Oracle Database: Conceptos Fundamentales de SQL I

Volumen I • Guía del Alumno

D64258CS10
Edición 1.0
Enero de 2010
D73590

ORACLE

Autores

Salome Clement
Brian Pottle
Puja Singh

Colaboradores y Revisores Técnicos

Anjulaponni Azhagulekshmi
Clair Bennett
Zarko Cesljas
Yanti Chang
Gerlinde Frenzen
Steve Friedberg
Joel Goodman
Nancy Greenberg
Pedro Neves
Manish Pawar
Surya Rekha
Helen Robertson
Lauran Serhal
Hilda Simon
Tulika Srivastava

Redactor

Amitha Narayan

Diseñador Gráfico

Rajiv Chandrabhanu

Publicador

Jobi Varghese

Contenido

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Definir los objetivos del curso
- Enumerar las funciones de Oracle Database 11*g*
- Describir el aspecto teórico y físico de una base de datos relacional
- Describir la implantación del servidor de Oracle de RDBMS y el sistema de gestión de bases de datos relacionales de objetos (ORDBMS)
- Identificar los entornos de desarrollo que se pueden utilizar para este curso
- Describir la base de datos y el esquema utilizados en este curso

ORACLE

Objetivos

En esta lección comprenderá el sistema de gestión de bases de datos relacionales (RDBMS) y el sistema de gestión de bases de datos relacionales de objetos (ORDBMS). Se presentan también Oracle SQL Developer y SQL*Plus como entornos de desarrollo utilizados para ejecutar sentencias SQL y para objetivos de formateo y creación de informes.

Agenda

- Objetivos del curso, agenda y apéndices utilizados en este curso
- Visión general de Oracle Database 11*g* y productos relacionados
- Visión general de los conceptos y terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- Esquema `HR` y tablas utilizadas en este curso
- Documentación y recursos adicionales de Oracle Database 11*g*

ORACLE

Objetivos del Curso

Al finalizar este curso, debería estar capacitado para:

- Identificar los principales componentes de Oracle Database
- Recuperar datos de filas y columnas de las tablas con la sentencia `SELECT`
- Crear informes de datos ordenados y restringidos
- Utilizar funciones SQL para generar y recuperar datos personalizados
- Ejecutar consultas complejas para recuperar datos de varias tablas
- Ejecutar sentencias del lenguaje de manipulación de datos (DML) para actualizar datos en Oracle Database
- Ejecutar sentencias de lenguaje de definición de datos (DDL) para crear y gestionar objetos de esquema

ORACLE

Objetivos del Curso

Este curso ofrece una introducción a la tecnología de Oracle Database. En esta clase, aprenderá los conceptos básicos relativos a las bases de datos relacionales y al potente lenguaje de programación SQL. El curso proporciona los conocimientos esenciales de SQL que permiten escribir consultas en una o varias tablas, manipular datos en tablas, crear objetos de bases de datos y realizar consultas en los metadatos.

Agenda

- Primer día:
  - Introducción
  - Recuperación de Datos mediante la Sentencia SQL `SELECT`
  - Restricción y Ordenación de Datos
  - Uso de Funciones de Una Sola Fila para Personalizar la Salida
  - Uso de Funciones de Conversión y Expresiones Condicionales
- Segundo día:
  - Informes de Datos Agregados con Funciones de Grupo
  - Visualización de Datos de Varias Tablas Utilizando Uniones
  - Uso de Subconsultas para Solucionar Consultas
  - Uso de los Operadores de Definición

ORACLE

Agenda

- Tercer día:
  - Manipulación de Datos
  - Uso de Sentencias DDL para Crear y Gestionar Tablas
  - Creación de Otros Objetos de Esquema

Apéndices Utilizados en el Curso

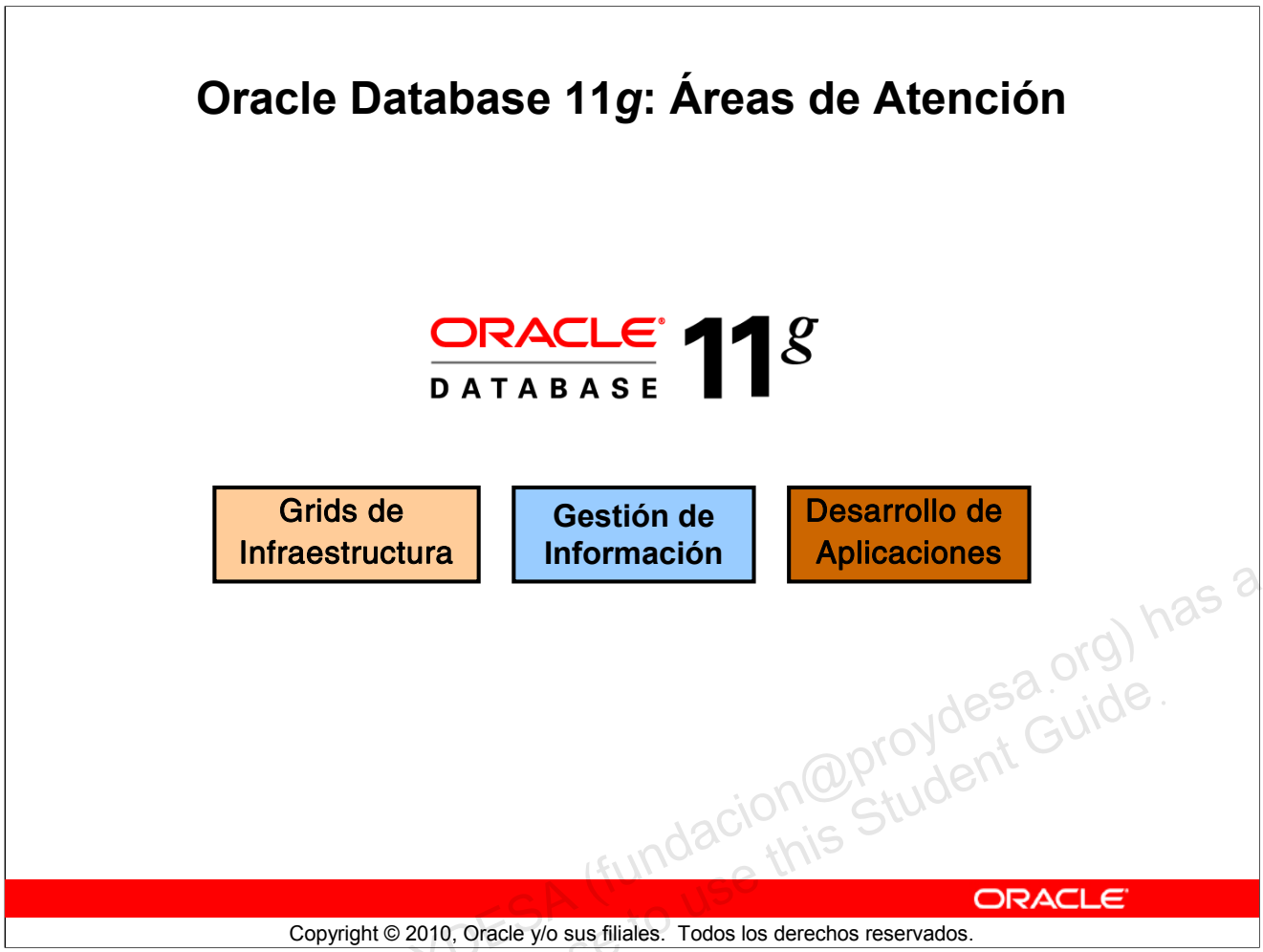
- Apéndice A: Prácticas y Soluciones
- Apéndice B: Descripciones de las Tablas
- Apéndice C: Uso de SQL Developer
- Apéndice D: Uso de SQL*Plus
- Apéndice E: Uso de JDeveloper
- Apéndice F: Sintaxis de Unión en Oracle
- Apéndice AP: Prácticas y Soluciones Adicionales

ORACLE

Agenda

- Objetivos del curso, agenda y apéndices utilizados en este curso
- Visión general de Oracle Database 11*g* y productos relacionados
- Visión general de los conceptos y terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- Esquema `HR` y tablas utilizadas en este curso
- Documentación y recursos adicionales de Oracle Database 11*g*

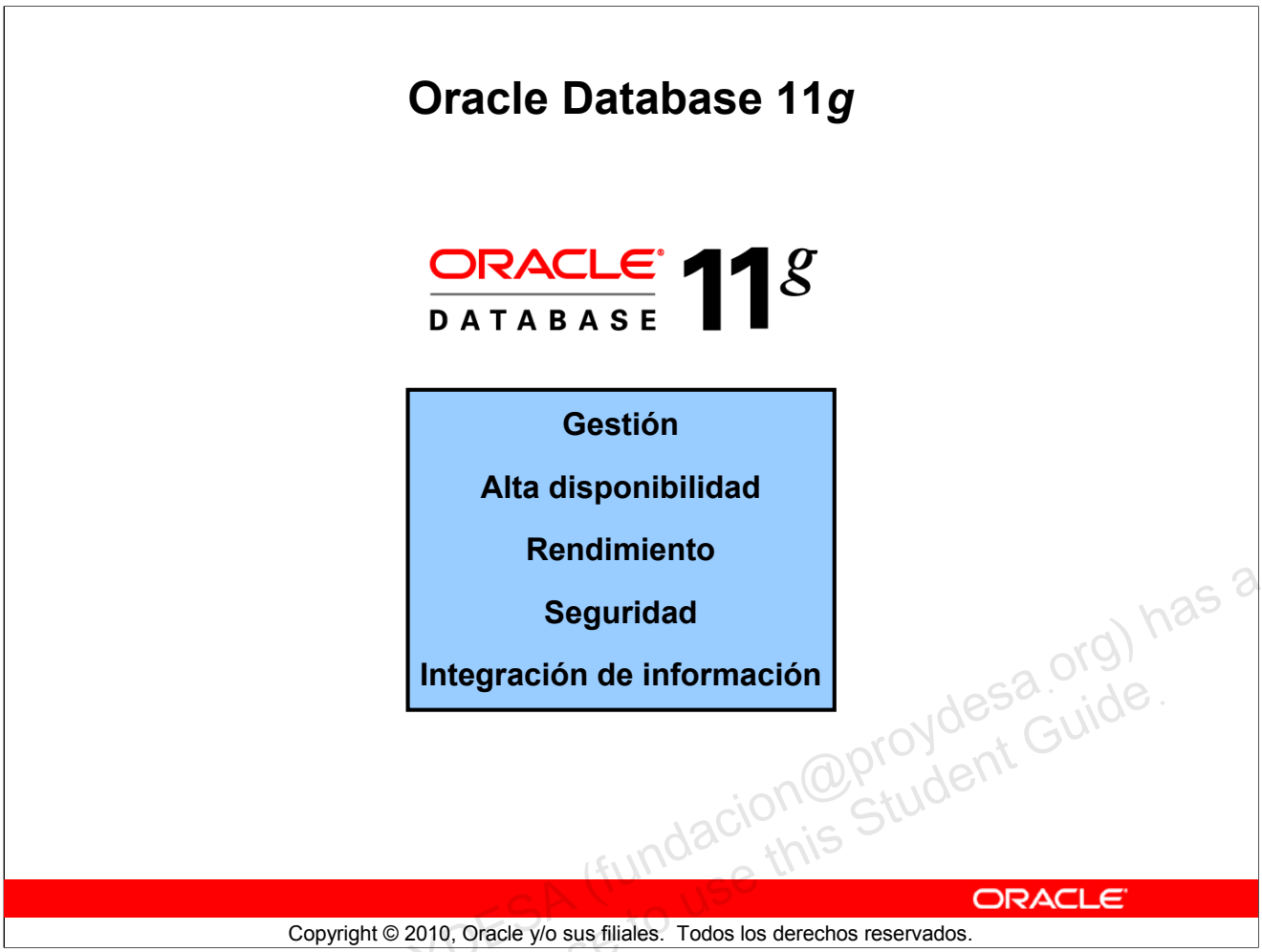
ORACLE



Oracle Database 11*g*: Áreas de Atención

Oracle Database 11*g* ofrece amplias funciones en las siguientes áreas de atención:

- **Grids de Infraestructura:** la tecnología de Grid de la infraestructura Oracle permite que se generen pools de almacenamiento y servidores de bajo costo que ofrecen la calidad de servicio más alta en lo referente a gestionabilidad, alta disponibilidad y rendimiento. Oracle Database 11*g* consolida y amplía las ventajas de Grid Computing. Aparte de aprovechar al máximo Grid Computing, Oracle Database 11*g* tiene funciones de garantía de cambios para gestionar cambios de una manera rentable y controlada.
- **Gestión de Información:** Oracle Database 11*g* amplía las capacidades de gestión de información existente en la gestión de contenido, integración de información y áreas de gestión del ciclo de vida de la información. Oracle proporciona gestión de contenido de tipos de dato avanzados, como Extensible Markup Language (XML), texto, espacial, multimedia, imágines médicas y tecnologías semánticas.
- **Desarrollo de Aplicaciones:** Oracle Database 11*g* tiene capacidades para utilizar y gestionar todos los entornos de desarrollo para aplicaciones principales, como PL/SQL, Java/JDBC, .NET y Windows, PHP, SQL Developer y Application Express.



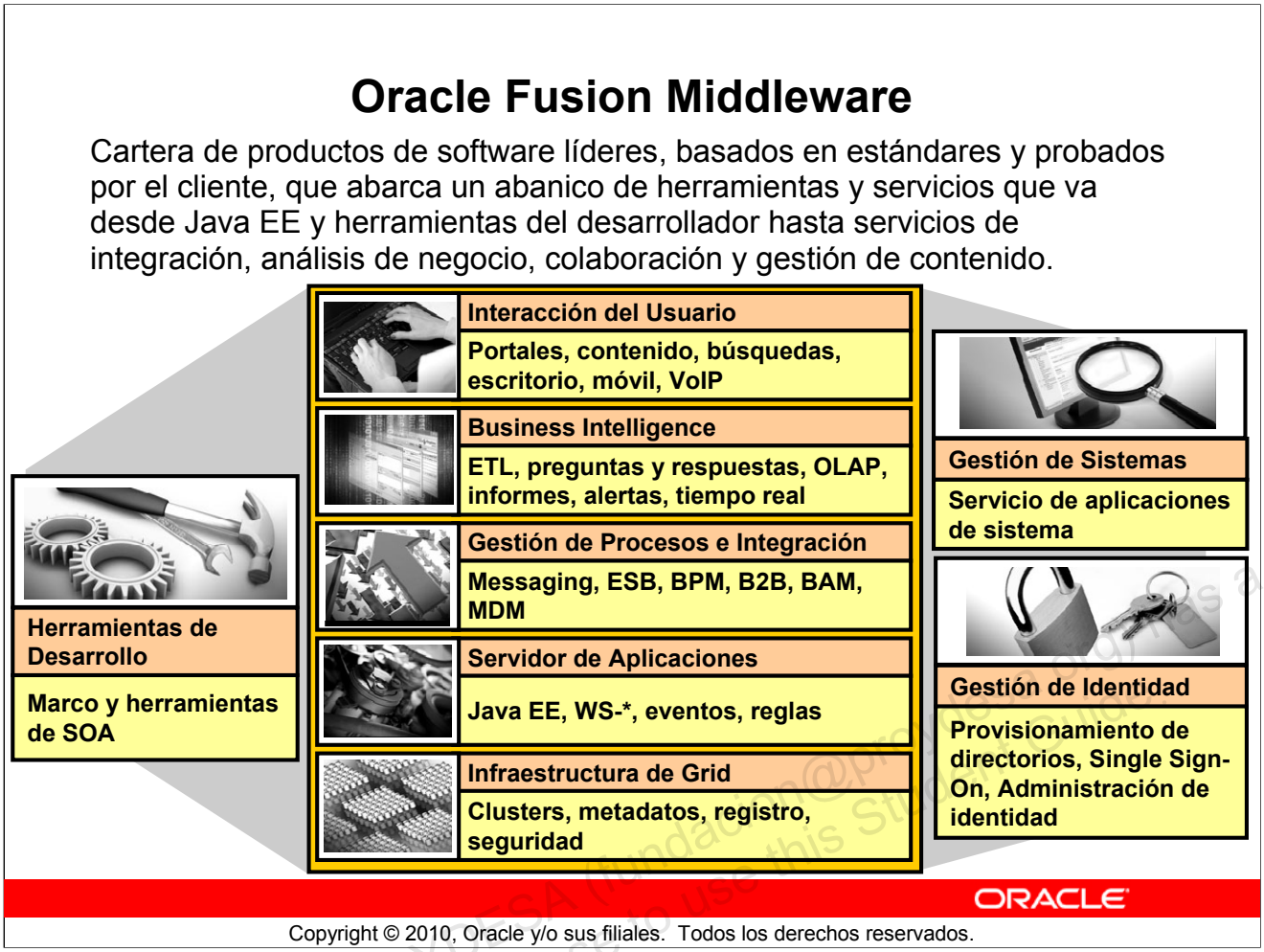
Oracle Database 11*g*

Las organizaciones tienen que soportar varios terabytes de información para usuarios que demanda un acceso rápido y seguro a aplicaciones de negocio durante todo el día. Los sistemas de base de datos deben ser fiables y se deben poder recuperar rápidamente en caso de cualquier tipo de fallo. Oracle Database 11*g* está diseñada junto con las siguientes áreas de funciones para ayudar a las organizaciones a gestionar los grids de infraestructura de forma sencilla y proporcionar un servicio de gran calidad.

- **Gestión:** mediante las funciones de garantía de cambios, automatización de gestión y diagnóstico de fallos, se puede aumentar la productividad de los administradores de bases de datos (DBA), reducir costos, minimizar errores y maximizar la calidad del servicio. Algunas de las funciones útiles que fomentan una mejor gestión son utilidad de reproducción de bases de datos, SQL Performance Analyzer y la utilidad de ajuste automático de SQL.
- **Alta disponibilidad:** mediante el uso de las funciones de alta disponibilidad, puede reducir el riesgo de tiempo de inactividad y pérdida de datos. Estas funciones mejoran las operaciones en línea y permiten realizar actualizaciones de la base de datos de una forma más rápida.

Oracle Database 11*g* (continuación)

- **Rendimiento:** el uso de funciones como archivos seguros, compresión para procesamiento de transacciones en línea (OLTP), optimizaciones de Real Application Clusters (RAC), cachés de consultas de resultados, etc., puede mejorar en gran medida el rendimiento de la base de datos. Oracle Database 11*g* permite a las organizaciones gestionar sistemas grandes, escalables, transaccionales y de almacenes de datos que proporciona un acceso rápido a los datos mediante el almacenamiento modular de bajo coste.
- **Seguridad:** Oracle Database 11*g* ayuda a las organizaciones a proteger la información con configuraciones de seguridad únicas, enmascaramiento y cifrado de datos, así como capacidades de auditoría sofisticadas. Proporciona una plataforma escalable y segura para un acceso rápido y fiable a todos los tipos de información mediante las interfaces estándar de la industria.
- **Integración de Información:** Oracle Database 11*g* tiene funciones para mejorar la integración de datos en toda la empresa. También soporta capacidades de gestión del ciclo de vida de la información avanzadas. Esto le ayudará a gestionar el cambio de datos en la base de datos.



Oracle Fusion Middleware

Oracle Fusion Middleware es una familia de productos completa y bien integrada que ofrece un soporte completo para el desarrollo, el despliegue y la gestión de Service-Oriented Architecture (SOA). SOA facilita el desarrollo de servicios de negocio modulares que se pueden integrar de forma sencilla y volver a utilizar, lo que reduce los costos de mantenimiento y desarrollo y proporciona una gran calidad de los servicios. La arquitectura de conexión de Oracle Fusion Middleware permite aprovechar la inversión en cualquier aplicación, sistema o tecnología existente. Su inquebrantable tecnología principal minimiza la interrupción provocada por interrupciones planificadas y no planificadas.

Algunos de los productos de la familia de Oracle Fusion Middleware son:

- **Enterprise Application Server:** Application Server
- **Gestión de Procesos e Integración:** BPEL Process Manager, Oracle Business Process Analysis Suite
- **Herramientas de Desarrollo:** Oracle Application Development Framework, JDeveloper, SOA Suite
- **Business Intelligence:** Oracle Business Activity Monitoring, Oracle Data Integrator
- **Gestión de sistemas:** Enterprise Manager
- **Oracle Identity Management:** Oracle Identity Management
- **Content Management:** Oracle Content Database Suite
- **User Interaction:** Portal, WebCenter

Oracle Enterprise Manager Grid Control

- Gestión eficiente de Oracle Fusion Middleware
- Simplificación de la gestión de ciclo de vida de la infraestructura y aplicación
- Capacidades de gestión de aplicaciones y administración de base de datos mejoradas

Oracle Enterprise Manager Grid Control

Al abarcar aplicaciones, software intermedio y gestión de bases de datos, Oracle Enterprise Manager Grid Control proporciona gestión empresarial integrada para sistemas ya sean de Oracle o no.

Oracle Enterprise Manager Grid Control proporciona capacidades de gestión de Oracle Fusion Middleware avanzadas para los servicios en los que confían las aplicaciones de negocio, incluidos SOA, Business Activity Monitoring e Identity Management.

- **Funcionalidad de gestión de amplio rango**: se encuentra disponible para aplicaciones que incluyen gestión de nivel de servicios, gestión del rendimiento de la aplicación, gestión de configuración y automatización de cambios.
- **Funciones de automatización de grid incorporadas**: significa que la tecnología de la información responde proactivamente a la demanda fluctuante e implanta nuevos servicios más rápidamente, de modo que los negocios puedan prosperar.
- **Diagnóstico detallado y soluciones disponibles de inmediato**: se puede aplicar en una amplia gama de aplicaciones, incluidas aplicaciones personalizadas, Oracle E-Business Suite, PeopleSoft, Siebel, Oracle Fusion Middleware, Oracle Database y la infraestructura subyacente.
- **Amplias capacidades de gestión de ciclo de vida**: amplía grid computing al proporcionar soluciones para el ciclo de vida de toda la aplicación e infraestructura, incluyendo prueba, etapa y producción mediante las operaciones. Ha simplificado la gestión de parches con la aplicación de parches sincronizada, el soporte del sistema operativo adicional y funciones de detección de conflictos.

Oracle BI Publisher

- Proporciona una arquitectura central para autorizar, gestionar y proporcionar información de forma segura y en varios formatos
- Reduce la complejidad y tiempo en el desarrollo, prueba y despliegue de todos los tipos de informes
  - Informes financieros,facturas, pedidos de ventas o compras, XML y EDI/EFT(documentos eText)
- Permite personalizaciones flexibles
  - Por ejemplo, un informe de un documento de Microsoft Word se puede generar en varios formatos, como PDF, HTML, Excel, RTF, etc.

Oracle BI Publisher

Oracle Database 11*g* también incluye Oracle BI Publisher, solución de creación de informes empresariales de Oracle. Oracle BI Publisher (antes denominado XML Publisher) proporciona la solución de creación de informes escalable y más eficaz disponible para entornos complejos y distribuidos.

Oracle BI Publisher reduce los elevados costos asociados al desarrollo, personalización y mantenimiento de los documentos de negocio, al mismo tiempo que aumenta la eficacia de la gestión de informes. Mediante un juego de herramientas de escritorio conocidas, los usuarios pueden crear y mantener sus propios formatos de informe según las consultas de datos creadas por los desarrolladores o personal de TI.

Los formatos de informe de Oracle BI Publisher se pueden diseñar mediante Microsoft Word o Adobe Acrobat, herramientas con las que la mayoría de los usuarios están familiarizados. Oracle BI Publisher también permite mostrar los datos desde diferentes orígenes de datos en un único documento de salida. Puede proporcionar informes mediantes impresora, correo electrónico o fax. Puede publicar el informe en un portal. Incluso puede permitir a los usuarios que colaboren en ediciones y gestión de informes en servidores web Web-based Distributed Authoring and Versioning (WebDav).

Agenda

- Objetivos del curso, agenda y apéndices utilizados en este curso
- Visión general de Oracle Database 11*g* y productos relacionados
- Visión general de los conceptos y terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- Esquema `HR` y tablas utilizadas en este curso
- Documentación y recursos adicionales de Oracle Database 11*g*

ORACLE

Sistema de Gestión de Bases de Datos Relacionales y Relacionales de Objetos

- Modelo relacional y modelo relacional de objeto
- Objetos y tipos de dato definidos por el usuario
- Compatibilidad total con la base de datos relacional
- Soporta objetos grandes y multimedia
- Funciones del servidor de bases de datos de alta calidad

ORACLE

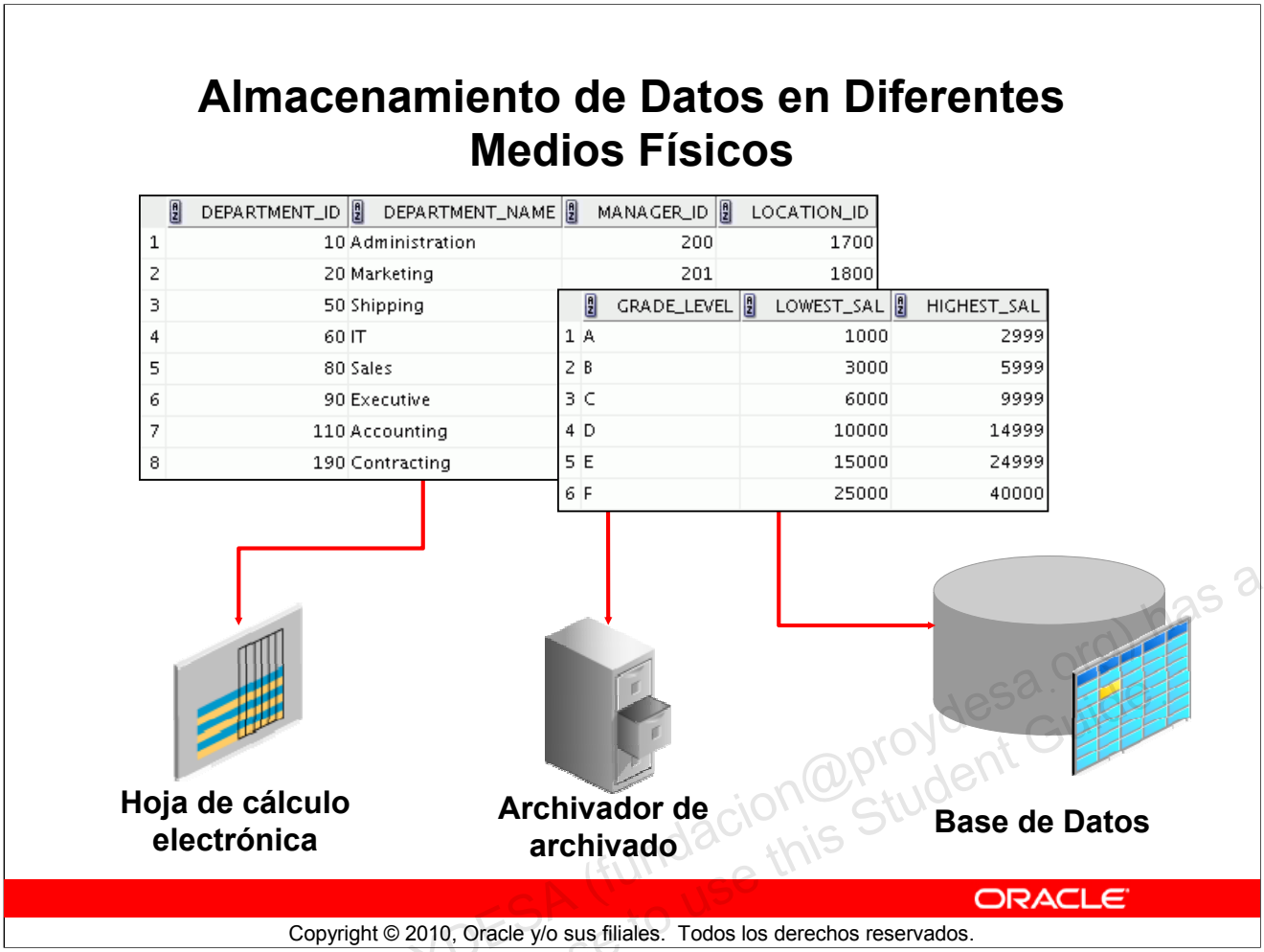
Sistema de Gestión de Bases de Datos Relacionales y Relacionales de Objetos

El servidor de Oracle soporta el modelo relacional y de base de datos relacional de objeto.

El servidor de Oracle amplía las capacidades de modelado de datos para soportar un modelo de base de datos relacional de objeto que proporciona programación orientada a objetos, tipos de dato complejos, objetos de negocio complejos y compatibilidad completa con el mundo relacional.

Incluye distintas funciones para una funcionalidad y rendimiento mejorado de las aplicaciones OLTP, como un uso compartido mejorado de las estructuras de datos de tiempo de ejecución, cachés de buffers grandes y restricciones diferibles. Las aplicaciones de almacén de datos aprovechan las mejoras, como la ejecución en paralelo de operaciones de inserción, la actualización y la supresión, además de la partición y optimización de consultas de detección en paralelo. El modelo de Oracle soporta aplicaciones basadas en web y de cliente/servidor distribuidas y de varias capas.

Para obtener más información sobre el modelo relacional de objeto o relacional, consulte *Oracle Database Concepts for 10g or 11g database* (Conceptos de Oracle Database para Base de Datos 10g u 11g).



Almacenamiento de Datos en Diferentes Medios Físicos

Cada organización tiene sus necesidades de información. Una biblioteca conserva una lista de los miembros, libros, fechas de vencimiento y sanciones. Una compañía necesita guardar la información sobre los empleados, departamentos y salarios. Esta información se denomina *datos*.

Las organizaciones pueden almacenar datos en diferentes medios físicos y en diferentes formatos, como un documento impreso en un archivador de archivado o datos almacenados en hojas de cálculo electrónicas o en bases de datos.

Una *base de datos* es una recopilación organizada de información.

Para gestionar bases de datos, necesita un sistema de gestión de bases de datos (DBMS). Un DBMS es un programa que almacena, recupera y modifica los datos en las bases de datos a petición. Hay cuatro tipos principales de bases de datos: *jerárquica*, *de red*, *relacional* y (la más reciente) *relacional de objeto*.

Concepto de Base de Datos Relacional

- El Dr. E. F. Codd propuso el modelo relacional del sistema de bases de datos en 1970.
- Es la base del sistema de gestión de bases de datos relacionales (RDBMS).
- El modelo relacional consta de lo siguiente:
  - Recopilación de objetos o relaciones
  - Juego de operadores que actúan en las relaciones
  - Integridad de datos para su precisión y consistencia

ORACLE

Concepto de Base de Datos Relacional

El Dr. E. F. Codd fue pionero al describir los principios del modelo relacional en junio de 1970, en una documentación técnica titulada *A Relational Model of Data for Large Shared Data Banks*. En esta documentación técnica, el Dr. Codd propuso el modelo relacional para los sistemas de bases de datos.

Los modelos comunes utilizados hasta ese momento eran jerárquicos y de red o incluso simples estructuras de datos de archivo plano. Los sistemas de gestión de bases de datos relacionales (RDBMS) pronto se volvieron muy conocidos, en especial por su facilidad de uso y su flexible estructura. Además, un número de proveedores innovadores, como Oracle, complementaron los RDBMS con una serie de productos de usuario y desarrollo de aplicaciones potentes, proporcionando una solución total.

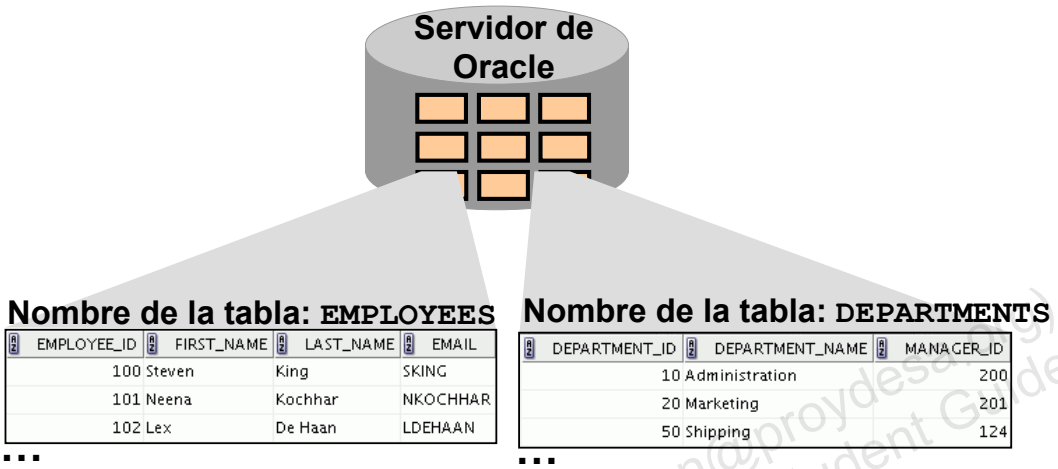
Componentes del Modelo Relacional

- Recopilaciones de objetos o relaciones que almacenan los datos.
- Un juego de operadores que pueden actuar en las relaciones para producir otras relaciones.
- Integridad de datos para su precisión y consistencia.

Para obtener más información, consulte *An Introduction to Database Systems, Eighth Edition* (Addison-Wesley: 2004), escrito por Chris Date.

Definición de una Base de Datos Relacional

Una base de datos relacional es una recopilación de relaciones o tablas bidimensionales.

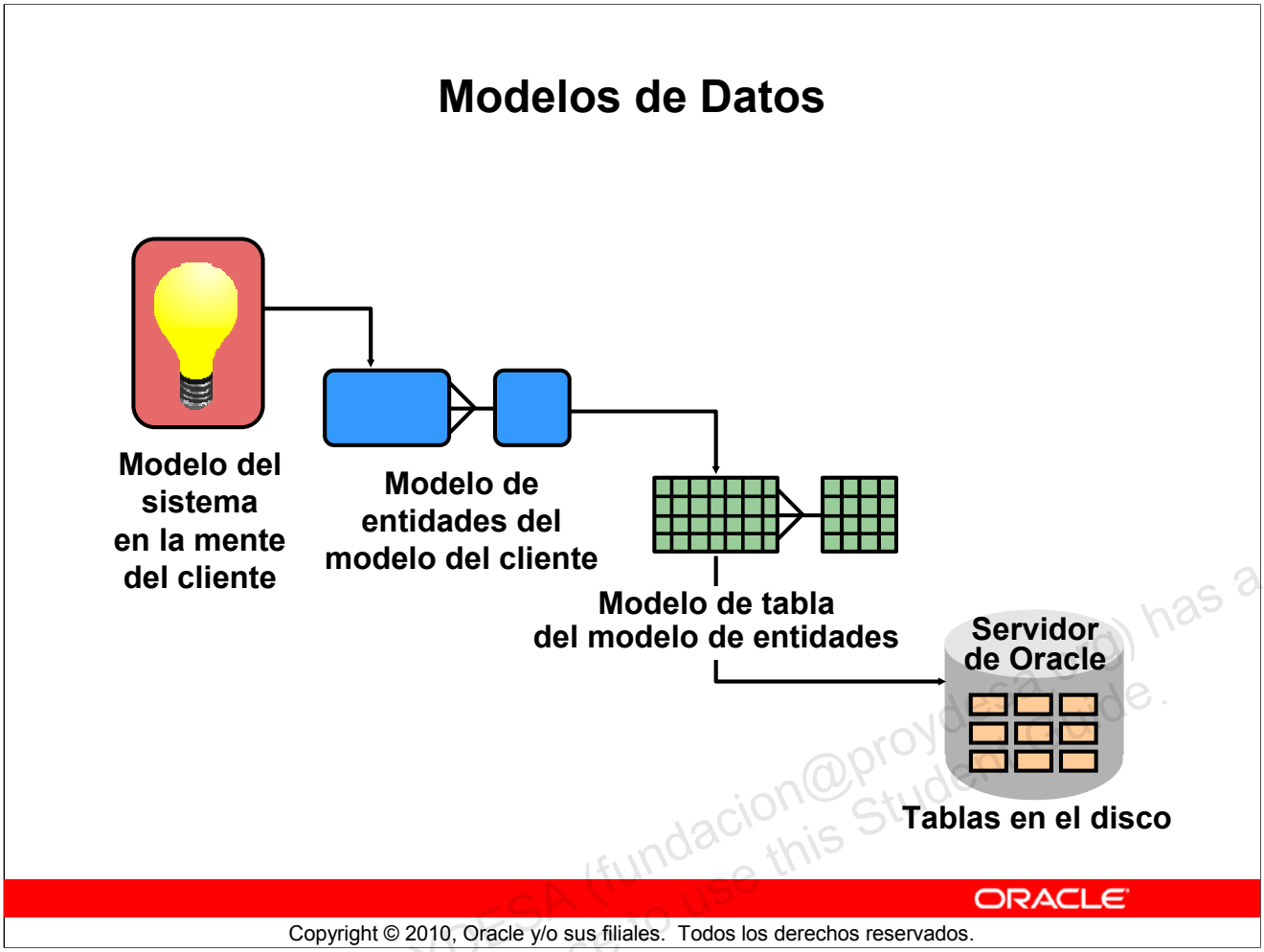


ORACLE

Definición de una Base de Datos Relacional

Una base de datos relacional utiliza relaciones o tablas bidimensionales para almacenar información.

Por ejemplo, puede que desee almacenar información sobre todos los empleados de la compañía. En una base de datos relacional, puede crear diferentes tablas para almacenar información diferente sobre los empleados, como una tabla de empleados, una de departamentos y una de salarios.



Modelos de Datos

Los modelos son la base del diseño. Los ingenieros crean un modelo de coche para solucionar cualquier problema antes de iniciar la producción. De la misma manera, los diseñadores desarrollan modelos para explorar ideas y mejorar el conocimiento del diseño de la base de datos.

Objetivo de los Modelos

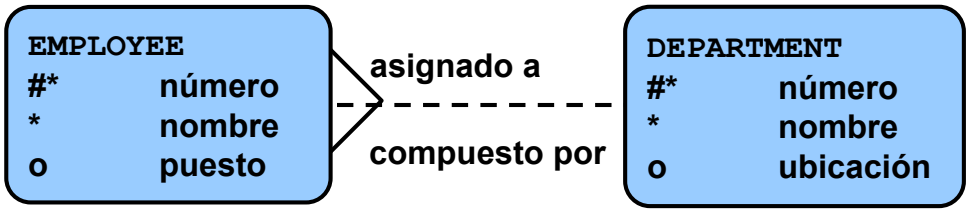
Los modelos ayudan a comunicar los conceptos que las personas tienen en mente. Se puede utilizar para realizar las siguientes acciones:

- Comunicar
- Clasificar
- Describir
- Especificar
- Investigar
- Evolucionar
- Analizar
- Imitar

El objetivo es producir un modelo que se ajuste a varios de estos usos, que el usuario lo pueda entender y que contenga los suficientes detalles para que un desarrollador cree un sistema de bases de datos.

Modelo de Relación de Entidades

- Crear un diagrama de entidad/relación a partir de narrativas o especificaciones de negocio:



- Supuesto:
  - ". . . Asignar uno o más empleados a un departamento . . ."
  - ". . . Algunos departamentos aún no tienen empleados asignados. . ."

Modelo de Relación de Entidades

En un sistema eficaz, los datos se dividen en entidades o categorías discretas. Un modelo de relación de entidades (ER) es una ilustración de varias entidades de un negocio y las relaciones entre ellas. Un modelo de ER se deriva de las especificaciones o descripciones de negocio y se crea durante la fase de análisis del ciclo de vida de desarrollo del sistema. Los modelos de ER separan la información que necesita un negocio a partir de las actividades realizadas en el mismo. Aunque los negocios pueden cambiar sus actividades, el tipo de información suele permanecer constante. Por lo tanto, las estructuras de datos también tienden a ser constantes.

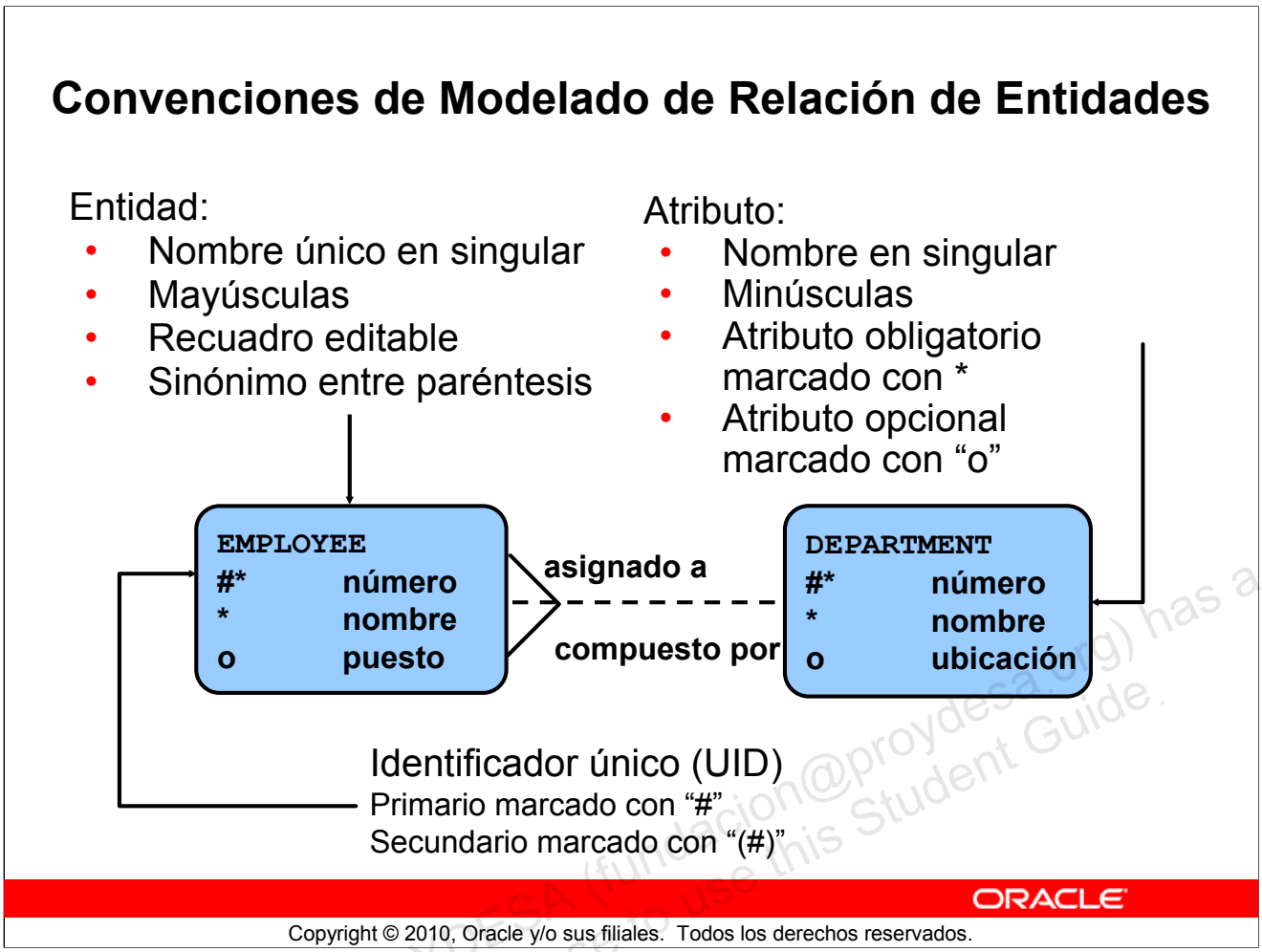
Modelo de Relación de Entidades (continuación)

Ventajas del Modelado de ER:

- Recopila información para la organización en un formato claro y preciso.
- Proporciona una imagen clara del ámbito de los requisitos de la información.
- Proporciona una asignación gráfica de fácil compresión para el diseño de la base de datos.
- Ofrece un marco efectivo para integrar varias aplicaciones.

Componentes Clave

- **Entidad:** aspecto significativo sobre qué información se debe conocer. Algunos ejemplos pueden ser los departamentos, empleados y pedidos.
- **Atributo:** algo que describe o cualifica una entidad. Por ejemplo, para la entidad de empleados, los atributos serían el número de empleado, el nombre, cargo, fecha de contratación, número de departamento, etc. Cada uno de los atributos puede ser necesario u opcional. Este estado se denomina *opciones*.
- **Relación:** asociación con nombre entre las entidades que muestran opciones y grados. Algunos ejemplos pueden ser los empleados y departamentos, pedidos y elementos.



Convenciones de Modelado de ER

Entidades

Para representar una entidad, utilice las siguientes convenciones:

- Nombre de entidad único en singular
- Nombre de entidad en mayúscula
- Recuadro editable
- Nombres de sinónimos opcionales en mayúsculas entre paréntesis: ( )

Atributos

Para representar un atributo en un modelo utilice las siguientes convenciones:

- Nombre en singular y en minúscula
- Etiqueta de asterisco (*) para los atributos obligatorios (es decir, valores que se *deben* conocer)
- Etiqueta de letra "o" para los atributos opcionales (es decir, los valores que se *pueden* conocer)

Relaciones

Símbolo	Descripción
Línea discontinua	Elemento opcional que indica "quizás".
Línea continua	Elemento obligatorio que indica "debe ser".
Pata de gallo	Elemento de grado que indica "uno o más".
Una sola línea	Elemento de grado que indica "único".

Convenciones de Modelado de ER (continuación)

Relaciones

Cada dirección de la relación contiene:

- **Una etiqueta:** por ejemplo, *enseñado por* o *asignado a*
- **Una opción:** *debe ser* o *quizás*
- **Un grado:** *único* o *uno o más*

Nota: el término *cardinalidad* es un sinónimo del término *grado*.

Cada entidad de origen {puede ser | debe ser} nombre de relación {único | uno o más} entidad de destino.

Nota: la convención se lee hacia la izquierda.

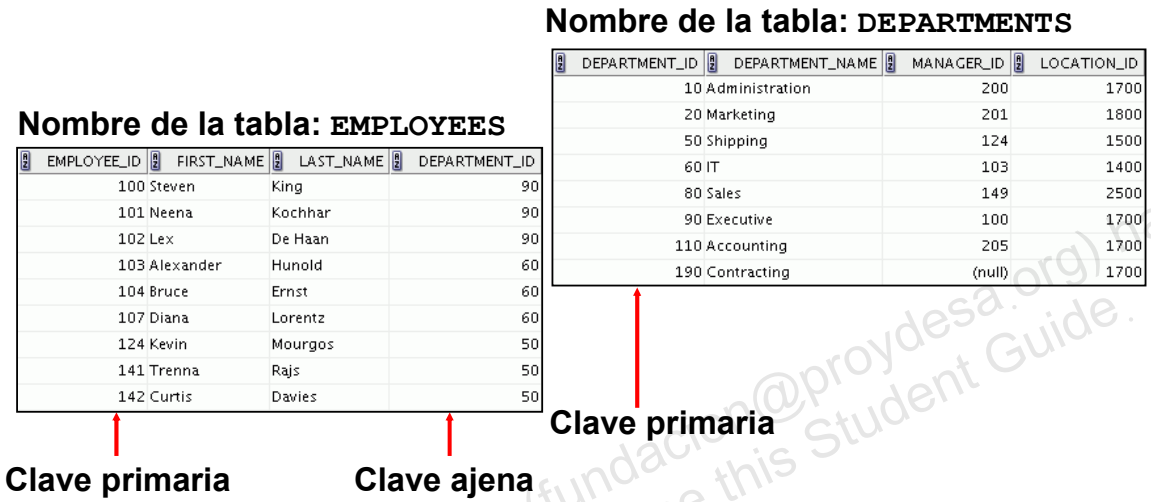
Identificadores Únicos

Un identificador único (UID) es cualquier combinación de atributos o relaciones, o de ambos, que sirve para distinguir las incidencias de una entidad. Cada incidencia de entidad se debe identificar de forma única.

- Etiquetar cada atributo que forma parte del UID con un signo de almohadilla "#".
- Etiquetar los UID secundarios con un signo de almohadilla entre paréntesis (#).

Relación de Varias Tablas

- Cada fila de datos de una tabla se identifica como única mediante una clave primaria.
- Puede relacionar de forma lógica desde varias tablas mediante claves ajenas.



Relación de Varias Tablas

Cada tabla contiene datos que describen exactamente una entidad. Por ejemplo, la tabla `EMPLOYEES` contiene información sobre los empleados. Las categorías de datos se muestran en la parte superior de cada tabla y los casos individuales se muestran a continuación. Si utiliza un formato de tabla, puede visualizar, comprender y utilizar la información de inmediato.

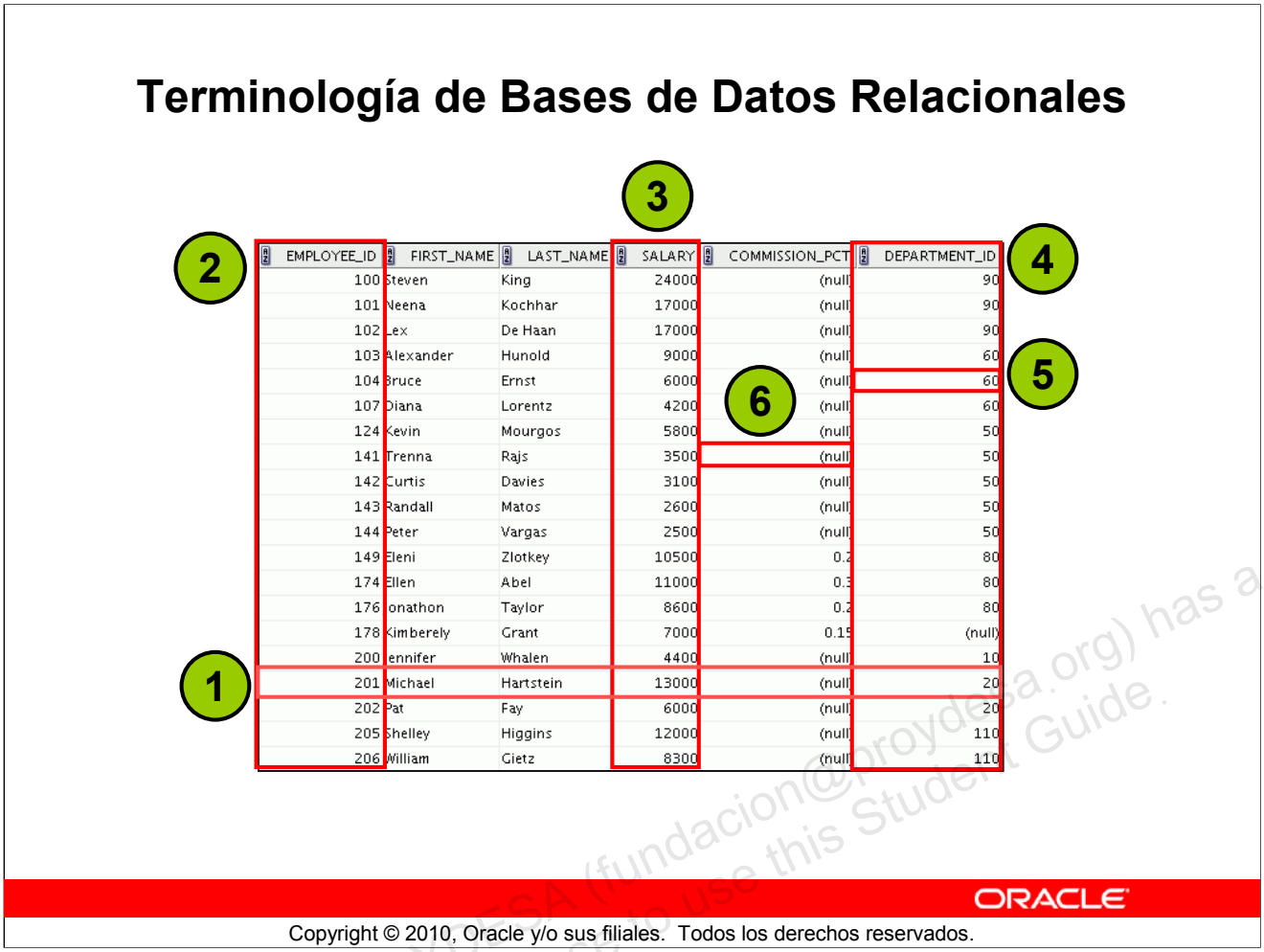
Debido a que los datos sobre las diferentes entidades se almacenan en diferentes tablas, puede que tenga que combinar dos o más tablas para responder a una pregunta determinada. Por ejemplo, puede que desee conocer la ubicación del departamento en el que trabaja un empleado. En este supuesto, necesita información de la tabla `EMPLOYEES` (que contiene datos sobre los empleados) y la tabla `DEPARTMENTS` (que contiene información sobre los departamentos). Con RDBMS, puede relacionar los datos de una tabla con los datos de otra utilizando las claves ajenas. Una clave ajena es una columna (o juego de columnas) que hace referencia a una clave primaria en la misma tabla o en otra tabla.

Tiene la posibilidad de relacionar datos de una tabla con datos de otra tabla para organizar la información en unidades separadas y gestionables. Los datos de empleado se pueden mantener, de forma lógica, separados de los datos de departamento almacenándolos en una tabla independiente.

Relación de Varias Tablas (continuación)

Instrucciones para Claves Primarias y Ajenas

- No puede utilizar valores duplicados en una clave primaria.
- Generalmente, las claves primarias no se pueden cambiar.
- Las claves ajenas se basan en valores de datos y son punteros puramente lógicos (no físicos).
- Un valor de clave ajena debe coincidir con un valor de clave primaria o de clave única existente o, de lo contrario, debe ser nulo.
- Una clave ajena debe hacer referencia a una columna de clave primaria o única.



Terminología de Bases de Datos Relacionales

Una base de datos relacional puede contener una o más tablas. Una *tabla* es la estructura básica de almacenamiento de RDBMS. Una tabla contiene todos los datos necesarios sobre algo del mundo real, como empleados, facturas o clientes.

La diapositiva muestra el contenido de la tabla o relación `EMPLOYEES`. Los números indican lo siguiente

1. Una única *fila* (o *tupla*), que representa todos los datos necesarios para un empleado concreto. Cada fila de una tabla se debe identificar por una clave primaria, que no permite duplicar filas. El orden de las filas no es importante; especifique el orden de la fila cuando se recuperen los datos.
2. Una *columna* o atributo, que contiene el número de empleado. El número de empleado identifica un *único* empleado en la tabla `EMPLOYEES`. En este ejemplo, la columna del número de empleado está designada como *clave primaria*. Una clave primaria debe contener un valor y el valor debe ser único.
3. Una columna que no es un valor clave. Una columna representa un tipo de dato en una tabla; en este ejemplo, los datos son los salarios de todos los empleados. El orden de las columnas no es importante al almacenar datos; especifique el orden de la columna al recuperar los datos.

Terminología de Bases de Datos Relacionales (continuación)

4. Una columna que contiene el número de departamento, que también es una *clave ajena*. Una clave ajena es una columna que define cómo se relacionan las tablas sí. Una clave ajena hace referencia a una clave primaria o única en la misma tabla o en otra tabla. En el ejemplo, `DEPARTMENT_ID` identifica de forma única un departamento en la tabla `DEPARTMENTS`.
5. Un *campo* está en la intersección de una fila y una columna. Sólo puede contener un valor.
6. Un campo puede no contener ningún valor. Esto se denomina valor nulo. En la tabla `EMPLOYEES` solo aquellos empleados que tienen el rol de vendedor tienen un valor en el campo `COMMISSION_PCT` (comisiones).

Agenda

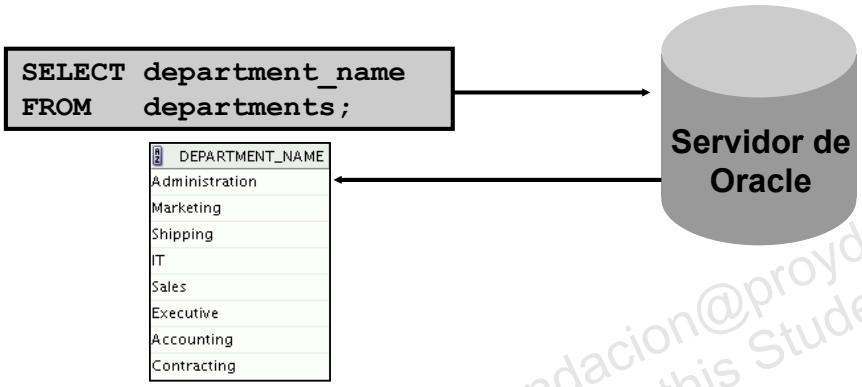
- Objetivos del curso, agenda y apéndices utilizados en este curso
- Visión general de Oracle Database 11*g* y productos relacionados
- Visión general de los conceptos y terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- Esquema `HR` y tablas utilizadas en este curso
- Documentación y recursos adicionales de Oracle Database 11*g*

ORACLE

Uso de SQL para Consultar Base de Datos

El lenguaje de consulta estructurado (SQL) es:

- Lenguaje estándar de ANSI para el funcionamiento de bases de datos relacionales
- Uso y aprendizaje sencillos y eficaces
- Funcionalidad completa (con SQL, puede definir, recuperar y manipular datos en las tablas)



ORACLE

Uso de SQL para Consultar Base de Datos

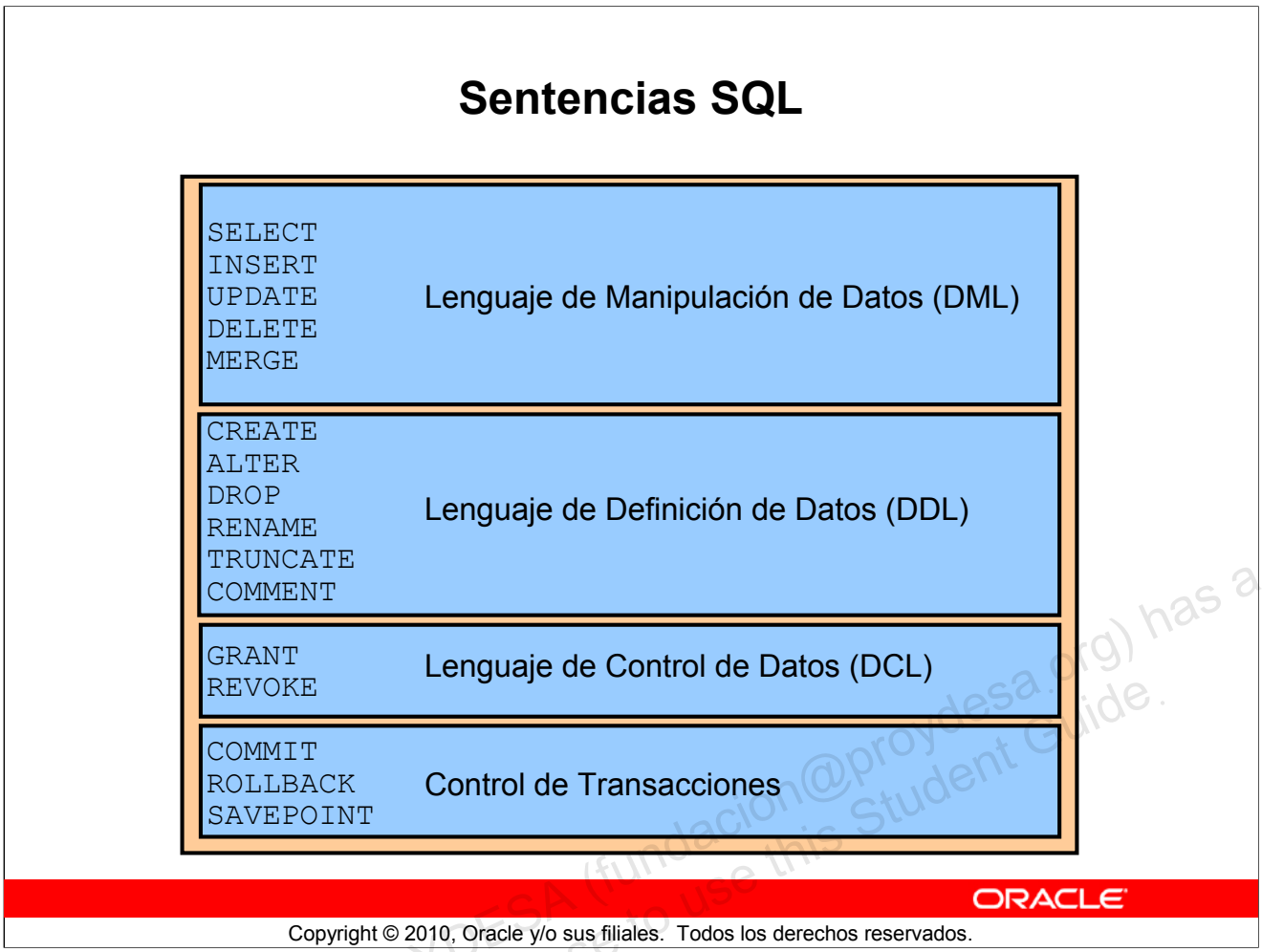
En una base de datos relacional, no es necesario especificar la ruta de acceso a las tablas y tampoco es necesario saber cómo se organizan los datos de forma física.

Para acceder a la base de datos, ejecute una sentencia de lenguaje de consulta estructurado (SQL), que es el lenguaje estándar de ANSI (American National Standards Institute) para el funcionamiento de bases de datos relacionales. SQL es un juego de sentencias con el que todos los programas y usuarios acceden a los datos de Oracle Database. Los programas y las herramientas de Oracle a menudo permiten el acceso de usuarios a la base de datos sin utilizar directamente SQL, pero estas aplicaciones a su vez deben utilizar SQL al ejecutar la solicitud del usuario.

SQL proporciona sentencias para distintas tareas, que incluyen las siguientes:

- Consulta de datos
- Inserción, actualización y supresión de filas en una tabla
- Creación, sustitución, modificación y borrado de objetos
- Control de acceso a la base de datos y los objetos
- Garantía de integridad y consistencia de la base de datos

SQL unifica todas las tareas anteriores en un lenguaje consistente y permite trabajar con datos en el nivel lógico.



Sentencias SQL

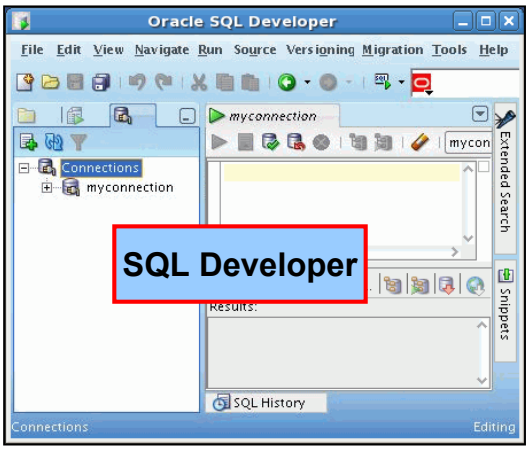
Sentencias SQL soportadas por el cumplimento de Oracle de los estándares de la industria Oracle Corporation asegura el cumplimiento futuro con la evolución de los estándares mediante la implicación de forma activa del personal clave en los comités de estándares SQL. Los comités aceptados por la industria son ANSI e ISO (International Standards Organization). Tanto ANSI como ISO han aceptado SQL como el lenguaje estándar de las bases de datos relacionales.

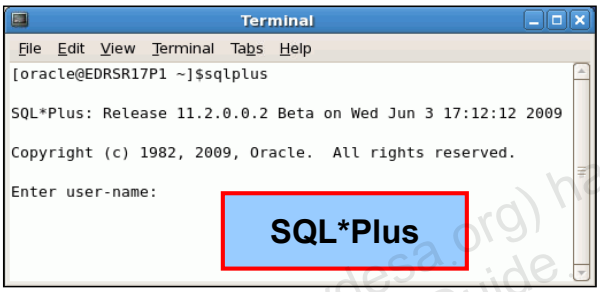
Sentencia	Descripción
SELECT INSERT UPDATE DELETE MERGE	Recupera datos de la base de datos, introduce nuevas filas, cambia las existentes y elimina las filas no deseadas de las tablas en la base de datos, respectivamente. Conocidos colectivamente *lenguaje de manipulación de datos* (DML).
CREATE ALTER DROP RENAME TRUNCATE COMMENT	Configura, cambia y elimina las estructuras de datos de las tablas. Conocidos colectivamente como *lenguaje de definición de datos* (DDL).
GRANT REVOKE	Proporciona o elimina los derechos de acceso a Oracle Database y a las. estructuras que contiene.
COMMIT ROLLBACK SAVEPOINT	Gestiona los cambios realizados por las sentencias DML. Los cambios en los datos se pueden agrupar en transacciones lógicas.

Entornos de Desarrollo para SQL

Existen dos entornos de desarrollo para este curso:

- La herramienta principal es Oracle SQL Developer.
- También se puede utilizar la interfaz de línea de comandos SQL*Plus.





Entornos de Desarrollo para SQL

SQL Developer

Este curso se desarrolla mediante Oracle SQL Developer como la herramienta para la ejecución de sentencias tratadas en los ejemplos de las lecciones y prácticas. SQL Developer versión 1.5.4 se incluye con Oracle Database 11*g* y es la herramienta por defecto para esta clase.

SQL*Plus

También se puede utilizar el entorno de SQL*Plus para ejecutar todos los comandos SQL tratados en este curso.

