

3

.....

Configuración del entorno de trabajo: *Oracle AI Database 26ai Free, SQL Developer, SQL*Plus y SQLcl sobre Windows 11*

OBJETIVOS DEL CAPÍTULO

.....

Al terminar este capítulo, el lector será capaz de:

1. Instalar Oracle AI Database 26ai Free en Windows 11 y explicar qué componentes han quedado funcionando en el equipo.
2. Distinguir la base de datos contenedora (CDB) de la base de datos de trabajo (PDB) y saber a cuál conectarse en cada caso.
3. Conectarse al motor desde SQL*Plus y SQLcl construyendo cadenas de conexión EZConnect.
4. Instalar SQL Developer 24.3 y definir conexiones gráficas reutilizables.
5. Crear el usuario ATENEA, cargar el esquema de la academia con su guion y verificar que la instalación completa funciona de principio a fin.

3.1 EL MAPA DEL ENTORNO QUE VAMOS A MONTAR

.....

Antes de descargar nada conviene saber qué piezas vamos a instalar y cómo encajan, porque la instalación de Oracle deja en el equipo bastantes cosas con nombres parecidos y es fácil perderse. La figura 3.1 es el plano completo: a la derecha, el motor de la base de datos; en el centro, el listener, que recibe las conexiones; a la izquierda, las tres herramientas cliente con las que trabajaremos. Todo convivirá en el mismo PC con Windows 11, pero hablándose por red local exactamente igual que si estuvieran en máquinas distintas, que es como ocurre en cualquier empresa.

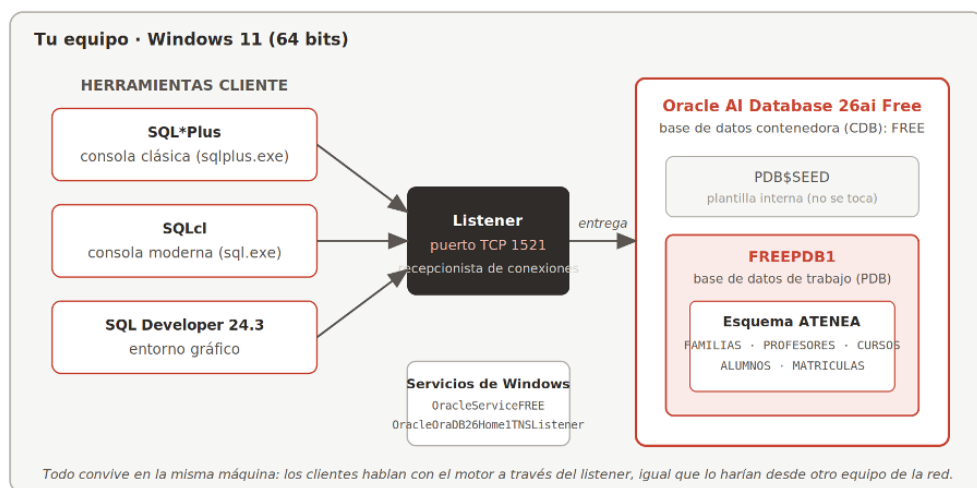


Figura 3.1. El entorno completo del manual: motor, listener y clientes conviviendo en el mismo equipo Windows 11.

Hay un concepto en la figura que merece explicación inmediata, porque condiciona cada conexión que hagamos a partir de hoy: la arquitectura **multitenant**. Desde hace varias versiones, una instalación de Oracle no contiene una base de datos, sino una base de datos *contenedora* (CDB, en nuestro caso llamada FREE) que aloja en su interior bases de datos *conectables* (PDB). La edición Free trae creada una PDB de trabajo llamada FREEPDB1, y esa será nuestra casa: ahí crearemos el usuario ATENEA y sus tablas, y ahí apuntarán todas nuestras conexiones. A la contenedora solo se asoman los administradores para tareas de mantenimiento; el desarrollador vive en la PDB. Si el lector recuerda una sola cosa de este apartado, que sea esta: **conectarse siempre a FREEPDB1**.

3.2 INSTALACIÓN DE ORACLE AI DATABASE 26AI FREE

Comprobemos primero que el equipo cumple los requisitos. Son modestos para un PC actual, pero el de la arquitectura es innegociable:

Requisito	Mínimo para Oracle AI Database 26ai Free en Windows
Sistema operativo	Windows 10/11 de 64 bits (x64). No existe versión de 32 bits.
Memoria RAM	4 GB como mínimo; 8 GB para trabajar con holgura junto a SQL Developer.
Espacio en disco	Unos 12 GB libres para la instalación y los ficheros de datos.
Permisos	Cuenta con derechos de administrador durante la instalación.
Red	Ninguna especial: todo funciona en local. El listener usará el puerto 1521.

