data economy was the polar opposite of the command economy in two critical ways. First, it completely decentralized both data supply and demand. Embedding analytics and AI into internet apps and services meant that using data created new data. Driving a flywheel effect and, in some cases, a winner take all outcome. Now, why does this matter to you? Because all three of these data economies are operating in your firm, usually separately. And now, where that leads us is to dead data capital.

Data assets that either go unused or fail to deliver their option value because they're trapped in data silos too difficult to repurpose or being undiscoverable or ineffectively governed such that they're allowed to be used in an unclear way. Obviously, this is unacceptable and the reason every company is bent on digital transformation, but there's a problem. As companies attempt to bring their data economies into the light, they face the twin threats of digital transformation. Namely, loss of competitive advantage from moving too slow on the one side and legal or reputational damage from moving too fast on the other.

Every firm must thread this needle over and over as they attempt to bring their data economy into the light and make it grow. But what's the alternative, you might ask? Can you safely grow your data economy balancing both data innovation and data compliance? The answer yes. To see how, let's take a big step back. The key is a data strategy that reinforces competitive strategy, and to define strategy, we turn to Michael Porter, godfather of competitive strategy. Porter defined strategy as creating unique value in a unique way.

It's not enough to create value your customer can only get from you. You have to deliver this value in a way that your rivals cannot easily copy. A firm's data strategy should reinforce its competitive strategy. Data strategy comes down to creating unique data assets and using them to enhance your firm's uniqueness. The way you use proprietary data assets should strengthen your differentiation and improve your cost position relative to rivals. But there's one more piece to a successful data strategy.

It must also protect both the observer, usually the company itself, and the observed, including customers, partners, and employees. Because companies must balance data innovation with data compliance, any data strategy must consider constraints of legislation, regulation, business ethics, and cultural norms when planning what data to capture and, more importantly, how to use it. But how do you plan a data strategy that reinforces competitive strategy?

The key is to examine the firm's activities. Everything a firm does consists of activities. The way an organization designs its products or services, the way it markets them, distributes them, services them, all of these processes and decision points are the firm's activities. Sustainable competitive strategy comes from creating a unique system of activities. Data products are the building blocks of data strategy.

It is a set of observations, and its complementary code designed to fulfill one or more jobs to be done. The supply side are the data sets, models, and libraries. The demand side is the need for analytics, algorithms for machine learning, and/or data services. To create a unified data economy, enterprises need a new data backbone. You need an enterprise architecture that helps you shift more activity into your market data economy, connect to critical back-office and financial data assets from your command data economy, and bring the informal data economy into the fold as well.

But how do you create such an architecture? Well, there are four principles for translating data strategy into data architecture. They are data liquidity, data productivity, data security, and data governance. A proper Data Mesh is a mindset. An organizational model and an enterprise data architecture approach. It should have some mix of data product thinking, decentralized data architecture, event-driven actions, and a streaming-centric service mesh style of microservices design.

A Data Mesh is not a single Cloud data lake, even with quote, "domains, catalogs, and SQL access." It's not a point product. No vendor has a singular product for Data Mesh. It's not an IT consulting project. Strategies or tactics still require platforms and tools. It's not data fabric, which is broadly inclusive of monolithic data architectures, and it's also not self-service analytics. Easy to use user experiences can front a mesh or a monolith. Starting with value-focused data product thinking. Now, data product thinking can be applied to other data architectures, but it's an essential part of a Data Mesh.

The next attribute is a decentralized IT system. Decentralized IT systems are a modern reality, and with the rise of SaaS applications and public Cloud infrastructures or IaaS, decentralization of applications and data is here to stay. A Data Mesh is not just one single kind of ledger, and it can make use of different types of event-driven data ledgers, depending on the use cases and requirements. For example, a general-purpose event ledger, such as Kafka or Pulsar.

Data event ledgers such as distributed CDC or replication tools, messaging middleware like ESB, MQ, JMS, and AQ, or a blockchain ledger for secure multi-party transactions. Together these ledgers can act as a sort of durable event log for the whole enterprise. Providing a running list of data events happening on systems of record and systems of analytics. Data streams may vary by event types, payloads, and different transaction semantics.

A Data Mesh should support the necessary stream types for a variety of enterprise data workloads. Events are funneled into declarative time processing for events, logging events, database events. Once it's in the streaming environment, they can be a part of the workload and then feed the data events into RESTful services, relational graphic columnar, and all different data types into the needed formats and structures. These events and processes can run in centralized and decentralized environments.

Oracle provides key solutions for each of these areas, which allows for simplified ways to implement the required data strategies and architectures. As we discussed, data is being produced in operational, application, compliance, and other data sources. As you look at the different Oracle tools for data products for the data in motion from producers to consumers of the data for event-driven integrations and streaming data pipelines.

We have Oracle tools such as Oracle IoT Cloud for IoT event services. Oracle GoldenGate to keep stores in sync and integrated, or Oracle Integration Cloud, which is a business activity monitor for alerting on app events. We also have tools for data collection, such as Oracle Database multi-model curated data sets in relational, JSON, graph, or other formats. Oracle APEX data-driven self-service apps produce and share data via simple GUI or Oracle Data Lakehouse, either operational raw data or curated data sets in various data types.

Data analytics, Oracle Analytics Cloud for consumer summary data through visualizations and dashboards. Oracle Stream Analytics for time series analysis on all events and Oracle Data Science, which are the machine learning features and models that produce the production level assets. This ties the data strategies to the Oracle tools because these are significant differentiators for Oracle along with Data Mesh for enterprise data, trusted corrected data, multi-cloud and portability options, and overall data management Thanks so much for stopping by. Take care.

7. Developing on Oracle Database

7.1 Manage Autonomous Database instances: Using REST APIs to manage ADB

Hi, my name is Kamryn Vinson. And in this lesson, we will focus on using REST APIs to manage the Autonomous Database.

Oracle Cloud offers full REST APIs for DBAs and developers who would prefer to interact with Oracle Autonomous Database Cloud Services programmatically over REST rather than log in to the cloud console and click through screens. This provides a mechanism for developing customized deployment and management scripts that can be saved and reused for deployments, setting gold standards, and storing entire application infrastructure stacks as version-controlled code.

The Oracle Cloud Infrastructure APIs are typical REST APIs that use HTTPS requests and responses and support HTTPS and SSL protocol TLS 1.2, the most secure industry standards. Calls to the Oracle Cloud Infrastructure using REST APIs can be written in popular scripting languages such as node.js, Python, Ruby, Perl, Java, bash or curl.

All Oracle Cloud Infrastructure API requests must be signed for authentication purposes. Oracle Cloud Infrastructure APIs are the ones used to interact with the Autonomous Database. These avoid using usernames, passwords, and are based on the draft-cavage-http-signatures-08 specification for secure communication over the internet.

The steps to create and sign API requests are-- form the HTTPS request. Create the signing string, which is based on parts of the request. Create the signature from the signing string using your private key and the RSA-SHA256 algorithm. Add the resulting signature and other required information to the authorization header in the request.

You will also need to generate an SSH key pair in the pem format. This example creates an Autonomous Database in the Phoenix data center, with a database name of "adatabasedb1," the specified password, 8 ECPUs, and 1 terabyte of storage. The response to an Autonomous Database API command, in this case, a CREATE command, will include the current status of the CREATE process and other relevant information regarding the database being created.

In this example, we see database parameters, such as the compartment or tenancy of the database, the database display name, the name of the Oracle Database, the current status of the database, the time the creation process started, the number of CPUs allocated, the number of terabytes of storage allocated, the license model being used, and a direct link to the database console.

In these examples, we see the basic API call for deleting an existing Autonomous Database and starting and stopping an existing Autonomous Database. Again, the result will be similar to what we saw in the previous slide.

In previous sections, we saw how to scale the database from the console. In this RES API call, the database is being scaled to a total of 20 ECPUs allocated. In the second example, a backup is initiated on the database. And this concludes the lesson on using REST APIs to manage the Autonomous Database. Thanks for watching.

7.2 Autonomous Database Built In Tools

Hey there, Nick here. Welcome to the Oracle University module on describing Oracle's data toolset. Let's get started. The Autonomous Database now includes a built-in tool suite to extend the benefits of autonomous operations to these other players.

Oracle provides industry leading tools for each of these legs in the data analysis medley. Oracle Data Integrator, or ODI, is an enterprise class data integration tool with Extract, Load, and Transform, or ELT architecture.

Enterprise Data Quality, or EDQ, is a sophisticated, powerful tool for profiling, cleaning, and preparing your data. Analytic views built into Oracle database provides a common framework for defining universally accessible semantic models. Oracle Analytics Cloud, or OAC, is the perfect complement providing beautiful and insightful analysis of this data.

For our traditional market, this is a comprehensive and compelling suite of tools, enterprise class tools for an enterprise class market. With Autonomous Database, we deliver an integrated platform. It's not a single tool with the customer left to buy the other tools that they'll need. Nor is it a solution delivered in kit form, with the customer left to cobble it all together.

It's pre-assembled, pre-configured, and pre-deployed. There's a consistent user experience with built-in best practices. It's like having an expert in a box there to guide you. Components are defined in the common database layer. So that they can be shared by all users in all tools.

And all of this metadata is brought together in the catalog. So it's not just the tools that are integrated. It's the data, too, a business model spanning data sources that can be federated when appropriate and defined in a common data catalog, which eliminates silos. The result is renewed confidence in data lineage and impact analysis.

In other words, we have collaboration by design. This built-in collaboration between specialists eliminates silos. For example, hierarchies recognized automatically in the data preparation phase are defined in the database itself are immediately accessible to the data analysts for aggregation purposes. Additional semantic modeling by the analyst, perhaps defining sophisticated calculations, such as percentage change since last year, and again, defined in the database itself, can be accessed by the data scientist.

This provides a great head start in developing predictive models. That, in turn, can be used by the CRM developer, who might want to augment a customer view with the most suitable campaign to discuss during the next meeting. So Autonomous Database comes with a sophisticated suite of tools pre-installed.

But it's not a walled garden. On the contrary, it's an open platform with open standards. If you want to speak SQL, speak SQL. So do we. We speak Python, too, if that's your preference. Whether your data is in a CSV file or a JSON format, it's going to be comfortably at home in Autonomous Database. Using the language of your choice, analyze your data using whatever tool you're most comfortable with. The whole idea is that there should be nothing new to learn.

Now, we move from the general to some of the specifics of Autonomous Database tools. Autonomous Database now includes a built in tool suite. There's nothing more to buy. There's nothing more to install. It's all right there at your fingertips in the OCI Console for Autonomous Database.

Click on the Database Actions button and log in using the credentials provided when you provisioned the ADW. This takes you to the launch pad for this tool suite. It's very easy to navigate this interface. I'll give you a quick tour.

Each tool is available via the card based interface. These are arranged in groups, such as Development, Data Tools, Administration, and Monitoring. There are some useful quick links accessible from this panel on the right hand side. Context sensitive help is available by clicking on the question mark. Preferences, an About menu, and logout dialogue are all available here.

You can always return to this launch pad by clicking on the database actions at the top of the screen. There's also direct access to any tool from the so-called Mega Menu, accessible by clicking on the hamburger menu at the top left. Anyone that's tried to load data knows that it's more easily said than done, until now, that is.

Just say what you're about to do. Load data into your ADB, link to data in a remote location, or even set up a live feed, then say where your data is, in a local file, a remote database, or in an object store in some Cloud, and press Go. That's it. The bottom row of the cards allows you to set up Cloud storage locations, monitor data load jobs, and explore data in the ADB.

We'll start with a simple load of data from local files. And then from a Cloud-based object store. In this case, I'm using the OCI Object Store, but we can just as easily access files in AWS S3, Azure blob storage, or GCP. This explore capability is very helpful for profiling data that you've just loaded.

After clicking the card, we see a list of tables in the Autonomous Database. I click to select one of these tables, Movie Sales 2020, and press Statistics. Immediately, we can identify some data quality problems. We have data for 12 months. But since I only want to analyze data for Q2, I have some superfluous months in this data set.