Nota

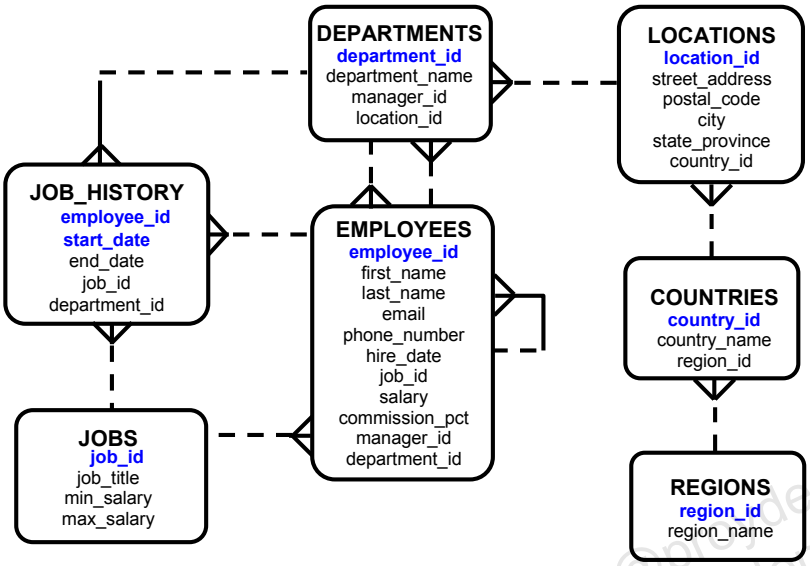
- Consulte el apéndice C para obtener información sobre el uso de SQL Developer, que incluye instrucciones simples sobre la instalación de la versión 1.5.4.
- Consulte el apéndice D para obtener más información sobre el uso de SQL*Plus.

Agenda

- Objetivos del curso, agenda y apéndices utilizados en este curso
- Visión general de Oracle Database 11*g* y productos relacionados
- Visión general de los conceptos y terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- Esquema `HR` y tablas utilizadas en este curso
- Documentación y recursos adicionales de Oracle Database 11*g*

ORACLE

Esquema Human Resources (HR)



Descripción del esquema Human Resources (HR)

El esquema Human Resources (HR) forma parte de los esquemas de ejemplo de Oracle que se pueden instalar en Oracle Database. Las sesiones prácticas de este curso utilizan datos del esquema HR.

Descripciones de las Tablas

- REGIONS contiene filas que representan una región, como América, Asia, etc.
- COUNTRIES contiene filas para países, que están asociados a una región.
- LOCATIONS contiene la dirección concreta de una oficina, almacén o fábrica de una compañía en un país determinado.
- DEPARTMENTS muestra detalles de los departamentos en los que trabajan los empleados. Cada departamento puede tener una relación que represente al gestor del departamento en la tabla EMPLOYEES.
- EMPLOYEES contiene detalles sobre cada empleado que trabaja en un departamento. Puede que algunos empleados no estén asignados a ningún departamento.
- JOBS contiene los tipos de cargos que puede tener cada empleado.
- JOB_HISTORY contiene el historial del trabajo de los empleados. Si un empleado cambia de departamento dentro de un mismo cargo o cambia de cargo dentro de un mismo departamento, se insertará una nueva fila en esta tabla con la información del antiguo cargo del empleado.

Tablas Utilizadas en el Curso

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	24000	(null)	90	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	17000	(null)	90	NKOCHHAR	515.123.4568	21-SEP-89
102	Lex	De Haan	17000	(null)	90	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	9000	(null)	60	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	6000	(null)	60	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	4200	(null)	60	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	5800	(null)	50	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	3500	(null)	50	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	3100	(null)	50	CDAVIES	650.121.2994	29-JAN-97
143	Randall	Matos	2600	(null)	50	RMATOS	650.121.2874	15-MAR-98
144	Peter	Vargas	2500	(null)	50	PVARGAS	650.121.2004	09-JUL-98
149	Eleni	Zlotkey	10500	0.2	80	EZLOTKEY	011.44.1344.429018	29-JAN-00
174	Ellen	Abel	11000	0.3	80	EABEL	011.44.1644.429267	11-MAY-96
176	Jonathon	Taylor	8600	0.2	80	JTAYLOR	011.44.1644.429265	24-MAR-98
178	Kimberely	Grant	7000	0.15	(null)	KGRANT	011.44.1644.429263	24-MAY-99
200	Jennifer	Whalen	4400	(null)	10	JWHALEN	515.123.4444	17-SEP-87
201	Michael	Hartstein	13000	(null)	20	MHARTSTE	515.123.5555	17-FEB-96
202	Pat	Fay	6000	(null)	20	PFAY	603.123.6666	17-AUG-97
205	Shelley	Higgins	12000	(null)	110	SHIGGINS	515.123.8080	07-JUN-94
206	William	Gietz	8300	(null)	110	WGIETZ	515.123.8181	07-JUN-94

GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

JOB_GRADES

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	(null)	1700

DEPARTMENTS

ORACLE

Tablas Utilizadas en el Curso

A continuación se muestran las tablas principales utilizadas en este curso:

- Tabla `EMPLOYEES`: proporciona detalles de todos los empleados
- Tabla `DEPARTMENTS`: proporciona detalles de todos los departamentos
- Tabla `JOB_GRADES`: proporciona detalles de los salarios de diferentes grados

Además de estas tablas, también utilizará otras tablas mostradas en la diapositiva anterior, como `LOCATIONS` y `JOB_HISTORY`.

Nota: la estructura y datos de todas las tablas se proporcionan en el Apéndice B.

Agenda

- Objetivos del curso, agenda y apéndices utilizados en este curso
- Visión general de Oracle Database 11*g* y productos relacionados
- Visión general de los conceptos y terminología de gestión de bases de datos relacionales
- Introducción a SQL y sus entornos de desarrollo
- Esquema `HR` y tablas utilizadas en este curso
- Documentación y recursos adicionales de Oracle Database 11*g*

ORACLE

Documentación de Oracle Database

- *Oracle Database New Features Guide*
- *Oracle Database Reference*
- *Oracle Database SQL Language Reference*
- *Oracle Database Concepts*
- *Oracle Database SQL Developer User's Guide, Release 1.5*

Documentación de Oracle Database

Navegue a http://www.oracle.com/pls/db102/homepage para acceder a la biblioteca de documentación de Oracle Database 10*g*.

Navegue a http://www.oracle.com/pls/db112/homepagepara acceder a la biblioteca de documentación de Oracle Database 11*g*.

Recursos Adicionales

Para obtener más información sobre Oracle Database 11*g*, consulte:

- *Oracle Database 11g: New Features eStudies*
- *Serie Oracle by Example (OBE): Oracle Database 11g*
  - http://www.oracle.com/technology/obe/11gr1_db/index.htm

ORACLE

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Oracle Database 11*g* amplía:
  - Las ventajas de grids de infraestructura
  - Las capacidades de gestión de información existente
  - Las funciones para utilizar y gestionar todos los entornos de desarrollo para aplicaciones principales como PL/SQL, Java/JDBC, .NET, XML y etc.
- La base de datos se basa en ORDBMS
- Las bases de datos relacionales se componen de relaciones, gestionadas mediante operaciones relacionales y regidas por restricciones de integridad de datos
- Con el servidor de Oracle, puede almacenar y gestionar información mediante SQL

ORACLE

Resumen

Los sistemas de gestión de bases de datos relacionales se componen de objetos o relaciones. Se gestionan mediante operaciones y se rigen por restricciones de integridad de datos.

Oracle Corporation genera productos y servicios para cumplir las necesidades de RDBMS. Los productos principales son los siguientes:

- Oracle Database, que almacena y gestiona información mediante SQL.
- Oracle Fusion Middleware , que desarrolla, despliega y gestiona servicios de negocio modulares que se pueden integrar y volver a utilizar
- Oracle Enterprise Manager Grid Control, que gestiona y automatizar tareas administrativas entre los juegos de sistemas en un entorno de cuadrícula

SQL

El servidor de Oracle soporta SQL del estándar de ANSI y contiene extensiones. SQL es un lenguaje que se utiliza para comunicarse con el servidor para acceder, manipular y controlar datos.

Práctica I: Visión General

En esta práctica se abordan los siguientes temas:

- Inicio de Oracle SQL Developer
- Creación de una conexión a la base de datos
- Examen de tablas `HR`

ORACLE

Práctica I: Visión General

En esta práctica, realiza las siguientes tareas:

- Iniciar Oracle SQL Developer y crear una nueva conexión a la cuenta `ora1`.
- Utilizar Oracle SQL Developer para examinar objetos de datos en la cuenta ora1. La cuenta `ora1` contiene las tablas de esquema `HR`.

Observe la siguiente ubicación de los archivos de los ejercicios prácticos:

```
\home\oracle\labs\sql1\labs
```

Si se le pide que guarde algún archivo de los ejercicios prácticos, guárdelo en esta ubicación.

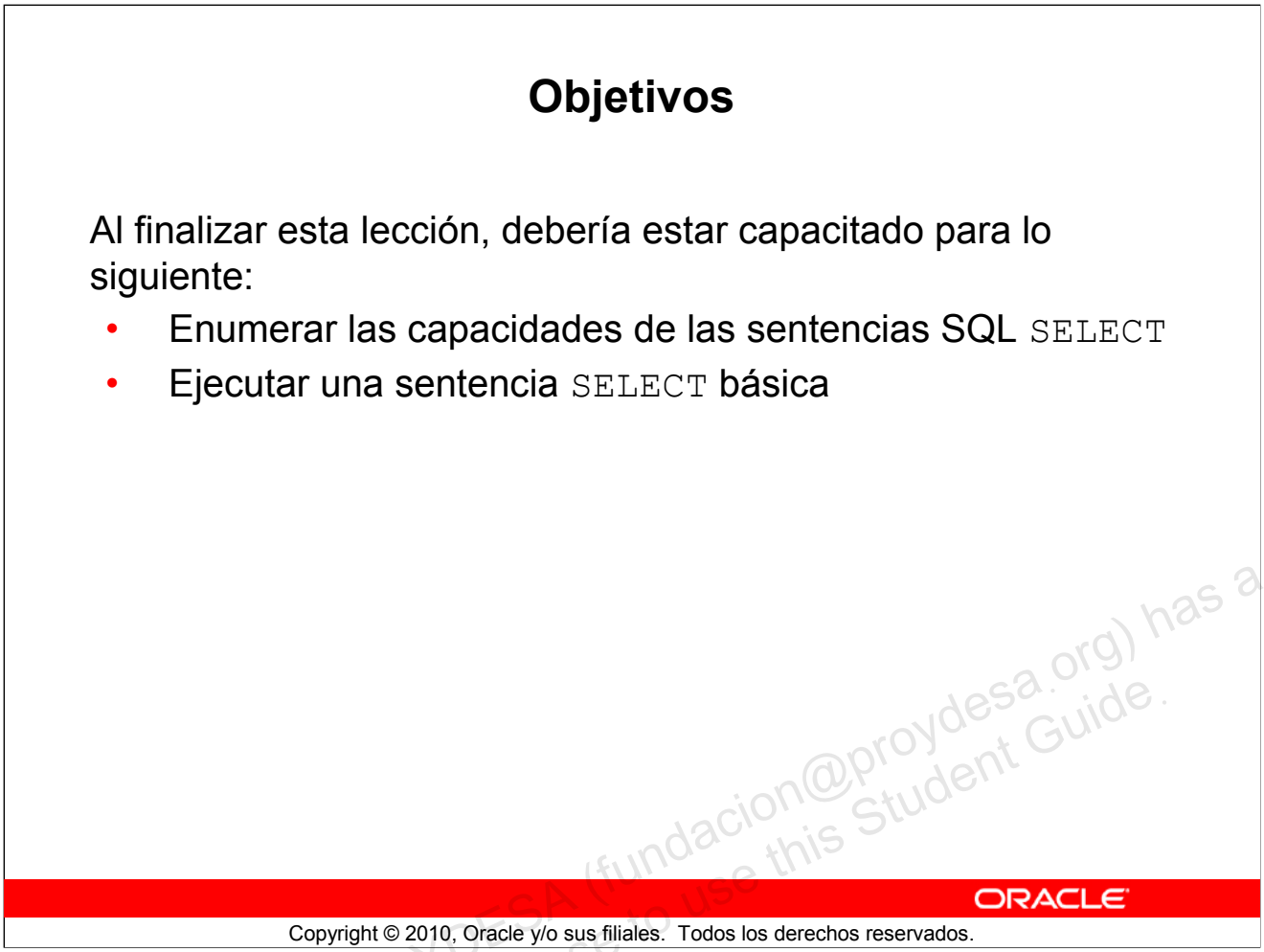
En las prácticas, puede que haya ejercicios que estén precedidos por las frases “Si tiene tiempo” o “Si desea superarse a sí mismo”. Realice estos ejercicios sólo si ha terminado el resto de ejercicios en el tiempo asignado y si desea poner a prueba sus habilidades.

Realice las prácticas despacio y de forma precisa. Puede experimentar guardando y ejecutando los archivos de comandos. Si tiene alguna duda en cualquier momento, pregunte a su instructor.

Nota: todas las prácticas escritas utilizan Oracle SQL Developer como entorno de desarrollo. Aunque se recomienda utilizar Oracle SQL Developer, también puede usar SQL*Plus disponible en este curso.

1

Recuperación de Datos Mediante la Sentencia SQL `SELECT`

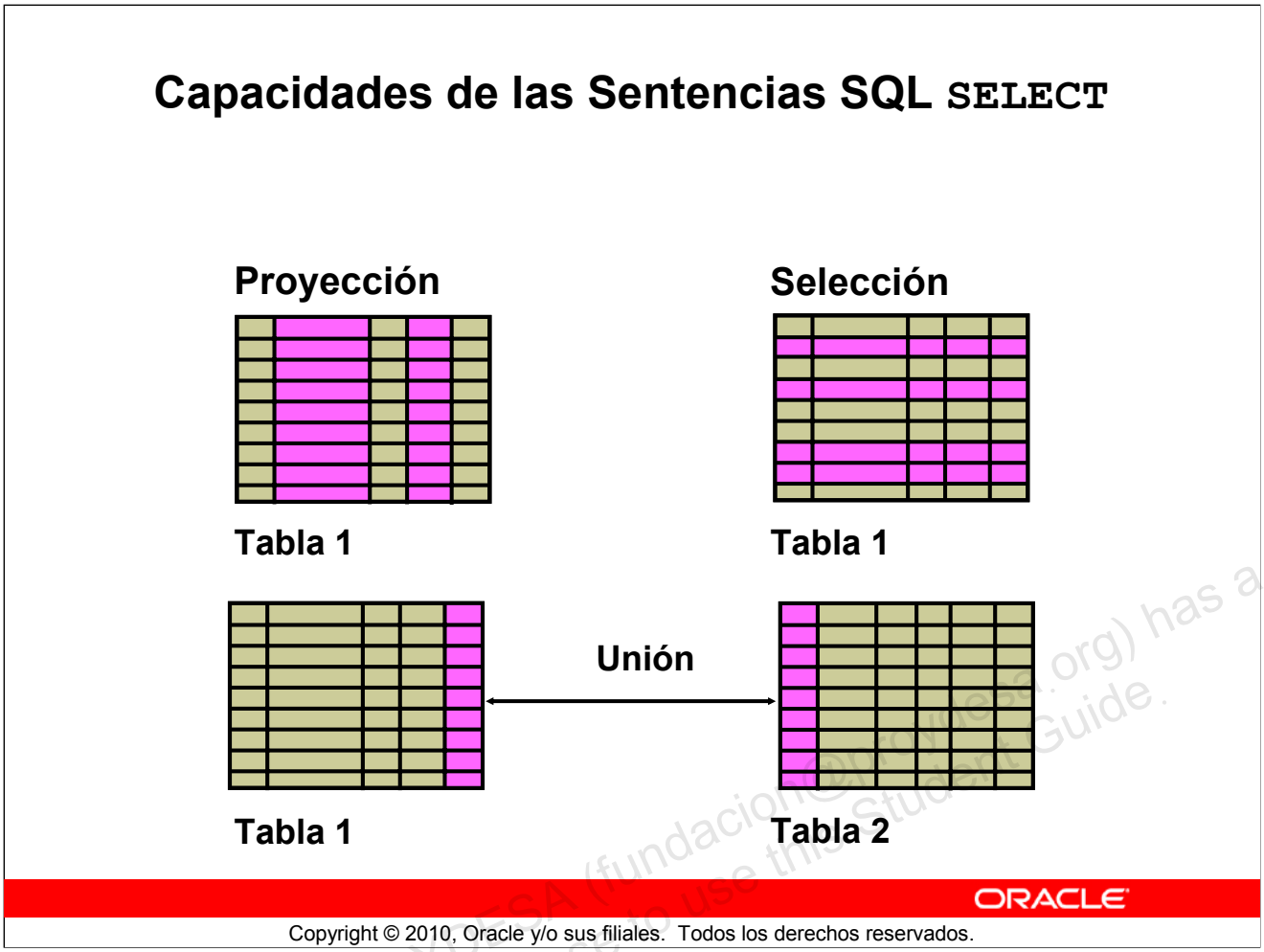


Objetivos

Para extraer datos de la base de datos, debe utilizar la sentencia SQL `SELECT`. Sin embargo, puede que necesite restringir las columnas que se muestran. Esta lección describe todas las sentencias `SELECT` necesarias para realizar estas acciones. Puede que desee crear sentencias `SELECT` que se pueden utilizar más de una vez.

Agenda

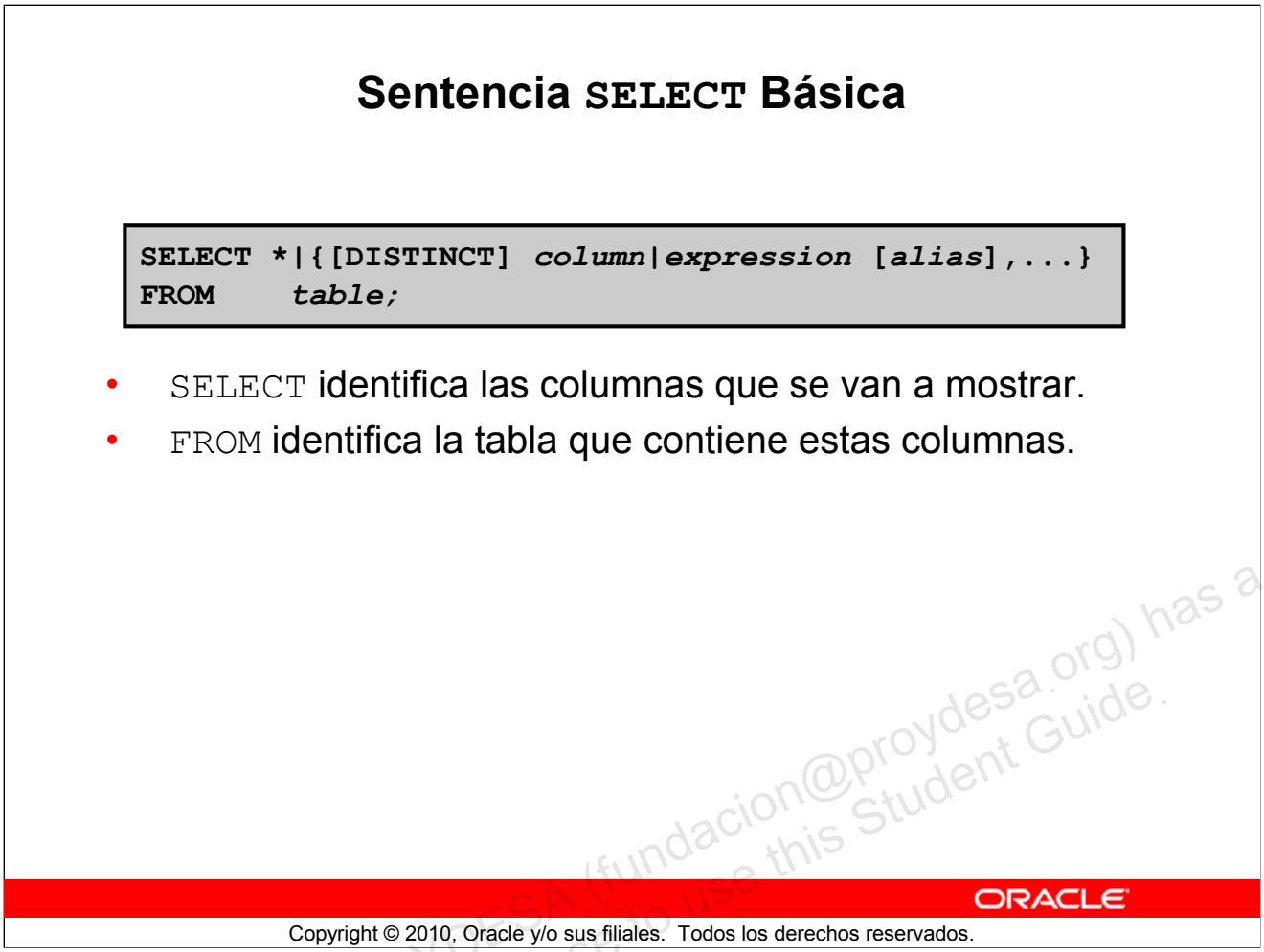
- Sentencia `SELECT` básica
- Expresiones aritméticas y valores `NULL` en la sentencia `SELECT`
- Alias de columnas
- Uso del operador de concatenación, cadenas de caracteres de literales, operador de comillas alternativo y palabra clave `DISTINCT`
- Comando `DESCRIBE`



Capacidades de las Sentencias SQL SELECT

Una sentencia SELECT recupera información de la base de datos. Con una sentencia SELECT, se puede hacer lo siguiente:

- **Proyección:** seleccione las columnas de una tabla devueltas por una consulta. Seleccione tantas columnas como sea necesario.
- **Selección:** seleccione las filas de una tabla devueltas por una consulta. Se pueden utilizar diferentes criterios para restringir las filas recuperadas.
- **Uniones:** reúna los datos almacenados en diferentes tablas especificando el enlace entre ellas. Las uniones SQL se tratan de forma más detallada en la lección titulada "Visualización de Datos de Varias Tablas mediante Uniones".



Sentencia SELECT Básica

En su formato más simple, una sentencia SELECT debe incluir lo siguiente:

- Una cláusula SELECT, que especifica las columnas que se van a mostrar.
- Una cláusula FROM, que identifica la tabla que contiene las columnas que se muestran en la cláusula SELECT.

En la sintaxis:

SELECT	es una lista de una o más columnas.
*	selecciona todas las columnas.
DISTINCT	suprime los duplicados.
column|expression	selecciona la columna o expresión especificada.
alias	proporciona diferentes cabeceras de las columnas seleccionadas.
FROM *table*	especifica la tabla que contiene las columnas.

Nota: a lo largo de este curso, las palabras *palabra clave*, *cláusula* y *sentencia* se utilizan como se describe a continuación:

- Una *palabra clave* hace referencia a un elemento SQL individual (por ejemplo, SELECT y FROM son palabras clave).
- Una *cláusula* es parte de una sentencia SQL (por ejemplo, SELECT employee_id, last_name, etc.).
- Una *sentencia* es una combinación de dos o más cláusulas (por ejemplo, SELECT * FROM employees).

Selección de Todas las Columnas

```
SELECT *
FROM   departments;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

ORACLE

Selección de Todas las Columnas

Puede mostrar todas las columnas de datos en una tabla insertando la siguiente palabra clave `SELECT` con un asterisco (`*`). En el ejemplo de la diapositiva, la tabla `DEPARTMENTS` contiene cuatro columnas: `DEPARTMENT_ID`, `DEPARTMENT_NAME`, `MANAGER_ID` y `LOCATION_ID`. La tabla contiene ocho filas, una por cada departamento.

También puede mostrar todas las columnas en la tabla mostrando todas las columnas después de la palabra clave `SELECT`. Por ejemplo, la siguiente sentencia SQL (como el ejemplo de la diapositiva) muestra todas las columnas y filas de la tabla `DEPARTMENTS`:

```
SELECT  department_id, department_name, manager_id, location_id
  FROM    departments;
```

Nota: en SQL Developer, puede introducir la sentencia SQL en una hoja de trabajo de SQL y hacer clic en el icono “Execute Statement” o pulsar [F9] para ejecutar la sentencia. La salida mostrada en la página con separadores Results aparece como se muestra en la diapositiva.

Selección de Columnas Concretas

```
SELECT department_id, location_id
FROM   departments;
```

	DEPARTMENT_ID	LOCATION_ID
1	10	1700
2	20	1800
3	50	1500
4	60	1400
5	80	2500
6	90	1700
7	110	1700
8	190	1700

ORACLE

Selección de Columnas Concretas

También puede utilizar la sentencia `SELECT` para mostrar las columnas concretas de la tabla especificando los nombres de columna, separados por comas. El ejemplo de la diapositiva muestra todos los números de departamento y ubicación de la tabla `DEPARTMENTS`.

En la cláusula `SELECT`, especifique las columnas que desee, en el orden en el que desee que aparezcan en la salida. Por ejemplo, para mostrar la ubicación antes del número de departamento (de izquierda a derecha), utilice la siguiente sentencia:

```
SELECT location_id, department_id
FROM   departments;
```

	LOCATION_ID	DEPARTMENT_ID
1	1700	10
2	1800	20
3	1500	50
4	1400	60

...

Escritura de Sentencias SQL

- Las sentencias SQL no son sensibles a mayúsculas/ minúsculas.
- Las sentencias SQL se pueden introducir en una o más líneas.
- Las palabras clave no se pueden abreviar o dividir entre líneas.
- Las cláusulas se suelen colocar en líneas independientes.
- El sangrado se utiliza para mejorar la legibilidad.
- En SQL Developer, las sentencias SQL también pueden terminar con un punto y coma (;). Los puntos y comas son necesarios si ejecuta varias sentencias SQL.
- En SQL*Plus, debe finalizar cada sentencia SQL con un punto y coma (;).

ORACLE

Escritura de Sentencias SQL

Con estas sencillas reglas y directrices, puede construir sentencias válidas que son fáciles de leer y editar:

- Las sentencias SQL no son sensibles a mayúsculas/minúsculas (a menos que se indique).
- Las sentencias SQL se pueden introducir en una o más líneas.
- Las palabras clave no se pueden dividir entre líneas o abreviar.
- Las cláusulas se suelen colocar en líneas independientes para que resulte más fácil su lectura o edición.
- El sangrado se debe utilizar para que sea más fácil de leer el código.
- Normalmente, las palabras clave se introducen en mayúsculas; el resto de palabras, como nombres de tablas y columnas, se introducen en minúsculas.

Ejecución de Sentencias SQL

En SQL Developer, haga clic en el icono Run Script o pulse [F5] para ejecutar el comando o los comandos en la hoja de trabajo de SQL. También puede hacer clic en el icono Execute Statement o pulsar [F9] para ejecutar una sentencia SQL en la hoja de trabajo de SQL. El icono Execute Statement ejecuta la sentencia que indica el puntero del mouse en el cuadro Enter SQL Statement mientras que el icono Run Script ejecuta todas las sentencias del cuadro Enter SQL Statement. El icono Execute Statement muestra la salida de la consulta en la página con separadores Results, mientras que el icono Run Script emula la visualización SQL*Plus y muestra la salida en la página con separadores Script Output.

Mediante SQL*Plus, termine la sentencia SQL con un punto y coma y, a continuación, pulse la tecla [Intro] para ejecutar el comando.

Valores por Defecto de Cabeceras de Columna

- SQL Developer:
  - Alineación de cabeceras por defecto: alineación a la izquierda
  - Visualización de cabeceras por defecto: mayúsculas
- SQL*Plus:
  - Las cabeceras de columna de carácter y de fecha se alinean a la izquierda
  - Las cabeceras de columna de número se alinean a la derecha
  - Visualización de cabeceras por defecto: mayúsculas

ORACLE

Valores por Defecto de Cabeceras de Columna

En SQL Developer, las cabeceras de columna se muestran en mayúsculas y alineadas a la izquierda.

```
SELECT last_name, hire_date, salary
FROM employees;
```

	LAST_NAME	HIRE_DATE	SALARY
1	Whalen	17-SEP-87	4400
2	Hartstein	17-FEB-96	13000
3	Fay	17-AUG-97	6000
4	Higgins	07-JUN-94	12000
5	Gietz	07-JUN-94	8300
6	King	17-JUN-87	24000
7	Kochhar	21-SEP-89	17000
8	De Haan	13-JAN-93	17000
9	Hunold	03-JAN-90	9000

...

Puede sustituir la visualización de cabecera de columna por un alias. Los alias de columna se tratarán más adelante en esta lección.

Agenda

- Sentencia `SELECT` básica
- Expresiones aritméticas y valores `NULL` en la sentencia `SELECT`
- Alias de columnas
- Uso del operador de concatenación, cadenas de caracteres de literales, operador de comillas alternativo y palabra clave `DISTINCT`
- Comando `DESCRIBE`

Expresiones Aritméticas

Crear expresiones con datos de fecha y números mediante operadores aritméticos.

Operador	Descripción
+	Sumar
-	Restar
*	Multiplicar
/	Dividir

ORACLE

Expresiones Aritméticas

Puede que necesite modificar la forma en la que se muestran los datos, realizar cálculos o consultar casos de posibilidades. Todo esto es posible mediante las expresiones aritméticas. Una expresión aritmética puede contener nombres de columna, valores numéricos constantes y operadores aritméticos.

Operadores Aritméticos

La diapositiva muestra los operadores aritméticos disponibles en SQL. Puede utilizar operadores aritméticos en cualquier cláusula de una sentencia SQL (excepto en la cláusula `FROM`).

Nota: con los tipos de dato `DATE` y `TIMESTAMP`, sólo puede utilizar los operadores de suma y resta.

Uso de Operadores Aritméticos

```
SELECT last_name, salary, salary + 300
FROM   employees;
```

	LAST_NAME	SALARY	SALARY+300
1	Whalen	4400	4700
2	Hartstein	13000	13300
3	Fay	6000	6300
4	Higgins	12000	12300
5	Gietz	8300	8600
6	King	24000	24300
7	Kochhar	17000	17300
8	De Haan	17000	17300
9	Hunold	9000	9300
10	Ernst	6000	6300

...

ORACLE

Uso de Operadores Aritméticos

El ejemplo de la diapositiva utiliza el operador de suma para calcular un aumento de salario de 300 dólares para todos los empleados. La diapositiva también muestra una columna `SALARY+300` en la salida.

Tenga en cuenta que la columna calculada resultante, `SALARY+300`, no es una nueva columna en la tabla `EMPLOYEES`; es sólo para visualización. Por defecto, el nombre de la nueva columna procede del cálculo que lo genera, en este caso, `salary+300`.

Nota: el servidor de Oracle ignora los espacios en blanco antes y después del operador aritmético.

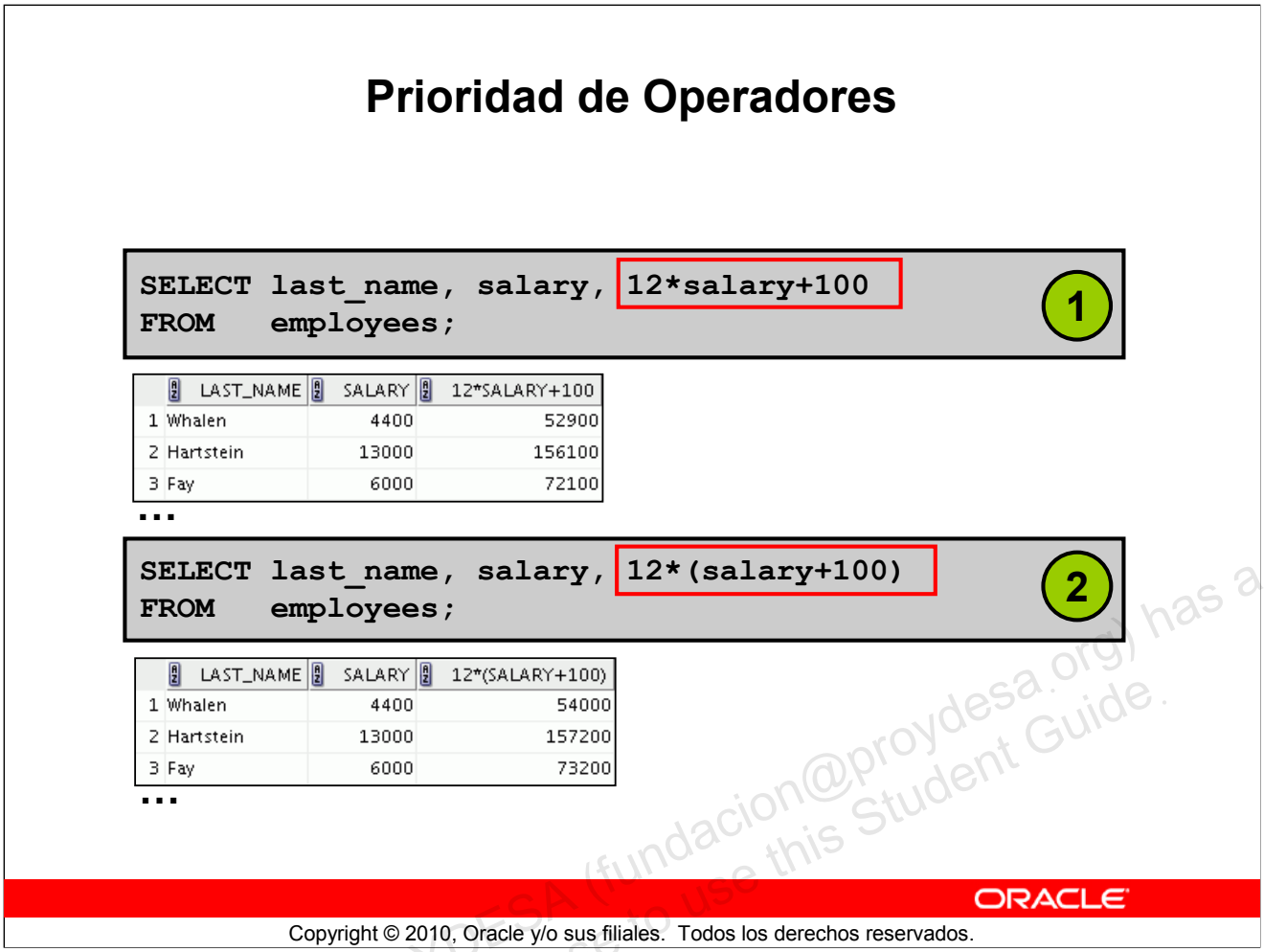
Prioridad de Operadores

Si una expresión aritmética contiene más de un operador, la multiplicación y división se evalúan primero. Si los operadores en una expresión tienen la misma prioridad, la evaluación se realiza de izquierda a derecha.

Puede utilizar los paréntesis para forzar la expresión que se incluye entre paréntesis para que se evalúe primero.

Reglas de Prioridad

- La multiplicación y división se producen antes de la suma y la resta.
- Los operadores de la misma prioridad se evalúan de izquierda a derecha.
- Los paréntesis se utilizan para sustituir la prioridad por defecto o para aclarar la sentencia.



Prioridad de Operadores (continuación)

El primer ejemplo de la diapositiva muestra el apellido, el salario y la compensación anual de los empleados. Calcula la compensación anual multiplicando el salario mensual por 12, más un incentivo de 100 dólares. Tenga en cuenta que la multiplicación se realiza antes de la suma.

Nota: utilice los paréntesis para reforzar el orden estándar de prioridad y mejorar la claridad. Por ejemplo, la expresión de la diapositiva se puede escribir como `(12*salary)+100` sin producir ningún cambio en el resultado.

Uso de Paréntesis

Puede sustituir las reglas de prioridad utilizando paréntesis para especificar el orden en el que desea que se ejecuten los operadores.

El segundo ejemplo de la diapositiva muestra el apellido, el salario y la compensación anual de los empleados. Calcula la compensación anual de la siguiente forma: suma un incentivo mensual de 100 dólares al salario mensual y, a continuación, multiplica dicho subtotal por 12. Debido a los paréntesis, la suma tiene prioridad sobre la multiplicación.

Definición de Valor Nulo

- Un valor nulo es un valor que no está disponible, sin asignar, desconocido o que no es aplicable.
- Un valor nulo no es lo mismo que un cero o un espacio en blanco.

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Whalen	AD_ASST	4400	(null)
2	Hartstein	MK_MAN	13000	(null)
...				
17	Zlotkey	SA_MAN	10500	0.2
18	Abel	SA_REP	11000	0.3
19	Taylor	SA_REP	8600	0.2
20	Grant	SA_REP	7000	0.15

ORACLE

Definición de Valor Nulo

Si una fila carece de un valor de datos para una columna concreta, se dice que dicho valor es *nulo* o que contiene un valor nulo.

Un valor nulo es un valor que no está disponible, sin asignar, desconocido o que no es aplicable. Un valor nulo no es lo mismo que un cero o un espacio en blanco. El cero es un número y el espacio en blanco es un carácter.

Las columnas de cualquier tipo de dato pueden contener valores nulos. Sin embargo, algunas restricciones (`NOT NULL` y `PRIMARY KEY`) evitan que se utilicen valores nulos en la columna.

En la columna `COMMISSION_PCT` de la tabla `EMPLOYEES`, observe que sólo pueden percibir una comisión un director de ventas o un vendedor. El resto de los empleados no tienen derecho a percibir comisiones. Un valor nulo representa este hecho.

Nota: por defecto, SQL Developer utiliza el literal, (null), para identificar los valores nulos. Sin embargo, se puede definir en un valor más relevante para el usuario. Para ello, seleccione Preferences en el menú Tools. En el cuadro de diálogo Preferences, amplíe el nodo Database. Haga clic en Advanced Parameters y en el panel derecho, para “Display Null value As”, introduzca un valor adecuado.

Valores Nulos en Expresiones Aritméticas

Las expresiones aritméticas que contengan un valor nulo se evalúan como nulas.

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```

	LAST_NAME	12*SALARY*COMMISSION_PCT
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)
...		
17	Zlotkey	25200
18	Abel	39600
19	Taylor	20640
20	Grant	12600

Valores Nulos en Expresiones Aritméticas

Si cualquier valor de columna en una expresión aritmética es nulo, el resultado es nulo. Por ejemplo, si intenta realizar una división entre cero, recibirá un error. Sin embargo, si divide un número entre un valor nulo, el resultado será nulo o desconocido.

En el ejemplo de la diapositiva, el empleado Whalen no percibe ninguna comisión. Porque la columna `COMMISSION_PCT` en la expresión aritmética es nula, por lo tanto, el resultado es nulo.

Para obtener más información, consulte la sección sobre elementos básicos de Oracle SQL en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Agenda

- Sentencia `SELECT` básica
- Expresiones aritméticas y valores `NULL` en la sentencia `SELECT`
- Alias de columnas
- Uso del operador de concatenación, cadenas de caracteres de literales, operador de comillas alternativo y palabra clave `DISTINCT`
- Comando `DESCRIBE`

Definición de Alias de Columna

Un alias de columna:

- Cambia el nombre de una cabecera de columna
- Es útil para realizar cálculos
- Sigue inmediatamente al nombre de columna (también puede ser la palabra clave opcional `AS` entre el nombre de columna y el alias)
- Necesita comillas dobles si contiene espacios o caracteres especiales o si es sensible a mayúsculas/minúsculas

ORACLE

Definición de Alias de Columna

Al mostrar el resultado de una consulta, SQL Developer normalmente utiliza el nombre de la columna seleccionada como la cabecera de columna. Esta cabecera puede que no sea descriptiva y, por lo tanto, puede resultar difícil de entender. Puede cambiar una cabecera de columna mediante un alias de columna.

Especifique el alias después de la columna en la lista `SELECT` con un espacio como separador. Por defecto, las cabeceras de alias aparecen en mayúscula. Si el alias contiene espacios o caracteres especiales (como # o $) o si es sensible a mayúsculas/minúsculas, incluya el alias entre comillas dobles ("").

Uso de Alias de Columna

```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```

	NAME	COMM
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)

...

```
SELECT last_name "Name" , salary*12 "Annual Salary"
FROM   employees;
```

	Name	Annual Salary
1	Whalen	52800
2	Hartstein	156000
3	Fay	72000

...

ORACLE

Uso de Alias de Columna

El primer ejemplo muestra los nombres y los porcentajes de comisión de todos los empleados. Observe que la palabra clave opcional `AS` se ha utilizado antes que el nombre de alias de columna. El resultado de la consulta es el mismo si se utiliza o no la palabra clave `AS`. Observe también que la sentencia SQL tiene los alias de columna `name` y `comm` en minúscula, mientras que el resultado de la consulta muestra las cabeceras de columna en mayúscula. Como se indica en la diapositiva anterior, las cabeceras de columna aparecen en mayúscula por defecto.

El segundo ejemplo muestra los apellidos y los salarios anuales de todos los empleados. Debido a que `Annual Salary` contiene un espacio, está incluido entre comillas dobles. Tenga en cuenta que la cabecera de columna de la salida es exactamente la misma que en el alias de columna.

Agenda

- Sentencia `SELECT` básica
- Expresiones aritméticas y valores NULL en la sentencia `SELECT`
- Alias de columnas
- Uso del operador de concatenación, cadenas de caracteres de literales, operador de comillas alternativo y palabra clave `DISTINCT`
- Comando `DESCRIBE`

Operador de Concatenación

Un operador de concatenación:

- Enlaza columnas o cadenas de caracteres a otras columnas
- Se representa con dos barras verticales (||)
- Crea una columna resultante que es una expresión de carácter

```
SELECT   last_name||job_id AS "Employees"
FROM     employees;
```

	Employees
1	AbelSA_REP
2	DaviesST_CLERK
3	De HaanAD_VP
4	ErnstIT_PROG
5	FayMK_REP
6	GietzAC_ACCOUNT

...

ORACLE

Operador de Concatenación

Puede enlazar columnas a otras columnas, expresiones aritméticas o valores constantes para crear una expresión de carácter con el operador de concatenación (||). Las columnas a ambos lados del operador se combinan para crear una sola columna de salida.

En el ejemplo, `LAST_NAME` y `JOB_ID` están concatenados y se les ha otorgado el alias `Employees`. Observe que el apellido del empleado y código de trabajo se combinan para crear una única columna de salida.

La palabra clave `AS` antes del nombre de alias facilita la lectura de la cláusula `SELECT`.

Valores Nulos con el Operador de Concatenación

Si concatena un valor nulo con una cadena de caracteres, el resultado es una cadena de caracteres. `LAST_NAME || NULL` tiene como resultado `LAST_NAME`.

Nota: también puede concatenar expresiones de fecha con otras expresiones o columnas.

Cadenas de Caracteres Literales

- Un literal es un carácter, un número o una fecha que se incluye en la sentencia `SELECT`.
- Los valores literales de caracteres y fecha se deben incluir entre comillas simples.
- Cada cadena de caracteres es la salida una vez para cada fila devuelta.

Cadenas de Caracteres Literales

Un literal es un carácter, un número o una fecha que se incluye en la lista `SELECT`. No es un nombre de columna o un alias de columna. Se imprime para cada fila devuelta. Las cadenas literales de texto de formato libre se pueden incluir en el resultado de consultas y se tratan igual que la lista `SELECT`.

Los literales de caracteres y fecha se *deben* delimitar con comillas simples (`' '`); no es necesario delimitar los literales de número de forma similar.

Uso de Cadenas de Caracteres Literales

```
SELECT last_name ||' is a '||job_id
       AS "Employee Details"
FROM   employees;
```

	Employee Details
1	Abel is a SA_REP
2	Davies is a ST_CLERK
3	De Haan is a AD_VP
4	Ernst is a IT_PROG
5	Fay is a MK_REP
6	Gietz is a AC_ACCOUNT
7	Grant is a SA_REP
8	Hartstein is a MK_MAN
9	Higgins is a AC_MGR
10	Hunold is a IT_PROG

...

ORACLE

Uso de Cadenas de Caracteres Literales

El ejemplo de la diapositiva muestra los apellidos y códigos de trabajo de todos los empleados. La columna tiene la cabecera Employee Details. Observe los espacios incluidos entre comillas simples en la sentencia `SELECT`. Los espacios mejoran la legibilidad de la salida.

En el siguiente ejemplo, el apellido y el salario de cada empleado se concatenan con un literal para proporcionar a las filas devueltas más significado:

```
SELECT last_name ||': 1 Month salary = '||salary Monthly
FROM   employees;
```

	MONTHLY
1	Whalen: 1 Month salary = 4400
2	Hartstein: 1 Month salary = 13000
3	Fay: 1 Month salary = 6000
4	Higgins: 1 Month salary = 12000
5	Gietz: 1 Month salary = 8300
6	King: 1 Month salary = 24000
7	Kochhar: 1 Month salary = 17000
8	De Haan: 1 Month salary = 17000

...

Operador de Comillas (q) Alternativo

- Especifique su propio delimitador de entrecomillado.
- Seleccione cualquier delimitador.
- Aumente la legibilidad y el uso.

```
SELECT department_name || q'[ Department's Manager Id: ]'
       || manager_id
       AS "Department and Manager"
FROM departments;
```

	Department and Manager
1	Administration Department's Manager Id: 200
2	Marketing Department's Manager Id: 201
3	Shipping Department's Manager Id: 124
4	IT Department's Manager Id: 103
5	Sales Department's Manager Id: 149
6	Executive Department's Manager Id: 100
7	Accounting Department's Manager Id: 205
8	Contracting Department's Manager Id:

Operador de Comillas (q) Alternativo

Muchas de las sentencias SQL utilizan literales de caracteres en expresiones o condiciones. Si el propio literal contiene una comilla simple, puede utilizar el operador de comillas (q) y seleccionar su propio delimitador de entrecomillado.

Puede seleccionar cualquier delimitador adecuado, ya sea de un solo byte o multibyte, o bien cualquiera de los siguientes pares de caracteres: [ ], { }, ( ) o < >.

En el ejemplo, la cadena contiene una comilla simple, que normalmente se interpreta como un delimitador de una cadena de caracteres. Sin embargo, al utilizar el operador q, los corchetes [] se utilizan como delimitador de entrecomillado. La cadena entre los delimitadores de corchetes se interpreta como una cadena de caracteres literales.

Filas Duplicadas

La visualización por defecto de las consultas incluye todas las filas, también las filas duplicadas.

1

```
SELECT department_id
FROM   employees;
```

	DEPARTMENT_ID
1	10
2	20
3	20
4	110
5	110

...

2

```
SELECT DISTINCT department_id
FROM   employees;
```

	DEPARTMENT_ID
1	(null)
2	20
3	90
4	110
5	50
6	80
7	10
8	60

ORACLE

Filas Duplicadas

A menos que indique lo contrario, SQL muestra los resultados de una consulta sin eliminar las filas duplicadas. El primer ejemplo de la diapositiva muestra todos los números de departamento de la tabla `EMPLOYEES`. Observe que los números de departamento se repiten.

Para eliminar filas duplicadas en el resultado, incluya la palabra clave `DISTINCT` en la cláusula `SELECT` inmediatamente después de la palabra clave `SELECT`. En el segundo ejemplo de la diapositiva, la tabla `EMPLOYEES` en realidad contiene 20 filas, pero sólo hay siete números de departamento únicos en la tabla.

Puede especificar varias columnas después del cualificador `DISTINCT`. El cualificador `DISTINCT` afecta a todas las columnas seleccionadas y el resultado es cada combinación distinta de columnas.

```
SELECT  DISTINCT department_id, job_id
FROM    employees;
```

	DEPARTMENT_ID	JOB_ID
1	110	AC_ACCOUNT
2	90	AD_VP
3	50	ST_CLERK

...

Nota: también puede especificar la palabra clave `UNIQUE`, que es un sinónimo de la palabra clave `DISTINCT`.

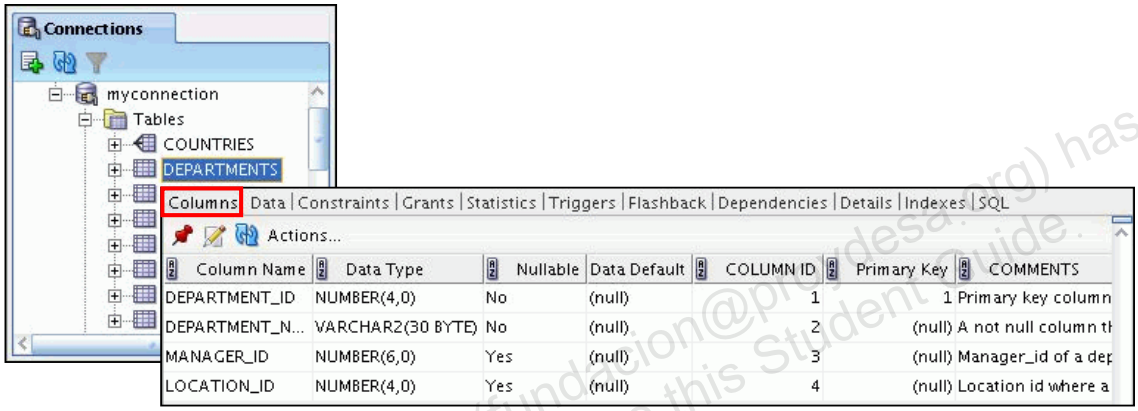
Agenda

- Sentencia `SELECT` básica
- Expresiones aritméticas y valores `NULL` en la sentencia `SELECT`
- Alias de columnas
- Uso del operador de concatenación, cadenas de caracteres de literales, operador de comillas alternativo y palabra clave `DISTINCT`
- Comando `DESCRIBE`

Visualización de la Estructura de la Tabla

- Utilizar el comando `DESCRIBE` para mostrar la estructura de una tabla.
- O seleccionar la tabla en el árbol Connections y utilizar el separador Columns para ver la estructura de la tabla.

```
DESC[RIBE] tablename
```



Visualización de la Estructura de la Tabla

Puede mostrar la estructura de una tabla mediante el comando `DESCRIBE`. El comando muestra los nombres de columna y los tipos de dato e indica si una columna *debe* contener datos (es decir, si la columna tiene una restricción `NOT NULL`).

En la sintaxis, *`table name`* es el nombre de cualquier tabla existente, vista o sinónimo accesible al usuario.

Mediante la interfaz gráfica de usuario de SQL Developer, puede seleccionar el árbol Connections y utilizar el separador Columns para ver la estructura de la tabla.

Nota: el comando `DESCRIBE` está soportado tanto en SQL*Plus como en SQL Developer.

Uso del Comando DESCRIBE

```
DESCRIBE employees
```

```
DESCRIBE employees
Name                          Null     Type
----------------------------- -------- --------------
EMPLOYEE_ID                   NOT NULL NUMBER(6)
FIRST_NAME                             VARCHAR2(20)
LAST_NAME                     NOT NULL VARCHAR2(25)
EMAIL                         NOT NULL VARCHAR2(25)
PHONE_NUMBER                           VARCHAR2(20)
HIRE_DATE                     NOT NULL DATE
JOB_ID                        NOT NULL VARCHAR2(10)
SALARY                                 NUMBER(8,2)
COMMISSION_PCT                         NUMBER(2,2)
MANAGER_ID                             NUMBER(6)
DEPARTMENT_ID                          NUMBER(4)

11 rows selected
```


Uso del Comando DESCRIBE

En el ejemplo de la diapositiva se muestra la información sobre la estructura de la tabla `EMPLOYEES` mediante el comando `DESCRIBE`.

En la visualización resultante, *Null* indica que los valores de esta columna pueden ser desconocidos. `NOT NULL` indica que una columna debe contener datos. *Type* muestra el tipo de dato de una columna.

Los tipos de dato se describen en la siguiente tabla:

Tipo de Dato	Descripción
`NUMBER(p,s)`	Valor numérico que tiene un número máximo de dígitos *p*, con dígitos *s* a la derecha del punto decimal.
`VARCHAR2(s)`	Valor del carácter de longitud de variable del tamaño máximo *s*.
`DATE`	Valor de fecha y hora entre el 1 de enero de 4712 a.C. y el 31 de diciembre de 9999 d.C.

Prueba

Identificar las sentencias `SELECT` que se ejecutan correctamente.

1.
```
SELECT first_name, last_name, job_id, salary*12
 AS Yearly Sal
FROM   employees;
```

2.
```
SELECT first_name, last_name, job_id, salary*12
 "yearly sal"
FROM   employees;
```

3.
```
SELECT first_name, last_name, job_id, salary AS
 "yearly sal"
FROM   employees;
```

4.
```
SELECT first_name+last_name AS name, job_Id,
 salary*12 yearly sal
FROM   employees;
```

ORACLE

Respuesta: 2, 3

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Escribir una sentencia `SELECT` que:
  - Devuelva todas las filas y columnas de una tabla
  - Devuelva las columnas especificadas de una tabla
  - Utilice alias de columna para mostrar cabeceras de columna más descriptivas

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM table;
```


ORACLE

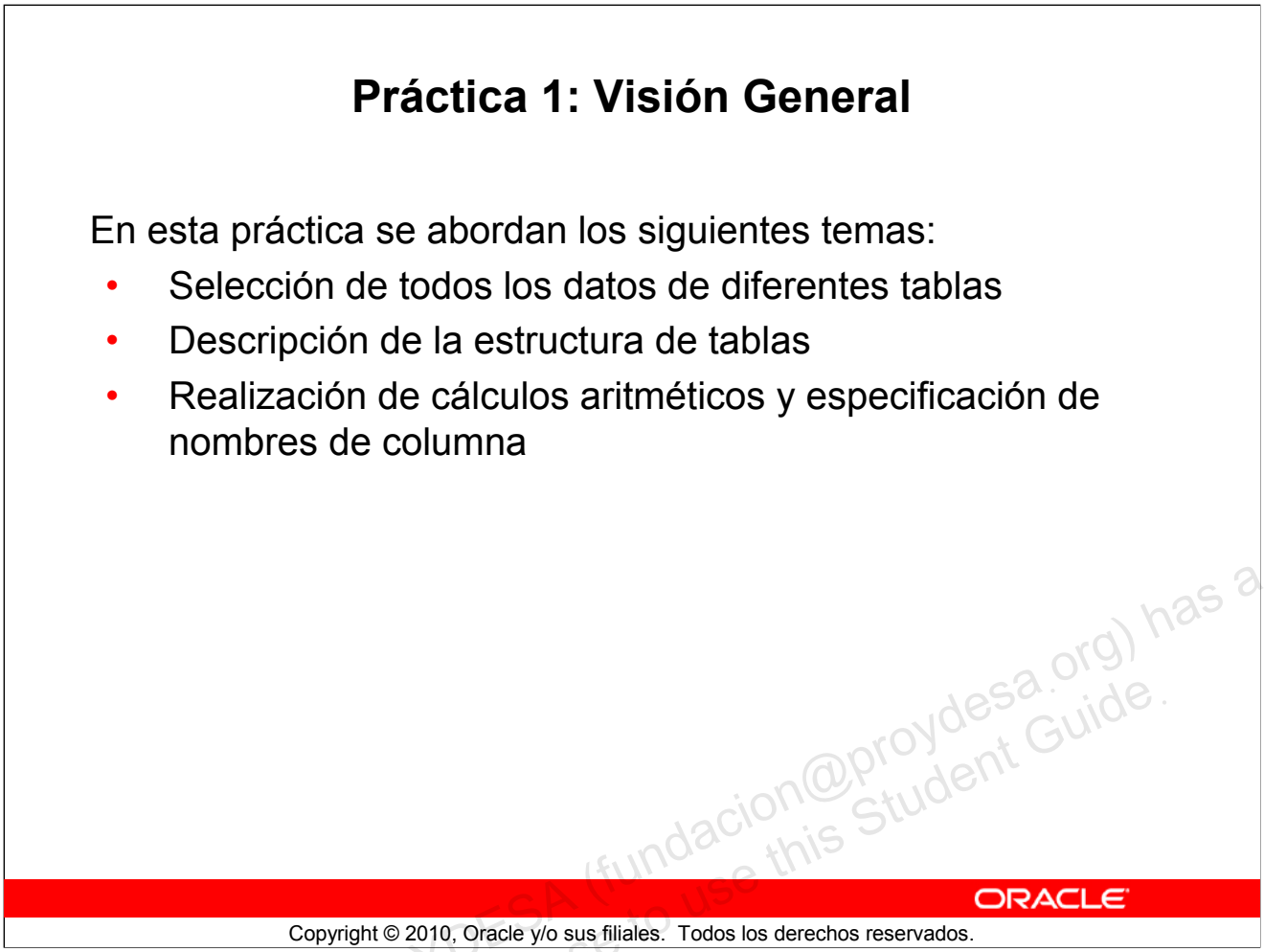
Resumen

En esta lección, debe haber aprendido cómo recuperar datos de una tabla de base de datos con la sentencia `SELECT`.

```
SELECT  *|{[DISTINCT] column [alias],...}
FROM    table;
```

En la sintaxis:

`SELECT`	es una lista de una o más columnas.
`*`	selecciona todas las columnas.
`DISTINCT`	suprime los duplicados.
`column|expression`	selecciona la columna o expresión especificada.
`alias`	proporciona diferentes cabeceras de las columnas seleccionadas.
`FROM` *`table`*	especifica la tabla que contiene las columnas.



Práctica 1: Visión General

En esta práctica, escribirá consultas simples `SELECT`. Las consultas tratan la mayoría de las operaciones y cláusulas `SELECT` que ha aprendido en esta lección.

2

Restricción y Ordenación de Datos

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Limitar las filas recuperadas por una consulta
- Ordenar las filas recuperadas por una consulta
- Usar la sustitución con ampersand para restringir y ordenar la salida en tiempo de ejecución

Objetivos

Al recuperar los datos de la base de datos, puede que necesite realizar las siguientes acciones:

- Restringir las filas de datos que se muestran
- Especificar el orden en el que aparecen las filas

Esta lección explica las sentencias SQL que se utilizan para realizar las acciones anteriores.

Agenda

- Limitación de filas con:
  - Cláusula `WHERE`
  - Condiciones de comparación con las condiciones =, <=, `BETWEEN`, `IN`, `LIKE` y `NULL`
  - Condiciones lógicas mediante los operadores `AND`, `OR` y `NOT`
- Reglas de prioridad de los operadores en una expresión
- Ordenación de filas mediante la cláusula `ORDER BY`
- Variables de sustitución
- Comandos `DEFINE` y `VERIFY`

Limitación de Filas con una Selección

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20
4	205	Higgins	AC_MGR	110
5	206	Gietz	AC_ACCOUNT	110

...

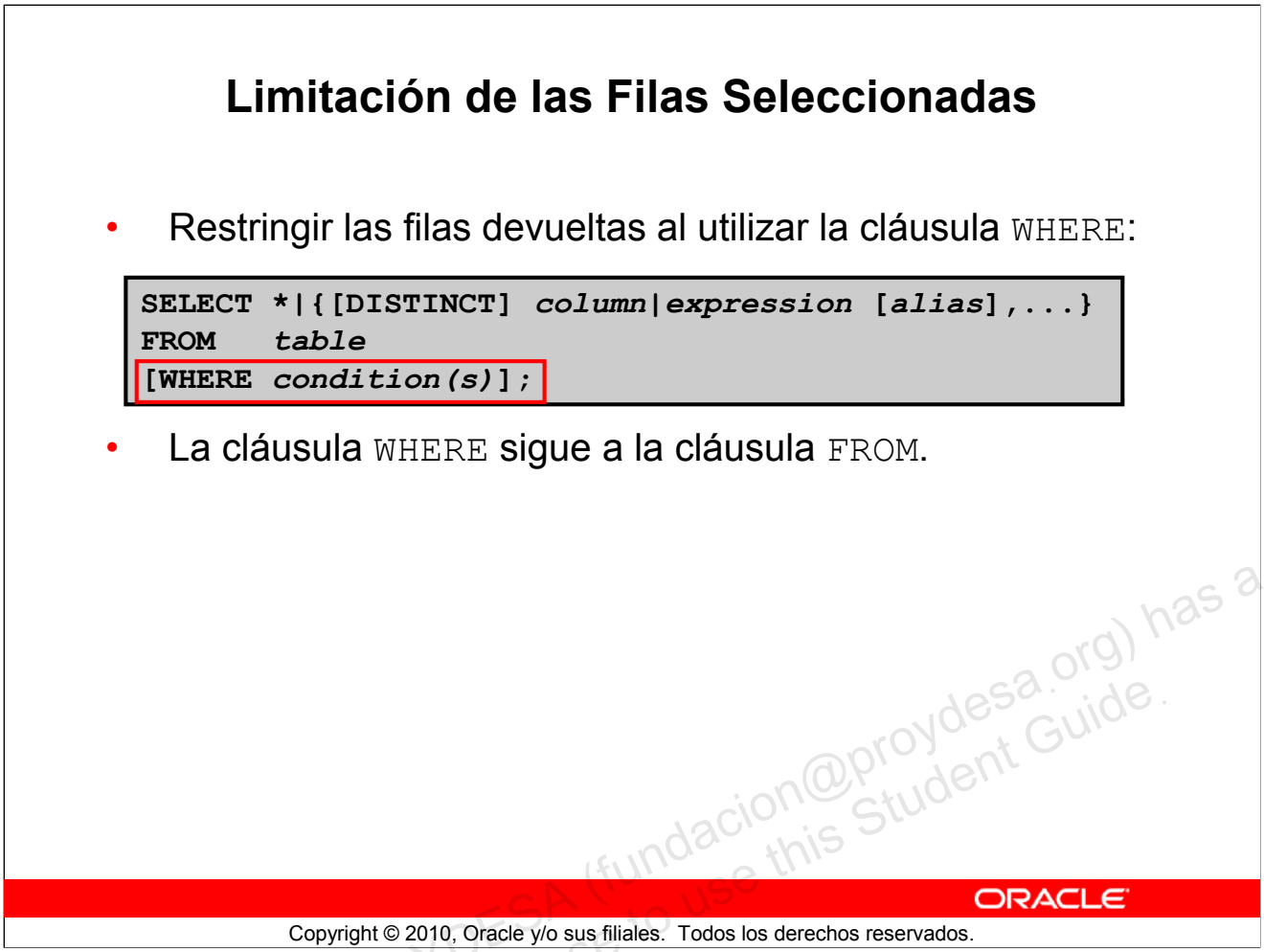
"recuperar todos los empleados del departamento 90"

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

ORACLE

Limitación de Filas con una Selección

En el ejemplo de la diapositiva, suponga que desea mostrar todos los empleados del departamento 90. Las filas con un valor de 90 de la columna `DEPARTMENT_ID` son las únicas que se devuelve. Este método de restricción es la base de la cláusula `WHERE` en SQL.



Limitación de las Filas Seleccionadas

Puede restringir las filas que devuelve la consulta al utilizar la cláusula `WHERE`. Una cláusula `WHERE` contiene una condición que se debe cumplir e, inmediatamente después, le sigue la cláusula `FROM`. Si la condición es verdadera, se devolverá la fila que cumpla con la condición.

En la sintaxis:

`WHERE`	restringe la consulta a filas que cumplan con una condición.
`condition`	está compuesto por nombres de columna, expresiones, constantes y un operador de comparación. Una condición especifica una combinación de una o más expresiones y operadores lógicos (booleanos) y devuelve un valor de `TRUE`, `FALSE` o `UNKNOWN`.

La cláusula `WHERE` puede comparar valores en columnas, literales, expresiones aritméticas o funciones. Consta de tres elementos:

- Nombre de la columna
- Condición de comparación
- Nombre de la columna, constante o lista de valores

Uso de la Cláusula WHERE

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

ORACLE

Uso de la Cláusula WHERE

En el ejemplo, la sentencia `SELECT` recupera el ID de empleado, apellido, ID de cargo y número de departamento de todos los empleados del departamento 90.

Nota: puede utilizar el alias de columna en la cláusula `WHERE`.

Fechas y Cadenas de Caracteres

- Las cadenas de caracteres y valores de fecha se incluyen entre comillas simples.
- Los valores de caracteres son sensibles a mayúsculas/minúsculas y los valores de datos son sensibles a formato.
- El formato de visualización de la fecha por defecto es `DD-MON-RR`.

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen' ;
```

```
SELECT last_name
FROM   employees
WHERE  hire_date = '17-FEB-96' ;
```

ORACLE

Fechas y Cadenas de Caracteres

Las fechas y cadenas de caracteres de la cláusula `WHERE` se deben incluir entre comillas simples (`''`). Sin embargo, las constantes numéricas no se deben incluir entre comillas simples.

Todas las búsquedas de caracteres son sensibles a mayúsculas/minúsculas. En el siguiente ejemplo, no se devuelve ninguna fila porque la tabla `EMPLOYEES` almacena todos los apellidos en minúscula y en mayúscula:

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'WHALEN';
```

Las bases de datos Oracle almacenan fechas en un formato numérico interno que representa el siglo, el año, el mes, las horas, los minutos y los segundos. La visualización de la fecha por defecto es `DD-MON-RR`.

Nota: para obtener más información sobre el formato RR y sobre cómo cambiar el formato de fecha por defecto, consulte la lección titulada “Uso de Funciones de Una Sola Fila para Personalizar la Salida”. Además, aprenderá acerca del uso de funciones de una sola fila como `UPPER` y `LOWER` para sustituir la sensibilidad a mayúsculas/minúsculas en la misma lección.

Operadores de Comparación

Operador	Significado
=	Igual que
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
<>	Distinto de
BETWEEN ...AND...	Entre dos valores (ambos incluidos)
IN(set)	Coincide con cualquiera de los
LIKE	Coincide con un patrón de
IS NULL	Es un valor nulo

ORACLE

Operadores de Comparación

Los operadores de comparación se utilizan en condiciones que comparan una expresión con otra expresión o valor. Se utilizan en la cláusula `WHERE` en el siguiente formato:

Sintaxis

```
... WHERE expr operator value
```

Ejemplo

```
... WHERE hire_date = '01-JAN-95'
... WHERE salary >= 6000
... WHERE last_name = 'Smith'
```

Recuerde, un alias no se puede utilizar en la cláusula `WHERE`.

Nota: los símbolos `!=` y `^=` también pueden representar la condición *not equal to*.

Uso de Operadores de Comparación

```
SELECT last_name, salary
FROM   employees
WHERE  salary <= 3000 ;
```

	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

ORACLE

Uso de Operadores de Comparación

En el ejemplo, la cláusula `SELECT` recupera el apellido y el salario de la tabla `EMPLOYEES` para cualquier empleado cuyo salario sea menor o igual que 3.000 dólares. Tenga en cuenta que existe un valor explícito proporcionado a la cláusula `WHERE`. El valor explícito de `3000` se compara con el valor de salario de la columna `SALARY` de la tabla `EMPLOYEES`.

Uso de Condiciones de Rango mediante el Operador BETWEEN

Utilizar el operador BETWEEN para mostrar las filas basadas en un rango de valores:

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500 ;
```

Límite inferior Límite superior

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

Uso de Condiciones de Rango mediante el Operador BETWEEN

Puede utilizar las filas basadas en un rango de valores utilizando la condición de rango BETWEEN. El rango que especifique contiene un límite inferior y un límite superior.