La descarga se realiza desde la página oficial de Oracle Database Free (accesible desde oracle.com/database/free), eligiendo el paquete para **Windows x64**. Es un archivo ZIP de algo más de dos gigabytes; descomprímalo en una carpeta temporal y ejecute setup.exe como administrador (botón derecho, “Ejecutar como administrador”). El asistente es deliberadamente corto; estos son sus pasos y las decisiones que tomaremos en cada uno:

1. **Carpeta de destino.** Aceptaremos la propuesta, del estilo C:\app\<usuario>\product\26ai\, que se convertirá en el ORACLE_HOME del equipo. Evite rutas con espacios o caracteres acentuados: ahorran disgustos.
2. **Contraseña única.** El asistente pide una sola contraseña que asignará a las tres cuentas administrativas: SYS, SYSTEM y PDBADMIN. Elija una que recuerde y anótelas en lugar seguro; en este manual nos referiremos a ella como “la contraseña de instalación”.
3. **Resumen e instalación.** El proceso tarda entre cinco y quince minutos según el equipo. Al terminar, el asistente muestra las cadenas de conexión de la base de datos recién creada; no hace falta copiarlas, las construiremos a mano enseguida.

ATENCIÓN

La contraseña de instalación protege cuentas con poder absoluto sobre la base de datos. Dos errores típicos del primer día: olvidarla (obliga a maniobras de recuperación nada agradables) y usarla para el trabajo diario. SYS y SYSTEM son cuentas de administración; el trabajo cotidiano se hace siempre con usuarios normales como el ATENEA que crearemos en el apartado 3.5. Es la misma higiene que no usar la cuenta de administrador de Windows para navegar.

¿Qué ha quedado instalado? Además de los binarios en el ORACLE_HOME, dos **servicios de Windows** que pueden verse en services.msc: OracleServiceFREE, que es el motor propiamente dicho, y el servicio del listener. Ambos quedan configurados para arrancar automáticamente con el sistema, de modo que la base de datos estará disponible cada vez que encendamos el equipo. Si algún día se prefiere arrancarlos a mano para ahorrar memoria, basta cambiar su tipo de inicio a “Manual”; lo practicaremos en el ejercicio 3.5.

3.3 PRIMER CONTACTO: SQL*PLUS Y SQLCL

La instalación incluye SQL*Plus, así que podemos hablar con la base de datos sin instalar nada más. Abra el **Símbolo del sistema** (tecla Windows, escriba “cmd”, Intro) y construyamos la conexión con la sintaxis **EZConnect**, que la figura 3.2 disecciona pieza a pieza: usuario, arroba, máquina, puerto y nombre de servicio.

Anatomía de una conexión EZConnect



*La contraseña no se escribe nunca en la cadena: SQL*Plus la pedirá aparte, sin mostrarla en pantalla, y así no queda guardada en el historial de la consola.*

Figura 3.2. La cadena de conexión EZConnect, desmontada: con esta sintaxis nos conectaremos durante todo el manual.

Como todavía no existe el usuario ATENEA, esta primera conexión la haremos como SYSTEM contra la PDB de trabajo. Teclee la primera línea del listado 3.1, introduzca la contraseña de instalación cuando la pida (no se muestra al escribir, es normal) y compare su pantalla con la captura:

```
C:\Users\Alumno>sqlplus system@localhost:1521/FREEPDB1

SQL*Plus: Release 23.26.1.0.0 - Production on Thu Jun 11 09:42:07 2026
Version 23.26.1.0.0

Copyright (c) 1982, 2026, Oracle. All rights reserved.

Enter password:

Connected to:
Oracle AI Database 26ai Free Release 23.26.1.0.0 - Develop, Learn,
and Run for Free
Version 23.26.1.0.0

SQL>
```

Listado 3.1. Primera conexión a la base de datos: el banner confirma motor, edición y versión.

Ese SQL> final es el indicador de SQL*Plus: la base de datos está escuchando y espera órdenes. Démosle dos, para estrenarnos: una consulta a DUAL —una tabla especial de una sola fila que Oracle incluye precisamente para esto, para consultar valores que no salen de ninguna tabla— y la orden de salida:

```
SQL> SELECT sysdate FROM dual;

SYSDATE
-----
11-JUN-26

SQL> EXIT
Disconnected from Oracle AI Database 26ai Free Release 23.26.1.0.0
C:\Users\Alumno>
```

Listado 3.2. La función SYSDATE devuelve la fecha del servidor; DUAL nos presta su única fila para verla.

Junto a SQL*Plus conviene tener instalada **SQLcl**, la consola moderna de Oracle: acepta todo lo que acepta SQL*Plus y añade historial persistente, autocompletado con el tabulador y formatos de salida como CSV o JSON. Se descarga desde la web de Oracle como un ZIP que se descomprime donde se prefiera (por ejemplo C:\app\sqlcl); el ejecutable es sql.exe y requiere Java 11 o superior, que el propio paquete incluye en sus versiones recientes. La conexión usa la misma sintaxis EZConnect: sql system@localhost:1521/FREEPDB1. La usaremos de forma puntual cuando sus comodidades aporten algo; el ejercicio 3.4 propone el primer paseo.