Here, we see data for 14 days in a week. That doesn't sound right. Drilling into investigate, we can quickly see the problem. Some of these days are in upper case. And others are in lower case.

So what have we seen here? In a few short minutes, we've loaded data from both files of various types and from multiple locations. We've quickly scanned that data and identified some problems that needed to be addressed. We'll do that next.

7.3 CI/CD for APEX and Oracle Database Developers - Part 1

Hey, everybody. Welcome to the Oracle University module on CI/CD for APEX and Oracle Database developers. My name is Alli, and I'll be your host today. Let's get started. Let's start off by explaining what CI/CD is. CI/CD is just Continuous Integration, Continuous Delivery, or Continuous Deployment. What this is going to help you do is add automation into the various stages of the development life cycle. So if you're developing an application, this will help you automate some of the processes in there.

It's going to help consolidate your code. It's going to help your developers be on the same page. And it's going to automatically have this code merged and tested and put into this big repository. Now it's mainly more about the why. So why are we even doing this? Why do we even care? And the why about this is very important because it does introduce consistency into your development process. And that's important.

You want to have consistent processes in place so that everyone's on the same page. And you don't have what they call the cowboy coders, the people that are just randomly coding stuff, not paying attention to the guidelines that you had set forth and just doing what they want to do. It's going to cause issues. And it's going to cause problems.

We have to know that you know that the coding standards set forth for security are in place and tested so that we can prevent these issues from happening. This process is going to help you with that because as you go through the merge process, you'll see what code actually is being put into your mainline.

Another thing is it's with these processes you're going to find issues. You're going to find quality issues a lot faster because every time that a certain event happens, whether it's going to be a push or commit to the mainline, it's going to spawn a process that's actually going to test the code that particular developer did on an instance on an Oracle Database, and install what it needs to install, and then report back what happened. We do this at various stages of this development life cycle now. And we're actually able to do it on clones of our production environment.

So when we go to these processes, when we deploy to production, there are a lot less surprises because we've done it so many times as we develop. Again, it's not the developer that has to do this. It's a completely automated way for doing this so that we can see if there are any issues. And once all this starts, you'll be able to do more frequent releases because they're a better quality. They're standard, they're secure, as well as consistent.

A lot of the times, we all develop with APEX in the database. We're all sitting in a single instance. And yes, sometimes they're going to step all over each other, and sometimes multiple people will be working on a package and not know. One person may compile it one day, and another person may come in and compile it right after that, pretty much erasing all of his changes. It happens.

And the other thing is, it's not always easy to create multiple Oracle databases. There is licensing involved sometimes. There are costs involved. Do we have the hardware? Do we not have the hardware? Do we have the ability to clone? Do we have the space on our array? There are a lot of issues that come with this, and it's not always a particular developer or coding issue.

And the database in the APEX flow is a lot different than the traditional one. We see with that, and it's mainly because it's a stateful versus stateless process. If I'm just compiling a bunch of JavaScript files or jars or whatnot. That's really easy to do. I bring up my doc containers, I can pilot, I can push it out, and I'm done, where APEX is mainly metadata-driven and the database has these versions and lack thereof, I should say.

It's not always easy to version in APEX, and it's not always easy to version in the database rolling back. Or do we roll forward? What do we do if there's an issue and how do we get around that? And a lot of times, we have these very manual processes out there. The change tracking isn't really there. It's kept in a bunch of scripts, or it's kept in a bunch of zip files on someone's laptop. And when it's time to deploy, we pass the zip file around an email, and then somebody jumps onto the development and production instances, and runs the scripts in hopes that it works. So there's a lot of issues around this that we've been facing.

So it hasn't really been a very popular option within the database in APEX realm. So we're going to show you some of the tools and processes that we have with the database tools, and mainly the database development tools to help with this process and then reiterate our commitment to this particular area.

So what do you need to start? Starting is best done slowly, if you have a team of developers that have never done this before or have never worked with these tools. So to start with that, just get a free account on GitHub or host your own GitHub or GitLab or Bitbucket. We're going to be using GitHub. The next tool we have is SQLcl and Liquibase. And we'll be talking about this a little bit more in a second.

Lastly is Jenkins for our pipeline and for some of our automation. And if you really want to be an overachiever and see some extra credit stuff, you're going to need VS Code and SQL Developer for deploying and writing your code, PL/SQL for all of your unit testing. That's going to become very important in our process. Terraform, and then an OCI account

SQLcl and Liquibase. So SQLcl is if you haven't used it, it's as they say in The Mandalorian, "It is the way." It is basically a command line SQL Plus tool that's about 35 megs big, which you can connect to pretty much any database you have without having to download the entire Oracle client. It's great because we have all these great features built into it.

There's a lot of ways to connect to a lot of different things. You can connect to Autonomous Databases. You can connect to on-premise databases. You can connect to cloud databases. You can connect a cloud database using the SQL Server, the SQL REST service. There's a bunch of ways to connect to this database. It's a quick install. It's a quick download, and you're ready to go. It has a bunch of features in it, and one of the features we have in it is Liquibase.

Liquibase comes free with SQLcl, so you don't have to pay extra. Liquibase is going to allow you to have this change tracking for database schema changes, as well as your Application Express apps in there. The Liquibase feature enables you to execute all these commands that allow you to do this change tracking, and then move them from environment to environment in a manual or automated way.

Whether it's scripted or called from a script that's scripted, either scripted or called from Terraform, or called from a pipe, SQLcl is there and able to run all these things to have that change management and change tracking across multiple environments.

One of the great things about SQLcl is we keep it up to date frequently, and we're pushing out a lot of updates. Every time we move this to an environment, we're going to have this change log table in there. And you can see two examples of what's in that change log table here. So we have a change log details, which is a view that sits on top of these two tables that are in there. It gives you a nice view. So you see the file name that was used. And then it's going to show you the SQL that was actually run. And obviously which helps with rollback because we can actually see what's on there.

We can pull it back or we can go and grab the old stuff and reapply it. But you can see here that in the first one in the red box, this was applied on a schema that already had the AMP table. So through the end table, that XML file has the full table created in there. Liquibase saw that the table existed and said, oh, you're missing this column. And it issued an ALTER TABLE and added the column to it instead of deleting the table, which would be bad, and recompiled the entire table. So it's smart enough to know, hey, this stuff exists. And I'm going to do this rather than that.

So Liquibase will generate the objects for Application Express. And there are two ways to do this. First, we can do that genobject, as we said, the type, and then the application ID. And what it's going to do is create a single file, just as you may do for the APEX report. And we're going to have this single file that we can use and go from environment to environment, if we want to move and split this up into multiple files, so we can actually see the changes that developers are making. We just add that -split.

We also want to do the skipExportDate and export the original IDs we're doing this because if we don't add these, the exports are going to have new IDs and they're going to have dates. And when we try to merge that into our Git repository, those are going to show up as changes. If this is a huge app, it's got hundreds of pages. There's 100 merges now you're going to have to dig through and look and say, oh, well, this is just the date or this is just the ID changing.

It's not an issue. It's fine. There's no actual change in the page. So if we do it this way and we pull them out like this, we're now able to see just the changes from page to page. Or if there's a shared component or an object or something that somebody changed, we can now see those individual pieces when we export that app.

Now, we're going to get a little bit more into the process of having pull requests and branches and whatnot. But when you are working with your application and you do this export with Liquibase and you're going to push those changes into your Git repository, Liquibase is smart enough to say, oh, this didn't change, this did change, this didn't change. And those are the files that are going to be updated.

So when you actually do that push into the Git repo, only the files that have been changed or your shared components or what you have from APEX will reflect that change, so we can actually see what the developer is working on. And when we go to that code review, when we go to that merging back into our main code line, we'll know exactly what was changed again.

Going back to that accountability, going back to that security, going back to that standardization, we can now see it in use through this. Then when we want to put this application into our new environments, we can do this with our Application Express apps as well. And it's the same way. We're kind of theoretically building our pipeline here. First thing it's going to do, it's going to get our environment and set up our environment.

The next thing it's going to do is apply the new database objects, and then one of the other things it's going to do after applying the database objects is we're going to install our Application Express app into the environment. Now, we've seen some of this about people asking about data. Yes, you can actually pull data out with Liquibase.

Now, I wouldn't use this for millions of rules. We have this thing called ORDS. And ORDS is great at moving data around, having this REST-enabled endpoints. And you can just move this data around or import the data, or you can use Data Pump. What this is great for is metadata tables.

So if I have a table that has a bunch of lookups, maybe a few hundred rows, we can do Liquibase data object and then the table name or table names of multiple tables. And it's going to pull all that data out as insert statements. Then we're going to do Liquibase update changelog the data.xml that it creates. And it's going to insert that data into that particular table and can move data with Liquibase.

But as Uncle Ben said in Spider-Man, "With great power comes great responsibility." You're not going to do this to a huge million row table. That's just not the way to do it. There are better ways to move that data around, so this is probably best used for metadata type tables or smaller tables.

Now with that said, you may not always need to move the data around. It all depends on what you're doing, and you can also roll your own scripts for moving the data around. If it's a small amount, it's going to come back to how you're cloning and setting up these environments for your developers.

Now, let's get into the actual pipe and what we can actually do. Now, this goes back to Terraform and OCI. Every cloud company out there has their Terraform APIs. But we in OCI also have our Terraform APIs. And using the Terraform APIs within our pipeline is going to allow us to use these OCI resources in Oracle Cloud to quickly create environments or quickly create pieces that we need for our pipeline or for our development.

This is basically the code you would need in Terraform to create an Autonomous Database. A couple line of code saying, create me a 1 terabyte database named testdb with some random string, and it's going to be an ATP and an autonomous transaction process instance. And that's pretty much it. That's all you need to put it in a compartment if you have a compartment instead.

But this is a quick side note. All the code and everything we're showing here, we're actually putting or in the process, or have already put into our Oracle DevTools GitHub repo. So if you need to refer back to it, you want to use it, you want to alter it, you want to play with it, it's all going to be in our GitHub repo. So you can take a look at it over there.

So back to the CLI and Terraform. We can use these to create these VMs or these in network security lists and whatnot to create our environment for testing. So this isn't a ploy for the Autonomous Database or to use what we need to use. But the Autonomous Database is great because you pay for it by the second. And in a lot of these times when you create these workflows, this database is only going to be up for maybe 30 seconds to a minute as you apply the code at machine speed and then do the tests that you need to do.

And then when we're done, we delete the Autonomous Database, so that whole process may cost you a dollar or two, depending on your licensing or what you want to do with your Autonomous

Database. But spending that dollar or two on testing in this Autonomous Database that you quickly created, let's say, we catch a major bug by doing this process. How much is that going to save you down the road? So again, we'll talk about databases and environments in a minute. But this is one of the ways that we're going to look at it.

7.4 CI/CD for APEX and Oracle Database Developers - Part 2

So the environment that we have here that we're setting up is an OCI. But it doesn't have to be. You can set up this environment on-premises very easily. This is what we have, this flow. It starts with the developers. Our developers are going to commit their code to the code repository.

This code repository is going to trigger via a webhook. It's going to trigger a compute VM with Jenkins. And Jenkins is going to run its pipeline. Jenkins has SQLcl and Git and Terraform already on there. So we can do a bunch of things.

Then we're going to orchestrate some of this with spin occurring. Being an OCI, we can set this compute VM up to have what we call instance principals. And what it allows this compute VM to do is act like a user. And we can say, OK, VM, you can do A, B, and C, but you can't do x, y, and z. So the VM can create an Autonomous Database. The VM can create a compute. It can create a load balancer. It can create networking. Or it can't do so when we go and we create this pipeline.

That Jenkins is kind of orchestrating with Terraform, the environment we want to test our code in. Now granted, this doesn't have to be an OCI. You can do this with scripts, or you can do this on-premises. But as we said, the Terraform makes it really, really easy to do with setting up this infrastructure as code type of environment.

And again, every cloud vendor has Terraform hooks, whether it's AWS, Oracle, Microsoft, what have you. So this can easily be replicated. And also, Jenkins is open-source, Spinnaker's open-source. SQLcl is free, Git is Git, and Terraform is terraformed. Just yum install Terraform, yum, install Git.

