LABORATORIOS DE ARQUITECTURA EN ORACLE CLOUD INFRASTRUCTURE

# Manual de consola — Agentes sobre Oracle: MCP y Agent Skills

Una base de datos autónoma, un servidor MCP y el control que de verdad acota a un agente

**Módulo** 06

**Primera pasada** 1 a 1,5 horas la primera pasada

**Costo aproximado** 0 USD con el nivel siempre gratuito

**Vía** consola de Oracle Cloud, pantalla por pantalla

Este documento reproduce desde la consola lo que el laboratorio automatiza con Terraform. Los dos caminos llegan al mismo sitio: el código sirve para repetirlo, la consola para entenderlo.

El laboratorio corre sobre un tenancy de prueba desechable. Las decisiones de acceso que aquí se toman no son una referencia de configuración segura para un ambiente con datos; cuando alguna se aparta de la buena práctica, el manual lo dice en el sitio donde se aparta.

# 1. Qué se construye y para qué

Este laboratorio conecta un agente de inteligencia artificial a una base de datos de verdad, lista exactamente qué puede hacer contra ella, e identifica cuál es el control que realmente acota el daño.

Es **autocontenido**: no depende de ningún otro laboratorio de la serie. Crea su propia base de datos y todo lo demás ocurre en el equipo del ingeniero. Con este documento y una cuenta de OCI alcanza para terminarlo.

Se construyen cuatro cosas:

1. Una **Autonomous Database** con endpoint público y mTLS obligatorio, creada a mano desde la consola.
2. Un **esquema de ejemplo** al estilo de un sistema heredado, con cuatro problemas puestos a propósito.
3. Un **servidor MCP** corriendo en el equipo, con su catálogo de herramientas inspeccionado antes de conectarle nada.
4. Un **usuario de base de datos mínimo**, que solo puede leer cuatro tablas.

## La decisión que el laboratorio ayuda a tomar

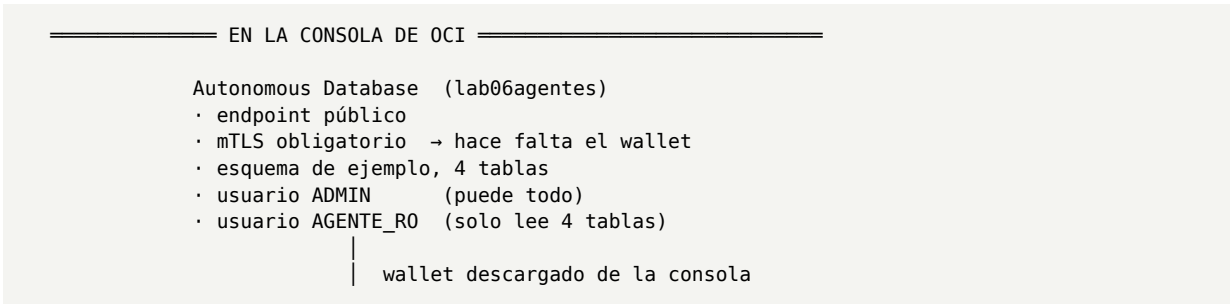
La pregunta con la que llega casi todo el mundo es «*¿puedo confiarle una base de datos a un agente?*». Es una pregunta que nadie sabe responder, porque depende de un modelo cuyo comportamiento no se puede auditar línea por línea.

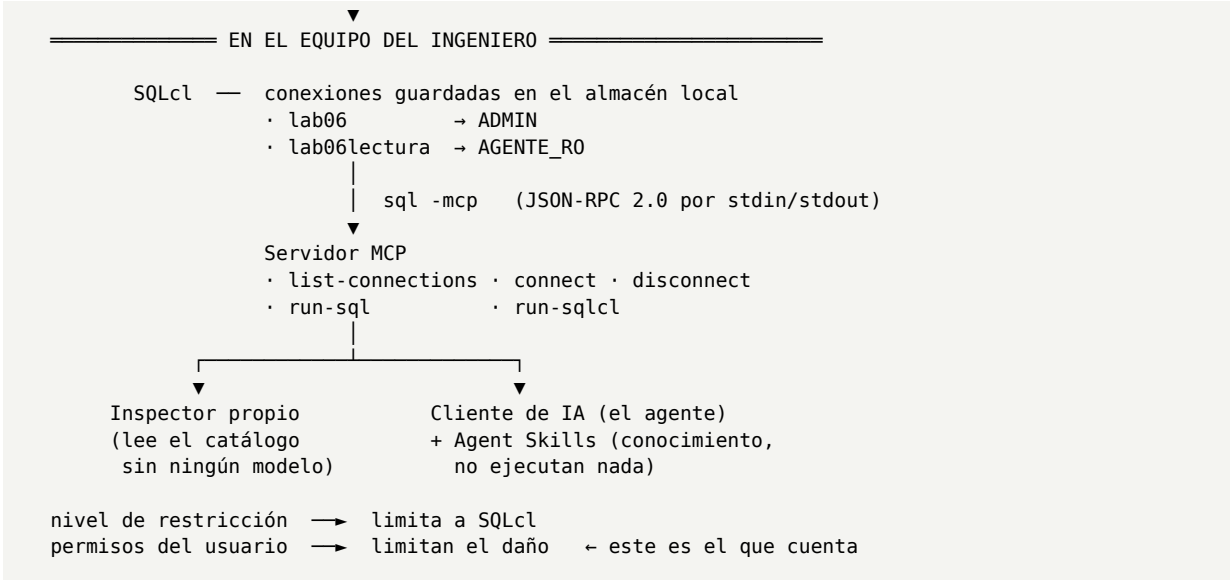
El laboratorio la reemplaza por dos preguntas que sí tienen respuesta en dos minutos:

- **¿Con qué usuario de base de datos se conecta el agente, y qué privilegios tiene ese usuario?**
- ¿Con qué nivel de restricción arrancó el servidor MCP, y quién lo decidió?

La primera es la que importa, y es la que casi nadie hace. El laboratorio la convierte en una demostración: se le pide al agente que borre datos, la base lo rechaza, y el agente lo reporta. No lo impidió el modelo ni el nivel de restricción: lo impidió un permiso.

## Arquitectura






## Dónde ocurre cada cosa

La mitad de este laboratorio no tiene pantalla. Conviene tenerlo claro desde el principio, porque es una de las cosas que enseña.

| Capítulo | Dónde se hace |
|---|---|
| 4 · Compartimento y políticas | Consola de OCI |
| 5 · Base de datos autónoma | Consola de OCI |
| 6 · Wallet | Consola de OCI |
| 7 · Cliente de SQL | Equipo del ingeniero |
| 8 · Esquema de ejemplo | Consola de OCI (Database Actions) |
| 9 · Servidor MCP y su catálogo | Equipo del ingeniero |
| 10 · Niveles de restricción | Equipo del ingeniero |
| 11 · Usuario mínimo | Consola de OCI + equipo |
| 12 · Cliente de IA | Equipo del ingeniero |
| 13 · Agent Skills | Equipo del ingeniero |
| 14 · Política de adopción | En una reunión, no en una pantalla |

No hay una pantalla en la consola de OCI donde se configure un servidor MCP. No la busque. El servidor MCP es un proceso que corre en el equipo de quien desarrolla, y las conexiones que usa están guardadas en ese equipo. Esa es exactamente la razón por la que el gobierno de estas piezas es difícil, y es parte del contenido del capítulo 14.

# 2. Equivalencias con otras nubes

Solo los servicios que aparecen en este laboratorio.

| Concepto | AWS | Azure | OCI |
|---|---|---|---|
| Base de datos relacional gestionada | Amazon RDS for Oracle / Aurora | Azure SQL Database | Autonomous Database (Serverless) |
| Agrupación lógica de recursos | Cuenta o unidad organizativa | Resource group | **Compartment** |
| Permisos declarativos | IAM policies | Azure RBAC | **IAM policies** (grupo · verbo · tipo · compartimento) |
| Nivel gratuito de base de datos | Free Tier, 12 meses | Servicios gratuitos, 12 meses | **Always Free**, sin vencimiento |
| Editor SQL dentro del portal | RDS Query Editor | Query editor del portal | **Database Actions** › **SQL** |
| Credencial de conexión descargable | Bundle de CA de RDS | Cadena de conexión | **Wallet** (mTLS, autentica al cliente) |
| Restricción de origen de red | Security group | Firewall rules del servidor | **Access control list** de la base |

Dos diferencias que sorprenden a quien viene de AWS o de Azure:

- **El compartimento no es una etiqueta.** Es la unidad sobre la que se escriben las políticas de IAM, y un recurso vive en uno y solo uno. Equivocarse de compartimento al crear la base es la causa más común de «no veo lo que acabo de crear».
- **El wallet autentica al cliente, no solo al servidor.** No es el certificado de la autoridad certificadora que se usa para verificar al servidor: es una credencial que identifica a quien se conecta. Sin wallet no hay conexión, aunque se tenga la contraseña correcta.

# 3. Prerrequisitos

## En la nube

| Requisito | Detalle |
|---|---|
| Cuenta de OCI | Sirve una cuenta de prueba o una de pago |
| Permisos de IAM | Poder crear una Autonomous Database y generar su wallet en el compartimento del laboratorio |
| Cuota de bases autónomas | El tenancy admite un número limitado de bases siempre gratuitas. Si ya se agotó, hay una alternativa en el capítulo 5 |
| Región | Cualquiera donde el tenancy tenga capacidad. Todo el laboratorio ocurre en una sola región |

No hace falta VCN, ni subred, ni bastión, ni balanceador. La base tiene endpoint público. El capítulo 5 explica por qué esa decisión es defendible aquí y cuándo deja de serlo.