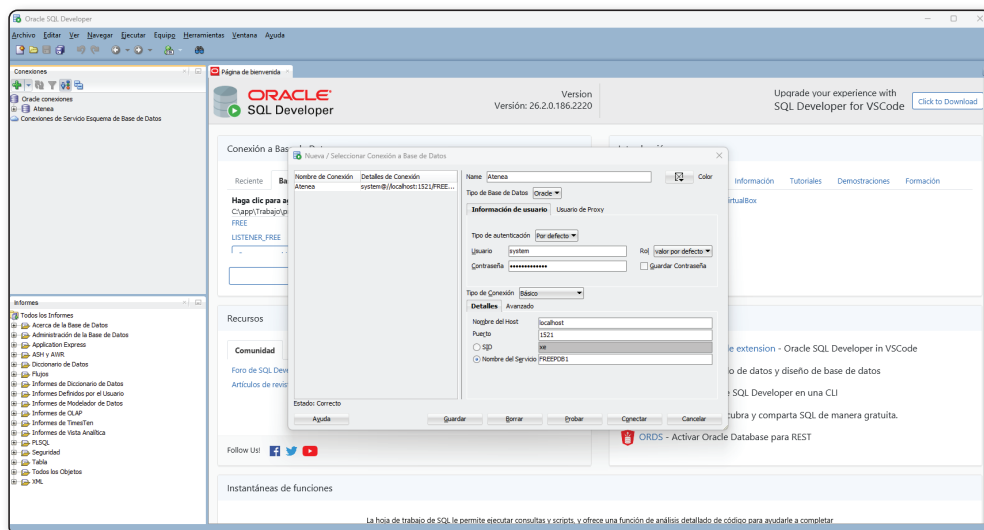
3.4 SQL DEVELOPER: INSTALACIÓN Y PRIMERA CONEXIÓN

SQL Developer 24.3 es gratuito y se descarga de la web de Oracle en una variante “Windows 64-bit with JDK included” que no requiere instalar Java por separado. Tampoco tiene instalador: el ZIP se descomprime en una carpeta estable —por ejemplo C:\app\sqldeveloper— y se ejecuta sqldeveloper.exe. En el primer arranque preguntará si queremos importar preferencias de versiones anteriores (no) y si deseamos enviar estadísticas de uso (a elección).

Definamos la conexión que usaremos a diario. Pulse el botón + verde del panel de conexiones (o menú Archivo, Nueva, Conexión de base de datos) y rellene el formulario con estos valores; el resto puede quedarse como está:

Campo del formulario	Valor para nuestro entorno
Nombre de la conexión	Atenea (FREEPDB1) — es solo una etiqueta descriptiva
Usuario y contraseña	system y la contraseña de instalación (la cambiaremos por atenea en cuanto exista)
Tipo de conexión	Básico
Nombre del host	localhost
Puerto	1521
Nombre del servicio	FREEPDB1 — atención: la opción es “Nombre del servicio”, no “SID”

El botón **Probar** debe responder “Estado: Correcto” en la esquina inferior izquierda del diálogo; entonces, **Guardar** y **Conectar**. Se abrirá una hoja de trabajo (worksheet) donde puede repetirse la consulta del listado 3.2 pulsando Ctrl+Intro, y a la izquierda, el árbol de la conexión: tablas, vistas, índices... De momento estará casi vacío; en unos minutos dejará de estarlo.



NOTA

¿Por qué “Nombre del servicio” y no “SID”? Son dos maneras históricas de identificar la base de datos de destino. El SID identifica la instancia completa (la contenedora FREE), mientras que el nombre de servicio permite apuntar a una PDB concreta, que es justo lo que necesitamos con FREEPDB1. En arquitectura multitenant la respuesta correcta es prácticamente siempre el nombre de servicio; confundir ambos campos es la causa número uno del error ORA-12505 en el primer día de clase.

3.5 EL USUARIO ATENEA Y LA CARGA DEL ESQUEMA

En Oracle, usuario y esquema son las dos caras de la misma moneda: crear el usuario ATENEA es crear el contenedor donde vivirán sus tablas. Lo haremos desde SQL*Plus conectados como SYSTEM a FREEPDB1, con tres sentencias: una CREATE USER y dos GRANT. No se preocupe ahora por su sintaxis al detalle —los privilegios son la materia del Bloque III—; hoy basta con entender el papel de cada línea, que el comentario indica:

```
SQL> -- el usuario y su contraseña (elija la suya)
SQL> CREATE USER atenea IDENTIFIED BY "Atenea#2026"
      2 QUOTA UNLIMITED ON users;
```

```
User created.
```

```
SQL> -- permiso para conectarse y para crear sus objetos
SQL> GRANT CREATE SESSION, CREATE TABLE, CREATE VIEW,
      2      CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER
      3      TO atenea;
```

```
Grant succeeded.
```

Listado 3.3. Nacimiento del usuario ATENEA: identidad, cuota de almacenamiento y privilegios de trabajo.