Oh, and by the way, in OCI, you can actually do a yum install SQLcl in yum install ORDS, and we'll get that latest version of origin SQLcl out of our Young repository. Install it on your VM or Database Cloud Service VM for you automatically. Nice little thing we added in there.

Now, let's put this all together. So we saw all the components that we need for this CI/CD flow. So let's start putting these building blocks together here. So what we do is we have the start. We're going to use this in the agile method methodology here because it kind of fits the best with this. But you may have your other terminology or other ways to do this.

So we're going to start our sprint, our agile methodology, and developers are going to start on their tickets. These tickets are handed out to the developers here. You work on feature A. You work on feature B. You work on feature C. The developers go off, and do what they need to do. They check in their code. They always want to do a code review with developers and stakeholders.

You're going to merge the code with main. The sprint ends. You're going to create a version-- version 2, version 3,p version 3.5. And then that re-version is releasing into DEV, UAT, in PROD. So let's dig a little bit into this and actually see a little bit deeper into what a developer would actually do.

So if you're starting from scratch or you're going to take a baseline of what you're building to put into gate, so you're going to have your initial schema and all the objects, your initial data model, your initial packages or stubs, as well as your initial APEX app. You're going to commit it. And you're going to call this version 0.

Now, the development cycle starts. Developers are going to pull down the code to their local laptops or local environments and create branches for them. At this point, the developers need to start working in individual developer environments. That's one of the very important things here is if this is going to work, it can't all be working in the same database because you're going to step on each other. There's going to be issues and the change tracking is just completely going to go out the window.

You have to use these individual environments. So pull the code. I start working, code is done. I'm going to push it up to my branch. And I'm going to start a pull request. Now depending on what you're using, whether it's Bitbucket, GitHub, GitLab, there are a bunch of different terms, but they all in essence do the same thing. But in GitHub, it's a pull request. And a pull request is going to lead to a merge. So I'm going to ask for a pull request.

This pull request is going to start a pipeline. This pipeline is going to take the code from that branch. It's going to clone or use a copy of production UAT or DEV, something that reflects the latest version of main or what's in production, so we can see how this code is actually going to work on what we have.

If this pipeline comes back clean, we're going to allow the merge into main. That merge into main is going to cause a pipeline to run. But once we've merged in the main, we'll delete our individual development environment, and we'll delete the branch. And now, once all the developers are done, we're going to create that version in the repository. So this is the flow that we see is working for our developers out there.

It's actually pretty simple. So how do I do these individual environments? How do I create these? Well, an individual environment needs to have the latest version of main and also needs to have all the objects, some data, as well as the Application Express apps and workplaces pre-created for you ready to use. So how do I do this?

Well, there's a lot of ways you can do this. In OCI, we have the Autonomous Database. We have the OCI, CLI, PLSQL SDK. We've actually created an application that you can use as a control plane, in essence to create these Autonomous Databases. And then with that, add Data Pump exports onto them. Then we basically take this. We add the Data Pump export. We install the Application Express app on top of it. And we're ready to go. These environments can be given out.

You can also just create these with SQL Plus. You can use the ORDS SQL Endpoint, the SQL REST Service to exit, just run SQL on it, or you can use Terraform. So what we're going to do here is we're actually going to take all these ways to do it. And we're going to pull this in our GitHub repository with examples for each one.

Autonomous Database clones is really easy. If you're working with the Autonomous Database, there's the ability to clone that environment directly from the UI using a CLI, using an API, or using Terraform. Really easy to do. If you're using the Autonomous Database, and if you're using it in OCI using the database VMs, you have the ability to clone that database environment. It's going to take a while, and we want these things up as fast as possible.

You can also create a new database VM from backup. And again, that can be done via Terraform. It can be done via OCI. It can be done from the UI. And if you're using Exadata Call Service or an

Exadata, you can use sparse cloning. I can sparse clone that database very quickly, create that database clone.

If you're on-prem or if you're on the Cloud, you're just using a database that can utilize multitenancy. With 19c, you can create three PDBs without having to pay for multitenancy. Or if you have multiple tenants, you can do this.

Now, we've exposed PDB lifecycle management via these database APIs via ORDS. So if you have ORDS installed in front of these databases, you can use that as a control plane to create clones, or PDBs, of the database directly through ORDS, through these REST APIs. It's very easy to do. You can also just jump in there and have a script that would run and create it, or use the REST-enabled SQL service that comes with ORDS to create a REST service that creates these pluggable databases for you.

When it comes time to give these environments out, I issue these commands. I create these environments for all the developers, and we're ready to go. Multitenancy makes this really, really easy to do. If I don't have multitenancy, I'm on 11 still. I'm on 12. I'm running on non-CDB.

But how do I do this? Well, you can do this with reusable instances. One of the ways you can do this is using a guaranteed restore point or flashback database. So for example, if you are on one of these databases, you apply version 1 of the code. And after that's done, you take a guaranteed restore point of your database. We're going to be in our GitHub repository for you to take a look at and actually run the SQL. We're going to set a guaranteed restore point.

Developers are going to do what they need to do on their environment. They're going to push their code, and then the release is cut. Then they're going to flashback the database to that point where we installed version 1. We're going to pull version 2 down from the Git repository, so we have everything. We're then going to apply version 2 onto the database and then take another guaranteed restore point and there to work and continue that process.

So it's going to create a nice little environment for the developers to create and use without having to have a multitenancy license, or be stuck having to work in a single instance. This database can be in Docker. This database can be in a VM. It can be anywhere. It could be on the Cloud, but it does not allow them to flashback the database to get clean versions, so they can continue development in their individual environment.

Then you can do an RMAN clone from an environment, that does involve database links. So just make sure that those are set up correctly. And then you can do Data Pump. If you just Data Pump export out that entire schema, you now have that Data Pump you can use and use DBM's Data Pump or SQL Developer to go and plug that Data Pump directly into your database.

And then when you're done, you can either pump the schema or go grab a new Data Pump and put it on there. So there's a bunch of ways you can do this without having a multitenancy license. And then as we have here, you can do Docker virtual machines. And then lastly, you can also do ACF cloning or GDB clone to create databases using ACF snapshots. This gets a little bit more technical, but there are easy ways to do this, especially using gDBClone.

So the code pushes in the processes. Every time something happens, you want to spawn the CI/CD pipeline. It's going to clone the database from somewhere. It's going to apply the changes. It's going to run the unit tests. It's going to report back on the status of the pipeline, and then it's going to destroy the environment if everything's good or keep it running.

So the developer can go in there and see what's going on and say, hey, what happened here? And here's the log file. Oh, I forgot to do this, or this code isn't compiling, and then the cycle is going to start again. They're going to fix it. They're going to push it into Git. They're going to create a pull request. And it's going to start that process again. Once that pipe is clean, you can review it, and you merge it.

So let's actually see a little bit more with the pipeline in action here. We're going to push our code. It's going to be a pull request. The Terraform is going to create a clone of a particular environment. The environment pre-setup scripts are going to run. SQLcl is going to apply the database objects. SQLcl is going to apply or install the Application Express. SQLcl is then going to run the UTP of SQL for the unit test, to make sure everything is run correctly, to make sure everything reports back.

The pipeline is going to look at the logs for ORA errors. If it comes back clean, we're going to delete the clone, and then the developer gets a notification of the status. And then we start the whole process of code review. If it's not clean, a notification goes out to fix it. For an Autonomous Database, maybe we include the wallet and the log files.

For a non-Autonomous Database, we just tell them where it is and they go and fix it. Again, this is spawned because of what we do in the GitHub repository. And we want this to happen on a PR, a merger, a push. It depends on what you want to do. You can actually configure what it does.

So how do we deploy to production? A lot of people don't trust the automation quite yet. So what they're doing is they're going to do it manually. Jenkins creates these artifacts and stores them off for you. We have a URL to those artifacts. We're going to just download those artifacts and apply them onto the environment and that's fine. A lot of people just don't trust that part of the process to deploy it.

Now, the good thing about this is you've tested this so many times with the automation, with the pull request, and with the merge request, that you know there is a really, really good chance that there are going to be no issues because this has been practiced multiple times on clones of production.

So that's the whole CI/CD process of eliminating these surprises because we know exactly how this is going to run on our production environment. So in essence, it should just be unzip the zip, run the file just as we do with the pipeline manually, and then we've deployed into production. If you don't want to do that and you want to automate this so it's automatic and does what it needs to do, we can do this.

So we have this flow here that, again, once you create that version, that version is checked, it comes back clean. We can then deploy that version to a new test or pre-dev environment to see how it's going to run. And then we can start our deployment pipeline. In our deployment pipeline, we can actually choose where we want to deploy this code to within that pipeline, and then deploy it directly there. And we can restart this pipeline multiple times to do this.

Rollback is an interesting one that we've seen. And with rollback, it depends on what you want to do. So if you have bought off on pluggable databases, if there's an issue, you can basically rollback the entire database. You can do flashback database, rollback the entire database to a guaranteed restore point that you created before you deploy the code, and you're back to square one, and you can start again. But not everyone is using multitenancy.

So what we can do is we can rollback the objects via Liquibase. Install the last good APEX app, and then we can do a flashback on particular tables and use the metadata tables to restore what they looked like if we had a data import as part of the process. We can then flash them back to what they looked like before we started that process.

Or a lot of folks are actually looking at rolling forward. They basically fix what happened in production, and then they take those changes and put them back into the mainline code. It all depends on what you're doing and how you want to do it.

It's been awesome spending this time with you today, learning about CI/CD for Oracle APEX and Oracle Database developers. Now go out there and build something cool.

8. Resiliency

8.1 Database Maximum Security Architecture

In this lesson, we are going to be talking about Oracle's maximum security architecture. My name is Kamryn. Let's get started. Securing the Oracle database is much like securing any IT system. It starts with implementing a secure configuration and monitoring that configuration for drift. As part of that, you will want to validate your configuration against best practices and generally accepted security templates, like the CIS benchmarks or DISA STIG.

Next, you need to encrypt your data and protect the encryption keys. Encryption is a fundamental security control, and without it, your data will always be exposed to at least one attack vector. Encryption should be for data at rest and in motion. Then, you want to control access to your data, ensuring that database accounts have only the privileges necessary to perform their business function. This includes removing unnecessary privileges from interactive accounts, controlling your privileged users, and enforcing a trusted path to data that considers session context as well as assigned privileges.

Remember that most database breaches involve the use of compromised accounts. So this step is very important. You also need to plan for failure, monitor access to the data so that if one of your other controls fails, you have the data you need to hopefully identify out of policy activity quickly and support your post-breach investigations. This includes a combination of native database auditing. The Oracle database has best-in-class capabilities to help in this area and network-based activity monitoring that examines all incoming SQL statements.

No single product or option fully secures your database. It takes a combination of several technologies working together. Assess your system to identify insecure configuration choices, configuration drift, and missing security patches. Minimize your attack surface by removing excessive or unused privileges, sensitive data from test, and development systems. Encrypt your data at rest and in motion.

For data at rest, you also need to protect your encryption keys from loss or theft. Strengthen authentication to the maximum degree possible, preferably employing multi-factor or strong authentication for interactive accounts. Control access to data and enforce separation of duties. And, again, plan for failure. Monitor database activity to detect anomalies and support investigations. You can't skip any of these areas. Leave out just one piece of the puzzle, and there is an easy path to your data for attackers to exploit.

From a database feature, option, product standpoint, here is what goes into each of those areas. Note that a good baseline level of security can be achieved by properly configuring included database features. Also, note that most of our Database Cloud Services include all database options and the Data Safe security service at no additional cost. Securing an Oracle database is much like securing any other system. You are protecting your data. That could be intellectual property, financial data, personal data about your customers or staff, or more likely, a combination of all three.

