***Chapter 6*: Understanding Database Choices on Oracle Cloud Infrastructure**

**Oracle Cloud Infrastructure** (**OCI**) provides a range of database services, and these are categorized as autonomous and co-managed database services. Despite the type of database service, these databases are designed to be robust so that enterprises can use them on a cloud scale. The Oracle Database service is backed by a robust infrastructure and is capable of handling mission-critical production workloads. This includes three availability domains and multiple regions. Currently, active redundancy can be implemented with features such as Data Guard configured to operate across all availability domains.

The networking that backs these database systems, along with every other system in OCI, is fully non-blocking and fully contextualized (multi-tenant with full isolation between networks). Speeds go from a minimum of 10 gigabits up to dual 25 gigabits per host, along with dedicated **InfiniBand** (**IB**) for cluster and storage networking for **Real Application Cluster** (**RAC**) and Exadata shapes.

Isolation is accomplished through off-box networking, which allows bare-metal hosts along with database systems such as Exadata to participate in virtual networks without needing virtual-switch software installed on the host.

At a very high level, there are three types of database systems offered by OCI. First, the bare-metal database systems come in a single-node shape. The second type of system is in the form of **virtual machine** (**VM**)-based shapes that support single- and two-node cluster operations. The third type of Oracle Database system in OCI is Exadata, which comes in quarter-, half-, and full-rack shapes.

In this chapter, we're going to cover the following main topics:

* Discussing OCI database choices
* Managing Oracle's Autonomous Database service

**Discussing OCI database choices**

Earlier, you learned that OCI databases predominately come in three different options. These database systems are protected by two- or three-way mirroring. In two-way mirroring, all the data gets mirrored, which means there are two copies of the same data. In three-way mirroring, you get additional protection from a bad disk sector in one disk and the failure of another disk. The database systems can be brought up as standalone or in a RAC cluster that is entirely configured and managed by the database service. In addition to RAC, Exadata systems are also available.

Because the systems are fully managed, they are **Maximum Availability Architecture** (**MAA**)-compliant.

Dynamic **central processing unit** (**CPU**) and storage scaling features are available, as well as the ability to upsize Exadata deployments across shapes. CPU core usage can be changed hourly to right-size the database system.

For security, there are several features and capabilities. These are a part of the identity services element, such as users, groups, compartments, and policies that can share or isolate the database system with fine-grained role-based controls. There is also networking security, implicit isolation, and off-box network virtualization—as well as security lists and on-host firewalls—in place.

Along with the policies and network security, there is a complete auditing service that tracks all actions of the users, whether through the **application programming interface** (**API**) or the console.

At the database level, encryption is on by default. Data at rest is transparently encrypted. Backups done to the object store are encrypted, and communications with the database service are encrypted by default.

Licensing flexibility is also available with **Bring Your Own License** (**BYOL**), so you can either use the database service with included licenses or bring existing Oracle licenses to the host for use on the cloud. All the database systems in OCI can be managed by tools such as Enterprise Manager, SQL Developer, and so on, just as with a regular on-premises database.

Let's look at the choices one by one. First, let's discuss VM database systems.

**VM database systems**

Database VMs offer a wide range of flexibility for the database service. Not all workloads need dedicated bare-metal servers. Customers ask for a cost-effective, easy-to-get-started, and durable database option that is well suited to a variety of workloads, ranging from **proof of concept** (**POC**) **development/testing** (**dev/test**) environments to production applications. VM-based database shapes can accommodate these workloads.

The database service on VMs is fully featured. While these instances run on VMs, the software can be configured with Standard, Enterprise, High, and Extreme editions. The database service on VMs is built on the same high-performance, enterprise-secure-grade, highly durable, and highly available cloud infrastructure used by all OCI services.

You will find the following two types of database systems running on VMs:

* A one-node VM database that is deployed using one VM
* A two-node VM database that is deployed using two VMs and then clustered using Oracle RAC

You will get a single database in a VM database offering. However, you can choose to allocate more computing power using different shapes of VM for these databases.

You can choose the storage size of the VM database system, but once deployed, you cannot change the number of CPUs allocated to this. You also have the ability to choose an older database when you select a VM database.

For the better **high-availability** (**HA**) architecture, OCI puts two VMs in two different availability domains when you select a two-node RAC cluster VM database. You can also configure Data Guard between these two VMs.