La sentencia SELECT de la diapositiva devuelve filas de la tabla EMPLOYEES para cualquier empleado cuyo salario esté entre 2.500 y 3.500 dólares.

También se incluyen los valores que se especifiquen con el operador BETWEEN. Debe especificar primero el límite inferior.

También puede utilizar la condición BETWEEN en los valores de caracteres:

```
SELECT last_name
FROM   employees
WHERE  last_name BETWEEN 'King' AND 'Smith';
```

	LAST_NAME
1	King
2	Kochhar
3	Lorentz
4	Matos
5	Mourgos
6	Rajs

Condición de Miembro mediante el Operador IN

Utilizar el operador IN para probar los valores de una lista:

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	201	Hartstein	13000	100
2	101	Kochhar	17000	100
3	102	De Haan	17000	100
4	124	Mourgos	5800	100
5	149	Zlotkey	10500	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

ORACLE

Condición de Miembro mediante el Operador IN

Para probar valores de un juego especificado de valores, utilice el operador IN. La condición definida mediante el operador IN también se denomina *condición de miembro*.

El ejemplo de la diapositiva muestra los números de empleado, apellidos, salarios y números de empleado de los gestores de todos los empleados cuyo número de empleado del gestor sea 100, 101 o 201.

Nota: el juego de valores se puede especificar en cualquier orden aleatorio (por ejemplo, 201,100,101).

La condición IN se puede utilizar con cualquier tipo de dato. El siguiente ejemplo devuelve una fila de la tabla EMPLOYEES para cualquier empleado cuyo apellido esté incluido en la lista de nombres de la cláusula WHERE:

```
SELECT employee_id, manager_id, department_id
FROM   employees
WHERE  last_name IN ('Hartstein', 'Vargas');
```

Si los caracteres o fechas se utilizan en la lista, se deben incluir entre comillas simples (' ').

Nota: el servidor de Oracle evalúa el operador IN internamente como un juego de condiciones OR, como a=value1 o a=value2 o a=value3. Por lo tanto, el uso del operador IN no tiene ninguna ventaja de rendimiento y sólo se utiliza para la simplicidad lógica.

Coincidencia de Patrones mediante el Operador `LIKE`

- Utilizar el operador `LIKE` para realizar búsquedas con comodines de valores de cadena de búsqueda válidos.
- Las condiciones de búsqueda pueden contener caracteres literales o números:
  - `%` indica cero o varios caracteres.
  - `_` indica un carácter.

```
SELECT   first_name
FROM     employees
WHERE    first_name LIKE 'S%' ;
```


Coincidencia de Patrones mediante el Operador `LIKE`

Puede que no siempre conozca el valor exacto que debe buscar. Puede seleccionar filas que coincidan con un patrón de caracteres utilizando la condición `LIKE`. Se hace referencia a la operación de coincidencia de patrón de caracteres como búsqueda con *comodines*. Para crear la cadena de búsqueda se pueden utilizar dos símbolos.

Símbolo	Descripción
%	Representa cualquier secuencia de cero o más caracteres
_	Representa un único carácter

La sentencia `SELECT` de la diapositiva devuelve el nombre del empleado de la tabla `EMPLOYEES` de cualquier empleado cuyo nombre empiece por la letra “S”. Observe que se trata de la “S” mayúscula. No se devolverán los nombres que empiecen por “s” minúscula.

La condición `LIKE` se puede utilizar como un método abreviado para algunas comparaciones `BETWEEN`. El siguiente ejemplo muestra los apellidos y fechas de contratación de los empleados que comenzaron a trabajar entre enero y diciembre de 1995:

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date LIKE '%95';
```

Combinación de Caracteres Comodín

- Puede combinar los dos caracteres comodín (%, _) con caracteres literales para la coincidencia de patrones:

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%' ;
```

	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

- Puede utilizar el identificador `ESCAPE` para buscar los símbolos `%` y _ reales.

Combinación de Caracteres Comodín

Los símbolos `%` y _ se pueden utilizar en cualquier combinación con los caracteres literales. El ejemplo de la diapositiva muestra los nombres de todos los empleados cuyos apellidos tengan la letra "o" como segundo carácter.

Identificador **`ESCAPE`**

Cuando necesite tener una coincidencia exacta para los caracteres `%` y _ reales, utilice el identificador `ESCAPE`. Esta opción especifica lo que representa el carácter de escape. Si desea buscar cadenas que contengan `SA_`, puede utilizar la siguiente sentencia SQL:

```
SELECT employee_id, last_name, job_id
FROM   employees WHERE  job_id LIKE '%SA\_%' ESCAPE '\';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID
1	149	Zlotkey	SA_MAN
2	174	Abel	SA_REP
3	176	Taylor	SA_REP
4	178	Grant	SA_REP

El identificador `ESCAPE` identifica la barra invertida (\) como carácter de escape. En la sentencia SQL, el carácter de escape precede al guión bajo (_). Esto hace que el servidor de Oracle interprete el carácter de subrayado literalmente.

Uso de las Condiciones NULL

Probar condiciones nulas con el operador `IS NULL`.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```

	LAST_NAME	MANAGER_ID
1	King	(null)

ORACLE

Uso de las Condiciones NULL

Las condiciones `NULL` incluyen las condiciones `IS NULL` e `IS NOT NULL`.

La condición `IS NULL` prueba las condiciones nulas. Un valor nulo significa que el valor no está disponible, no está asignado, se desconoce o no es aplicable. Por lo tanto, no puede probar con = porque un valor nulo no puede ser igual o desigual a cualquier valor. El ejemplo de la diapositiva recupera los apellidos y gestores de todos los empleados que no tienen un gestor.

Aquí se muestra otro ejemplo: Para mostrar un apellido, ID de cargo y comisión de todos los empleados que *no* tienen derecho a recibir una comisión, utilice la siguiente sentencia SQL:

```
SELECT last_name, job_id, commission_pct
FROM   employees
WHERE  commission_pct IS NULL;
```

	LAST_NAME	JOB_ID	COMMISSION_PCT
1	Whalen	AD_ASST	(null)
2	Hartstein	MK_MAN	(null)
3	Fay	MK_REP	(null)
4	Higgins	AC_MGR	(null)
5	Gietz	AC_ACCOUNT	(null)

...

Definición de Condiciones mediante los Operadores Lógicos

Operador	Significado
AND	Devuelve TRUE si *ambas* condiciones de componente son verdaderas
OR	Devuelve TRUE si *cualquier* condición de componente es verdadera
NOT	Devuelve TRUE si la condición es falsa

ORACLE

Definición de Condiciones mediante los Operadores Lógicos

Una condición lógica combina el resultado de dos condiciones de componentes para producir un resultado único basado en dichas condiciones o invierte el resultado de una condición única. Se devuelve una fila sólo si el resultado global de la condición es verdadera.

En SQL, están disponibles tres operadores lógicos:

- AND
- OR
- NOT

Todos los ejemplos indicados hasta ahora han especificado sólo una condición en la cláusula WHERE. Puede utilizar varias condiciones en una única cláusula WHERE mediante los operadores AND y OR.

Uso del Operador AND

AND necesita que ambas condiciones sean verdaderas:

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
AND    job_id LIKE '%MAN%' ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	201	Hartstein	MK_MAN	13000
2	149	Zlotkey	SA_MAN	10500

ORACLE

Uso del Operador AND

En el ejemplo, ambas condiciones deben ser verdaderas para poder seleccionar cualquier registro. Por lo tanto, sólo se seleccionan los empleados que tengan un puesto que contenga la cadena 'MAN' *y* que ganen 10.000 dólares o más.

Todas las búsquedas de caracteres son sensibles a mayúsculas/minúsculas, es decir, no se devuelve ninguna fila si 'MAN' no está en mayúsculas. Las cadenas de caracteres se deben incluir entre comillas simples.

Tabla de Verdad AND

La siguiente tabla muestra los resultados de combinar dos expresiones con AND:

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Uso del Operador OR

OR necesita que cualquier condición sea verdadera:

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%' ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	201	Hartstein	MK_MAN	13000
2	205	Higgins	AC_MGR	12000
3	100	King	AD_PRES	24000
4	101	Kochhar	AD_VP	17000
5	102	De Haan	AD_VP	17000
6	124	Mourgos	ST_MAN	5800
7	149	Zlotkey	SA_MAN	10500
8	174	Abel	SA_REP	11000

Uso del Operador OR

En el ejemplo, cualquier condición de componente debe ser verdadera para poder seleccionar cualquier registro. Por lo tanto, sólo se seleccionan los empleados que tengan un ID de trabajo que contenga la cadena 'MAN' *o* que ganen 10.000 dólares o más.

Tabla de Verdad OR

La siguiente tabla muestra los resultados de combinar dos expresiones con OR:

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Uso del Operador NOT

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

ORACLE

Uso del Operador NOT

En el ejemplo de la diapositiva se muestra el apellido y el ID de trabajo de todos los empleados cuyo ID de trabajo *no sea* `IT_PROG`, `ST_CLERK` o `SA_REP`.

Tabla de Verdad NOT

La siguiente tabla muestra el resultado de aplicar el operador `NOT` a una condición:

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

Nota: el operador `NOT` también se puede utilizar con otros operadores SQL, como `BETWEEN`, `LIKE` y `NULL`.

```
... WHERE  job_id     NOT  IN (‘AC_ACCOUNT’, ‘AD_VP’)
... WHERE  salary     NOT  BETWEEN  10000 AND  15000
... WHERE  last_name NOT  LIKE ‘%A%’
... WHERE  commission_pct  IS   NOT  NULL
```

Agenda

- Limitación de filas con:
  - Cláusula `WHERE`
  - Condiciones de comparación con las condiciones `=`, `<=`, `BETWEEN`, `IN`, `LIKE` y `NULL`
  - Condiciones lógicas mediante los operadores `AND`, `OR` y `NOT`
- Reglas de prioridad de los operadores en una expresión
- Ordenación de filas mediante la cláusula `ORDER BY`
- Variables de sustitución
- Comandos `DEFINE` y `VERIFY`

ORACLE

Reglas de Prioridad

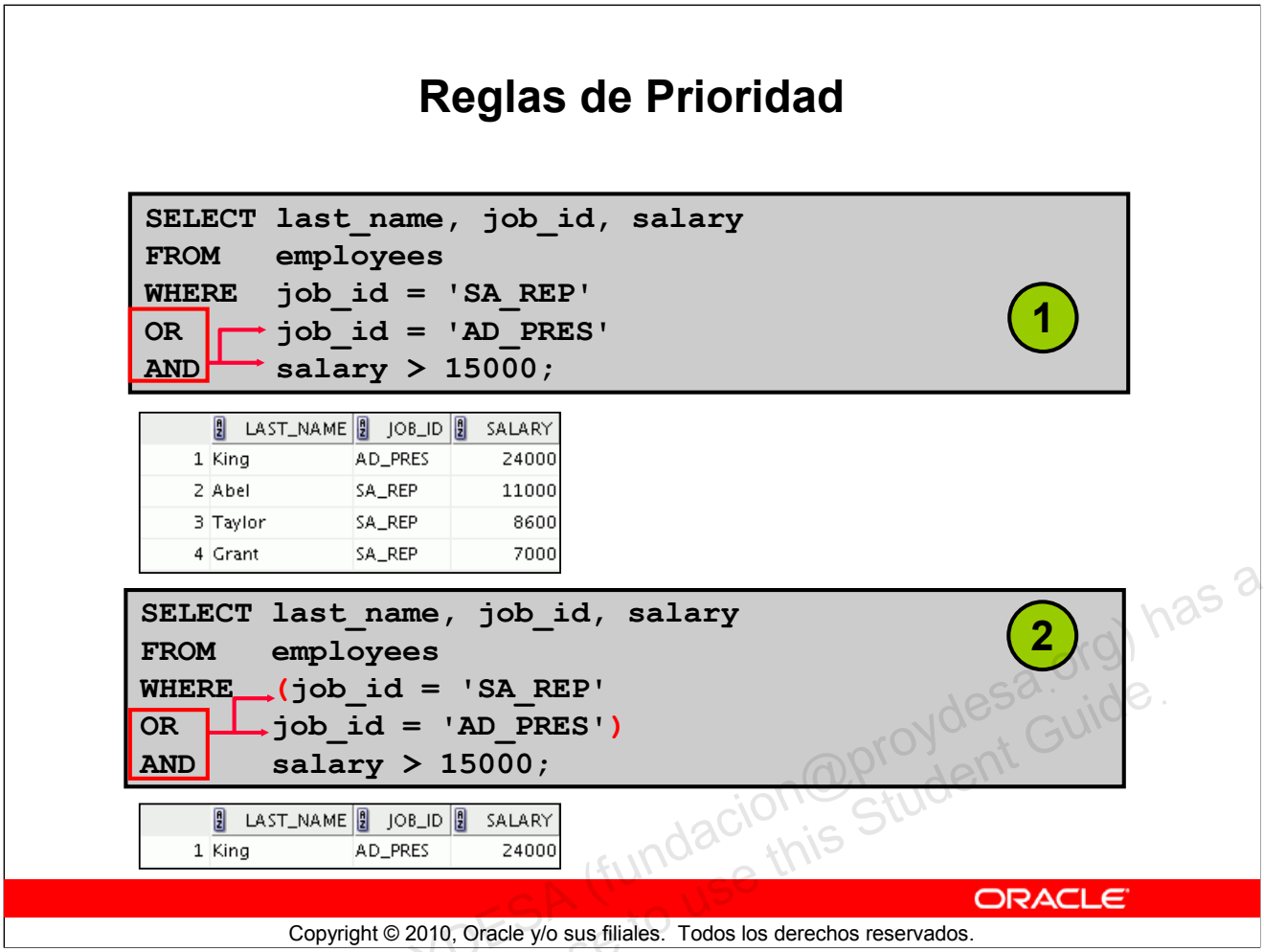
Operador	Significado
1	Operadores aritméticos
2	Operador de concatenación
3	Condiciones de comparación
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Distinto de
7	Condición lógica NOT
8	Condición lógica AND
9	Condición lógica OR

Puede utilizar los paréntesis para sustituir las reglas de prioridad.

ORACLE

Reglas de Prioridad

Las reglas de prioridad determinan el orden en el que se evalúan y calculan las expresiones. La tabla de la diapositiva muestra el orden de prioridad por defecto. Sin embargo, puede sustituir el orden por defecto utilizando paréntesis en las expresiones que desee calcular primero.



Reglas de Prioridad (continuación)

1. **Prioridad del Operador AND: Ejemplo**
   En este ejemplo, hay dos condiciones:
   - La primera condición es que el ID de cargo sea `AD_PRES` *y* que el salario sea superior a 15.000 dólares.
   - La segunda condición es que el ID de cargo sea `SA_REP`.

   Por lo tanto, la sentencia `SELECT` sería la siguiente:
   "Select the row if an employee is a president *and* earns more than $15,000, *or* if the employee is a sales representative."
2. **Uso de Paréntesis: Ejemplo**
   En este ejemplo, hay dos condiciones:
   - La primera condición es que el ID de trabajo sea `AD_PRES` *o* `SA_REP`.
   - La segunda condición es que el salario sea superior a 15.000 dólares.

   Por lo tanto, la sentencia `SELECT` sería la siguiente:
   "Select the row if an employee is a president *or* a sales representative, *and* if the employee earns more than $15,000."

Agenda

- Limitación de filas con:
  - Cláusula `WHERE`
  - Condiciones de comparación con las condiciones `=`, `<=`, `BETWEEN`, `IN`, `LIKE` y `NULL`
  - Condiciones lógicas mediante los operadores `AND`, `OR` y `NOT`
- Reglas de prioridad de los operadores en una expresión
- Ordenación de filas mediante la cláusula `ORDER BY`
- Variables de sustitución
- Comandos `DEFINE` y `VERIFY`

ORACLE

Uso de la Cláusula ORDER BY

- Ordenar las filas recuperadas con la cláusula ORDER BY:
  - ASC: orden ascendente, valor por defecto
  - DESC: orden descendente
- La cláusula ORDER BY es la última en una sentencia SELECT:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date ;
```

	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES	90	17-JUN-87
2	Whalen	AD_ASST	10	17-SEP-87
3	Kochhar	AD_VP	90	21-SEP-89
4	Hunold	IT_PROG	60	03-JAN-90
5	Ernst	IT_PROG	60	21-MAY-91
6	De Haan	AD_VP	90	13-JAN-93

...

ORACLE

Uso de la Cláusula ORDER BY

El orden de las filas devueltas en un resultado de consulta no está definido. La cláusula ORDER BY se puede utilizar para ordenar las filas. Sin embargo, si utiliza la cláusula ORDER BY, debe ser la última cláusula de la sentencia SQL. Además, puede especificar una expresión, un alias o una posición de columna como la condición de ordenación.

Sintaxis

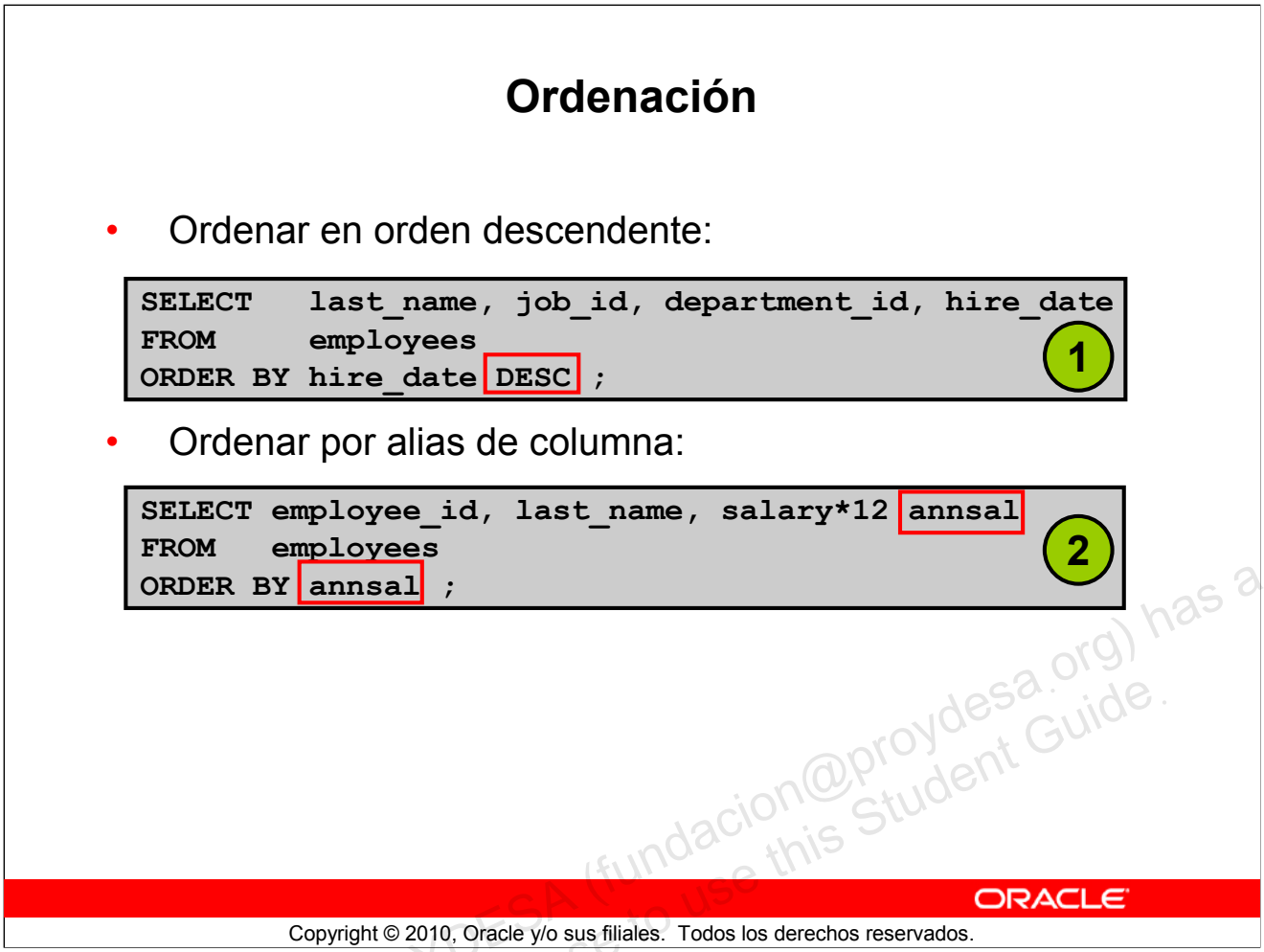
```
SELECT          expr
FROM            Tabla
[WHERE          condition(s)]
[ORDER BY  {column, expr, numeric_position} [ASC|DESC]];
```

En la sintaxis:

ORDER BY	especifica el orden en el que aparecen las filas recuperadas.
ASC	ordena las filas en orden ascendente (orden por defecto).
DESC	ordena las filas en orden descendente.

Si la cláusula ORDER BY no se utiliza, el orden no está definido y puede que el servidor de Oracle no recupere dos veces las filas en el mismo orden para la misma consulta. Utilice la cláusula ORDER BY para mostrar las filas en un orden específico.

Nota: utilice las palabras clave NULLS FIRST o NULLS LAST para especificar si las filas devueltas que contengan valores nulos deben aparecer en primer o en último lugar en la secuencia de ordenación.



Ordenación

El orden por defecto es el orden ascendente:

- Los valores numéricos se muestran con los valores más bajos primero (por ejemplo, de 1 a 999).
- Los valores de fecha se muestran con el primer valor en primer lugar (por ejemplo, 01-ENE-92 antes de 01-ENE-95).
- Los valores de caracteres se muestran en orden alfabético (por ejemplo, primero la "A" y por último la "Z").
- Los valores nulos se muestran al final para las secuencias ascendentes y al principio para las secuencias descendentes.
- Puede ordenar por una columna que no esté en la lista `SELECT`.

Ejemplos

1. Para invertir el orden en el que se muestran las filas, especifique la palabra clave `DESC` después del nombre de columna en la cláusula `ORDER BY`. El ejemplo de la diapositiva ordena el resultado por el empleado contratado más recientemente.
2. También puede utilizar un alias de columna en la cláusula `ORDER BY`. El ejemplo de la diapositiva ordena los datos por salario anual.

Nota: la palabra clave `DESC` que se utiliza aquí para la ordenación en orden descendente no se debe confundir con la palabra clave `DESC` utilizada para describir las estructuras de tabla.

Ordenación

- Ordenar por posición numérica de la columna:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY 3;
```

3

- Ordenar por varias columnas:

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

4

ORACLE

Ordenación (continuación)

Ejemplos

3. Puede ordenar los resultados de la consulta especificando la posición numérica de la columna en la cláusula `SELECT`. El ejemplo de la diapositiva ordena el resultado por `department_id`, ya que esta columna está en la tercera posición en la cláusula `SELECT`.
4. Puede ordenar los resultados de la consulta por más de una columna. El límite de ordenación es el número de columnas de la tabla determinada. En la cláusula `ORDER BY`, especifique las columnas y separe los nombres de columna con comas. Si desea invertir el orden de una columna, especifique `DESC` después del nombre. El resultado del ejemplo de consulta que se muestra en la diapositiva está ordenado por department_id en orden ascendente y por salary en orden descendente.

Agenda

- Limitación de filas con:
  - Cláusula `WHERE`
  - Condiciones de comparación con las condiciones `=`, `<=`, `BETWEEN`, `IN`, `LIKE` y `NULL`
  - Condiciones lógicas mediante los operadores `AND`, `OR` y `NOT`
- Reglas de prioridad de los operadores en una expresión
- Ordenación de filas mediante la cláusula `ORDER BY`
- Variables de sustitución
- Comandos `DEFINE` y `VERIFY`

ORACLE

Variables de Sustitución

Hasta ahora las sentencias SQL se han ejecutado con columnas y condiciones predeterminadas y sus valores. Supongamos que desea realizar una consulta que muestre los empleados con diferentes cargos, excepto aquéllos cuyo `job_ID` no sea `SA_REP`. Puede editar la cláusula `WHERE` para proporcionar un valor diferente cada vez que ejecute el comando, pero existe también una forma más sencilla.

Si se utiliza una variable de sustitución en lugar de los valores exactos en la cláusula `WHERE`, puede ejecutar la misma consulta para diferentes valores.

Puede crear informes que soliciten a los usuarios que proporcionen sus propios valores para restringir el rango de datos devuelto utilizando las variables de sustitución. Puede embeber las *variables de sustitución* en un archivo de comandos o en una única sentencia SQL. Una variable se puede considerar como un contenedor en el que se almacenan los valores temporalmente. Cuando se ejecuta la sentencia, el valor se sustituye.

Variables de Sustitución

- Utilizar variables de sustitución para:
  - Almacenar valores temporalmente con una sustitución de un solo ampersand (`&`) y de dos ampersands (`&&`)
- Utilizar las variables de sustitución para complementar:
  - Condiciones `WHERE`
  - Cláusulas `ORDER BY`
  - Expresiones de columna
  - Nombres de tabla
  - Sentencias `SELECT` completas

Variables de Sustitución (continuación)

Puede utilizar variables de sustitución de un solo ampersand (`&`) para almacenar valores temporalmente.

También puede predefinir variables mediante el comando `DEFINE`. `DEFINE` crea y asigna un valor a una variable.

Rangos de Datos Restringidos: Ejemplos

- Informes de cifras sólo del trimestre actual o de un rango de fechas específico
- Informes sobre datos relevantes sólo del usuario que solicita el informe
- Visualización del personal de sólo un departamento determinado

Otros Efectos Interactivos

Los efectos interactivos no están restringidos para dirigir la interacción del usuario a la cláusula `WHERE`. Los mismos principios se pueden utilizar también para conseguir otros objetivos, como:

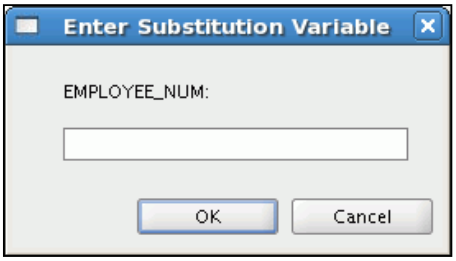
- Obtener los valores de entrada de un archivo en lugar de obtenerlos de una persona
- Transferir valores de una sentencia SQL a otra

Nota: tanto SQL Developer como SQL* Plus soportan las variables de sustitución y los comandos `DEFINE/UNDEFINE`. Ni SQL Developer ni iSQL*Plus soportan las comprobaciones de validación (excepto para tipos de dato) en la entrada de usuario. Si se utilizan en scripts que se despliegan a usuarios, las variables de sustitución se pueden manipular para ataques de inyección de SQL.

Uso de la Variable de Sustitución de Un Solo Ampersand

Utilizar una variable prefijada con un ampersand (&) para solicitar al usuario un valor:

```
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num ;
```



ORACLE

Uso de la Variable de Sustitución de Un Solo Ampersand

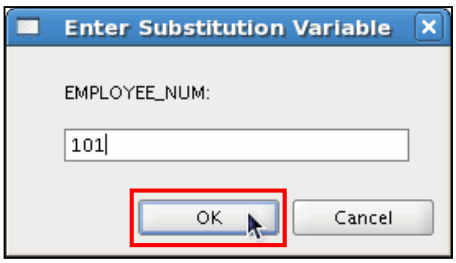
Al ejecutar un informe, los usuarios a menudo desean restringir los datos que se devuelve de forma dinámica. SQL*Plus o SQL Developer proporciona esta flexibilidad con las variables de usuario. Utilice un ampersand (&) para identificar cada variable en la sentencia SQL. Sin embargo, no es necesario que defina el valor de cada variable.

Notación	Descripción
&user_variable	Indica una variable en una sentencia SQL; si la variable no existe, SQL*Plus o SQL Developer solicita al Usuario un valor (la nueva variable se desecha después de utilizarla.)

El ejemplo de la diapositiva crea una variable de sustitución de SQL Developer para un número de empleado. Al ejecutar la sentencia, SQL Developer solicita al usuario un número de empleado y, a continuación, muestra el número de empleado, apellido, salario y número de departamento de ese empleado.

Con un solo ampersand, si la variable no existe, se pregunta al usuario cada vez que se ejecuta el comando.

Uso de la Variable de Sustitución de Un Solo Ampersand



	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	101	Kochhar	17000	90

Uso de la Variable de Sustitución de Un Solo Ampersand (continuación)

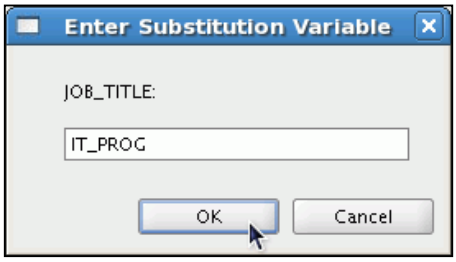
Cuando SQL Developer detecte que la sentencia SQL contiene un ampersand, se le solicitará que introduzca un valor para la variable de sustitución que se menciona en la sentencia SQL.

Después de introducir un valor y hacer clic en el botón OK, los resultados aparecen en el separador Results de la sesión de SQL Developer.

Valores de Fecha y Carácter con Variables de Sustitución

Utilizar las comillas simples para los valores de fecha y carácter:

```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title' ;
```



	LAST_NAME	DEPARTMENT_ID	SALARY*12
1	Hunold	60	108000
2	Ernst	60	72000
3	Lorentz	60	50400

ORACLE

Valores de Fecha y Carácter con Variables de Sustitución

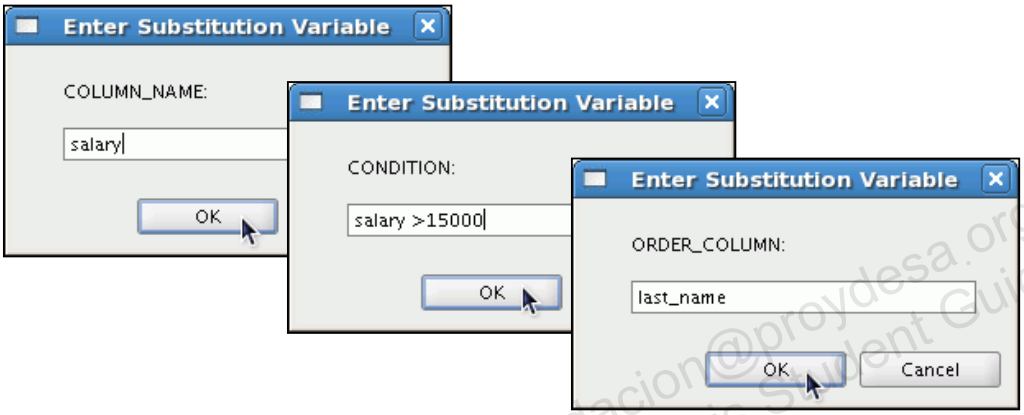
En la cláusula `WHERE`, los valores de fecha y carácter se deben incluir entre comillas simples. La misma regla se aplica a las variables de sustitución.

Incluya la variable entre comillas simples en la sentencia SQL en sí.

La diapositiva muestra una consulta para recuperar los nombres de los empleados, números de departamento y salarios anuales de todos los empleados según el valor del puesto de la variable de sustitución SQL Developer.

Especificación de Nombres de Columna, Expresiones y Texto

```
SELECT employee_id, last_name, job_id,&column_name
FROM   employees
WHERE  &condition
ORDER BY &order_column ;
```



ORACLE

Especificación de Nombres de Columna, Expresiones y Texto

No sólo puede utilizar las variables de sustitución en la cláusula `WHERE` de una sentencia SQL, sino también como sustitución de nombres de columna, expresiones o texto.

Ejemplo

El ejemplo de la diapositiva muestra el número de empleado, apellido, puesto y cualquier otra columna especificada por el usuario en tiempo de ejecución de la tabla `EMPLOYEES`. Para cada variable de sustitución de la sentencia `SELECT`, se le solicitará que introduzca un valor y, a continuación, tendrá que hacer clic en el botón OK para continuar.

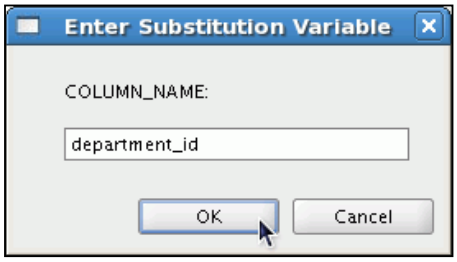
Si no introduce un valor para la variable de sustitución, obtendrá un error cuando ejecute la sentencia anterior.

Nota: una variable de sustitución se puede utilizar en cualquier lugar de la sentencia `SELECT`, excepto como la primera palabra introducida en el símbolo del sistema.

Uso de Variables de Sustitución de Dos Ampersands

Usar dos ampersands (&&) si se desea reutilizar el valor de la variable sin preguntar siempre al usuario:

```
SELECT   employee_id, last_name, job_id, &&column_name
FROM     employees
ORDER BY &column_name ;
```



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20

...

ORACLE

Uso de Variables de Sustitución de Dos Ampersands

Puede utilizar la variable de sustitución de dos ampersands (&&) si desea reutilizar el valor de la variable sin preguntar siempre al usuario. El usuario visualiza la solicitud del valor sólo una vez. En el ejemplo de la diapositiva, se le solicita al usuario que proporcione el valor para la variable `column_name` una vez. El valor proporcionado por el usuario (`department_id`) se utiliza para la visualización y el orden de los datos. Si vuelve a ejecutar la consulta, no se le pedirá el valor de la variable.

SQL Developer almacena el valor proporcionado con el comando `DEFINE`; lo vuelve a utilizar cada vez que haga referencia al nombre de la variable. Después de que una variable de usuario esté colocada, necesitará utilizar el comando `UNDEFINE` para suprimirla de la siguiente forma:

```
UNDEFINE column_name
```

Agenda

- Limitación de filas con:
  - Cláusula `WHERE`
  - Condiciones de comparación con las condiciones `=`, `<=`, `BETWEEN`, `IN`, `LIKE` y `NULL`
  - Condiciones lógicas mediante los operadores `AND`, `OR` y `NOT`
- Reglas de prioridad de los operadores en una expresión
- Ordenación de filas mediante la cláusula `ORDER BY`
- Variables de sustitución
- Comandos `DEFINE` y `VERIFY`

ORACLE

Uso del Comando DEFINE

- Usar el comando DEFINE para crear y asignar un valor a una variable.
- Usar el comando UNDEFINE de iSQL*Plus para eliminar una variable.

```
DEFINE employee_num = 200

SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num ;

UNDEFINE employee_num
```


Uso del Comando DEFINE

El ejemplo mostrado crea una variable de sustitución para un número de empleado utilizando el comando DEFINE. En tiempo de ejecución, muestra el número de empleado, nombre, salario y número de departamento de ese empleado.

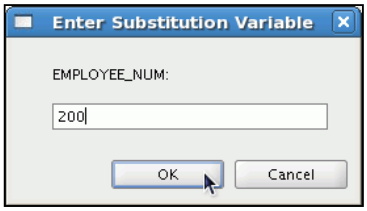
Debido a que la variable se crea utilizando el comando DEFINE de SQL Developer, no se le solicita al usuario introducir un valor para el número de empleado. En su lugar, el valor de la variable definida se sustituye automáticamente en la sentencia SELECT.

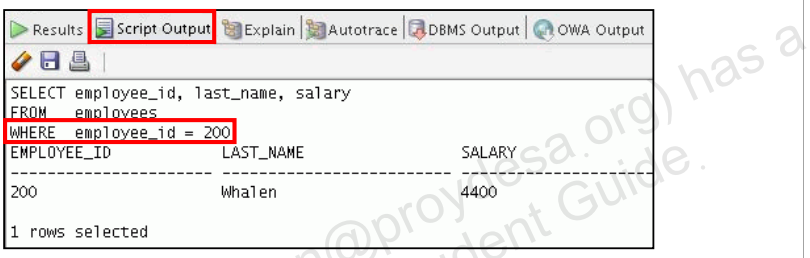
La variable de sustitución EMPLOYEE_NUM está en la sesión hasta que el usuario anula su definición o si sale de la sesión de SQL Developer.

Uso del Comando VERIFY

Usar el comando VERIFY para cambiar la visualización de la variable de sustitución, antes y después de que SQL Developer sustituya las variables de sustitución con los valores:

```
SET VERIFY ON
SELECT employee_id, last_name, salary
FROM   employees
WHERE  employee_id = &employee_num;
```





ORACLE

Uso del Comando VERIFY

Para confirmar los cambios en la sentencia SQL, utilice el comando VERIFY. La definición de SET VERIFY ON fuerza a SQL Developer a mostrar el texto de un comando después de que sustituya las variables de sustitución por valores. Para ver la salida de VERIFY, debe utilizar el icono Run Script (F5) en la hoja de trabajo de SQL. SQL Developer muestra el texto de un comando después de que sustituya las variables de sustitución por valores, en el separador Script Output como se muestra en la diapositiva.

El ejemplo de la diapositiva muestra el nuevo valor de la columna EMPLOYEE_ID en la sentencia SQL seguida de la salida.

Variables del Sistema SQL*Plus

SQL*Plus utiliza varias variables del sistema que controlan el entorno de trabajo. Una de esas variables es VERIFY. Para obtener una lista completa de todas las variables del sistema, puede emitir el comando SHOW ALL en el símbolo del sistema de SQL*Plus.

Prueba

¿Cuáles de los siguientes operadores son válidos para la cláusula `WHERE`?

1. `>=`
2. `IS NULL`
3. `!=`
4. `IS LIKE`
5. `IN BETWEEN`
6. `<>`

ORACLE

Respuesta: 1, 2, 3, 6

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Usar la cláusula `WHERE` para restringir las filas de la salida:
  - Usar las condiciones de comparación
  - Usar los operadores `BETWEEN`, `IN`, `LIKE` y `NULL`
  - Aplicar los operadores lógicos `AND`, `OR` y `NOT`
- Usar la cláusula `ORDER BY` para ordenar las filas de la salida:

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM    table
[WHERE  condition(s)]
[ORDER BY {column, expr, alias} [ASC|DESC]] ;
```

- Usar la sustitución con ampersand para restringir y ordenar la salida en tiempo de ejecución

ORACLE

Resumen

En esta lección debe haber aprendido a restringir y ordenar filas devueltas por la sentencia `SELECT`. También debe haber aprendido cómo implantar varios operadores y condiciones.

Al utilizar las variables de sustitución, puede agregar flexibilidad a las sentencias SQL. Esto permite que las consultas soliciten condiciones de filtro para las filas durante el tiempo de ejecución.

Práctica 2: Visión General

En esta práctica se abordan los siguientes temas:

- Selección de datos y cambio del orden de las filas que se muestran
- Restricción de filas mediante la cláusula `WHERE`
- Ordenación de filas mediante la cláusula `ORDER BY`
- Uso de las variables de sustitución para agregar flexibilidad a las sentencias SQL `SELECT`

ORACLE

Práctica 2: Visión General

En esta práctica se crean más informes, incluidas sentencias que utilizan las cláusulas `WHERE` y `ORDER BY`. Puede hacer que las sentencias SQL sean más reutilizables y genéricas incluyendo la sustitución con ampersand.

3

Uso de Funciones de Una Sola Fila para Personalizar la Salida

ORACLE

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Describir los diferentes tipos de funciones disponibles en SQL
- Utilizar caracteres, números y funciones de fecha en sentencias `SELECT`

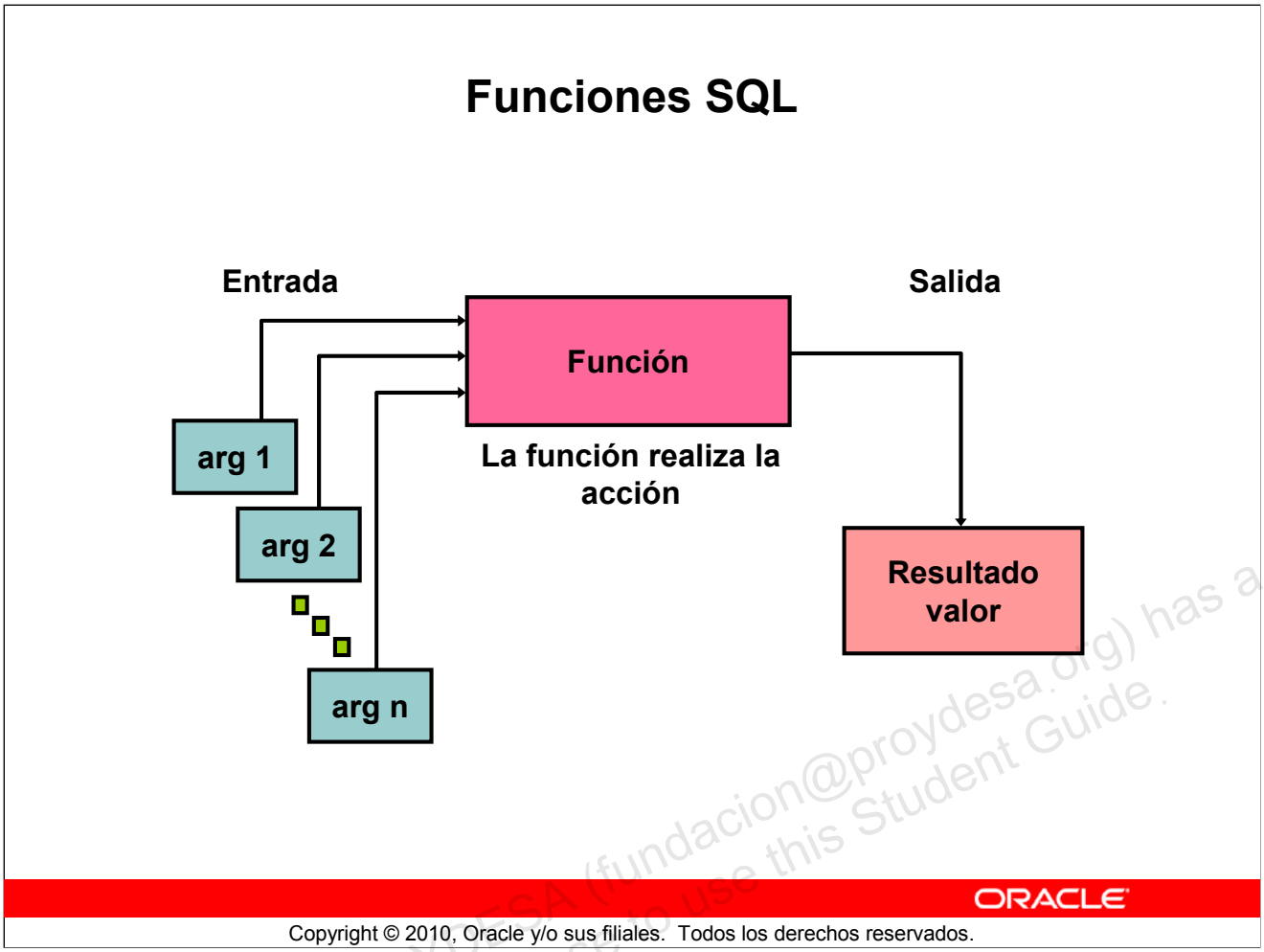
ORACLE

Objetivos

Las funciones hacen más potente el bloque de consultas básico y se utilizan para manipular los valores de datos. Las primeras dos lecciones de este curso analizan las funciones. Se centra en funciones de fecha, de carácter de una fila y de número.

Agenda

- Funciones SQL de una sola fila
- Funciones de carácter
- Funciones de número
- Trabajo con fechas
- Funciones de fecha



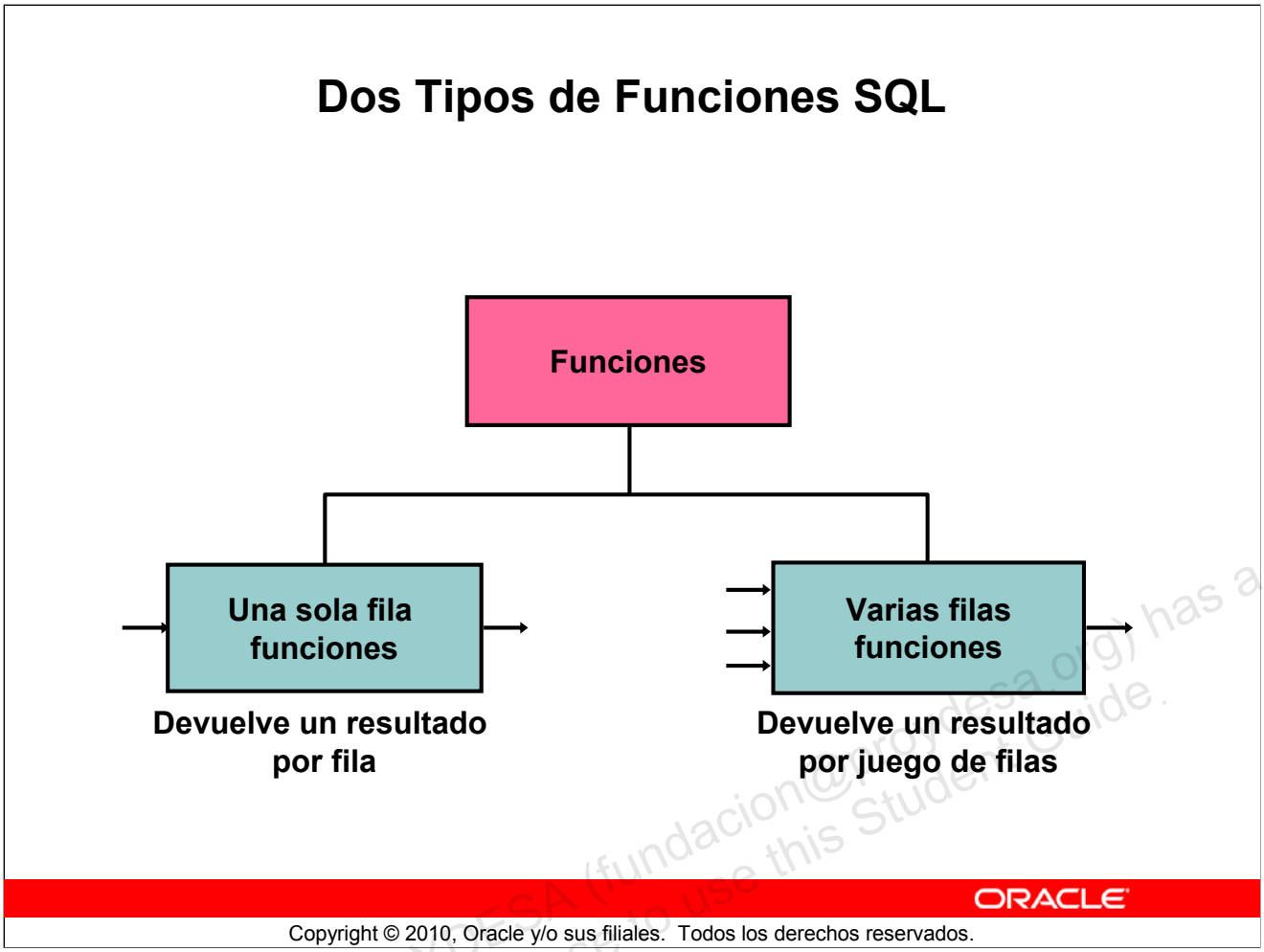
Funciones SQL

Las funciones son una característica muy potente de SQL. Se puede utilizar para realizar las siguientes acciones:

- Realizar cálculos en los datos
- Modificar elementos de datos individuales
- Manipular la salida para grupos de filas
- Formatear fechas y números para su visualización
- Convertir tipos de dato de columna

Algunas veces, las funciones SQL toman argumentos y siempre devuelven un valor.

Nota: si desea saber si una función es compatible con SQL:2003, consulte la sección sobre compatibilidad con SQL:2003 de la guía *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.



Dos Tipos de Funciones SQL

Hay dos tipos de grupos de funciones:

- Funciones de una sola fila
- Funciones de varias filas

Funciones de Una Sola Fila

Estas funciones funcionan sólo en filas únicas y devuelven un resultado por fila. Existen distintos tipos de funciones de una sola fila. En esta lección se abordan los siguientes temas:

- Carácter
- Número
- Fecha
- Conversión
- General

Funciones de Varias Filas

Las funciones pueden manipular grupos de filas para proporcionar un resultado por grupo de filas. Estas funciones también se conocen como *funciones de grupo* (se tratan en la lección titulada "Informes de Datos Agregados con Funciones de Grupo").

Nota: para obtener más información y una lista completa de las funciones disponibles y su sintaxis, consulte la sección sobre funciones en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Funciones de Una Sola Fila

Funciones de una sola fila:

- Manipular elementos de datos
- Aceptar argumentos y devolver un valor
- Actuar en cada fila devuelta
- Devolver un resultado por fila
- Posibilidad de modificar el tipo de dato
- Posibilidad de anidamiento
- Aceptar argumentos que pueden ser una columna o una expresión

```
function_name [(arg1, arg2,...)]
```


ORACLE

Funciones de Una Sola Fila

Las funciones de una sola fila se utilizan para manipular elementos de datos. Aceptan uno o varios argumentos y devuelven un valor para cada fila devuelta por la consulta. Un argumento puede ser uno de los siguientes elementos:

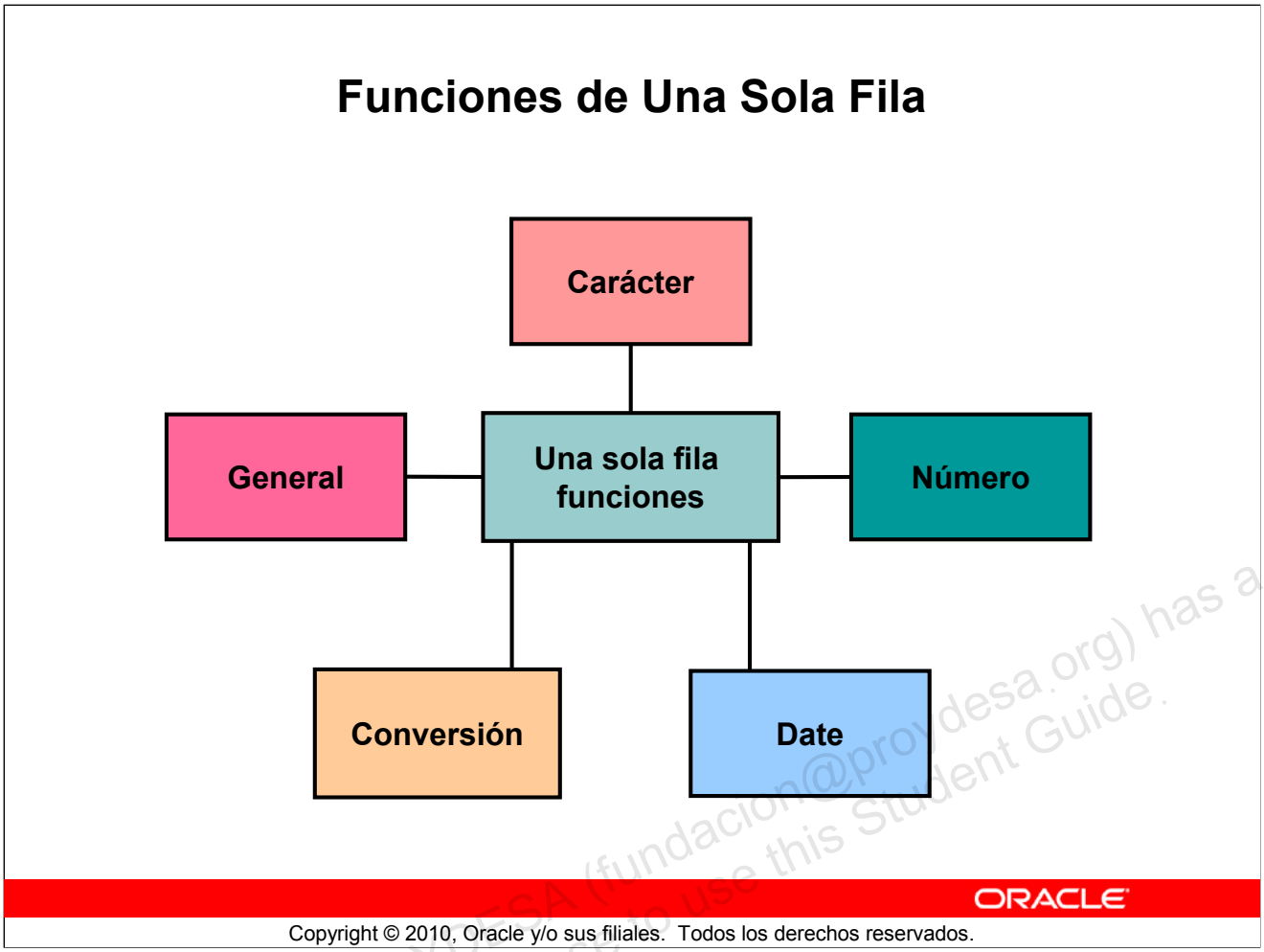
- Constante proporcionada por el usuario
- Valor de variable
- Nombre de columna
- Expresiones

Las características de las funciones de una sola fila son:

- Actuar en cada fila devuelta en la consulta
- Devolver un resultado por fila
- Posibilidad de devolver un valor de datos de un tipo diferente al que se hace referencia
- Posibilidad de esperar uno o más argumentos
- Se pueden utilizar en cláusulas `SELECT`, `WHERE` y `ORDER BY`; posibilidad de anidamiento

En la sintaxis:

`function_name`	Es el nombre de la función.
`arg1, arg2`	Es cualquier argumento que utilizará la función. Pueden estar representados por un nombre de columna o expresión.



Funciones de Una Sola Fila (continuación)

Esta lección trata las siguientes funciones de una sola fila:

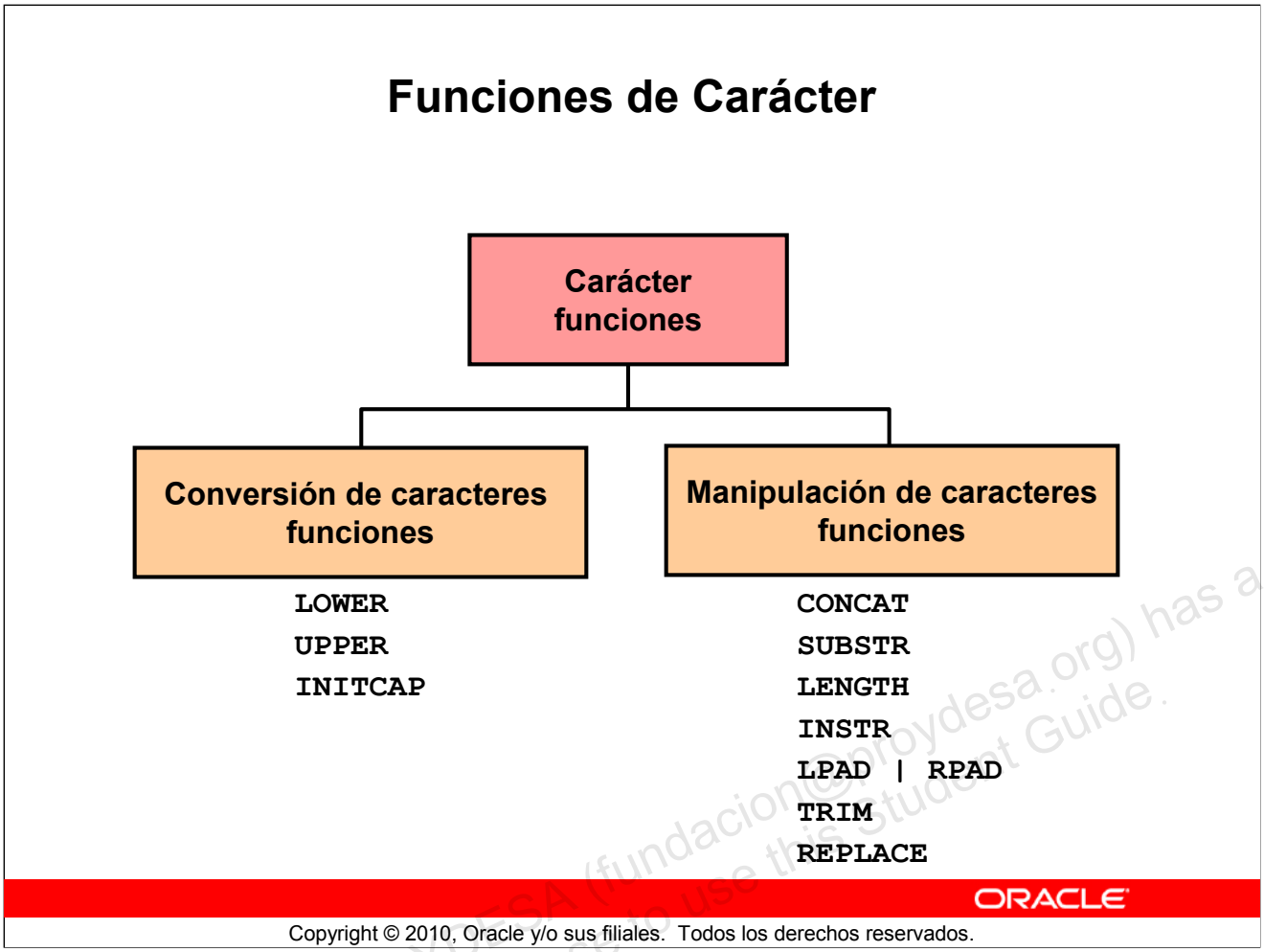
- **Funciones de carácter:** αceptan la entrada de caracteres y pueden devolver valores de número y de carácter.
- **Funciones numéricas:** aceptan valores de entrada y devuelven valores numéricos.
- **Funciones de fecha:** operan en valores del tipo de dato `DATE`. (Todas las funciones de fecha devuelven un valor de tipo de dato `DATE` excepto la función `MONTHS_BETWEEN`, que devuelve un número.)

Las siguientes funciones de una sola fila se tratan en las lecciones tituladas “Uso de Funciones de Conversión y Expresiones Condicionales”:

- **Funciones de conversión:** Convierten un valor de un tipo de dato a otro
- **Funciones generales:**
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
  - `CASE`
  - `DECODE`

Agenda

- Funciones SQL de una sola fila
- Funciones de carácter
- Funciones de número
- Trabajo con fechas
- Funciones de fecha



Funciones de Carácter

Las funciones de carácter de una sola fila aceptan los datos de caracteres como entrada y pueden devolver valores numéricos y de carácter. Las funciones de carácter se pueden dividir en:

- Funciones de conversión de caracteres
- Funciones de manipulación de caracteres

Función	Objetivo
LOWER(*column|expression*)	Convertir valores de carácter alfabético a minúsculas.
UPPER(*column|expression*)	Convertir valores de carácter alfabético a mayúsculas.
INITCAP(*column|expression*)	Convertir valores de carácter alfabético en mayúsculas; la primera letra de cada palabra y el resto en minúsculas.
CONCAT(*column1|expression1, column2|expression2*)	Concatenar el primer valor de carácter con el segundo valor de carácter; equivalente al operador de concatenación (||).
SUBSTR(*column|expression,m[,n]*)	Devolver los caracteres especificados de un valor de carácter que empieza por la posición de carácter *m*, de una longitud de *n* caracteres (si m es negativo, el recuento empieza desde el final del valor de carácter. Si se omite *n*, se devolverán todos los caracteres del final de la cadena).

Nota: las funciones tratadas en esta lección son sólo algunas de las funciones disponibles.

Funciones de Caracteres (continuación)

Función	Objetivo
LENGTH(*column*|*expression*)	Devolver el número de caracteres en la expression.
INSTR(*column*|*expression*, *'string', [,m], [n]*)	Devolver la posición numérica de una cadena denominada. Opcionalmente, puede proporcionar una posición *m* para iniciar la búsqueda y el *n* de incidencias de la cadena. *m* y *n* tienen un valor por defecto de 1, lo cual significa que se inicia la búsqueda al principio y se informa de la primera incidencia.
LPAD(*column*|*expression*, *n*, '*string*') RPAD(*column*|*expression*, *n*, '*string*')	Devolver una expresión con relleno a la izquierda de *n* caracteres con una expresión de caracteres. Devolver una expresión con relleno a la derecha de *n* caracteres con una expresión de caracteres.
TRIM(*leading*|*trailing*|*both*, *trim_character FROM trim_source*)	Permite recortar los caracteres finales o de encabezado (o ambos) de una cadena de caracteres. Si *trim_character* o *trim_source* es un literal de carácter, debe incluirlo entre comillas simples. Ésta es una función disponible en Oracle8*i* y versiones posteriores.
REPLACE(*text*, *search_string*, *replacement_string*)	Busca una expresión de texto para una cadena de caracteres y, si la encuentra, sustituirla por una cadena de sustitución especificada.

Nota: algunas de las funciones total o parcialmente compatibles con SQL:2003 son:

UPPER
LOWER
TRIM
LENGTH
SUBSTR
INSTR

Para obtener más información, consulte la sección sobre compatibilidad de Oracle con Core SQL:2003" en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g* .

Funciones de Conversión de Caracteres

Estas funciones convierten las mayúsculas/minúsculas para cadenas de caracteres:

Función	Resultado
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

Funciones de Conversión de Caracteres

LOWER, UPPER y INITCAP son las tres funciones de conversión de caracteres.

- LOWER: convierte las cadenas de caracteres en mayúsculas o en mayúsculas/minúsculas a minúsculas.
- UPPER: convierte las cadenas de caracteres en minúscula o en mayúsculas/minúsculas a mayúsculas.
- INITCAP: convierte la primera letra de cada palabra a mayúsculas y el resto de las letras a minúsculas.

```
SELECT 'The job id for '||UPPER(last_name)||' is '
 ||LOWER(job_id) AS "EMPLOYEE DETAILS"
FROM   employees;
```

	EMPLOYEE DETAILS
1	The job id for ABEL is sa_rep
2	The job id for DAVIES is st_clerk
3	The job id for DE HAAN is ad_vp
4	The job id for ERNST is it_prog
5	The job id for FAY is mk_rep
6	The job id for GIETZ is ac_account

...

Uso de Funciones de Conversión de Caracteres

Mostrar el número de empleado, nombre y número de departamento del empleado Higgins:

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
```

```
0 rows selected
```

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	205	Higgins	110

Uso de Funciones de Conversión de Caracteres

El ejemplo de la diapositiva muestra el número de empleado, nombre y número de departamento del empleado Higgins:

La cláusula `WHERE` de la primera sentencia SQL especifica el nombre del empleado como `higgins`. Debido a que todos los datos de la tabla `EMPLOYEES` están almacenados correctamente, el nombre `higgins` no encuentra ninguna coincidencia en la tabla y no se selecciona ninguna fila.

La cláusula `WHERE` de la segunda sentencia SQL especifica que el nombre del empleado de la tabla `EMPLOYEES` se compara con `higgins`, convirtiendo la columna `LAST_NAME` a minúsculas para poder compararla. Ya que ambos nombres no están en minúsculas, se ha encontrado una coincidencia y se ha seleccionado una fila. La cláusula `WHERE` se puede volver a escribir de la siguiente forma para que produzca el mismo resultado:

```
...WHERE last_name = 'Higgins'
```

El nombre de la salida aparece tal y como se almacenó en la base de datos. Para mostrar el nombre en mayúsculas, utilice la función `UPPER` de la sentencia `SELECT`.