Because data is valuable, you need to Guard against theft and misuse. This data is used for business purposes, and that means users and applications connect to the database, and you need to safeguard

that data with security controls that restrict access to the data according to your policy. To do this, you'll need to do three things-- assess the system to determine its current state and develop a remediation plan. Is the system configured properly? Patch is applied regularly? How are user privileges managed? Are you enforcing least privilege? What types and how much of sensitive data is the system holding?
Your existing investment in the Oracle database gives you the features and utilities you need to assess your database and identify areas for improvement and risk reduction. To detect, detect attempts to access data outside of policy and identify anomalies in data access. Almost all database activity is repetitive. So anomalies are frequently a leading edge indicator of attempted data theft. Prevent access to data that doesn't go through the database control mechanisms, sniffing traffic over the network, reading the underlying data storage layer or misuse of database exports and backups. Block inappropriate access to data through control mechanisms that consider the context of the access, not just the identity of the account accessing the data. Oracle provides industry-leading capabilities for each of these security control objectives. Our team can help you identify the right technical enforcement for virtually any control objectives.
How are databases attacked? Databases are valuable repositories of sensitive information, and a data thief is almost always going to target databases in an attempt to steal data. Before we think about defending against this type of attack, let's look at how the attacks normally occur. Databases don't just operate in a vacuum. A database is accessed by users and applications. In most cases, there are copies of databases for test, development, and stage and user acceptance testing. Databases persist data into a storage medium and run on servers with operating systems and peripherals. All of these are managed by administrators. And administrators are a hacker's favorite point of attack. If they can compromise an administrator account, they're in with elevated privileges and, in many cases, no controls over what they can do.
If the attackers can't compromise an administrator account, they can often compromise an end user account, lower privileges, but often still with access to data. Or, it can be used as a stepping stone to get to that access. Applications also make an attractive target. They are frequently more exposed than the database or database server, often even available from outside the corporate firewall. Speaking of those firewalls, if an attacker has managed to penetrate to the internal network, they may choose to go after data traveling over that network. This type of attack is much less likely to be detected than attempts to access the database directly.
Another popular attack is against the underlying data files, database backups, or database exports. Here, again, if the attacker is successful, they may be able to steal the entire database without ever having to try to log into it. If none of those things work, perhaps the database has an unpatched vulnerability. In many cases, there are automated attack toolkits that help exploit those vulnerabilities.
And don't forget those non-production copies of the database. In many systems, the test and development instances are just clones of production and almost never are they monitored as closely as production. There are some things we always expect to be done, what we call the baseline security posture. Establishing the baseline security posture involves several different types of controls. We will assess the system state, prevent unauthorized activity and detect activity that is relevant to our security controls.
Our first control is assessing the database configuration. We want to ensure we haven't made configuration decisions that introduce unnecessary risk into the environment. We'll also check to make sure that the database is current on security patches. For this, we have two tools available to us, Database Security Assessment Tool and Data Safe. Database Security Assessment Tool is a free utility available for download via My Oracle Support.
Data Safe is a cloud service that is included at no additional cost with Oracle Cloud database services. Data Safe is also available for on-premises databases, but there is an additional cost for those. Users and applications connect to the database. We want to ensure that if they're connecting with the username and password, we're practicing good password discipline. We also want to

consider the use of strong authentication. Your Oracle database supports Kerberos, PKI certificate, and multi-factor authentication.
We'll want to make sure those users are really able to connect to the database identifying dormant accounts and checking to be sure we haven't granted privileges that don't make sense in our environment. Here, again, Database Security Assessment Tool and Data Safe help by pointing out use of things, like the SELECT ANY TABLE privilege or grants of the DBA roles.
We should also check that database accounts are actually using the privileges we granted. Privilege analysis monitors, privilege usage and can report on privileges that an account has that are not used. We can then remove those unnecessary privileges, reducing the attack surface presented by those users. Note that privilege analysis is only available for Oracle Enterprise Edition database. It is not present in Oracle Standard Edition database. Users are inserting and updating data and also retrieving data.
That data is traveling over the network. And in most cases, we'll want to encrypt the data to reduce the chances that an attacker can simply sniff the network to steal data. The Oracle database supports two different types of network encryption, native network encryption and industry standard certificate-based TLS. Depending on how many users connect to our database and how many databases we have, we may want to implement centralized authentication.
Your Oracle database supports two types of centralized user management. One feature, enterprise user security, is available on all currently supported database versions and allows the Oracle database to consult an Oracle LDAP directory for users and role membership. The other feature, centrally manage users, was new in Oracle 18c and allows the Oracle database to connect to Microsoft Active Directory for users and role membership. We need to know what those users are doing. And for this we use Database auditing.
The Oracle database offers a comprehensive auditing capability, and you will usually want to audit database connections, especially failed logins, data control language, including creation of users and privileged grants. In data definition language like creation of stored procedures, database links, et cetera, all of these are fairly rare in most databases. So this level of auditing presents minimal performance impact. Finally, we'll want to make sure that we know what sensitive data resides in the database.
Is the baseline security posture appropriate for the level of risk presented by the data, or should we do more to protect our data? Here, we return to DBSAT or Data Safe, which allow us to scan the database for sensitive data, reporting on what types of data are found and how much of it there is. All of the controls we've talked about so far are baseline. These are the things we think any database should do. And everything we've discussed so far can be done without the need for additional costs, products, or options.

8.2 Oracle's Maximum Availability Architecture

Hello, welcome, friends, to the Oracle University module on Oracle Cloud Infrastructure Maximum Availability Architecture Evolving. I'm your host, Alex. Let's get started.

Now, before we actually jump into the specifics, it's important to understand the problem we're trying to address, and that is database downtime and data protection. We don't want any data loss. And the impact of both of these types of occurrences can be significant, as we can see from the slide.

Now, $350K on average of costs of downtime per hour, 87 hours average amount of downtime per year is pretty significant. So, it's a very, very common occurrence. It's $10 million for a single outage, depending on how critical the application is. And 91% of companies have experienced unplanned data center outages in the last two years, which means this occurs fairly often.

So, if we look at how we're going to go ahead and address this problem, it's important to understand a different terminology first. So we'll start with high availability. High availability provides redundant components to go ahead and ensure your service is uninterrupted in case of a type of hardware failure. So if one server goes down, the other servers will be up. Ideally, you'll have a cluster to go ahead and provide that level of redundancy.

And then we talk about scalability. Depending upon the workload, you want to ensure that you still have your performance. So as your application becomes more popular, and more end users go ahead and join it, the workload increases. So you want to ensure that the performance is not impacted at all. Rolling updates and patching, so if we want to go ahead and minimize the time of our planned maintenance, which happens more often, and a lot more often than unplanned outages, we need to do so in a rolling fashion. And that's where rolling upgrades, rolling patches, and all these types of features come into play.

Disaster recovery, so we move from high availability to disaster recovery, protecting us from a complete site outage. So if the site goes down entirely, we want to have a redundant site to be able to failover to. That's where disaster recovery comes into play. And then, how do we measure downtime and data loss? So, we do so with Recovery Point Objectives, or RPOs, measuring data loss and Recovery Time Objectives, or RTOs, measuring our downtime.

Now, how do we actually create these blueprints and reference architectures? Well, we use a technique called chaos engineering. Essentially, it's an art form at the end of the day because it's constantly evolving and changing over time. We're proactively breaking things in the system and we're testing how our failover, how our resiliency, and how our switch overs, and how everything goes ahead and works under the covers with all our different features.

And there's a lot of different components that can go out. So here's just kind of a sampling. We have networks and servers, storage. And all these different components can fail. But also human error, someone can delete a table. You could delete a bunch of rows. So they can make a mistake on the system as well. That occurs very often. Data corruption and then, of course, power failures. Godzilla could attack and take out the entire data center. And then you want to be able to go ahead and have disaster recovery in place.

And then there's all kinds of maintenance activities that happen with application updates. You might want to reorganize the data without changing the application, and the small, little optimizations. And these can all happen in isolation or in combination with each other. And so, chaos engineers take all this into consideration and build out the use cases to go ahead and test the system.

Best practices, blueprints for high availability-- Oracle Maximum Availability Architecture, MAA, is Oracle's best practice blueprint based on proven Oracle high availability technologies, end-to-end validation, expert recommendations, and customer experiences. The key goal of MAA is to achieve optimal high availability, data protection, and disaster recovery for Oracle customers at the lowest cost and complexity. MAA consists of reference architectures for various buckets of HA service level agreements, configuration practices, and HA lifecycle operational best practices, and are applicable for non-engineered systems, engineered systems, non-cloud and cloud deployments.

Availability of data and applications is an important element of every IT strategy. At Oracle, we've used our decades of Enterprise experience to develop an all-encompassing framework that we can all call Oracle MAA, for Maximum Availability Architecture. Oracle MAA starts at the top of this diagram with customer insights and expert recommendations. These have been collected from our huge pool of customers and community of database architects, software engineers, and database strategists.

Over the years, this has helped the Oracle MAA development team gain deep and complete understanding of various kinds of events that can affect availability. Through this, they have developed an array of availability reference architectures, as shown on the left. These reference architectures acknowledge not all data or applications require the same protection and that there are real tradeoffs in terms of cost and effort that should be considered. Whatever your availability goals may be for a database or related applications, Oracle has the product functionality and guidance to ensure you can make the right decision with full knowledge of the tradeoffs, in terms of downtime, potential data loss, and costs.

These reference architectures use a wide array of our HA features, configurations, and operational practices, which can be seen on the right. They help our end customers achieve primarily four goals. Number one, data protection, reducing data loss through flashback and absolute data protection through zero data loss recovery appliance. Number two, active replication which allows customers to connect their applications to replicated sites in an active-active HA solution, through Active Data Guard and GoldenGate.

Number three, scale out, which allows customers the ability to scale compute nodes linearly through RAC, ASM, and Sharding. Four, continuous availability, this allows transparent failures of services across sites distributed locally or remote, through AC and GDS. The above features and solutions allow customers to mitigate not only planned events, such as software upgrades, data schema changes, and patching, but also unplanned events, such as hardware failures and software crashes due to bugs.

Finally, customers have various deployment choices, as seen at the bottom, on which we can deploy these HA solutions. The insights, recommendations, reference architectures, features, configurations, best practices, and deployment choices combine to form a holistic blueprint, which allows customers to successfully achieve their high availability goals.

So let's take a look at different technologies here and we'll start with RAC. So RAC, as we said, is it clustering technology to spread your different nodes across your different servers, so you don't have a single point of failure. From a scalability standpoint and performance standpoint, you get a lot of benefit associated with that. You constantly add a new node whenever you want to without experiencing any downtime.

So you have that flexibility at this point. And if any type of outage occurs, all the committed transactions are going to be protected. And we'll go ahead and we'll move that session over to a new service. So from that point, we want to go ahead and also protect our in-flight transactions.

So in-flight transactions, how are we going to protect those in addition to the RAC nodes? Well, we can go ahead and do so with another piece of technology that's built into RAC, and that's the Transparent Application Continuity feature. So this feature is going to expand the capabilities of RAC. It's a feature of RAC to go ahead and protect our in-flight transactions. So our application doesn't experience those transactions failing and coming back up to the layer, or even up to the end users.

We want to capture those. We want to replay them. So that's what application continuity does. It allows us to go in and do that. It supports a whole bunch of different technologies, from Java, .NET, PHP. You don't have to make any changes to the application. All you have to do is use the correct driver and have the connection string appropriately configured. And everything else is happening in the database.

Active Data Guard is the Oracle solution for disaster recovery. It eliminates single point of failure by providing one or more synchronized physical replicas of the production database. It uses Oracle Aware Replication to efficiently use network bandwidth and provide unique levels of data protection. It provides data availability with fast, manual or automatic, failover to the standby should a primary fail and fast switch over to a standby for the purpose of minimizing planned downtime as well. And an Active Data Guard standby is open, read only, while it is being synchronized, providing advanced features for data protection, availability, and protection offload that we will discuss in just a moment.

So we have different database services. We have our Oracle Database Cloud Servers, we have Exadata Cloud Servers, and we have Autonomous Database. And they have varying technologies that are built into them. All of them are Database Aware architecture at the end of the day. And the Oracle Database Cloud Service, you have the choice of single instance, or you can go ahead and choose between RAC as well. You can use quick migration via Zero Downtime Migration, or ZDM, for short.