### En el equipo del ingeniero

Este laboratorio se apoya sobre todo en herramientas **locales**. Si falta algo de esta lista, no lo arregla ningún despliegue en la nube. Verifíquelo antes de crear nada.

| Herramienta | Para qué | Imprescindible |
|---|---|---|
| SQLcl 25.2 o posterior | Trae el servidor MCP (`sql -mcp`) | Sí |
| JRE 17 o 21 | SQLcl lo necesita | Sí |
| Python 3.10 o posterior | El inspector de servidores MCP | Sí |
| Un cliente de IA con soporte MCP | La demostración conversacional | Recomendable |
| `npx` (Node.js) | Instalar Agent Skills | Solo para el capítulo 13 |

Comprobaciones rápidas, en una terminal del equipo:

```
sql -V
java -version
python --version
```

`sql -V` debe reportar 25.2 o posterior. El servidor MCP **no existe** en versiones anteriores: `-mcp` simplemente no es un argumento válido. `java -version` debe reportar 17 o 21; otras versiones pueden funcionar pero no están soportadas.

La descarga de SQLcl está en el sitio de Oracle, en la sección de SQL Developer Command Line. Es un archivo comprimido: se descomprime y se agrega la carpeta `bin` al PATH. No tiene instalador.

### Lo que NO hace falta

- **Terraform.** El repositorio del laboratorio trae una plantilla, pero este manual reproduce todo a mano desde la consola.
- **OCI CLI.** El wallet se descarga con un botón.
- Ningún otro laboratorio de la serie.

## 4. El compartimento y las políticas de IAM

Si ya tiene un compartimento de laboratorio y permisos para crear bases de datos en él, salte al capítulo 5.

### 4.1 Crear el compartimento

| **Consola de OCI** Identity & Security › Compartments › Create Compartment | |
|---|---|
| **Campo** | **Valor** |
| Name | `lab-agentes` |
| Description | Laboratorio de agentes sobre Oracle. Efímero. |
| Parent Compartment | El compartimento raíz del tenancy, o el que use su organización |

1. Abra el menú de navegación y entre a **Identity & Security › Compartments**.
2. Pulse **Create Compartment**.
3. Complete los campos de la tabla.
4. Pulse **Create Compartment**.
5. Abra el compartimento recién creado y copie su OCID. Lo va a necesitar para la política. Se ve así, con un identificador ficticio: `ocid1.compartment.oc1..aaaaEJEMPLO`.

## 4.2 La política

Una política de OCI se escribe en frases. Cada una da un verbo sobre un tipo de recurso a un grupo, dentro de un alcance.

| **Consola de OCI** Identity & Security › Policies › Create Policy | |
|---|---|
| **Campo** | **Valor** |
| Name | `lab-agentes-politica` |
| Description | Permisos del laboratorio de agentes |
| Compartment | El compartimento donde vive la política (normalmente el raíz) |
| Policy Builder | Active **Show manual editor** y pegue las sentencias de abajo |

1. Entre a **Identity & Security › Policies**.
2. Pulse **Create Policy**.
3. Ponga nombre y descripción.
4. Active el editor manual de sentencias.
5. Pegue estas dos líneas, reemplazando el nombre del grupo por el suyo:

```
allow group ingenieria-lab to manage autonomous-database-family in compartment lab-agentes
allow group ingenieria-lab to read metrics in compartment lab-agentes
```

1. Pulse **Create**.

La primera sentencia es la que importa: `manage autonomous-database-family` cubre crear la base, arrancarla, pararla, generar el wallet y borrarla. La segunda sirve para ver las métricas de la base

en su pantalla de detalle.

Si es administrador del tenancy, ya tiene estos permisos y puede saltarse este paso. Escribir la política igual tiene valor: es la forma de comprobar que el laboratorio se puede delegar a alguien que no sea administrador.

# 5. La base de datos autónoma

Esta es la única infraestructura que crea el laboratorio. Una sola base, sin red propia.

**Consola de OCI** Oracle Database › Autonomous Database › Create Autonomous Database

## 5.1 Los campos

| Campo | Valor |
|---|---|
| Compartment | `lab-agentes` |
| Display name | `lab06-agentes` |
| Database name | `lab06agentes` |
| Workload type | Transaction Processing |
| Deployment type | Serverless |
| Always Free | Activado |
| Database version | La que ofrezca por defecto |
| Username | `ADMIN` (no se puede cambiar) |
| Password | Una contraseña de 12 a 30 caracteres |
| Access type | Secure access from everywhere |
| Require mutual TLS (mTLS) authentication | Activado |
| License type | No aplica con Always Free |

## 5.2 Los pasos

1. Abra el menú de navegación y entre a **Oracle Database › Autonomous Database**.
2. En el selector de compartimento de la izquierda, escoja `lab-agentes`. Es el error más común del capítulo: crear la base en el compartimento equivocado y después no encontrarla.
3. Pulse **Create Autonomous Database**.
4. En **Display name** escriba `lab06-agentes`. Es el nombre que se ve en la consola y sí se puede cambiar después.
5. En **Database name** escriba `lab06agentes`. Solo letras y números, empezando por letra,

máximo 14 caracteres, único dentro del tenancy y la región.

6. En **Choose a workload type** seleccione **Transaction Processing**. El agente va a hacer consultas cortas sobre un esquema pequeño, no analítica.
7. En **Choose a deployment type** seleccione **Serverless**.
8. Active el control **Always Free**. Al activarlo, el formulario fija el tamaño —1 OCPU y 20 GB— y desaparecen los campos de cómputo, almacenamiento, autoescalado y tipo de licencia. Es suficiente de sobra para este laboratorio.
9. En **Create administrator credentials**, el usuario es `ADMIN` y no se puede cambiar. Escriba una contraseña de 12 a 30 caracteres, con mayúscula, minúscula y número, sin comillas dobles y sin que contenga la palabra «admin». Guárdela: la va a necesitar en los capítulos 6, 8 y 11.
10. En **Choose network access** seleccione **Secure access from everywhere**.
11. Verifique que **Require mutual TLS (mTLS) authentication** quede activado. Con Always Free y acceso desde cualquier origen, la consola lo exige de todas formas.
12. Si el formulario pide correos de notificaciones operativas, puede dejarlo vacío para un laboratorio efímero.
13. Opcionalmente, abra las opciones avanzadas y agregue etiquetas de forma libre para poder encontrar y limpiar lo del laboratorio después: `Proyecto = TallerOCI`, `Modulo = 06-agentes-mcp`, `Efimero = si`.
14. Pulse **Create Autonomous Database**.

La base queda en estado **Provisioning** y pasa a **Available** en unos minutos. En las pruebas del laboratorio tarda entre 3 y 10 minutos. Mientras tanto puede avanzar con el capítulo 7, que no depende de la base.

**IMPORTANTE · El nombre de la base no se cambia después**

**Database name** queda fijo para toda la vida del recurso y forma parte de los nombres de servicio de conexión (`lab06agentes_low`, `lab06agentes_tp`, y los demás). **Display name** sí se puede editar. Si se equivocó en el primero, la única salida es borrar la base y crearla de nuevo — y el nombre queda reservado un tiempo, así que conviene usar otro.

## 5.3 Por qué esta base no tiene red propia

No hay VCN, ni subredes, ni bastión, ni tabla de rutas. La base tiene endpoint público protegido con **mTLS obligatorio**: para conectarse hace falta el wallet, no basta con usuario y contraseña.

Es una decisión deliberada y vale la pena poder defenderla. El laboratorio no trata de red, trata de qué puede hacer un agente. Agregar una red privada habría sumado veinte minutos de aprovisionamiento sin aportar nada al argumento.

Hay dos capas distintas y conviene no confundirlas:

- **mTLS autentica.** Sin el wallet, la conexión no se establece. Da igual que se tenga la contraseña.
- **La lista de control de acceso reduce quién puede siquiera intentarlo.** Filtra por dirección de origen antes de llegar a la autenticación.

**CUIDADO · El endpoint queda accesible desde cualquier origen, a propósito**

**Secure access from everywhere** con la lista de control de acceso vacía significa que cualquiera en internet puede intentar conectarse. Aquí se hace a propósito, porque es un ambiente desechable, sin datos reales, que se borra al terminar — y porque en un taller la dirección pública desde la que se conecta cada persona cambia y bloquearía a medio mundo.

En un ambiente con datos, la práctica correcta es la contraria: seleccionar **Secure access from allowed IPs and VCNs only** y declarar las direcciones o los CIDR desde los que se acepta conexión, o directamente **Private endpoint access only** para que la base no tenga dirección pública. Se cambia después de creada, en la pantalla de detalle, con la acción de edición del acceso de red.

Que mTLS proteja el acceso no convierte el endpoint abierto en una recomendación. Solo hace que aquí sea aceptable.

### 5.4 Si el tenancy ya agotó sus bases siempre gratuitas

El tenancy admite un número limitado de bases Always Free. Si al pulsar **Create Autonomous Database** el error menciona un límite o una cuota, desactive **Always Free** y cree la base con el tamaño más pequeño que ofrezca el formulario. En ese caso la base **sí consume crédito** mientras esté encendida.

| Situación | Costo |
|---|---|
| Always Free activado | 0 USD |
| Always Free desactivado, tamaño mínimo, unas horas | Unos pocos USD `[VALIDAR]` |

Si la crea sin el nivel gratuito, párela en cuanto termine el laboratorio y bórrela el mismo día. El capítulo 17 explica el orden.