```
SELECT employee_id, UPPER(last_name), department_id
FROM   employees
WHERE  INITCAP(last_name) = 'Higgins';
```

Funciones de Manipulación de Caracteres

Estas funciones manipulan las cadenas de caracteres:

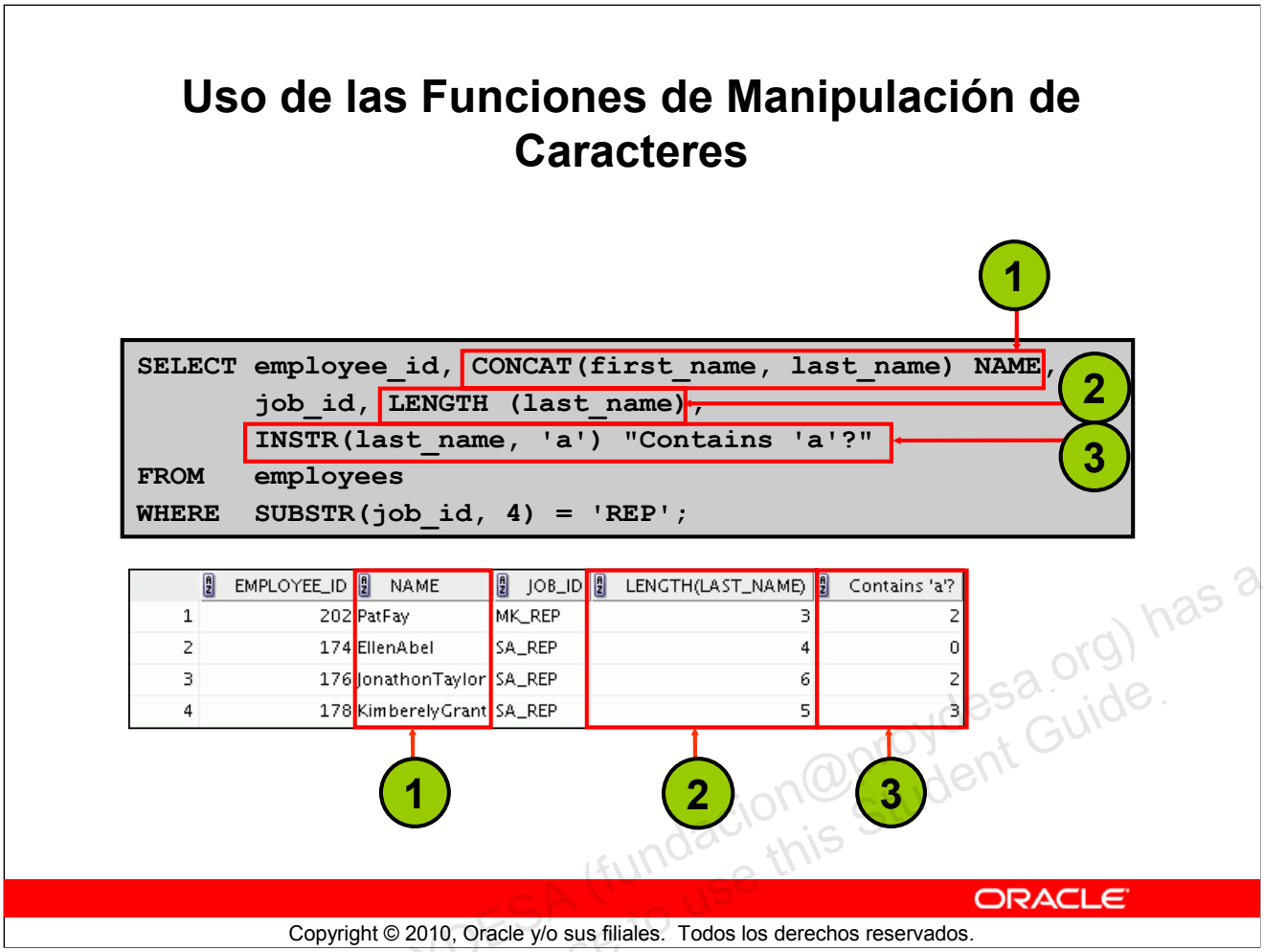
Función	Resultado
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
REPLACE ('JACK and JUE','J','BL')	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld

Funciones de Manipulación de Caracteres

CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD y TRIM son funciones de manipulación de caracteres que se tratan en esta lección.

- CONCAT: une los valores. (Sólo se pueden utilizar dos parámetros con CONCAT).
- SUBSTR: extrae una cadena de una longitud determinada
- LENGTH: muestra la longitud de una cadena como un valor numérico
- INSTR: obtiene la posición numérica de un carácter denominado
- LPAD: dDevuelve una expresión con relleno a la izquierda de caracteres *n* con una expresión de caracteres
- RPAD: devuelve una expresión con relleno a la derecha de caracteres *n* con una expresión de caracteres
- TRIM: recorta los caracteres finales o de encabezado (o ambos) de una cadena de caracteres (si *trim_character* o *trim_source* es un carácter literal, debe incluirlo entre comillas simples)

Nota: puede utilizar funciones como UPPER y LOWER con una sustitución con ampersand. Por ejemplo, utilice UPPER('&job_title') para que el usuario no tenga que introducir el puesto en un formato específico.



Uso de las Funciones de Manipulación de Caracteres

El ejemplo de la diapositiva muestra los nombres y apellidos de los empleados que se han unido, la longitud del apellido del empleado y la posición numérica de la letra “a” en el apellido del empleado de todos los empleados que tienen la cadena `REP` incluida en el ID de trabajo que empieza en la cuarta posición de dicho ID.

Ejemplo:

Modifique la sentencia SQL de la diapositiva para mostrar los datos de los empleados cuyos apellidos acaben con la letra “n”.

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
LENGTH (last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(last_name, -1, 1) = 'n';
```

	EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
1	102	LexDe Haan	7	5
2	200	JenniferWhalen	6	3
3	201	MichaelHartstein	9	2

Agenda

- Funciones SQL de una sola fila
- Funciones de carácter
- Funciones de número
- Trabajo con fechas
- Funciones de fecha

Funciones Numéricas

- ROUND: redondea el valor a un decimal especificado
- TRUNC: trunca el valor a un decimal especificado
- MOD: devuelve el resto de la división

Función	Resultado
ROUND(45.926, 2)	45.93
TRUNC(45.926, 2)	45.92
MOD(1600, 300)	100

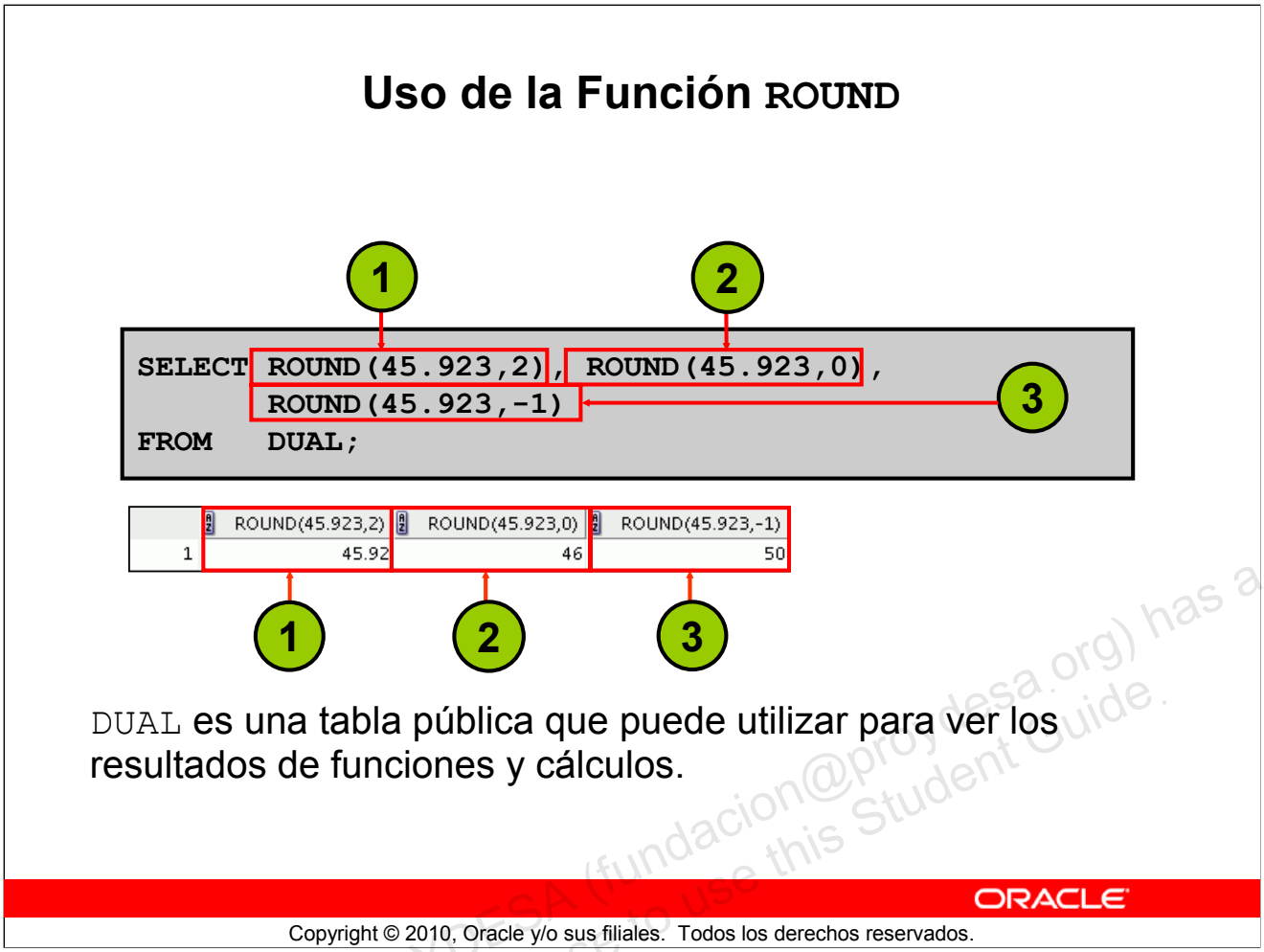
Funciones Numéricas

Las funciones numéricas aceptan entradas numéricas y devuelven valores numéricos. Esta sección describe algunas de las funciones numéricas.

Función	Objetivo
ROUND(*column|expression, n*)	Redondear la columna, expresión o valor a n decimales o, si se omite *n*, a ningún decimal (Si *n* es negativo, se redondearán los números a la izquierda del decimal.)
TRUNC(*column|expression, n*)	Truncar la columna, expresión o valor a *n* decimales o, si *n* se omite, el valor por defecto de *n* es cero
MOD(*m*,*n*)	Devolver el resto de *m* dividido entre *n*

Nota: esta lista contiene sólo algunas de las funciones numéricas disponibles.

Para obtener más información, consulte la sección sobre funciones numéricas en la guía *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.



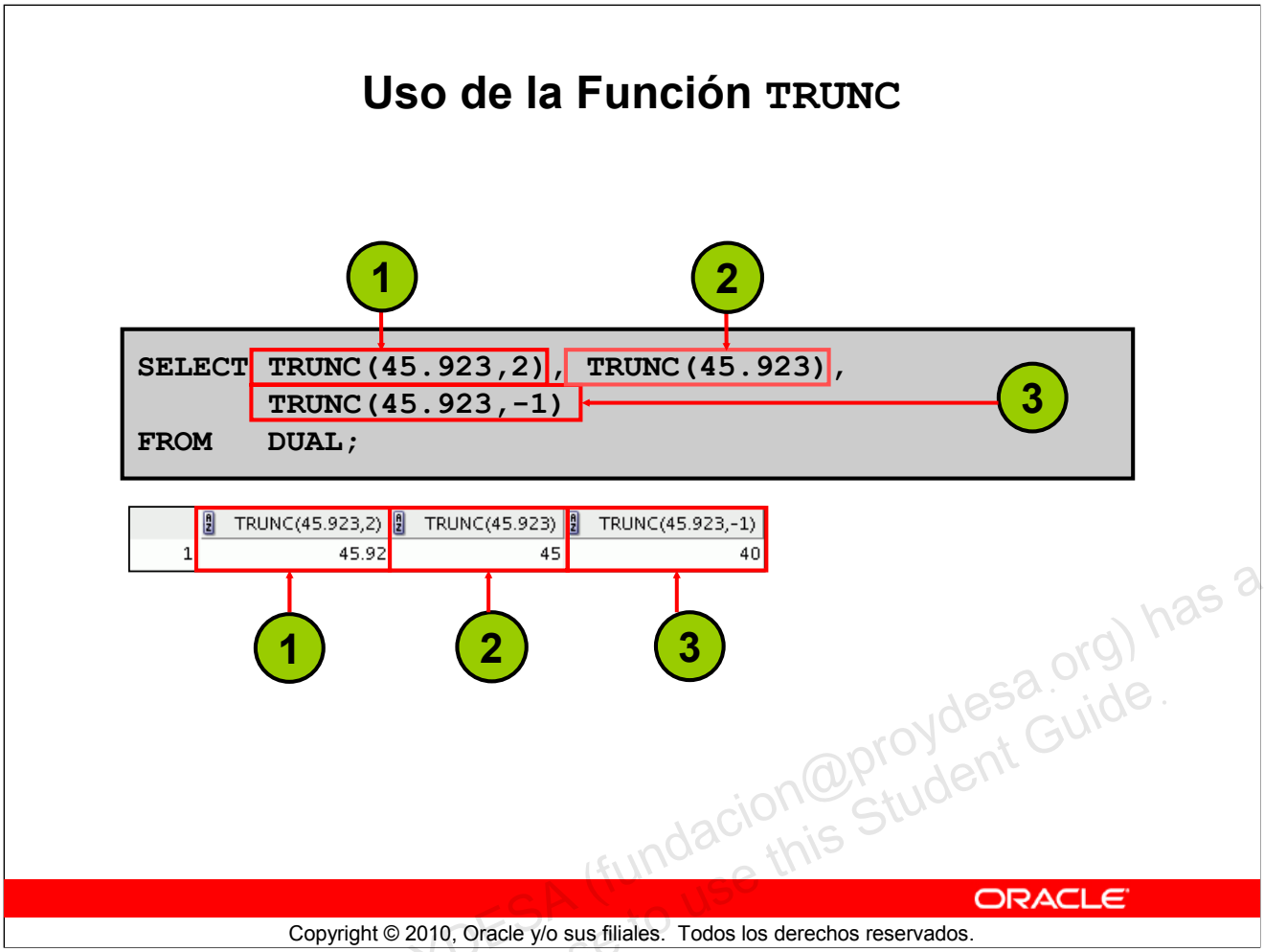
Uso de la Función ROUND

La función ROUND redondea la columna, expresión o valor a *n* decimales. Si falta el segundo argumento o tiene un valor de 0, el valor se redondea a cero decimales. Si el segundo argumento tiene un valor de 2, el valor se redondea a dos decimales. Por el contrario, si el segundo argumento es –2, el valor se redondea a dos decimales a la izquierda (redondeados a la unidad más cercana a 100).

La función ROUND también se puede utilizar con las funciones de fecha. Verá varios ejemplos más adelante en esta lección.

Tabla **DUAL**

El usuario SYS es el propietario de la tabla DUAL, a la que pueden acceder todos los usuarios. Contiene una columna, DUMMY, y una fila con el valor X. La tabla DUAL es útil cuando sólo desea devolver un valor una vez (por ejemplo, el valor de una constante, pseudocolumna o expresión que no se deriva de una tabla con datos de usuario). La tabla DUAL se utiliza generalmente para obtener una visión más completa de la sintaxis de la cláusula SELECT, porque las cláusulas SELECT y FROM son obligatorias y muchos cálculos no tienen que realizar selecciones en las tablas reales.



Uso de la Función TRUNC

La función TRUNC trunca la columna, expresión o valor a *n* decimales.

La función TRUNC funciona con argumentos similares a los de la función ROUND. Si falta el segundo argumento o tiene un valor de 0, el valor se trunca a cero decimales. Si el segundo argumento tiene un valor de 2, el valor se trunca a dos decimales. Por el contrario, si el segundo argumento tiene un valor de –2, el valor se trunca a dos decimales a la izquierda. Si el segundo argumento tiene un valor de –1, el valor se trunca a un decimal a la izquierda.

Al igual que la función ROUND, la función TRUNC se puede utilizar con funciones de fecha.

Uso de la Función MOD

Para todos los empleados con un puesto de vendedor, calcular el resto del salario después de dividirlo entre 5.000.

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

	LAST_NAME	SALARY	MOD(SALARY,5000)
1	Abel	11000	1000
2	Taylor	8600	3600
3	Grant	7000	2000

ORACLE

Uso de la Función MOD

La función MOD obtiene el resto del primer argumento dividido entre el segundo argumento. El ejemplo de la diapositiva calcula el resto del salario después de dividirlo entre 5.000 de todos los empleados cuyo ID de trabajo sea SA_REP.

Nota: la función MOD se suele utilizar para determinar si un valor es par o impar. La función MOD es también la función hash de Oracle.

Agenda

- Funciones SQL de una sola fila
- Funciones de carácter
- Funciones de número
- Trabajo con fechas
- Funciones de fecha

Trabajo con Fechas

- Oracle Database almacena fechas en un formato numérico interno: siglo, año, mes, día, horas, minutos y segundos.
- El formato de visualización de la fecha por defecto es DD-MON-RR.
  - Permite almacenar fechas del siglo 21 en el siglo 20 especificando sólo los dos últimos dígitos del año
  - De la misma forma, permite almacenar fechas del siglo 20 en el siglo 21

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date < '01-FEB-88';
```

	LAST_NAME	HIRE_DATE
1	Whalen	17-SEP-87
2	King	17-JUN-87

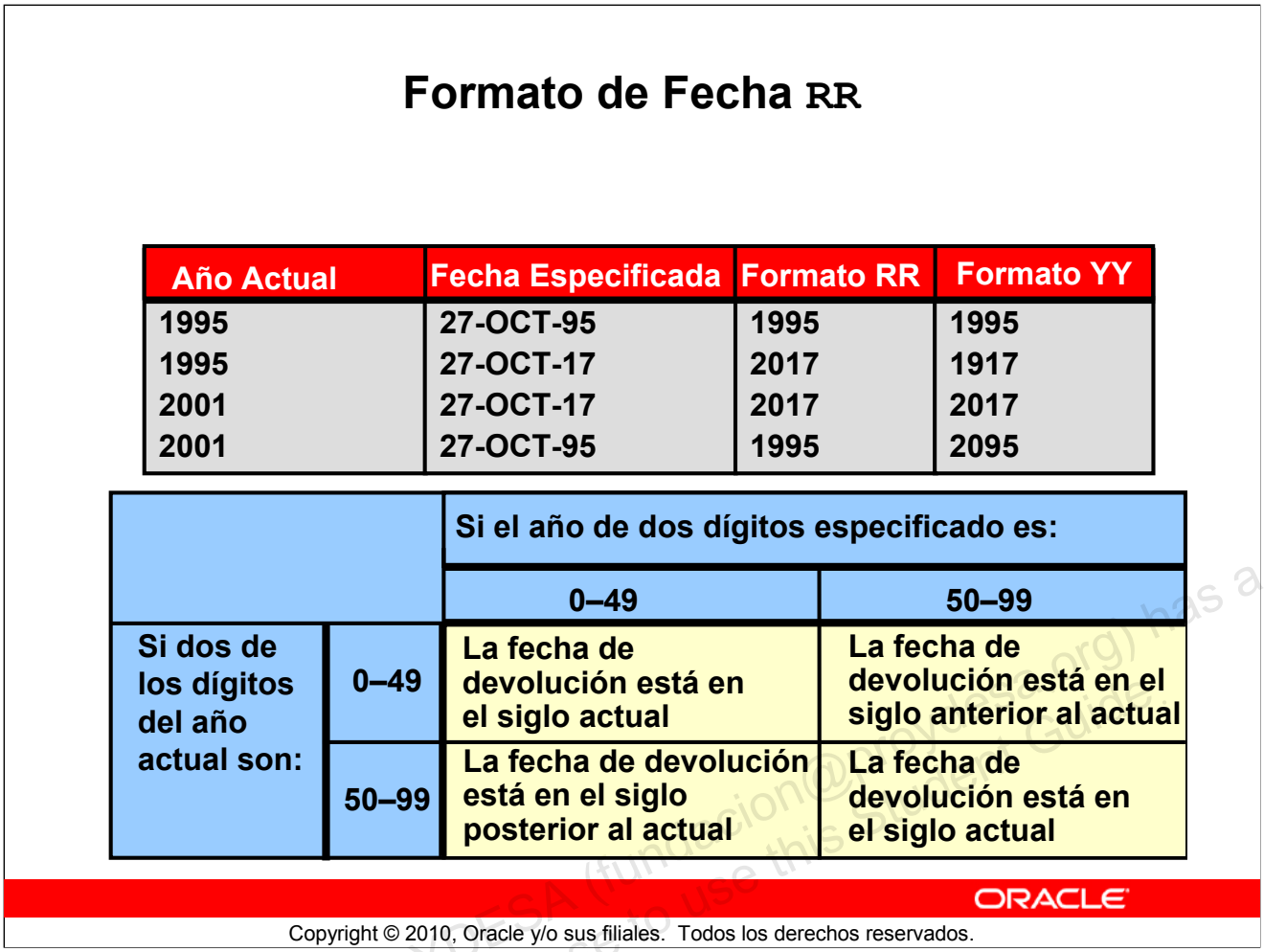
ORACLE

Trabajo con Fechas

Oracle Database almacena fechas en un formato numérico interno que representa el siglo, el año, el mes, las horas, los minutos y los segundos.

La visualización por defecto y el formato de entrada de cualquier fecha es DD-MON-RR. Las fechas de Oracle válidas son del 1 de enero de 4712 A.C. y el 31 de diciembre de 9999 D.C.

En el ejemplo de la diapositiva, la salida de la columna `HIRE_DATE` aparece en el formato por defecto DD-MON-RR. Sin embargo, las fechas no se almacenan en la base de datos en este formato. Se almacenan todos los componentes de la fecha y la hora. Por lo tanto, aunque un valor `HIRE_DATE` de 17-JUN-87 aparezca como el día, mes y año, también existe información de *hora* y *siglo* asociada a la fecha. Los datos completos podrían ser 17 de junio de 1987, 5:10:43 PM.



Formato de Fecha RR

El formato de fecha `RR` es similar al elemento `YY`, pero puede utilizarlo para especificar siglos diferentes. Utilice el elemento de formato de fecha `RR` en lugar de `YY` para que el siglo del valor de retorno varíe según el año de dos dígitos especificado y los últimos dos dígitos del año actual. La tabla de la diapositiva resume el comportamiento del elemento `RR`.

Año Actual	Fecha Proporcionada	Interpretada (RR)	Interpretada (YY)
1994	27-OCT-95	1995	1995
1994	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2048	27-OCT-52	1952	2052
2051	27-OCT-47	2147	2047

Observe que los valores mostrados en las dos últimas filas de la tabla anterior. Conforme nos acercamos a la mitad del siglo, el comportamiento de RR puede que no sea lo que desea.

Formato de Fecha RR (continuación)

Estos datos se almacenan internamente de la siguiente forma:

CENTURY	YEAR	MONTH	DAY	HOUR	MINUTE	SECOND
19	87	06	17	17	10	43

Siglos y el Año 2000

Cuando se inserta un registro con una columna de fecha en una tabla, la información de *siglo* se selecciona de la función SYSDATE. Sin embargo, cuando la columna de fecha se muestra en la pantalla, el componente de siglo no aparece (por defecto).

El tipo de dato DATE utiliza 2 bytes para la información de año, uno para el siglo y otro para el año. El valor de siglo siempre se incluye, independientemente de si se especifica o se muestra. En este caso, RR determina el valor por defecto para el siglo en INSERT.

Uso de la Función SYSDATE

SYSDATE es una función que devuelve:

- Fecha
- Hora

```
SELECT sysdate
FROM   dual;
```

	SYSDATE
1	10-JUN-09

ORACLE

Uso de la Función SYSDATE

SYSDATE es una función de fecha que devuelve la fecha y hora actuales del servidor de base de datos. Puede utilizar SYSDATE como si utilizara cualquier otro nombre de columna. Por ejemplo, puede mostrar la fecha actual seleccionando SYSDATE de una tabla. Es muy común seleccionar SYSDATE de una tabla ficticia denominada DUAL.

Nota: SYSDATE devuelve la fecha y hora actuales definidas para el sistema operativo en el que reside la base de datos. Por lo tanto, si está en algún lugar de Australia y se conecta a una base de datos remota en una ubicación de Estados Unidos (EE. UU.), la función sysdate devolverá la fecha y hora de EE. UU. En ese caso, puede utilizar la función CURRENT_DATE que devuelve la fecha actual en la zona horaria de la sesión.

La función CURRENT_DATE y otras funciones de zona horaria relacionadas se abordan con mayor detalles en *Oracle Database: Conceptos Fundamentales de SQL II*.

Operadores Aritméticos con Fechas

- Sumar o restar un número de una fecha para obtener un valor de fecha resultante.
- Restar dos fechas para obtener el número de días entre esas fechas.
- Agregar horas a una fecha dividendo entre el número de horas entre 24.

Operadores Aritméticos con Fechas

Debido a que la base de datos almacena fechas como números, puede realizar cálculos utilizando operadores aritméticos como la suma y la resta. Puede agregar y restar constantes numéricas y fechas.

Puede realizar las siguientes operaciones:

Operación	Resultado	Descripción
fecha + número	Fecha	Suma un número de días a una fecha
fecha – número	Fecha	Resta un número de días de una fecha
fecha – fecha	Número de días	Resta una fecha de otra
fecha + número/24	Fecha	Suma un número de horas a una fecha

Uso de Operadores Aritméticos con Fechas

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS
FROM   employees
WHERE  department_id = 90;
```

	LAST_NAME	WEEKS
1	King	1147.102432208994708994708994708994708995
2	Kochhar	1028.959575066137566137566137566137566138
3	De Haan	856.102432208994708994708994708994708995

Uso de Operadores Aritméticos con Fechas

El ejemplo de la diapositiva muestra el apellido y el número de semanas durante las que han trabajado todos los empleados del departamento 90. Resta la fecha de contratación del empleado de la fecha actual (`SYSDATE`) y divide el resultado entre 7 para calcular el número de semanas durante las que ha trabajado el empleado.

Nota: `SYSDATE` es una función SQL que devuelve la fecha y hora actuales. Los resultados pueden ser diferentes según la fecha y hora definidas para el sistema operativo de la base de datos local al ejecutar la consulta SQL.

Si se resta una fecha más actual de una fecha más antigua, la diferencia es un número negativo.

Agenda

- Funciones SQL de una sola fila
- Funciones de carácter
- Funciones de número
- Trabajo con fechas
- Funciones de fecha

Funciones de Manipulación de Fecha

Función	Resultado
MONTHS_BETWEEN	Número de meses entre dos fechas
ADD_MONTHS	Agregar meses de calendario a fecha
NEXT_DAY	Siguiente día de la fecha especificada
LAST_DAY	Último día del mes
ROUND	Redondear fecha
TRUNC	Truncar fecha

ORACLE

Funciones de Manipulación de Fecha

Las funciones de fecha funcionan en fechas de Oracle. Todas las funciones de fecha devuelven un valor del tipo de dato DATE excepto MONTHS_BETWEEN, que devuelve un valor numérico.

- MONTHS_BETWEEN(*date1, date2*): obtiene el número de meses entre *date1* y *date2*. El resultado puede ser positivo o negativo. Si *date1* es posterior a *date2*, el resultado es positivo; si *date1* es anterior a *date2*, el resultado es negativo. La parte del resultado que no sea un entero representa una parte del mes.
- ADD_MONTHS(*date, n*): agrega el número *n* de los meses de calendario a *date*. El valor de *n* debe ser un entero y puede ser negativo.
- NEXT_DAY(*date, 'char'*): obtiene la fecha del siguiente día de la semana especificado ('*char*') que le sigue a *date*. El valor de *char* puede ser un número que representa un día o una cadena de caracteres.
- LAST_DAY(*date*): obtiene la fecha del último día del mes que contiene *date*.

Esta lista es un subjuego de las funciones de fecha disponibles. Las funciones de número ROUND y TRUNC también se pueden utilizar para manipular los valores de fecha como se muestra a continuación:

- ROUND(*date*[,'*fmt*']): devuelve *date* redondeado a la unidad especificada por el modelo de formato *fmt*. Si se omite el modelo de formato *fmt* , *date* se redondea a la fecha más cercana.
- TRUNC(*date*[, '*fmt*']): devuelve *date* con la parte de la hora del día truncada a la unidad especificada por el modelo de formato *fmt*. Si se omite el modelo de formato *fmt*,*date* se redondea a la fecha más cercana.

Los siguientes modelos de formato se tratan en las lecciones tituladas “Uso de Funciones de Conversión y Expresiones Condicionales”:

Uso de las Funciones de Fecha

Función	Resultado
MONTHS_BETWEEN ('01-SEP-95','11-JAN-94')	19.6774194
ADD_MONTHS ('31-JAN-96',1)	'29-FEB-96'
NEXT_DAY ('01-SEP-95','FRIDAY')	'08-SEP-95'
LAST_DAY ('01-FEB-95')	'28-FEB-95'

ORACLE

Uso de las Funciones de Fecha

En el ejemplo de la diapositiva, la función ADD_MONTHS agrega un mes al valor de fecha proporcionado “31-JAN-96” y devuelve “29-FEB-96”. La función reorganiza el año 1996 como año bisiesto y, por lo tanto, devuelve la última fecha del mes de febrero. Si cambia el valor de fecha de entrada a “31-JAN-95”, la función devuelve “28-FEB-95”.

Por ejemplo, muestre el número de empleado, fecha de contratación, número de meses durante los que ha trabajado, fecha de revisión de seis meses, primer viernes después de la fecha de contratación y último día del mes de contratación de todos los empleados que han trabajado menos de 150 meses.

```
SELECT employee_id, hire_date, MONTHS_BETWEEN (SYSDATE, hire_date)
TENURE, ADD_MONTHS (hire_date, 6) REVIEW,  NEXT_DAY (hire_date,
'FRIDAY'), LAST_DAY(hire_date)
FROM   employees WHERE  MONTHS_BETWEEN (SYSDATE, hire_date) < 150;
```

	EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DA...	LAST_DAY...
1	202	17-AUG-97	141.79757989...	17-FEB-98	22-AUG-97	31-AUG-97
2	107	07-FEB-99	124.12016054...	07-AUG-99	12-FEB-99	28-FEB-99
3	124	16-NOV-99	114.82983796...	16-MAY-00	19-NOV-99	30-NOV-99
4	142	29-JAN-97	148.41048312...	29-JUL-97	31-JAN-97	31-JAN-97
5	143	15-MAR-98	134.86209602...	15-SEP-98	20-MAR-98	31-MAR-98
6	144	09-JUL-98	131.05564441...	09-JAN-99	10-JUL-98	31-JUL-98
7	149	29-JAN-00	112.41048312...	29-JUL-00	04-FEB-00	31-JAN-00
8	176	24-MAR-98	134.57177344...	24-SEP-98	27-MAR-98	31-MAR-98
9	178	24-MAY-99	120.57177344...	24-NOV-99	28-MAY-99	31-MAY-99

Uso de las Funciones ROUND y TRUNC con Fechas

Supongamos SYSDATE = '25-JUL-03':

Función	Resultado
ROUND(SYSDATE,'MONTH')	01-AUG-03
ROUND(SYSDATE ,'YEAR')	01-JAN-04
TRUNC(SYSDATE ,'MONTH')	01-JUL-03
TRUNC(SYSDATE ,'YEAR')	01-JAN-03

ORACLE

Uso de las Funciones ROUND y TRUNC con Fechas

Las funciones ROUND y TRUNC se pueden utilizar para valores de fecha y número. Al utilizarlas con fechas, estas funciones redondean o truncan el modelo de formato especificado. Por lo tanto, puede redondear las fechas al año o mes más cercano. Si el modelo de formato es mes, el resultado de las fechas 1-15 es el primer día del mes actual. El resultado de las fechas 16-31 es el primer día del siguiente mes. Si el modelo de formato es mes, el resultado de las fechas 1-6 es el 1 de enero del año actual. El resultado de los meses 7-12 es el 1 de enero del siguiente año.

Ejemplo:

Compare las fechas de contratación de todos los empleados que empezaron en 1997. Muestre el número de empleado, fecha de contratación y mes de inicio con las funciones ROUND y TRUNC.

```
SELECT employee_id, hire_date,
 ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')
FROM   employees
WHERE  hire_date LIKE '%97';
```

	EMPLOYEE_ID	HIRE_DATE	ROUND(HIRE_DATE,'MONTH')	TRUNC(HIRE_DATE,'MONTH')
1	202	17-AUG-97	01-SEP-97	01-AUG-97
2	142	29-JAN-97	01-FEB-97	01-JAN-97

Prueba

¿Cuáles de las siguientes afirmaciones sobre funciones de una sola fila son ciertas?

1. Manipular elementos de datos
2. Aceptar argumentos y devolver un valor por argumento
3. Actuar en cada fila devuelta
4. Devuelve un resultado por juego de filas
5. Posibilidad de modificar el tipo de dato
6. Posibilidad de anidamiento
7. Aceptar argumentos que pueden ser una columna o una expresión

ORACLE

Respuesta: 1, 3, 6, 7

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Realizar cálculos de datos utilizando funciones
- Modificar elementos de datos individuales utilizando funciones

ORACLE

Resumen

Las funciones de una sola fila se pueden anidar en cualquier nivel. Las funciones de una sola fila pueden manipular los siguientes elementos:

- **Datos de caracteres:** LOWER, UPPER, INITCAP, CONCAT, SUBSTR, INSTR, LENGTH
- **Datos de número:** ROUND, TRUNC, MOD
- **Valores de fecha:** SYSDATE, MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY

Recuerde lo siguiente:

- Los valores de fecha también pueden utilizar operadores aritméticos.
- Las funciones ROUND y TRUNC también se pueden utilizar con valores de fecha.

SYSDATE y DUAL

SYSDATE es una función de fecha que devuelve la fecha y hora actuales. Es muy común seleccionar SYSDATE de una tabla pública denominada DUAL.

Práctica 3: Visión General

En esta práctica se abordan los siguientes temas:

- Escritura de una consulta que muestre la fecha actual
- Creación de consultas que requieran el uso de funciones numéricas, de carácter y de fecha
- Realización de cálculos de años y meses de servicio de un empleado

ORACLE

Práctica 3: Visión General

Esta práctica proporciona una variedad de ejercicios que utilizan diferentes funciones que están disponibles para tipos de dato de carácter, de número y de fecha.

4

Uso de Funciones de Conversión y Expresiones Condicionales

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Describir varios tipos de funciones de conversión que están disponibles en SQL
- Utilizar las funciones de conversión `TO_CHAR`, `TO_NUMBER` y `TO_DATE`
- Aplicar expresiones condicionales en una sentencia `SELECT`

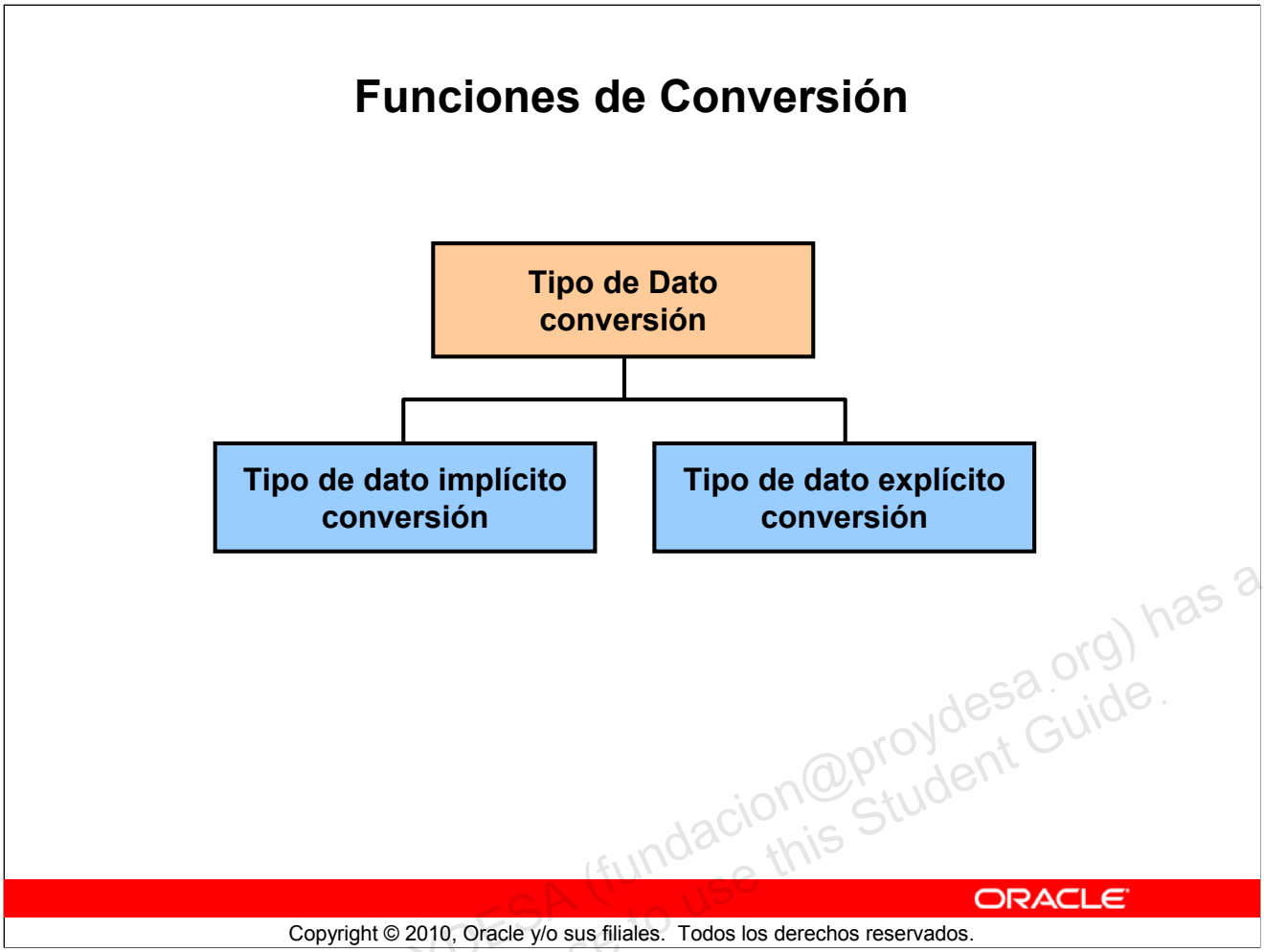
ORACLE

Objetivos

Esta lección se centra en funciones que convierten los datos de un tipo en otro (por ejemplo, conversión de datos de caracteres en datos numéricos) y describe las expresiones condicionales en sentencias SQL `SELECT`.

Agenda

- Conversión de tipo de dato implícito y explícito
- Funciones `TO_CHAR`, `TO_DATE`, `TO_NUMBER`
- Funciones de anidación
- Funciones generales:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Expresiones condicionales:
  - `CASE`
  - `DECODE`



Funciones de Conversión

Además de los tipos de dato de Oracle, las columnas de las tablas de una Oracle Database se pueden definir utilizando los tipos de dato American National Standards Institute (ANSI), DB2 y SQL/DS. Sin embargo, el servidor de Oracle convierte internamente esos tipos de dato a tipos de dato de Oracle.

En algunos casos, el servidor de Oracle recibe datos de un tipo de dato cuando espera datos de un tipo de dato diferente. Cuando esto ocurre, el servidor de Oracle puede convertir automáticamente los datos al tipo de dato esperado. Esta conversión de tipo de dato puede realizarla el servidor de Oracle *implícitamente* o el usuario *explícitamente*.

Las conversiones de tipo de dato funcionan según las reglas explicadas en las siguientes diapositivas.

Las conversiones de tipo de dato explícitas se realizan utilizando las funciones de conversión. Las funciones de conversión sirven para convertir los valores de un tipo de dato a otro. Generalmente, el formato de los nombres de funciones sigue la convención *data type* `TO` *data type*. El primer tipo de dato es el tipo de dato de entrada; el segundo tipo de dato es la salida.

Nota: aunque la conversión de tipo de dato implícita está disponible, se recomienda que realice una conversión de tipo de dato explícita para asegurar la fiabilidad de las sentencias SQL.

Conversión Implícita del Tipo de Dato

En expresiones, el servidor de Oracle puede convertir automáticamente:

A	De
VARCHAR2 o CHAR	NUMBER
VARCHAR2 o CHAR	DATE

Conversión Implícita del Tipo de Dato

El servidor de Oracle puede realizar automáticamente una conversión de tipo de dato en una expresión. Por ejemplo, la expresión `hire_date > '01-JAN-90'` tiene como resultado la conversión implícita de la cadena `'01-JAN-90'` a una fecha. Por lo tanto, un valor `VARCHAR2` o `CHAR` se puede convertir de forma implícita a un tipo de dato de fecha o número en una expresión

Conversión Implícita del Tipo de Dato

Para la evaluación de expresiones, el servidor de Oracle puede convertir automáticamente:

A	De
NUMBER	VARCHAR2 o CHAR
DATE	VARCHAR2 o CHAR

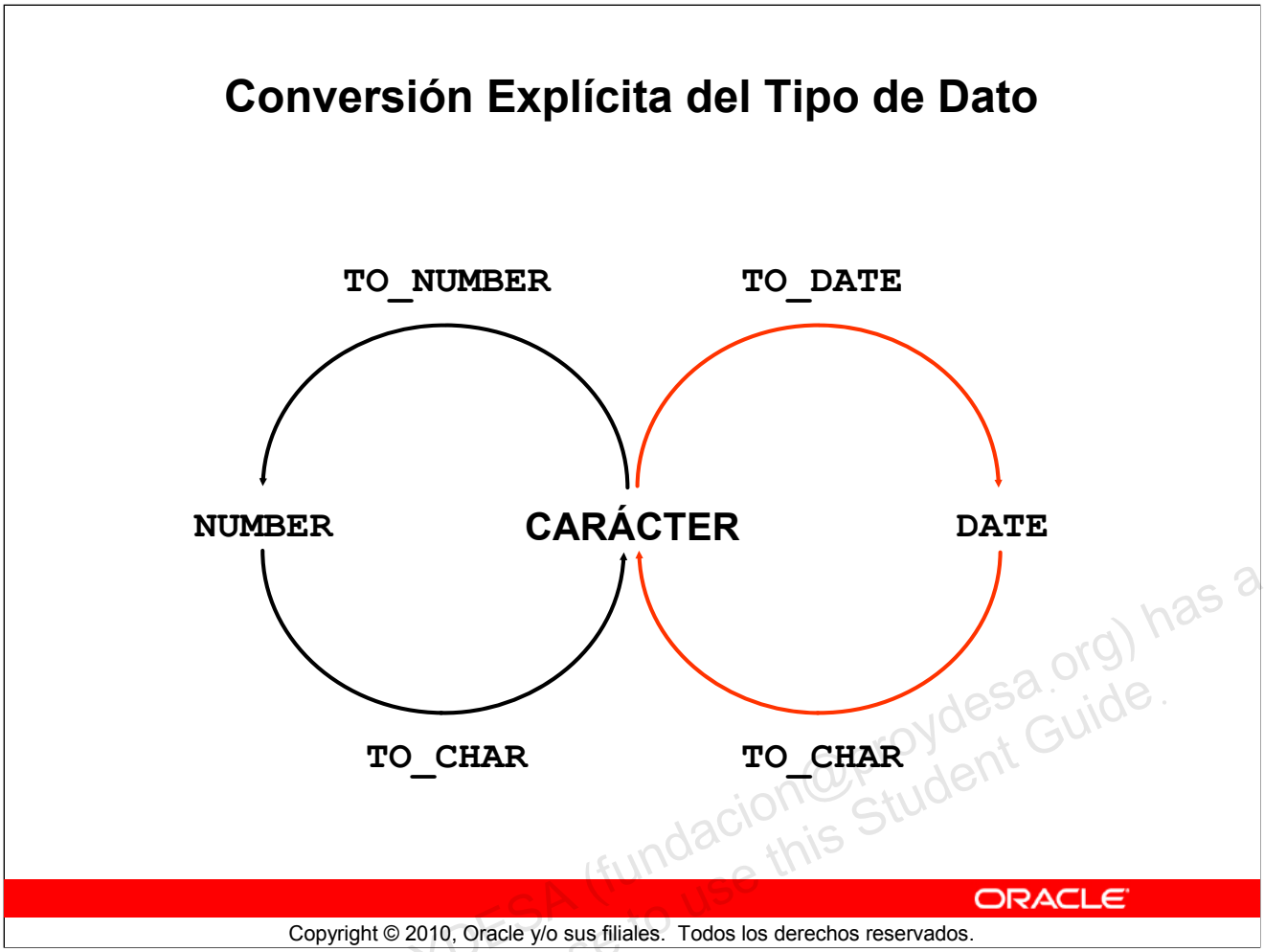
ORACLE

Conversión Implícita del Tipo de Dato (continuación)

En general, el servidor de Oracle utiliza la regla de expresiones cuando se necesita una conversión de tipo de dato. Por ejemplo, la expresión `grade = 2` tiene como resultado la conversión implícita del número `2` a la cadena "2" porque el grado es una columna `CHAR(2)`..

Nota: las conversiones de `CHAR` a `NUMBER` se realizan correctamente sólo si la cadena de caracteres representa un número válido.

Conversión Explícita del Tipo de Dato

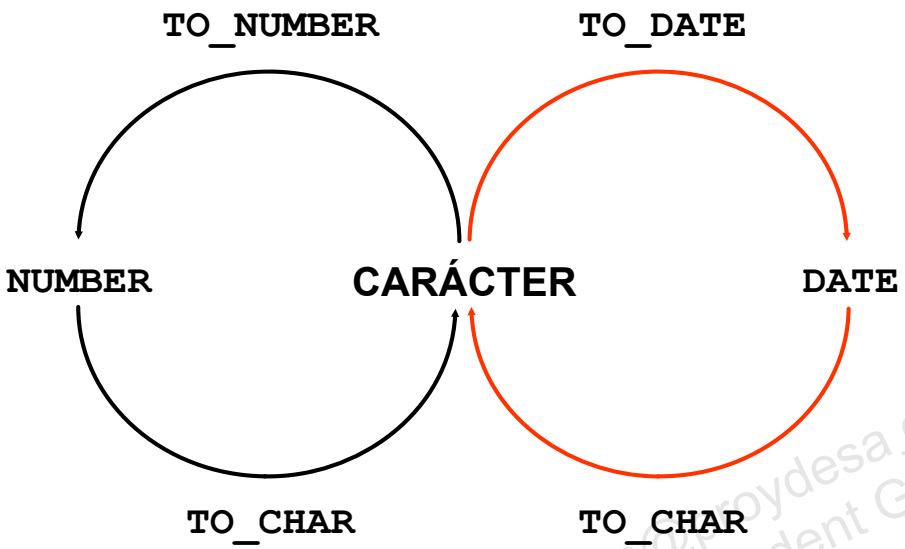


Conversión Explícita del Tipo de Dato

SQL proporciona tres funciones para convertir un valor de un tipo de dato a otro:

Función	Objetivo
`TO_CHAR(number|date,[ fmt],[nlsparams])`	Convertir un valor de fecha o número a una cadena de caracteres `VARCHAR2` con el modelo de formato *fmt* **Conversión numérica:** el parámetro `nlsparams` especifica los siguientes caracteres, devueltos por los elementos de formato numérico: • Carácter decimal • Separador de grupo • Símbolo de la divisa local • Símbolo de la divisa internacional Si se omite el parámetro `nlsparams` o cualquier otro parámetro, esta función utiliza para la sesión los valores de parámetro por defecto.

Conversión Explícita del Tipo de Dato



Conversión Explícita del Tipo de Dato (continuación)

Función	Objetivo
`TO_CHAR(number|date,[fmt], [nlsparams])`	**Conversión de fecha:** el parámetro `nlsparams` especifica el idioma en el que se devuelven las abreviaturas y los nombres del mes y del día. Si se omite este parámetro, esta función utiliza los idiomas de fecha por defecto para la sesión.
`TO_NUMBER(char,[fmt], [nlsparams])`	Convertir una cadena de caracteres que contiene dígitos a un número en el formato especificado por el modelo de formato opcional *fmt*. El parámetro `nlsparams` tiene el mismo objetivo en esta función que en la función `TO_CHAR` para la conversión numérica.
`TO_DATE(char,[fmt],[nlsparams])`	Convertir una cadena de caracteres que representa una fecha en un valor de fecha según el valor de *fmt* especificado. Si se omite *fmt*, el formato es DD-MON-YY. El parámetro `nlsparams` tiene el mismo objetivo en esta función que en la función `TO_CHAR` para la conversión de fecha.

Conversión Explícita del Tipo de Dato (continuación)

Nota: la lista de funciones mencionadas en esta lección incluye sólo algunas de las funciones de conversión disponibles.

Para obtener más información, consulte la sección sobre funciones de conversión en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Agenda

- Conversión de tipo de dato implícito y explícito
- Funciones `TO_CHAR`, `TO_DATE`, `TO_NUMBER`
- Funciones de anidación
- Funciones generales:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Expresiones condicionales:
  - `CASE`
  - `DECODE`

Uso de la Función TO_CHAR con Fechas

```
TO_CHAR(date, 'format_model')
```

El modelo de formato:

- Debe estar entre comillas simples
- Es sensible a mayúsculas/minúsculas
- Puede incluir cualquier elemento de formato de fecha válido
- Tiene un elemento `fm` para eliminar los espacios en blanco o suprimir ceros iniciales
- Está separado del valor de fecha por una coma

Uso de la Función TO_CHAR con Fechas

`TO_CHAR` convierte un tipo de dato de fecha y hora a un valor de tipo de dato `VARCHAR2` con el formato especificado por*format_model*. Un modelo de formato es un carácter literal que describe el formato de fecha y hora almacenado en una cadena de caracteres. Por ejemplo, el modelo de formato de fecha y hora `'11-Nov-1999'` es `'DD-Mon-YYYY'`. Puede utilizar la función `TO_CHAR` para convertir una fecha de su formato por defecto a uno que especifique.

Instrucciones

- El modelo de formato debe estar entre comillas simples y es sensible a mayúsculas/minúsculas.
- El modelo de formato puede incluir cualquier elemento de formato de fecha válido. Asegúrese de separar el valor de fecha del modelo de formato con una coma.
- Los nombres de los días y meses en la salida se rellenan automáticamente con espacios en blanco.
- Para eliminar los espacios en blanco o suprimir los ceros iniciales, utilice el elemento *fm* de modo de relleno.

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired
FROM   employees
WHERE  last_name = 'Higgins';
```

	EMPLOYEE_ID	MONTH_HIRED
1	205	06/94

Elementos del Modelo de Formato de Fecha

Elemento	Resultado
YYYY	Año completo en números
YEAR	Año en letra (en inglés)
MM	Valor de dos dígitos del mes
MONTH	Nombre completo del mes
MON	Abreviatura de tres letras del mes
DY	Abreviatura de tres letras del día de la semana
DAY	Nombre completo del día de la semana
DD	Día numérico del mes

ORACLE

Elementos de Formato de Ejemplo de Formatos de Fecha Válidos

Elemento	Descripción
SCC o bien CC	Siglo; el servidor agrega como prefijo el signo - a los años A.C.
Años en fechas YYYY o bien SYYYY	Año; el servidor agrega como prefijo el signo - a los años A.C.
YYY o bien YY o bien Y	Indica el último, los tres últimos o los dos últimos dígitos de un año
Y,YYY	Año con una coma en esta posición
IYYY, IYY, IY, I	Año de cuatro, tres, dos o un dígito basado en el estándar ISO
SYEAR o bien YEAR	Año en letra; el servidor agrega como prefijo el signo - a los años A.C.
BC o bien AD	Indica un año A.C. o D.C.
B.C. o bien A.D.	Indica un año A.C. o D.C. que utiliza períodos
Q	Trimestre del año
MM	Mes: valor de dos dígitos
MONTH	Nombre del mes rellenado con espacios en blanco hasta una longitud de nueve caracteres
MON	Nombre del mes, abreviatura de tres letras
RM	Número romano del mes
WW o bien W	Semana del año o del mes
DDD o bien DD o bien D	Día del año, mes o semana
DAY	Nombre del día rellenado con espacios en blanco hasta una longitud de nueve caracteres
DY	Nombre del día, abreviatura de tres letras
J	Día juliano actual, número de días desde el 31 de diciembre de 4713 a.C.
IW	Semanas del año según el estándar (1 a 53)

Elementos del Modelo de Formato de Fecha

- Los elementos de tiempo formatean la parte de la hora de la fecha:

`HH24:MI:SS AM`	`15:45:32 PM`

- Agregan cadenas de caracteres entre comillas dobles:

`DD "of" MONTH`	`12 of OCTOBER`

- El número se agrega como sufijo de los números en letra:

`ddspth`	`fourteenth`

ORACLE

Elementos del Modelo de Formato de Fecha

Utilice los formatos de las siguientes tablas para mostrar la información de tiempo y los literales y para cambiar los numerales a números en letra.

Elemento	Descripción
AM o PM	Indicador de meridiano
A.M. o P.M.	Indicador de meridiano con puntos
HH o HH12 o HH24	Hora del día, hora (1–12) u hora (0–23)
MI	Minuto (0–59)
SS	Segundo (0–59)
SSSSS	Segundos después de la media noche (0–86399)

Elementos del Modelo de Formato de Fecha (continuación)

Otros Formatos

Elemento	Descripción
/ . ,	La puntuación se reproduce en el resultado.
“of the”	La cadena entre comillas se reproduce en el resultado.

Especificación de Sufijos para Influenciar la Visualización de Números

Elemento	Descripción
TH	Número ordinal (por ejemplo, DDTH para 4TH)
SP	Número en letra (por ejemplo, DDSP para FOUR)
SPTH or THSP	Números ordinales en letra (por ejemplo, DDSPTH para FOURTH)

Uso de la Función TO_CHAR con Fechas

```
SELECT last_name,
       TO_CHAR(hire_date, 'fmDD Month YYYY')
       AS HIREDATE
FROM   employees;
```

	LAST_NAME	HIREDATE
1	Whalen	17 September 1987
2	Hartstein	17 February 1996
3	Fay	17 August 1997
4	Higgins	7 June 1994
5	Gietz	7 June 1994
6	King	17 June 1987
7	Kochhar	21 September 1989
8	De Haan	13 January 1993
9	Hunold	3 January 1990
10	Ernst	21 May 1991

...

ORACLE

Uso de la Función TO_CHAR con Fechas

La sentencia SQL de la diapositiva muestra los apellidos y fechas de contratación de todos los empleados. La fecha de contratación aparece como 17 de junio de 1987.

Ejemplo:

Modifique el ejemplo de la diapositiva para mostrar las fechas en un formato que aparezca como "Diecisiete de junio de 1987 12:00:00 AM."

```
SELECT last_name,
 TO_CHAR(hire_date,
      'fmDdspth "of" Month YYYY fmHH:MI:SS AM')
 HIREDATE
FROM    employees;
```

	LAST_NAME	HIREDATE
1	Whalen	Seventeenth of September 1987 12:00:00 AM
2	Hartstein	Seventeenth of February 1996 12:00:00 AM

...

Observe que el mes sigue el modelo de formato especificado; es decir, la primera letra está en mayúscula y el resto en minúsculas.

Uso de la Función TO_CHAR con Números

```
TO_CHAR(number, 'format_model')
```

Éstos son algunos de los elementos de formato que puede utilizar con la función TO_CHAR para mostrar un valor de número como un carácter:

Elemento	Resultado
9	Representa un número
0	Fuerza para que aparezca un cero
$	Coloca un signo de dólar flotante
L	Utiliza el símbolo de divisa local flotante
.	Imprime un punto decimal
,	Imprime una coma como indicador de miles

ORACLE

Uso de la Función TO_CHAR con Números

Al trabajar con valores de número como cadenas de caracteres, debe convertir dichos números al tipo de dato de carácter utilizando la función TO_CHAR que traduce un valor del tipo de dato NUMBER a un tipo de dato VARCHAR2. Esta técnica es muy útil con la concatenación.

Uso de la Función TO_CHAR con Números (continuación)

Elementos de Formato de Número

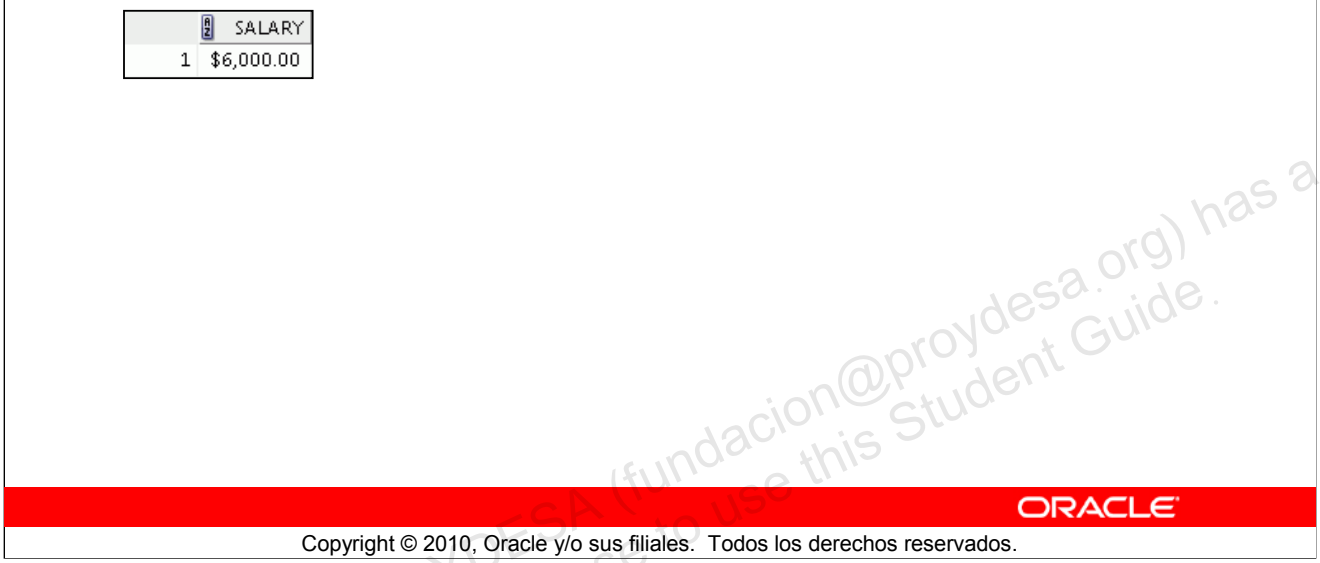
Si está convirtiendo un número al tipo de dato de caracteres, puede utilizar los siguientes elementos de formato:

Elemento	Descripción	Ejemplo	Resultado
9	Posición numérica (número de 9 que determinan el ancho de la visualización)	999999	1234
0	Muestra los ceros iniciales	099999	001234
$	Signo de dólar flotante	$999999	$1234
L	Símbolo de divisa local flotante	L999999	FF1234
D	Devuelve el carácter decimal en la posición especificada. El valor por defecto es un punto (.)	99D99	99.99
.	Punto decimal en la posición especificada	999999.99	1234.00
G	Devuelve un separador de grupo en la posición especificada. Puede especificar varios separadores de grupo en un modelo de formato de número.	9,999	9G999
,	Coma en la posición especificada	999,999	1,234
MI	Signos menos a la derecha (valores negativos)	999999MI	1234-
PR	Introduce entre paréntesis los números negativos	999999PR	<1234>
EEEE	Notación científica (el formato debe especificar cuatro E)	99.999EEEE	1.234E+03
U	Devuelve la divisa dual "Euro" (u otra) en la posición especificada	U9999	€1234
V	Multiplica por 10 *n* veces (n = número de 9 después de V)	9999V99	123400
S	Devuelve un valor negativo o positivo	S9999	-1234 or +1234
B	Muestra los valores cero como espacios en blanco, no como 0	B9999.99	1234.00

Uso de la Función TO_CHAR con Números

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```

	SALARY
1	$6,000.00



Uso de la Función TO_CHAR con Números (continuación)

- El servidor de Oracle muestra una cadena de signos de número (#) en lugar de un número completo cuyos dígitos exceden el número de dígitos proporcionado en el modelo de formato.
- El servidor de Oracle redondea el valor decimal almacenado al número de decimales proporcionado en el modelo de formato.

Uso de Funciones TO_NUMBER y TO_DATE

- Convertir una cadena de caracteres a un formato de número que utiliza la función TO_NUMBER:

```
TO_NUMBER(char[, 'format_model'])
```

- Convertir una cadena de caracteres a un formato de fecha que utiliza la función TO_DATE:

```
TO_DATE(char[, 'format_model'])
```

- Estas funciones tienen un modificador fx. Este modificador especifica la coincidencia exacta para el argumento de carácter y el modelo de formato de fecha de una función TO_DATE.

Uso de Funciones TO_NUMBER y TO_DATE

Puede que desee convertir una cadena de caracteres a un número o a una fecha. Para realizar esta tarea, utilice las funcione TO_NUMBER o TO_DATE. El modelo de formato que seleccione está basado en los elementos de formato demostrados anteriormente.

El modificador fx especifica la coincidencia exacta para el argumento de carácter y el modelo de formato de fecha de una función TO_DATE:

- La puntuación y el texto entre comillas del argumento de carácter debe coincidir exactamente (excepto en las mayúsculas/minúsculas) con las partes correspondientes del modelo de formato.
- El argumento de carácter no puede tener espacios en blanco adicionales. Sin fx, el servidor de Oracle ignora los espacios en blanco adicionales.
- Los datos numéricos del argumento de carácter deben tener el mismo número de dígitos que el elemento correspondiente en el modelo de formato. Sin fx, los números del argumento de carácter no pueden omitir los ceros iniciales.

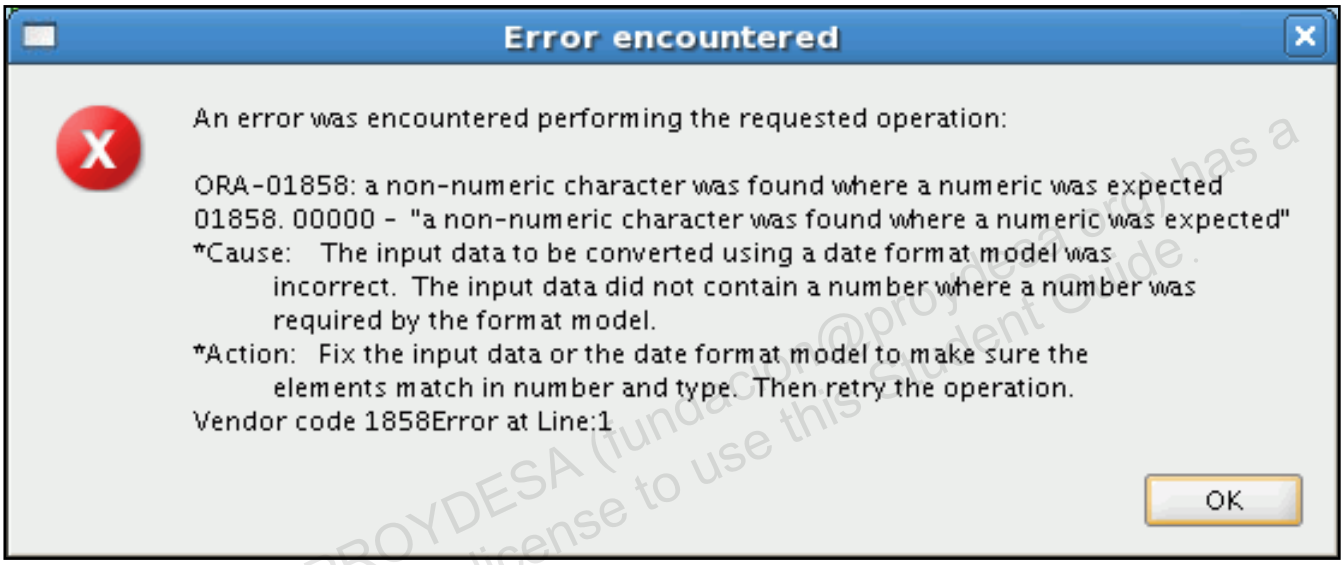
Uso de Funciones TO_NUMBER y TO_DATE (continuación)

Ejemplo:

Muestre el nombre y fecha de contratación de todos los empleados que empezaron a trabajar el 24 de mayo de 1999. En el siguiente ejemplo, hay dos espacios después del mes *May* y antes del número *24*. Ya que se utiliza `fx`, se necesita una coincidencia exacta y los espacios después de la palabra *May* no se reconocerán:

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date = TO_DATE('May  24, 1999', 'fxMonth DD, YYYY');
```

La salida del error resultante debe tener el siguiente aspecto:



Para ver la salida, corrija la consulta suprimiendo el espacio adicional entre "May" y "24".

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY');
```

	LAST_NAME	HIRE_DATE
1	Grant	24-MAY-99

Uso de las Funciones TO_CHAR y TO_DATE con el Formato de Fecha RR

Para buscar los empleados contratados antes de 1990, utilice el formato de fecha RR, que produce los mismos resultados si se ejecutara el comando en 1999 o en la actualidad:

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM  employees
WHERE hire_date < TO_DATE('01-Jan-90','DD-Mon-RR');
```

	LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1	Whalen	17-Sep-1987
2	King	17-Jun-1987
3	Kochhar	21-Sep-1989

ORACLE

Uso de las Funciones TO_CHAR y TO_DATE con el Formato de Fecha RR

Para buscar los empleados contratados antes de 1990, se puede utilizar el formato RR. Debido a que el año actual es mayor que 1999, el formato RR interpreta la parte del año de la fecha de 1950 a 1999.

Por otro lado, en el siguiente comando no se selecciona ninguna fila porque el formato YY interpreta la parte del año de la fecha en el siglo actual (2090).

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-yyyy')
FROM   employees
WHERE  TO_DATE(hire_date, 'DD-Mon-yy') < '01-Jan-1990';
```

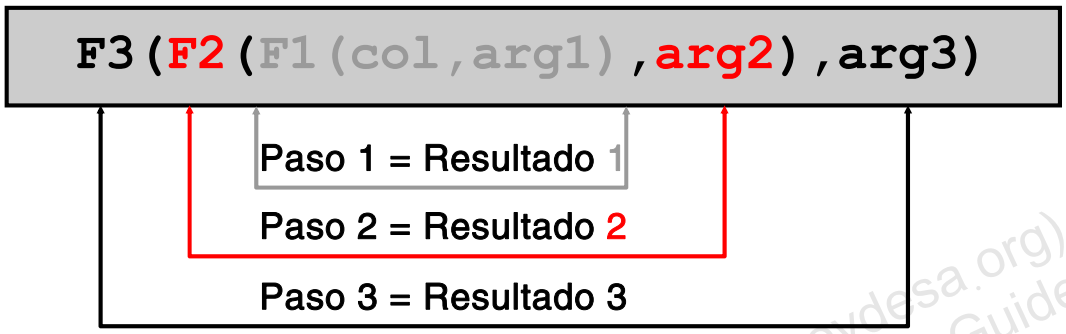
```
0 rows selected
```

Agenda

- Conversión de tipo de dato implícito y explícito
- Funciones `TO_CHAR`, `TO_DATE`, `TO_NUMBER`
- Funciones de anidación
- Funciones generales:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Expresiones condicionales:
  - `CASE`
  - `DECODE`

Funciones de Anidación

- Las funciones de una sola fila se pueden anidar en cualquier nivel.
- Las funciones anidadas se evalúan desde el nivel más profundo hasta el nivel menos profundo.



ORACLE

Funciones de Anidación

Las funciones de una sola fila se pueden anidar en cualquier profundidad. Las funciones anidadas se evalúan desde el nivel más profundo hasta el nivel menos profundo. Los siguientes ejemplos muestran la flexibilidad de estas funciones.

Funciones de Anidación: Ejemplo 1

```
SELECT last_name,
  UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))
FROM   employees
WHERE  department_id = 60;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US

Funciones de Anidación (continuación)

El ejemplo de la diapositiva muestra los apellidos de los empleados en el departamento 60. La evaluación de la sentencia SQL implica tres pasos:

1. La función interna recupera los primeros ocho caracteres del apellido.
   `Result1 = SUBSTR (LAST_NAME, 1, 8)`
2. La función externa concatena el resultado con `_US`.
   `Result2 = CONCAT(Result1, '_US')`
3. La función más externa convierte los resultados a mayúsculas.