We have automated backups built in. And you can set up cross-regional or cross availability to do any DR with Active Data Guard through our control play. And we build on that with Exadata Cloud Service, by going ahead and changing the foundation to Exadata, with all the rich benefits of performance, scalability, and optimizations for the Oracle Database, and all the different HA and DR technologies that run within it, to the cloud.

Very easy to go ahead and move from Exadata on-premise to Exadata Cloud Service. And you have choices. You can do the public cloud, or you can do Cloud @ Customer, or ExaCC, as we call it, to go ahead and run Exadata within your own data center-- Exadata Cloud Service and your own data center. And building on top of that, we have Autonomous, which also builds on top of that Exadata infrastructure. And we have two flavors of that. We have shared and we have dedicated, depending upon your requirements.

Now, at this point, everything's managed by Oracle. And things like Data Guard can be configured-- we call it Autonomous Data Guard in the Autonomous Database, with a simple two clicks. You can set up cross-regional or cross availability domain VR. And then everything is built, of course, from a high-available, multitenant RAC infrastructure. So it's using all other technologies and optimizations that we've been talking about. So these are our different offerings that we have.

ADBD is grayed out for the following reasons. One implies the others. ADBD does not support hybrid Data Guard. ADBD hybrid configuration must be achieved through GoldenGate. ADBD does not have exact capabilities yet-- planned 2021. ADBS exact capabilities have been announced at the end of 2020, so hybrid GoldenGate for ADBS is possible. Technically, Exadata to ADBD would reach platinum if both sides are protected with RAC and Data Guard or Autonomous Data Guard. Customers still have to adapt its environment application.

Now, it's not the only option, though. I want to also give a multicloud example. So with multicloud, you can go ahead and have a database tier running on the Oracle to take advantage of RAC and Active Data Guard. And you might have a third party for the other cloud systems, represented by, perhaps, that pink block, that you could see, which could be a third party cloud, which your mid-tier will be running in. This is a gold MAA architecture.

So we have two different regions in here. We have a primary, we have a local standby associated, we have a remote standby, and then we also have another local connected to that remote standby. So if we do a roll transition, we will still have a local standby tied to it. So it's set up, even for roll transition. We could roll between the two different data sets. And this is a very easy way to go ahead and get in multicloud environment. And then we have Megaport, one of our partners providing network redundancy as well, in that third-party data center.

So with that, I will summarize. So high availability, disaster recovery, absolute requirements-- everybody should have it. Everybody should think of it ahead of time. We have different blueprints, different tiers of our MAA architecture that map different RTO and RPO requirements, depending upon your needs. And those may change over time. And finally, the business continuity we can provide with MAA is for both planned maintenance and unplanned outage events. So it's for both, and that's a critical part to this as well.

Thanks for hanging with me for Oracle Cloud Infrastructure Maximum Availability Architecture Evolving. I hope you learned something useful.

9. Upgrades and Migrations

9.1 Oracle Database Cloud Migration

Welcome to Oracle University's lesson Oracle Database Cloud Migration. My name's Nicholas Cusato, and I'll be your host. In this lesson, we're going to start by taking a quick look at the reasons for migration to Oracle Database Cloud Services, portfolio, migration strategies, and then migration process.

Customers regularly migrate their databases, which include the following-- upgrade to major new database versions, or maybe it's a move to new hardware platforms, or even a re-platform to the Cloud.

Every database migration requires work and processes, such as migration of the data or configuring of networking and client tools to connect to the new database or verifying of compatibility of apps and tools, as well as testing of connectivity, apps' performance, and many, many more.

Oracle Database Cloud Services range across the spectrum with service additions and automation levels to meet whatever your SLAs are within easy roadmap. If and when your businesses want more advanced technology and automation capabilities, these service offerings are some of the target opportunities for your migrations to meet the different business requirements and reasons for migrations.

Migration strategies with downtime or with continuous operations, we know that some businesses may only require business hour operations or eight hours, five or six days a week. They can schedule an outage for the migration. And this opens up different methods of migrations.

Other businesses require 24/7 operations, and the applications must be available during the migration. We'll talk about these methods of migrations as well. Database migration to the Cloud is to assess or evaluate, then choose the product for migration and migrate.

Cloud Pre-Migration Advisor Tool, or CPAT, helps determine the suitability of migrating an Oracle database instance to one of Oracle's cloud offerings. CPAT will assess your source database instance, checking for potentially problematic content and other factors that could impede a successful migration.

There are a number of other tools that I'm not going to go into too much. However, one interesting point is the differentiation between them. For OCI Data Migration, DMS, our focus is ease of use. We have focused on zero downtime migration. You can choose to use either one.

In the Zero Downtime Migration, ZDM, you'll get more fine-grained control. You have more options to choose from. Right now, ZDM support non-autonomous targets and also migration to ExaCC. So you can install ZDM on-premises and do basically an on-premises ExaCC migration. This is not possible with OCI Database Migration.

The other tools have their own capabilities as well. You can think of basically the three levels where you have the EMS being the most abstracted one, which is based on ZDM. And the ZDM itself is based on the database tools.

Migrate from Oracle-supported source versions to new Oracle target versions, we'll go into more depth in the next few lessons. But understanding the supported source and target is important for a successful migration.

Here's a quick look at the tools used for the migration process. In the following modules, we'll be diving into the tools and when to use. But for now, we are looking at the general process, where the tools fit in. You can use the Migration Advisor to make a decision and Cloud Pre-Migration Advisor Tool, or CPAT, for playing things, such as, is my data supported for migration?

CPAT is now integrated in DMS and validates the source database for compatibility as part of the migration to move databases, that is, for data migration, you use the data migration service and to move your applications and VMs to the cloud, which is application migration, you use OCI application migration.

The GoldenGate Veridata tool is used to validate data by comparing source and target databases. This concludes our lesson, Target Database Environment in OCI. Thank you. And I hope you learned something useful.

9.2 Database Upgrade Best Practices

In this session, we're going to talk about database upgrade best practices and show you the keys for a successful database upgrade.

So before you start your Oracle database upgrade, you need to be prepared for it. Choosing the target release, and right now, the target release that we recommend you go to for any Oracle database upgrade is the 19c. Why? Because 19c is currently the only long term support release we have in Oracle.

A long term support release means that we're giving five years of premier support plus two years of extended support, at least two years. Can be even more, but right now the 19c support will only end by April 2027. So the next long term support release will be the 23c. 23c will be releasing in 2023, so you have four years, from 2023 until April 2027, to do your migration from 19c to 23c. So it's a good time for doing an upgrade.

And another reason why you shouldn't go to any other versions, if you shall, if you check this lifetime support policy muzzle, you will see that all the previous releases, the support life is ending very soon or already ended, like 18c, where the latest CPU cycle for 18c was in April 2021. So no more patches for 18c.

So 19c, as you can see here again, it's going until April 2027, and Oracle 21c is an innovation release, not a long term support release, meaning we will only have two years of premier support on it. So only go with 21c if you want to play with the new features on your lab environments or dev or tests, never in production environments.

So after deploy your base release for 19c, it's time to apply your latest RU on top of it. Always go with the latest Ru. And remember that in 19c, you can have up to three user pdbs without requiring any extra license.

You should avoid plan regression after the upgrade. Even though this is after upgrade, you must be prepared before they operate for this. One feature that we recommend you to use to be prepared is SQL Plan Management. With SQL Plan Management, you can get all of the SQL queries you have in your system before upgrading and forcing a plan to run on those SQLs.

So after you get your system upgraded, you won't have any plan regression because all the SQLs will have the same plan as before. So SQL Plan Management is a way to force an SQL plan so you don't have any regression, any unexpected behavior in the upgraded system.

Oracle also offers you some performance stability prescription. I won't go through all of them. There's many you can use. SQL Plan Management, SQL Tuning Advisor, SQL Patch, or even Database Replay, which is part of a real application testing to simulate our production workload in the non-prod system and compare the performance. So keep in mind that Oracle offers all the kinds of stuff to guarantee that you won't have any performance issue after you upgrade your database.

The statistics are very important to think about before upgrading in your system. The reason being, good rating tests are very dictionary intensive tests. So you should take dictionary and fix said objective stats before running the upgrade in your database. If you go with AutoUpgrade, which is our recommendation, you must take it at least seven days in advance or the tool will take it for you.

As you can see in this graph, if you collect the stats before running AutoUpgrade you can save some time. This allows the CPU utilization of an upgrade of a cdb with 52 pdbs. By gathering dictionary and fixed objects statistics in advance, you can save some time, in this example, 12 minutes.

Now that we're prepared, let's go with the next step, the upgrade. Let's go ahead and upgrade. The very first recommendation that I can give is that you should always use AutoUpgrade to operate your database. We have put so much new functionality into AutoUpgrade that you're really missing out if you use any of the alternate methods. You can use AutoUpgrade to upgrade a single database or multiple databases in just one command, which really enables you to do end-to-end automation with AutoUpgrade.

Let me show you how easily you can upgrade a database in this little demo. I have my config file. It's just a simple text file which has information about the source and target Oracle Home. And then, I need to specify the side of the database that I want to upgrade.

Now, I start AutoUpgrade in Analyze mode. I specify the location of my config file and I use Mode Analyze. This will connect to the database to see whether or not that specific database is ready to be upgraded. Any critical errors will be reported and the upgrade cannot proceed.

I can use the lsj command to see how far the analysis is progressing. Currently, one out of 79 checks is remaining. Once the analysis completes, we can investigate to log files to get more details about the analysis. We have log and HTML files which are easy for humans to read. But we also have XML and JSON file, if you want to integrate AutoUpgrade with other tools.

If you look in one of the log files, you can see that, for instance, there's a check about dictionary stats. But don't worry, because AutoUpgrade has a fix available, meaning it can fix it for you. You don't have to worry.

So let's go ahead and upgrade the database, and we'll use ddeploy mode for that. Here, we're speeding things up a little so that we don't have to sit and wait for the database upgrade to progress. We can see that AutoUpgrade is now fixing things in the database. Now it's draining, meaning it restarts the database in upgrade mode. And now the actual database upgrade has been started and is currently at 61.

If you want even more details, you can use the Status command. And here you can see the timings of each of the faces. You can see where all the log files are stored and you can get information about each database that is currently being upgraded. After a while, the job completes, meaning that the database has been upgraded and all the post operate tasks has been executed as well. So I reset the environment and connect to the database. When I query v$instance, you can see that the database has now been upgraded to 19c.

So it was actually quite easy. It only took two commands, an Analyze and a Deploy, to do the entire database upgrade, including all the pre-upgrade checks and post upgrade checks.

You should always get the latest version of AutoUpgrade from My Oracle Support. Even though you can find a recent version of AutoUpgrade in your RDBMS Admin folder, you should always go to My Oracle Support to get the latest version. We really push our changes so fast that even though you have a recent release update, it might be that we have already created a new version that you can get from My Oracle Support.

You can use the version option on the command line to see which version of AutoUpgrade that you're currently using. If you scroll a little down, you can see that you can use this version of AutoUpgrade. This is AutoUpgrade version 21.2. You can use this version to upgrade not only to database 21 but also to database 19c, 18, 12.2. AutoUpgrade is fully backwards compatible.

You should think about your fallback plans. My recommendation is that you use guaranteed restore points and Flashback Database as your primary fallback option. Flashback Database is reliable and super fast. It's the best way and the fastest way to flashback or to fall back a database upgrade. When you're using the AutoUpgrade, you don't have to worry, it's built into the tool. AutoUpgrade will automatically set a guaranteed restore point and you can flashback the database with AutoUpgrade as well. If you're on Standard Edition you have to use another method because you're not allowed to use Flashback Database. In that case, I would recommend that you look into partial offline backup.

Something to be aware of with Flashback Database, you cannot use it after go-live. Once users have been allowed to connect to the database and enter new data into the database, you cannot do a Flashback Database because that would mean data loss. If you need to fall back after go-live, you have two options. Either use Data Pump or a database downgrade. Either method you choose, be sure to test it before you do the production upgrade.

Also, you should think about when you want to raise the compatible parameter. Our recommendation is that you do it 7 to 10 days after upgrade. It could be the following weekend after the upgrade. This will give you enough time to see whether there are some critical issues that could cause you to start a database downgrade, because as soon as you change the compatible parameter you can no longer do Flashback Database or do a database downgrade. So only raise the compatible parameter when you are sure that you don't want to fall back.