Queda poblar el esquema. El guion academia_atenea.sql —reproducido íntegro en el anexo B y disponible para descarga junto al manual— crea las cinco tablas que diseñamos en el capítulo 2, con sus claves y restricciones, e inserta los datos de ejemplo: 4 familias, 6 profesores, 12 cursos, 20 alumnos y 45 matrículas, con los que están calculadas todas las salidas de los capítulos siguientes. Guárdelo en C:\atenea\, conéctese ya como el nuevo usuario y ejecútelo con la orden @ de SQL*Plus, que lee y ejecuta un archivo de principio a fin:

```
C:\Users\Alumno>sqlplus atenea@localhost:1521/FREEPDB1
```

```
Enter password:
```

```
...
```

```
SQL> @C:\atenea\academia_atenea.sql
```

```
Table created.
```

```
Table created.
```

```
Table created.
```

```
Table created.
```

```
Table created.
```

```
1 row created.
```

```
1 row created.
```

```
... (87 filas insertadas en total) ...
```

```
Commit complete.
```

TABLA	FILAS

FAMILIAS	4
PROFESORES	6
CURSOS	12
ALUMNOS	20
MATRICULAS	45

Listado 3.4. Carga del esquema: el propio guion termina con una consulta de verificación que recuenta cada tabla.

Si su recuento coincide con el del listado 3.4, el entorno está completo y es idéntico al del manual: misma estructura, mismos datos, mismas salidas en cada ejemplo que viene. La primera ejecución mostrará, justo al principio, cinco errores ORA-00942 (“table or view does not exist”): son los DROP TABLE de limpieza intentando borrar tablas que aún no existían, y pueden ignorarse tranquilamente; están ahí para que el guion pueda relanzarse cuantas veces haga falta y dejar siempre el esquema recién hecho. Esa idempotencia nos dará mucha tranquilidad: cualquier experimento que estropee los datos se arregla relanzando el guion.

3.6 PRÁCTICA GUIADA 3: LISTA DE VERIFICACIÓN DEL ENTORNO

Cerremos el capítulo con una revisión sistemática, la misma que haríamos en un aula antes de empezar el curso. Cuatro comprobaciones, cada una con su resultado esperado; si las cuatro pasan, el equipo está listo para todo el Bloque I.

Comprobación 1: los servicios están en marcha

En services.msc, OracleServiceFREE y el servicio del listener deben figurar “En ejecución”. Alternativa de consola, más profesional: preguntar al propio listener, que responde con su estado y los servicios que atiende:

```
C:\Users\Alumno>lsnrctl status

LSNRCTL for 64-bit Windows: Version 23.26.1.0.0 - Production
...
STATUS of the LISTENER
-----
Alias                LISTENER
Uptime                0 days 2 hr. 14 min. 53 sec
...
Service "FREEPDB1" has 1 instance(s).
  Instance "FREE", status READY, has 1 handler(s) for this service...
The command completed successfully
```

Listado 3.5. El listener confirma que atiende el servicio FREEPDB1: la puerta está abierta.

Comprobación 2: el motor responde y es quien dice ser

Conectado como ATENEA, consulte la vista V\$VERSION, que identifica el software con precisión:

```
SQL> SELECT banner FROM v$version;
```

```
BANNER
```

```
-----  
Oracle AI Database 26ai Free Release 23.26.1.0.0 - Develop,  
Learn, and Run for Free
```

Listado 3.6. Verificación de la versión del motor desde dentro de la sesión.

Comprobación 3: las tablas y sus datos están donde deben

La vista USER_TABLES lista las tablas del usuario conectado (el diccionario de datos, del que es un primer aperitivo, se estudia en el capítulo 14):

```
SQL> SELECT table_name FROM user_tables ORDER BY table_name;
```

```
TABLE_NAME
```

```
-----  
ALUMNOS  
CURSOS  
FAMILIAS  
MATRICULAS  
PROFESORES
```

```
5 rows selected.
```

Listado 3.7. Las cinco tablas de la academia Atenea, listas para los capítulos de consulta.

Comprobación 4: SQL Developer ve lo mismo

Edite la conexión “Atenea (FREEPDB1)” para que use ya el usuario atenea, conéctese y despliegue el nodo Tablas del árbol: deben aparecer las cinco. Haga doble clic en MATRICULAS y visite la pestaña Datos; esas 45 filas, con sus calificaciones a medio rellenar, van a ser nuestras compañeras de viaje durante mucho tiempo. Merecen un vistazo de presentación.

3.7 RESUMEN DEL CAPÍTULO

El entorno del manual ya existe fuera del papel. Hemos instalado Oracle AI Database 26ai Free en Windows 11 y entendido lo que la instalación deja en el equipo: el motor como servicio de Windows, el listener atendiendo el puerto 1521 y la pareja contenedora-conectable de la arquitectura multitenant, con FREEPDB1 como nuestra base de trabajo. Nos hemos conectado con SQL*Plus mediante cadenas EZConnect, hemos conocido

SQLcl, hemos configurado SQL Developer con una conexión reutilizable y, sobre todo, hemos dado vida al diseño del capítulo 2: el usuario ATENEA existe, sus cinco tablas están creadas y sus 87 filas de datos cargadas y verificadas. La lista de comprobación de la práctica guiada queda, además, como protocolo de diagnóstico para cualquier sesión futura que amanezca torcida.

En el **capítulo 4** empieza lo que el lector probablemente estaba esperando desde la primera página: consultar. SELECT, FROM y WHERE, las tres palabras con las que se pregunta a una base de datos, ejecutadas una tras otra contra la academia Atenea. A partir de ahí, y durante once capítulos, no soltaremos el teclado.