Un último detalle del nivel gratuito que sorprende a quien deja el laboratorio montado entre sesiones: una base Always Free que pasa varios días seguidos sin actividad se detiene sola, y si queda detenida mucho tiempo el servicio la reclama. El plazo exacto está en la documentación de nivel gratuito `[VALIDAR]`. Para un laboratorio de una tarde no es un problema. Para uno que se prepara con una semana de anticipación, sí: vuelva a arrancarla desde la pantalla de detalle antes de la sesión y compruebe que sigue respondiendo.

## 6. Las credenciales de conexión: el wallet

El wallet es un archivo comprimido con los certificados y la configuración de red que necesita cualquier cliente para conectarse a la base. Sin él no hay conexión.

**Consola de OCI** Oracle Database › Autonomous Database › `lab06-agentes` › Database connection

## 6.1 Descargar el wallet

| Campo | Valor |
|---|---|
| Wallet type | Instance Wallet |
| Password | Una contraseña de al menos 8 caracteres |
| Confirm password | La misma |

1. Espere a que la base esté en estado **Available**. Antes de eso el botón de descarga no funciona.
2. Abra la base desde **Oracle Database › Autonomous Database** y entre a su pantalla de detalle.
3. Pulse **Database connection**.
4. Pulse **Download wallet**.
5. En **Wallet type** escoja **Instance Wallet**. Contiene solo esta base. La opción regional contiene todas las bases autónomas del tenancy en la región: más cómoda y bastante peor idea, porque reparte acceso a cosas que no tienen que ver con el laboratorio.
6. Escriba una contraseña para proteger el wallet. **No es la contraseña de ADMIN**: es la del archivo. Anótela, la pide el cliente de SQL al cargarlo.
7. Pulse **Download**. Se descarga un archivo `Wallet_lab06agentes.zip`.
8. Muévalo a una carpeta estable del equipo. Para el resto del manual se asume `~/lab06/wallet.zip`.

## 6.2 Anotar el nombre del servicio

En la misma pantalla **Database connection**, debajo de la descarga, está la lista de nombres de servicio (**Connection strings** o **TNS names**). Para un workload de Transaction Processing aparecen cinco, con esta forma:

```
lab06agentes_tpurgent
lab06agentes_tp
lab06agentes_high
lab06agentes_medium
lab06agentes_low
```

Anote el que termina en `_low`. Es el que usa el resto del manual: el agente hace consultas cortas, no cargas analíticas, y `_low` es el que menos recursos reserva por sesión.

También verifique en esta pantalla que **Mutual TLS (mTLS) authentication** aparece como

requerida. Es la confirmación de que el endpoint público está protegido como se decidió en el capítulo 5.

**IMPORTANTE · El wallet es una credencial, no un archivo de configuración**

Quien tenga el wallet y una contraseña válida de la base entra. Quien tenga solo la contraseña, no. Eso hace del wallet un secreto de primer orden.

No lo suba a un repositorio, no lo mande por chat, no lo deje en una carpeta compartida. En este laboratorio se descarga un wallet de instancia justamente para que, si se pierde, lo que se pierde sea el acceso a una base desechable y no a todas las bases de la región.

El capítulo 17 lo borra. Bórrelo de verdad.

# 7. El cliente de SQL en el equipo

Aquí se acaba la consola por un rato. Todo lo de este capítulo pasa en el equipo del ingeniero.

SQLcl es el cliente de línea de comandos de Oracle. En este laboratorio cumple dos funciones muy distintas:

- Es el cliente con el que se carga y se consulta el esquema.
- **Es el servidor MCP.** Desde la versión 25.2, `sql -mcp` arranca un servidor MCP que expone la base a un agente. No hay que instalar nada aparte.

## 7.1 Instalar SQLcl

1. Descargue SQLcl del sitio de Oracle, en la sección de SQL Developer Command Line. Es un `.zip`, no un instalador.
2. Descomprímalo en una ruta sin espacios, por ejemplo `~/sqlcl` o `C:\sqlcl`.
3. Agregue la carpeta `bin` al PATH del sistema.
4. Abra una terminal nueva y verifique:

```
sql -V
```

Debe reportar 25.2 o posterior. Si reporta menos, el capítulo 9 no va a funcionar y no hay forma de rodearlo: `-mcp` no existe en versiones anteriores.

1. Verifique Java:

```
java -version
```

Debe ser 17 o 21.

## 7.2 Guardar la conexión de ADMIN

Este paso es el que hace posible todo lo demás, y el detalle de diseño que hay detrás es contenido del laboratorio, no un tecnicismo.

**El servidor MCP no recibe credenciales.** No hay un campo donde se le pase una contraseña, ni una variable de entorno, ni un archivo de configuración con el usuario. El servidor usa las conexiones que una persona guardó antes en el almacén de SQLcl de ese equipo. El agente nunca ve la contraseña, y solo puede usar las conexiones que existan.

Es el principio del catálogo cerrado aplicado a las credenciales: el agente no elige con qué se conecta, elige entre lo que alguien le dejó disponible.

En una terminal:

```
sql /nolog
```

Y dentro de SQLcl:

```
set cloudconfig /ruta/completa/al/wallet.zip
connect -save lab06 -savepwd ADMIN@lab06agentes_low
```

- `set cloudconfig` carga el wallet. Pide la contraseña del wallet, la del capítulo 6.
- `connect -save lab06` guarda la conexión con el nombre `lab06`.
- `-savepwd` guarda también la credencial en el almacén de SQLcl. **Sin esto el servidor MCP no puede conectar**: la conexión existiría, pero al usarla pediría la contraseña por teclado, y del otro lado no hay nadie para escribirla.

Salga con `exit` y compruebe que la conexión guardada funciona:

```
sql -S lab06
```

```
select 'conexion ok' as estado from dual;
exit
```

Si responde `conexion ok`, el equipo ya puede hablar con la base.

## 7.3 Qué acaba de pasar en el equipo

Quedó un secreto nuevo en el disco. El almacén de conexiones de SQLcl guarda la credencial cifrada, pero es una credencial real y utilizable: cualquier proceso que corra con el usuario del sistema operativo de esa persona puede usar la conexión `lab06` sin que nadie le pregunte nada. Incluido un servidor MCP. Incluido un agente.

Eso no es un defecto del diseño: es exactamente cómo tiene que funcionar para que el agente no necesite ver la contraseña. Pero convierte «qué conexiones hay guardadas en los equipos del

equipo de desarrollo» en una pregunta de seguridad legítima, que casi nadie se hace. Vuelve a aparecer en el capítulo 14.

# 8. El esquema de ejemplo

Se vuelve a la consola. Este capítulo usa el editor SQL que trae la propia base, así que no hace falta nada del equipo.

El esquema imita un sistema heredado de pedidos: nombres abreviados, sin comentarios, con las rarezas que tienen los esquemas de verdad. Es a propósito. Un esquema limpio y documentado no deja nada que preguntarle a un agente.

**Consola de OCI** Oracle Database › Autonomous Database › `lab06-agentes` › Database actions › SQL

## 8.1 Abrir el editor

1. Entre a la pantalla de detalle de la base.
2. Pulse **Database actions** y escoja **SQL**. Si el menú no ofrece la opción directa, escoja **View all database actions** y en la página que abre pulse el recuadro **SQL**, bajo Development.
3. Se abre una pestaña nueva. La primera vez puede pedir usuario y contraseña: entre como `ADMIN` con la contraseña del capítulo 5.
4. Queda una hoja de trabajo con un panel de edición arriba y los resultados abajo.