Also, when you change the compatible, you should always set it to the default of the release that you're upgrading to. When you're going to Oracle database 19c, the value should be 19. It doesn't matter whether you're upgrading to 19.13 or 19.12, always set it to the default, being 19.00. If you don't follow this recommendation, you won't break anything, But if you have multi-tenant databases, you have container databases, it will make your life so much easier if you have a uniform compatible setting in your entire environment. And be sure to only change the compatible parameter when you're upgrading. You don't have to change it when you're just patching.

If you cannot take the additional downtime it requires to change the compatible parameter after the upgrade in the following weekend, for instance, you have to change compatible immediately after the upgrade. But be aware, if you do that, you cannot do Flashback Database and you cannot do a database downgrade. But in some situations, additional downtime is simply not acceptable. And in that case, you can use AutoUpgrade to change the compatible parameter immediately after a successful upgrade simply by specifying these two parameters in your config file.

Now let us talk here on post upgrade. What's important here? First of all, statistics. Keep your optimizer statistics when you upgrade. There's no reason to regather stats. And even if you would regather, keep in mind, it will refresh stale and missing stats but it won't change your existing statistics unless they're marked as stale.

System stats that are on the characteristics of the system, CPU speed, I and O throughput, et cetera, it may sound like a good idea to gather such, but generally it's not our recommendation. Real world experience shows that it causes more troubles than gains. If you accidentally gathered system stats, get rid of them again by using dbms, underscore stats.delete, underscore system, underscore stats.

Only on some Exadata systems that does a pure data warehouse load. It's considered something to investigate, but even then it requires a significant testing effort.

What are fixed object stats? It is statistics on certain memory structures, the x$ tables. They are not tables but memory structures. For instance, there x$bh, which is metadata about the buffer cache. It is of utmost importance that you only gather fixed object stats when the system is warmed up, for example, it's been used by the application. Right after a database restart or upgrade, the buffer cache is mostly empty and if you then gather fixed object stats, you can metadata above the buffer cache, which doesn't match the reality when the system is running your important workloads.

Generally speaking, the fewer parameters, the better. Obviously, you must have some parameters, like db underscore name, location of control file, size of SGA, et cetera. However, the more you add, the more you drift away from the default configuration. And we, at Oracle, run as close to the default configuration as possible in our dev and test environments, in our Cloud and our own workload. We also encourage customers to do so. Hence, the more you get away from the default, the more likely you are to hit edge case issues.

Also, underscores and events should be evaluated during every upgrade project to check whether some can be removed. They must be removed at least during the upgrade, but afterwards is a good opportunity to see if you can get away from using the underscores and events, for the same reasons as the regular parameters. Having said that, you should always adhere to the recommendations of application vendors. If a MOS note on EBS tells you to set a parameter, you should follow that recommendation.

So it's time to wrap up. Let me wrap up your key to successful Oracle database upgrades. You start with downloading and installing Oracle 19c from eDelivery.oracle.com. Then you take the newest release update, RU. Then, as we explained before, you take the most recent AutoUpgrade, always the most recent one. And finally, put a lot of emphasis on performance stability. Use SQL Plan Management, use the SQL Tuning Advisor. Use if you have a license for Real Application testing. If not, you may use the Cloud to evaluate it.

Thank you for your attention and we hope that you have a successful database upgrade to 19c. Thank you.

10. MySQL and NoSQL

10.1 Describe Heatwave MySQL

Let's now talk about MySQL HeatWave. We'll start by going over MySQL HeatWave background, a features overview, and then security and ease of use. MySQL is the number one open source database and the second most popular database overall after the Oracle Database.

MySQL also received the DBMs of the Year Award from the db-engines.com. According to a recent stack overflow survey, MySQL has been around for a long time and remains the number one choice for developers, primarily because of its ease of use, reliability, and performance.

MySQL is used by the world's most innovative companies. This includes Twitter, Facebook, Netflix, and Uber. MySQL's popularity extends across industries, such as finance, manufacturing, telecom, and government. It is also used by students and small companies.

MySQL became very popular as part of the LAMP stack, Linux, Apache, MySQL, PHP, Perl, Python for web and e-commerce applications. Finance companies are using MySQL to deliver next-generation digital payment processing applications and fraud detection.

Manufacturing companies are using MySQL to deliver IoT platforms that monitor and control appliances, devices, and other products. MySQL is easy to use, performs well. And what our users say is we love it. It just works.

Despite its popularity, developers and DBAs face challenges using MySQL or MySQL-based offerings from other cloud providers. MySQL is optimized for OLTP. But it's not designed for analytic processing.

As a result, customers must run two separate databases for transaction processing and analytics. First, all data must be moved from MySQL to the analytics database through a complex and time-consuming extract, transform and load, or ETL process.

By the time your data is available in the separate analytics database, it's already stale. So you don't get real-time analytics. Additionally, security and compliance risks increase as data moves between data stores. To top these challenges off, your cost increase when using two different databases plus the ETL process.

What's better than two databases? One, MySQL HeatWave provides a single unified platform for transactional and analytics workloads. HeatWave is a massively parallel high-performance in-memory query accelerator that increases MySQL performance by orders of magnitude for analytics and mixed workloads.

With MySQL HeatWave, customers run analytics on data stored in MySQL databases without the need to ETL. Not only does MySQL HeatWave eliminate the need for a separate analytics database, but as we'll see in a second, it is also faster than comparable database services at a fraction of the cost.

Changes made by MySQL transactions are replicated in real time to the HeatWave cluster and become immediately available for analytics queries, thus enabling real-time analytics. Since data isn't transferred between databases, the risk that it could be compromised is eliminated.

HeatWave is designed as a MySQL pluggable storage engine, which completely shields all the low-level implementation details from customers. As a result, existing MySQL in Amazon Aurora-based applications work without any modification, seamlessly accessing HeatWave through MySQL using standard connectors.

MySQL HeatWave is the only MySQL cloud service with a built-in massively scalable real-time query accelerator in a fully automated in-database machine learning engine. This service overcomes the limitations of traditional data warehouse, analytics, and machine learning environments that use periodic long-running ETL batch jobs to refresh the data.

MySQL HeatWave provides a unified MySQL Cloud Database Service for transactions, real-time analytics, across data warehouses and data lakes, and machine learning without the complexity, latency, risk, and cost of ETL duplication.

We're facing a data deluge that is very challenging to manage for organizations. Although databases are systems of record, they're not very well suited to store the data coming from various devices, social media videos, IoT sensors, and so on.

Data is voluminous and changes rapidly. Loading into the database takes time and resources, so databases aren't always aligned to have rapidly changing data. The mass of data you put in files is much larger than what is designed for the database.

Over 80% of that data is in files and massively growing. 99.5% of collected data remains unused due to the lack of time, resources, and expertise to process different data formats across different data sources.

This is precisely where the new MySQL HeatWave Lakehouse becomes extremely useful. MySQL HeatWave Lakehouse enables customers to query half a petabyte of data in multiple formats, such as CSV, Parquet, Aurora, Redshift export files in the object store, allowing customers to leverage the benefits of HeatWave, even when their data is not stored inside a MySQL database.

Customers can query transactional data in MySQL databases, data in various formats in object storage, or a combination of both using standard SQL commands. Customers can query up to 500 terabytes of data with MySQL HeatWave Lakehouse and the HeatWave cluster scales to 512 nodes.

Querying the data and object store is as fast as querying the database, an industry first. Data is processed, transforms into the HeatWave in memory optimized hybrid columnar format and the object store and not copied to the MySQL database. That means that customers can take advantage of HeatWave, also for non-MySQL workloads.

With MySQL HeatWave Lakehouse, MySQL HeatWave provides one cloud database service for transaction processing, analytics across data warehouses and data lakes, and machine learning without ETL across cloud services.

MySQL users don't have an easy way to build machine learning models using data in their databases. And that's a big issue as the volume of data is dramatically rising and most companies want to take advantage of machine learning on that data.

Developers, data analysts, and data scientists typically need to, here again, move ETL data to separate machine learning products or Cloud Services to build and train machine learning models. This takes time, delays organization's ability to obtain quick answers and increases costs.

Security and compliance risks also increase as data is moved between systems. Additionally, developers and data analysts may need to learn new tools and languages. And it actually gets worse when using other databases.

HeatWave AutoML includes everything users need to build, train, deploy, and explain machine learning models with MySQL HeatWave at no additional cost. There's no need for a separate machine learning service. With native, in-database machine learning, and MySQL HeatWave, one can easily and securely apply machine learning, training, inference, and exploration to data stored inside MySQL HeatWave.

As a result, they can accelerate machine learning initiatives, increase security, and reduce costs. Save time and effort with machine learning lifecycle automation. HeatWave AutoML automates the machine learning lifecycle, including algorithm selection, intelligent data sampling for model training, feature selection, and hyperparameter optimization, saving data analysts and data scientists significant time and effort.

Aspects of the machine learning pipeline can be customized, including algorithm selection, feature selection, and hyperparameter optimization. All the models trained by HeatWave AutoML are explainable. HeatWave AutoML delivers predictions with an explanation of the results helping organizations with regulatory compliance, fairness, repeatability, casualty and trust.

Developers and data analysts can build machine learning models using familiar SQL commands. They don't have to learn new tools and languages. Additionally, HeatWave AutoML is integrated with popular notebooks such as Jupyter and Apache Zeppelin.

One of the many reasons MySQL HeatWave performs as well as it does is because of MySQL AutoPilot. Waves MySQL AutoPilot automates many of the most important and often challenging aspects of achieving high query performance at scale, including provisioning, data loading, query execution, and so on.

MySQL autopilot uses advanced machine learning techniques to automate heatwave, further improving performance and scalability and making it easier to use, saving developers and DBAs significant time.

It provides numerous capabilities for HeatWave and OLTP, including auto provisioning, predicts the number of HeatWave nodes required for running a workload, so DBAs and developers don't need to guess the optimal size of the cluster. Auto shape prediction continuously monitors the workload to recommend the right compute shape, allowing customers to always get the best price performance.

Auto data placement predicts optimal columns to partition data in memory based on recent queries, helping improve runtime. Auto query plan improvement learns various statistics from the execution of queries and boosts performance of the system as more queries are run.

Auto thread pooling allows the database service to process more transactions for a given hardware configuration, delivering higher throughput for OLTP workloads, and preventing the throughput from dropping at high levels of transactions in concurrency. And last but not least, we have auto error recovery, which is fairly self-explanatory. No other MySQL Database Service provides those capabilities.

MySQL HeatWave security and ease of use. MySQL DBAs tend to be overloaded with mundane database administration tasks. They're responsible for many databases, their performance, security, availability, and more. It is difficult for them to focus on innovation and addressing the demands of lines of business.

MySQL is fully managed on OCI. MySQL HeatWave automates all those time-consuming tasks so they can improve productivity and focus on higher value tasks. Developers can quickly get all the latest features directly from the MySQL team to deliver new modern app.

They don't get that in other clouds that rely on outdated or forked versions of MySQL. Developers can use the MySQL document store to mix and match SQL and NoSQL content in the same database as well as the same application.

MySQL HeatWave security means it is built on Gen 2 Cloud Infrastructure. Data is encrypted for privacy. Data is on OCI block volumes. Now let's talk about Gen 2 Cloud Infrastructure. Oracle Cloud is secure by design and architected very differently from the Gen 1 clouds of our competitors.

Gen 2 provides maximum isolation and protection. That means Oracle cannot see customer data and users cannot access our cloud control computer. Gen 2 architecture allows us to offer superior performance on our compute objects. Finally, Oracle Cloud is open. Customers can run Oracle software, third-party options, open source, whatever you choose without modifications, trade-offs, or lock-ins.

Data security has become a top priority for all organizations. MySQL HeatWave can help you protect your data against external attacks as well as internal malicious users with a range of advanced security features. Those advanced security features can also help you meet industry and regulatory compliance requirements, including GDPR, PCI, and HIPAA.