4

.....

La consulta básica: SELECT, FROM y WHERE

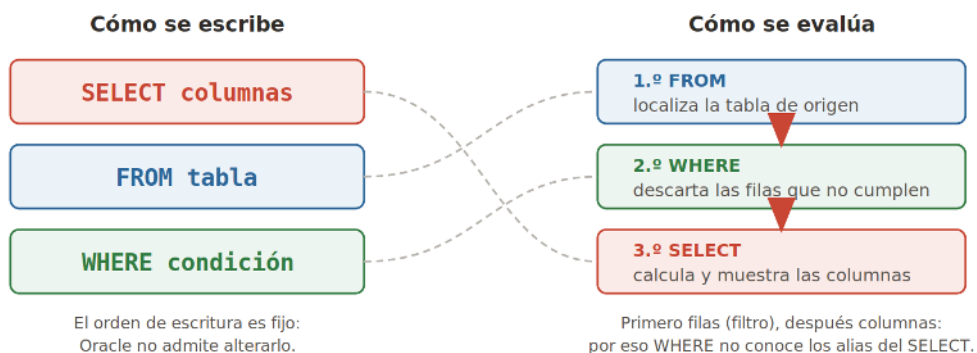
OBJETIVOS DEL CAPÍTULO

Al terminar este capítulo, el lector será capaz de:

1. Escribir consultas que recuperen las columnas deseadas de una tabla, con expresiones calculadas, alias legibles y concatenación de textos.
2. Eliminar duplicados de un resultado con DISTINCT y explicar por qué el orden de las filas no está garantizado.
3. Filtrar filas con WHERE usando comparadores sobre números, textos y fechas, escribiendo correctamente cada tipo de literal.
4. Predecir el comportamiento de NULL en comparaciones y cálculos, y usar IS NULL e IS NOT NULL donde corresponde.
5. Interpretar los errores ORA más frecuentes de las primeras semanas y corregir su causa.

4.1 ANATOMÍA DE LA CONSULTA

Toda consulta responde a la misma pregunta de tres piezas: *qué columnas* quiero ver (SELECT), *de qué tabla* salen (FROM) y *qué filas* me interesan (WHERE, que es opcional). Hay un matiz que conviene fijar desde hoy, porque explica varios comportamientos que de otro modo parecen caprichos: el orden en que se escriben las cláusulas no es el orden en que Oracle las evalúa. Se escribe SELECT-FROM-WHERE, pero se evalúa FROM-WHERE-SELECT: primero la tabla, después el filtro de filas y solo al final el cálculo de columnas. La figura 4.1 enfrenta ambos órdenes.



Memorice la pareja de órdenes: se escribe SELECT-FROM-WHERE, se piensa FROM-WHERE-SELECT.

Figura 4.1. La consulta se escribe en un orden y se evalúa en otro: conviene memorizar ambos.

Empecemos por la forma más perezosa de proyección: el asterisco, que significa “todas las columnas, en el orden en que se definieron”. Conectados como ATENEA (capítulo 3), pidamos la tabla más pequeña del esquema:

```
SQL> SELECT * FROM familias;

ID_FAMILIA NOMBRE
-----
1 Informática
2 Idiomas
3 Gestión empresarial
4 Diseño
```

Listado 4.1. El asterisco recupera todas las columnas; útil para explorar, impropio de código definitivo.

El asterisco es perfecto para curiosear, pero en scripts y aplicaciones se considera mala práctica: si mañana la tabla gana columnas, la consulta devolverá más de lo previsto, y el orden de columnas pasa a depender

de la definición de la tabla en lugar de depender de nosotros. La forma profesional es la **lista de columnas**, separadas por comas y en el orden que queramos verlas:

```
SQL> SELECT nombre, horas, precio FROM cursos;
```

NOMBRE	HORAS	PRECIO
-----	-----	-----
Iniciación a SQL	50	290
Programación en Python	80	420
Ofimática avanzada	40	210
Inglés B1	90	380
Inglés B2	90	410
Alemán A1	60	350
Nóminas y seguros sociales	45	260
Contabilidad para pymes	70	340
Diseño con Illustrator	55	330
Retoque fotográfico	35	195
Bases de datos Oracle	100	540
Atención al cliente	30	150

12 rows selected.

Listado 4.2. Proyección con lista de columnas: el catálogo completo de la academia.

NOTA

Sobre el aspecto de las salidas en este manual: SQL*Plus, por defecto, da a cada columna de texto el ancho de su definición (una VARCHAR2(60) ocupa 60 caracteres, aunque el dato mida diez), lo que produce listados que no caben en una página. Para que cada salida quepa, las capturas del manual usan anchos ajustados al contenido, como hacen SQLcl y SQL Developer automáticamente y como puede hacerse en SQL*Plus con la orden COLUMN. Los datos, su orden y los mensajes finales son siempre los reales.