Let's look at the storage architecture of VM database systems. The OCI VM database system uses **Automatic Storage Management** (**ASM**), which in turn uses Block Volumes to mirror the data. ASM directly interfaces with the disks and inherently uses the triple-mirroring capability of the OCI block storage. The actual data and recovery data uses independent block storage volumes, where it's ASM's job to monitor these storage volumes for hard and soft failures. Disks are mounted on **ASM Cluster File System** (**ACFS**) or another filesystem providing maximum **input/output** (**I/O**). Some resources such as wallets are mounted in a common store along with database homes (binaries), but the **DATA** and **RECOVERY** areas are found within ASM. You need to have this storage architecture for VM RAC database systems. You can see a logical representation of this storage architecture in the following screenshot:

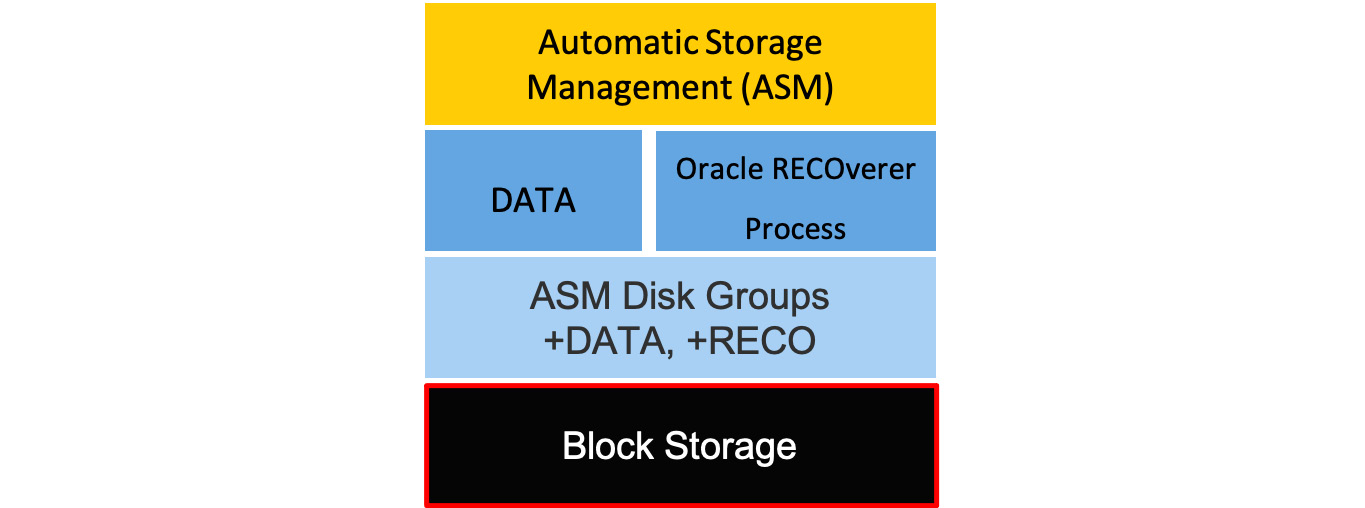


Figure 6.1 – VM database storage architecture

Storage is continuously monitored for any failures with the disks. These disks refer to **Non-Volatile Memory Express** (**NVMe**) and **solid-state drives** (**SSDs**). In the case of VM shapes, block volumes are used, which is NVMe-based, and multiple block volumes are brought in and managed the same way as these disks.

Any disks that fail will be managed. Space is reserved for rebalancing, so the amount of free space is calculated based on that reservation. Whenever the shapes list a maximum amount of usable space in **DATA** and **RECO**, these reservations for rebalancing are already considered.

The root user has complete control over the storage subsystem, so customization and tuning are possible, but the service sets these up by default in an optimal way.

Up to now, you have learned how VM database systems work on OCI for customers who don't have a requirement to run it on bare-metal servers. In the next section, we will talk about bare-metal database systems.

**Bare-metal database systems**

A bare-metal database system comprises a 1-node database system. OCI uses Oracle Linux 6.8 as the base operating system for this bare-metal box and uses locally attached NVMe disks for storage. OCI recommends this for test and dev environments because of its lower cost. You can restore the database using a backup onto a different server, in case a failure happens.

As with other services, this is also manageable via the OCI **REpresentational State Transfer** (**REST**) API console, but note that you can also use Enterprise Manager, Enterprise Manager Express, SQL Developer, and the database **command-line interface** (**CLI**) to manage this database offering.