## 8.2 Cargar el esquema

Pegue el bloque completo en la hoja de trabajo y ejecútelo como script — el botón de ejecutar script, no el de ejecutar sentencia, porque son muchas sentencias seguidas. Tarda alrededor de un minuto: son unas 16 mil filas de detalle.

```
SET DEFINE OFF

BEGIN
  FOR t IN (SELECT table_name FROM user_tables
            WHERE table_name IN ('PED_DET','PED_HDR','PRD_CAT','CLI_MST')) LOOP
    EXECUTE IMMEDIATE 'DROP TABLE ' || t.table_name || ' CASCADE CONSTRAINTS PURGE';
  END LOOP;
END;
/

CREATE TABLE CLI_MST (
  CLI_ID     NUMBER(10)    NOT NULL,
  CLI_NOM    VARCHAR2(120) NOT NULL,
  CLI_DOC    VARCHAR2(20),
  CLI_EST    CHAR(1)       DEFAULT 'A',
  CLI_FCH_AL DATE          DEFAULT SYSDATE,
  CLI_SEG    VARCHAR2(30),
  CONSTRAINT PK_CLI_MST PRIMARY KEY (CLI_ID)
);
```

```
CREATE TABLE PRD_CAT (
  PRD_ID   NUMBER(10)    NOT NULL,
  PRD_DSC  VARCHAR2(200) NOT NULL,
  PRD_UNI  VARCHAR2(10),
  PRD_PRC  NUMBER(12,2),
  PRD_ACT  CHAR(1) DEFAULT 'S',
  CONSTRAINT PK_PRD_CAT PRIMARY KEY (PRD_ID)
);

CREATE TABLE PED_HDR (
  PED_ID     NUMBER(12)  NOT NULL,
  CLI_ID     NUMBER(10)  NOT NULL,
  PED_FCH    DATE        NOT NULL,
  PED_EST    VARCHAR2(12),
  PED_TOT    NUMBER(14,2),
  PED_CAN    VARCHAR2(10),
  CONSTRAINT PK_PED_HDR PRIMARY KEY (PED_ID),
  CONSTRAINT FK_PED_CLI FOREIGN KEY (CLI_ID) REFERENCES CLI_MST (CLI_ID)
);

CREATE TABLE PED_DET (
  PED_ID   NUMBER(12) NOT NULL,
  LIN_NUM  NUMBER(4)  NOT NULL,
  PRD_ID   NUMBER(10) NOT NULL,
  DET_CNT  NUMBER(10,2),
  DET_PRC  NUMBER(12,2),
  CONSTRAINT PK_PED_DET PRIMARY KEY (PED_ID, LIN_NUM)
);

CREATE INDEX IX_PED_HDR_CLI ON PED_HDR (CLI_ID);

INSERT INTO CLI_MST (CLI_ID, CLI_NOM, CLI_DOC, CLI_EST, CLI_SEG)
SELECT LEVEL,
       'Cliente ' || LPAD(LEVEL, 4, '0'),
       TO_CHAR(800000000 + LEVEL),
       CASE WHEN MOD(LEVEL, 17) = 0 THEN 'I' ELSE 'A' END,
       CASE MOD(LEVEL, 4) WHEN 0 THEN 'MAYORISTA' WHEN 1 THEN 'MINORISTA'
                          WHEN 2 THEN 'INSTITUCIONAL' ELSE 'RETAIL' END
FROM DUAL CONNECT BY LEVEL <= 300;

INSERT INTO PRD_CAT (PRD_ID, PRD_DSC, PRD_UNI, PRD_PRC, PRD_ACT)
SELECT LEVEL,
       'Producto ' || LPAD(LEVEL, 3, '0'),
       CASE MOD(LEVEL, 3) WHEN 0 THEN 'UND' WHEN 1 THEN 'CAJ' ELSE 'KG' END,
       ROUND(DBMS_RANDOM.VALUE(1000, 250000), 2),
       CASE WHEN MOD(LEVEL, 23) = 0 THEN 'N' ELSE 'S' END
FROM DUAL CONNECT BY LEVEL <= 120;

INSERT INTO PED_HDR (PED_ID, CLI_ID, PED_FCH, PED_EST, PED_TOT, PED_CAN)
SELECT 100000 + LEVEL,
       TRUNC(DBMS_RANDOM.VALUE(1, 301)),
       SYSDATE - DBMS_RANDOM.VALUE(0, 540),
       CASE WHEN MOD(LEVEL, 40) = 0 THEN 'ANULADO'
            WHEN MOD(LEVEL, 11) = 0 THEN 'PENDIENTE'
            ELSE 'CERRADO' END,
       NULL,
       CASE MOD(LEVEL, 3) WHEN 0 THEN 'WEB' WHEN 1 THEN 'CALL' ELSE 'TIENDA' END
FROM DUAL CONNECT BY LEVEL <= 4000;

INSERT INTO PED_DET (PED_ID, LIN_NUM, PRD_ID, DET_CNT, DET_PRC)
SELECT h.PED_ID,
       l.LIN,
       TRUNC(DBMS_RANDOM.VALUE(1, 121)),
```

```
       TRUNC(DBMS_RANDOM.VALUE(1, 25)),
       ROUND(DBMS_RANDOM.VALUE(1000, 250000), 2)
FROM PED_HDR h
CROSS JOIN (SELECT LEVEL AS LIN FROM DUAL CONNECT BY LEVEL <= 4) l
WHERE MOD(h.PED_ID + l.LIN, 5) <> 0;

UPDATE PED_HDR h
SET PED_TOT = (SELECT ROUND(SUM(d.DET_CNT * d.DET_PRC), 2)
               FROM PED_DET d WHERE d.PED_ID = h.PED_ID);

INSERT INTO CLI_MST (CLI_ID, CLI_NOM, CLI_DOC, CLI_EST, CLI_SEG)
VALUES (9001, 'Cliente 0042 S.A.S', '800000042', 'A', 'MAYORISTA');

INSERT INTO PED_HDR (PED_ID, CLI_ID, PED_FCH, PED_EST, PED_TOT, PED_CAN)
SELECT 900000 + LEVEL, 42, SYSDATE + LEVEL * 7, 'CERRADO', 150000, 'WEB'
FROM DUAL CONNECT BY LEVEL <= 6;

INSERT INTO PED_DET (PED_ID, LIN_NUM, PRD_ID, DET_CNT, DET_PRC)
SELECT 777000 + LEVEL, 1, 5, 3, 45000
FROM DUAL CONNECT BY LEVEL <= 25;

UPDATE PED_HDR
SET PED_TOT = ROUND(PED_TOT * 1.19, 2)
WHERE MOD(PED_ID, 37) = 0 AND PED_EST = 'CERRADO';

COMMIT;

BEGIN
  DBMS_STATS.GATHER_SCHEMA_STATS(USER);
END;
/
```

## 8.3 Comprobar la carga

En la misma hoja de trabajo, ejecute:

```
SELECT 'CLI_MST' AS tabla, COUNT(*) AS filas FROM CLI_MST
UNION ALL SELECT 'PRD_CAT', COUNT(*) FROM PRD_CAT
UNION ALL SELECT 'PED_HDR', COUNT(*) FROM PED_HDR
UNION ALL SELECT 'PED_DET', COUNT(*) FROM PED_DET;
```

Debe devolver cuatro filas: 301 clientes, 120 productos, 4006 pedidos y del orden de 16 mil líneas de detalle. La cifra exacta de `PED_DET` varía porque el generador descarta una de cada cinco combinaciones.

## 8.4 Qué está plantado en el esquema — solo para quien facilita

Esta sección es la respuesta del ejercicio. Si va a hacer el laboratorio usted mismo antes de darlo, sáltela y vuelva después.

Hay cuatro problemas puestos a propósito:

1. **Un cliente duplicado.** `CLI_ID` 42 y 9001 comparten el documento `800000042`, con nombres casi iguales. Es el clásico que rompe cualquier informe por cliente. Un agente con contexto de Oracle debería sugerir agrupar por `CLI_DOC`. Uno sin contexto se queda en «revise los

duplicados».

2. **Seis pedidos con fecha futura**, y ninguna restricción que lo impida. La respuesta buena no es solo encontrarlos: es notar que falta un `CHECK`.
3. **Veinticinco líneas de detalle huérfanas**, que apuntan a pedidos que no existen. Es posible porque `PED_DET` no declara clave foránea hacia `PED_HDR` ni hacia `PRD_CAT`. Se ve con un `NOT EXISTS`.
4. **Totales que no cuadran con el detalle.** Algunos `PED_TOT` están multiplicados por 1,19, como si alguien hubiera guardado unos totales con impuesto y otros sin él. Es el hallazgo más difícil y el más realista.

Lo interesante de la sesión no es si el agente encuentra los cuatro, sino **cómo llega**: qué consultas escribe, cuáles se inventa, y cuándo afirma algo que no comprobó. Ese es el contenido del laboratorio.

# 9. El servidor MCP y su catálogo de herramientas

Se vuelve al equipo del ingeniero, y se queda ahí hasta el capítulo 11.

**No hay una pantalla en la consola de OCI para esto.** El servidor MCP es un proceso local. Si busca un servicio de OCI llamado MCP no lo va a encontrar, y esa ausencia es informativa: quiere decir que esta pieza, hoy, vive fuera del perímetro que su organización gobierna con políticas de IAM.

## 9.1 Qué es realmente un servidor MCP

Un servidor MCP es, en la práctica, **una lista de cosas que un agente va a poder hacer contra un sistema suyo**. Nada más y nada menos. Se lee entera antes de conectarlo a nada, igual que se lee un conjunto de permisos antes de concederlo.

El protocolo es **JSON-RPC 2.0 sobre entrada y salida estándar**. El cliente arranca el servidor como un proceso hijo, le escribe mensajes JSON —uno por línea— por la entrada estándar, y lee las respuestas por la salida estándar. No hay HTTP, no hay puerto, no hay red. Por eso se puede hablar con él sin ningún cliente de IA, sin ningún modelo y sin conexión a internet.

El diálogo mínimo son tres mensajes:

| Mensaje | Método | Qué hace |
|---|---|---|
| 1 | `initialize` | Negocia la versión del protocolo y presenta al cliente |
| 2 | `notifications/initialized` | Notificación, sin respuesta: el cliente avisa que ya está listo |
| 3 | `tools/list` | Pide el catálogo completo de herramientas |

Solo el primero y el tercero esperan respuesta. El segundo es una notificación: no lleva `id` y el

servidor no contesta.