4.2 EXPRESIONES, ALIAS Y CONCATENACIÓN

La lista del SELECT no se limita a nombres de columnas: admite **expresiones** construidas con los operadores aritméticos de toda la vida (+, -, *, / y los paréntesis para agrupar). La tabla original no cambia jamás: el cálculo ocurre al vuelo, fila a fila, solo en el resultado. Calculemos el salario anual de la plantilla, que en la academia se paga en catorce mensualidades:

```
SQL> SELECT apellidos, salario, salario * 14
2      FROM profesores;
```

APELLIDOS	SALARIO	SALARIO*14
Soto Vargas	2350	32900
Gil Ramos	2180	30520
Navarro Prieto	2520	35280
Ortiz Lara	1990	27860
Márquez Salas	2275	31850
Castro Ruiz	1875	26250

6 rows selected.

Listado 4.3. Una expresión aritmética en el SELECT; la cabecera resultante, SALARIO*14, pide a gritos un alias.

Esa cabecera autogenerada es legal pero fea, y aquí entran los **alias de columna**: un nombre alternativo que bautiza la columna en el resultado. Se escribe tras la expresión, opcionalmente precedido de AS. Sin comillas, Oracle lo guarda en mayúsculas y solo admite letras, números y guion bajo; entre **comillas dobles** se respetan mayúsculas, espacios y acentos. Compárense ambos estilos:

```
SQL> SELECT apellidos,
2           salario
3           salario * 14
4      FROM profesores;
           AS mensual,
           AS "Salario anual"
```

APELLIDOS	MENSUAL	Salario anual
Soto Vargas	2350	32900
Gil Ramos	2180	30520
Navarro Prieto	2520	35280
Ortiz Lara	1990	27860
Márquez Salas	2275	31850
Castro Ruiz	1875	26250

6 rows selected.

Listado 4.4. Alias sin comillas (sale en mayúsculas) y alias con comillas dobles (se respeta tal cual).

ATENCIÓN

Las comillas tienen reparto fijo en Oracle y confundirlas es el tropiezo más común del principiante: las simples ('Espartinas') encierran literales de texto, es decir, datos; las dobles ("Salario anual") encierran identificadores, es decir, nombres de columnas o alias. Escribir WHERE localidad = "Espartinas" no busca un texto: busca una columna llamada Espartinas, que no existe, y produce el error ORA-00904 que veremos en el apartado 4.6.

Para textos, el operador estrella es la **concatenación**, que en Oracle se escribe con la doble barra vertical || y pega cadenas y columnas en una sola. Es el modo natural de componer nombres completos:

```
SQL> SELECT nombre || ' ' || apellidos AS profesor, email
2 FROM profesores;
```

PROFESOR	EMAIL

Pablo Soto Vargas	psoto@atenea.es
Lucía Gil Ramos	lgil@atenea.es
Andrés Navarro Prieto	anavarro@atenea.es
Carmen Ortiz Lara	cortiz@atenea.es
Hugo Márquez Salas	hmarquez@atenea.es
Elena Castro Ruiz	ecastro@atenea.es

6 rows selected.

Listado 4.5. Concatenación con ||: el espacio entre nombre y apellidos es un literal de un carácter.

4.3 ELIMINAR DUPLICADOS: DISTINCT

Una proyección devuelve tantas filas como tenga la tabla, repeticiones incluidas: si pedimos la localidad de los veinte alumnos, Espartinas saldrá tres veces. Cuando lo que interesa es el catálogo de valores distintos, se antepone DISTINCT a la lista de columnas:

```
SQL> SELECT DISTINCT localidad FROM alumnos;
```

LOCALIDAD

Espartinas
Bormujos
Sevilla
Tomares
Mairena del Aljarafe
Camas
Gines
Umbrete
Sanlúcar la Mayor
Valencina de la Concepción

10 rows selected.

Listado 4.6. Las diez localidades distintas del alumnado: veinte filas de origen, diez en el resultado.

Dos avisos sobre este resultado. Primero: DISTINCT actúa sobre la *combinación completa* de columnas pedidas; con SELECT DISTINCT nombre, localidad se eliminarían solo las filas idénticas en ambas cosas a la vez. Segundo: el orden en que han salido las localidades no está garantizado en absoluto; otro día, u otra instalación, puede devolverlas en otro orden, porque sin cláusula de ordenación Oracle entrega las filas como mejor le conviene. Ordenar es asunto del capítulo 5, y hasta entonces ningún listado de este manual debe leerse como “ordenado”.

4.4 EL FILTRO WHERE

La cláusula WHERE somete cada fila a una condición y deja pasar solo las que la cumplen. Los comparadores básicos son los esperables —=, <, >, <=, >=— más el de desigualdad, que en Oracle se escribe <> (también se acepta !=). Lo que cambia según el tipo de dato es cómo se escribe el literal contra el que comparamos. Con números, tal cual:

```
SQL> SELECT nombre, precio FROM cursos
      2  WHERE precio > 350;
```

NOMBRE	PRECIO
Programación en Python	420
Inglés B1	380
Inglés B2	410
Bases de datos Oracle	540

Listado 4.7. Filtro numérico: los cursos de más de 350 euros.