La expresión completa se convierte en la cabecera de columna porque no se ha proporcionado ningún alias de columna.

Ejemplo:

Muestra la fecha del siguiente viernes seis meses después de la fecha de contratación. La fecha resultante debe aparecer como viernes 13 de agosto de 1999. Ordene los resultados por fecha de contratación.

```
SELECT   TO_CHAR(NEXT_DAY(ADD_MONTHS
         (hire_date, 6), 'FRIDAY'),
         'fmDay, Month ddth, YYYY')
         "Next 6 Month Review"
FROM      employees
ORDER BY hire_date ;
```

Funciones de Anidación: Ejemplo 2

```
SELECT   TO_CHAR(ROUND((salary/7), 2),'99G999D99',
         'NLS_NUMERIC_CHARACTERS = '',.'' ')
         "Formatted Salary"
FROM employees;
```

	Formatted Salary
1	628,57
2	1.857,14
3	857,14
4	1.714,29
5	1.185,71
6	3.428,57

...

ORACLE

Funciones de Anidación (continuación)

El ejemplo de la diapositiva muestra los salarios de los empleados dividido entre7 y redondeado a dos decimales. El resultado se formatea para mostrar el salario con notación en danés. Es decir, la coma se utilizar como decimal y el punto para los miles.

En primer lugar, la función interna `ROUND` se ejecuta para redondear el valor del salario dividido entre 7 a dos decimales. La función `TO_CHAR`se utiliza entonces para aplicar formato al resultado de la función `ROUND`.

Nota: los elementos D y G especificados en el parámetro de función `TO_CHAR` son elementos de formato de número. D devuelve un carácter decimal en la posición especificada. G se utiliza como un separador de grupo.

Agenda

- Conversión de tipo de dato implícito y explícito
- Funciones `TO_CHAR`, `TO_DATE`, `TO_NUMBER`
- Funciones de anidación
- Funciones generales:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Expresiones condicionales:
  - `CASE`
  - `DECODE`

Funciones Generales

Las siguientes funciones funcionan con cualquier tipo de dato y pertenecen al uso de valores nulos:

- `NVL (expr1, expr2)`
- `NVL2 (expr1, expr2, expr3)`
- `NULLIF (expr1, expr2)`
- `COALESCE (expr1, expr2, ..., exprn)`

Funciones Generales

Estas funciones funcionan con cualquier tipo de dato y pertenecen al uso de valores nulos en la lista de expresiones.

Función	Descripción
`NVL`	Convierte un valor nulo en un valor real
`NVL2`	Si `expr1` no es nulo, `NVL2` devuelve `expr2`. Si `expr1` es nulo, `NVL2` devuelve `expr3`. El argumento `expr1` puede tener cualquier tipo de dato.
`NULLIF`	Compara dos expresiones y devuelve un valor nulo si son iguales; si no son iguales, devuelve la primera expresión
`COALESCE`	Devuelve la primera expresión no nula en la lista de expresiones

Nota: para obtener más información sobre los cientos de funciones disponibles, consulte la sección sobre funciones en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Función NVL

Convierte un valor nulo a un valor real:

- Los tipos de dato que se pueden utilizar son fecha, carácter y número.
- Los tipos de dato deben coincidir con:
  - `NVL(commission_pct,0)`
  - `NVL(hire_date,'01-JAN-97')`
  - `NVL(job_id,'No Job Yet')`

ORACLE

Función NVL

Para convertir un valor nulo a un valor real, utilice la función `NVL`.

Sintaxis

```
NVL (expr1, expr2)
```

En la sintaxis:

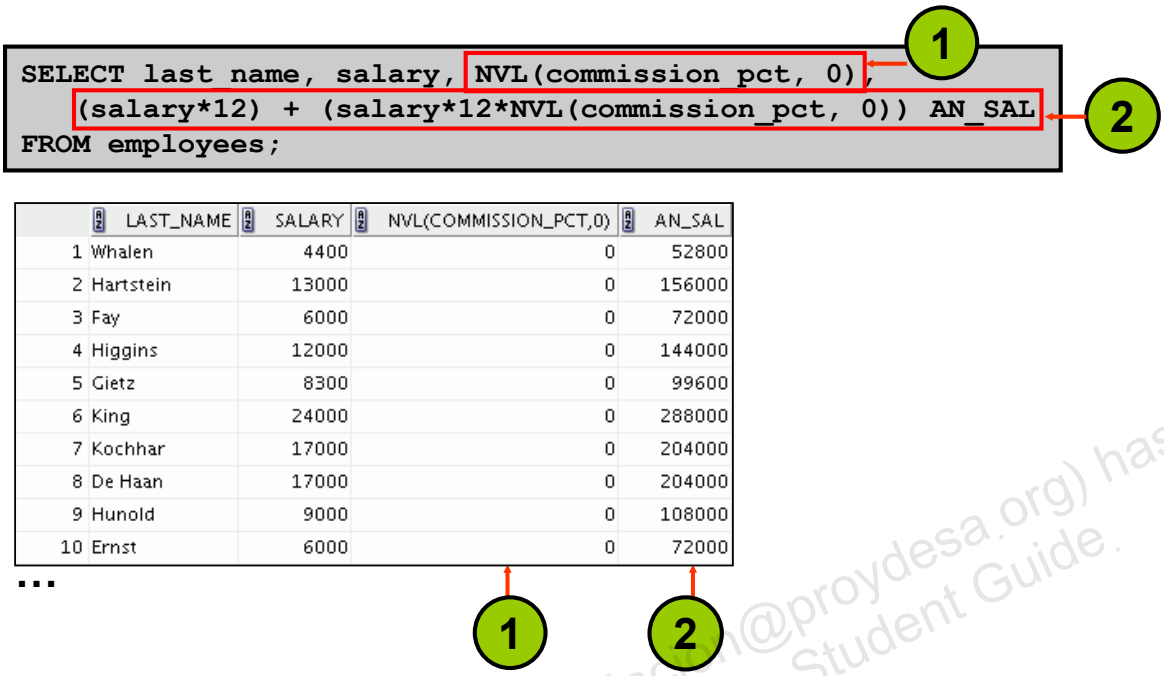
- *expr1* es el valor de origen o expresión que puede contener un valor nulo
- *expr2* es el valor de destino para convertir el valor nulo

Puede utilizar la función `NVL` para convertir cualquier tipo de dato, pero el valor de retorno siempre es el mismo que el tipo de dato de *expr1*.

Conversiones NVL para Varios Tipos de Dato

Tipo de Dato	Ejemplo de Conversión
NUMBER	NVL(*number_column*,9)
DATE	NVL(*date_column*, '01-JAN-95')
CHAR or VARCHAR2	NVL(*character_column*, 'Unavailable')

Uso de la Función NVL



Uso de la Función NVL

Para calcular la compensación anual de los empleados, necesita multiplicar el salario mensual por 12 y, a continuación, agregue el porcentaje de comisión al resultado:

```
SELECT last_name,  salary, commission_pct,
 (salary*12) + (salary*12*commission_pct) AN_SAL
FROM   employees;
```

	LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
1	Whalen	4400	(null)	(null)
...				
16	Vargas	2500	(null)	(null)
17	Zlotkey	10500	0.2	151200
18	Abel	11000	0.3	171600
19	Taylor	8600	0.2	123840
20	Grant	7000	0.15	96600

Tenga en cuenta que la compensación anual se calcula sólo para los empleados que obtienen una comisión. Si cualquier valor de columna de una expresión es nulo, el resultado es nulo. Para calcular los valores para todos los empleados, puede convertir los valores nulos a un número antes de aplicar el operador aritmético. En el ejemplo de la diapositiva, la función NVL se utiliza para convertir valores nulos a cero.

Uso de la Función NVL2

```
SELECT last_name,  salary, commission_pct
       NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

1

2

	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM

1 2

ORACLE

Uso de la Función NVL2

La función NVL2 examina la primera expresión. Si la primera expresión no es nula, la función NVL2 devuelve la segunda expresión. Si la primera expresión es nula, se devolverá la tercera expresión.

Sintaxis

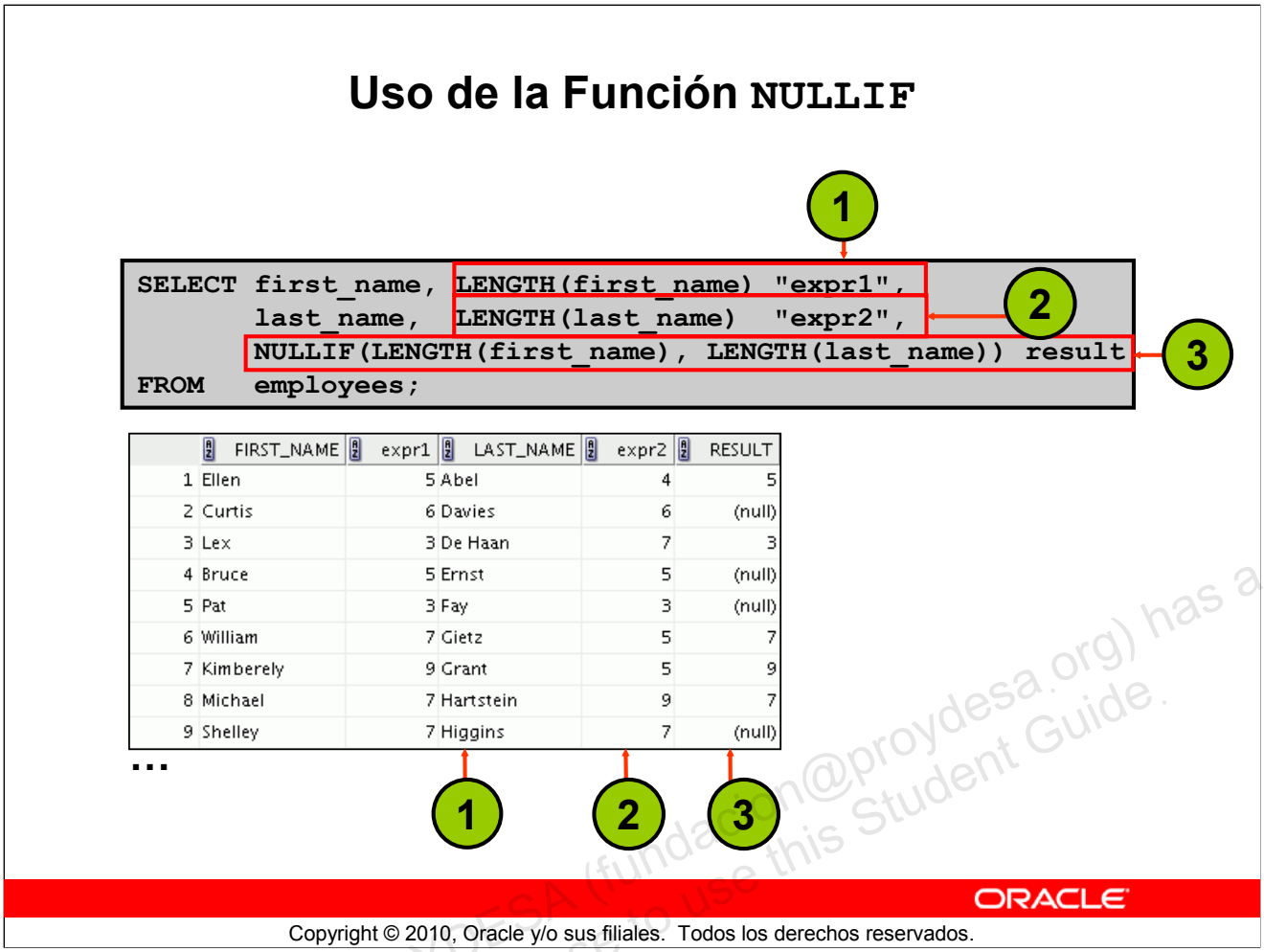
```
NVL2(expr1, expr2, expr3)
```

En la sintaxis:

- *expr1* es el valor de origen o expresión que puede contener un valor nulo
- *expr2* es el valor que se devuelve si *expr1* no es nulo
- *expr3* es el valor que se devuelve si *expr1* es nulo

En el ejemplo de la diapositiva se examina la columna COMMISSION_PCT. Si se detecta un valor, se devolverá el valor literal de texto de SAL+COMM. Si la columna COMMISSION_PCT contiene un valor nulo, se devolverá el valor literal de texto SAL.

Nota: el argumento *expr1* puede tener cualquier tipo de dato. Los argumentos *expr2* y *expr3* pueden tener cualquier tipo de dato excepto LONG.



Uso de la Función NULLIF

La función NULLIF compara dos expresiones.

Sintaxis

```
NULLIF (expr1, expr2)
```

En la sintaxis:

- NULLIF compara *expr1* y *expr2*. Si son iguales, la función devuelve un valor nulo. Si no, la función devuelve *expr1*. Sin embargo, no puede especificar el literal NULL para *expr1*.

En el ejemplo de la diapositiva, la longitud del nombre en la tabla EMPLOYEES se compara con la longitud del apellido en la tabla EMPLOYEES. Cuando las longitudes de ambos son iguales, se mostrará un valor nulo. Cuando las longitudes no son iguales, se mostrará la longitud del nombre.

Nota: la función NULLIF es el equivalente lógico de la siguiente expresión CASE. La expresión CASE se trata en la siguiente página:

```
CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END
```

Uso de la Función COALESCE

- La ventaja de la función COALESCE con respecto a la función NVL es que la función COALESCE puede obtener múltiples valores alternativos.
- Si la primera expresión no es nula, la función COALESCE devuelve esa expresión; de lo contrario, aplica la función COALESCE de las expresiones restantes.

ORACLE

Uso de la Función COALESCE

La función COALESCE devuelve la primera expresión no nula de la lista.

Sintaxis

```
COALESCE (expr1, expr2, ... exprn)
```

En la sintaxis:

- *expr1* devuelve esta expresión si no es nula
- *expr2* devuelve esta expresión si la primera expresión es nula y ésta no es nula
- *exprn* devuelve esta expresión si las expresiones anteriores son nulas

Tenga en cuenta que todas las expresiones deben ser del mismo tipo de dato.

Uso de la Función COALESCE

```
SELECT last_name, employee_id,
COALESCE(TO_CHAR(commission_pct),TO_CHAR(manager_id),
         'No commission and no manager')
FROM employees;
```

	LAST_NAME	EMPLOYEE_ID	COALESCE(TO_CHAR(COMMISSI...
1	Whalen	200	101
2	Hartstein	201	100
3	Fay	202	201
4	Higgins	205	101
5	Gietz	206	205
6	King	100	No commission and no manager

...

17	Zlotkey	149	.2
18	Abel	174	.3
19	Taylor	176	.2
20	Grant	178	.15

ORACLE

Uso de la Función COALESCE (continuación)

En el ejemplo de la diapositiva se muestra el valor `manager_id` si éste no es nulo. Si el valor `manager_id` es nulo, se mostrará `commission_pct`. Si los valores `manager_id` y `commission_pct` son nulos, se muestra “No commission and no manager”. Tenga en cuenta que la función `TO_CHAR` se aplica de modo que todas las expresiones sean del mismo tipo de dato.

Uso de la Función COALESCE (continuación)

Ejemplo:

Para los empleados que no perciben ninguna comisión, la organización desea proporcionar un aumento de salario de 2.000 dólares y para los empleados que perciben comisión, la consulta debe calcular el nuevo salario que es igual al salario existente sumado a la comisión.

```
SELECT last_name,  salary, commission_pct,
 COALESCE((salary+(commission_pct*salary)), salary+2000, salary) "New
  Salary"
FROM   employees;
```

Nota: examine la salida. Para los empleados que no perciben ninguna comisión, la columna de nuevo salario muestra el salario aumentado en 2.000 dólares y para los empleados que perciben comisión, la columna de nuevo salario muestra la comisión calculada sumada al salario.

	LAST_NAME	SALARY	COMMISSION_PCT	New Salary
1	Whalen	4400	(null)	6400
2	Hartstein	13000	(null)	15000
3	Fay	6000	(null)	8000
4	Higgins	12000	(null)	14000
5	Gietz	8300	(null)	10300
6	King	24000	(null)	26000

...

17	Zlotkey	10500	0.2	12600
18	Abel	11000	0.3	14300
19	Taylor	8600	0.2	10320
20	Grant	7000	0.15	8050

Agenda

- Conversión de tipo de dato implícito y explícito
- Funciones `TO_CHAR`, `TO_DATE`, `TO_NUMBER`
- Funciones de anidación
- Funciones generales:
  - `NVL`
  - `NVL2`
  - `NULLIF`
  - `COALESCE`
- Expresiones condicionales:
  - `CASE`
  - `DECODE`

Expresiones Condicionales

- Proporcionar el uso de la lógica `IF-THEN-ELSE` en una sentencia SQL
- Utilizar dos métodos:
  - Expresión `CASE`
  - Función `DECODE`

ORACLE

Expresiones Condicionales

Los dos métodos que se utilizan para implantar el procesamiento condicional (lógica `IF-THEN-ELSE`) en la sentencia SQL son la expresión `CASE` y la función `DECODE`.

Nota: la expresión `CASE` cumple con SQL ANSI. La función `DECODE` es específica de la sintaxis de Oracle.

Expresión CASE

Facilita las consultas condicionales realizando el trabajo de una sentencia IF-THEN-ELSE:

```
CASE expr WHEN comparison_expr1 THEN return_expr1
          [WHEN comparison_expr2 THEN return_expr2
           WHEN comparison_exprn THEN return_exprn
           ELSE else_expr]
END
```


Expresión CASE

Las expresiones CASE permiten utilizar la lógica IF-THEN-ELSE en las sentencias SQL sin tener que llamar a los procedimientos.

En una expresión CASE simple, el servidor de Oracle busca el primer par WHEN ... THEN para el que expr es igual a comparison_expr y devuelve return_expr. Si ninguno de los pares WHEN ... THEN cumple con esta condición y existe una cláusula ELSE, el servidor de Oracle devuelve else_expr. De lo contrario, el servidor de Oracle devuelve un valor nulo. No puede especificar el valor literal NULL para todas las expresiones return_expr y else_expr.

Las expresiones expr y comparison_expr deben ser del mismo tipo de dato, que puede ser CHAR, VARCHAR2, NCHAR o NVARCHAR2. Todos los valores de retorno (return_expr) deben ser del mismo tipo de dato.

Uso de la Expresión CASE

Facilita las consultas condicionales realizando el trabajo de una sentencia IF-THEN-ELSE:

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE       salary END     "REVISED_SALARY"
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
1	Whalen	AD_ASST	4400	4400
...				
9	Hunold	IT_PROG	9000	9900
10	Ernst	IT_PROG	6000	6600
11	Lorentz	IT_PROG	4200	4620
12	Mourgos	ST_MAN	5800	5800
13	Rajs	ST_CLERK	3500	4025
14	Davies	ST_CLERK	3100	3565
...				
19	Taylor	SA_REP	8600	10320
20	Grant	SA_REP	7000	8400

ORACLE

Uso de la Expresión CASE

En la sentencia SQL de la diapositiva, se descodifica el valor de JOB_ID. Si JOB_ID es IT_PROG, el aumento de salario es del 10%; si JOB_ID es ST_CLERK, el aumento de salario es del 15%; si JOB_ID es SA_REP, el aumento del salario es del 20%. Para el resto de otros roles de cargo, no hay ningún aumento de salario.

La misma sentencia se puede escribir con la función DECODE.

El siguiente código es un ejemplo de la expresión buscada CASE. En una expresión CASE buscada, la búsqueda se realiza de izquierda a derecha hasta que se encuentra una coincidencia de la condición mostrada y, a continuación, se devuelve la expresión de retorno. Si no se encuentra ninguna condición verdadera y existe una cláusula ELSE, se devuelve la expresión de retorno en la cláusula ELSE; de lo contrario, se devuelve NULL.

```
SELECT last_name,salary,
(CASE WHEN salary<5000 THEN 'Low'
      WHEN salary<10000 THEN 'Medium'
      WHEN salary<20000 THEN 'Good'
      ELSE 'Excellent'
END) qualified_salary
FROM employees;
```

Función DECODE

Facilita las consultas condicionales realizando el trabajo de una expresión CASE o una sentencia IF-THEN-ELSE:

```
DECODE(col|expression, search1, result1
                       [, search2, result2,...,]
                       [, default])
```


Función DECODE

La función DECODE decodifica una expresión de una forma similar a la lógica IF-THEN-ELSE que se utiliza en varios idiomas. La función DECODE descodifica *expression* después de compararla con cada valor *search*. Si la expresión es la misma que *search*, se devuelve *result*.

Si se omite el valor por defecto, se devuelve un valor nulo donde un valor de búsqueda no coincida con ninguno de los valores de resultado.

Uso de la Función DECODE

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
10 Ernst	IT_PROG	6000	6600
11 Lorentz	IT_PROG	4200	4620
12 Mourgos	ST_MAN	5800	5800
13 Rajs	ST_CLERK	3500	4025
...			
19 Taylor	SA_REP	8600	10320
20 Grant	SA_REP	7000	8400

ORACLE

Uso de la Función DECODE

En la sentencia SQL de la diapositiva, se prueba el valor de JOB_ID. Si JOB_ID es IT_PROG, el aumento de salario es del 10%; si JOB_ID es ST_CLERK, el aumento de salario es del 15%; si JOB_ID es SA_REP, el aumento del salario es del 20%. Para el resto de otros roles de cargo, no hay ningún aumento de salario.

La misma sentencia se puede expresar en pseudocódigo como una sentencia IF-THEN-ELSE:

```
IF job_id = 'IT_PROG'     THEN  salary = salary*1.10
IF job_id = 'ST_CLERK'    THEN  salary = salary*1.15
IF job_id = 'SA_REP'      THEN  salary = salary*1.20
ELSE salary = salary
```

Uso de la Función DECODE

Mostrar el impuesto aplicable para cada empleado del departamento 80:

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                         0, 0.00,
                         1, 0.09,
                         2, 0.20,
                         3, 0.30,
                         4, 0.40,
                         5, 0.42,
                         6, 0.44,
                            0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

ORACLE

Uso de la Función DECODE (continuación)

Esta diapositiva muestra otro ejemplo que utiliza la función DECODE. En este ejemplo, puede determinar los impuestos para cada empleado del departamento 80 según su salario mensual. Los impuestos son los siguientes:

Rango de Salario Mensual	*Impuesto*
$0,00–1.999,99	00%
$2.000,00–3.999,99	09%
$4.000,00–5.999,99	20%
$6.000,00–7.999,99	30%
$8.000,00–9.999,99	40%
$10.000,00–11.999,99	42%
$12.200,00–13.999,99	44%
14.000,00 o superior	45%

	LAST_NAME	SALARY	TAX_RATE
1	Zlotkey	10500	0.42
2	Abel	11000	0.42
3	Taylor	8600	0.4

Prueba

La función `TO_NUMBER` convierte las cadenas de caracteres o los valores de fecha a un número con el formato especificado por el modelo de formato opcional.

1. True
2. Falso

ORACLE

Respuesta: 2

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Modificar formatos de fecha para su visualización utilizando funciones
- Convertir tipos de dato de columna utilizando funciones
- Utilizar funciones `NVL`
- Utilizar la lógica `IF-THEN-ELSE` y otra expresión condicional en una sentencia `SELECT`

Resumen

Recuerde lo siguiente:

- Las funciones de conversión pueden convertir valores numéricos, de fecha y de carácter: Funciones `TO_CHAR`, `TO_DATE`, `TO_NUMBER`
- Existen varias funciones que pertenecen a valores nulos, incluyendo `NVL`, `NVL2`, `NULLIF` y `COALESCE`.
- La lógica `IF-THEN-ELSE` se puede aplicar en una sentencia SQL utilizando la expresión `CASE` o la función `DECODE`.

Práctica 4: Visión General

En esta práctica se abordan los siguientes temas:

- Creación de consultas que utilizan `TO_CHAR`, `TO_DATE` y otras funciones `DATE`
- Creación de consultas que utilizan expresiones condicionales como `DECODE` y `CASE`

ORACLE

Práctica 4: Visión General

Esta práctica proporciona una variedad de ejercicios que utilizan las funciones `TO_CHAR` y `TO_DATE` y expresiones condicionales como `DECODE` y `CASE`. Recuerde que para las funciones anidadas, los resultados se evalúan desde la función más profunda hasta la función menos profunda.

5

Informes de Datos Agregados con Funciones de Grupo

ORACLE

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Identificar las funciones de grupo disponibles
- Describir los usos de las funciones de grupo
- Agrupar datos con la cláusula `GROUP BY`
- Incluir o excluir filas agrupadas con la cláusula `HAVING`

ORACLE

Objetivos

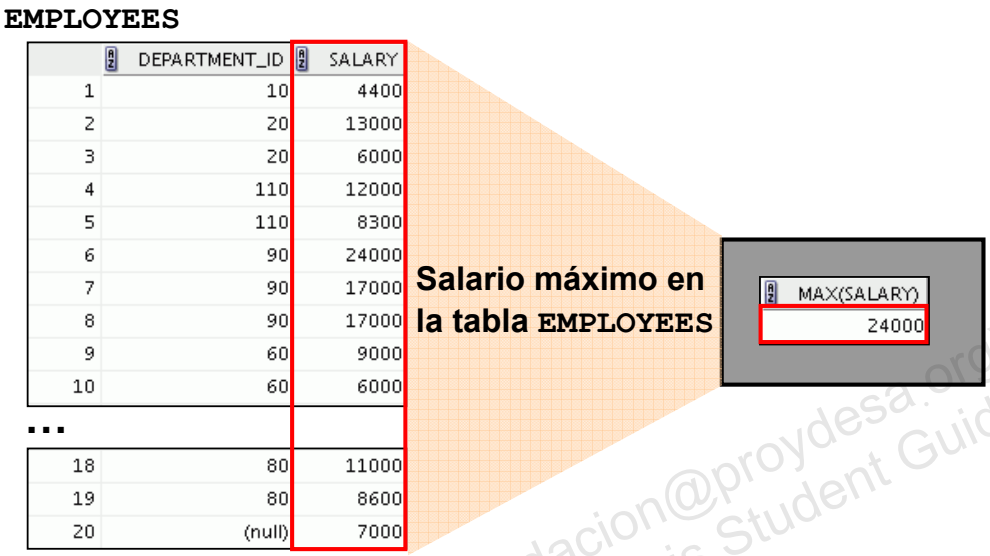
Esta lección trata las funciones más detalladamente. Se centra en obtener información de resumen (como medias) para grupos de filas. Describe cómo agrupar las filas de una tabla en juegos más pequeños y cómo especificar los criterios de búsqueda para grupos de filas.

Agenda

- Funciones de grupo:
  - Tipos y sintaxis
  - Uso de `AVG`, `SUM`, `MIN`, `MAX`, `COUNT`
  - Uso de la palabra clave `DISTINCT` en funciones de grupo
  - Valores `NULL` en una función de grupo
- Agrupar filas:
  - Cláusula `GROUP BY`
  - Cláusula `HAVING`
- Anidamiento de funciones de grupo

¿Qué Son las Funciones de Grupo?

Las funciones de grupo funcionan en juegos de filas para proporcionar un resultado por grupo.



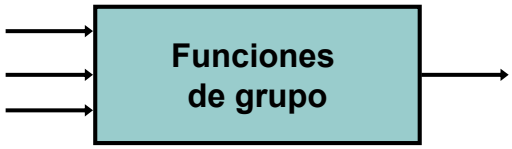
ORACLE

¿Qué Son las Funciones de Grupo?

A diferencia de las funciones de una sola fila, las funciones de grupo funcionan en juegos de filas para proporcionar un resultado por grupo. Estos juegos pueden formar la tabla completa o la tabla dividida en grupos.

Tipos de Funciones de Grupo

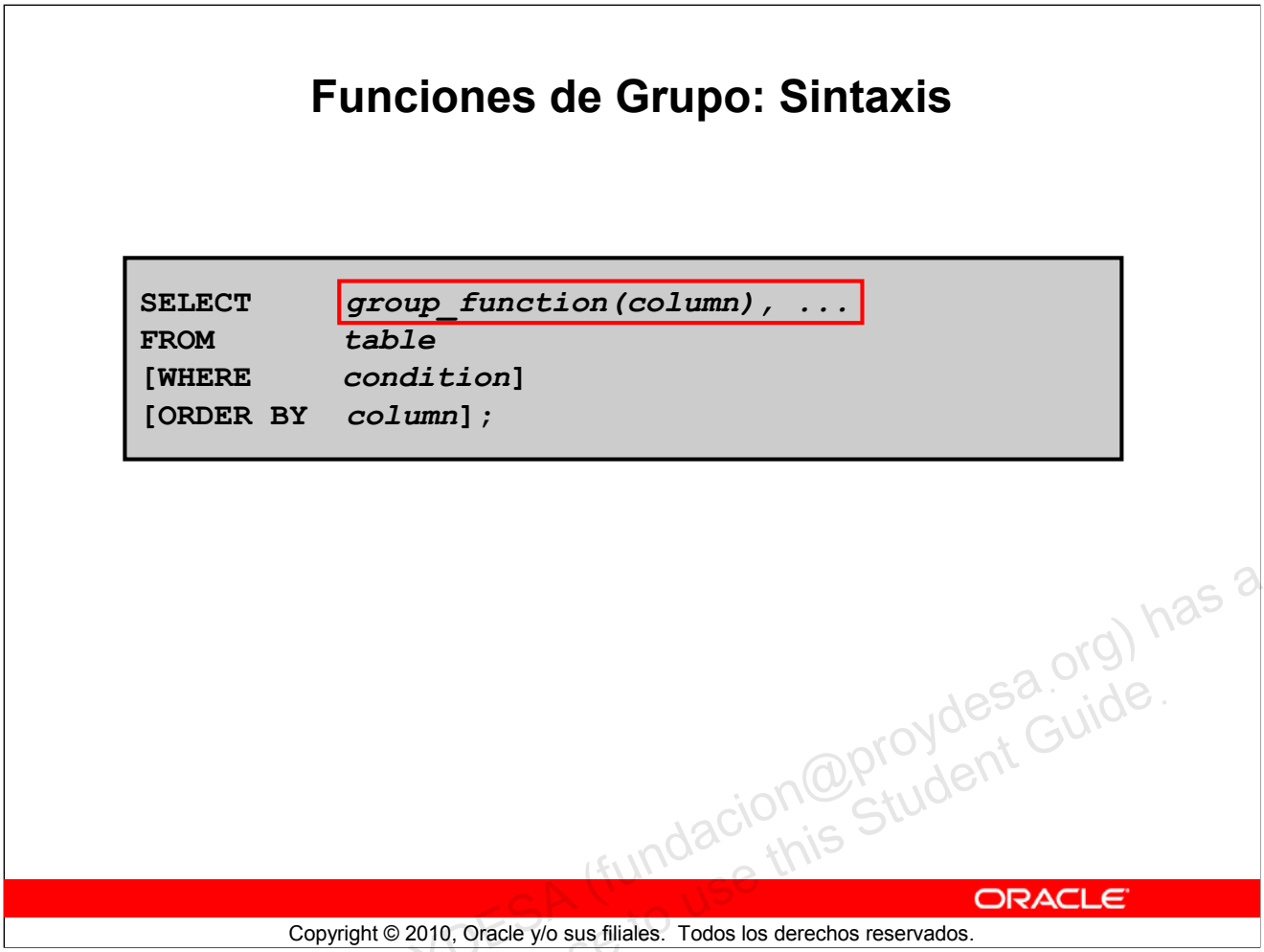
- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE



Tipos de Funciones de Grupo

Cada una de las funciones acepta un argumento. La siguiente tabla identifica las opciones que se pueden utilizar en la sintaxis:

Función	Descripción
AVG([DISTINCT|ALL]*n*)	Valor medio de *n*; ignora los valores nulos
COUNT({*|[DISTINCT|ALL]*expr*})	Número de filas donde expr evalúa otros valores que no son nulos (tiene en cuenta todas filas seleccionadas con *, incluyendo duplicados y filas con valores nulos)
MAX([DISTINCT|ALL]*expr*)	Valor máximo de expr; ignora los valores nulos
MIN([DISTINCT|ALL]*expr*)	Valor mínimo de expr; ignora los valores nulos
STDDEV([DISTINCT|ALL]*n*)	Desviación estándar de *n*; ignora los valores nulos
SUM([DISTINCT|ALL]*n*)	Valores de suma de *n*; ignora los valores nulos
VARIANCE([DISTINCT|ALL]*n*)	Varianza de *n*; ignora los valores nulos



Funciones de Grupo: Sintaxis

La función de grupo se coloca después de la palabra clave `SELECT`. Puede que tenga varias funciones de grupo separadas por comas.

Instrucciones para utilizar las funciones de grupo:

- `DISTINCT` hace que la función considere sólo los valores no duplicados; `ALL` hace que considere cada valor, incluyendo los duplicados. El valor por defecto es `ALL` y, por lo tanto, no es necesario especificarlo.
- Los tipos de dato para las funciones con el argumento `expr` pueden ser `CHAR`, `VARCHAR2`, `NUMBER` o `DATE`.
- Todas las funciones de grupo ignoran los valores nulos. Para sustituir un valor para valores nulos, utilice las funciones `NVL`, `NVL2`, `COALESCE`, `CASE` o `DECODE`.

Uso de las Funciones AVG y SUM

Puede utilizar AVG y SUM para datos numéricos.

```
SELECT AVG(salary), MAX(salary),
       MIN(salary), SUM(salary)
FROM   employees
WHERE  job_id LIKE '%REP%';
```

	AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
1	8150	11000	6000	32600

ORACLE

Uso de las Funciones AVG y SUM

Puede utilizar las funciones AVG, SUM, MIN y MAX en las columnas que pueden almacenar datos numéricos. El ejemplo de la diapositiva muestra la media, el valor más alto, más bajo y la suma de los salarios mensuales de todos los vendedores.

Uso de Funciones MIN y MAX

Puede utilizar MIN y MAX para tipos de dato numéricos, de caracteres y de fecha.

```
SELECT MIN(hire_date), MAX(hire_date)
FROM       employees;
```

	MIN(HIRE_DATE)	MAX(HIRE_DATE)
1	17-JUN-87	29-JAN-00

ORACLE

Uso de Funciones MIN y MAX

Puede utilizar las funciones MAX y MIN para tipos de dato numéricos, de caracteres y de fecha. El ejemplo de la diapositiva muestra los empleados más y menos experimentados.

El siguiente ejemplo muestra el apellido del empleado que está en primer lugar y del que está en último lugar en una lista de todos los empleados ordenada alfabéticamente:

```
SELECT MIN(last_name), MAX(last_name)
FROM   employees;
```

	MIN(LAST_NAME)	MAX(LAST_NAME)
1	Abel	Zlotkey

Nota: las funciones AVG, SUM, VARIANCE y STDDEV se pueden utilizar sólo con los tipos de dato numéricos. MAX and MIN no se pueden utilizar con tipos de dato LOB o LONG.

Uso de la Función COUNT

`COUNT(*)` devuelve el número de filas en una tabla:

1
```
SELECT COUNT(*)
FROM   employees
WHERE  department_id = 50;
```

	COUNT(*)
1	5

`COUNT(expr)` devuelve el número de filas con valores no nulos para la expresión *expr*:

2
```
SELECT COUNT(commission_pct)
FROM   employees
WHERE  department_id = 80;
```

	COUNT(COMMISSION_PCT)
1	3

ORACLE

Uso de la Función COUNT

La función `COUNT` tiene tres formatos:

- `COUNT(*)`
- `COUNT(expr)`
- `COUNT(DISTINCT expr)`

`COUNT(*)` devuelve el número de filas en una tabla que cumplan con el criterio de la sentencia `SELECT`, incluyendo las filas duplicadas y las filas que contengan valores nulos en cualquiera de las columnas. Si se incluye una cláusula`WHERE` en la sentencia `SELECT`, `COUNT(*)` devuelve el número de filas que cumpla con la condición de la cláusula `WHERE`.

Por el contrario, `COUNT(expr)` devuelve el número de valores no nulos que están en la columna identificada con *expr*.

`COUNT(DISTINCT expr)` devuelve el número de valores únicos no nulos que están en la columna identificada con *expr*.

Ejemplos:

1. El ejemplo de la diapositiva muestra el número de empleados del departamento 50.
2. También muestra el número de empleados del departamento 80 que pueden percibir una comisión.

Uso de la Palabra Clave DISTINCT

- COUNT(DISTINCT expr) devuelve el número con valores distintos no nulos de *expr*.
- Para mostrar el número de valores distintos de departamento en la tabla EMPLOYEES:

```
SELECT COUNT(DISTINCT department_id)
FROM   employees;
```

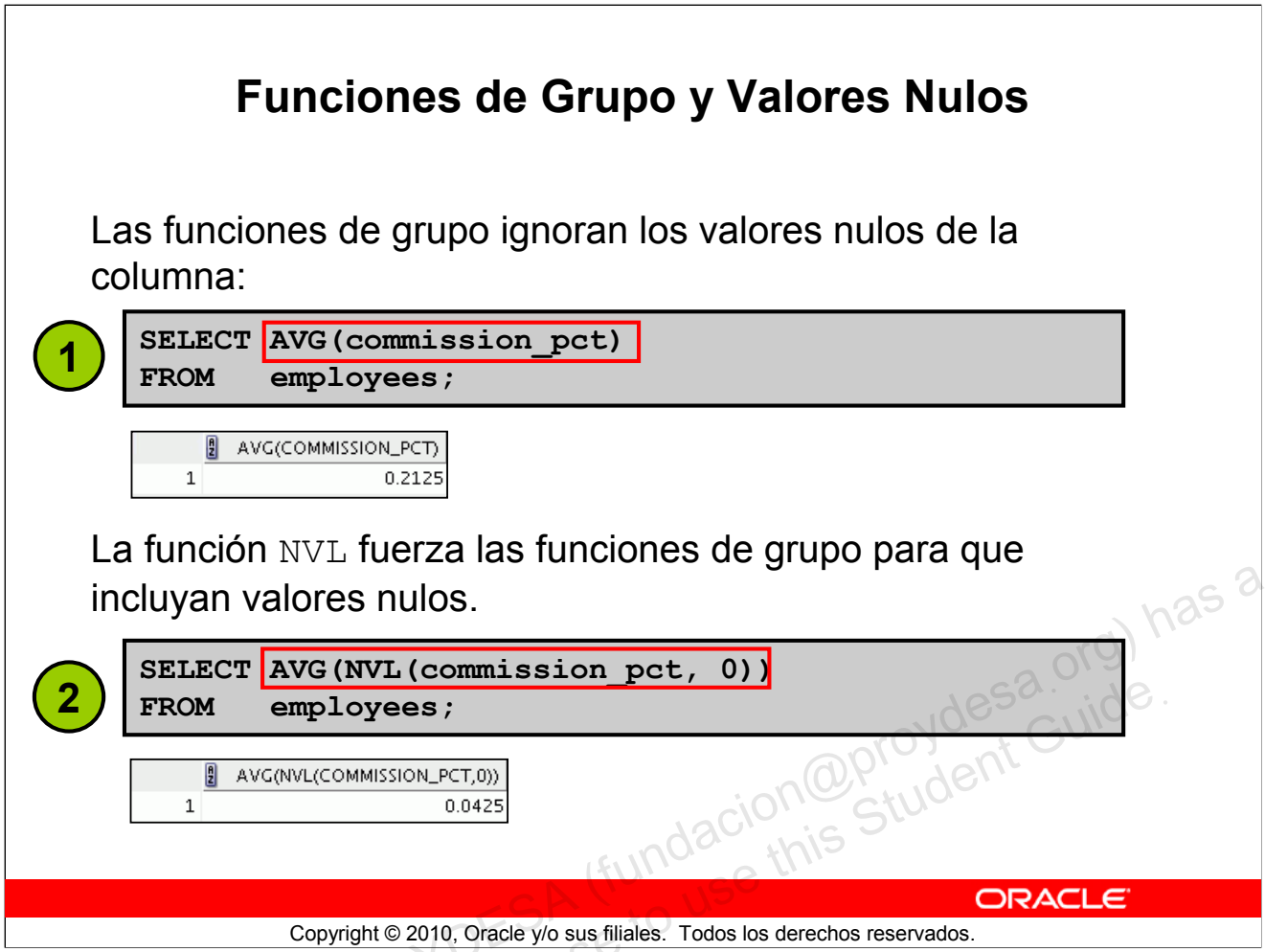
	COUNT(DISTINCTDEPARTMENT_ID)
1	7

ORACLE

Uso de la Palabra Clave DISTINCT

Utilice la palabra clave DISTINCT para suprimir el recuento de cualquier valor duplicado en una columna.

El ejemplo de la diapositiva muestra el número de valores de departamento distintos que están en la tabla EMPLOYEES.



Funciones de Grupo y Valores Nulos

Todas las funciones de grupo ignoran los valores nulos de la columna.

Sin embargo, `NVL` fuerza las funciones de grupo para que incluyan valores nulos.

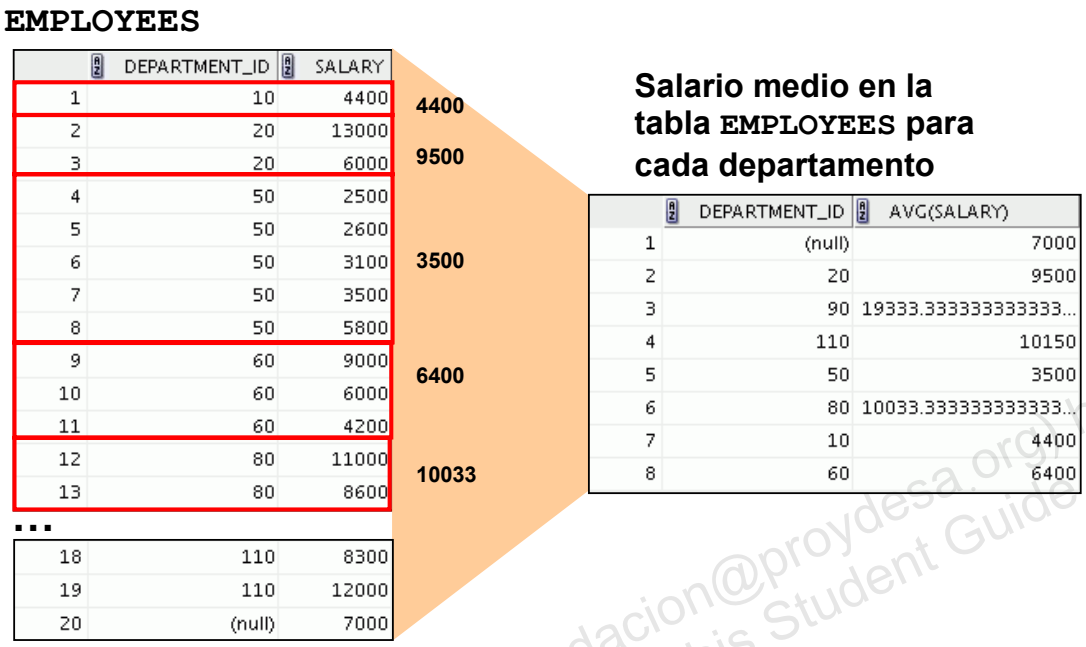
Ejemplos:

1. La media se calcula *únicamente* en base a las filas de la tabla en las que se almacena un valor válido en la columna `COMMISSION_PCT`. La media se calcula con la comisión total pagada a todos los empleados dividida entre el número de empleados que perciben una comisión (cuatro).
2. La media se calcula en base a *todas* las filas de la tabla, independientemente de si los valores nulos se almacenan en la columna `COMMISSION_PCT`. La media se calcula con la comisión total pagada a todos los empleados dividida entre el número de empleados de la compañía (20).

Agenda

- Funciones de grupo:
  - Tipos y sintaxis
  - Uso de `AVG`, `SUM`, `MIN`, `MAX`, `COUNT`
  - Uso de la palabra clave `DISTINCT` en funciones de grupo
  - Valores `NULL` en una función de grupo
- Agrupar filas:
  - Cláusula `GROUP BY`
  - Cláusula `HAVING`
- Anidamiento de funciones de grupo

Creación de Grupos de Datos



Creación de Grupos de Datos

Hasta este punto, todas las funciones de grupo han considerado la tabla como un grupo de información de gran tamaño. Sin embargo, en ocasiones es necesario dividir la tabla de información en grupos más pequeños. Para ello, hay que utilizar la cláusula `GROUP BY`.

Creación de Grupos de Datos: Sintaxis de la Cláusula GROUP BY

Puede dividir las filas de una tabla en grupos más pequeños utilizando la cláusula GROUP BY.

```
SELECT     column, group_function(column)
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[ORDER BY  column];
```

ORACLE

Creación de Grupos de Datos: Sintaxis de la Cláusula GROUP BY

Puede utilizar la cláusula GROUP BY para dividir las filas de la tabla en grupos. A continuación puede utilizar las funciones de grupo para devolver información de resumen de cada grupo.

En la sintaxis:

group_by_expression	Especifica columnas cuyos valores determinan la base para agrupar filas

Instrucciones

- Si incluye una función de grupo en una cláusula SELECT, no puede seleccionar también resultados individuales *a menos que* la columna individual aparezca en la cláusula GROUP BY. Recibirá un mensaje de error si no puede incluir la lista de columnas en la cláusula GROUP BY.
- Al utilizar la cláusula WHERE, puede excluir las filas antes de dividirlas en grupos.
- Debe incluir las *columnas* en la cláusula GROUP BY.
- No puede utilizar un alias de columna en la cláusula GROUP BY.

Uso de la Cláusula GROUP BY

Todas las columnas de la lista SELECT que no están incluidas en las funciones de grupo deben estar en la cláusula GROUP BY.

```
SELECT   department_id, AVG(salary)
FROM     employees
GROUP BY department_id ;
```

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	20	9500
3	90	19333.333333333333...
4	110	10150
5	50	3500
6	80	10033.333333333333...
7	10	4400
8	60	6400

Uso de la Cláusula GROUP BY

Al utilizar la cláusula GROUP BY, asegúrese de que todas las columnas de la lista SELECT que no están en las funciones de grupo están incluidas en la cláusula GROUP BY. El ejemplo de la diapositiva muestra el número de departamento y el salario medio de cada departamento. A continuación se muestra cómo se evalúa esta sentencia SELECT que contiene una cláusula GROUP BY:

- La cláusula SELECT especifica las columnas que se van a recuperar de la siguiente forma:
  - Columna del número de departamento de la tabla EMPLOYEES
  - Media de todos los salarios del grupo especificada en la cláusula GROUP BY
- La cláusula FROM especifica las dos tablas a las que la base de datos debe acceder: tabla EMPLOYEES.
- La cláusula WHERE especifica las filas que se van a recuperar. Ya que no existe ninguna cláusula WHERE, por defecto se recuperarán todas las filas.
- La cláusula GROUP BY especifica cómo se deben agrupar las filas. Las filas se agrupan por número de departamento, por lo tanto, la función AVG aplicada a la columna de salario calcula el salario medio de cada departamento.

Nota: para ordenar los resultados de las consultas en orden ascendente o descendente, incluya la cláusula ORDER BY en la consulta.

Uso de la Cláusula GROUP BY

No es necesario que la columna GROUP BY esté en la lista SELECT.

```
SELECT   AVG(salary)
FROM     employees
GROUP BY department_id ;
```

	AVG(SALARY)
1	7000
2	9500
3	19333.333333333333333333...
4	10150
5	3500
6	10033.333333333333333333...
7	4400
8	6400

ORACLE

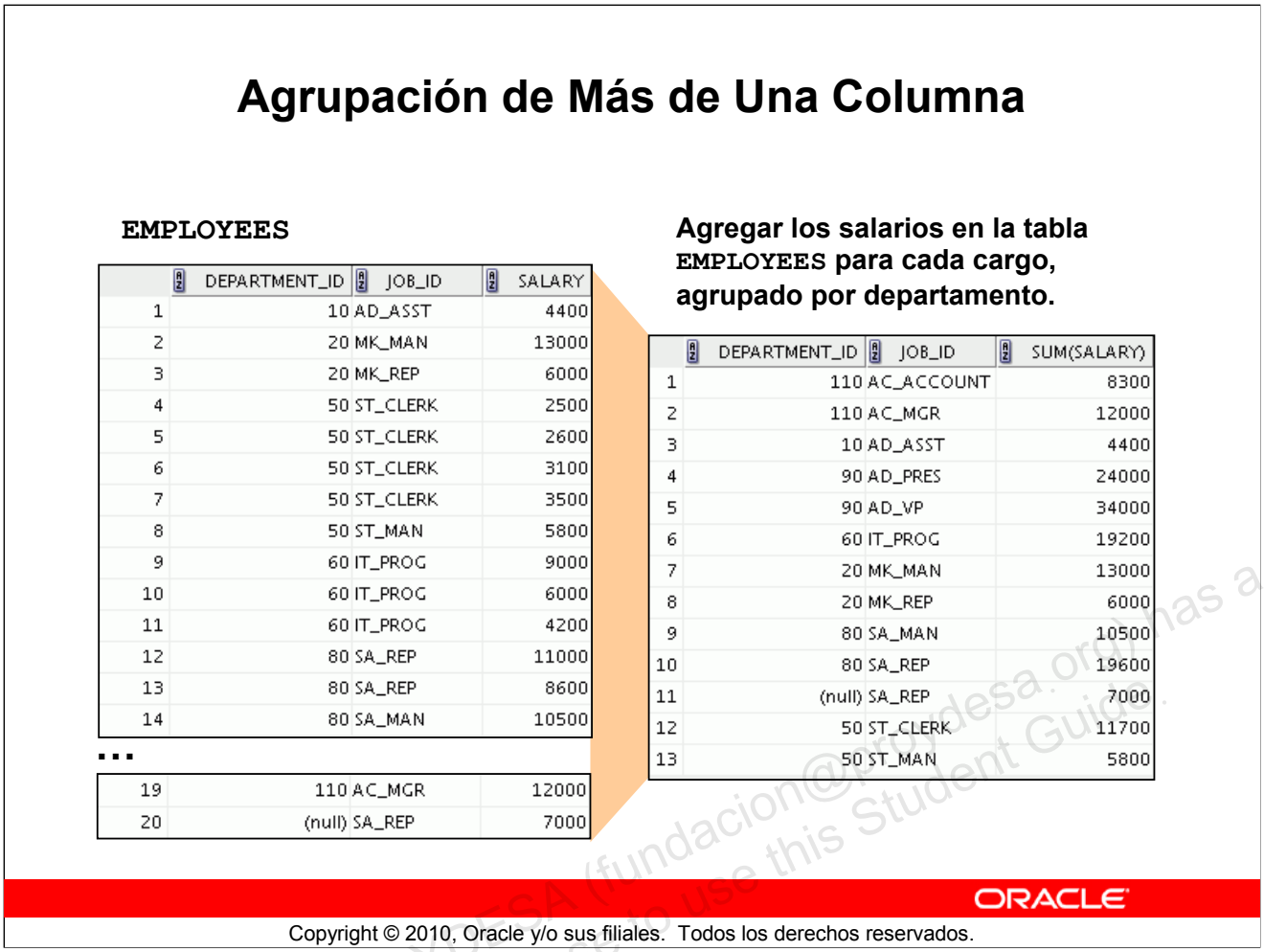
Uso de la Cláusula GROUP BY Clause (continuación)

No es necesario que la columna GROUP BY esté en la cláusula SELECT. Por ejemplo, la sentencia SELECT de la diapositiva muestra los salarios medios de cada departamento sin mostrar los respectivos números de departamento. Sin embargo, sin los números de departamento, los resultados no parecen significativos.

También puede utilizar la función de grupo en la cláusula ORDER BY:

```
SELECT   department_id, AVG(salary)
FROM     employees
GROUP BY department_id
ORDER BY AVG(salary);
```

	DEPARTMENT_ID	AVG(SALARY)
1	50	3500
2	10	4400
3	60	6400
...		
7	110	10150
8	90	19333.333333333333333333333333333333333



Agrupación de Más de Una Columna

En ocasiones necesitará ver los resultados de grupos dentro de grupos. La diapositiva muestra un informe que muestra el salario total pagado a cada puesto de cada departamento.

Según ese agrupamiento, la tabla `EMPLOYEES` se agrupa en primer lugar por número de departamento y, a continuación, por puesto. Por ejemplo, los cuatro oficinistas en el departamento de stock del departamento 50 se agrupan conjuntamente y se produce un resultado único (salario total) para todos los oficinistas en el departamento de stock del grupo.

La siguiente sentencia `SELECT` devuelve el resultado mostrado en la diapositiva:

```
SELECT   department_id, job_id, sum(salary)
FROM     employees
GROUP BY department_id, job_id
ORDER BY job_id;
```

Uso de la Cláusula GROUP BY en Varias Columnas

```
SELECT   department_id, job_id, SUM(salary)
FROM     employees
WHERE     department_id > 40
GROUP BY department_id, job_id
ORDER BY department_id;
```

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	50	ST_CLERK	11700
2	50	ST_MAN	5800
3	60	IT_PROG	19200
4	80	SA_MAN	10500
5	80	SA_REP	19600
6	90	AD_PRES	24000
7	90	AD_VP	34000
8	110	AC_ACCOUNT	8300
9	110	AC_MGR	12000

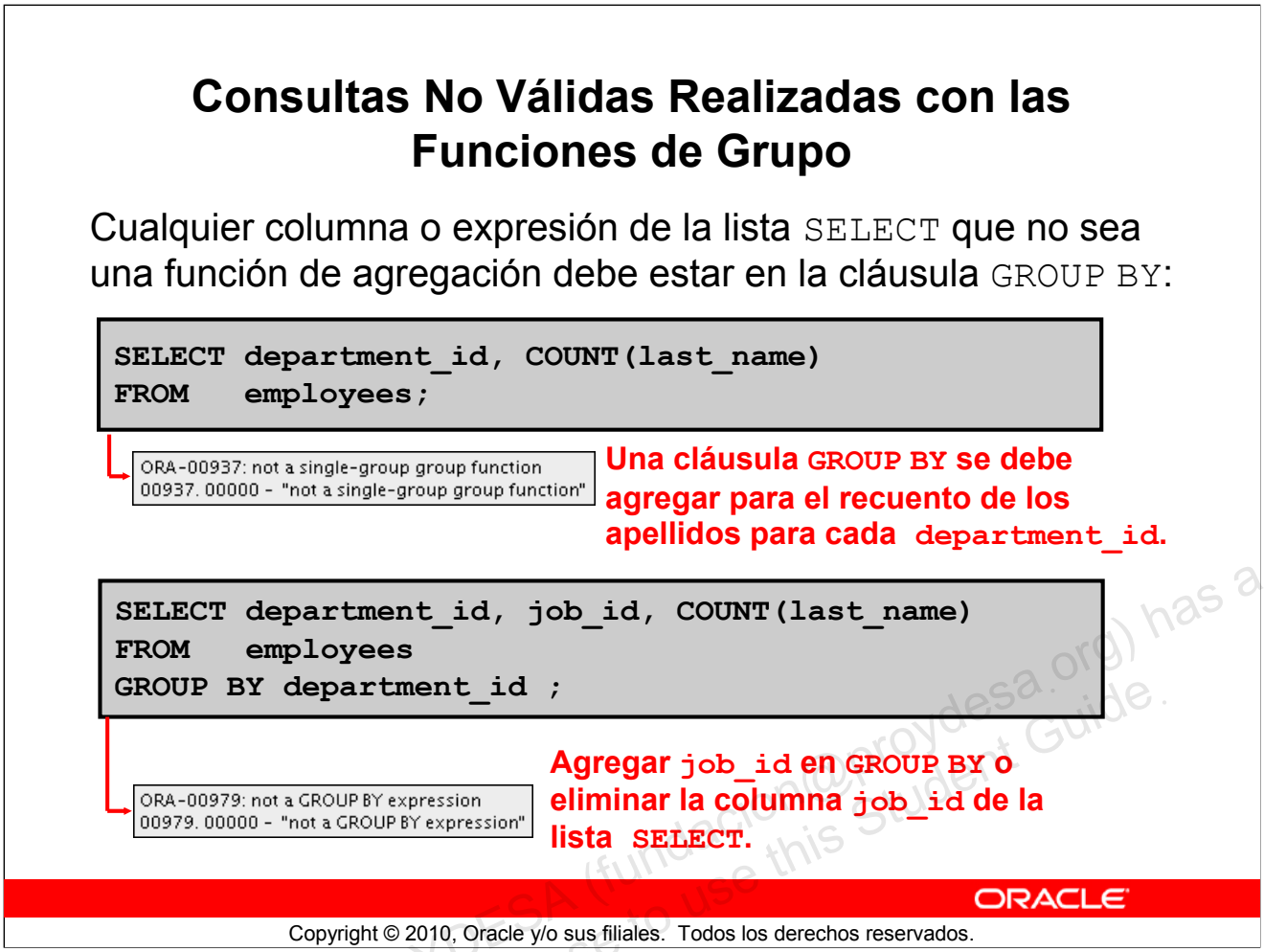
Uso de la Cláusula Group By en Varias Columnas

Puede devolver resultados de resumen para grupos y subgrupos mostrando varias columnas GROUP BY. La cláusula GROUP BY agrupa filas pero no garantiza el orden del juego de resultados. Para ordenar los agrupamientos, utilice la cláusula ORDER BY.

En el ejemplo de la diapositiva, la sentencia SELECT que contiene una cláusula GROUP BY se evalúa de la siguiente forma:

- La cláusula SELECT especifica la columna que se van a recuperar:
  - ID de departamento en la tabla EMPLOYEES
  - ID de cargo de la tabla EMPLOYEES
  - Suma de todos los salarios del grupo especificada en la cláusula GROUP BY
- La cláusula FROM especifica las dos tablas a las que la base de datos debe acceder: tabla EMPLOYEES.
- La cláusula WHERE reduce el juego de resultados a aquellas filas en las que el ID de departamento es mayor de 40.
- La cláusula GROUP BY especifica cómo debe agrupar las filas resultantes:
  - En primer lugar, las filas se agrupan por ID de departamento.
  - En segundo lugar, las filas se agrupan por ID de cargo en los grupos de ID de departamento.
- La cláusula ORDER BY ordena los resultados por ID de departamento.

Nota: la función SUM se aplica a la columna de salario de todos los ID de cargo en el juego de resultados de cada grupo de ID de departamento. Además, tenga en cuenta que la fila SA_REP no se devuelve. El ID de departamento para esta fila es NULL y, por lo tanto, no cumple la condición WHERE.



Consultas No Válidas Realizadas con las Funciones de Grupo

Cuando utilice una mezcla de elementos individual (DEPARTMENT_ID) y funciones de grupo (COUNT) en la misma sentencia SELECT debe incluir una cláusula GROUP BY que especifique los elementos individuales (en este caso, DEPARTMENT_ID). Si falta la cláusula GROUP BY, aparecerá el mensaje de error "not a single-group group function" y la columna incorrecta estará indicada con un asterisco (*). Puede corregir el error del primer ejemplo de la diapositiva agregando la cláusula GROUP BY:

```
SELECT   department_id, count(last_name)
FROM employees
GROUP BY department_id ;
```

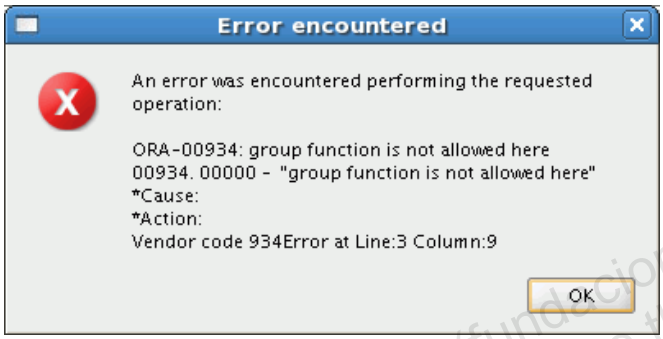
Cualquier columna o expresión de la lista SELECT que no sea una función de agregación debe estar en la cláusula GROUP BY. En el segundo ejemplo de la diapositiva, job_id no está en la cláusula GROUP BY ni la utiliza una función de grupo, por lo que se produce un error "not a GROUP BY expression". Puede corregir el error del segundo ejemplo de la diapositiva agregando job_id en la cláusula GROUP BY.

```
SELECT department_id, job_id, COUNT(last_name)
FROM   employees
GROUP BY department_id, job_id;
```

Consultas No Válidas Realizadas con las Funciones de Grupo

- No puede utilizar la cláusula `WHERE` para restringir grupos.
- Debe utilizar la cláusula `HAVING` para restringir grupos.
- No puede utilizar las funciones de grupo de la cláusula `WHERE`.

```
SELECT   department_id, AVG(salary)
FROM     employees
WHERE    AVG(salary) > 8000
GROUP BY department_id;
```



No puede utilizar la cláusula `WHERE` para restringir grupos

ORACLE

Consultas No Válidas Realizadas con las Funciones de Grupo (continuación)

No se puede utilizar la cláusula `WHERE` para restringir grupos. La sentencia `SELECT` del ejemplo de la diapositiva muestra un error porque se está utilizando la cláusula `WHERE` para restringir la visualización de los salarios medios de los departamentos que tienen un salario medio superior a 8.000 dólares.

Puede corregir el error del ejemplo utilizando la cláusula `HAVING` para restringir grupos:

```
SELECT   department_id, AVG(salary)
FROM     employees
GROUP BY department_id
HAVING   AVG(salary) > 8000;
```

	DEPARTMENT_ID	AVG(SALARY)
1	20	9500
2	90	19333.3333333333...
3	110	10150
4	80	10033.3333333333...

Restricción de Resultados de Grupo

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	2500
5	50	2600
6	50	3100
7	50	3500
8	50	5800
9	60	9000
10	60	6000
11	60	4200
12	80	11000
13	80	8600
...		
18	110	8300
19	110	12000
20	(null)	7000

Salario máximo por departamento cuando es es superior a 10.000 dólares

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

Restricción de Resultados de Grupo

Utilice la cláusula `HAVING` para restringir grupos de la misma forma que utiliza la cláusula `WHERE` para restringir las filas seleccionadas. Para buscar el salario máximo de cada uno de los departamentos que tienen un salario máximo superior a 10.000 dólares, necesitará realizar las siguientes acciones:

1. Buscar el salario medio de cada departamento realizando una agrupación por número de departamento.
2. Restringir los grupos a los departamentos con un salario máximo superior a 10.000 dólares.

Restricción de Resultados de Grupo con la Cláusula HAVING

Al utilizar la cláusula HAVING, el servidor de Oracle restringe los grupos de la siguiente forma:

1. Agrupa las filas.
2. Aplica la función de grupo.
3. Muestra los grupos que coinciden con la cláusula HAVING.