When a security vulnerability is discovered, you'll get the fix directly from the MySQL team from the team that actually develops MySQL. MySQL HeatWave is the only public cloud service built on MySQL Enterprise edition, which includes 24/7 support from the team that actually builds MySQL at no additional cost.

All of the other cloud vendors are using the community edition of MySQL. So they lack the Enterprise Edition features and tools. The following features are available and enabled by default in MySQL HeatWave.

MySQL enterprise scalability, also known as the thread pool plugin, data at rest encryption, native backup, OCI built-in native monitoring. You can also install MySQL Enterprise monitor to monitor MySQL HeatWave remotely.

MySQL works well with your existing Oracle investment, such as Oracle data integrator, Oracle analytics cloud, Oracle APEX, and more. MySQL HeatWave customers can easily use Docker in Kubernetes for DevOps operations. Finally, MySQL HeatWave is a fully managed database service that allows users to focus on their business.

The MySQL HeatWave user is responsible for logical schema modeling, query design and optimization, define data access and retention policies.

The MySQL team is responsible for providing automation for backup and recovery, the system that backs up the data for you. But in an emergency, you can restore to a new instance with a click. Database and OS patching, including security patches. Monitoring and log handling. Security with advanced options available in MySQL Enterprise Edition, and OS installation.

There are requisites for starting, such as a tenancy to sign into, compartment to store your resources, and a group with granted policies.

A quick review of things to remember when you study. MySQL HeatWave is fully managed. All the mundane stuff backup patching are done for you by OCI. MySQL HeatWave is the only fully managed Database Service that combines transactions, analytics, and machine learning services into one MySQL database.

MySQL HeatWave uses advanced security for regulatory compliance. It has the tools you need to handle GDPR, PCI, and HIPAA. It integrates with Oracle technologies. MySQL HeatWave works well with other Oracle technologies on OCI without extra work.

MySQL autopilot uses advanced machine learning techniques to automate HeatWave, further improving performance and scalability and making it easier to use. In this lesson, you should have learned about MySQL HeatWave background, features overview, and security and ease of use. Thanks for watching.

10.2 Oracle NoSQL Database Cloud Service Overview

Hi, friends. Welcome to the Oracle University module on Oracle NoSQL Database Cloud Service Overview I'm your host, Sarah. Let's get started. As part of this presentation, we're going to cover five different topics. We'll cover modern application challenges and an overview of the NoSQL Database Cloud Service. Next, we'll talk about use cases and features. And then we'll wrap up with the different development tools that we have available to create your application to run against the Cloud Service.

Today's modern applications face many different challenges. Applications are generating and ingesting large amounts of data at a very high speed. For example, IoT applications generate quite a lot of time series data that then has to be ingested. Application response time is critical to match users' expectations. For example, relevant product recommendations on a shopping website need to show up at the speed of every customer's click.

With today's global, competitive, and dynamic business environment, enterprises expect innovations to happen rapidly to keep up with the user's needs and expectations. Applications demand a database management solution that can be deployed rapidly and is easily maintained. The application data models may change constantly and dynamically to meet various business challenges. They may include structured, semi-structured, or even unstructured data. And these different data models need to be able to interoperate with one another.

Today's applications are expected to be always on. Customers expect non-disruptive fast services. On-demand scaling of compute resources is a must-have for dynamic application workloads during different times. For example, let's say we take a company that sells goods. We can expect there to be a peak in activity during the winter holiday season, so we would want to be able to grow the ability of the service during that time. Then, in off-peak hours, we would want to be able to reduce the services that are unnecessary. Businesses need a database management solution that addresses these challenges while keeping their operating costs to a minimum.

Introducing the Oracle NoSQL Database Cloud Service, which addresses many, if not, all of the challenges that we've presented on the last slide. The service as a whole is a serverless, always-on, fully managed solution by Oracle. Developers can focus on application development without having to worry about managing things like the server, storage, cluster, software setup, backup, and restore. All of those sorts of things are managed by Oracle.

It is also fully elastic. You just provisioned the throughput and storage capacity required by your application workloads. Resources are automatically allocated and scaled accordingly to meet the dynamic workload needs. The service provides predictable low latency for all types of application workloads, whether it is at its peak or at its minimum, offering a latency of less than 10 milliseconds.

Another feature is its flexible data models. It offers document, columnar, and key-value to really capture any kind of data format that you may have. These models can interoperate with each other using a single application interface. It offers developer friendly APIs and is fully integrated with popular application development tools. It comes with fully enterprise-grade security. It's very cost-effective. And lastly, and most importantly, it makes hybrid cloud or multi-cloud deployment with Oracle NoSQL database extremely easy. This enables enterprises to expand their business operations, open up new business potential and opportunities.

What do we mean by fully managed? Well, in this case, Oracle is responsible for the back-end software and hardware. Today's modern developers really aren't interested in what's going on behind the scenes. They want to be sure that they can get the resources they need and when they need them to put forth the best application possible. The Oracle NoSQL Database Cloud Service allows developers to focus on the application and who can use the application without having to worry about all of the infrastructure that goes on in the background.

Here are some of the common use cases that are used by the Oracle NoSQL Database Cloud Service. Let's just highlight a couple of these. For social media applications, NoSQL database is used for storing and querying user-generated data like comments, ratings, tweets, chat sessions, and more. For IoT, data is ingested at high speed from device sensors from different locations and stored in the NoSQL database. This data is consumed by downstream systems and analytic services to gain real-time insights.

For Customer 360 View, enterprises need to gain a comprehensive view of the customer behavior and preferences by aggregating data from a lot of different data sources. NoSQL databases can be used to enable Customer 360 service, delivering real-time customer reviews to business teams like sales, marketing, service, support, and even more. This contextual insight can help enterprises delight customers with a personalized experience.

For catalog data, information on products, people, and places can be stored in a NoSQL database, and queries can run against this information. The examples of catalog data are user accounts, product catalogs, IoT, device registries, bill of material systems. There's many of them out there.

For online gaming, the game console client depends upon a NoSQL database to store and deliver customized and personalized content like in-game stats, social media interaction, and high-score leaderboards. Online games require a very low latency response from NoSQL databases to provide an engaging, interactive game experience. User profile management and modern applications need to provide interactive, personalized user experience with complex service views based on user preferences, their moods, and other real-time parameters. Such applications need to access user profiles fast to render application UI for a real-time, customized user experience. Other common use cases are content management, social networking, real-time big data, online advertising, anomaly detection, and even more.

As of March 2021, the Oracle NoSQL Database Cloud Service is offered in 22 regions worldwide, and this list will continue to grow as new regions become available.

Many of these points have already been mentioned earlier in the presentation. However, let's chat for a few moments on what run anywhere means. Since we have both an on-premise database and a fully managed cloud service, we have a huge advantage over our competition. With the functionality of these things essentially being the same, this allows you to run your application anywhere.

So what does that really mean? Well, it means that you can run in any one's cloud environment. You can run in your own cloud environment. You can run on our cloud environment, or you can run on-premise. The same NoSQL application can be run in any of those locations. What if you have some data that you do not feel comfortable with it being run from the cloud? No big deal for Oracle NoSQL. You can run a hybrid setup, where some of the data is on-prem, and the rest of the data is in the cloud. You can use this using the same application.

We support different types of data models, and you get to select which one you want, depending upon what you need. First is the columnar schema, sometimes referred to as a fixed schema or relational schema. This is where you have columns defined, and then each of those columns has a defined data type. Then we have a document type of structure, sometimes referred to as schema-less, frequently referred to as a JSON table or a document table.

There are several different terms being used out there in the industry, but essentially, this is a JSON object. And JSON objects have the ability to be variable, as all of the records or rows can look a bit different. Lastly, we have a key-value. Key-value is often referred to as a key-value pair. You have a key, and then you have an associated value. So from a table standpoint, the table has two columns. Now, all of these different data models work with each other. So you can use the same SQL and APIs to work with any of these different data models. They all interoperate with one another.

We offer both encryption and authentication control. Authentication is covered by Oracle's Identity and Access Management Service, which lets you control what type of access a group of users has and to which specific resources they have access to. Oracle offers better and easier governance with capabilities such as compartments. A compartment is a logical isolation of resources for usage and billing. Additionally, we offer policies with simple SQL-like syntax that are extremely easy to create, manage, and update. We also offer encryption both at rest and in motion.

We are fully elastic, which means we have the ability to scale up or down. Earlier, we talked about allocating throughput and storage capacity for a NoSQL table. Scaling these capacities dynamically can provide a very cost-effective operation for running your application using the Oracle NoSQL Database Cloud Service. Let's look at an example of a NoSQL table that is created for a website that has online activities during weekdays.

Let me direct your attention now to the graphs on the right. The first graph on the top represents the activities from Monday on the left to Friday on the right. As you can see, the workload tends to increase from Monday through Wednesday, drops a little bit on Thursday, and again, it increases on Friday. Let's assume the support services does queries and insert operations to the table. The queries and the new data inserts vary every day based upon the workload chart.

For example, we see on Monday the queries, and inserts are the same. On Tuesday, we see insert operations are higher than the queries. Then the reverse is true for Wednesday. And finally, Thursday and Friday have slightly different operational patterns. So based on the queries and the new data insert operations for each day, the right amount write-and-read throughput can be provisioned to meet the demand. As you can see in the throughput chart, Wednesday and Friday require higher throughput to handle the higher levels of queries and inserts.

Now, looking at the last chart on the bottom, you can see the operating costs for each day fluctuates based upon the throughput provisioned according to the demands or the requirements of the workload. Businesses no longer need to purchase and maintain the infrastructure to support peak workloads. This avoids overprovisioning the hardware and paying more than what the workload actually needs. IT no longer has to size hardware based upon the server capacity or the OCPU.

So how do we get this fast, predictable performance? Well, we run on the Oracle Cloud using Gen 2 hardware. The Oracle Cloud is designed to provide enterprise customers with security, rock-solid reliability, and powerful management capabilities for large and complex deployments, all while beating industry performance and pricing standards. The Oracle Cloud infrastructure leverages the latest CPUs, GPUs, out-of-the-box networking, NVMe-SSD-based storage.

From a raw performance point of view, the NVMe solid-state storage is capable of millions read and write transactions per second. Unlike most cloud providers, the Oracle NoSQL Database Cloud Service is never oversubscribed, so each tenant gets predictable, high performance, and low latency.

To save on costs, different tables can be created for different parts of the business. In this example, each table can have a different capacity at different times based upon the workload. This picture shows the benefit of dynamic billing, which allows businesses to operate efficiently and cost-effectively.

At the end of the day, what is the Oracle NoSQL Database Cloud Service? And how does it help businesses? Well, it's basically a client-server architecture. Let's take a deeper look into that. On the client side, an application interacts with NoSQL drivers. We have drivers available for Java, Python, Node.js, Go, and other programming languages. We just rolled out a Spring driver using Spring data. The application runs database operations like inserts, updates, deletes, and queries against the NoSQL tables in the server side.

From the application developer's point of view, that's really all they need to know. It's extremely simple. NoSQL tables can be created in a matter of seconds. Developers can start right away to develop and deploy their applications. Businesses can focus on rapid innovations to better serve their customers' needs and expectations. Developers and IT don't need to manage any of the computing infrastructure.

The software updates, the high availability setup, all of the underlying computer resources and software maintenance are fully managed by Oracle to host the NoSQL tables. Database administrators manage the authentication and the roles and the privileges for accessing the NoSQL tables and the various application interfaces.

Let's touch upon how the drivers connect to the NoSQL tables and the various performances of those operations. Each table has two key components. The first is the data component, which consists of the table definition. We support many different data types, including integer, string, binary, long, double, JSON, records, and many others. And as you can see in the simple example here, each column can be defined to support a specific data type.

So the example that we have here is more of a fixed scheme type of approach. In this case, the primary key column is an integer. That's on the far left. The next two columns are strings, and the last column is a JSON data type. So we can intermix the different data types in the same row. The primary key is an index, and it's used to access the data.

For a more complex table definition, shard keys can also be defined. The purpose of shard keys is to distribute data across shards for efficiency and to store records with the same shard key in the same shard for a locality of reference and quicker access. Records that share the same shard key will be located very close to one another.