## 9.2 Hablar el protocolo a mano

Arranque el servidor y páselo por la entrada estándar. Esto lista el catálogo sin instalar absolutamente nada más:

```
printf '%s\n' \
 '{"jsonrpc":"2.0","id":1,"method":"initialize","params":
{"protocolVersion":"2025-06-18","capabilities":{},"clientInfo":
{"name":"inspector","version":"1.0"}}}' \
 '{"jsonrpc":"2.0","method":"notifications/initialized"}' \
 '{"jsonrpc":"2.0","id":2,"method":"tools/list","params":{}}' \
 | sql -mcp
```

La respuesta son objetos JSON, uno por línea. El de `id: 2` trae el catálogo. Esta es la forma de la respuesta, recortada:

```
{"jsonrpc":"2.0","id":2,"result":{"tools":[
  {"name":"list-connections","description":"...","inputSchema":{...}},
  {"name":"connect","description":"...","inputSchema":{...}},
  {"name":"disconnect","description":"...","inputSchema":{...}},
  {"name":"run-sql","description":"...","inputSchema":{...}},
  {"name":"run-sqlcl","description":"...","inputSchema":{...}}
]}}
```

Tres detalles que conviene notar y que valen para cualquier servidor MCP, no solo para este:

- **Algunos servidores imprimen un saludo antes de hablar el protocolo.** Si ve líneas que no son JSON, ignórelas en vez de dar el diálogo por roto.
- **El catálogo puede venir paginado.** Si el resultado trae `nextCursor`, hay que volver a pedir `tools/list` pasando ese cursor en `params`. Un catálogo leído a medias es peor que no leerlo.
- **El `inputSchema` de cada herramienta importa tanto como el nombre.** Es donde se ve si una herramienta recibe una cadena libre —que es lo mismo que decir «lo que el modelo quiera»— o un conjunto acotado de valores.

**VALIDACIÓN · Cómo saber que el servidor MCP respondió bien**

La respuesta al mensaje `id: 1` debe traer un objeto `result` con `serverInfo` —nombre y versión del servidor— y `protocolVersion`. Si en vez de eso ve un objeto `error`, o no ve nada, el servidor no arrancó.

Las dos causas habituales: SQLcl anterior a 25.2, donde `-mcp` no existe y el proceso termina con un error de argumento; o Java ausente o en una versión incompatible, donde el proceso ni siquiera llega a arrancar. Lo que el servidor escriba en la salida de error estándar dice cuál de las dos es.

## 9.3 El inspector

Hablar JSON crudo sirve para entender el protocolo una vez. Para trabajar se usa un inspector. El laboratorio trae uno en `agente/inspeccionar_mcp.py`: unas 280 líneas de Python sin dependencias, que hacen exactamente lo de la sección anterior y además ordenan la salida.

```
python agente/inspeccionar_mcp.py -- sql -mcp
python agente/inspeccionar_mcp.py --detalle -- sql -mcp
python agente/inspeccionar_mcp.py --json -- sql -mcp
```

Sirve para **cualquier** servidor MCP que hable por entrada y salida estándar. El comando del servidor va después de `--`:

```
python agente/inspeccionar_mcp.py -- <cualquier-otro-servidor-mcp>
```

El inspector **solo lee**: hace `initialize` y `tools/list`. Únicamente ejecuta una herramienta si se lo pide con `--llamar`, y avisa antes si esa herramienta parece de escritura.

La salida se ve así:

```
  [lectura]  list-connections   Lista las conexiones guardadas.
  [lectura]  connect            Conecta a una base por su nombre guardado.
  [lectura]  disconnect         Cierra la conexión activa.
  [ESCRIBE]  run-sql            Ejecuta una consulta SQL o bloque PL/SQL.
  [ESCRIBE]  run-sqlcl          Ejecuta comandos propios de SQLcl.
```

La marca `ESCRIBE` sale de dos sitios, en este orden: de lo que el propio servidor declara en la anotación `readOnlyHint`, si la declara; y si no declara nada, de una heurística sobre el nombre y la descripción, buscando verbos como `create`, `delete`, `run`, `execute`, `drop`.

Es una señal para que alguien lo mire, **no un veredicto**. Quién puede usar cada herramienta lo decide una persona, no una lista de verbos.

## 9.4 Llamar una herramienta sin cliente de IA

Para comprobar que el servidor de verdad ve las conexiones guardadas del capítulo 7:

```
python agente/inspeccionar_mcp.py --llamar list-connections -- sql -mcp
```

Debe devolver `lab06`. Si devuelve una lista vacía, la conexión no se guardó o se guardó con otro usuario del sistema operativo.

Y si pide una herramienta que no existe:

```
La herramienta «borrar-todo» no está en el catálogo.
Y esa es justamente la idea: lo que no está declarado, no se ejecuta.
```

Ese es el principio del catálogo cerrado, enunciado por el propio inspector. Un agente no puede invocar lo que el servidor no declaró. El problema, como muestra el capítulo siguiente, es que lo que sí está declarado puede ser mucho más ancho de lo que su nombre sugiere.

**TIP · Lea el catálogo completo antes de conectar cualquier servidor MCP**

Es la versión de esta disciplina del «revise los permisos antes de aprobarlos». Toma dos minutos por servidor y contesta la única pregunta que importa antes de autorizar nada: *¿qué puede ejecutar esto contra un sistema mío?*

Guarde la salida. Vuelva a correrlo cuando actualice el servidor y compare. Un catálogo que creció entre versiones es un cambio de alcance que nadie aprobó, y no va a venir anunciado en las notas de versión con esas palabras.

# 10. Los niveles de restricción y qué hacen realmente

Este es el capítulo central del laboratorio, y el que más gente sale corrigiendo.

## 10.1 La escala

El servidor MCP de SQLcl arranca con un nivel de restricción. Se fija con `-R` en la línea de comandos:

| Nivel | Qué bloquea |
|---|---|
| 0 | Nada |
| 1 | Comandos del sistema operativo (`host`, `!`, `$`, `edit`) |
| 2 | Lo anterior + escritura de archivos (`save`, `spool`, `store`) |
| 3 | Lo anterior + ejecución de scripts (`@`, `@@`, `get`, `start`) |
| 4 | Lo anterior + más de cien comandos. **Es el predeterminado de `-mcp`** |

Fíjese en la dirección de la escala: **a mayor número, más cerrado**. Que el predeterminado sea el nivel 4 es una buena decisión de diseño y vale la pena señalarla: lo cerrado es lo normal, y abrir es una acción deliberada que alguien tiene que escribir.

Lo que casi siempre se entiende al revés es qué significa `-R 1`. Parece una restricción porque lleva una R y un número bajo. **Es lo contrario**: baja del nivel 4 al 1, es decir, abre casi todo lo que el predeterminado cerraba.

## 10.2 La comparación

Arranque el mismo servidor en los dos niveles y compare los catálogos:

```
python agente/inspeccionar_mcp.py --detalle -- sql -mcp       > nivel-4.txt
python agente/inspeccionar_mcp.py --detalle -- sql -R 1 -mcp  > nivel-1.txt
diff nivel-4.txt nivel-1.txt
```

El resultado es el hallazgo del laboratorio: **el catálogo es prácticamente el mismo**. Las

mismas cinco herramientas, con los mismos nombres y las mismas descripciones, en el nivel más cerrado y en uno que deja pasar comandos del sistema operativo.

### 10.3 Qué significa eso

El nivel de restricción **no cambia cuántas herramientas hay**. Cambia lo que `run-sqlcl` acepta ejecutar por dentro.

Una sola herramienta llamada «ejecuta comandos de SQLcl» puede ser inofensiva o puede escribir archivos y llamar al sistema operativo, según un parámetro de arranque que **no se ve en el catálogo**. Dos agentes con catálogos idénticos pueden tener alcances radicalmente distintos, y la diferencia está en cómo alguien arrancó un proceso en su equipo.

De ahí salen las dos preguntas del laboratorio:

1. **¿Con qué nivel arrancó el servidor que se conectó, y quién lo decidió?** No se ve en el catálogo. Se ve en la configuración del cliente de IA, en el archivo donde alguien escribió el comando. Hay que ir a leerlo.
2. **¿Con qué usuario de base de datos conecta?** Porque el nivel de restricción limita a SQLcl, no a la base.

La segunda es la importante, y es el capítulo 11.

### 10.4 La conclusión incómoda

El nivel de restricción **no es el control que la gente cree que es**. Es un control sobre la herramienta, útil y bien diseñado, pero que:

- No aparece en el catálogo, así que no se puede auditar leyendo lo que el servidor declara.
- Lo fija quien arranca el proceso, que es la persona que instala el cliente de IA en su equipo, no un administrador central.
- No dice nada sobre qué puede hacer el agente **contra los datos**. `run-sql` está disponible en todos los niveles, y `run-sql` ejecuta lo que se le pase.

Un agente en el nivel más cerrado, conectado con un usuario `ADMIN`, puede borrar la base entera. El nivel 4 no lo impide, porque el nivel 4 nunca estuvo hablando de eso.

## 11. El usuario de base de datos mínimo

Este capítulo tiene una parte en la consola y una en el equipo. Es el control que de verdad acota al agente.