```
SELECT     column, group_function
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column];
```

ORACLE

Restricción de Resultados de Grupo con la Cláusula HAVING

Puede utilizar la cláusula HAVING para especificar los grupos que se van a mostrar y, por lo tanto, restringir los grupos según la información de agregación.

En la sintaxis, *group_condition* restringe los grupos de filas de aquellos grupos para los que la condición especificada es verdadera.

El servidor de Oracle realiza los siguientes pasos al utilizar la cláusula HAVING:

1. Agrupa las filas.
2. Aplica la función de grupo al grupo.
3. Muestra los grupos que coinciden con los criterios de la cláusula HAVING.

La cláusula HAVING puede preceder a la cláusula GROUP BY, pero se recomienda que coloque la cláusula GROUP BY primero porque es más lógico. Los grupos están formados y las funciones de grupo se calculan antes de aplicar la cláusula HAVING a los grupos de la lista SELECT.

Nota: la cláusula WHERE restringe filas, mientras que la cláusula HAVING restringe grupos.

Uso de la Cláusula HAVING

```
SELECT    department_id, MAX(salary)
FROM      employees
GROUP BY department_id
HAVING    MAX(salary)>10000 ;
```

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

ORACLE

Uso de la Cláusula HAVING

El ejemplo de la diapositiva muestra los número de departamento y los salarios máximos de los departamentos cuyo salario máximo sea superior a 10.000 dólares.

Puede utilizar la cláusula `GROUP BY` sin utilizar una función de grupo en la lista `SELECT`. Si restringe las filas según el resultado de una función de grupo, debe tener una cláusula `GROUP BY` y una cláusula `HAVING`.

El siguiente ejemplo muestra los números de departamento y los salarios medios de los departamentos cuyo salario máximo sea superior a 10.000 dólares:

```
SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY department_id
HAVING    max(salary)>10000;
```

	DEPARTMENT_ID	AVG(SALARY)
1	20	9500
2	90	19333.333333333...
3	110	10150
4	80	10033.333333333...

Uso de la Cláusula HAVING

```
SELECT   job_id, SUM(salary) PAYROLL
FROM     employees
WHERE    job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING   SUM(salary) > 13000
ORDER BY SUM(salary);
```

	JOB_ID	PAYROLL
1	IT_PROG	19200
2	AD_PRES	24000
3	AD_VP	34000

Uso de la Cláusula HAVING (continuación)

El ejemplo de la diapositiva muestra el ID de cargo y el salario mensual total de cada cargo que tiene una nómina total excedente de 13.000 dólares. El ejemplo excluye a los vendedores y ordena la lista por el salario mensual total.

Agenda

- Funciones de grupo:
  - Tipos y sintaxis
  - Uso de `AVG`, `SUM`, `MIN`, `MAX`, `COUNT`
  - Uso de la palabra clave `DISTINCT` en funciones de grupo
  - Valores `NULL` en una función de grupo
- Agrupar filas:
  - Cláusula `GROUP BY`
  - Cláusula `HAVING`
- Anidamiento de funciones de grupo

Anidamiento de Funciones de Grupo

Mostrar el salario máximo medio:

```
SELECT   MAX(AVG(salary))
FROM     employees
GROUP BY department_id;
```

	MAX(AVG(SALARY))
1	19333.333333333333333333333333333333333333

Anidamiento de Funciones de Grupo

Las funciones de grupo se pueden anidar en una profundidad de dos. El ejemplo de la diapositiva calcula el salario medio para cada `department_id` y, a continuación, muestra el salario máximo medio.

Tenga en cuenta que la cláusula `GROUP BY` es obligatoria al anidar funciones de grupo.

Prueba

Identificar las instrucciones para las funciones de grupo y la cláusula `GROUP BY`.

1. No puede utilizar un alias de columna en la cláusula `GROUP BY`.
2. La columna `GROUP BY` debe estar en la cláusula `SELECT`.
3. Al utilizar la cláusula `WHERE`, puede excluir las filas antes de dividirlas en grupos.
4. La cláusula `GROUP BY` agrupa filas y asegura el orden del juego de resultados.
5. Si incluye una función de grupo en una cláusula `SELECT`, no puede seleccionar también resultados individuales.

ORACLE

Respuesta: 1, 3

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Utilizar funciones de grupo COUNT, MAX, MIN, SUM y AVG
- Escribir consultas que utilicen la cláusula GROUP BY
- Escribir consultas que utilicen la cláusula HAVING

```
SELECT     column, group_function
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column];
```

ORACLE

Resumen

Existen diferentes funciones de grupo disponibles en SQL, como AVG, COUNT, MAX, MIN, SUM, STDDEV y VARIANCE.

Puede crear subgrupos utilizando la cláusula GROUP BY. Además, los grupos se pueden restringir utilizando la cláusula HAVING.

Coloque las cláusulas HAVING y GROUP BY después de la cláusula WHERE en una sentencia. El orden de las cláusulas GROUP BY y HAVING después de la cláusula WHERE no es importante. Coloque la cláusula ORDER BY al final.

El servidor de Oracle evalúa las cláusulas en el siguiente orden:

1. Si la sentencia contiene una cláusula WHERE, el servidor establece las filas candidatas.
2. El servidor identifica los grupos que están especificados en la cláusula GROUP BY.
3. La cláusula HAVING restringe aún más los grupos de resultados que no cumplen con los criterios de grupo en la cláusula HAVING.

Nota: para obtener una lista completa de las funciones de grupo, consulte *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Práctica 5: Visión General

En esta práctica se abordan los siguientes temas:

- Escritura de consultas que utilizan las funciones de grupo
- Agrupación por filas para obtener más de un resultado
- Restricción de grupos utilizando la cláusula `HAVING`

ORACLE

Práctica 5: Visión General

En esta práctica, aprenderá a utilizar funciones de grupo y seleccionar grupos de datos.

6

Visualización de Datos de Varias Tablas Utilizando Uniones

ORACLE

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Escribir sentencias `SELECT` para acceder a datos de más de una tabla mediante uniones igualitarias y no igualitarias
- Unir una tabla consigo misma mediante autounión
- Ver datos que normalmente no cumplen una condición de unión mediante uniones `OUTER`
- Generar un producto cartesiano de todas las filas de una o más tablas

ORACLE

Objetivos

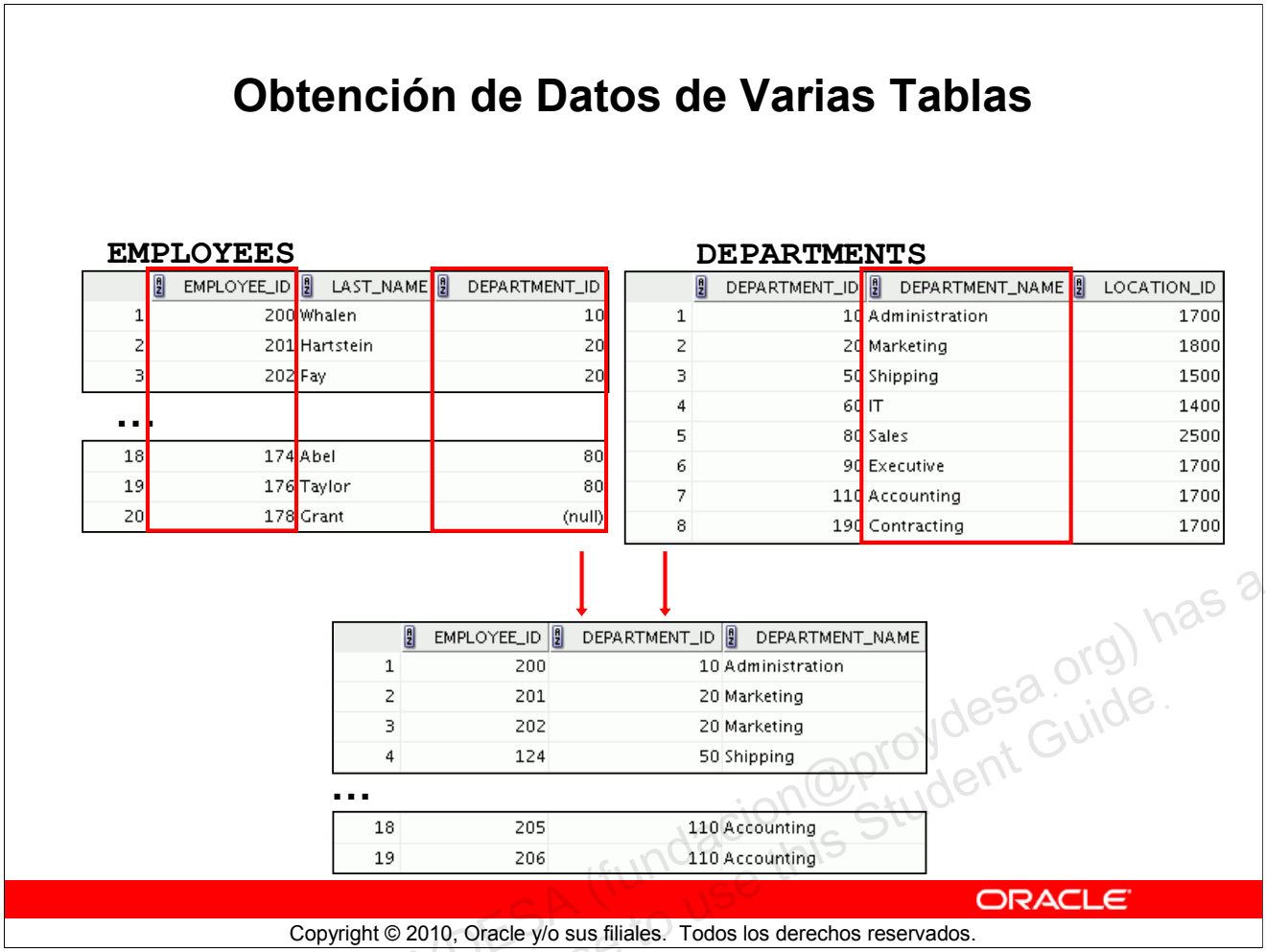
Esta lección explica cómo obtener datos de más de una tabla. Una *unión* se utiliza para ver información de varias tablas. Por lo tanto, puede *unir* tablas para ver información de más de una tabla.

Nota: para obtener más información sobre las uniones, consulte la sección sobre consultas y subconsultas SQL: uniones en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Uniones no igualitarias
- Unión OUTER:
  - Unión LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada

ORACLE



Obtención de Datos de Varias Tablas

A veces necesita utilizar datos de más de una tabla. En el ejemplo de la diapositiva, el informe muestra datos de dos tablas independientes:

- La tabla `EMPLOYEES` contiene los ID de empleado.
- Las tablas `EMPLOYEES` y `DEPARTMENTS` contienen los ID de departamento.
- La tabla `DEPARTMENTS` contiene los nombres de departamento.

Para producir el informe, necesita enlazar las tablas `EMPLOYEES` y `DEPARTMENTS` y acceder a los datos de ambas.

Tipos de Uniones

Las uniones compatibles con el estándar SQL:1999 incluyen los siguientes elementos:

- Uniones naturales:
  - Cláusula `NATURAL JOIN`
  - Cláusula `USING`
  - Cláusula `ON`
- Uniones`OUTER`:
  - `LEFT OUTER JOIN`
  - `RIGHT OUTER JOIN`
  - `FULL OUTER JOIN`
- Uniones cruzadas

ORACLE

Tipos de Uniones

Para unir tablas, puede utilizar la sintaxis de unión que cumpla con el estándar SQL:1999.

Nota

- En versiones anteriores a Oracle9*i* la sintaxis de unión era diferente a los estándares American National Standards Institute (ANSI). La sintaxis de unión compatible con SQL:1999 no ofrece ninguna ventaja en el rendimiento respecto a la sintaxis de unión propietaria de Oracle que existía en las versiones anteriores. Para obtener más información sobre la sintaxis de unión propiedad de Oracle, consulte el apéndice F: Sintaxis de Unión en Oracle.
- En la siguiente diapositiva se trata la sintaxis de unión compatible con SQL:1999.

Unión de Tablas mediante la Sintaxis SQL:1999

Utilizar una unión para consultar datos de más de una tabla:

```
SELECT   table1.column, table2.column
FROM     table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)]|
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)]|
[CROSS JOIN table2];
```

ORACLE

Unión de Tablas mediante la Sintaxis SQL:1999

En la sintaxis:

- `table1.column` indica la tabla y la columna desde las que se recuperan los datos
- `NATURAL JOIN` une dos tablas basándose en el mismo nombre de la columna
- `JOIN table2 USING column_name` realiza una unión igualitaria basándose en el nombre de la columna
- `JOIN table2 ON table1.column_name = table2.column_name performs` realiza una unión igualitaria basándose en la condición de la cláusula `ON`
- `LEFT/RIGHT/FULL OUTER` se utiliza para realizar uniones `OUTER`
- `CROSS JOIN` devuelve un producto cartesiano de las dos tablas

Para obtener más información, consulte la sección sobre `SELECT` en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Cualificación de Nombres de Columna Ambiguos

- Utilizar prefijos de tabla para cualificar los nombres de columna que están en varias tablas.
- Utilizar prefijos de tabla para mejorar el rendimiento.
- En lugar de prefijos de nombre de tabla completos, utilizar alias de tabla.
- Los alias de tablas proporciona un nombre más corto de una tabla:
  - Mantiene el código SQL más pequeño, utiliza menos memoria
- Utilizar alias de columna para distinguir columnas que tienen nombres idénticos, pero que residen en diferentes tablas.

Cualificación de Nombres de Columna Ambiguos

Al unir dos o más tablas, debe cualificar los nombres de las columnas con el nombre de la tabla para evitar ambigüedad. Sin los prefijos de tabla, la columna `DEPARTMENT_ID` de la lista `SELECT` puede provenir de la tabla `DEPARTMENTS` o de la tabla `EMPLOYEES`. Es necesario agregar el prefijo de tabla para ejecutar la consulta. Si no existen nombres de columna comunes entre las dos tablas, no es necesario cualificar las columnas. Sin embargo, el uso del prefijo de tabla mejora el rendimiento, ya que indica al servidor de Oracle dónde encontrar exactamente las columnas.

Sin embargo, la cualificación de nombres de columna con nombres de tabla puede llevar bastante tiempo, especialmente si los nombres de tabla son largos. En su lugar, puede utilizar *alias de tabla*. Igual que un alias de columna proporciona otro nombre a una columna, un alias de tabla proporciona otro nombre a una tabla. Los alias de tabla ayudan a mantener el código SQL más pequeño y, por lo tanto, menos uso de memoria

El nombre de tabla se especifica por completo, seguido de un espacio y del alias de tabla. Por ejemplo, a la tabla `EMPLOYEES` se le puede proporcionar el alias `e` y, a la tabla `DEPARTMENTS` el alias `d`.

Instrucciones

- Los alias de tabla pueden tener hasta 30 caracteres de longitud, pero los alias más cortos son mejores que los largos.
- Si se utiliza un alias de tabla para un nombre de tabla determinado en la cláusula `FROM`, el alias de tabla se deberá sustituir por el nombre de tabla mediante la sentencia `SELECT`.
- Los alias de tabla deben ser significativos.
- El alias de tabla es válido sólo para la sentencia actual `SELECT`.

Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Uniones no igualitarias
- Unión OUTER:
  - Unión LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada

ORACLE

Creación de Uniones Naturales

- La cláusula `NATURAL JOIN` está basada en todas las columnas de las dos tablas que tienen el mismo nombre.
- Selecciona filas de las dos tablas que tienen valores iguales en todas las columnas coincidentes.
- Si las columnas que tienen el mismo nombre tienen tipos de dato diferentes, se devolverá un error.

ORACLE

Creación de Uniones Naturales

Puede unir tablas automáticamente basándose en las columnas de las dos tablas que tienen el mismo nombre y los mismos tipos de dato. Puede realizar esta acción utilizando las palabras clave `NATURAL JOIN`.

Nota: la unión se puede producir únicamente en las columnas que tienen los mismos nombres y los mismos tipos de dato en ambas tablas. Si las columnas tienen el mismo nombre pero tipos de dato diferentes, la sintaxis `NATURAL JOIN` produce un error.

Recuperación de Registros con Uniones Naturales

```
SELECT  department_id, department_name,
        location_id, city
FROM    departments
NATURAL JOIN locations ;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford

ORACLE

Recuperación de Registros con Uniones Naturales

En el ejemplo de la diapositiva, la tabla `LOCATIONS` está unida a la tabla `DEPARTMENT` mediante la columna `LOCATION_ID`, que es la única con el mismo nombre en ambas. Si están presentes otras columnas comunes, la unión debe haberlas utilizado todas.

Uniones Naturales con la Cláusula `WHERE`

Las restricciones adicionales en una unión natural se implantan utilizando una cláusula `WHERE`. El siguiente ejemplo limita las filas de salida a las que tienen un ID de departamento igual a 20 o a 50:

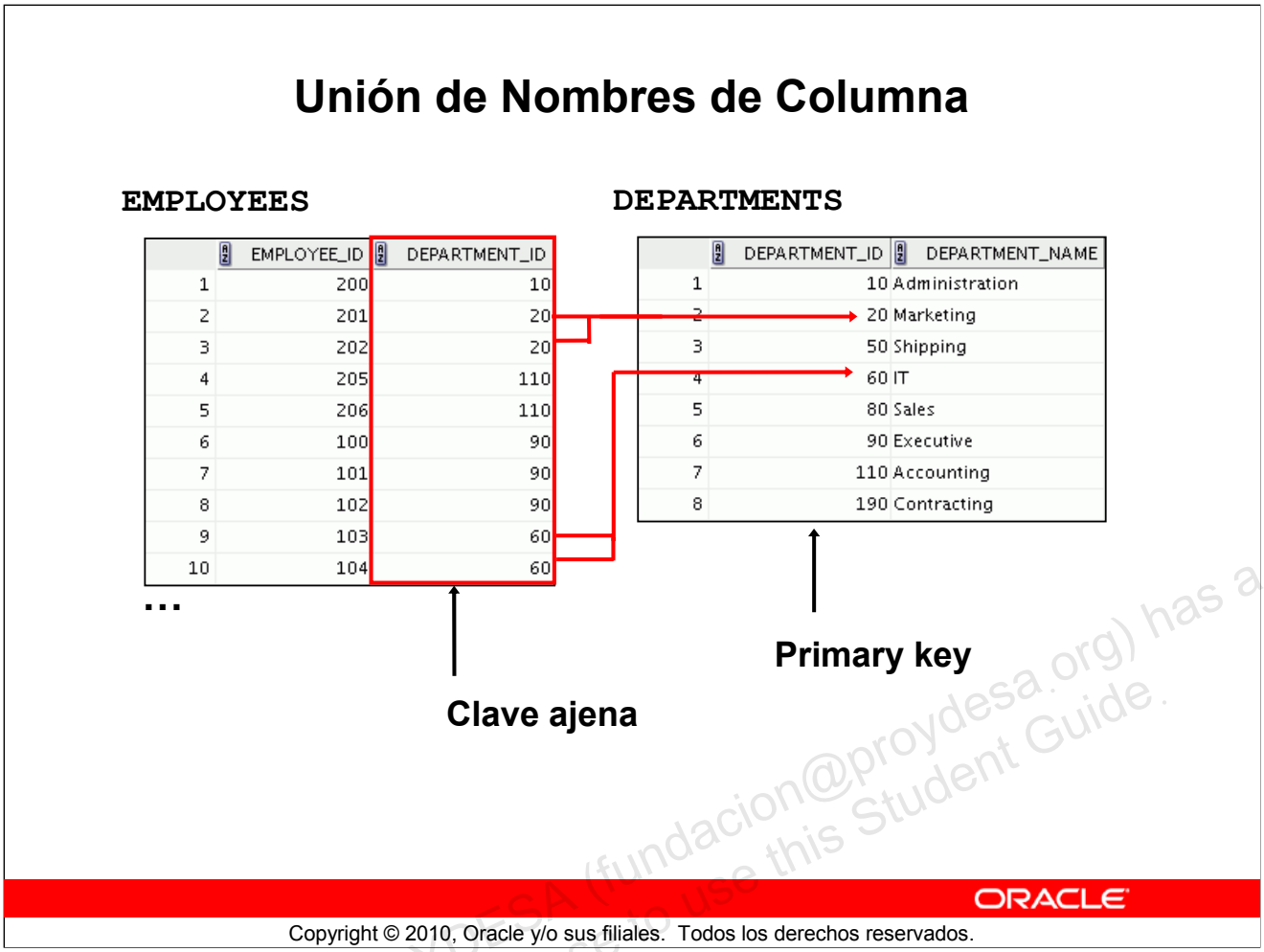
```
SELECT  department_id, department_name,
        location_id, city
FROM    departments
NATURAL JOIN locations
WHERE   department_id IN (20, 50);
```

Creación de Uniones con la Cláusula `USING`

- Si varias columnas tienen el mismo nombre pero los tipos de dato no coinciden, utilizar la cláusula `USING` para especificar las columnas para la unión igualitaria.
- Utilizar `USING` para que sólo coincida una columna en caso de que coincida más de una.
- Las cláusulas `NATURAL JOIN` y `USING` se excluyen mutuamente.

Creación de Uniones con la Cláusula `USING`

Las uniones naturales utilizan todas las columnas con tipos de dato y nombres coincidentes para unir las tablas. La cláusula `USING` se puede utilizar para especificar sólo las columnas que se deben utilizar para una unión igualitaria.



Unión de Nombres de Columna

Para determinar el nombre de departamento de un empleado, compare el valor de la columna `DEPARTMENT_ID` en la tabla `EMPLOYEES` con los valores de `DEPARTMENT_ID` de la tabla `DEPARTMENTS`. La relación entre las tablas `EMPLOYEES` y `DEPARTMENTS` es una *unión igualitaria*; es decir, los valores de la columna `DEPARTMENT_ID` de ambas tablas deben ser iguales. Normalmente, este tipo de unión implica complementos de clave primaria y ajena.

Nota: las uniones igualitarias también se denominan *uniones simples* o *uniones internas*.

Recuperación de Registros con la Cláusula USING

```
SELECT employee_id, last_name,
       location_id, department_id
FROM   employees JOIN departments
USING (department_id) ;
```

	EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	144	Vargas	1500	50
5	143	Matos	1500	50
6	142	Davies	1500	50
7	141	Rajs	1500	50
8	124	Mourgos	1500	50

...

18	206	Gietz	1700	110
19	205	Higgins	1700	110

ORACLE

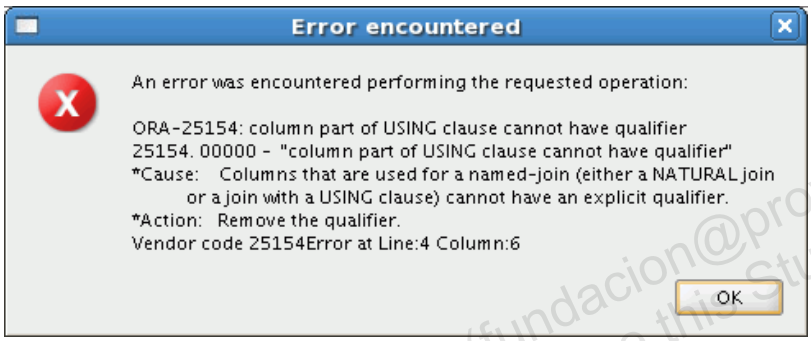
Recuperación de Registros con la Cláusula USING

En el ejemplo de la diapositiva, las columnas DEPARTMENT_ID de las tablas EMPLOYEES y DEPARTMENTS se unen y, por lo tanto, se muestra el LOCATION_ID del departamento en el que trabaja el empleado.

Uso de Alias de Tabla con la Cláusula USING

- No cualificar una columna que se utilice en la cláusula USING.
- Si la misma columna se utiliza en otro lugar de la sentencia SQL, no se le puede agregar un alias.

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d
USING (location_id)
WHERE d.location_id = 1400;
```



ORACLE

Uso de Alias de Tabla con la Cláusula USING

Al realizar una unión con la cláusula USING, no puede cualificar una columna que se utiliza en la cláusula USING en sí. Además, si esa columna se utiliza en cualquier ubicación de la sentencia SQL, no se le puede agregar ningún alias. Por ejemplo, en la consulta mencionada en la diapositiva, no debe agregar un alias a la columna location_id en la cláusula WHERE porque la columna se utiliza en la cláusula USING.

Las columnas a las que se hace referencia en la cláusula USING no deben tener un cualificador (nombre o alias de la tabla) en cualquier ubicación de la sentencia SQL. Por ejemplo, la siguiente sentencia es válida:

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d USING (location_id)
WHERE  location_id = 1400;
```

Las columnas comunes en ambas tablas, pero que no se utilizan en la cláusula USING, se deben prefijar con un alias de tabla, de lo contrario aparecerá el error “column ambiguously defined”.

En la siguiente sentencia , manager_id está presente tanto la tabla employees como en departments ; si no se prefija manager_id con un alias de tabla, se produce el error “column ambiguously defined”.

El siguiente ejemplo es válido:

```
SELECT first_name, d.department_name, d.manager_id
FROM   employees e JOIN departments d USING (department_id)
WHERE  department_id = 50;
```

Creación de Uniones con la Cláusula ON

- La condición de unión de la unión natural es básicamente una unión igualitaria de todas las columnas con el mismo nombre.
- Utilizar la cláusula ON para especificar condiciones arbitrarias o columnas que se van a unir.
- La condición de unión está separada de otras condiciones de búsqueda.
- La clausula ON facilita la comprensión del código.

Creación de Uniones con la Cláusula ON

Utilice la sentencia ON para especificar una condición de unión. Esto permite especificar condiciones de unión independientes de cualquier condición de filtro o de búsqueda en la cláusula WHERE.

Recuperación de Registros con la Cláusula ON

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id);
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200	Whalen	10	10	1700
2	201	Hartstein	20	20	1800
3	202	Fay	20	20	1800
4	144	Vargas	50	50	1500
5	143	Matos	50	50	1500
6	142	Davies	50	50	1500
7	141	Rajs	50	50	1500
8	124	Mourgos	50	50	1500
9	103	Hunold	60	60	1400
10	104	Ernst	60	60	1400
11	107	Lorentz	60	60	1400

...

ORACLE

Recuperación de Registros con la Cláusula ON

En este ejemplo, las columnas `DEPARTMENT_ID` en la tabla `EMPLOYEES` y `DEPARTMENTS` se unen con la cláusula `ON`. Cuando un ID de departamento de la tabla `EMPLOYEES` sea igual al ID de departamento de la tabla `DEPARTMENTS`, se devolverá una fila. El alias de tabla es necesario para cualificar los `column_names` coincidentes.

También puede utilizar la cláusula `ON` para unir columnas que tienen nombres diferentes. Los paréntesis de las columnas unidas, como se muestra en el ejemplo de la diapositiva, `(e.department_id = d.department_id)` son opcionales. Incluso, `ON e.department_id = d.department_id` funcionará.

Nota: al utilizar el icono Execute Statement para ejecutar la consulta, SQL Developer agrega el sufijo "_1" para diferenciar entre los dos `department_ids`.

Creación de Uniones en 3 Direcciones con la Cláusula ON

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

	EMPLOYEE_ID	CITY	DEPARTMENT_NAME
1	100	Seattle	Executive
2	101	Seattle	Executive
3	102	Seattle	Executive
4	103	Southlake	IT
5	104	Southlake	IT
6	107	Southlake	IT
7	124	South San Francisco	Shipping
8	141	South San Francisco	Shipping
9	142	South San Francisco	Shipping

...

ORACLE

Creación de Uniones en 3 Direcciones con la Cláusula ON

Una unión en tres direcciones es una unión de tres tablas. En la sintaxis compatible con SQL:1999, las uniones se realizan de izquierda a derecha. Por lo tanto, la primera unión que se realiza es `EMPLOYEES JOIN DEPARTMENTS`. La primera condición de unión puede hacer referencia a las columnas de `EMPLOYEES` y `DEPARTMENTS`, pero no puede hacer referencia a las columnas de `LOCATIONS`. La segunda condición de unión puede hacer referencia a las columnas de las tres tablas.

Nota: el ejemplo de código de la diapositiva también se puede realizar con la cláusula `USING`:

```
SELECT e.employee_id, l.city, d.department_name
FROM employees e
JOIN   departments d
USING (department_id)
JOIN   locations l
USING (location_id)
```

Aplicación de Condiciones Adicionales a una Unión

Uso de la cláusula AND o la cláusula WHERE para aplicar condiciones adicionales:

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
AND    e.manager_id = 149 ;
```

O bien

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
WHERE   e.manager_id = 149 ;
```

ORACLE

Aplicación de Condiciones Adicionales a una Unión

Puede aplicar condiciones adicionales a la unión.

El ejemplo realiza una unión en las tablas EMPLOYEES y DEPARTMENTS y, además, muestra sólo los empleados que tienen un ID de gestor de 149. Para agregar condiciones adicionales a la cláusula ON, puede agregar cláusulas AND. Por otro lado, puede utilizar una cláusula WHERE para aplicar condiciones adicionales.

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	174	Abel	80	80	2500
2	176	Taylor	80	80	2500

Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Uniones no igualitarias
- Unión OUTER:
  - Cláusula LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada

ORACLE

Unión de una Tabla consigo Misma

EMPLOYEES (WORKER)

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
200	Whalen	101
201	Hartstein	100
202	Fay	201
205	Higgins	101
206	Gietz	205
100	King	(null)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103

...

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
200	Whalen
201	Hartstein
202	Fay
205	Higgins
206	Gietz
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst

...

MANAGER_ID en la tabla WORKER es igual a EMPLOYEE_ID en la tabla MANAGER.

ORACLE

Unión de una Tabla consigo Misma

Puede que a veces necesite unir una tabla consigo misma. Para buscar el nombre de cada gestor del empleado, necesita unir la tabla `EMPLOYEES` consigo misma o realizar una autounión. Por ejemplo, para buscar el nombre del gestor de Lorentz, necesita:

- Buscar a Lorentz en la tabla `EMPLOYEES` buscando en la columna `LAST_NAME`.
- Busque el número de gestor de Lorentz consultando la columna `MANAGER_ID`. El número de gestor de Lorentz es 103.
- Busque el nombre del gestor con un `EMPLOYEE_ID` de 103 consultando la columna `LAST_NAME`. El número de empleado de Hunold es 103, por lo que Hunold en el gestor de Lorentz.

Durante este proceso, busca dos veces en la tabla. La primera vez, cuando consulta la tabla para buscar a Lorentz en la columna `LAST_NAME` y el valor de `MANAGER_ID` de 103. La segunda vez, cuando busca en la columna `EMPLOYEE_ID` para buscar 103 y en la columna `LAST_NAME` para buscar a Hunold.

Autouniones que Utilizan la Cláusula ON

```
SELECT worker.last_name emp, manager.last_name mgr
FROM   employees worker JOIN employees manager
ON     (worker.manager_id = manager.employee_id);
```

	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King

...

Autouniones que Utilizan la Cláusula ON

La cláusula ON también se puede utilizar para unir columnas con nombres diferentes, ya sea en la misma tabla o en otra diferente.

En el ejemplo se muestra una autounión de la tabla EMPLOYEES basada en las columnas EMPLOYEE_ID y MANAGER_ID.

Nota: los paréntesis de las columnas unidas, como se muestra en la diapositiva (e.manager_id = m.employee_id) son **opcionales**. Incluso ON e.manager_id = m.employee_id funcionará.

Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Uniones no igualitarias
- Unión OUTER:
  - Cláusula LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada

ORACLE

Uniones no igualitarias

EMPLOYEES

	LAST_NAME	SALARY
1	Whalen	4400
2	Hartstein	13000
3	Fay	6000
4	Higgins	12000
5	Gietz	8300
6	King	24000
7	Kochhar	17000
8	De Haan	17000
9	Hunold	9000
10	Ernst	6000
...		
19	Taylor	8600
20	Grant	7000

JOB_GRADES

	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
1	A	1000	2999
2	B	3000	5999
3	C	6000	9999
4	D	10000	14999
5	E	15000	24999
6	F	25000	40000

JOB_GRADES define el rango de valores de LOWEST_SAL y HIGHEST_SAL de cada GRADE_LEVEL. Por lo tanto, la columna GRADE_LEVEL se puede utilizar para asignar grados a cada empleado.

Uniones no igualitarias

Una unión no igualitaria es una condición de unión que contiene algún operador diferente del operador de igualdad.

La relación entre la tabla `EMPLOYEES` y `JOB_GRADES` es un ejemplo de unión no igualitaria. Los rangos de la columna `SALARY` en la tabla `EMPLOYEES` entre los valores en las columnas `LOWEST_SAL` y `HIGHEST_SAL` de la tabla `JOB_GRADES`. Por lo tanto, se pueden agregar grados a cada empleado según el salario. La relación se obtiene mediante un operador distinto del de igualdad (=).

Recuperación de Registros con Uniones no Igualitarias

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Mourgos	5800	B
8	Ernst	6000	C
9	Fay	6000	C
10	Grant	7000	C

...

Recuperación de Registros con Uniones no Igualitarias

La diapositiva del ejemplo crea una unión no igualitaria para evaluar el grado de salario de un empleado. El salario debe estar *entre* cualquier par de los rangos de salario bajos y altos.

Es importante tener en cuenta que todos los empleados aparecerán sólo una vez al ejecutar esta consulta. No se repite ningún empleado en la lista. Existen dos motivos por los que se produce este hecho:

- Ninguna de las filas de la tabla `JOB_GRADES` contiene grados que se solapen. Es decir, el valor de salario de un empleado sólo puede oscilar entre los valores de salario bajo y alto de una de las filas de la tabla de grados de salario.
- Todos los salarios de los empleados oscilan entre los límites proporcionados por la tabla de grados de cargo. Es decir, ningún empleado gana menos que el valor más bajo de la columna `LOWEST_SAL` o más que el valor más alto de la columna `HIGHEST_SAL`.

Nota: se pueden utilizar otras condiciones (como <= y >=) , pero `BETWEEN` es la más simple. Recuerde especificar primero el valor bajo y, a continuación, el alto al utilizar `BETWEEN`. El servidor Oracle convierte la condición `BETWEEN` en un par de condiciones `AND`. Por lo tanto, el uso de `BETWEEN` no proporciona ninguna ventaja, se recomienda sólo para la simplicidad lógica.

En el ejemplo de la diapositiva se han especificado los alias de tabla por motivos de rendimiento, no por una posible ambigüedad.

Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Uniones no igualitarias
- Unión OUTER:
  - Cláusula LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada

Devolución de Registros sin Coincidencia Directa con las uniones OUTER

DEPARTMENTS

	DEPARTMENT_NAME	DEPARTMENT_ID
1	Administration	10
2	Marketing	20
3	Shipping	50
4	IT	60
5	Sales	80
6	Executive	90
7	Accounting	110
8	Contracting	190

Uniones Igualitarias con EMPLOYEES

	DEPARTMENT_ID	LAST_NAME
1	10	Whalen
2	20	Hartstein
3	20	Fay
4	110	Higgins
5	110	Gietz
6	90	King
7	90	Kochhar
8	90	De Haan
9	60	Hunold
10	60	Ernst
...		
18	80	Abel
19	80	Taylor

No hay ningún empleado en el departamento 190.

Al empleado "Grant" no se le ha asignado un ID departamento.

ORACLE

Devolución de Registros sin Coincidencia Directa con Uniones OUTER

Si una fila no cumple una condición de unión, la fila no aparece en el resultado de consultas.

En el ejemplo de la diapositiva, se utiliza una condición de unión igualitaria en las tablas EMPLOYEES y DEPARTMENTS para devolver el resultado a la derecha. El juego de resultados no contiene lo siguiente:

- El ID de departamento 190, porque no hay ningún empleado con dicho ID registrado en la tabla EMPLOYEES.
- El empleado con apellido Grant, porque a este empleado no se ha asignado un ID departamento.

Para volver al registro de departamento que no tiene ningún empleado o empleados que no tienen un departamento asignado, puede utilizar la unión OUTER.

Uniones INNER frente a Uniones OUTER

- En SQL:1999, la unión de dos tablas que devuelven sólo filas coincidentes se denomina unión INNER.
- Una unión entre dos tablas que devuelve los resultados de la unión INNER y las filas no coincidentes de las tablas izquierda (o derecha) se denomina una unión OUTER.
- Una unión entre dos tablas que devuelve los resultados de una unión INNER y los resultados de una unión izquierda y derecha da como resultado una unión OUTER completa.

ORACLE

Uniones INNER frente a Uniones OUTER

La unión de las tablas con cláusulas NATURAL JOIN, USING u ON da como resultado una unión INNER. Cualquier fila no coincidente no aparece en la salida. Para devolver las filas no coincidentes, puede utilizar una unión OUTER. Una unión OUTER devuelve todas las filas que cumplen la condición de unión y también algunas o todas las filas de una tabla para la que ninguna fila de la otra tabla cumple la condición de unión.

Hay tres tipos de uniones OUTER:

- LEFT OUTER
- RIGHT OUTER
- FULL OUTER

LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON     (e.department_id = d.department_id);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping
...			
16	Kochhar	90	Executive
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

ORACLE

LEFT OUTER JOIN

Esta consulta recupera todas las filas de la tabla EMPLOYEES que es la tabla de la izquierda, incluso si no hay ninguna coincidencia en la tabla DEPARTMENTS.

RIGHT OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting
20	(null)	190	Contracting

ORACLE

RIGHT OUTER JOIN

Esta consulta recupera todas las filas de la tabla `DEPARTMENTS` que es la tabla de la izquierda, incluso si no hay ninguna coincidencia en la tabla `EMPLOYEES`.

FULL OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON     (e.department_id = d.department_id);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting
...			
17	Zlotkey	80	Sales
18	Abel	80	Sales
19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

ORACLE

FULL OUTER JOIN

Esta consulta recupera todas las filas de la tabla `EMPLOYEES`, incluso si no hay ninguna coincidencia en la tabla `DEPARTMENTS`. También recupera todas las filas de la tabla `DEPARTMENTS`, incluso si no hay ninguna coincidencia en la tabla `EMPLOYEES`.

Agenda

- Tipos de JOINS y sintaxis
- Unión natural:
  - Cláusula USING
  - Cláusula ON
- Autounión
- Unión no igualitaria
- Unión OUTER:
  - Cláusula LEFT OUTER
  - Unión RIGHT OUTER
  - Unión FULL OUTER
- Producto cartesiano
  - Unión cruzada

ORACLE

Productos Cartesianos

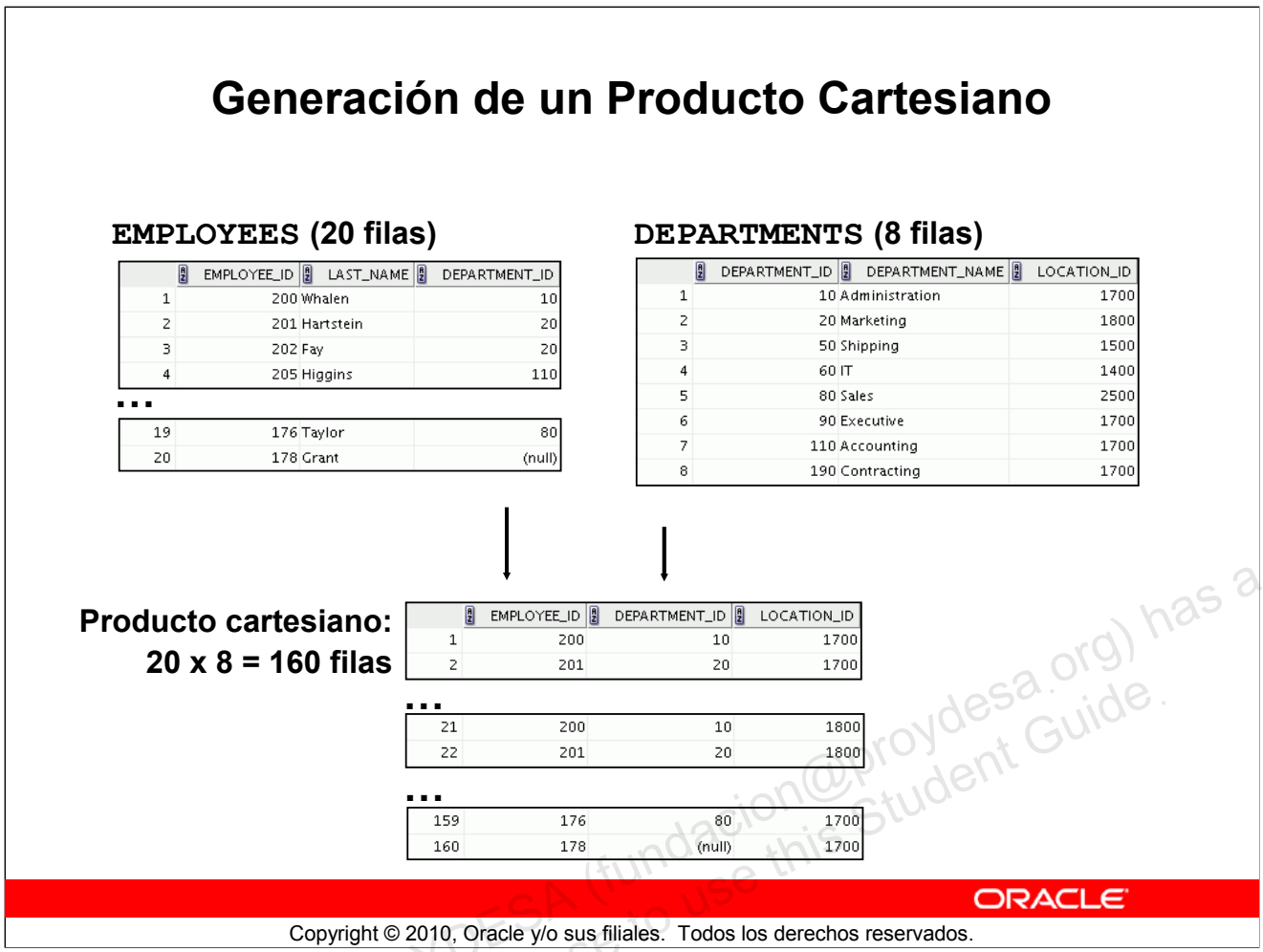
- Un producto cartesiano se forma cuando:
  - Se omite una condición de unión
  - Una condición de unión no es válida
  - Todas las filas de la primera tabla se unen a todas las filas de la segunda tabla
- Se incluye siempre una condición de unión válida si desea evitar un producto cartesiano.

Productos Cartesianos

Cuando una condición de unión no es válida o se omite completamente, el resultado es un *producto cartesiano*, en el que se muestran todas las combinaciones de filas. Todas las filas de la primera tabla se unen a todas las filas de la segunda tabla.

Un producto cartesiano tiende a generar un gran número de filas y el resultado es poco útil. Debe incluir siempre una condición de unión válida a menos que tenga necesidades específicas de combinar todas las filas de todas las tablas.

Los productos cartesianos son útiles para algunas pruebas para las que necesite generar un gran número de filas para simular una cantidad razonable de datos.



Generación de un Producto Cartesiano

Se genera un producto cartesiano si se omite una condición de unión. En el ejemplo de la diapositiva se muestra el apellido del empleado y el nombre del departamento de las tablas `EMPLOYEES` y `DEPARTMENTS`. Ya que no se ha especificado ninguna condición de unión, todas las filas (20 filas) de la tabla `EMPLOYEES` se unen con todas las filas (8 filas) de la tabla `DEPARTMENTS` por lo que se generan 160 filas en la salida.

Creación de Uniones Cruzadas

- La cláusula `CROSS JOIN` produce el producto combinado de dos tablas.
- Esto también se denomina un producto cartesiano entre dos tablas.

```
SELECT last_name, department_name
FROM   employees
CROSS JOIN departments ;
```

	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration
...		
158	Vargas	Contracting
159	Whalen	Contracting
160	Zlotkey	Contracting

ORACLE

Creación de Uniones Cruzadas

El ejemplo de la diapositiva produce un producto cartesiano de las tablas `EMPLOYEES` y `DEPARTMENTS`.

La técnica `CROSS JOIN` se puede aplicar a muchas situaciones de forma útil. Por ejemplo, para devolver el costo de mano de obra total por oficina por mes, incluso si el mes X no tiene costo de mano de obra, puede realizar un unión cruzada de oficinas con la tabla de todos los meses.

Es una práctica aconsejable declarar de forma explícita `CROSS JOIN` en `SELECT` si desea crear un producto cartesiano. Por lo tanto, queda muy claro que desea que esto ocurra y que no se trata del resultado de las uniones que faltan

Prueba

La sintaxis de unión estándar SQL:1999 soporta los siguientes tipos de uniones. ¿Qué tipos de unión no soporta la sintaxis de unión de Oracle?

1. Uniones igualitarias
2. Uniones no igualitarias
3. Unión `OUTER` izquierda
4. Unión `OUTER` derecha
5. Unión `OUTER` completa
6. Autouniones
7. Uniones naturales
8. Productos cartesianos

ORACLE

Respuesta: 1, 2, 3, 4, 6, 8

Resumen

En esta lección debe haber aprendido a utilizar uniones para mostrar los datos de varias tablas utilizando:

- Uniones igualitarias
- Uniones no igualitarias
- Uniones `OUTER`
- Autouniones
- Uniones cruzadas
- Uniones naturales
- Unión `OUTER` completa (o de dos lados)

ORACLE

Resumen

Hay varios modos de unir tablas.

Tipos de Uniones

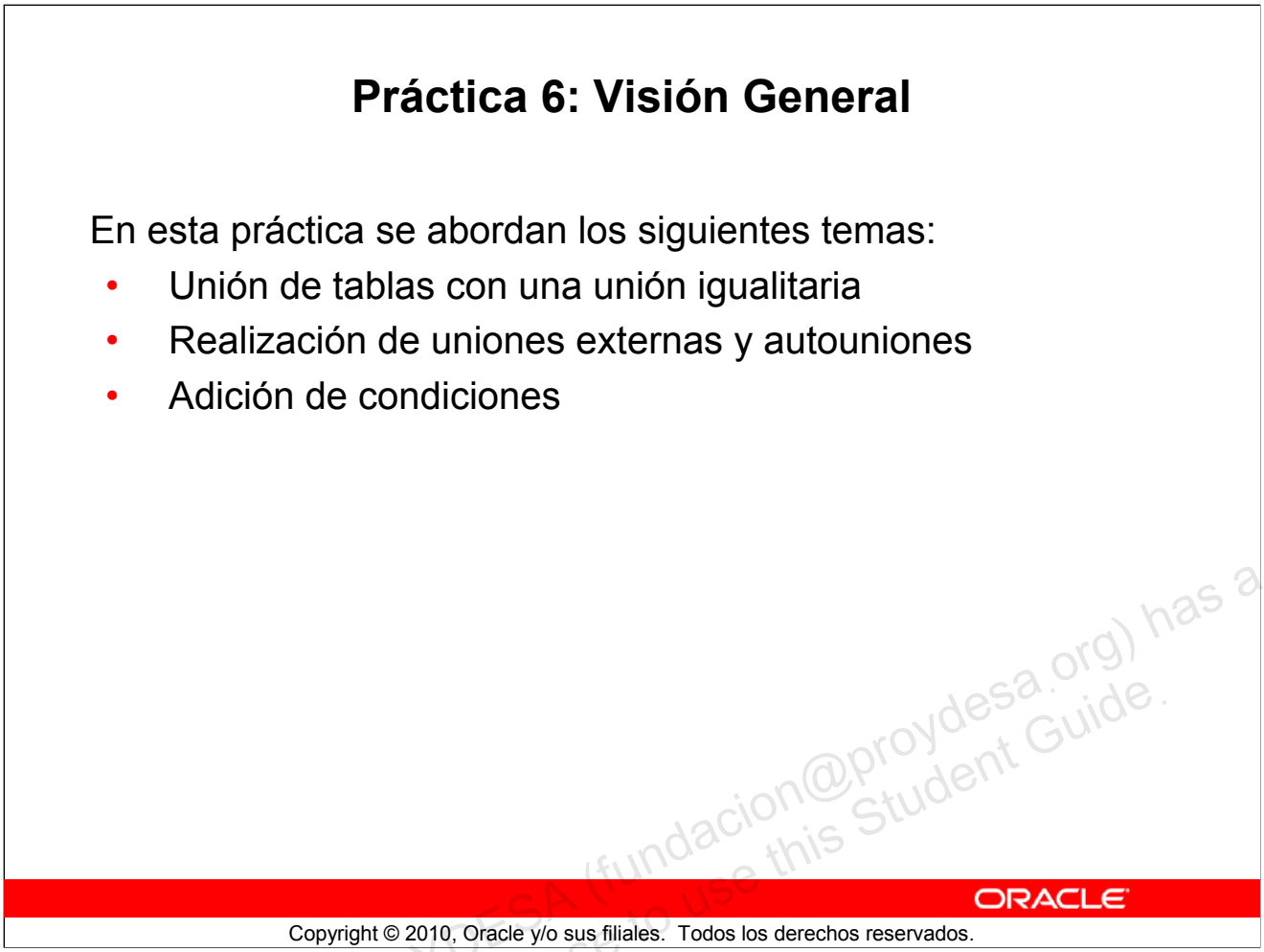
- Uniones igualitarias
- Uniones no igualitarias
- Uniones `OUTER`
- Autouniones
- Uniones cruzadas
- Uniones naturales
- Unión `OUTER` completa (o de dos lados)

Productos Cartesianos

Un producto cartesiano da como resultado una visualización de todas las combinaciones de filas. Para ello debe omitir la cláusula `WHERE` o especificar la cláusula `CROSS JOIN`.

Alias de Tabla

- Los alias de tabla aceleran el acceso a la base de datos.
- Los alias pueden ayudar a mantener el código SQL más pequeño utilizando de esta forma menos memoria.
- A veces los alias de tabla son obligatorios para evitar la ambigüedad de columna.



Práctica 6: Visión General

Esta práctica está destinada a proporcionarle experiencia en la extracción de datos desde más de una tabla utilizando uniones compatibles con SQL:1999.

7

Uso de Subconsultas para Solucionar Consultas

ORACLE

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Definir subconsultas
- Describir los tipos de problemas que pueden solucionar las subconsultas
- Mostrar los tipos de subconsultas
- Escribir subconsultas de una o varias filas

ORACLE

Objetivos

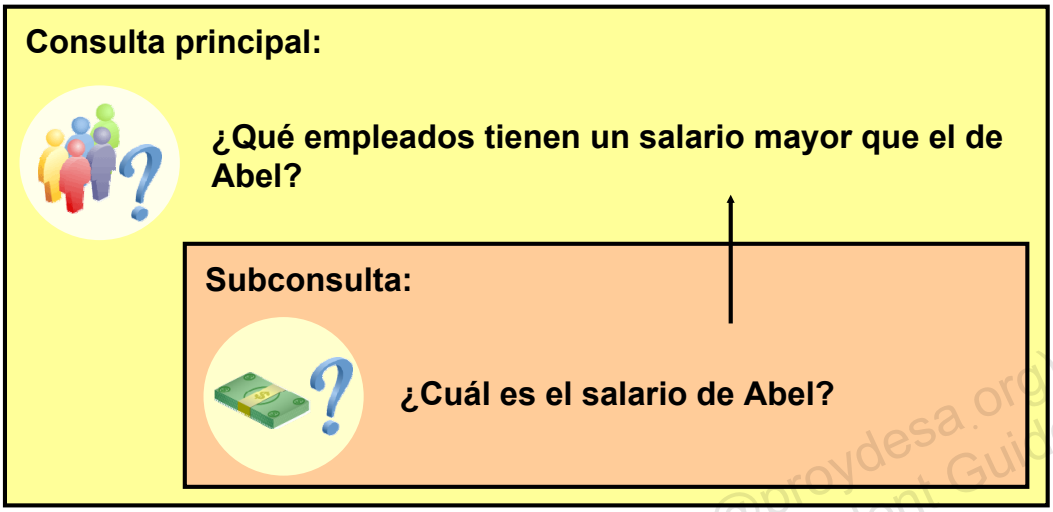
En esta lección aprenderá funciones avanzadas de la sentencia `SELECT`. Puede escribir subconsultas en la cláusula `WHERE` de otra sentencia SQL para obtener valores basados en un valor condicional desconocido. En esta lección se tratan también subconsultas de una y de varias filas.

Agenda

- Subconsulta: tipos, sintaxis e instrucciones
- Subconsultas de una sola fila:
  - Funciones de grupo en una subconsulta
  - Cláusula `HAVING` con subconsultas
- Subconsultas de varias filas
  - Usar el operador `ALL` o `ANY`.
- Uso del operador `EXISTS`
- Valores nulos en una subconsulta

Uso de una Subconsulta para Solucionar Problemas

¿Quién tiene un salario mayor que el de Abel?



ORACLE

Uso de una Subconsulta para Solucionar Problemas

Supongamos que desea escribir una consulta para saber quién gana un salario mayor que el de Abel.

Para solucionar este problema, necesita *dos* consultas: Una para saber cuánto gana Abel y otra para saber quién gana más que esa cantidad.

También puede solucionar este problema combinando las dos consultas y colocando una *dentro* de la otra.

La consulta interna (o *subconsulta*) devuelve un valor que utiliza la consulta externa (o *consulta principal*). El uso de una subconsulta es equivalente a la realización de dos consultas secuenciales y al uso del resultado de la primera consulta como el valor de búsqueda en la segunda consulta.

Sintaxis de la Subconsulta

- La subconsulta (consulta interna) se ejecuta una vez *antes* de la consulta principal (consulta externa).
- La consulta principal utiliza el resultado de la subconsulta.

```
SELECT   select_list
FROM     Tabla
WHERE    expr operator
                    (SELECT       select_list
                    FROM          table);
```


Sintaxis de la Subconsulta

Una subconsulta es una sentencia `SELECT` que está embebida en la cláusula de otra sentencia `SELECT`. Puede crear sentencias potentes a partir de sentencias simples utilizando las subconsultas. Pueden ser muy útiles cuando necesite seleccionar filas de una tabla con una condición de que dependa de los datos de la propia tabla.

Puede colocar la subconsulta en diferentes cláusulas SQL, entre las que se incluyen las siguientes:

- Cláusula `WHERE`
- Cláusula `HAVING`
- Cláusula `FROM`

En la sintaxis:

`operator` incluye una condición de comparación como >, = o `IN`

Nota: las condiciones de comparación pueden ser de dos clases: operadores de una sola fila (>, =, >=, <, <>, <=) y operadores de varias filas (`IN`, `ANY`, `ALL`, `EXISTS`).

Se suele hacer referencia a la subconsulta como una sentencia `SELECT` anidada, una subconsulta `SELECT` o una sentencia `SELECT` interna. Por lo general, la subconsulta se ejecuta en primer lugar y su resultado se utiliza para completar la condición de consulta para la consulta principal (o externa).

Uso de Subconsultas

```
SELECT last_name, salary
FROM   employees
WHERE  salary >   11000
                (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Abel');
```

	LAST_NAME	SALARY
1	Hartstein	13000
2	Higgins	12000
3	King	24000
4	Kochhar	17000
5	De Haan	17000

ORACLE

Uso de Subconsultas

En la diapositiva, la consulta interna determina el salario del empleado Abel. La consulta externa obtiene el resultado de la consulta interna y lo utiliza para mostrar a todos los empleados que ganan más que el empleado Abel.

Instrucciones para el Uso de Subconsultas

- Incluir subconsultas entre paréntesis.
- Coloque las subconsultas a la derecha de la condición de comparación para facilitar la legibilidad. (Sin embargo, la subconsulta puede aparecer a cualquier lado del operador de comparación.)
- Utilizar operadores de una sola fila con subconsultas de una fila y utilizar los operadores de varias filas con subconsultas de varias filas.

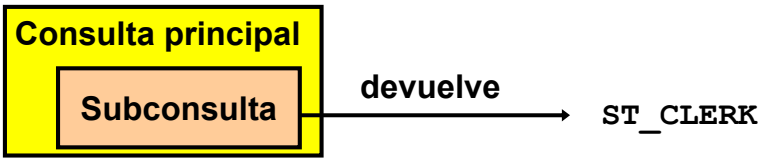
ORACLE

Instrucciones para el Uso de Subconsultas

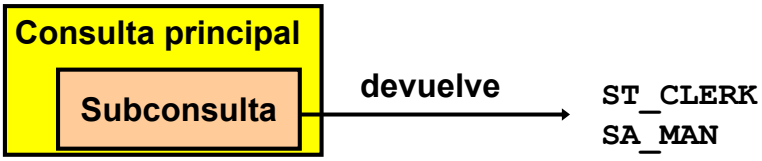
- Una subconsulta debe estar incluida entre paréntesis.
- Coloque las subconsultas a la derecha de la condición de comparación para facilitar la legibilidad. Sin embargo, la subconsulta puede aparecer a cualquier lado del operador de comparación.
- En las subconsultas, se utilizan dos clases de condiciones de comparación: operadores de una sola fila y operadores de varias filas.

Tipos de Subconsultas

- Subconsulta de una sola fila



- Subconsulta de varias filas



Tipos de Subconsultas

- **Subconsultas de una sola fila:** consultas que devuelven sólo una fila de la sentencia `SELECT` interna.
- **Subconsultas de varias filas:** consultas que devuelven más de una fila de la sentencia `SELECT` interna.

Nota: también existen varias subconsultas de varias columnas, que son consultas que devuelven más de una columna de la sentencia `SELECT` interna. Éstas se tratan en el curso *Oracle Database: Conceptos Fundamentales de SQL II.*.

Agenda

- Subconsulta: tipos, sintaxis e instrucciones
- Subconsultas de una sola fila:
  - Funciones de grupo en una subconsulta
  - Cláusula `HAVING` con subconsultas
- Subconsultas de varias filas
  - Usar el operador `ALL` o `ANY`
- Uso del operador `EXISTS`
- Valores nulos en una subconsulta

ORACLE

Subconsultas de Una Sola Fila

- Devuelven una sola fila
- Utilizan operadores de comparación de una sola fila

Operador	Significado
=	Igual que
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
<>	Distinto de

Subconsultas de Una Sola Fila

Una subconsulta de una sola fila devuelve una fila de la sentencia `SELECT` interna. Este tipo de subconsulta utiliza un operador de una sola fila. La diapositiva proporciona una lista de los operadores de una sola fila.

Ejemplo:

Muestre los empleados cuyo ID de cargo sea el mismo que el del empleado 141:

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141);
```

	LAST_NAME	JOB_ID
1	Rajs	ST_CLERK
2	Davies	ST_CLERK
3	Matos	ST_CLERK
4	Vargas	ST_CLERK

Ejecución de Subconsultas de una Sola Fila

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =                    SA_REP
                (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Taylor')
AND    salary >                    8600
                (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Taylor');
```

	LAST_NAME	JOB_ID	SALARY
1	Abel	SA_REP	11000

ORACLE

Ejecución de Subconsultas de una Sola Fila

Una sentencia `SELECT` se puede considerar como un bloque de consulta. El ejemplo de la diapositiva muestra los empleados que realizan el mismo trabajo que “Taylor”, pero que tienen un salario mayor que él.

El ejemplo tiene tres bloques de consulta: la consulta externa y las dos consultas internas. Los bloques de consulta internos se ejecutan primero, produciendo los resultados de consulta `SA_REP` y `8600`, respectivamente. A continuación, se procesa el bloque de consulta externo y utiliza los valores devueltos por las consultas internas para completar sus condiciones de búsqueda.

Tanto las consultas internas como externas devuelven valores únicos (`SA_REP` y `8600`, respectivamente), por lo que a esta sentencia SQL se la denomina subconsulta de una sola fila.

Nota: las consultas externas e internas pueden obtener datos de diferentes tablas.

Uso de Funciones de Grupo en una Subconsulta

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);
```

2500

	LAST_NAME	JOB_ID	SALARY
1	Vargas	ST_CLERK	2500

ORACLE

Uso de Funciones de Grupo en una Subconsulta

Puede mostrar los datos de una consulta principal utilizando una función de grupo en una subconsulta para que devuelva una sola fila. La subconsulta está entre paréntesis y se coloca al final de la condición de comparación.

El ejemplo de la diapositiva muestra el apellido del empleado, el ID de cargo y el salario de todos los empleados cuyo salario sea igual al salario mínimo. La función de grupo `MIN` devuelve un valor único (`2500`) a la consulta externa.

Cláusula HAVING con Subconsultas

- El servidor de Oracle ejecuta primero las subconsultas.
- El servidor de Oracle devuelve los resultados a la cláusula HAVING de la consulta principal.

```
SELECT department_id, MIN(salary)
FROM     employees
GROUP BY department_id
HAVING   MIN(salary) > (SELECT MIN(salary)
                        FROM   employees
                        WHERE  department_id = 50);
```

2500

	DEPARTMENT_ID	MIN(SALARY)
1	(null)	7000
2	20	6000
3	90	17000
4	110	8300
5	80	8600
6	10	4400
7	60	4200

ORACLE

Cláusula HAVING con Subconsultas

Puede utilizar las subconsultas en la cláusula WHERE y en la cláusula HAVING. El servidor de Oracle ejecuta la subconsulta y los resultados se devuelven en la cláusula HAVING de la consulta principal.

La sentencia SQL de la diapositiva muestra todos los departamentos que tienen un salario mínimo mayor que el del departamento 50.

Ejemplo:

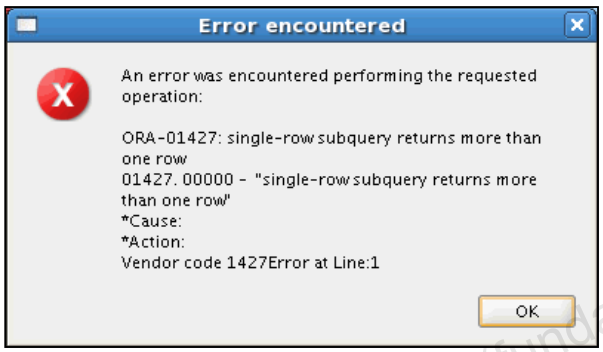
Busque el puesto con el salario medio más bajo.

```
SELECT   job_id, AVG(salary)
FROM     employees
GROUP BY job_id
HAVING   AVG(salary) = (SELECT   MIN(AVG(salary))
                        FROM     employees
                        GROUP BY job_id);
```

	JOB_ID	AVG(SALARY)
1	ST_CLERK	2925

¿Qué Parte de esta Sentencia Es Incorrecta?

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM     employees
                 GROUP BY department_id);
```



Operador de una sola fila con una subconsulta de varias filas

ORACLE

¿Qué Parte de esta Sentencia Es Incorrecta?

Un error muy común en las subconsultas se produce cuando se devuelve más de una fila para una subconsulta de una sola fila.

En la sentencia SQL de la diapositiva, la subconsulta contiene una cláusula `GROUP BY`, que implica que la subconsulta devolverá varias filas, una por cada grupo que encuentre. En este caso, los resultados de la subconsulta son `4400`, `6000`, `2500`, `4200`, `7000`, `17000` y `8300`.

La consulta externa toma esos resultados y los utiliza en su cláusula `WHERE`. La cláusula `WHERE` contiene un operador igual (=), un operador de comparación de una sola fila que espera un único valor. El operador = no puede aceptar más de un valor de la subconsulta y, por lo tanto, genera el error.

Para corregir este error, cambie el operador = a `IN`.

La Consulta Interna No Devuelve Ningún Resultado

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Haas');
```

0 rows selected

La subconsulta no devuelve ninguna fila porque no hay ningún empleado con el nombre “Haas”.

La Consulta Interna No Devuelve Ningún Resultado

Otro problema común de las subconsultas se produce cuando la consulta interna no devuelve ninguna fila.

En la sentencia SQL de la diapositiva, la subconsulta contiene una cláusula `WHERE`. Supuestamente, la intención es encontrar el empleado cuyo nombre es Haas. La sentencia es correcta, pero no selecciona ninguna fila al ejecutarse porque no hay ningún empleado con el nombre Haas. Por lo tanto, la subconsulta no devuelve ninguna fila.

La consulta externa toma los resultados de la subconsulta (nula) y los utiliza en la cláusula `WHERE`. La consulta externa no encuentra ningún empleado con un ID de cargo igual que un valor nulo, por lo tanto, no devuelve ninguna fila. Si existía un cargo con un valor nulo, la fila no se devolverá porque la comparación de dos valores nulos tiene como resultado un valor nulo; por lo tanto, la condición `WHERE` no es verdadera.

Agenda

- Subconsulta: tipos, sintaxis e instrucciones
- Subconsultas de una sola fila:
  - Funciones de grupo en una subconsulta
  - Cláusula `HAVING` con subconsultas
- Subconsultas de varias filas
  - Usar el operador `IN`, `ALL` o `ANY`
- Uso del operador `EXISTS`
- Valores nulos en una subconsulta

ORACLE

Subconsultas de Varias Filas

- Devuelven más de una fila
- Utilizan operadores de comparación de varias filas

Operador	Significado
IN	Igual a cualquier miembro de la lista
ANY	Debe estar precedido por =, !=, >, <, <=, >=. Compara un valor con cada valor de la lista o devuelto por una consulta. Se evalúa en FALSE si la consulta no devuelve ninguna fila.
ALL	Debe estar precedido por =, !=, >, <, <=, >=. Compara un valor con cada valor de la lista o devuelto por una consulta. Se evalúa en TRUE si la consulta no devuelve ninguna fila.

ORACLE

Subconsultas de Varias Filas

Las subconsultas que devuelven más de una fila se denominan subconsultas de varias filas. Con una subconsulta de varias filas, puede utilizar un operador de varias filas en lugar de utilizar un operador de una sola fila. El operador de varias filas espera uno o más valores:

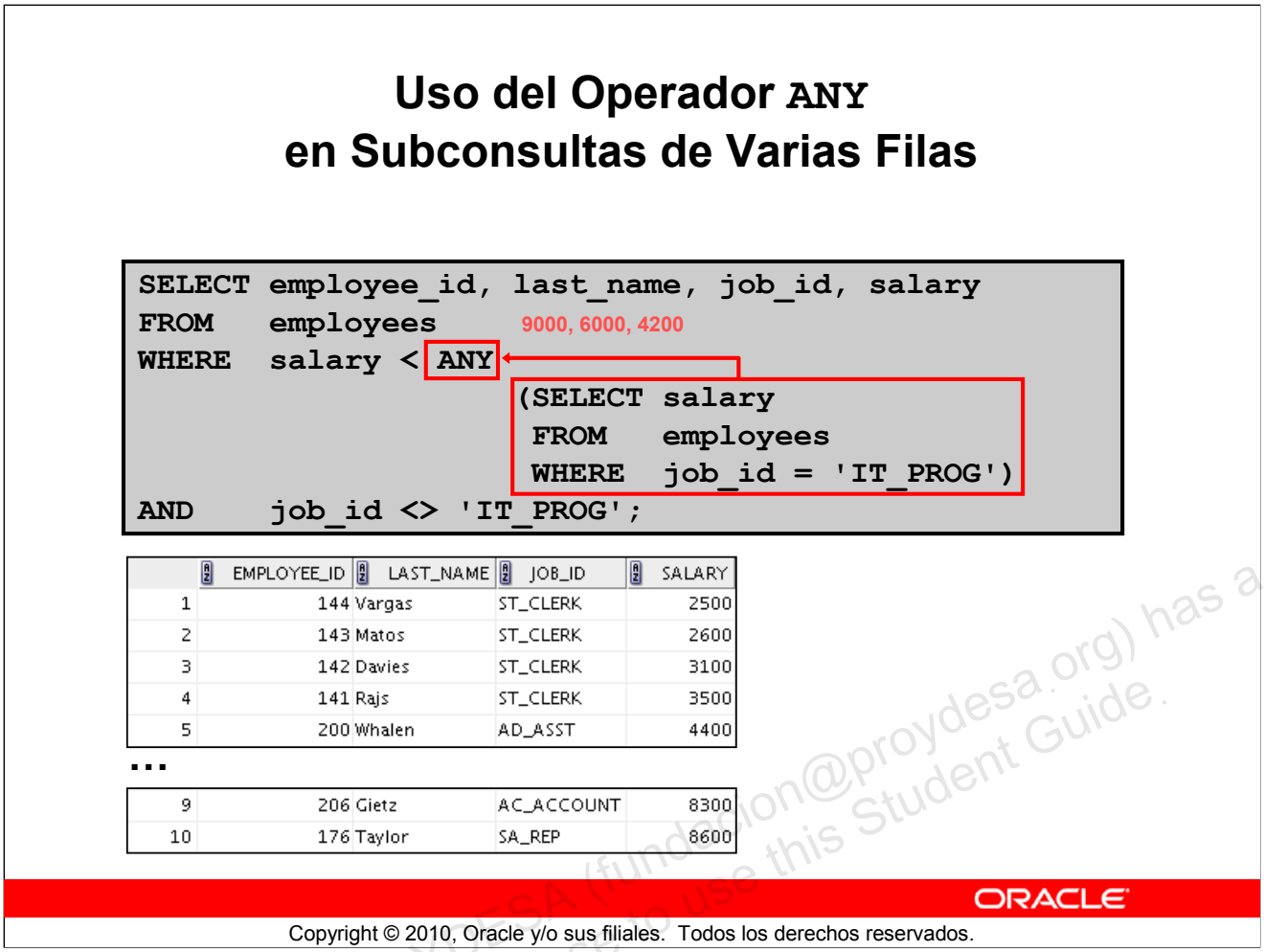
```
SELECT last_name, salary, department_id
FROM   employees
WHERE  salary IN (SELECT   MIN(salary)
                  FROM     employees
                  GROUP BY department_id);
```

Ejemplo:

Busque los empleados que ganan el mismo salario que el salario mínimo de cada departamento.

Primero se ejecuta la consulta interna, produciendo el resultado de consulta. A continuación, se procesa el bloque de consulta principal y se utilizan los valores devueltos por la consulta interna para completar la condición de búsqueda. De hecho, la consulta principal aparece en el servidor de Oracle de la siguiente forma:

```
SELECT last_name, salary, department_id
FROM   employees
WHERE  salary IN (2500, 4200, 4400, 6000, 7000, 8300,
 8600, 17000);
```



Uso del Operador ANY en Subconsultas de Varias Filas

El operador ANY (y su sinónimo, el operador SOME) compara un valor con *cada* valor devuelto por una subconsulta. El ejemplo de la diapositiva muestra los empleados que no son programadores de TI y cuyo salario es menor al de cualquier programador de TI. El salario máximo de un programador es de 9.000 dólares.

- <ANY significa menos que el máximo.
- >ANY significa más que el mínimo.
- =ANY es igual que IN.

Uso del Operador ALL en Subconsultas de Varias Filas

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
                (SELECT salary
                 FROM   employees
                 WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	141	Rajs	ST_CLERK	3500
2	142	Davies	ST_CLERK	3100
3	143	Matos	ST_CLERK	2600
4	144	Vargas	ST_CLERK	2500

Uso del Operador ALL en Subconsultas de Varias Filas

El operador ALL compara un valor con *cada* valor devuelto por una subconsulta. El ejemplo de la diapositiva muestra los empleados cuyo salario es inferior al salario de todos los empleados con un ID de cargo IT_PROG y cuyo trabajo no es IT_PROG.

>ALL significa más que el máximo y <ALL significa menos que el mínimo.

El operador NOT se puede utilizar con los operadores IN, ANY y ALL.

Uso del Operador EXISTS

```
SELECT * FROM departments
WHERE NOT EXISTS
(SELECT * FROM employees
 WHERE employees.department_id=departments.department_id);
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	190	Contracting	(null)	1700

Uso del Operador EXISTS

El operador EXISTS se utiliza en consultas en las que el resultado depende de si existen determinadas filas en la tabla o no. Se evalúa en TRUE si la subconsulta devuelve al menos una fila.