Con textos, el literal va entre comillas simples y la comparación distingue mayúsculas de minúsculas: ‘Espartinas’ y ‘ESPARTINAS’ son valores diferentes, y buscar el segundo no encontraría nada en nuestra tabla. (Cómo buscar ignorando esa distinción se resuelve con las funciones del capítulo 6.)

```
SQL> SELECT nombre, apellidos, localidad FROM alumnos
      2  WHERE localidad = 'Espartinas';
```

NOMBRE	APELLIDOS	LOCALIDAD
Marta	Reina Calvo	Espartinas
Pablo	Serrano Hidalgo	Espartinas
Paula	Méndez Sosa	Espartinas

Listado 4.8. Filtro de texto: comillas simples y sensibilidad a mayúsculas.

Con fechas, la forma más limpia y portable es el literal DATE 'AAAA-MM-DD', que no depende de ningún formato regional. ¿Qué profesores se incorporaron después del 1 de enero de 2021?

```
SQL> SELECT nombre, apellidos, fecha_alta FROM profesores
2  WHERE fecha_alta > DATE '2021-01-01';
```

NOMBRE	APELLIDOS	FECHA_ALTA
Lucía	Gil Ramos	03-FEB-21
Carmen	Ortiz Lara	28-JAN-23
Elena	Castro Ruiz	15-APR-24

Listado 4.9. Filtro de fechas con el literal DATE, independiente de la configuración regional.

NOTA

El formato con el que SQL*Plus muestra las fechas (aquí, 03-FEB-21) depende de la configuración de idioma de la sesión, así que en un Windows en español es habitual ver 03/02/21. No es un error ni una diferencia de datos: es solo presentación. En el capítulo 6 aprenderemos a decidir nosotros el formato con la función TO_CHAR y a no depender nunca del azar regional; mientras tanto, para escribir fechas en condiciones, el literal DATE del listado 4.9 funciona igual en cualquier máquina.

4.5 EL CASO ESPECIAL DE NULL

En el capítulo 2 dejamos doce calificaciones sin valor: los cursos aún en marcha. Es el momento de aprender a preguntarles. El instinto dice WHERE calificacion = NULL, y el instinto se equivoca:

```
SQL> SELECT id_matricula, calificacion FROM matriculas
2  WHERE calificacion = NULL;
```

no rows selected

Listado 4.10. La trampa clásica: comparar con NULL mediante el igual no encuentra nada... y no da error.

NULL significa “valor desconocido”, y comparar cualquier cosa con un desconocido no da verdadero ni falso: da *desconocido*, y WHERE solo deja pasar lo verdadero. Ni siquiera NULL = NULL es verdadero (¿son iguales dos cantidades que ignoramos?). Lo retorcido del asunto es que la consulta no falla: devuelve cero filas en silencio, que es la peor manera de equivocarse. Para preguntar por la ausencia existe un operador específico, IS NULL (y su contrario IS NOT NULL):

```
SQL> SELECT id_matricula, id_curso, calificacion FROM matriculas
2   WHERE calificacion IS NULL;
```

ID_MATRICULA	ID_CURSO	CALIFICACION
1034	105	
1035	105	
1036	105	
1037	106	
1038	106	
1039	111	
1040	111	
1041	111	
1042	111	
1043	111	
1044	111	
1045	112	

12 rows selected.

Listado 4.11. IS NULL localiza las doce matrículas pendientes de calificar: los cuatro cursos en marcha.

NULL también se propaga por la aritmética: cualquier cálculo con un desconocido produce un desconocido. Obsérvese en el expediente de Marta Reina (alumna 1), que tiene dos cursos calificados y uno en marcha:

```
SQL> SELECT id_matricula, calificacion, calificacion + 0.5
2   FROM matriculas
3   WHERE id_alumno = 1;
```

ID_MATRICULA	CALIFICACION	CALIFICACION+0.5
1001	8.5	9
1009	9.3	9.8
1039		

Listado 4.12. Propagación: 8,5 + 0,5 es 9, pero NULL + 0,5 sigue siendo NULL.

Esa tercera fila vacía no es un fallo de la captura: la matrícula 1039 (el curso de Oracle, en marcha) no tiene nota, y sumarle medio punto a lo desconocido sigue siendo desconocido. Cuando haga falta sustituir el NULL por un valor de trabajo —un cero, un texto “pendiente”— usaremos la función NVL, primera invitada del capítulo 6.