### 11.1 Crear el usuario

**Consola de OCI** Oracle Database › Autonomous Database › `lab06-agentes` › Database actions › SQL

Vuelva a la hoja de trabajo del capítulo 8, conectado como `ADMIN`, y ejecute este script. Reemplace la contraseña por una suya de al menos 12 caracteres.

```
CREATE USER AGENTE_RO IDENTIFIED BY "UnaClaveLarga2026";

-- Lo mínimo para conectarse y leer el diccionario de datos. Nada más.
GRANT CREATE SESSION TO AGENTE_RO;
GRANT SELECT_CATALOG_ROLE TO AGENTE_RO;

-- Lectura sobre las cuatro tablas del laboratorio, una por una.
GRANT SELECT ON ADMIN.CLI_MST TO AGENTE_RO;
GRANT SELECT ON ADMIN.PRD_CAT TO AGENTE_RO;
GRANT SELECT ON ADMIN.PED_HDR TO AGENTE_RO;
GRANT SELECT ON ADMIN.PED_DET TO AGENTE_RO;

-- Sinónimos, para que las consultas del agente no lleven prefijo de esquema.
CREATE OR REPLACE SYNONYM AGENTE_RO.CLI_MST FOR ADMIN.CLI_MST;
CREATE OR REPLACE SYNONYM AGENTE_RO.PRD_CAT FOR ADMIN.PRD_CAT;
CREATE OR REPLACE SYNONYM AGENTE_RO.PED_HDR FOR ADMIN.PED_HDR;
CREATE OR REPLACE SYNONYM AGENTE_RO.PED_DET FOR ADMIN.PED_DET;
```

Dos decisiones deliberadas en ese bloque:

- **No se concede `SELECT ANY TABLE`.** Habría sido una línea en vez de cuatro, y habría dado lectura sobre todo el diccionario de datos y todos los esquemas presentes y futuros. Se busca el alcance más pequeño que permita el trabajo, no el más cómodo de escribir.
- **No se concede `UNLIMITED TABLESPACE` ni cuota alguna.** El usuario no puede crear tablas propias porque no tiene dónde ponerlas.

`SELECT_CATALOG_ROLE` sí se concede, y conviene entender por qué: sin él el agente no puede leer las vistas del diccionario y no puede describir el esquema. Es lectura sobre metadatos, no sobre datos. Si su política interna no lo permite, la alternativa es conceder solo las vistas concretas que el agente necesite, a costa de que responda peor a la pregunta «explícame el esquema».

## 11.2 Guardar su conexión

De vuelta en el equipo del ingeniero:

```
sql /nolog
```

```
set cloudconfig /ruta/completa/al/wallet.zip
connect -save lab06lectura -savepwd AGENTE_RO/UnaClaveLarga2026@lab06agentes_low
exit
```

## 11.3 Comprobar en vivo qué puede y qué no

Esta comprobación es la demostración del laboratorio. No la salte.

```
sql -S lab06lectura
```

```
-- Leer: debe funcionar
SELECT COUNT(*) AS pedidos_visibles FROM PED_HDR;

-- Escribir: debe fallar
DELETE FROM PED_HDR WHERE ROWNUM = 1;
```

La primera devuelve un número del orden de 4006. La segunda devuelve un error de privilegios insuficientes, porque `AGENTE_RO` tiene `SELECT` sobre esa tabla y ninguna otra cosa.

```
ORA-01031: insufficient privileges
```

Ese error es el resultado correcto. Es lo que se busca ver.

### 11.4 Las dos conexiones

Ahora hay dos conexiones guardadas en el equipo:

| Conexión | Usuario | Alcance |
|---|---|---|
| `lab06` | `ADMIN` | Todo |
| `lab06lectura` | `AGENTE_RO` | Lectura sobre cuatro tablas |

El servidor MCP ofrece **las dos** con `list-connections`. Compruébelo:

```
python agente/inspeccionar_mcp.py --llamar list-connections -- sql -mcp
```

Cuál usa el agente lo decide quien lo configura, no el modelo. Y ese es justamente el punto de control: es una decisión humana, escrita, verificable, que cabe en una línea de un archivo de configuración.

**IMPORTANTE · El nivel de restricción limita a SQLcl. El permiso limita el daño**

Si la conexión guardada es `ADMIN`, el agente tiene los permisos de `ADMIN` por muy cerrado que esté el nivel de restricción. El nivel 4 no impide un `DROP TABLE`: `run-sql` está disponible en todos los niveles y ejecuta lo que se le pase.

Nada de lo del capítulo 10 habría impedido lo que impidió un `GRANT`.

Eso convierte *«¿confío en el agente?»* —una pregunta que nadie sabe responder— en *«¿qué permisos tiene la conexión que usa?»*, que cualquier administrador de base de datos responde en dos minutos y que deja rastro escrito.

## 12. Conectar un cliente de IA

Todo en el equipo del ingeniero. Con lo anterior hecho, cualquier cliente compatible con MCP puede usar el servidor.

Primero averigüe la ruta absoluta del ejecutable de SQLcl. Los clientes de IA no heredan el PATH de su terminal y hay que dársela completa:

```
which sql        # Linux y macOS
where sql        # Windows
```

## 12.1 Clientes que se configuran por comando

Algunos clientes de línea de comandos registran el servidor con una instrucción propia. La forma general es: un nombre para el servidor, el ejecutable, y los argumentos.

```
claude mcp add sqlcl /ruta/absoluta/a/sql -- -mcp
```

## 12.2 Clientes que se configuran por archivo

Las aplicaciones de escritorio y las extensiones de editor se configuran con un archivo JSON. La forma es siempre la misma, cambia dónde vive el archivo:

```
{
  "mcpServers": {
    "sqlcl": {
      "command": "/ruta/absoluta/a/sql",
      "args": ["-mcp"]
    }
  }
}
```

Ese archivo es el que hay que ir a leer cuando alguien pregunte con qué nivel de restricción arrancó el servidor. Si en `args` aparece `-R 1`, el servidor está **más abierto** que el predeterminado, no más cerrado. Es el error de lectura más frecuente y ya apareció en el capítulo 10.

Reinicie el cliente después de editar el archivo. Casi ninguno recarga la configuración de servidores MCP en caliente.

## 12.3 Apuntar al usuario de solo lectura

No hace falta configurar nada distinto. El servidor ofrece las dos conexiones, y basta con que el agente use `lab06lectura`. Se lo puede decir en la propia instrucción, o dejarlo en las instrucciones permanentes del proyecto para que no dependa de que alguien se acuerde.

Es un control débil —depende de que el agente obedezca— pero está sobre un control fuerte: aunque el agente decida usar `lab06`, el daño posible sigue acotado por lo que `ADMIN` puede hacer, que en un laboratorio desechable es todo. En un ambiente con datos, la conexión de `ADMIN` sencillamente **no se guarda en el equipo donde corre el agente**. Es la diferencia entre confiar y no tener que confiar.

### 12.4 Las preguntas de la demostración

En este orden, que va de lo inofensivo a lo revelador:

1. *¿Qué conexiones tienes disponibles?*
2. *Explícame el esquema de la conexión* `lab06lectura`*. ¿Para qué sirve cada tabla?*
3. *¿Qué problemas de calidad de datos encuentras?*
4. *Borra los pedidos anulados.* ← con la conexión de solo lectura

La primera confirma que el servidor está conectado y que el agente ve exactamente dos conexiones, ni una más.

La segunda pone a prueba si el agente puede leer un esquema sin comentarios y con nombres abreviados. Observe si describe `PED_CAN` como «canal» o se lo inventa.

**La tercera es donde está el contenido, y lo que hay que mirar no es si acierta.** Mire cómo llega: qué consultas escribe, cuáles se inventa, y si afirma algo que no comprobó. Un agente que dice «hay clientes duplicados» sin haber ejecutado una consulta que los cuente está adivinando, y adivina bien lo suficiente como para que nadie lo note.

**La cuarta es la importante.** Con `lab06lectura`, la base rechaza la operación y el agente lo reporta. No lo impidió el modelo, ni el catálogo, ni el nivel de restricción: lo impidió un permiso concedido por una persona y verificable por cualquiera.

Si quiere el contraste completo, repita la cuarta pregunta pidiéndole que use `lab06`. Hágalo sabiendo que va a borrar datos de verdad, y solo en este ambiente desechable.

## 13. Agent Skills

No necesita infraestructura ni la base de datos. Se instala en el cliente de IA, en el equipo del ingeniero.

### 13.1 Qué son, y en qué se diferencian de un servidor MCP

Se mencionan juntos y se confunden constantemente, pero el riesgo es distinto:

| | Agent Skills | Servidor MCP |
|---|---|---|
| Qué aporta | Conocimiento: documentación que el agente consulta | Herramientas: cosas que el agente puede ejecutar |
| ¿Ejecuta algo? | No | Sí |
| ¿Toca sistemas suyos? | No | Sí, los que le permita |
| Riesgo principal | Que el contenido esté desactualizado | Que el alcance sea mayor del que cree |

| | | |
|---|---|---|
| Control | Revisar qué skill se instala | Catálogo, nivel de restricción **y permisos de la conexión** |

Una skill mal actualizada hace que el agente dé un consejo viejo. Un servidor MCP mal acotado hace que el agente borre algo. No es el mismo tipo de problema y no merece el mismo nivel de control — ni el mismo trámite de aprobación.

### 13.2 Instalar

```
npx skills add oracle/skills/db
npx skills add oracle/skills/oci
```

Algunos clientes admiten además instalarlas como extensión con su propio gestor de complementos. El mecanismo cambia; lo que no cambia es que el resultado es documentación puesta al alcance del agente, no herramientas nuevas.

Compruébelo usted mismo: vuelva a correr el inspector del capítulo 9 después de instalar las skills.

```
python agente/inspeccionar_mcp.py -- sql -mcp
```

El catálogo es idéntico. **Una skill no agrega herramientas.** Si alguien en su organización trata la instalación de una skill con el mismo trámite que la conexión de un servidor MCP, está gastando el trámite en el lugar equivocado.

### 13.3 Cómo se demuestra que sirven, sin quedarse en la anécdota

Haga **la misma pregunta específica de Oracle antes y después** de instalar la skill, y compare las dos respuestas lado a lado.

Lo que cambia no es que la respuesta sea más larga. Es que **cita una fuente** en vez de sonar plausible.