El ejemplo de la diapositiva muestra los departamentos que no tienen empleados. Para cada fila de la tabla DEPARTMENTS, la condición comprueba si existe o no una fila en la tabla EMPLOYEES con el mismo ID de departamento. En caso de que no exista dicha fila, la condición se cumple para la fila en cuestión y se selecciona. Si existe una fila correspondiente en la tabla EMPLOYEES, la fila no se selecciona.

Agenda

- Subconsulta: tipos, sintaxis e instrucciones
- Subconsultas de una sola fila:
  - Funciones de grupo en una subconsulta
  - Cláusula `HAVING` con subconsultas
- Subconsultas de varias filas
  - Usar el operador `ALL` o `ANY`
- Uso del operador `EXISTS`
- Valores nulos en una subconsulta

Valores Nulos en una Subconsulta

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
                            (SELECT mgr.manager_id
                             FROM   employees mgr);
```

```
0 rows selected
```


Valores Nulos en una Subconsulta

La sentencia SQL de la diapositiva intenta mostrar todos los empleados que no tienen ningún subordinado. Lógicamente, esta sentencia SQL debe haber devuelto 12 filas. Sin embargo, la sentencia SQL no devuelve ninguna fila. Uno de los valores devueltos por la consulta interna es un valor nulo y, por lo tanto, la consulta completa no devuelve ninguna fila.

El motivo es que todas las condiciones que comparan un valor nulo tienen un resultado nulo. Por lo tanto, cuando los valores nulos probablemente formen parte del juego de resultados de una subconsulta, no utilice el operador `NOT IN`. El operador `NOT IN` es equivalente a `<> ALL`.

Tenga en cuenta que el valor nulo que forma parte del juego de resultados de una subconsulta no representa ningún problema si se utiliza el operador `IN`. El operador `IN` es equivalente a `=ANY`. Por ejemplo, para mostrar los empleados que tienen subordinados, utilice la siguiente sentencia SQL:

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id  IN
                          (SELECT mgr.manager_id
                           FROM   employees mgr);
```

Valores Nulos en una Subconsulta (continuación)

Por otro lado, una cláusula `WHERE` se puede incluir en la subconsulta para mostrar todos los empleados que no tienen ningún subordinado:

```
SELECT last_name FROM employees
WHERE  employee_id NOT IN
                        (SELECT manager_id
                         FROM   employees
                         WHERE  manager_id IS NOT NULL);
```

Prueba

El uso de una subconsulta es equivalente a la realización de dos consultas secuenciales y al uso del resultado de la primera consulta como los valores de búsqueda en la segunda consulta.

1. Verdadero
2. Falso

ORACLE

Respuesta: 1

Resumen

En esta lección, debe haber aprendido lo siguiente:

- Identificar cuándo una subconsulta puede ayudar a resolver un problema
- Escribir subconsultas cuando una consulta está basada en valores desconocidos

```
SELECT   select_list
FROM     table
WHERE    expr operator
                    (SELECT select_list
                    FROM    table);
```


ORACLE

Resumen

En esta lección, debe haber aprendido a utilizar las subconsultas. Una subconsulta es una sentencia `SELECT` que está embebida en una cláusula de otra sentencia SQL. Las subconsultas son útiles cuando una consulta está basada en criterios de búsqueda con valores intermedios desconocidos.

Las subconsultas tienen las siguientes características:

- Pueden transferir una fila de datos a una sentencia principal que contenga un operador de una sola fila como =, <>, >, >=, < o <=.
- Pueden transferir varias filas de datos a una sentencia principal que contenga un operador de varias filas, como `IN`.
- Primero las procesa el servidor de Oracle y, a continuación, la cláusula `WHERE` o `HAVING` utiliza los resultados.
- Pueden contener funciones de grupo.

Práctica 7: Visión General

En esta práctica se abordan los siguientes temas:

- Creación de subconsultas para consultar valores basados en criterios desconocidos
- Uso de subconsultas para saber qué valores existen en un juego de datos y no en otro

Práctica 7: Visión General

En esta práctica escribirá consultas complejas utilizando sentencias `SELECT`.

Para las preguntas de la práctica, puede crear la consulta interna primero. Antes de codificar la consulta externa, asegúrese de que la consulta interna se ejecuta y produce los datos que anticipe.

8

Uso de los Operadores de Definición

ORACLE

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Describir los operadores de definición
- Utilizar un operador de definición para combinar varias consultas en una sola
- Controlar del orden de las filas devueltas

ORACLE

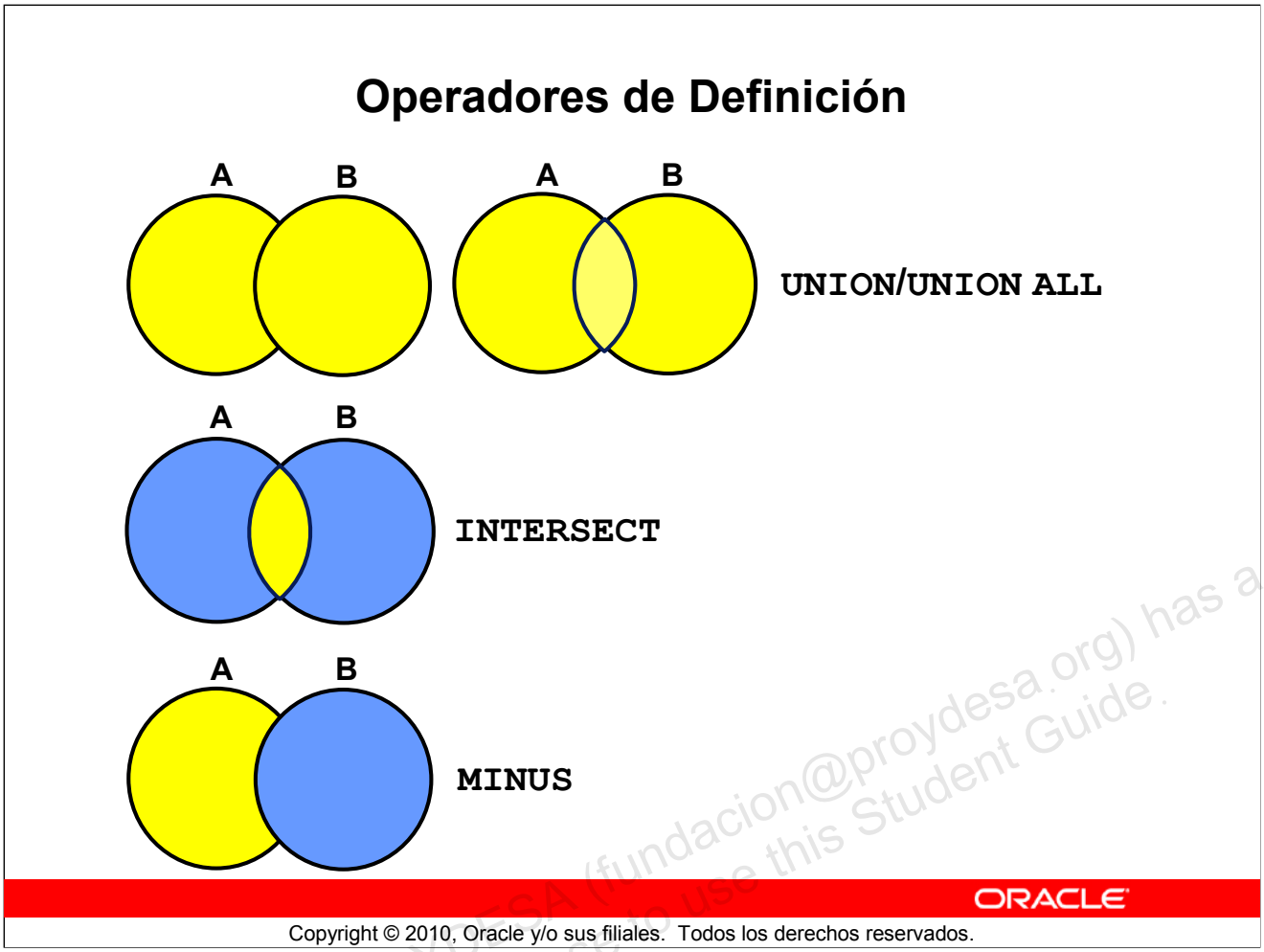
Objetivos

En esta lección aprenderá a escribir consultas utilizando los operadores de definición.

Agenda

- Operadores de Definición: tipos e instrucciones
- Tablas utilizadas en esta lección
- Operador `UNION` y `UNION ALL`
- Operador `INTERSECT`
- Operador `MINUS`
- Coincidencia de las sentencias `SELECT`
- Uso de la cláusula `ORDER BY` en operaciones de definición

ORACLE



Operadores de Definición

Los operadores de definición combinan el resultado de dos o más consultas de componente en un resultado. Las consultas que contienen operadores de definición se denominan *consultas compuestas*.

Operador	Devuelve
UNION	Resultados de ambas consultas después de eliminar la duplicación
UNION ALL	Resultados de ambas consultas, incluidas todas las duplicaciones
INTERSECT	Resultados comunes a ambas consultas
MINUS	Resultados de la primera consulta no presentes en la segunda

Todos los operadores de definición tienen la misma prioridad. Si una sentencia SQL contiene varios operadores de definición, el servidor de Oracle los evalúa de izquierda (parte superior) a derecha (parte inferior), si no hay ningún paréntesis que especifique explícitamente otro orden. Debe utilizar los paréntesis para especificar el orden de evaluación explícitamente en las consultas que utilizan el operador `INTERSECT` en otros operadores de definición.

Instrucciones de los Operadores de Definición

- La expresiones de las listas `SELECT` debe coincidir en número.
- Los tipos de dato para cada columna de la segunda consulta deben coincidir con los tipos de dato de su columna correspondiente en la primera consulta.
- Los paréntesis se pueden utilizar para modificar la secuencia de ejecución.
- La sentencia `ORDER BY` puede aparecer sólo una vez al final de la sentencia.

Instrucciones de los Operadores de Definición

- La expresiones de las listas `SELECT` deben coincidir en número y tipo de dato. Las consultas que utilizan los operadores `UNION`, `UNION ALL`, `INTERSECT` y `MINUS` en su cláusula `WHERE` deben tener el mismo número y tipo de dato de columnas en su lista `SELECT`. El tipo de dato de las columnas de la lista `SELECT` de las consultas de la consulta compuesta puede no ser exactamente el mismo. La columna de la segunda consulta debe estar en el mismo grupo de tipo de dato (por ejemplo, numérico o de caracteres) que la columna correspondiente de la primera consulta.
- Los operadores de definición se pueden utilizar en subconsultas.
- Debe utilizar los paréntesis para especificar el orden de evaluación en las consultas que utilizan el operador `INTERSECT` en otros operadores de definición. Esto garantiza el cumplimiento de los emergentes estándares SQL que otorgarán al operador `INTERSECT` mayor prioridad que los operadores de definición.

Servidor de Oracle y Operadores de Definición

- Las filas duplicadas se eliminan automáticamente excepto en `UNION ALL`.
- Los nombres de columna de la primera consulta aparecen en el resultado.
- Por defecto, la salida se ordena en orden ascendente, excepto en `UNION ALL`.

ORACLE

Servidor de Oracle y Operadores de Definición

Cuando una consulta utiliza operadores de definición, el servidor de Oracle elimina automáticamente las filas duplicadas, excepto en el caso del operador `UNION ALL`. Los nombres de columna de la salida están determinados por la lista de columnas de la primera sentencia `SELECT`. Por defecto, la salida se ordena en orden ascendente según la primera columna de la cláusula `SELECT`.

Las expresiones correspondientes en las listas `SELECT` de las consultas de componente de una consulta compuesta deben coincidir en número y tipo de dato. Si las consultas de componente seleccionan datos de caracteres, el tipo de dato del valor de retorno se determina de la siguiente forma:

- Si ambas consultas seleccionan valores del tipo de dato `CHAR`, de la misma longitud, los valores devueltos tienen el mismo tipo de dato `CHAR` con esa longitud. Si las consultas seleccionan valores del tipo `CHAR` con diferentes longitudes, el valor devuelto es `VARCHAR2` con la longitud del valor `CHAR` más largo.
- Si una o ambas consultas seleccionan valores del tipo de dato `VARCHAR2`, los valores devueltos tienen un tipo de dato `VARCHAR2`.

Si las consultas de componente seleccionan datos numéricos, el tipo de dato del valor de retorno se determina según la prioridad numérica. Si todas las consultas seleccionan valores del tipo `NUMBER`, los valores devueltos tienen el tipo de dato `NUMBER`. En las consultas que utilizan operadores de definición, el servidor de Oracle no realiza una conversión explícita a través de los grupos de tipos de dato. Por lo tanto, si las expresiones correspondientes de las consultas de componente se resuelven tanto en datos numéricos como de caracteres, el servidor de Oracle devuelve un error.

Agenda

- Operadores de Definición: tipos e instrucciones
- Tablas utilizadas en esta lección
- Operador `UNION` y `UNION ALL`
- Operador `INTERSECT`
- Operador `MINUS`
- Coincidencia de las sentencias `SELECT`
- Uso de la cláusula `ORDER BY` en operaciones de definición

ORACLE

Tablas Utilizadas en esta Lección

Las tablas utilizadas en esta lección son:

- `EMPLOYEES`: proporciona los detalles de todos los empleados actuales.
- `JOB_HISTORY`: cuando un empleado cambia de cargo, registra los detalles de la fecha de inicio y de finalización del cargo anterior, el número de identificación del cargo y el departamento.

ORACLE

Tablas Utilizadas en esta Lección

En esta lección se utilizan dos tablas: la tabla `EMPLOYEES` y la tabla `JOB_HISTORY`.

Ya está familiarizado con la tabla `EMPLOYEES` que almacena detalles sobre los empleados como un número único de identificación, la dirección de correo electrónico, el ID de cargo (por ejemplo, `ST_CLERK`, `SA_REP`, etc.), el salario, el gestor, etc.

Algunos de los empleados llevan mucho tiempo en la compañía y han cambiado varias veces de puesto. Esto se supervisa con la tabla `JOB_HISTORY`. Cuando un empleado cambia de cargo, los detalles de fecha de inicio y finalización del cargo anterior, el `job_id` (por ejemplo, `ST_CLERK`, `SA_REP`, etc.) y el departamento se registran en la tabla `JOB_HISTORY`.

La estructura y los datos de las tablas `EMPLOYEES` y `JOB_HISTORY` se muestran en las siguientes páginas.

Tablas Utilizadas en esta Lección (continuación)

Existen casos en los que la misma persona ha ocupado el mismo puesto más de una vez durante el tiempo que ha permanecido en la compañía. Por ejemplo, el empleado Taylor, que empezó a trabajar en la empresa el 24 de marzo de 1998. Taylor ocupó el puesto de `SA_REP` desde el 24 de marzo de 1998 al 31de diciembre de 1998 y el puesto de `SA_MAN` desde el 1 de enero de 1999 al 31 de diciembre de 1999. Taylor volvió a ocupar el puesto de `SA_REP`, que es su puesto actual.

```
DESCRIBE employees
```

```
DESCRIBE employees
Name                          Null     Type
----------------------------- -------- --------------
EMPLOYEE_ID                   NOT NULL NUMBER(6)
FIRST_NAME                             VARCHAR2(20)
LAST_NAME                     NOT NULL VARCHAR2(25)
EMAIL                         NOT NULL VARCHAR2(25)
PHONE_NUMBER                           VARCHAR2(20)
HIRE_DATE                     NOT NULL DATE
JOB_ID                        NOT NULL VARCHAR2(10)
SALARY                                 NUMBER(8,2)
COMMISSION_PCT                         NUMBER(2,2)
MANAGER_ID                             NUMBER(6)
DEPARTMENT_ID                          NUMBER(4)

11 rows selected
```

Tablas Utilizadas en esta Lección (continuación)

```
SELECT employee_id, last_name, job_id, hire_date, department_id
FROM employees;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
1	200	Whalen	AD_ASST	17-SEP-87	10
2	201	Hartstein	MK_MAN	17-FEB-96	20
3	202	Fay	MK_REP	17-AUG-97	20
4	205	Higgins	AC_MGR	07-JUN-94	110
5	206	Gietz	AC_ACCOUNT	07-JUN-94	110
6	100	King	AD_PRES	17-JUN-87	90
7	101	Kochhar	AD_VP	21-SEP-89	90
8	102	De Haan	AD_VP	13-JAN-93	90
9	103	Hunold	IT_PROG	03-JAN-90	60
10	104	Ernst	IT_PROG	21-MAY-91	60
11	107	Lorentz	IT_PROG	07-FEB-99	60
12	124	Mourgos	ST_MAN	16-NOV-99	50
13	141	Rajs	ST_CLERK	17-OCT-95	50
14	142	Davies	ST_CLERK	29-JAN-97	50
15	143	Matos	ST_CLERK	15-MAR-98	50
16	144	Vargas	ST_CLERK	09-JUL-98	50
17	149	Zlotkey	SA_MAN	29-JAN-00	80
18	174	Abel	SA_REP	11-MAY-96	80
19	176	Taylor	SA_REP	24-MAR-98	80
20	178	Grant	SA_REP	24-MAY-99	(null)

```
DESCRIBE job_history
```

```
DESCRIBE job_history
Name                          Null     Type
----------------------------- -------- --------------
EMPLOYEE_ID                   NOT NULL NUMBER(6)
START_DATE                    NOT NULL DATE
END_DATE                      NOT NULL DATE
JOB_ID                        NOT NULL VARCHAR2(10)
DEPARTMENT_ID                          NUMBER(4)

5 rows selected
```

Tablas Utilizadas en esta Lección (continuación)

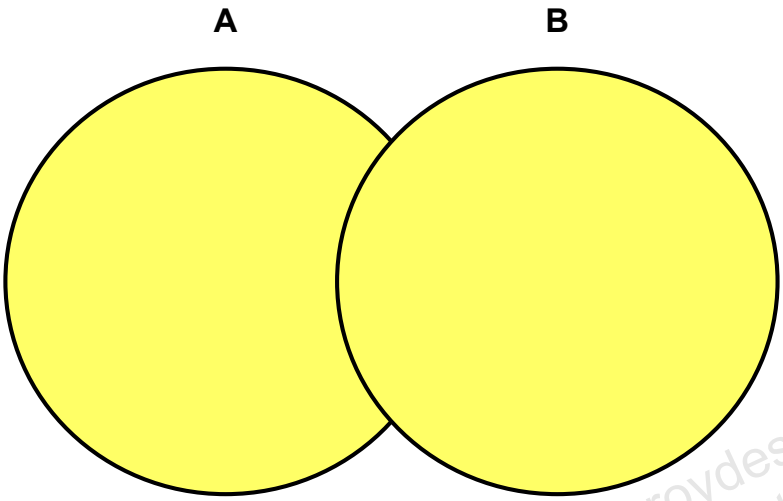
```
SELECT * FROM job_history;
```

	EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
1	102	13-JAN-93	24-JUL-98	IT_PROG	60
2	101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
3	101	28-OCT-93	15-MAR-97	AC_MGR	110
4	201	17-FEB-96	19-DEC-99	MK_REP	20
5	114	24-MAR-98	31-DEC-99	ST_CLERK	50
6	122	01-JAN-99	31-DEC-99	ST_CLERK	50
7	200	17-SEP-87	17-JUN-93	AD_ASST	90
8	176	24-MAR-98	31-DEC-98	SA_REP	80
9	176	01-JAN-99	31-DEC-99	SA_MAN	80
10	200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

Agenda

- Operadores de Definición: tipos e instrucciones
- Tablas utilizadas en esta lección
- Operador `UNION` y `UNION ALL`
- Operador `INTERSECT`
- Operador `MINUS`
- Coincidencia de las sentencias `SELECT`
- Uso de la cláusula `ORDER BY` en operaciones de definición

Operador UNION



El operador UNION devuelve los resultados de ambas consultas después de eliminar la duplicación.

ORACLE

Operador UNION

El operador UNION devuelve todas las filas seleccionadas en cualquier consulta. Utilice el operador UNION para devolver todas las filas de varias tablas y eliminar las filas duplicadas.

Instrucciones

- El número de columnas seleccionadas debe ser el mismo.
- Los tipos de dato de las columnas seleccionadas deben pertenecer al mismo grupo de tipo de dato (por ejemplo, numérico o de caracteres).
- No es necesario que los nombres de las columnas sean idénticos.
- El operador UNION funciona en todas las columnas seleccionadas.
- Los valores NULL no se ignoran durante la comprobación de duplicados.
- Por defecto, la salida se ordena en orden ascendente según de las columnas de la cláusula SELECT.

Uso del Operador UNION

Mostrar los detalles actuales y anteriores del puesto de todos los empleados. Mostrar cada empleado sólo una vez.

```
SELECT employee_id, job_id
FROM   employees
UNION
SELECT employee_id, job_id
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID
1	100	AD_PRES
2	101	AC_ACCOUNT
...		
22	200	AC_ACCOUNT
23	200	AD_ASST
...		
27	205	AC_MGR
28	206	AC_ACCOUNT

ORACLE

Uso del Operador UNION

El operador UNION elimina cualquier registro duplicado. Si los registros de las tablas EMPLOYEES y JOB_HISTORY son idénticos, éstos sólo aparecerán una vez. Observe en la salida de la diapositiva que el registro del empleado con un EMPLOYEE_ID de 200 aparece dos veces porque JOB_ID es diferente en cada fila.

Considere el siguiente ejemplo:

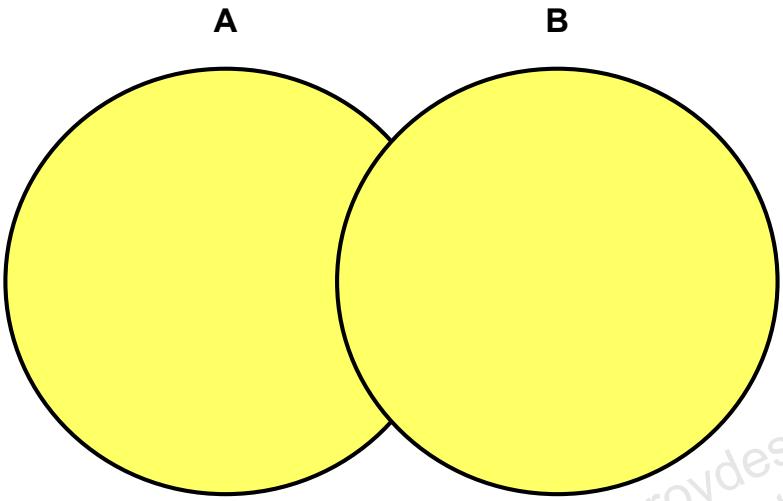
```
SELECT  employee_id, job_id, department_id
FROM    employees
UNION
SELECT  employee_id, job_id, department_id
FROM    job_history;
```

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100	AD_PRES	90
...			
22	200	AC_ACCOUNT	90
23	200	AD_ASST	10
24	200	AD_ASST	90
...			
29	206	AC_ACCOUNT	110

Uso del Operador UNION (continuación)

En la salida anterior, el empleado 200 aparece tres veces. ¿Por qué? Observe los valores de DEPARTMENT_ID del empleado 200. Una fila tiene un DEPARTMENT_ID de 90, otra 10 y la tercera, 90. Debido a estas combinaciones únicas de ID de cargos y de departamentos, cada fila del empleado 200 es única y, por lo tanto, no se considera un duplicado. Observe que la salida está ordenada en orden ascendente según la primera columna de la cláusula SELECT (en este caso, EMPLOYEE_ID).

Operador UNION ALL



El operador UNION ALL devuelve los resultados de ambas consultas, incluidas todas las duplicaciones.

ORACLE

Operador UNION ALL

Utilice el operador UNION ALL para devolver todas las filas de varias consultas.

Instrucciones

Las instrucciones para UNION y UNION ALL son las mismas, excepto en los dos siguientes casos que pertenecen a UNION ALL: a diferencia de UNION, las filas duplicadas no se eliminan y la salida no se ordena por defecto.

Uso del Operador UNION ALL

Mostrar los departamentos actuales y anteriores de todos los empleados.

```
SELECT  employee_id, job_id, department_id
FROM    employees
UNION ALL
SELECT  employee_id, job_id, department_id
FROM    job_history
ORDER BY  employee_id;
```

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100	AD_PRES	90
...			
17	149	SA_MAN	80
18	174	SA_REP	80
19	176	SA_REP	80
20	176	SA_MAN	80
21	176	SA_REP	80
22	178	SA_REP	(null)
23	200	AD_ASST	10
...			
30	206	AC_ACCOUNT	110

ORACLE

Uso del Operador UNION ALL

En el ejemplo se seleccionan 30 filas. La combinación de las dos tablas tiene un total de 30 filas. El operador UNION ALL no elimina las filas duplicadas. UNION devuelve todas las filas distintas seleccionadas por cualquier consulta. UNION ALL devuelve todas las filas seleccionadas por cualquier consulta, incluyendo las duplicadas. Considere la consulta de la diapositiva escrita ahora con la cláusula UNION:

```
SELECT   employee_id, job_id,department_id
FROM     employees
UNION
SELECT   employee_id, job_id,department_id
FROM     job_history
ORDER BY employee_id;
```

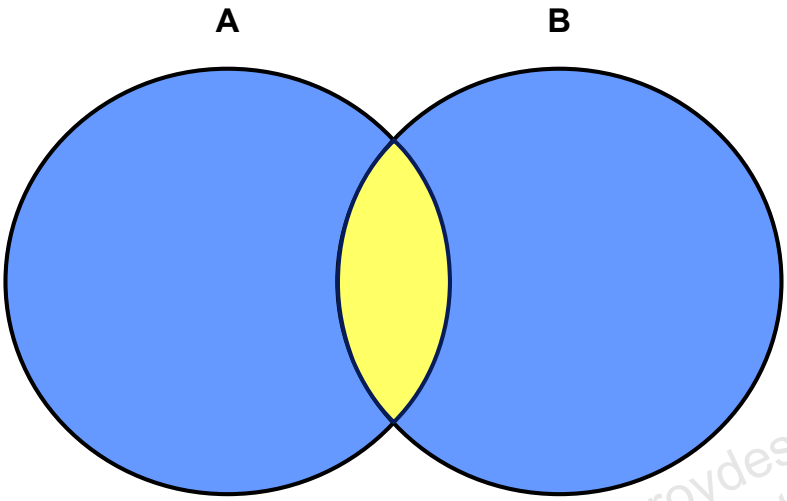
la consulta anterior devuelve 29 filas. Esto se debe a que elimina la siguiente fila (porque está duplicada):

176	SA_REP	80

Agenda

- Operadores de Definición: tipos e instrucciones
- Tablas utilizadas en esta lección
- Operador `UNION` y `UNION ALL`
- Operador `INTERSECT`
- Operador `MINUS`
- Coincidencia de las sentencias `SELECT`
- Uso de la cláusula `ORDER BY` en operaciones de definición

Operador INTERSECT



El operador INTERSECT devuelve filas comunes a ambas consultas.

ORACLE

Operador INTERSECT

Utilice el operador INTERSECT para devolver todas las filas comunes a varias consultas.

Instrucciones

- El número de columnas y los tipos de dato de las columnas seleccionadas por las sentencias SELECT en las consultas deben ser idénticos en todas las sentencias SELECT utilizadas en la consulta. No es necesario, sin embargo, que los nombres de las columnas sean idénticos.
- Si se invierte el orden de las tablas intersectadas no se alterará el resultado.
- INTERSECT no ignora los valores NULL.

Uso del Operador INTERSECT

Mostrar los ID de empleado y de cargo de los empleados que actualmente tienen el mismo puesto que anteriormente (es decir, han cambiado de cargo pero ahora han vuelto a realizar el mismo trabajo que realizaban anteriormente).

```
SELECT employee_id, job_id
FROM   employees
INTERSECT
SELECT employee_id, job_id
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

ORACLE

Uso del Operador INTERSECT

En el ejemplo de la diapositiva, la consulta devuelve sólo los registros que tienen los mismos valores en las columnas seleccionadas en ambas tablas.

¿Cuál será el resultado si agrega la columna DEPARTMENT_ID a la sentencia SELECT de la tabla EMPLOYEES y la columna DEPARTMENT_ID a la sentencia SELECT de la tabla JOB_HISTORY y ejecuta esta consulta? El resultado puede variar debido a la introducción de otra columna cuyos valores pueden o no estar duplicados.

Ejemplo:

```
SELECT  employee_id, job_id, department_id
FROM    employees
INTERSECT
SELECT  employee_id, job_id, department_id
FROM    job_history;
```

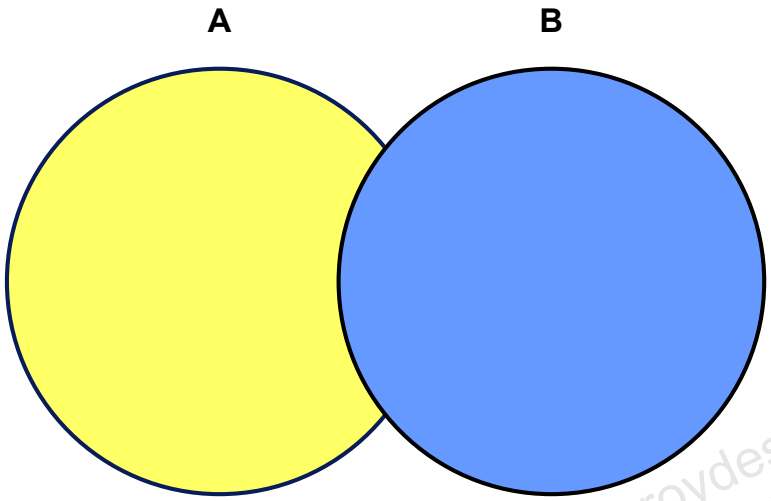
	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	176	SA_REP	80

El empleado 200 ya no forma parte de los resultados porque el valor EMPLOYEES.DEPARTMENT_ID es diferente del valor JOB_HISTORY.DEPARTMENT_ID.

Agenda

- Operadores de Definición: tipos e instrucciones
- Tablas utilizadas en esta lección
- Operador `UNION` y `UNION ALL`
- Operador `INTERSECT`
- Operador `MINUS`
- Coincidencia de las sentencias `SELECT`
- Uso de la cláusula `ORDER BY` en operaciones de definición

Operador MINUS



El operador MINUS devuelve todas las filas distintas seleccionadas por la primera consulta, pero que no están presentes en el juego de resultados de la segunda consulta.

ORACLE

Operador MINUS

Utilice el operador `MINUS` para devolver todas las filas distintas seleccionadas por la primero consulta, pero que no están presentes en el juego de resultados de la segunda consulta (la primera sentencia `SELECT` `MINUS` la segunda sentencia `SELECT`).

Nota: el número de columnas y los tipos de dato de las columnas seleccionadas por las sentencias `SELECT` de las consultas deben pertenecer al mismo grupo de tipo de dato en todas las sentencias `SELECT` utilizadas en la consulta. No es necesario, sin embargo, que los nombres de las columnas sean idénticos.

Uso del Operador MINUS

Mostrar los identificadores de empleado cuyos empleados no han cambiado sus puestos ni una vez.

```
SELECT employee_id
FROM   employees
MINUS
SELECT employee_id
FROM   job_history;
```

	EMPLOYEE_ID
1	100
2	103
3	104
...	
13	202
14	205
15	206

ORACLE

Uso del Operador MINUS

En el ejemplo de la diapositiva, los ID de empleado de la tabla JOB_HISTORY se restan de los de la tabla EMPLOYEES. El juego de resultados muestra los empleados resultantes después de la resta; están representados por filas que existen en la tabla EMPLOYEES pero que no existen en la tabla JOB_HISTORY. Éstos son los registros de los empleados que no han cambiado sus puestos ni una vez.

Agenda

- Operadores de Definición: tipos e instrucciones
- Tablas utilizadas en esta lección
- Operador `UNION` y `UNION ALL`
- Operador `INTERSECT`
- Operador `MINUS`
- Coincidencia de las sentencias `SELECT`
- Uso de la cláusula `ORDER BY` en operaciones de definición

Coincidencia de las Sentencias SELECT

- Con el operador UNION, se muestra el ID de ubicación, el nombre de departamento y el estado en el que está ubicado.
- Debe hacer que coincida el tipo de dato (mediante la función TO_CHAR o cualquier otra función de conversión) cuando las columnas no existan en una tabla o en la otra.

```
SELECT location_id, department_name "Department",
   TO_CHAR(NULL) "Warehouse location"
FROM departments
UNION
SELECT location_id, TO_CHAR(NULL) "Department",
   state_province
FROM locations;
```

ORACLE

Coincidencia de las Sentencias SELECT

Debido a que las expresiones de las listas SELECT de las consultas deben coincidir en número, puede utilizar columnas ficticias y funciones de conversión de tipos de dato para cumplir con esta regla. En la diapositiva, se asigna el nombre Warehouse location como la cabecera de columna ficticia. La función TO_CHAR se utiliza en la primera consulta para que coincida el tipo de dato VARCHAR2 de la columna state_province que recupera la segunda consulta. Igualmente, la función TO_CHAR se utiliza en la segunda consulta para que coincida el tipo de dato VARCHAR2 de la columna department_name que recupera la primera consulta.

La salida de la consulta se muestra a continuación:

	LOCATION_ID	Department	Warehouse location
1	1400	IT	(null)
2	1400	(null)	Texas
3	1500	Shipping	(null)
4	1500	(null)	California
5	1700	Accounting	(null)
6	1700	Administration	(null)
7	1700	Contracting	(null)
8	1700	Executive	(null)

...

Coincidencia de las Sentencias SELECT: Ejemplo

Utilizar el operador UNION, mostrar el ID de empleado, ID de cargo y salario de todos los empleados.

```
SELECT employee_id, job_id,salary
FROM   employees
UNION
SELECT employee_id, job_id,0
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID	SALARY
1	100	AD_PRES	24000
2	101	AC_ACCOUNT	0
3	101	AC_MGR	0
4	101	AD_VP	17000
5	102	AD_VP	17000
...			
29	205	AC_MGR	12000
30	206	AC_ACCOUNT	8300

ORACLE

Coincidencia de las Sentencias SELECT: Ejemplo

Las tablas EMPLOYEES y JOB_HISTORY tienen varias columnas en común (por ejemplo, EMPLOYEE_ID, JOB_ID y DEPARTMENT_ID). Pero, ¿y si lo que desea es que la consulta muestre el ID de empleado, ID de cargo y salario con el operador UNION sabiendo que el salario sólo existe en la tabla EMPLOYEES?

El código de ejemplo de la diapositiva coinciden las columnas EMPLOYEE_ID y JOB_ID de las tablas EMPLOYEES y JOB_HISTORY. Se agrega el valor literal 0 a la sentencia JOB_HISTORY SELECT para que coincida con columna numérica SALARY de la sentencia EMPLOYEES SELECT.

En los resultados mostrados en la diapositiva, cada fila de la salida que corresponde a un registro de la tabla JOB_HISTORY contiene un 0 en la columna SALARY.

Agenda

- Operadores de Definición: tipos e instrucciones
- Tablas utilizadas en esta lección
- Operador `UNION` y `UNION ALL`
- Operador `INTERSECT`
- Operador `MINUS`
- Coincidencia de las sentencias `SELECT`
- Uso de la cláusula `ORDER BY` en operaciones de definición

ORACLE

Uso de la Cláusula ORDER BY en Operaciones de Definición

- La cláusula ORDER BY sólo puede aparecer una vez al final de la consulta compuesta.
- Las consultas de componente no pueden tener cláusulas ORDER BY individuales.
- La cláusula ORDER BY reconoce sólo las columnas de la primera consulta SELECT.
- Por defecto, la primera columna de la primera consulta SELECT se utiliza para ordenar la salida en orden ascendente.

ORACLE

Uso de la Cláusula ORDER BY en Operaciones de Definición

La cláusula ORDER BY sólo se puede utilizar una vez en una consulta compuesta. Si se utiliza, la cláusula ORDER BY se debe colocar al final de la consulta. La cláusula ORDER BY acepta el nombre de columna o alias. Por defecto, la salida se ordena en orden ascendente según la primera columna de la cláusula SELECT.

Nota: la cláusula ORDER BY no reconoce los nombres de columna de la segunda consulta SELECT. Para evitar la confusión con los nombres de columna, es una práctica habitual utilizar cláusula ORDER BY según las posiciones de las columnas.

Por ejemplo, en la siguiente sentencia, la salida se mostrará en orden ascendente según job_id.

```
SELECT employee_id, job_id,salary
FROM   employees
UNION
SELECT employee_id, job_id,0
FROM   job_history
ORDER BY 2;
```

Si omite ORDER BY, por defecto, la salida se ordenará en orden ascendente según employee_id. No puede utilizar las columnas de la segunda consulta para ordenar la salida.

Prueba

Identifique las Instrucciones del operador de definición.

1. La expresiones de las listas `SELECT` debe coincidir en número.
2. Los paréntesis no se pueden utilizar para modificar la secuencia de ejecución.
3. Los tipos de dato para cada columna de la segunda consulta deben coincidir con los tipos de dato de su columna correspondiente en la primera consulta.
4. La cláusula `ORDER BY` sólo se puede utilizar una vez en la consulta compuesta, a menos que se utilice un operador `UNION ALL`.

ORACLE

Respuesta: 1, 3

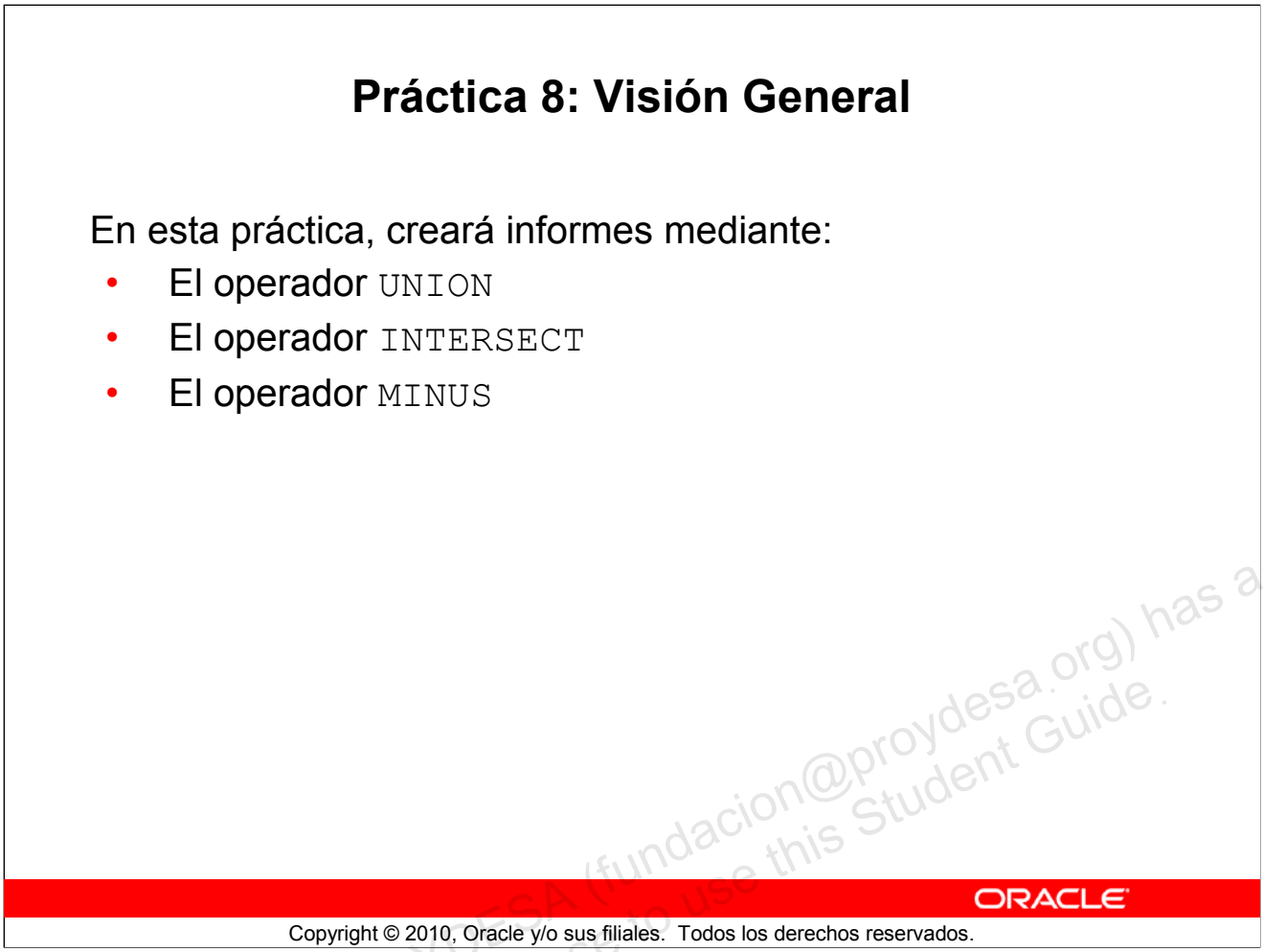
Resumen

En esta lección, debe haber aprendido a utilizar:

- `UNION` para devolver todas las filas distintas
- `UNION ALL` para devolver todas las filas, incluyendo los duplicados
- `INTERSECT` para devolver todas las filas que comparten ambas consultas
- `MINUS` para devolver todas las filas distintas seleccionadas por la primera consulta, pero no por la segunda
- `ORDER BY` sólo al final de la sentencia

Resumen

- El operador `UNION` devuelve todas las filas distintas seleccionadas por cada consulta de la consulta compuesta. Utilice el operador `UNION` para devolver todas las filas de varias tablas y eliminar las filas duplicadas.
- Utilice el operador `UNION ALL` para devolver todas las filas de varias consultas. A diferencia del operador `UNION`, las filas duplicadas no se eliminan y la salida no se ordena por defecto.
- Utilice el operador `INTERSECT` para devolver todas las filas comunes a varias consultas.
- Utilice el operador `MINUS` para devolver filas en la primera consulta que no están en la segunda.
- Recuerde utilizar la cláusula `ORDER BY` sólo al final de la sentencia compuesta.
- Asegúrese de que las expresiones correspondientes de las listas `SELECT` coinciden en número y tipo de dato.



Práctica 8: Visión General

En esta práctica escribirá consultas utilizando los operadores de definición.

9

Manipulación de Datos

ORACLE

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Describir cada sentencia de lenguaje de manipulación de datos (DML)
- Insertar filas en una tabla
- Actualizar filas en una tabla
- Suprimir filas de una tabla
- Controlar transacciones

ORACLE

Objetivo

En esta lección aprenderá a utilizar las sentencias de lenguaje de manipulación de datos (DML) para insertar filas en una tabla, actualizar filas existentes en una tabla y suprimir filas existentes de una tabla. También aprenderá a controlar transacciones con las sentencias `COMMIT`, `SAVEPOINT` y `ROLLBACK`.

Agenda

- Adición de nuevas filas a una tabla
  - Sentencia `INSERT`
- Cambio de datos en la tabla
  - Sentencia `UPDATE`
- Eliminación de filas de una tabla:
  - Sentencia `DELETE`
  - Sentencia `TRUNCATE`
- Control de transacciones de la base de datos mediante `COMMIT`, `ROLLBACK` y `SAVEPOINT`
- Consistencia de lectura
- Cláusula `FOR UPDATE` en una sentencia `SELECT`

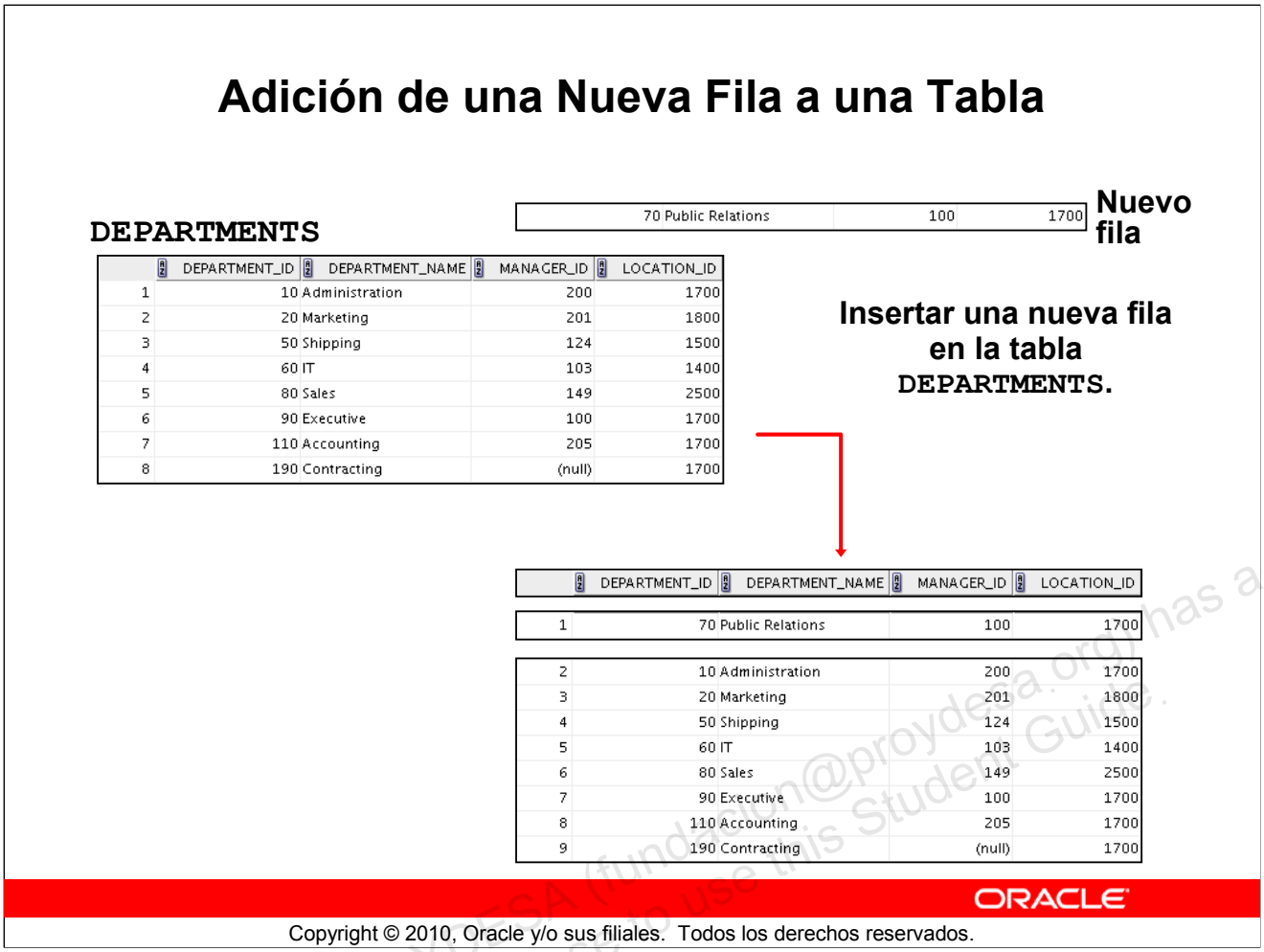
Lenguaje de Manipulación de Datos

El lenguaje de manipulación de datos (DML) es una parte fundamentalmente de SQL. Para agregar, actualizar o suprimir los datos de la base de datos, ejecute una sentencia DML. La recopilación de sentencias DML que forman una unidad lógica de trabajo se denomina *transacción*.

Piense en una base de datos bancaria. Cuando un cliente del banco transfiere dinero de su cuenta de ahorro a una cuenta corriente, la transacción puede constar de las siguientes tres acciones diferentes: reducir la cuenta de ahorro, aumentar la cuenta corriente y registrar la transacción en el diario de transacciones. El servidor de Oracle debe garantizar que se ejecuten las tres sentencias SQL para mantener el balance correcto de las cuentas. Si algo impide que una de las sentencias de la transacción se ejecute, las demás sentencias de la transacción se deben deshacer.

Nota

- La mayoría de las sentencias DML de esta lección asumen que no se ha violado ninguna restricción de esta tabla. Las restricciones se tratarán más adelante en este curso.
- En SQL Developer, haga clic en el icono Run Script o pulse [F5] para ejecutar las sentencias DML. Los mensajes de comentarios se mostrarán en la página con separadores Script Output.



Adición de una Nueva Fila a una Tabla

El gráfico de la diapositiva ilustra un nuevo departamento a la tabla `DEPARTMENTS`.

Sintaxis de las Sentencias INSERT

- Agregar nuevas filas a una tabla mediante la sentencia INSERT:

```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

- Con esta sintaxis, sólo se inserta una fila cada vez.

Sintaxis de las Sentencias INSERT

Puede agregar nuevas filas a una tabla emitiendo la sentencia INSERT.

En la sintaxis:

table	es el nombre de la tabla.
column	es el nombre de la columna de la tabla que se debe rellenar.
value	es el valor correspondiente para la columna.

Nota: esta sentencia con la cláusula VALUES agrega sólo una fila cada vez a la tabla.

Inserción de Filas

- Insertar una nueva fila que contenga los valores de cada columna.
- Mostrar valores en el orden por defecto de las columnas de la tabla.
- Opcionalmente, mostrar la lista de columnas en la cláusula `INSERT`.

```
INSERT INTO departments(department_id,
       department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
1 rows inserted
```

- Encerrar valores de caracteres y de fecha entre comillas simples.

ORACLE

Inserción de Filas

Puesto que puede insertar una nueva fila que contenga los valores de cada columna, no es necesaria la lista de columnas en la cláusula `INSERT`. Sin embargo, si no utiliza la lista de columnas, los valores se deben mostrar según el orden por defecto de las columnas en la tabla y se debe proporcionar un valor para cada columna.

```
DESCRIBE departments
```

Name	Null	Type
DEPARTMENT_ID	NOT NULL	NUMBER(4)
DEPARTMENT_NAME	NOT NULL	VARCHAR2(30)
MANAGER_ID		NUMBER(6)
LOCATION_ID		NUMBER(4)

Para mayor claridad, utilice la lista de columnas en la cláusula `INSERT`.

Encierre los valores de caracteres y fecha entre comillas simples; sin embargo, no se recomienda encerrar valores numéricos entre comillas simples.

Inserción de Filas con Valores Nulos

- Método implícito: omitir la columna de la lista de columnas.

```
INSERT INTO   departments (department_id,
                           department_name)
VALUES        (30, 'Purchasing');
1 rows inserted
```

- Método explícita: especificar la palabra clave `NULL` en la cláusula `VALUES`.

```
INSERT INTO   departments
VALUES        (100, 'Finance', NULL, NULL);
1 rows inserted
```


ORACLE

Inserción de Filas con Valores Nulos

Metodo	Descripción
Implícito	Omite la columna de la lista de columnas.
Explícito	Introduzca la palabra clave NULL en la lista VALUES; especifique la cadena vacía ('') en la lista VALUES de las cadenas de caracteres y fechas.

Asegúrese de que puede utilizar valores nulos en la columna de destino mediante la verificación del estado `Null` con el comando `DESCRIBE`.

El servidor de Oracle aplica automáticamente todos los tipos de dato, rangos de datos y restricciones de integridad de los datos. Las columnas que no se muestran explícitamente obtienen un valor nulo en la nueva fila.

Los errores comunes que se producen durante la entrada del usuario se comprueban en el siguiente orden:

- Valor obligatorio que falta para una columna `NOT NULL`
- Valor duplicado que viola cualquier restricción de clave única o primaria
- `Cualquier` valor que viole una restricción `CHECK`
- Mantenimiento de la integridad referencial para la restricción de clave ajena
- No coincidencias de tipo de dato o valores demasiado anchos para la columna

Nota: se recomienda el uso de la lista de columnas porque hace la sentencia `INSERT` más legible y fiable o menos proclive a errores.

Inserción de Valores Especiales

La función SYSDATE registra la fecha y hora actuales.

```
INSERT INTO employees (employee_id,
                       first_name, last_name,
                       email, phone_number,
                       hire_date, job_id, salary,
                       commission_pct, manager_id,
                       department_id)
VALUES                (113,
                       'Louis', 'Popp',
                       'LPOPP', '515.124.4567',
                       SYSDATE, 'AC_ACCOUNT', 6900,
                       NULL, 205, 110);
```

1 rows inserted

ORACLE

Inserción de Valores Especiales

Puede utilizar funciones para introducir valores especiales en la tabla.

En el ejemplo de la diapositiva se registra información sobre el empleado Popp en la tabla EMPLOYEES. Proporciona la fecha y hora actuales en la columna HIRE_DATE. Utiliza la función SYSDATE que devuelve la fecha y hora actuales del servidor de base de datos. También puede utilizar la función CURRENT_DATE para obtener la fecha actual en la zona horaria de la sesión. Además puede utilizar la función USER al insertar filas en la tabla. La función USER registra el nombre de usuario actual.

Confirmación de Adiciones a la Tabla

```
SELECT employee_id, last_name, job_id, hire_date, commission_pct
FROM   employees
WHERE  employee_id = 113;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	COMMISSION_PCT
1	113	Popp	AC_ACCOUNT	10-JUL-09	(null)

Inserción de Valores de Fecha y Hora Específicos

- Agregar un nuevo empleado.

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'SA_REP', 11000, 0.2, 100, 60);
```

1 rows inserted

- Verificar la adición.

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
1	114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	SA_REP	11000	0.2

ORACLE

Inserción de Valores de Fecha y Hora Específicos

El formato DD-MON-RR se suele utilizar para insertar el valor de fecha. Con el formato `RR`, el sistema proporciona el siglo correcto automáticamente.

También puede especificar el valor de fecha en formato DD-MON-YYYY. Se recomienda hacerlo así porque especifica de forma clara el siglo y no depende de la lógica de formato `RR` interna de la especificación del siglo correcto.

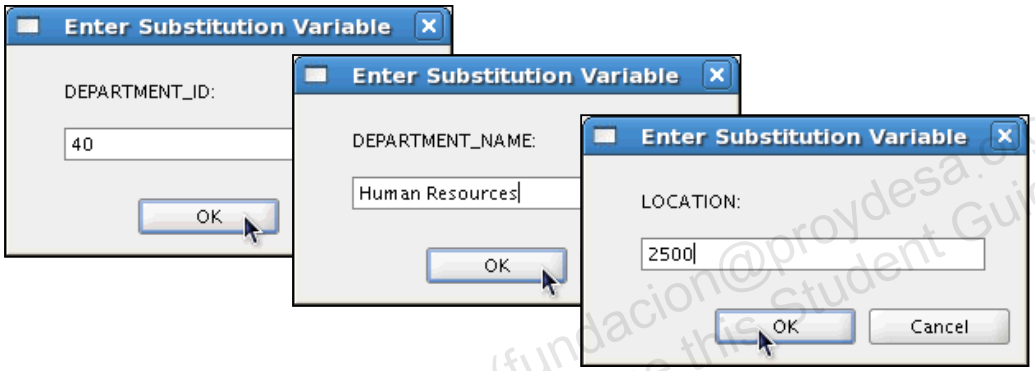
Si necesita introducir una fecha en un formato diferente al formato por defecto (por ejemplo, con otro siglo o con una hora concreta), debe utilizar la función `TO_DATE`.

En el ejemplo de la diapositiva se registra información del empleado Raphealy en la tabla `EMPLOYEES`. Define la columna `HIRE_DATE` en el 3 de febrero de 1999.

Creación de un Script

- Utilizar el carácter de sustitución & en una sentencia SQL para solicitar valores.
- & es un marcador de posición del valor de la variable.

```
INSERT INTO departments
            (department_id, department_name, location_id)
VALUES      (&department_id, '&department_name',&location);
```



ORACLE

Creación de un Script

Puede guardar comandos con variables de sustitución en un archivo y ejecutar los comandos en el archivo. En el ejemplo de la diapositiva se registra información sobre un departamento de la tabla DEPARTMENTS.

Ejecute el script y se le solicitará una entrada para cada una de las variables de sustitución (&). Después de introducir un valor para la variable de sustitución, haga clic en el botón OK. Los valores que introduce el usuario son los que se sustituyen en la sentencia. Esto permite ejecutar el mismo script una y otra vez, pero proporcionar un juego de valores diferente cada vez que lo ejecute.

Copia de Filas de Otra Tabla

- Escribir la sentencia `INSERT` con una subconsulta:

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';
4 rows inserted
```

- No utilizar la cláusula `VALUES`.
- Hacer coincidir el número de columnas de la cláusula `INSERT` con el de la subconsulta.
- Inserta todas las filas devueltas por la subconsulta en la tabla, `sales_reps`.

Copia de Filas de Otra Tabla

Puede utilizar la sentencia `INSERT` para agregar filas a la tabla en la que se derivan los valores de las tablas existentes. En el ejemplo de la diapositiva, para que la sentencia `INSERT INTO` funcione, debe haber creado antes la tabla `sales_reps` mediante la sentencia `CREATE TABLE`. `CREATE TABLE` se trata en la lección titulada “Uso de Sentencias DDL para Crear y Gestionar Tablas”.

En lugar de la cláusula `VALUES`, utilice una subconsulta.

Sintaxis

```
INSERT INTO table [ column (, column) ] subquery;
```

En la sintaxis:

table	es el nombre de la tabla.
column	es el nombre de la columna de la tabla que se debe rellenar.
subquery	es la subconsulta que devuelve filas a la tabla.

El número de columnas y sus tipos de dato de la lista de columnas de la cláusula `INSERT` deben coincidir con el número de valores y sus tipos de dato en la subconsulta. Se agregan cero o más filas en función del número de filas devueltas por la subconsulta. Para crear una copia de la filas de una tabla, utilice `SELECT *` en la subconsulta:

```
INSERT INTO copy_emp
   SELECT *
   FROM   employees;
```

Agenda

- Adición de nuevas filas a una tabla
  - Sentencia `INSERT`
- Cambio de datos en la tabla
  - Sentencia `UPDATE`
- Eliminación de filas de una tabla:
  - Sentencia `DELETE`
  - Sentencia `TRUNCATE`
- Control de transacciones de la base de datos mediante `COMMIT`, `ROLLBACK` y `SAVEPOINT`
- Consistencia de lectura
- Cláusula `FOR UPDATE` en una sentencia `SELECT`

ORACLE

Cambio de Datos en la Tabla

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	60
104	Bruce	Ernst	6000	103	(null)	60
107	Diana	Lorentz	4200	103	(null)	60
124	Kevin	Mourgos	5800	100	(null)	50

Actualizar filas en la tabla EMPLOYEES:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	80
104	Bruce	Ernst	6000	103	(null)	80
107	Diana	Lorentz	4200	103	(null)	80
124	Kevin	Mourgos	5800	100	(null)	50

ORACLE

Cambio de Datos en la Tabla

La diapositiva ilustra el cambio de número de departamento para los empleados del departamento 60 al departamento 80.

Sintaxis de Sentencias UPDATE

- Modificar los valores existentes en una tabla con la sentencia UPDATE:

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

- Actualizar más de una fila cada vez (si es necesario).

ORACLE

Sintaxis de Sentencias UPDATE

Puede modificar los valores existentes en una tabla mediante la sentencia UPDATE.

En la sintaxis:

table	es el nombre de la tabla.
column	es el nombre de la columna de la tabla que se debe rellenar.
value	es el valor o subconsulta correspondiente para la columna.
condition	identifica las filas que se deben actualizar y se compone de nombres de columna, expresiones, constantes, subconsultas y operadores de comparación.

Para confirmar la operación de actualización, consulte la tabla para visualizar las filas actualizadas.

Para obtener más información, consulte la sección sobre "UPDATE" en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Nota: en general, la primera columna de clave primaria se utiliza en la cláusula WHERE para identificar una única fila para la actualización. El uso de otras columnas puede provocar una actualización inesperada de varias filas. Por ejemplo, identificar una fila de la tabla EMPLOYEES por nombre es peligroso, ya que puede que más de un empleado tengan el mismo nombre.

Actualización de Filas en una Tabla

- Si se especifica la cláusula `WHERE`, se modifican los valores de una fila o varias filas específicas:

```
UPDATE employees
SET    department_id = 50
WHERE employee_id = 113;
1 rows updated
```

- Si se omite la cláusula `WHERE`, se modifican los valores de todas las filas de la tabla:

```
UPDATE   copy_emp
SET      department_id = 110;
22 rows updated
```

- Especificar `SET column_name= NULL` para actualizar un valor de columna a `NULL`.

ORACLE

Actualización de Filas en una Tabla

La sentencia `UPDATE` modifica los valores de una fila o varias filas específicas si se especifica la cláusula `WHERE`. El ejemplo de la diapositiva muestra la transferencia del empleado 113 (Popp) al departamento 50.

Si omite la cláusula `WHERE`, se modifican todas las filas de la tabla. Examine las filas actualizadas en la tabla `COPY_EMP`.

```
SELECT last_name, department_id
FROM   copy_emp;
```

	LAST_NAME	DEPARTMENT_ID
1	Whalen	110
2	Hartstein	110
3	Fay	110

...

Por ejemplo, un empleado que era `SA_REP` ahora ha cambiado su puesto a `IT_PROG`. Por lo tanto, se debe actualizar su `JOB_ID` y el campo de comisiones se debe definir en `NULL`.

```
UPDATE employees
SET job_id = 'IT_PROG', commission_pct = NULL
WHERE employee_id = 114;
```

Nota: la tabla `COPY_EMP` tiene los mismos datos que la tabla `EMPLOYEES`.