4.6 APRENDER A LEER LOS ERRORES

Tres errores acaparan las primeras semanas de cualquier curso de SQL, y los tres se diagnostican en segundos cuando se sabe qué mirar. Provoquémoslos a propósito:

```
SQL> SELECT nombre FROM curso;
SELECT nombre FROM curso
                *

ERROR at line 1:
ORA-00942: table or view does not exist

SQL> SELECT nombe FROM cursos;
SELECT nombe FROM cursos
                *

ERROR at line 1:
ORA-00904: "NOMBE": invalid identifier

SQL> SELECT nombre precio FROM cursos;
SELECT nombre precio FROM cursos
                *

ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

Listado 4.13. La triada del principiante: tabla mal escrita, columna mal escrita y coma olvidada.

ORA-00942 dice que la tabla no existe con ese nombre para este usuario (aquí, curso por cursos; también aparece al conectarse con el usuario equivocado). **ORA-00904** señala un identificador inválido: una columna mal escrita o, como vimos, un literal con comillas dobles. Y el desconcertante **ORA-00923** rara vez significa que falte el FROM: casi siempre falta una *coma* entre dos columnas, con lo que Oracle interpreta la segunda como alias de la primera y se pierde. El asterisco bajo la sentencia marca dónde se rindió el analizador, que suele estar muy cerca del culpable. Acostúmbrese a leer el código del error, la palabra entrecomillada y la posición del asterisco, por ese orden: con esas tres pistas, la inmensa mayoría de los errores se corrige sin pensar.

4.7 PRÁCTICA GUIADA 4: LA TARIFA DE INFORMÁTICA

La dirección de Atenea pide un listado de tarifas de los cursos de informática (familia 1) para el folleto del próximo trimestre: precio oficial, precio con el 10 % de descuento por pago anticipado y precio por hora. Construyámoslo por capas, ejecutando y mirando en cada paso, que es como se trabaja de verdad. Primera capa: las columnas en bruto, filtrando ya la familia:

```
SQL> SELECT nombre, horas, precio FROM cursos
2   WHERE id_familia = 1;
```

NOMBRE	HORAS	PRECIO
Iniciación a SQL	50	290
Programación en Python	80	420
Ofimática avanzada	40	210
Bases de datos Oracle	100	540

Listado 4.14. Paso 1: comprobar que el filtro selecciona las filas correctas antes de complicar el SELECT.

Segunda capa: las dos columnas calculadas. El descuento es $\text{precio} * 0.9$; el precio por hora, $\text{precio} / \text{horas}$, envuelto en $\text{ROUND}(\text{expr}, 2)$ —un pequeño anticipo del capítulo 6— para no arrastrar decimales infinitos:

```
SQL> SELECT nombre, precio, precio * 0.9,
2   ROUND(precio / horas, 2)
3   FROM cursos
4   WHERE id_familia = 1;
```

NOMBRE	PRECIO	PRECIO*0.9	ROUND(PRECIO/HORAS,2)
Iniciación a SQL	290	261	5.8
Programación en Python	420	378	5.25
Ofimática avanzada	210	189	5.25
Bases de datos Oracle	540	486	5.4

Listado 4.15. Paso 2: las expresiones funcionan; las cabeceras, todavía impresentables.

Tercera capa: alias dignos de un folleto y un último requisito de dirección, que llega —como siempre— a última hora: solo cursos de 290 euros en adelante. Nótese que el filtro usa la columna precio, no el alias TARIFA: como enseñó la figura 4.1, WHERE se evalúa antes que SELECT y los alias aún no existen cuando se filtra.

```
SQL> SELECT nombre                AS curso,
2   precio                        AS tarifa,
3   precio * 0.9                  AS "Con descuento",
4   ROUND(precio / horas, 2)      AS "Precio/hora"
5   FROM cursos
6   WHERE id_familia = 1
7   AND precio >= 290;
```

CURSO	TARIFA	Con descuento	Precio/hora
Iniciación a SQL	290	261	5.8
Programación en Python	420	378	5.25
Bases de datos Oracle	540	486	5.4

Listado 4.16. Paso 3: la tarifa terminada. El AND que combina condiciones se estudia a fondo en el capítulo 5.

4.8 RESUMEN DEL CAPÍTULO

Ya consultamos. `SELECT` decide las columnas —nombres, expresiones aritméticas, concatenaciones con `||`— y los alias las bautizan, en mayúsculas sin comillas o tal cual entre comillas dobles; `FROM` aporta la tabla; `WHERE` criba las filas con comparadores cuyos literales se escriben según el tipo: números desnudos, textos entre comillas simples y fechas con el literal `DATE`. `DISTINCT` depura duplicados sobre la combinación completa de columnas, y ningún resultado tiene orden garantizado mientras no aprendamos a pedirlo. `NULL` exige trato aparte: no se compara con el igual sino con `IS NULL`, y contamina cualquier cálculo en el que participe. Y los errores han dejado de ser jeroglíficos: código, palabra entrecomillada y asterisco de posición bastan para diagnosticar la tríada `ORA-00942`, `ORA-00904` y `ORA-00923`.

En el **capítulo 5** el filtro y el resultado ganarán músculo: combinaremos condiciones con `AND`, `OR` y `NOT` y su tabla de precedencias, estrenaremos los operadores `BETWEEN`, `IN` y `LIKE` —el de los comodines— y pondremos orden de verdad con `ORDER BY`, incluido el destino de los `NULL` al ordenar. La academia Atenea seguirá poniendo los datos.

5

Ordenación y filtrado avanzado: *ORDER BY*, operadores y lógica de condiciones

OBJETIVOS DEL CAPÍTULO

Al terminar este capítulo, el lector será capaz de:

1. Ordenar resultados por una o varias columnas, en sentido ascendente o descendente, por alias y controlando la posición de los nulos.
2. Combinar condiciones con `AND`, `OR` y `NOT` aplicando correctamente la precedencia y los paréntesis.
3. Expresar rangos con `BETWEEN`, pertenencia a listas con `IN` y patrones de texto con `LIKE` y sus comodines.