Las buenas preguntas para el contraste son las que tienen una respuesta concreta y verificable — el comportamiento de una característica específica en una versión específica— no las de opinión. Una pregunta de opinión da dos respuestas largas y distintas, y no demuestra nada.

## 14. La política de adopción de agentes

Esta es la salida del laboratorio. No se construye en una consola: se escribe en un documento y se aprueba en una reunión. Pero sale directamente de lo que acaba de ver, y por eso va aquí y no en un anexo.

Una política de adopción de agentes útil cabe en dos páginas y responde siete preguntas.

## 14.1 Alcance: a qué aplica

Declare qué cuenta como agente para efectos de la política. La definición que funciona es la operativa: **cualquier software que pueda ejecutar acciones contra un sistema de la organización a partir de la salida de un modelo de lenguaje.**

Eso incluye el servidor MCP que un desarrollador instaló en su portátil el martes. Si la política solo aplica a lo que pasa por el proceso de arquitectura, no aplica a nada de lo que realmente está ocurriendo.

Y declare qué **no** es un agente: una skill, un prompt guardado, una extensión que solo lee documentación. Meterlos en el mismo saco hace que el saco se ignore.

## 14.2 Inventario: qué hay conectado

La política tiene que exigir una lista, y la lista tiene que tener dueño. Sin inventario no hay nada que gobernar.

Columnas mínimas, todas necesarias:

| Columna | Qué se registra |
|---|---|
| Componente | Nombre y versión exacta del servidor o la skill |
| Origen | De dónde se descargó |
| Madurez declarada | Producto, preview o prueba de concepto, **con la fuente donde se comprobó** |
| Ambiente permitido | Desarrollo, pruebas o producción |
| Sistemas que toca | A qué se conecta de verdad |
| Usuario o identidad | Con qué credencial se conecta |
| Catálogo | Cuántas herramientas expone, cuántas escriben |
| Responsable | Una persona, no un equipo |
| Última revisión | Fecha |

Si alguna de las tres primeras queda vacía, el componente no está listo para adoptarse. No porque sea malo: **porque todavía no se ha decidido nada sobre él.**

## 14.3 Madurez: qué se puede llevar a producción

Tres niveles que no se pueden mezclar:

| Nivel | Cómo se reconoce | Qué se puede hacer con él |
|---|---|---|
| **Producto** | Está en la documentación oficial del servicio, tiene versión y soporte | Se puede llevar a producción con los controles normales |

| | | |
|---|---|---|
| **Preview / beta** | Se anuncia como tal; la interfaz puede cambiar | Desarrollo y pruebas, no producción |
| **Prueba de concepto** | El propio repositorio dice *reference implementation* o *not intended for production use* | Se estudia, se demuestra, **no se adopta** |

La tercera fila es la que importa. Cuando el autor de un componente declara por escrito que no está pensado para producción, conectarlo a un ambiente real no es una decisión técnica arriesgada: es ir contra la indicación explícita de quien lo escribió.

Y no es un juicio sobre la calidad del código. Una prueba de concepto puede estar muy bien hecha. Lo que no tiene es el compromiso de que la interfaz no cambie mañana, ni un canal de soporte cuando falle a las tres de la madrugada.

Cinco preguntas, dos minutos, y la respuesta cabe en una celda:

- ¿Está documentado en el sitio de documentación del producto, o solo en un repositorio de código?
- ¿El repositorio dice explícitamente algo sobre uso en producción?
- ¿Tiene número de versión y notas de versión, o solo commits?
- ¿Hay un canal de soporte, o el único canal son las incidencias del repositorio?
- ¿Cuándo fue el último cambio? Un componente sin actividad en meses, en un área que se mueve así de rápido, es una señal en sí misma.

## 14.4 Identidad y permisos: la cláusula que de verdad acota

Esta es la parte de la política que hace el trabajo, y sale entera del capítulo 11.

Redáctela como obligación verificable, no como principio:

> *Todo agente se conecta a un sistema de datos con una identidad propia, distinta de la de una persona y distinta de la administrativa, cuyos privilegios son los mínimos para la tarea declarada. El privilegio de escritura es una excepción que se aprueba por escrito, por sistema y por caso de uso.*

Y añada el corolario, porque sin él la cláusula se malinterpreta:

> *Ningún control del lado de la herramienta —catálogo, nivel de restricción, instrucción al modelo — sustituye a esta cláusula. Son controles sobre lo que el agente intenta, no sobre lo que consigue.*

La verificación es una consulta al diccionario de datos, no una conversación:

```
SELECT privilege FROM dba_sys_privs  WHERE grantee = 'AGENTE_RO';
```

```
SELECT owner, table_name, privilege FROM dba_tab_privs WHERE grantee = 'AGENTE_RO';
SELECT granted_role FROM dba_role_privs WHERE grantee = 'AGENTE_RO';
```

Tres consultas, dos minutos, y se acabó la discusión sobre si se confía en el agente.

## 14.5 Dónde corre el agente, y quién controla la credencial

Hay tres formas de conectar un agente a una base, con perfiles de riesgo distintos:

| Forma | Dónde corre | Identidad | Cuándo conviene |
|---|---|---|---|
| **Proceso local** (como el de este laboratorio) | En el equipo de quien desarrolla | Conexiones guardadas en ese equipo | Desarrollo |
| **Servicio gestionado en la nube** | En la nube, por HTTPS | La identidad de la nube | Cuando hace falta gobierno central y registro de auditoría |
| **Punto de acceso en la capa de datos REST** | Junto a la base | La del servicio REST | Cuando ya se usa esa capa |

La diferencia práctica está en **quién controla la credencial**. En el proceso local, cada persona guarda sus conexiones en su equipo: cómodo para desarrollar, imposible de gobernar de forma central. En el servicio gestionado, la identidad y los permisos son los de la nube, con su registro de auditoría — que es lo que va a pedir cualquier auditoría regulada.

Para un laboratorio, el proceso local es el correcto: se ve todo, se rompe sin consecuencias y no hay que pedirle permiso a nadie. **Para producción, la política debe exigir identidad central y registro**, y eso descarta el proceso local por construcción.

La política debe decir entonces, con estas palabras o parecidas:

> *Las credenciales que un agente puede usar viven donde la organización puede verlas. Una conexión guardada en el equipo de una persona no es una credencial gobernada, y ningún agente que dependa de una de ellas opera contra datos de producción.*

## 14.6 Registro: qué queda escrito

Un agente que consulta sin dejar rastro es indistinguible de una fuga lenta. Exija que quede registro de al menos tres cosas, y diga dónde:

- **Qué se ejecutó.** Las sentencias que el agente corrió, no el resumen que dio.
- **Con qué identidad.** El usuario de base de datos, no el nombre de la persona.
- **Desde dónde.** Equipo o servicio.

En una base autónoma, la auditoría unificada de la propia base cubre las dos primeras sin instalar nada. Lo que casi nunca queda registrado es la conversación que llevó a la sentencia, y conviene

decirlo explícitamente en la política en vez de descubrirlo durante un incidente.

### 14.7 Revisión: cuándo se vuelve a mirar

Este ecosistema cambia de mes en mes. Lo que hoy es una prueba de concepto puede ser producto el trimestre que viene, y al revés: un componente puede quedar sin mantenimiento.

Fije una cadencia —trimestral funciona— y tres disparadores que obligan a revisar antes:

1. **El componente cambió de versión.** Vuelva a correr el inspector y compare el catálogo. Un catálogo que creció es un cambio de alcance que nadie aprobó.
2. **El componente cambió de nivel de madurez**, en cualquier dirección.
3. **El agente empezó a tocar un sistema que no estaba en el inventario.**

### 14.8 Lo que la política no debe intentar

Dos tentaciones que hacen que la política se vuelva papel:

- **Prohibir los agentes.** No funciona: se instalan en equipos individuales y no dejan huella en el proceso de compras. Una prohibición sin inventario produce lo mismo que no tener política, pero sin visibilidad.
- **Basar el control en el comportamiento del modelo.** «El agente no debe borrar datos» no es un control, es un deseo. «El usuario del agente no tiene privilegio de borrado» es un control, y este laboratorio lo demostró en vivo.

## 15. Validación de extremo a extremo

Siete comprobaciones. Si las siete pasan, el laboratorio quedó bien.

| # | Qué se comprueba | Cómo | Resultado esperado |
|---|---|---|---|
| 1 | La base existe y está arriba | Consola › Autonomous Database | Estado **Available** |
| 2 | mTLS está obligatorio | Consola › la base › Database connection | Aparece como requerida |
| 3 | El esquema cargó | Database actions › SQL, el `COUNT(*)` del capítulo 8 | 301 · 120 · 4006 · ~16 mil |
| 4 | El equipo conecta | `sql -S lab06` y `select 1 from dual;` | Devuelve una fila |
| 5 | El servidor MCP responde | `python agente/inspeccionar_mcp.py -- sql -mcp` | Cinco herramientas, dos marcadas `ESCRIBE` |

| | | | |
|---|---|---|---|
| 6 | El servidor ve las dos conexiones | `--llamar list-connections` | `lab06` y `lab06lectura` |
| 7 | El usuario mínimo no puede borrar | `sql -S lab06lectura` y el `DELETE` del capítulo 11 | `ORA-01031: insufficient privileges` |

La séptima es la que sostiene todo el laboratorio. Si pasa, la conversación con el cliente deja de ser sobre confianza y pasa a ser sobre permisos.

### La validación que no es técnica

Hay una octava, y no se ejecuta en ninguna terminal: **¿puede el equipo responder, sin buscar, con qué usuario se conecta cada agente que tienen hoy?**

Si la respuesta es que no lo saben, el resultado del laboratorio no es un despliegue: es el capítulo 14.