The second component of the tables is what we call capacity. The table capacity indicates the resource limits that are allocated to this particular table for its operations. Capacities can be divided up into two types. We have storage and throughput. Storage capacity is expressed in terms of gigabytes. This is the maximum amount of storage allocated to this particular table. Throughput captures the reads and writes that are performed on this table. Write capacity is expressed in what we call write units, and read capacity is expressed in what we call read units.

To discuss throughput provisioning, we need to talk about read units and write units and what those really mean. We have a definition of each of them up on the slide. And what I want to talk about briefly here is the notion of eventual consistency and absolute consistency. Eventual consistency is a consistency model used in distributed computing to achieve high availability, which informally guarantees that if no new updates are made to a given data item, then eventually, all accesses to that item will return the last updated value.

If you are in an environment where updates are occurring frequently, then quite possibly with eventual consistency, you can return what is often commonly referred to as stale data. So the basis of our read unit is eventual consistency. However, if you want to use absolute consistency and we have that option available, then basically, the charge for each I/O is 2 read units as opposed to 1 read unit. With absolute consistency in this model, you're going to be guaranteed that you will always read the most recently updated value.

10.3 Oracle NoSQL Database Cloud Service Overview - Part 2

Here are a couple features that I want to cover, that is time to live and our access via either API or SQL. Time to live functionality is basically auto-aging of data. We have it at the table level and at the record level, if you have a particular record that you would like to have aged out differently than what you set default for a given table.

Secondly, is our set of APIs and SQL. Within the APIs, there's a mechanism to perform calls within SQL. We have a very rich set of SQL, such that we can access JSON. We have complex filtering expressions with support for conjunctions and disjunctions, as well as SQL interoperability between schema-less and fixed schema.

In the Oracle NoSQL Database Cloud Service, you pay for the throughput and storage you provision on an hourly basis. The throughput must be provisioned to ensure that sufficient resources are available for you at all times. The cost of all database operations is essentially normalized and expressed in terms read or write units. It abstracts the system resources such as CPU, IOPS, and memory that are required to perform the database operations supported by the Oracle NoSQL Database Cloud Service.

The service does not charge you for the underlying system resources. It charges you for the read and write units. Here is a more detailed description of read units and write units. The different operations-- read write, update, and delete may use different units. For example, if you perform a write, it will primarily consist of write units. What if you perform a read? You guessed it, it will primarily consist of read units. Now, if you perform an update or delete, those operations will consist of a combination of read and write units.

This slide covers an example of provisioned throughput and the simple API used to increase or decrease capacities via a table request. In the parameters of table limits, we have set the number of read units to 2,000, write units to 100, and gigabytes of storage to 500. Now, let's look at how to modify those. To modify that such that you either want to dynamically increase it or decrease it, you would use the exact same API to do that.

In the example that we have on the screen, we are taking the 2,000 read units and reducing them down to 1,000. Every one of the table request calls is a DDL call to alter the information of the table. Lastly, please note that there is a limit of four DDL calls per minute within your tenancy.

Oracle NoSQL Database Cloud Service supports many common data types. Here is a complete list of the different data types that are supported by Oracle NoSQL Database Cloud Service.

When it comes to connecting to the Cloud Service, you are going to need a series of connection credentials. On the left-hand side, we list the different credentials that you need to have. You'll need two Oracle Cloud identifiers or OCI ID, a tenancy ID, and a user ID. Down there at the bottom of the slide, we give a generic structure of what an OCI ID looks like. There is an API Signing Key, a signing fingerprint, and optionally for additional security, you can have a signing key passphrase.

Within the Oracle NoSQL Database Cloud Service, we have different types of resources. We have three primary resources that you will have access to. Those are tables, rows and indexes. And when you think about that in the sense of a relational database, most people can very easily relate to that terminology of tables, rows, and indexes.

The next set of slides cover some of the various permissions that we have to cover the different resources. As mentioned before, we have resources on tables, rows, and indexes. So we have different sets of permissions that can be assigned to each one of those different types of resources.

This chart up here shows some of those permissions that are available for a table resource. For example, there's the ability to inspect, read, alter, create, move. Then what we have toward the right hand side of the chart are the different kinds of REST APIs you can call and what permissions those would need.

This next table here, structured the same way as the previous table, shows the permissions for rows, where we are showing you the REST APIs, and then the various cloud driver APIs that need those various permissions. And hopefully, the permissions would make sense to most people.

For rows, you have to be able to read the rows, insert the rows, and delete the rows. Inserting the rows also covers things like updates. We also have what is called an upsert operation, which is a combination of update and insert.

And finally here, we have the third resource, that is, indexes, presented the same way as the prior two slides, except this one is focused on the index resources. This slide shows the permissions for the NoSQL driver request. It shows the requests, the permissions, and the operation ID for each one of those. Hopefully, this gives you an idea of the different permissions that can be allocated and controlled, and what level you can do that on.

Schema-less and fixed schema can interoperate with one another. Let's take a look at the schema-less definition on the left. Here, we have a primary key and a JSON object, which we've named content. If we were to take that exact same structure and convert that to a fixed schema model, then we basically take that JSON object and convert it to a record. There's the record definition on the right.

We asked the same question to both types of data models. That is, we want to find all visitors to the website in November who are males between the ages of 24 and 30 years old. The bottom shows you the syntax from a SQL standpoint on how you query that based upon whether it's schema-less or fixed schema. And as you can see, the schema is 100% identical between those two different approaches.

We offer extremely rich secondary indexing. We use path expressions to access JSON data. It's very easy to use. We can index scalars, non-scalars, composites, JSON fields within a JSON structure. You can index just about anything that you have within your database record. This is very, very powerful and something that our competitors don't offer.

The next couple of slides show you some of the rich offering that we have in our SQL language. We've color-coded them with descriptions underneath the SQL to make it easier to look at and understand. On the far left, we have predicates, projections, and paging. On the right, we have group by and aggregates. For projections, you can do that with simple scalars or JSON document fragments. You don't have to display the entire JSON document. You can just display a fragment of it.

Here, we are showing you a WHERE clause and some of our string functionality. We're looking under the JSON field, content.demographic.gender for the values that start with a lowercase f. We can also sort. And then finally, at the bottom with the limit of the offset clause, we have the ability to do paging. The limit clause specifies the maximum number of results to return. The offset clause specifies the number of initial query results that should be skipped. So if you have a very long list, you can basically display some subsection of that list that starts partially in that list.

And, again, as we already mentioned, we also offer aggregate functions, time extractions, group by, which is there on the right-hand side, where we're showing the simple aggregates, min, max, average, sum, and count. Here, we're doing a year, and we're extracting the year out of a timestamp. We have the ability to, if you wanted to, extract the month, extract the day, hours, minutes, what have you. You can extract any of those. And here in this example on the right, we are also showing a group by.

Continuing on this general idea, in addition to insert and upsert, we have the ability to do shard local joins and regular expressions. You can insert a new row while upsert is used for updates. These all work well with the different data types and the nested data structures. The example on the left shows SQL that updates a row, and a simple put operation to insert a new value into the map data type based upon the cookie ID.

The SQL on the left is doing a lot of different things, so let's break it down. We've got the set in there, which would be more of an update kind of operation. We have the add in there, which is more of an insert type of operation. Then we have the Remove in there, which is going to do almost the equivalent of a delete. It's going to remove an element of an array. And then we have a put in there, which will put new arrays into an existing document. All of this combination is done via a single SQL statement.

The example on the right shows operations on nested tables. This is a way to create joins between what we like to call parent and child tables, or sometimes referred to as nested tables. The nested tables clause is equivalent to a number of left outer join operations centered around the target table. In this case, the target table is the shopping cart. A target table may have ancestors or descendants, which can go to unlimited depths.

If data needs to be projected from the parent. The ancestors clause is used in the query. If the data from the child tables need to be projected, then the descendant's clause can be used in the specified query. The beauty of the parent-child table structures is each table is treated like an individual table. If you're retrieving data in the parent, then you don't necessarily need to access the child. Or if you want to retrieve data from the child, you don't necessarily need to access the parent.

NoSQL supports the standard GeoJSON RFC 7946 structure. Each GeoJSON object has a type and a set of coordinates. We have points, lines, and polygons. With GeoJSON, we have four different functions to look at relationships between these different types of objects. We have Geo intersect, Geo inside, Geo within distance, and Geo near. We also offer a Geo distance. A Geo is geometry to determine the type of geometry that a particular path an object will take.

On the right, we offer ID Generation. Identity columns are a way to auto-generate values. You define the start point and the increment. The cache determines the number of values generated at any given point in time. In the example above, we have cache 1,000. So we're going to create 1,000 entries at a time and then cache those. There's also the ability, if you want, to recycle after you hit the max. So we're not showing the recycle clause up here. But we also have that functionality.

Switching gears a bit to IDE plugins, we have plugins available for Eclipse and IntelliJ, which are very popular. Here are the three different approaches to access the Oracle NoSQL Database Cloud Service for testing, development, and deployment. We have a UI service, a Cloud Service production, and a local standalone client server, NoSQL simulator.

Through the UI console, a standard console interface provides basic functionality. You can do things like create tables, insert rows, update rows, select rows, and basic query functionality. You can't do absolutely everything through the UI, but it's intended to give you a flavor of the Cloud Service, get you up and running quickly, and help you develop some basic understanding.

In the middle, we have the primary approach, which is using the Cloud Service in a production environment. You have your application and have compiled in that application your driver. You can use that for deployment, for development, for production. That's going to give you the entire functionality. And then on the far right, we're showing you our NoSQL Cloud Simulator. That's a standalone single process kind of environment that simulates the cloud.

The beauty with the NoSQL Cloud Simulator is that there's nothing you have to pay for. You can test your application and make sure that your application runs without having to pay any of the expenses involved with actually running in the Cloud Service.

The UI console is a very simple and easy way to get an initial experience of the service. You don't have to download any drivers or develop any software. You can go through several of the screens to explore what's going on and test everything. The following slides will show screenshots of what the UI console looks like.

Once you log into the console, you can select your compartment. As a reminder, compartments are where you group your table resources together. This initial screen comes up, which lists the tables you've allocated in the selected compartment. It tells you what capacities you've allocated for your tables, including read units, write units, and storage.

We also have some additional information, such as when they were created. And then on the far left side, we have the status. As you can see, the first three tables are in the active status. And the table on the bottom named Foo was deleted some time ago.

We also have the ability to monitor through the OCI console. You can monitor all of your capacities and what they're doing. You can monitor those at different time intervals and take different snapshots. What we're showing up here is also throttling. We have this concept called throttling, where if you try to do more work in terms read units or write units than what you have allocated, then we will throttle those requests for you.

Let's say you want to do 100 read units and yet you're trying to do 200, we will throttle that 200 down to 100. When that happens, you could come look here and look at the monitoring. And then you could see if you're being throttled. That could be an indicator that, hey, you might be getting into a time period where you need to increase your read units or your write units. That's a very good indicator and a very good clue to tell you that maybe you need to go and adjust those different capacities.

Also for storage, the same sort of issues are there. So if you have under allocated the amount of storage that you need, you could come and take a look and see if there's been any throttling expectations, which would tell you to increase your storage appropriately.

How is Oracle NoSQL different? From a high-level standpoint, we have a seamless multi-model. We have key value, fixed schema, and schema-less all stored in a single data store. And as we saw a few slides ago, all of these interoperate with one another.

We also have tunable ACID. We have shard local full ACID. We have parent/child tables, which is an easy way to operate on multiple tables in a fully ACID compliant manner. We also offer adjustable consistency, including eventual and strong consistency. We offer a fully managed throughput provision, flexible, no lock-in service, so you can run as a fully managed cloud service in the Oracle Cloud, or if you want, you can run in someone else's cloud.

We offer a full hybrid approach as well, where if you want to run some of your applications on-prem and take a subcomponent of that application and run in the cloud, you can do that too. And that's a very powerful differentiator that we have, that our competitors do not have.

Thanks for hanging with me. I hope you enjoyed today's module on Oracle NoSQL Database Cloud Service.