Let's look at the physical architecture of a bare-metal database system. You can see a logical representation of this in the following screenshot:

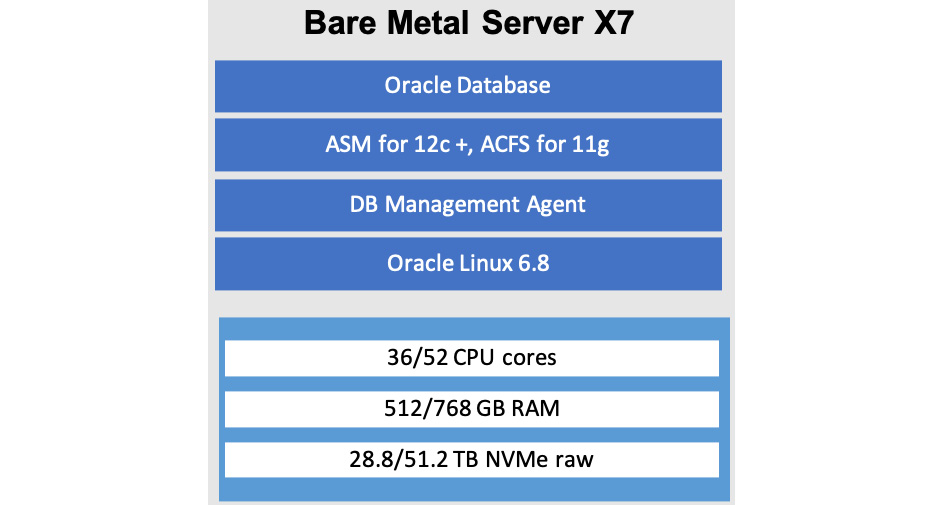


Figure 6.2 – Bare-metal database system: physical architecture

Now, let's look at the storage architecture of a bare-metal database system. You can see a logical representation of this in the following screenshot:

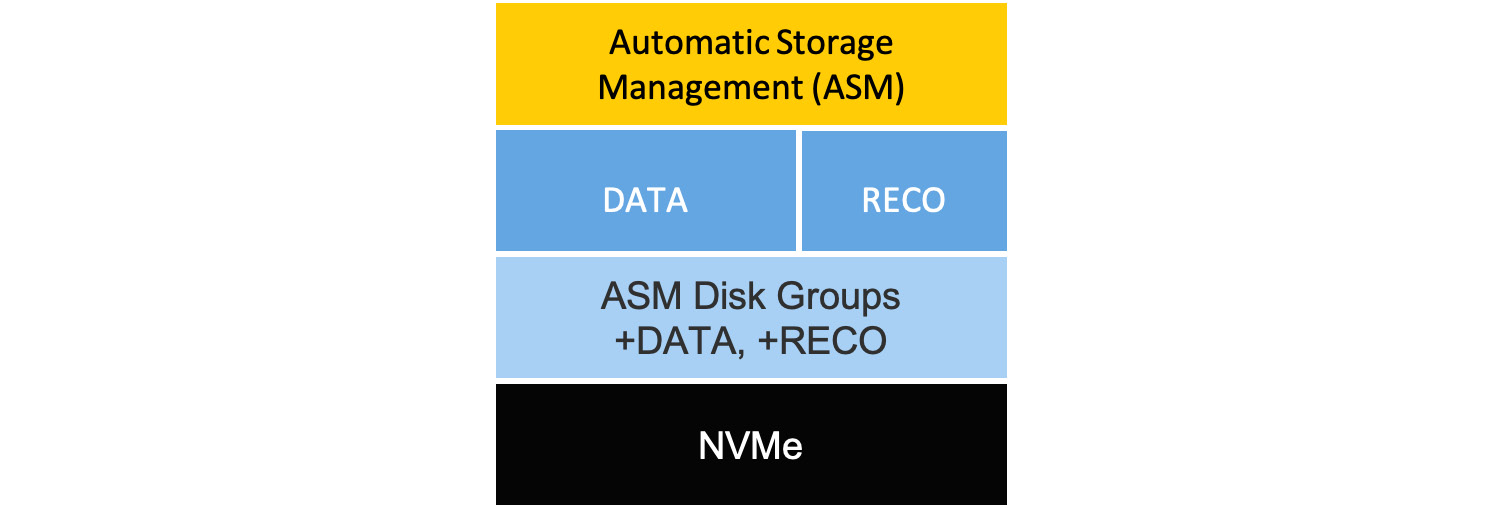


Figure 6.3 – Bare-metal database system: physical architecture

You can see that the ASM manages a large part of storage management. ASM manages the mirroring of NVMe disks, where disks are partitioned in such a way that one mirror is for DATA and one is for RECO. ASM also monitors the disks for hard and soft failures. It proactively puts the disks that get failed, have been predicted to fail, or are performing poorly into offline mode and performs corrective actions. On a disk failure, the database system automatically creates an internal ticket and notifies the internal team to contact the customer.

Let's look at Exadata database systems now.

**Exadata database systems**

In addition to VMs, bare-metal hosts, and bare-metal RACs and VMs, Exadata is also available on OCI. The Exadata systems are provided in three shapes, and all of them have all Exadata's advanced options turned on.