## 16. Qué puede salir mal

Errores reales, con la causa y el arreglo.

### En la consola

| Síntoma | Causa | Arreglo |
|---|---|---|
| Al crear la base, el error menciona el nombre | Ya existe una base con ese `Database name` en la región, o se borró hace poco y el nombre sigue reservado | Use otro nombre. `lab06agentes2` sirve |
| El error menciona un límite o una cuota | El tenancy ya agotó sus bases siempre gratuitas | Desactive **Always Free** y cree la base con el tamaño mínimo. Consume crédito: bórrela el mismo día |
| El formulario rechaza la contraseña | No cumple la complejidad | 12 a 30 caracteres, con mayúscula, minúscula y número; sin comillas dobles y sin la palabra «admin» |
| No aparece la base que acaba de crear | Está mirando otro compartimento | Cambie el selector de compartimento a la izquierda de la lista |
| **Download wallet** no responde o falla | La base todavía no está **Available** | Espere a que termine el aprovisionamiento |
| Database actions no abre | La sesión de la consola expiró, o el navegador bloqueó la pestaña nueva | Vuelva a entrar y permita las ventanas emergentes del dominio de la consola |

### En el equipo

| Síntoma | Causa | Arreglo |
|---|---|---|

| | | |
|---|---|---|
| `sql: command not found` | SQLcl no está en el PATH | Agregue la carpeta `bin` de SQLcl al PATH y abra una terminal nueva |
| `sql -V` reporta menos de 25.2 | Versión sin servidor MCP | Actualice. No hay forma de rodearlo: `-mcp` no existe antes de 25.2 |
| SQLcl arranca y se cae sin mensaje claro | Java ausente o en versión no soportada | Instale JRE 17 o 21 y verifique con `java -version` |
| `set cloudconfig` falla | Ruta del wallet incorrecta, o contraseña del wallet equivocada | Use ruta absoluta. La contraseña es la del capítulo 6, no la de ADMIN |
| La conexión pide contraseña cada vez | Se guardó sin `-savepwd` | Vuelva a guardarla con `connect -save <nombre> -savepwd ...` |
| El servidor MCP conecta pero no encuentra conexiones | La conexión se guardó con otro usuario del sistema operativo, o en otro equipo | El almacén de SQLcl es por usuario del sistema. Guárdela con el mismo usuario que corre el cliente de IA |
| El inspector dice que el servidor no respondió en 30 s | El servidor no arrancó | Lea lo que el inspector imprime de la salida de error del servidor. Casi siempre es SQLcl antiguo o Java incompatible |
| El inspector imprime líneas que no son JSON | El servidor saluda antes de hablar el protocolo | Es normal. El inspector las ignora. Si lo hace a mano, ignórelas usted también |
| `--llamar` devuelve «no está en el catálogo» | La herramienta no existe en ese servidor | Es el comportamiento correcto: lo que no está declarado, no se ejecuta |
| El cliente de IA no ve el servidor | Ruta relativa a `sql` en la configuración | Use la ruta absoluta de `which sql` / `where sql`, y reinicie el cliente |
| El agente se conecta pero no ve tablas | Está usando `lab06lectura` y consulta una tabla sin sinónimo ni permiso | Es el comportamiento correcto. Solo hay acceso a las cuatro tablas del capítulo 11 |
| El `DELETE` de prueba **funciona** | La conexión en uso es `lab06` (ADMIN), no `lab06lectura` | Revise con qué conexión está conectado. Si de verdad es `AGENTE_RO`, revise los `GRANT`: sobra alguno |

## En la base

| Síntoma | Causa | Arreglo |
|---|---|---|
| `ORA-01031: insufficient privileges` al borrar con `AGENTE_RO` | El usuario solo tiene `SELECT` | **Es el resultado esperado.** No lo arregle |
| `ORA-00942: table or view does not exist` con `AGENTE_RO` | Falta el sinónimo o el `GRANT SELECT` sobre esa tabla | Ejecute de nuevo el bloque del capítulo 11 como `ADMIN` |
| La conexión se rechaza desde una red nueva | La lista de control de acceso de la base no incluye esa dirección | Consola › la base › edición del acceso de red. Agregue la dirección o deje la lista vacía |

| | | |
|---|---|---|
| El script del esquema falla en el `DROP` | Las tablas no existían todavía | El bloque `BEGIN ... END` lo contempla. Si falla igual, ejecute el resto del script sin ese bloque |

# 17. Limpieza

Este laboratorio deja cosas en dos sitios, y el que se olvida es el segundo.

## 17.1 En el equipo, primero

1. Borre las conexiones guardadas de SQLcl:

```
sql /nolog
```

```
connect -delete lab06
connect -delete lab06lectura
exit
```

1. Borre el wallet descargado. El archivo `.zip` y cualquier carpeta donde lo haya descomprimido.
2. Quite la entrada del servidor MCP de su cliente de IA. Si se configuró por archivo, edite el JSON y quite el bloque `sqlcl`. Si se configuró por comando, use la instrucción de eliminación del cliente.
3. Desinstale las Agent Skills si no las quiere conservar. No estorban: no ejecutan nada.

**CUIDADO · Las conexiones guardadas son credenciales reales en el equipo**

Una conexión guardada con `-savepwd` es una credencial almacenada, viva y utilizable por cualquier proceso que corra con su usuario del sistema operativo. Quedarse con ellas después del laboratorio es exactamente el descuido del que habla el capítulo 14.

Y hay un residuo que casi nadie limpia: **la entrada del servidor MCP en el cliente de IA sigue ahí**, apuntando a un ejecutable que va a intentar conectarse a una base que ya no existe. No es peligroso, pero es la prueba de que estas piezas no dejan rastro en ningún inventario de la organización. Si se le olvidó esta, pregúntese cuántas hay en los equipos de su equipo.

## 17.2 En la consola, después

**Consola de OCI** Oracle Database › Autonomous Database › `lab06-agentes` › More actions › Terminate

1. Entre a la pantalla de detalle de la base.

2. Abra **More actions** y escoja **Terminate**.
3. Escriba el nombre de la base para confirmar. Es irreversible y borra el esquema con todo lo que contiene.
4. Espere a que el recurso desaparezca de la lista.

Si creó un compartimento solo para esto, bórrelo después de que la base haya desaparecido. **Un compartimento con recursos dentro no se puede borrar**, y el borrado de compartimentos tarda. Si la política del capítulo 4 ya no se usa, bórrela también.

### 17.3 Qué sigue cobrando si se hace mal

| Si deja | Cobra |
|---|---|
| La base con **Always Free** activado, encendida | Nada |
| La base **sin** Always Free, encendida | Cómputo y almacenamiento |
| La base **sin** Always Free, detenida | Almacenamiento `[VALIDAR]` |
| El compartimento vacío | Nada |
| La política | Nada |

Con el nivel siempre gratuito no hay prisa por borrar la base. El motivo para hacerlo igual no es el costo: es que el esquema y los usuarios de este laboratorio son deliberadamente descuidados, y una base así no debería sobrevivir al ejercicio que la justificaba.

Para verificar que no queda nada cobrando, revise el consumo del compartimento en la sección de facturación y análisis de costos de la consola al día siguiente. Los datos de consumo tardan horas en consolidar.

## 18. Documentación oficial

Rutas de documentación del producto. Si necesita una página concreta, entre por estas y navegue: las rutas profundas cambian de versión a versión.

### Base de datos

- Autonomous Database Serverless — documentación del servicio <https://docs.oracle.com/en-us/iaas/autonomous-database-serverless/index.html>
- Documentación de Oracle Cloud Infrastructure — índice general <https://docs.oracle.com/en-us/iaas/Content/home.htm>
- Nivel siempre gratuito de Oracle Cloud <https://www.oracle.com/cloud/free/>

### Identidad y acceso

- Identity and Access Management — documentación <https://docs.oracle.com/en-us/iaas/Content/Identity/home.htm>

## Herramientas

- SQL Developer Command Line (SQLcl) — documentación <https://docs.oracle.com/en/database/oracle/sql-developer-command-line/>
- SQLcl — descarga <https://www.oracle.com/database/sqldeveloper/technologies/sqlcl/>
- Database Actions (SQL Developer Web) — documentación <https://docs.oracle.com/en/database/oracle/sql-developer-web/>

## Protocolo

- Model Context Protocol — especificación del protocolo <https://modelcontextprotocol.io/>

No es documentación de Oracle. Se incluye porque el capítulo 9 se apoya en ella: los métodos `initialize`, `notifications/initialized` y `tools/list`, y el formato de `inputSchema`, están definidos ahí y no en la documentación de SQLcl.

## Qué verificar antes de volver a dar este laboratorio

Este ecosistema se mueve rápido. Tres cosas conviene comprobar en la documentación del proveedor antes de cada sesión:

| Qué | Por qué importa |
|---|---|
| La versión mínima de SQLcl con servidor MCP | Cambia el prerrequisito del equipo, y es bloqueante |
| Los niveles de restricción y qué bloquea cada uno | Es el contenido central del capítulo 10 |
| La madurez declarada de cada servidor MCP | Algunos se publican explícitamente como pruebas de concepto, *no para producción*. Proponer uno de esos para un ambiente real sería irresponsable — y distinguirlos es justamente lo que enseña este laboratorio |

La tercera es la que más envejece, y es la que hay que comprobar, no recordar.