These are physical Exadata engineered systems that come with IB networking and scalable compute and storage nodes that can run on OCI without modification. Exadata database systems have complete isolation of tenants. Whenever partial shapes of Exadata are used, tenants are completely isolated.

Exadata on OCI gives all the features, performance, and capabilities of on-premises Exadata but with the flexibility of the cloud. All the installation, from systems to firmware, to operating system install and maintenance to patching, is managed by Oracle and presented as a public cloud service.

You can choose from three different Exadata database offerings—a quarter-rack, half-rack, and full-rack Exadata database, where you will get physical compute nodes and storage servers as part of the configuration you choose. It is very nice to be able to try out Exadata for your database needs on the cloud without having to deal with procuring a physical Exadata database.

Each compute node is configured to have root access to a virtual context running on the compute hosts. In case you need to install and run other software and tools, OCI has provided you root access to these nodes. This root access is only limited to the operating-system level, and you can't alter configuration for physical hardware, switches, **power distribution units** (**PDUs**), **Integrated Lights Out Manager** (**ILOM**), and storage servers. Not only do you have root access to the operating system, but you will also get admin access to the database, which you can connect to using either its private or public **Internet Protocol** (**IP**) address.

Database administration is the customer's job, whereby they are responsible for creating tablespaces, users, and so on. It is also the customer's job to control the recovery of the database in case of a failure, plus they also are in charge of the automated maintenance of the database.

Exadata DB systems on OCI benefit from having the **Identity and Access Management** (**IAM**) service, which helps create policies on which users and groups can perform actions on Exadata and database systems. You can have compartments and **virtual control networks** (**VCNs**) for these database services and either isolate or share them.

All the virtual cloud network capabilities and advantages are added to the Exadata and database systems. You do not have to use a public IP address for any of the instances if you do not want to. You can use a **virtual private network** (**VPN**) and FastConnect to connect to your on-premises environments. Because of the capabilities of OCI, we can have the application tier seamlessly running on VMs while the database is running on bare metal.

So, you can see that for different use cases, you have different database choices in OCI. In the next section, we will discuss Oracle's Autonomous Database service.

**Managing Oracle's Autonomous Database service**

Unlike other databases that you have learned about so far, where each database is individual, Autonomous Database is a set of OCI databases whereby each database has been built to serve a particular type of workload. For data warehouse workloads, which are an analytical type of workload, OCI provides **Autonomous Data Warehouse** (**ADW**).

However, if you are looking for a database to support more transactional data, then OCI has the **Autonomous Transaction Processing** (**ATP**) database service.

The third member of the family is **Autonomous JSON Database**. This is built for applications that have **JavaScript Object Notation** (**JSON**)-centric schema requirements. This service also offers document APIs and native JSON storage.

The OCI Autonomous Database offering is fully managed like the other Oracle services. As it is autonomous in nature, you have no responsibility to configure and manage any hardware and don't even need to install any software. The CPU and storage capacity are scalable.

With Autonomous Database, you don't need to do the following tasks:

* Back up the database
* Patch the database
* Upgrade the database
* Tune the database

Let's create an autonomous transaction processing database on OCI, as follows:

1. Sign in to the OCI console.
2. Open the navigation menu, select **Oracle Database**, and then **Autonomous Database**.
3. Click on **Create Autonomous Database**. You can see a sample screenshot here:

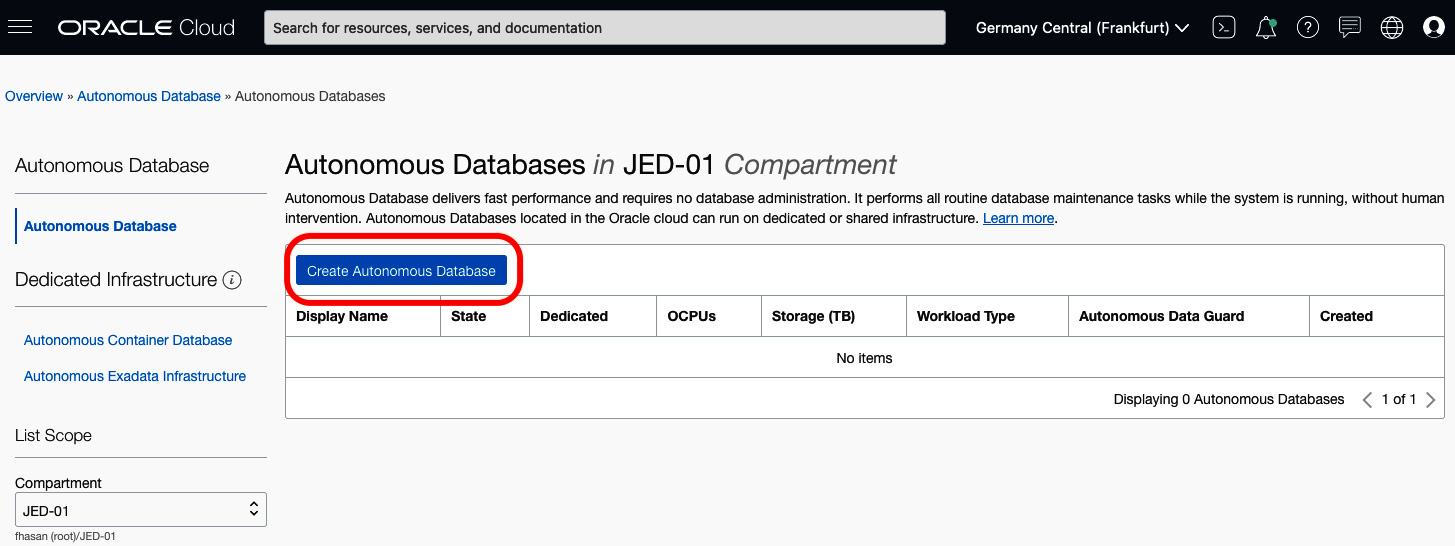


Figure 6.4 – Creating an autonomous database

1. Choose which compartment you want this database to be in.
2. Provide a display name and a database name.
3. From the **Choose a workload type** section, select **Transaction Processing**.
4. From the **Choose a deployment type** section, select **Shared Infrastructure**.
5. From **Choose database version**, choose the latest available database. At the time of this writing, the available option was **19c**.
6. Provide the **OCPU count** and **Storage (TB)** requirements. By default, both are set as **1**.
7. Check the **Auto scaling** checkbox if you want OCI to handle the scale-up operation when the load increases. You can see a sample output in the following screenshot:

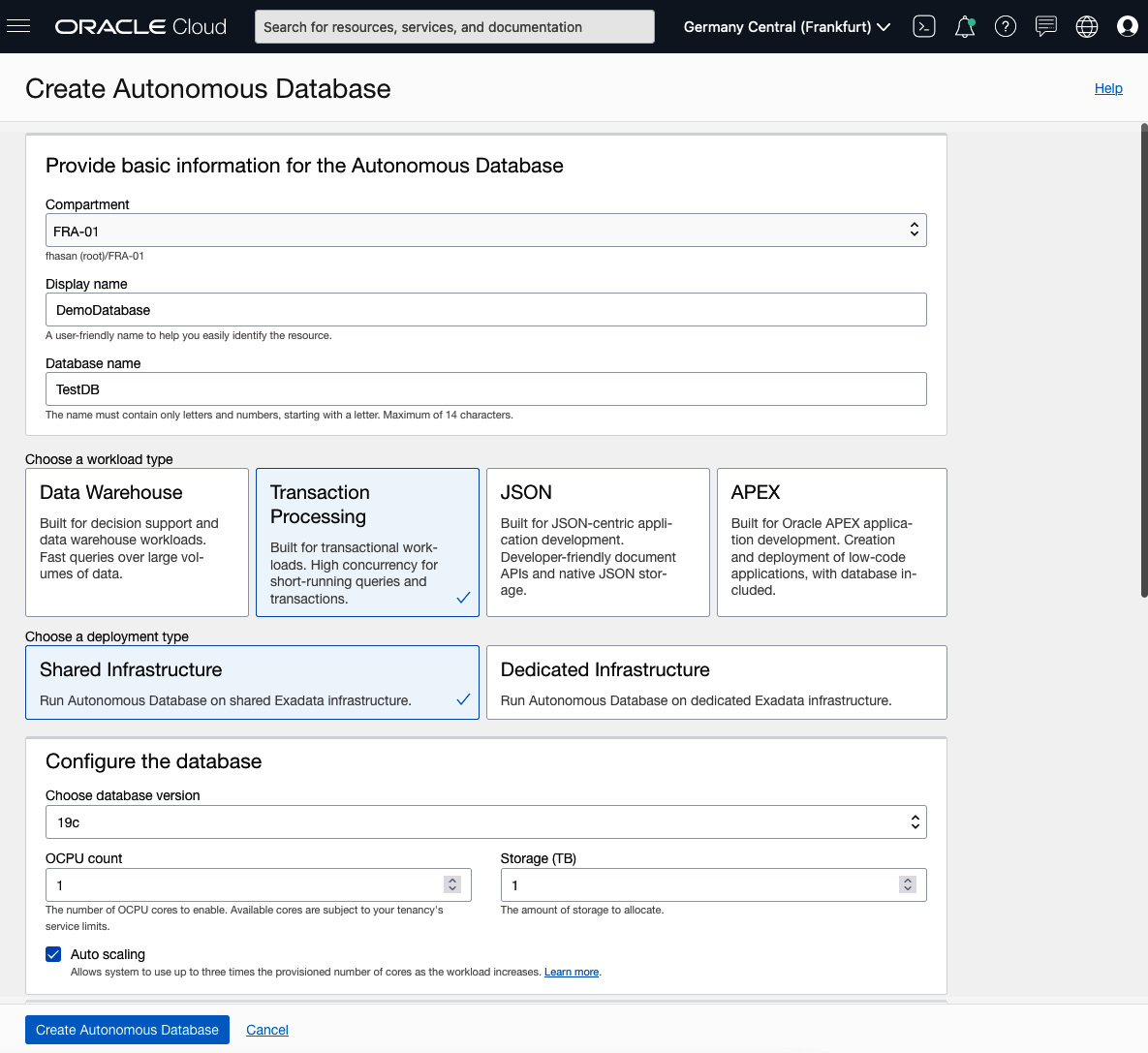


Figure 6.5 – Autonomous database creation workflow

1. Provide a password for the admin user.
2. Choose the **Secure access from everywhere** option under the **Access Type** heading. This is to simplify our testing option.
3. Select **License Included** as that is the easiest way to provision this database.
4. Click on **Create Autonomous Database**. You can see a sample output in the following screenshot:

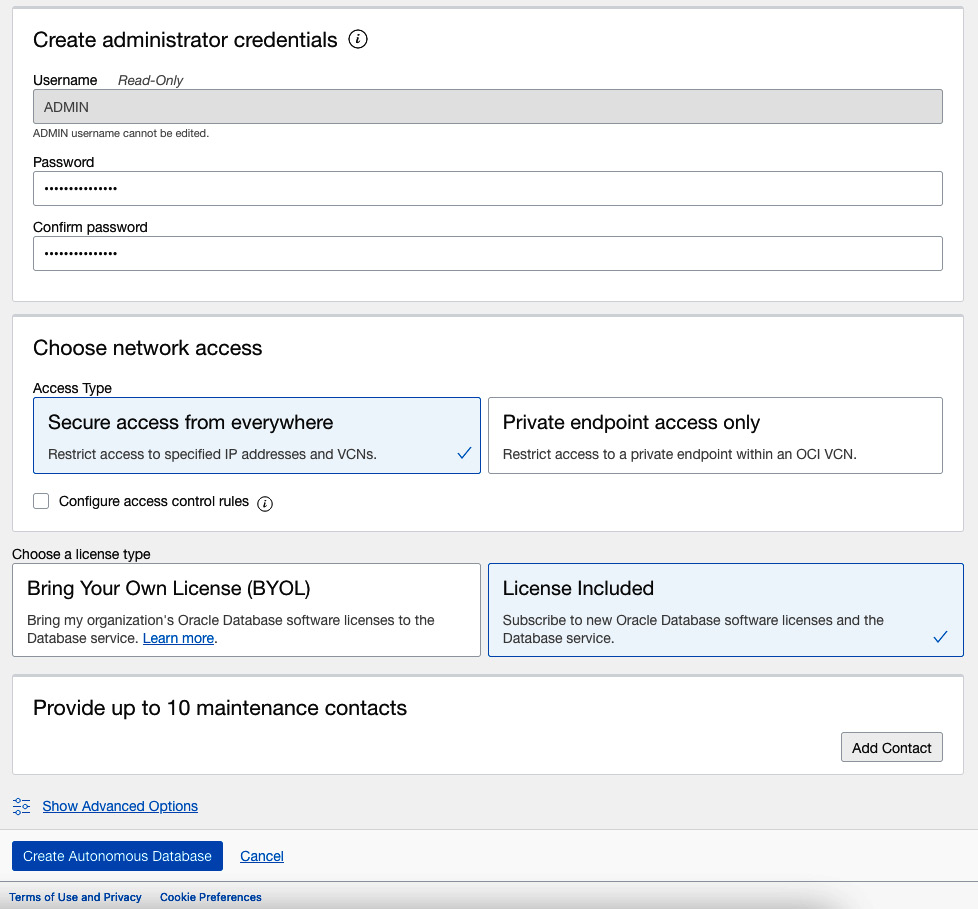


Figure 6.6 – Autonomous database creation workflow (continued)

1. Once it is finished, you will see the database available for consumption.
2. From the **Autonomous Database Details** screen, you can go to **SQL Developer Studio** to run a query from the web browser.
3. Click on the **Tools** tab and click on **Open Database Actions** from the **Database Actions** section. You can see a sample output in the following screenshot:

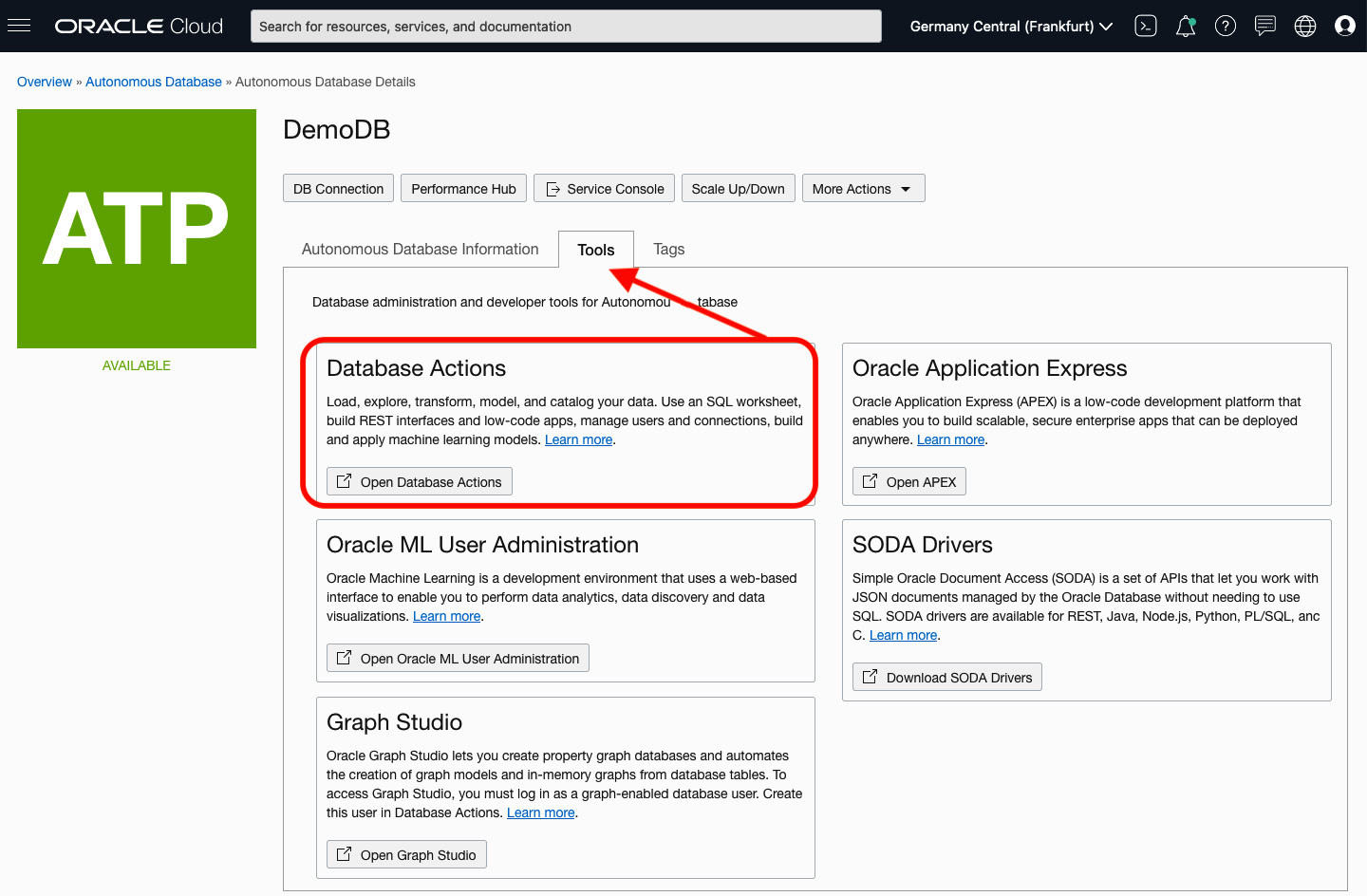


Figure 6.7 – Opening Database Actions

1. Provide a username and click on **Next**. Provide a password and click on **Sign in**.
2. Select **SQL** from the **Development** section. You can see an example output in the following screenshot:

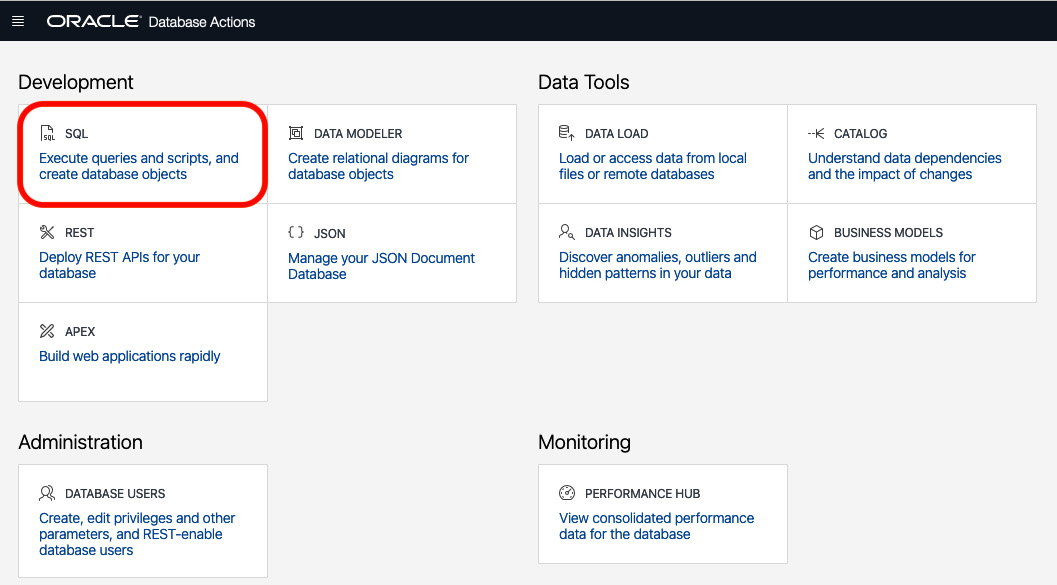


Figure 6.8 – Opening SQL Studio

1. Here, you can run any query or run any **Structured Query Language** (**SQL**) script to create database-related resources such as a table. You can see a sample output in the following screenshot:

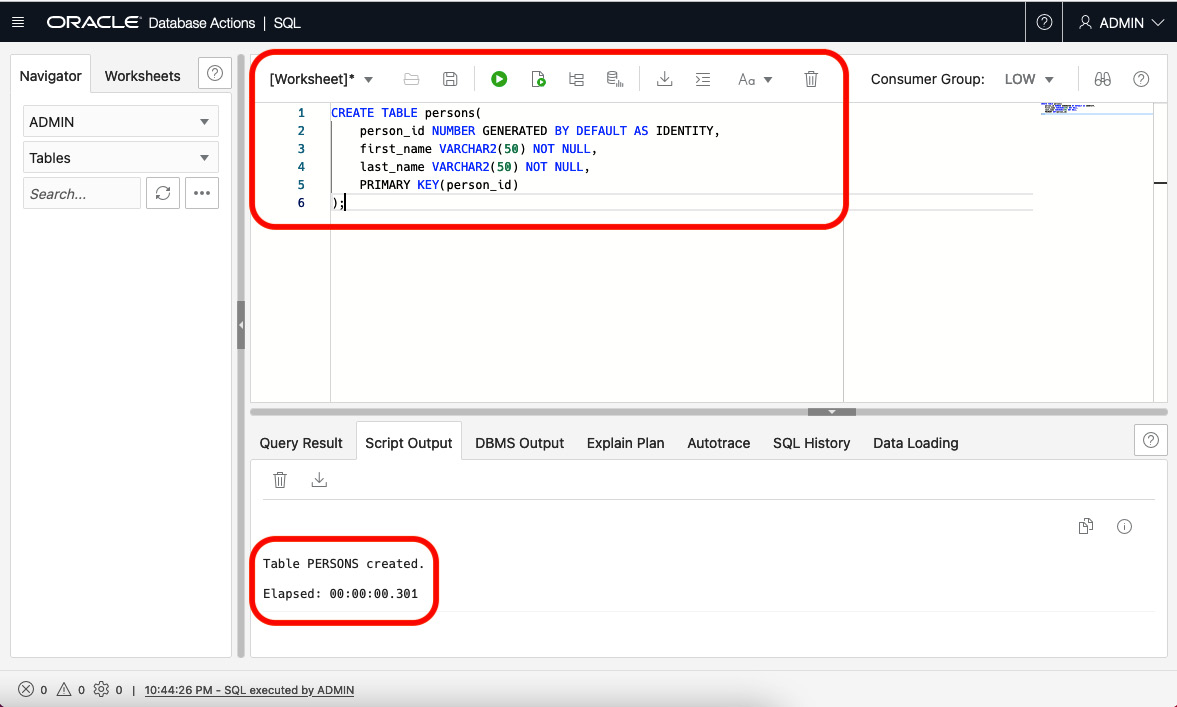


Figure 6.9 – Running SQL query

So, you can see how you can easily provision an autonomous database on OCI and use provided tools to manage the database as well.

**Summary**

In this chapter, you have learned about the various choices you have for adopting a particular database based on your workload type and how OCI can take care of database operations using the Autonomous Database service.

In the next chapter, you will see how you can build a cloud-native application on OCI.