Actualización de Dos Columnas con una Subconsulta

Actualizar el cargo y el salario del empleado 113 para que coincida con los del empleado 205.

```
UPDATE employees
SET      job_id  = (SELECT  job_id
                    FROM    employees
                    WHERE   employee_id = 205),
         salary  = (SELECT  salary
                    FROM    employees
                    WHERE   employee_id = 205)
WHERE employee_id = 113;
1 rows updated
```


Actualización de Dos Columnas con una Subconsulta

Puede actualizar varias columnas en la cláusula `SET` de una sentencia `UPDATE` mediante la escritura de varias subconsultas. La sintaxis es la siguiente:

```
UPDATE table
SET      column  =
       (SELECT   column
                      FROM table
                      WHERE condition)
        [ ,
         column  =
                     (SELECT    column
                      FROM table
                      WHERE condition)]
  [WHERE  condition ]   ;
```

El ejemplo de la diapositiva también se puede escribir del siguiente modo:

```
UPDATE employees
SET (job_id, salary)  = (SELECT  job_id, salary
                    FROM    employees
                    WHERE   employee_id = 205)
WHERE    employee_id    =  113;
```

Actualización de Filas Basada en Otra Tabla

Utilizar subconsultas en las sentencias `UPDATE` para actualizar los valores de fila de una tabla según los valores de otra tabla:

```
UPDATE   copy_emp
SET      department_id  =  (SELECT department_id
                           FROM employees
                           WHERE   employee_id = 100)
WHERE    job_id         =  (SELECT job_id
                           FROM employees
                           WHERE  employee_id = 200);
```

```
1 rows updated
```

ORACLE

Actualización de Filas Basada en Otra Tabla

Puede utilizar las subconsultas `UPDATE` para actualizar las filas de una tabla. El ejemplo de la diapositiva actualiza la tabla `COPY_EMP` según los valores de la tabla `EMPLOYEES`. Cambia el número de departamento de todos los empleados con el ID de cargo del empleado 200 al número de departamento actual del empleado 100.

Agenda

- Adición de nuevas filas a una tabla
  - Sentencia `INSERT`
- Cambio de datos en la tabla
  - Sentencia `UPDATE`
- Eliminación de filas de una tabla:
  - Sentencia `DELETE`
  - Sentencia `TRUNCATE`
- Control de transacciones de la base de datos mediante `COMMIT`, `ROLLBACK` y `SAVEPOINT`
- Consistencia de lectura
- Cláusula `FOR UPDATE` en una sentencia `SELECT`

Eliminación de Filas de Tablas

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

Suprimir una fila de la tabla DEPARTMENTS:

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700

ORACLE

Eliminación de Filas de Tablas

El departamento de contratos se ha eliminado de la tabla `DEPARTMENTS` (siempre que no haya violado ninguna restricción en la tabla `DEPARTMENTS`), como se muestra en el gráfico de la diapositiva.

Sentencia DELETE

Puede eliminar filas existentes de una tabla mediante la sentencia DELETE:

```
DELETE [FROM]   table
[WHERE          condition];
```


ORACLE

Sintaxis de Sentencias DELETE

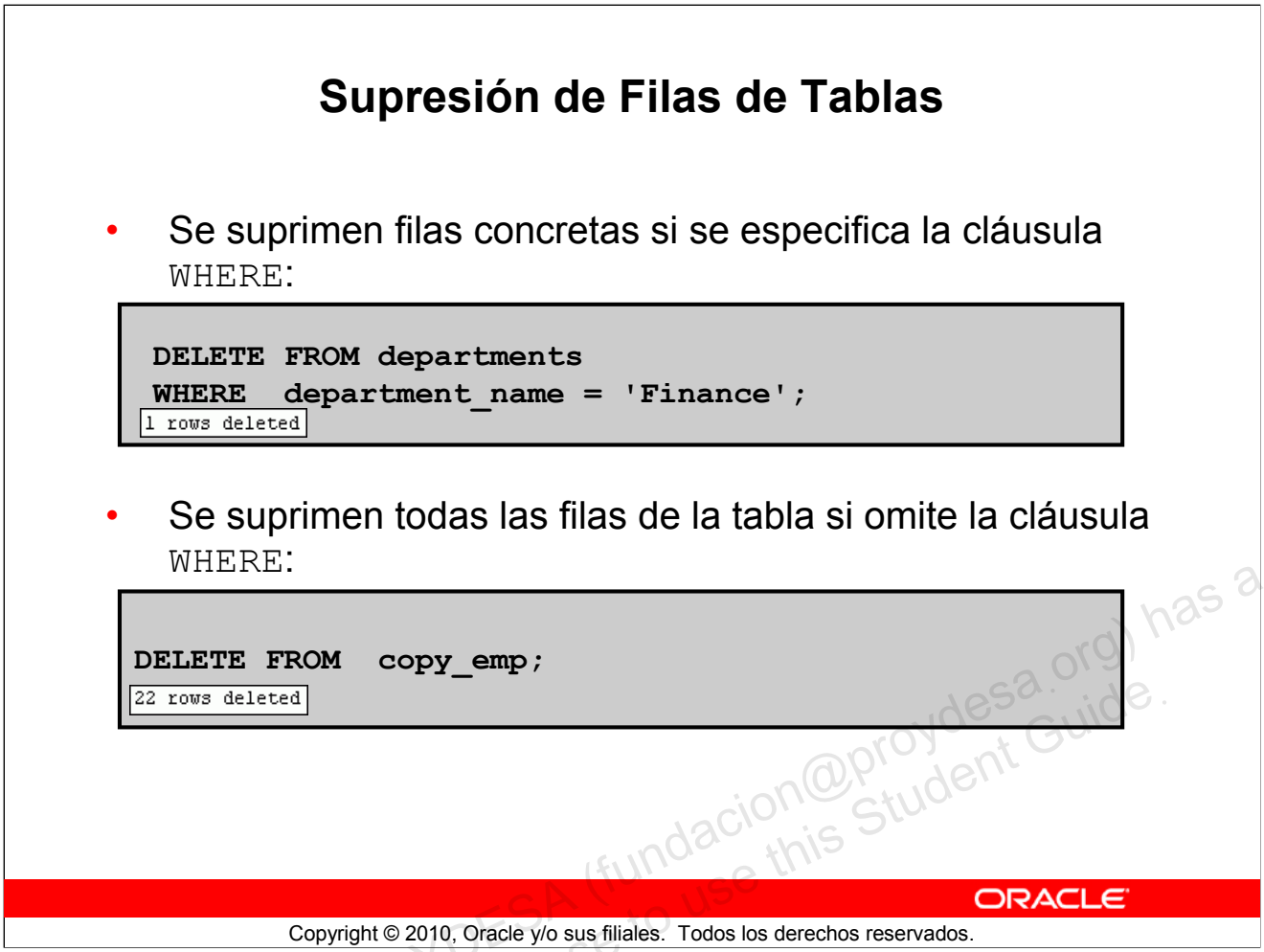
Puede eliminar filas existentes de una tabla mediante la sentencia DELETE.

En la sintaxis:

table	es el nombre de la tabla.
condition	identifica las filas que se deben suprimir y se compone de nombres de columna, expresiones, constantes, subconsultas y operadores de comparación.

Nota: si no se suprime ninguna fila, se devuelve el mensaje "0 rows deleted" (en el separador Script Output de SQL Developer)

Para obtener más información, consulte la sección sobre "DELETE" en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.



Supresión de Filas de Tablas

Puede suprimir filas concretas mediante la especificación de la cláusula `WHERE` en la sentencia `DELETE`. En el primer ejemplo de la diapositiva se suprime el departamento Accounting de la tabla `DEPARTMENTS`. Para confirmar la operación de supresión, visualice las filas suprimidas con la sentencia `SELECT`.

```
SELECT  *
FROM    departments
WHERE  department_name = 'Finance';
```

```
0 rows selected
```

Si omite la cláusula `WHERE`, se suprimen todas las filas de la tabla. En el segundo ejemplo de la diapositiva se suprimen todas las filas de la tabla `COPY_EMP`, porque no se ha especificado ninguna cláusula `WHERE`.

Ejemplo:

Se eliminan las filas identificadas en la cláusula `WHERE`.

```
DELETE FROM  employees WHERE employee_id = 114;
```

```
1 rows deleted
```

```
DELETE FROM  departments WHERE department_id IN (30, 40);
```

```
2 rows deleted
```

Supresión de Filas Basada en Otra Tabla

Utilizar subconsultas en las sentencias `DELETE` para eliminar filas de una tabla según los valores de otra tabla:

```
DELETE FROM employees
WHERE  department_id =
                (SELECT department_id
                 FROM   departments
                 WHERE  department_name
                        LIKE '%Public%');
```

```
1 rows deleted
```


Supresión de Filas Basada en Otra Tabla

Puede utilizar subconsultas para suprimir filas de una tabla según los valores de otra tabla. En el ejemplo de la diapositiva se suprimen todos los empleados de un departamento, donde el nombre del departamento consta de la cadena `Public`.

La subconsulta busca en la tabla `DEPARTMENTS` el número de departamento según el nombre de departamento que contenga la cadena `Public`. A continuación, la subconsulta introduce el número de departamento en la consulta principal, con lo que se suprimen las filas de datos de la tabla `EMPLOYEES` en base a este número de departamento.

Sentencia TRUNCATE

- Elimina todas las filas de una tabla, dejando la tabla vacía y la estructura de la misma intacta.
- Es una sentencia de lenguaje de definición de datos (DDL) en lugar de una sentencia DML; no se puede deshacer fácilmente.
- Sintaxis:

```
TRUNCATE TABLE table_name;
```

- Ejemplo:

```
TRUNCATE TABLE copy_emp;
```


ORACLE

Sentencia TRUNCATE

Una forma más eficaz de vaciar una tabla es con la sentencia `TRUNCATE`.
Puede utilizar la sentencia `TRUNCATE` para eliminar fácilmente todas las filas de una tabla o un cluster. La eliminación de filas con las sentencia `TRUNCATE` es más rápida que la eliminación con la sentencia `DELETE` por las siguientes razones:

- La sentencia `TRUNCATE` es una sentencia de lenguaje de definición de datos (DDL) y no genera ninguna información de rollback. La información de rollback se trata más adelante en esta lección.
- El truncamiento de una tabla no arranca los disparadores de supresión de la tabla.

Si la tabla es el principal de una restricción de integridad referencial, no puede truncarla. Debe desactivar la restricción antes de emitir la sentencia `TRUNCATE`. La desactivación de restricciones se trata en la lección titulada “Uso de Sentencias DDL para Crear y Gestionar Tablas”.

Agenda

- Adición de nuevas filas a una tabla
  - Sentencia `INSERT`
- Cambio de datos en la tabla
  - Sentencia `UPDATE`
- Eliminación de filas de una tabla:
  - Sentencia `DELETE`
  - Sentencia `TRUNCATE`
- Control de transacciones de base de datos mediante `COMMIT`, `ROLLBACK` y `SAVEPOINT`
- Consistencia de lectura
- Cláusula `FOR UPDATE` en una sentencia `SELECT`

ORACLE

Transacciones de Base de Datos

Una transacción de base de datos consiste en una de las siguientes opciones:

- Sentencias DML que constituyen un cambio consistente de los datos
- Una sentencia DDL
- Una sentencia de lenguaje de control de datos (DCL)

ORACLE

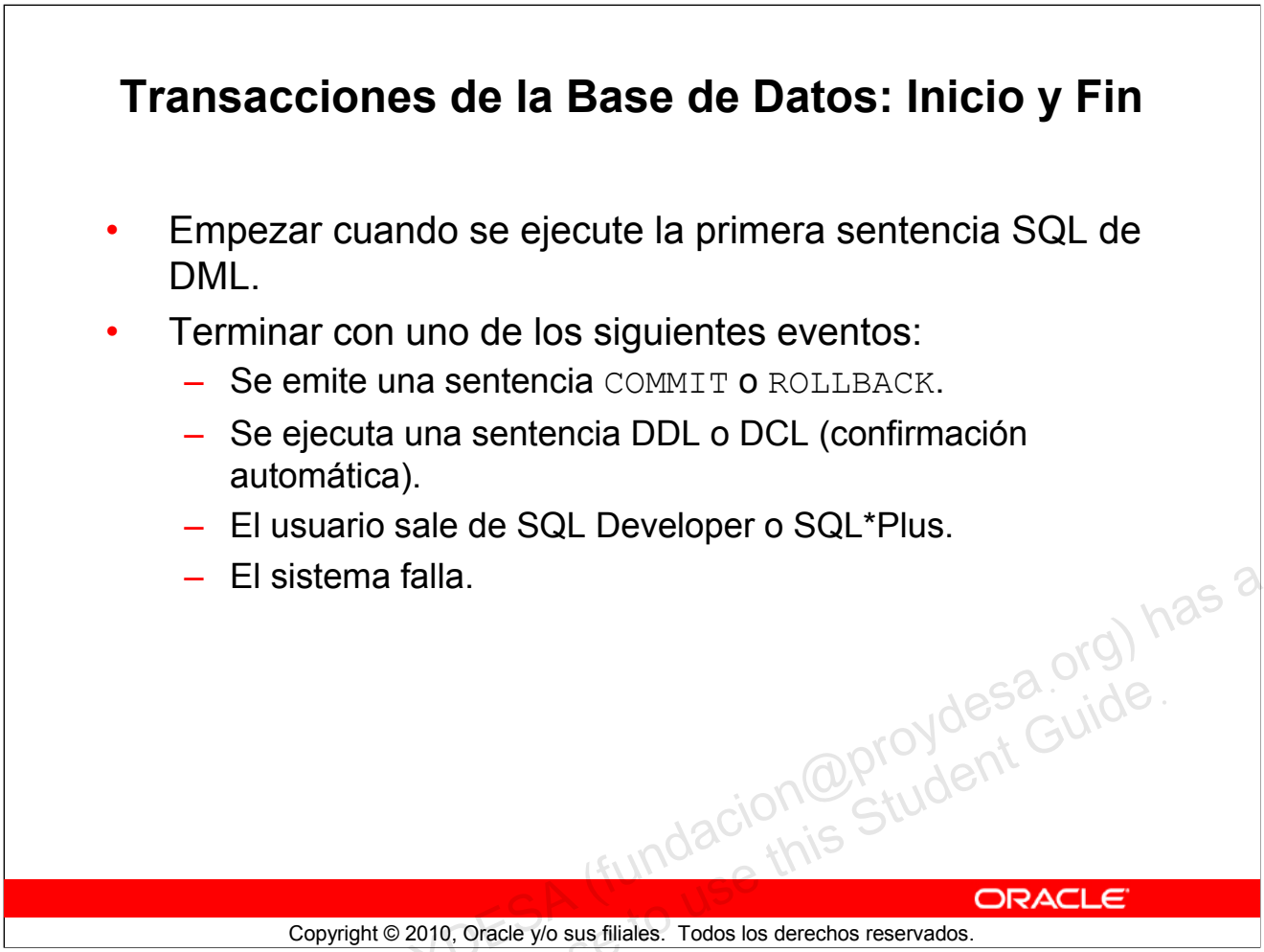
Transacciones de Base de Datos

El servidor de Oracle garantiza la consistencia de los datos basada en transacciones. Las transacciones proporcionan más flexibilidad y control al cambiar los datos y garantizan la consistencia de los datos en caso de fallo de proceso de usuario o del sistema.

Las transacciones constan de sentencias DML que suponen un cambio en los datos. Por ejemplo, una transferencia de fondos entre dos cuentas debe incluir un débito en una cuenta y un crédito en otra por la misma cantidad. Ambas acciones deben ser correctas o incorrectas por igual; el crédito no se debe confirmar sin el débito.

Tipos de Transacción

Tipo	Descripción
Lenguaje de manipulación de datos (DML)	Consta de un número de sentencias DML que el servidor de Oracle trata como una entidad única o unidad de trabajo lógica.
Lenguaje de definición de datos (DDL)	Consta de una única sentencia DDL.
Lenguaje de control de datos (DCL)	Consta de una única sentencia DCL.



Transacción de la Base de Datos: Inicio y Fin

¿Cuándo se inicia y termina una transacción de la base de datos?

Una transacción empieza cuando se encuentra la primera sentencia DML y termina cuando se produce una de las siguientes condiciones:

- Se emite una sentencia `COMMIT` o `ROLLBACK`.
- Se emite una sentencia DDL, como `CREATE`.
- Se emite una sentencia DCL.
- El usuario sale de SQL Developer o SQL*Plus.
- Falla una máquina o el sistema.

Tras la finalización de una transacción, la siguiente sentencia SQL ejecutable inicia automáticamente la próxima transacción.

Se confirma automáticamente una nueva sentencia DDL o DCL y, por lo tanto, finaliza de forma implícita una transacción.

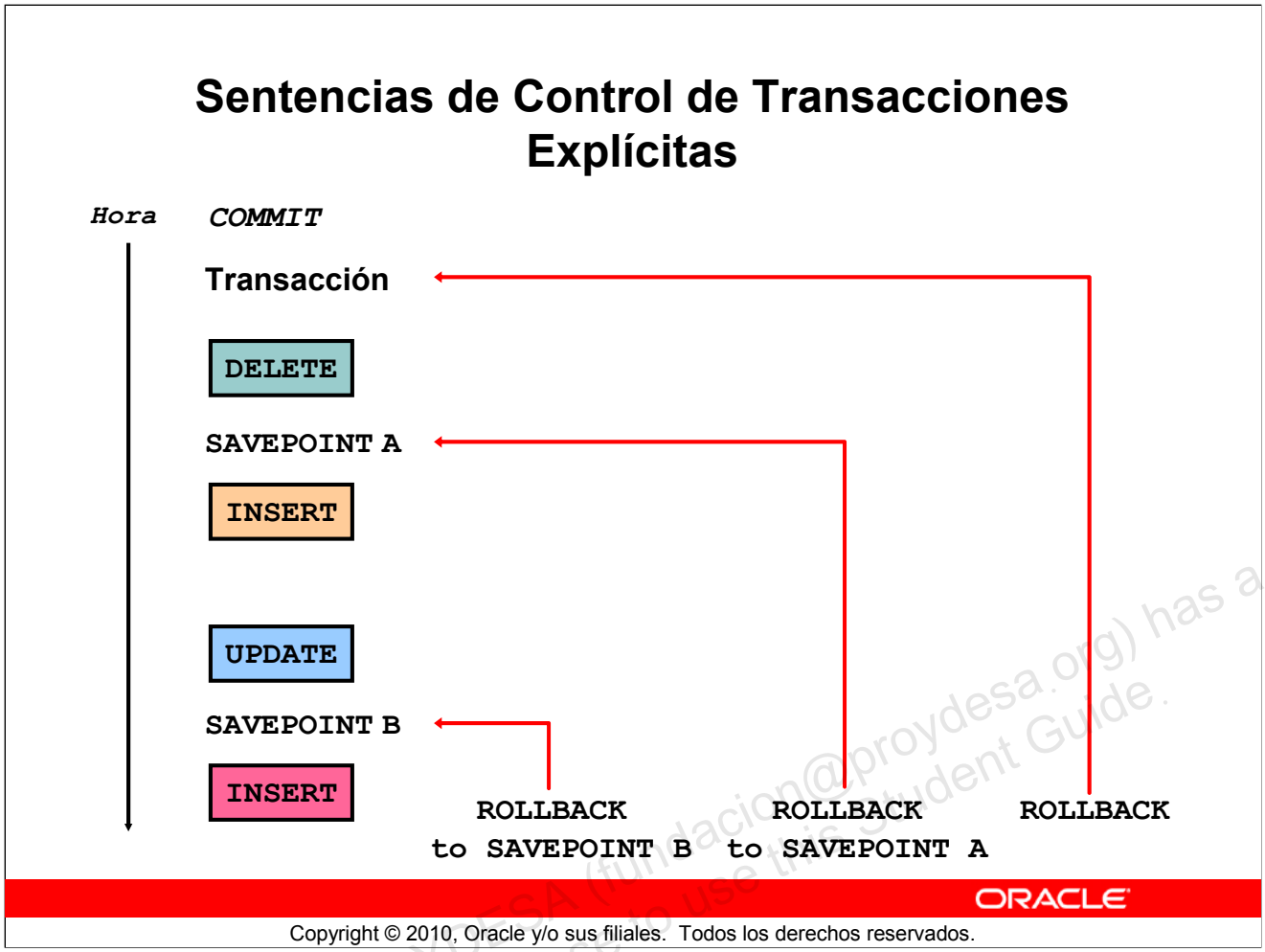
Ventajas de las Sentencias COMMIT y ROLLBACK

Con las sentencias COMMIT y ROLLBACK, puede:

- Garantizar la consistencia
- Visualizar una presentación preliminar de los cambios de los datos antes de hacerlos permanentes
- Agrupar componentes relacionados de forma lógica

Ventajas de las Sentencias COMMIT y ROLLBACK

Con las sentencias COMMIT y ROLLBACK, puede controlar cuándo hacer permanentes los cambios realizados a los datos.



Sentencias de Control de Transacciones Explícitas

Puede controlar la lógica de las transacciones mediante las sentencias `COMMIT`, `SAVEPOINT` y `ROLLBACK`.

Sentencia	Descripción
`COMMIT`	`COMMIT` finaliza la transacción actual convirtiendo todos los cambios de datos pendientes en permanentes.
`SAVEPOINT` *name*	`SAVEPOINT` marca un punto de grabación en la transacción actual.
`ROLLBACK`	`ROLLBACK` finaliza la transacción actual descartando todos los cambios de datos pendientes.
`ROLLBACK TO` *SAVEPOINT name*	`ROLLBACK TO SAVEPOINT` realiza un rollback de la transacción actual en el punto de grabación especificado, descartando de esta forma la posibilidad de realizar cambios y puntos de grabación creados después del punto de grabación en el que está realizando el rollback. Si omite la cláusula TO SAVEPOINT, la sentencia ROLLBACK realiza un rollback de toda la transacción. Puesto que los puntos de grabación son lógicos, no hay forma de mostrar lo que ha creado.

Nota: no puede aplicar la sentencia `COMMIT` en `SAVEPOINT`. `SAVEPOINT` no es SQL estándar de ANSI.

Rollback de los Cambios de un Marcador

- Crear un marcador en una transacción actual mediante la sentencia `SAVEPOINT`.
- Realizar rollback en dicho marcador mediante la sentencia `ROLLBACK TO SAVEPOINT`.

```
UPDATE...
SAVEPOINT update_done;
SAVEPOINT update_done succeeded.
INSERT...
ROLLBACK TO update_done;
ROLLBACK TO succeeded.
```


Rollback de los Cambios de un Marcador

Puede crear un marcador en la transacción actual mediante la sentencia `SAVEPOINT`, que divide la transacción en secciones más pequeñas. Puede descartar los cambios pendientes hasta dicho marcador con la sentencia `ROLLBACK TO SAVEPOINT`.

Si crea un segundo punto de grabación con el mismo nombre que un punto de grabación anterior, éste se suprime.

Procesamiento de Transacciones Implícitas

- Una transacción automática se produce en las siguientes circunstancias:
  - Se emite una sentencia DDL.
  - Se emite una sentencia DCL.
  - Salida normal de SQL Developer o SQL*Plus, sin emitir explícitamente las sentencias `COMMIT` o `ROLLBACK`.
- Un rollback automático se produce con la terminación anormal de SQL Developer o SQL*Plus o por un fallo del sistema.

Procesamiento de Transacciones Implícitas

Estado	Circumstances
Confirmación automática	Se emite una sentencia DDL o DCL, se sale normalmente de SQL*Plus sin emitir explícitamente los comandos COMMIT o ROLLBACK
Rollback automático	Terminación anormal de SQL Developer o SQL*Plus o fallo del sistema

Nota: en SQL*Plus, el comando `AUTOCOMMIT` se puede activar o desactivar. Si se define en ON, cada sentencia DML individual se confirma en cuanto se ejecuta. No puede realizar rollback de los cambios. Si se define en OFF, la sentencia `COMMIT` aún se puede emitir explícitamente. Además, la sentencia `COMMIT` se emite al emitir una sentencia DDL o al salir de SQL*Plus. El comando `SET AUTOCOMMIT ON/OFF` se salta en SQL Developer. DML se confirma en una salida normal de SQL Developer sólo si tiene activada la preferencia Autocommit. Para activar Autocommit, realice los siguientes pasos:

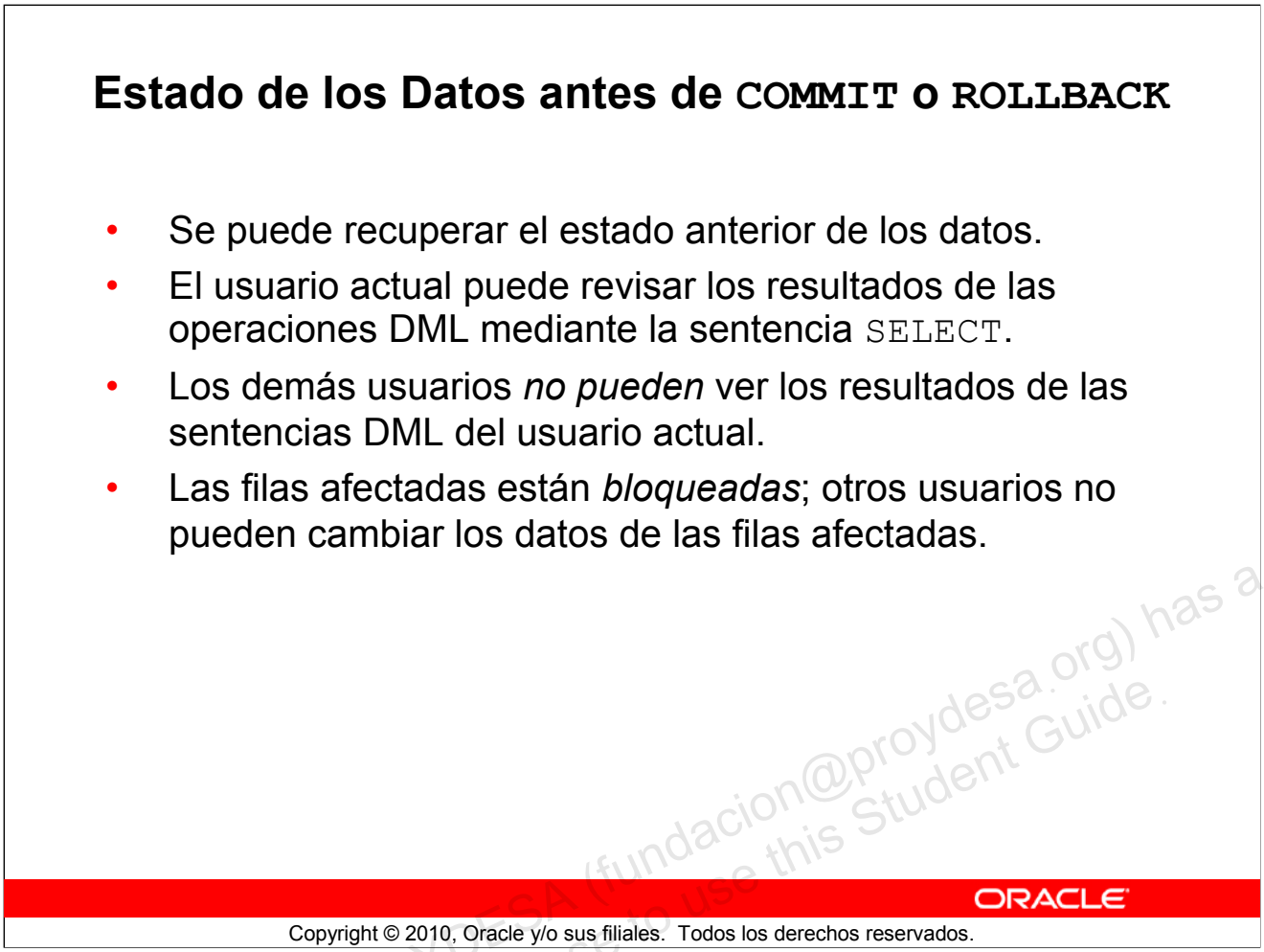
- En el menú Tools, seleccione Preferences. En el cuadro de diálogo Preferences, amplíe Database y seleccione Worksheet Parameters.
- En el panel de la derecha, seleccione la opción “Autocommit in SQL Worksheet”. Haga clic en OK.

Procesamiento de Transacciones Implícitas (continuación)

Fallos del Sistema

Cuando se interrumpe una transacción debido a un fallo del sistema, se realiza un rollback automático de toda la transacción. Esto impide que el error provoque cambios no deseados en los datos y que devuelva las tablas a su estado en el momento de la última confirmación. De esta forma, el servidor de Oracle protege la integridad de las tablas.

En SQL Developer, la salida normal de la sesión se realiza seleccionando Exit en el menú File. Con SQL*Plus, la salida normal se realiza introduciendo el comando `EXIT` en el símbolo del sistema. Si se cierra la ventana, se interpreta como una salida anormal.



Estado de los Datos antes de COMMIT o ROLLBACK

Todos los cambios de datos realizados durante la transacción son temporales hasta que se confirma la transacción.

El estado de los datos antes de emitir las sentencias COMMIT o ROLLBACK se describe a continuación:

- Las operaciones de manipulación de datos afectan principalmente al buffer de la base de datos y, por lo tanto, se puede recuperar el estado anterior de los datos.
- El usuario actual puede revisar los resultados de las operaciones de manipulación de datos mediante la consulta de las tablas.
- Los demás usuarios no pueden ver los resultados de las operaciones de manipulación realizadas por el usuario actual. El servidor de Oracle establece la consistencia de lectura para garantizar que todos los usuarios ven los datos tal y como estaban en el momento de la última confirmación.
- Las filas afectadas están bloqueadas; otros usuarios no pueden cambiar los datos de las filas afectadas.

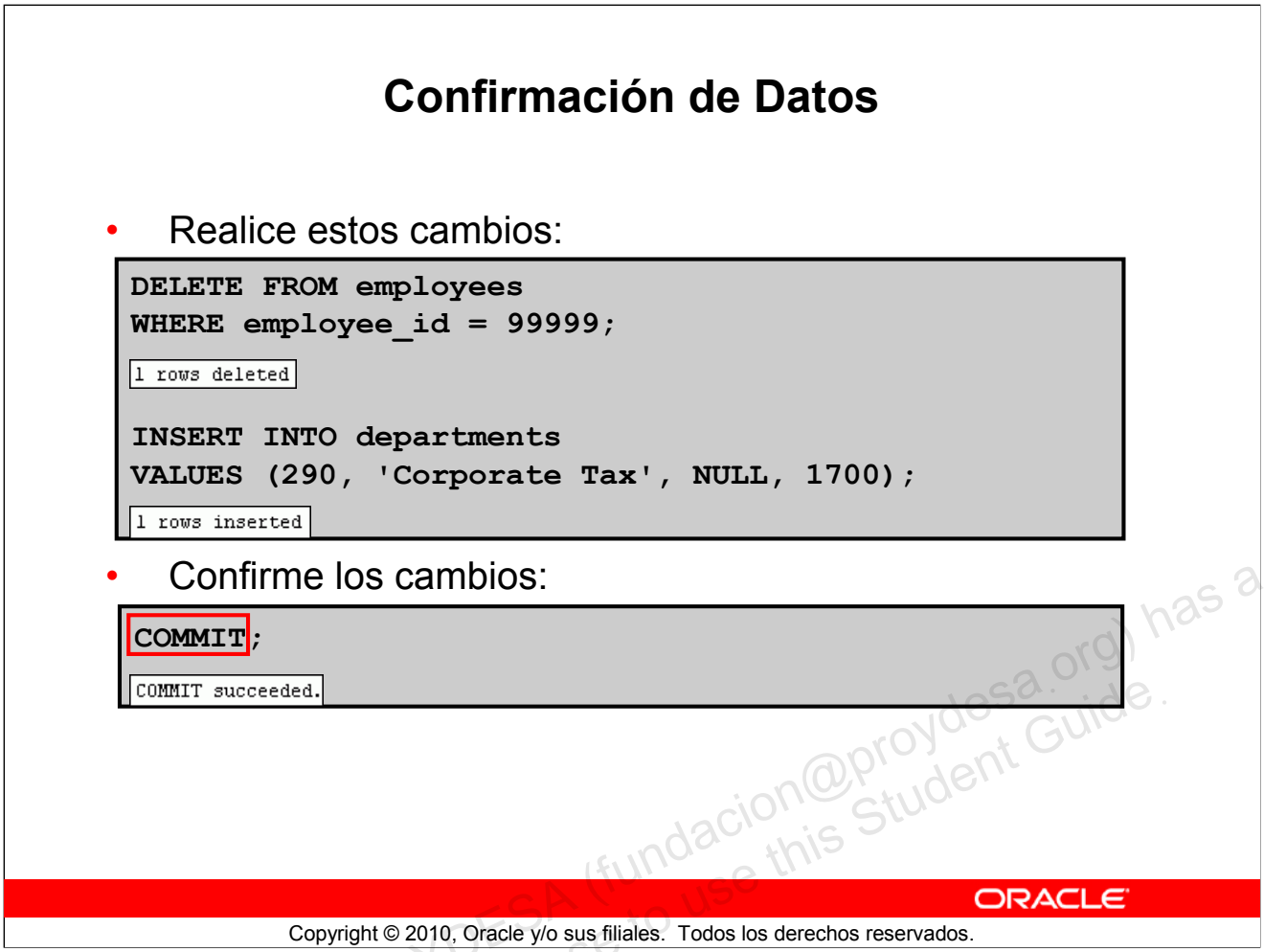
Estado de los Datos después de COMMIT

- Los cambios de datos se guardan en la base de datos.
- Se sobrescribe el estado anterior de los datos.
- Todos los usuarios pueden ver los resultados.
- Los bloqueos de las filas afectadas se liberan y dichas filas quedan disponibles para que las manipulen otros usuarios.
- Se borran todos los puntos de grabación.

Estado de los Datos después de COMMIT

Para convertir todos los cambios pendientes en permanentes, utilice la sentencia COMMIT. A continuación, se muestra lo que ocurre después de ejecutar una sentencia COMMIT:

- Los cambios de datos se escriben en la base de datos.
- El estado anterior de los datos ya no está disponible con las consultas SQL normales.
- Todos los usuarios pueden ver los resultados de la transacción.
- Los bloqueos de las filas afectadas se liberan y las filas quedan ahora disponibles para que otros usuarios realicen nuevos cambios en los datos.
- Se borran todos los puntos de grabación.



Confirmación de Datos

En el ejemplo de la diapositiva, se suprime una fila de la tabla `EMPLOYEES` y se inserta una nueva en la tabla `DEPARTMENTS`. Los cambios se guardan emitiendo la sentencia `COMMIT`.

Ejemplo:

Elimine los departamentos 290 y 300 de la tabla `DEPARTMENTS` y actualice una fila en la tabla `EMPLOYEES`. Guarde el cambio de datos.

```
DELETE FROM departments
WHERE   department_id IN (290, 300);

UPDATE employees
  SET    department_id = 80
  WHERE employee_id = 206;

COMMIT;
```

Estado de los Datos después de ROLLBACK

Desechar todos los cambios pendientes mediante la sentencia ROLLBACK:

- Se deshacen los cambios de datos.
- Se restaura el estado anterior de los datos.
- Se liberan los bloqueos de las filas afectadas.

```
DELETE FROM copy_emp;
ROLLBACK ;
```


Estado de los Datos después de ROLLBACK

Para descartar todos los cambios pendientes, utilice la sentencia ROLLBACK obteniendo los siguientes resultados:

- Se deshacen los cambios de datos.
- Se restaura el estado anterior de los datos.
- Se liberan los bloqueos de las filas afectadas.

Estado de los Datos después de ROLLBACK: Ejemplo

```
DELETE FROM test;
25,000 rows deleted.

ROLLBACK ;
Rollback complete.

DELETE FROM test WHERE  id = 100;
1 row deleted.

SELECT * FROM   test WHERE  id = 100;
No rows selected.

COMMIT;
Commit complete.
```

ORACLE

Estado de los Datos después de ROLLBACK: Ejemplo

Al intentar eliminar un registro de la tabla TEST, puede borrar accidentalmente la tabla. Puede corregir el error, volver a emitir la sentencia correcta y hacer permanentes los cambios de datos.

Rollback a Nivel de Sentencias

- Si falla una sentencia DML durante la ejecución, sólo se realiza un rollback de dicha sentencia.
- El servidor de Oracle implanta un punto de grabación implícito.
- Los demás cambios se retienen.
- El usuario debe terminar las transacciones explícitamente con la ejecución de una sentencia `COMMIT` o `ROLLBACK`.

ORACLE

Rollback a Nivel de Sentencias

Se puede desechar parte de una transacción mediante un rollback implícito si se detecta un error de ejecución de sentencia. Si falla una única sentencia DML durante la ejecución de una transacción, su efecto se deshace mediante un rollback a nivel de sentencia, pero los cambios realizados por las sentencias DML anteriores en la transacción no se desechan. El usuario los puede omitir o realizar un rollback explícito.

El servidor de Oracle emite una confirmación implícita antes y después de cualquier sentencia DDL. Por lo tanto, si la sentencia DDL no se ejecuta correctamente, no podrá realizar rollback de la sentencia anterior porque el servidor ha emitido una confirmación.

El usuario debe terminar las transacciones explícitamente con la ejecución de una sentencia `COMMIT` o `ROLLBACK`.

Agenda

- Adición de nuevas filas a una tabla
  - Sentencia `INSERT`
- Cambio de datos en la tabla
  - Sentencia `UPDATE`
- Eliminación de filas de una tabla:
  - Sentencia `DELETE`
  - Sentencia `TRUNCATE`
- Control de transacciones de base de datos mediante `COMMIT`, `ROLLBACK` y `SAVEPOINT`
- Consistencia de lectura
- Cláusula `FOR UPDATE` en una sentencia `SELECT`

Consistencia de Lectura

- La consistencia de lectura garantiza una vista consistente de los datos en todo momento.
- Los cambios realizados por un usuario no entran en conflicto con los cambios realizados por otro usuario.
- La consistencia de lectura garantiza que en los mismos datos:
  - Los lectores no esperen los escritores.
  - Los escritores no esperen a los lectores.
  - Los escritores esperen a los escritores.

ORACLE

Consistencia de Lectura

Los usuarios de la base de datos pueden acceder a la base de datos de dos formas:

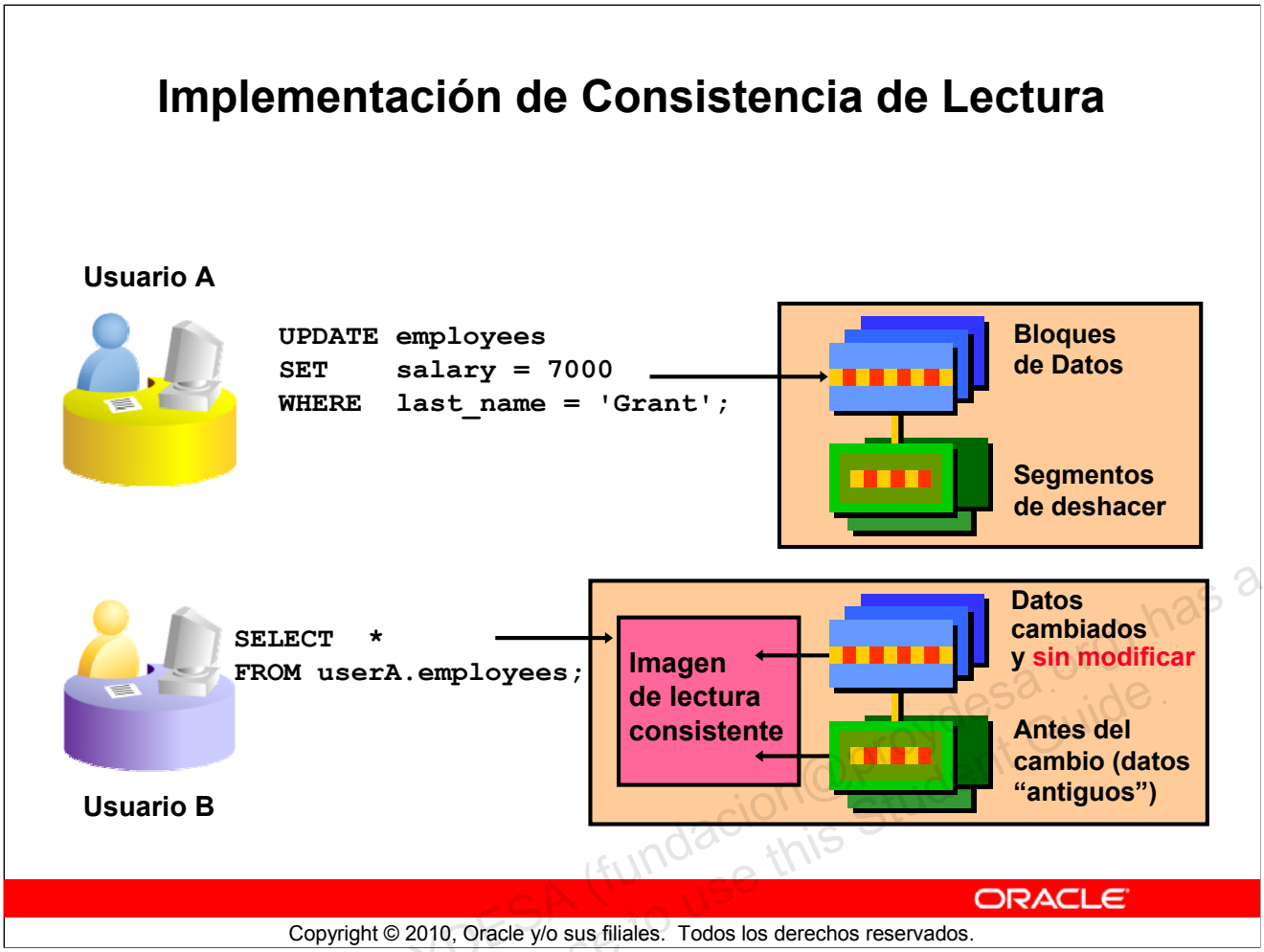
- Operaciones de lectura (sentencia `SELECT`)
- Operaciones de escritura (sentencias `INSERT`, `UPDATE`, `DELETE`)

La consistencia de lectura es necesaria para lo siguiente:

- Que el lector y el escritor de la base de datos tengan garantizada una vista consistente de los datos.
- Que los lectores no visualicen datos en proceso de cambio.
- Que los escritores tengan garantizado que los cambios en la base de datos se realizarán de forma consistente.
- Los cambios realizados por un escritor no entran en conflicto con los cambios realizados por otro escritor.

El objetivo de la consistencia es garantizar que todos los usuarios vean los datos como existían en el momento de la confirmación, antes de iniciar una operación DML.

Nota: el mismo usuario se puede conectar a diferentes sesiones. Cada sesión mantiene la consistencia de lectura tal y como se describe anteriormente, incluso si son los mismos usuarios.



Implementación de Consistencia de Lectura

La consistencia de lectura es una implantación automática. Mantiene una copia parcial de la base de datos en los segmentos de deshacer. La imagen de lectura consistente se crea a partir de los datos confirmados de la tabla y de los datos antiguos que se van a cambiar y que aún no se han confirmado del segmento de deshacer.

Al realizar una operación de inserción, actualización o supresión en la base de datos, el servidor de Oracle realiza una copia de los datos antes de cambiarlos y los escribe en un *segmento de deshacer*.

Todos los lectores, excepto el que haya emitido el cambio, seguirán viendo la base de datos como existía antes de que comenzaran los cambios; verán una "instantánea" de los datos del segmento de deshacer.

Antes de confirmar los cambios en la base de datos, sólo el usuario que modifica los datos ve la base de datos con modificaciones. Todos los demás verán la instantánea en el segmento de deshacer. Esto garantiza que los lectores de los datos lean datos consistentes en los que no se esté realizando actualmente ningún cambio.

Al confirmar una sentencia DML, el cambio realizado en la base de datos se hace visible para todos aquéllos que emiten una sentencia `SELECT` *después de* realizar la confirmación. El espacio ocupado por los datos *antiguos* en el archivo de segmentos de deshacer se libera para volver a utilizarlo.

Si se realiza un rollback de la transacción, los cambios se deshacen:

- La versión original anterior de los datos del segmento de deshacer se vuelve a escribir en la tabla.
- Todos los usuarios ven la base de datos como existía antes de comenzar la transacción.

Agenda

- Adición de nuevas filas a una tabla
  - Sentencia `INSERT`
- Cambio de datos en la tabla
  - Sentencia `UPDATE`
- Eliminación de filas de una tabla:
  - Sentencia `DELETE`
  - Sentencia `TRUNCATE`
- Control de transacciones de la base de datos mediante `COMMIT`, `ROLLBACK` y `SAVEPOINT`
- Consistencia de lectura
- Cláusula `FOR UPDATE` en una sentencia `SELECT`

ORACLE

Cláusula FOR UPDATE en una Sentencia SELECT

- Bloquea las filas de la tabla EMPLOYEES en las que job_id sea SA_REP.

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE
ORDER BY employee_id;
```

- El bloqueo sólo se libera si emite ROLLBACK o COMMIT.
- Si la sentencia SELECT intenta bloquear una fila bloqueada por otro usuario, la base de datos espera a que la fila esté disponible y, a continuación, devuelve los resultados de la sentencia SELECT.

ORACLE

Cláusula FOR UPDATE en una Sentencia SELECT

Cuando se emite una sentencia SELECT en la base de datos para consultar algunos registros, no se coloca ningún bloqueo en las filas seleccionadas. En general, se debe a que el número de registros bloqueados en un momento determinado se mantiene (por defecto) en el mínimo absoluto: sólo se bloquean aquellos registros que se han cambiado, pero que aún no se han confirmado. Aún así, otros usuarios podrán leer dichos registros tal y como aparecían antes del cambio (la "imagen anterior" de los datos). Hay ocasiones, sin embargo, en las que puede que desee bloquear un juego de registros incluso antes de cambiarlos en el programa. Oracle ofrece la cláusula FOR UPDATE de la sentencia SELECT para realizar este bloqueo.

Cuando emite una sentencia SELECT...FOR UPDATE, el sistema de gestión de bases de datos relacionales (RDBMS) obtiene automáticamente los bloqueos a nivel de fila exclusivos de todas las filas identificadas por la sentencia SELECT, reteniendo así los registros "sólo para sus cambios". Nadie más podrá cambiar ninguno de estos registros hasta que realice una operación ROLLBACK o COMMIT.

Puede adjuntar la palabra clave opcional NOWAIT a la cláusula FOR UPDATE para indicar al servidor de Oracle que no espere si otro usuario ha bloqueado la tabla. En este caso, el control volverá inmediatamente al programa o al entorno de SQL Developer para que pueda realizar otro trabajo o simplemente esperar un período de tiempo antes de volver a intentarlo. Sin la cláusula NOWAIT, el proceso se bloqueará hasta que la tabla esté disponible, cuando otro usuario libere los bloqueos a través de la emisión un comando COMMIT o ROLLBACK.

Cláusula FOR UPDATE: Ejemplos

- Puede utilizar la cláusula FOR UPDATE en una sentencia SELECT en varias tablas.

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK'
AND location_id = 1500
FOR UPDATE
ORDER BY e.employee_id;
```

- Las filas de las tablas EMPLOYEES y DEPARTMENTS están bloqueadas.
- Utilizar FOR UPDATE OF *column_name* para cualificar la columna que desea cambiar, de modo que sólo se bloqueen las filas de esa tabla específica.

ORACLE

Cláusula FOR UPDATE: Ejemplos

En el ejemplo de la diapositiva, la sentencia bloquea las filas de la tabla EMPLOYEES con JOB_ID definido en ST_CLERK y LOCATION_ID definido en 1500 y bloquea las filas de la tabla DEPARTMENTS con los departamentos de LOCATION_ID definidos en 1500.

Puede utilizar FOR UPDATE OF *column_name* para cualificar la columna que desea cambiar. La lista OF de la cláusula FOR UPDATE no limita el cambio sólo a aquellas columnas de las filas seleccionadas. Aún se mantienen los bloqueos en todas las filas; si simplemente declara FOR UPDATE en la consulta y no incluye una o más columnas después de la palabra clave OF, la base de datos bloqueará todas las filas identificadas en todas las tablas incluidas en la cláusula FROM.

La siguiente sentencia sólo bloquea las filas de la tabla EMPLOYEES con ST_CLERK ubicado en LOCATION_ID 1500. No se bloquea ninguna fila en la tabla DEPARTMENTS:

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK' AND location_id = 1500
FOR UPDATE OF e.salary
ORDER BY e.employee_id;
```

Cláusula `FOR UPDATE`: Ejemplos (continuación)

En el siguiente ejemplo, se indica a la base de datos que espere cinco segundos hasta que la fila esté disponible y, a continuación, se devuelve el control al usuario.

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE WAIT 5
ORDER BY employee_id;
```

Prueba

Las siguientes sentencias producen los mismos resultados:

```
DELETE FROM copy_emp;
```

```
TRUNCATE TABLE copy_emp;
```

1. Verdadero
2. Falso

ORACLE

Respuesta: 2

Resumen

En esta lección debe haber aprendido a utilizar las siguientes sentencias:

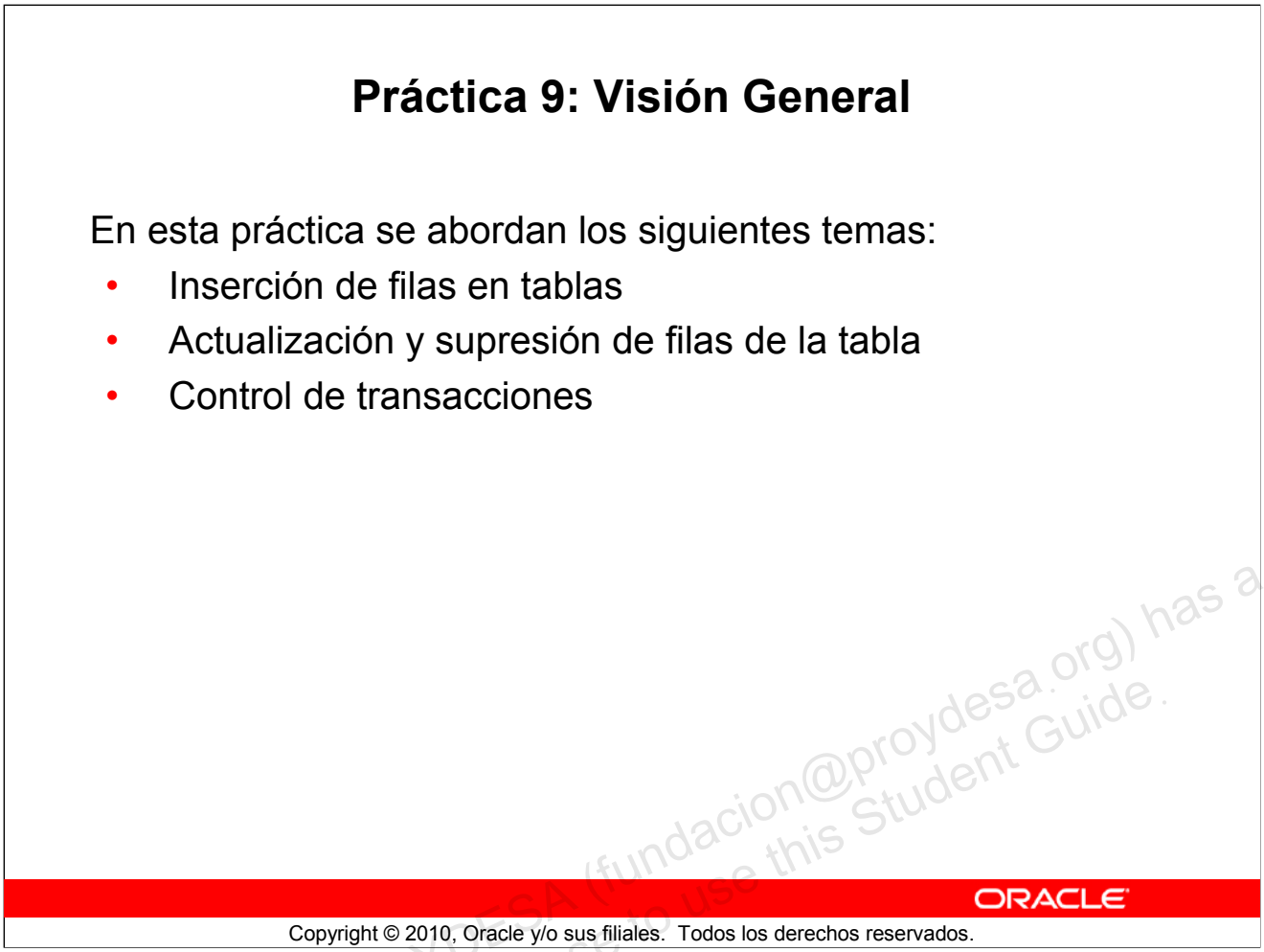
Función	Descripción
INSERT	Agrega una nueva fila a la tabla.
UPDATE	Modifica filas existentes en la tabla.
DELETE	Elimina filas existentes de la tabla.
TRUNCATE	Elimina todas las filas de una tabla.
COMMIT	Convierte todos los cambios pendientes en
SAVEPOINT	Se utiliza para realizar un rollback en el marcador de punto de grabación.
ROLLBACK	Descarta todos los cambios de datos pendientes.
Cláusula FOR UPDATE en SELECT	Bloquea las filas identificadas por la consulta SELECT.

ORACLE

Resumen

En esta lección, debe haber aprendido a manipular datos en Oracle Database mediante las sentencias INSERT, UPDATE, DELETE y TRUNCATE, así como a controlar los cambios de datos mediante las sentencias COMMIT, SAVEPOINT y ROLLBACK. También ha aprendido a utilizar la cláusula FOR UPDATE de la sentencia SELECT para bloquear filas sólo para sus cambios.

Recuerde que el servidor de Oracle garantiza una vista consistente de los datos en todo momento.



Práctica 9: Visión General

En esta práctica, agregará filas a la tabla `MY_EMPLOYEE`, actualizará y suprimirá datos de la tabla y controlará las transacciones. Ejecute un script para crear la tabla `MY_EMPLOYEE`.

10

Uso de Sentencias DDL para Crear y Gestionar Tablas

ORACLE

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Clasificar los principales objetos de base de datos
- Revisar la estructura de la tabla
- Mostrar los tipos de dato disponibles para columnas
- Crear una tabla simple
- Explicar cómo crear restricciones en el momento de la creación de la tabla
- Describir el funcionamiento de los objetos de esquema

ORACLE

Objetivos

En esta lección, se ofrece una introducción a las sentencias de lenguaje de definición de datos (DDL). Aprenderá los fundamentos para crear tablas simples, modificarlas y eliminarlas. Se muestran los tipos de dato disponibles en DDL y se ofrece una introducción sobre los conceptos de esquema. Las restricciones se tratarán en esta lección. Se muestran y explican los mensajes de excepción generados por las restricciones de violación durante las operaciones DML.

Agenda

- Objetos de base de datos
  - Reglas de nomenclatura
- Sentencia `CREATE TABLE`:
  - Acceso a otras tablas de usuario
  - Opción `DEFAULT`
- Tipos de dato
- Visión general de restricciones: Restricciones `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`
- Creación de una tabla mediante una subconsulta
- `ALTER TABLE`
  - Tablas de sólo lectura
- Sentencia `DROP TABLE`

ORACLE

Objetos de Base de Datos

Objeto	Descripción
Tabla	Unidad básica de almacenamiento; compuesta por filas.
Vista	Representa de forma lógica subjuegos de datos de una o más tablas.
Secuencia	Genera valores numéricos.
Índice	Mejora el rendimiento de algunas consultas.
Sinónimo	Ofrece nombres alternativos para un objeto

ORACLE

Objetos de Base de Datos

Oracle Database puede contener varias estructuras de datos. Cada estructura se debe describir en el diseño de la base de datos para que se pueda crear durante la fase de creación del desarrollo de la base de datos.

- **Tabla:** almacena datos.
- **Vista:** subjuego de datos de una o más tablas.
- **Secuencia:** genera valores numéricos.
- **Índice:** mejora el rendimiento de algunas consultas.
- **Sinónimo:** ofrece nombres alternativos para un objeto.

Estructuras de Tabla de Oracle

- Las tablas se pueden crear en cualquier momento, incluso mientras los usuarios utilizan la base de datos.
- No necesita especificar el tamaño de la tabla. El tamaño se define en última instancia según la cantidad de espacio asignado a la base de datos completa. No obstante, es importante calcular la cantidad de espacio que utilizará una tabla a lo largo del tiempo.
- La estructura de la tabla se puede modificar en línea.

Nota: hay más objetos de base de datos disponibles, pero no se tratan en este curso.

Reglas de Nomenclatura

Los nombres de tabla y de columna deben:

- Empezar por una letra
- Tener entre 1 y 30 caracteres
- Contener sólo A–Z, a–z, 0–9, _, $ y #
- No deben ser un duplicado de otro nombre de objeto propiedad del mismo usuario
- No debe ser una palabra reservada del servidor de Oracle

Reglas de Nomenclatura

Las tablas y columnas de la base de datos se deben nombrar según las reglas estándar de nomenclatura de objetos de Oracle Database:

- Los nombres de tabla y de columna deben comenzar por una letra y tener entre 1 y 30 caracteres.
- Los nombres sólo deben contener los caracteres A–Z, a–z, 0–9, _ (subrayado), $ y # (caracteres válidos, pero cuyo uso no es aconsejable).
- Los nombres no deben ser un duplicado de otro nombre de objeto propiedad del mismo usuario del servidor de Oracle.
- Los nombres no deben ser una palabra reservada del servidor de Oracle.
  - Puede utilizar identificadores entre comillas para representar el nombre de un objeto. Un identificador entre comillas empieza y finaliza con comillas dobles (""). Si asigna un nombre a un objeto de esquema mediante un identificador entre comillas, debe utilizar las comillas dobles cuando haga referencia a dicho objeto. Los identificadores entre comillas pueden ser palabras reservadas, aunque no se recomienda.

Reglas de Nomenclatura (continuación)

Directrices de Nomenclatura

Utilice nombres descriptivos para las tablas y otros objetos de la base de datos.

Nota: los nombres no son sensibles a mayúsculas/minúsculas. Por ejemplo, `EMPLOYEES` se considera el mismo nombre que `eMPloyees` o `eMpLOYEES`. Sin embargo, los identificadores entre comillas son sensibles a mayúsculas/minúsculas.

Para obtener más información, consulte la sección sobre nombres de objeto de esquema y cualificadores en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Agenda

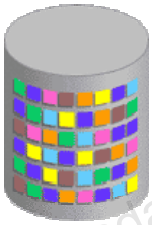
- Objetos de base de datos
  - Reglas de nomenclatura
- Sentencia `CREATE TABLE`:
  - Acceso a otras tablas de usuario
  - Opción `DEFAULT`
- Tipos de dato
- Visión general de restricciones: Restricciones `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`
- Creación de una tabla mediante una subconsulta
- `ALTER TABLE`
  - Tablas de sólo lectura
- Sentencia `DROP TABLE`

Sentencia CREATE TABLE

- Debe tener:
  - El privilegio CREATE TABLE
  - Un área de almacenamiento

```
CREATE TABLE [schema.]table
        (column datatype [DEFAULT expr][, ...]);
```

- Debe especificar:
  - El nombre de tabla
  - El nombre de columna, tipo de dato de columna y tamaño de columna



ORACLE

Sentencia CREATE TABLE

Puede crear tablas para almacenar datos ejecutando la sentencia SQL CREATE TABLE. Esta sentencia es una de las sentencias DDL, que son un subjuego de sentencias SQL que se utilizan para crear, modificar o eliminar estructuras de Oracle Database. Estas sentencias tienen un efecto inmediato en la base de datos y registran información en el diccionario de datos.

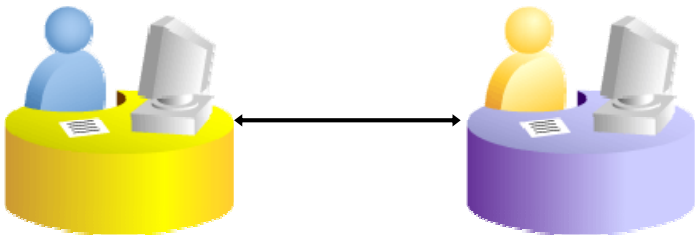
Para crear una tabla, un usuario debe tener el privilegio CREATE TABLE y un área de almacenamiento en la que crear los objetos. El administrador de la base de datos (DBA) utiliza sentencias de lenguaje de control de datos (DCL) para otorgar privilegios a los usuarios.

En la sintaxis:

schema	es el mismo nombre que el del propietario
table	es el nombre de la tabla
DEFAULT *expr*	especifica un valor por defecto si se omite un valor en la sentencia INSERT
column	es el nombre de la columna
datatype	es el tipo de dato y la longitud de la columna

Referencia a Tablas de Otro Usuario

- Las tablas que pertenecen a otros usuarios no se incluyen en el esquema del usuario.
- Debe utilizar el nombre del propietario como prefijo de dichas tablas.



USERA

```
SELECT *
FROM userB.employees;
```

USERB

```
SELECT *
FROM userA.employees;
```

ORACLE

Referencia a Tablas de Otro Usuario

Un esquema es una recopilación de estructuras lógicas de datos u *objetos de esquema*. Un esquema es propiedad de un usuario de base de datos y tiene el mismo nombre que el usuario. Cada usuario es propietario de un único esquema.

Los objetos de esquema se pueden crear y manipular con SQL e incluyen tablas, vistas, sinónimos, secuencias, procedimientos almacenados, índices, clusters y enlaces de base de datos.

Si una tabla no pertenece al usuario, el nombre del propietario se debe anteponer a la tabla. Por ejemplo, dados los esquemas `USERA` y `USERB` y ambos contienen la tabla `EMPLOYEES`, si `USERA` desea acceder a la tabla `EMPLOYEES` que pertenece a `USERB`, `USERA` debe anteponer el nombre del esquema al nombre de la tabla:

```
SELECT *
FROM   userb.employees;
```

Si `USERB` desea acceder a la tabla `EMPLOYEES` propiedad de `USERA`, `USERB` debe anteponer el nombre del esquema al nombre de la tabla:

```
SELECT *
FROM   usera.employees;
```

Opción DEFAULT

- Especificar un valor por defecto para una columna durante una inserción.

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- Los valores literales, expresiones o funciones SQL son valores válidos.
- El nombre de otra columna o una pseudocolumna son valores no válidos.
- El tipo de dato por defecto debe coincidir con el tipo de dato de la columna.

```
CREATE TABLE hire_dates
        (id          NUMBER(8),
        hire_date DATE DEFAULT SYSDATE);
CREATE TABLE succeeded.
```

ORACLE

Opción DEFAULT

Al definir una tabla, puede especificar que se proporcione un valor por defecto a una columna mediante la opción `DEFAULT`. Esta opción evita que se introduzcan valores nulos en las columnas si se inserta una fila sin un valor para la columna. El valor por defecto puede ser un literal, una expresión o una función SQL (como `SYSDATE` o `USER`), pero el valor no puede ser el nombre de otra columna o una pseudocolumna (como `NEXTVAL` o `CURRVAL`). La expresión por defecto debe coincidir con el tipo de dato de la columna.

Considere los siguientes ejemplos:

```
INSERT INTO hire_dates values(45, NULL);
```

La sentencia anterior insertará el valor nulo en lugar del valor por defecto.

```
INSERT INTO hire_dates(id) values(35);
```

La sentencia anterior insertará `SYSDATE` para la columna `HIRE_DATE`.

Nota:en SQL Developer, haga clic en el icono Run Script o pulse [F5] para ejecutar las sentencias DDL. Los comentarios se mostrarán en la página con separadores Script Output.

Creación de Tablas

- Crear una tabla:

```
CREATE TABLE dept
        (deptno      NUMBER(2),
         dname       VARCHAR2(14),
         loc         VARCHAR2(13),
         create_date DATE DEFAULT SYSDATE);
```

```
CREATE TABLE succeeded.
```

- Confirmar la creación de la tabla:

```
DESCRIBE dept
```

```
DESCRIBE dept
Name                          Null     Type
----------------------------- -------- ----------------
DEPTNO                                 NUMBER(2)
DNAME                                  VARCHAR2(14)
LOC                                    VARCHAR2(13)
CREATE_DATE                            DATE

4 rows selected
```

ORACLE

Creación de Tablas

El ejemplo de la diapositiva crea la tabla `DEPT`, con cuatro columnas: `DEPTNO`, `DNAME`, `LOC` y `CREATE_DATE`. La columna `CREATE_DATE` tiene un valor por defecto. Si no se proporciona ningún valor para una sentencia `INSERT`, se inserta automáticamente la fecha del sistema.

Para confirmar que se ha creado la tabla, ejecute el comando `DESCRIBE`.

Puesto que la creación de una tabla constituye una sentencia DDL, se realiza una confirmación automática al ejecutar esta sentencia.

Nota: puede ver la lista de las tablas que posee consultando el diccionario de datos. Por ejemplo:

```
select table_name from user_tables
```

Mediante las vista de diccionario de datos, puede obtener información sobre otros objetos de base de datos como vistas, índices, etc. Los diccionarios de datos se explican en detalle en el curso *Oracle Database: Conceptos Fundamentales de SQL II*.

Agenda

- Objetos de base de datos
  - Reglas de nomenclatura
- Sentencia `CREATE TABLE`:
  - Acceso a otras tablas de usuario
  - Opción `DEFAULT`
- Tipos de dato
- Visión general de restricciones: Restricciones `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`
- Creación de una tabla mediante una subconsulta
- `ALTER TABLE`
  - Tablas de sólo lectura
- Sentencia `DROP TABLE`

ORACLE

Tipos de Dato

Tipo de Dato	Descripción
VARCHAR2(*size*)	Datos de caracteres de longitud variable
CHAR(*size*)	Datos de caracteres de longitud fija
NUMBER(*p*,*s*)	Datos numéricos de longitud variable
DATE	Valores de fecha y hora
LONG	Datos de caracteres de longitud variable (hasta 2 GB).
CLOB	Datos binarios (hasta 4 GB).
RAW and LONG RAW	Datos binarios raw
BLOB	Datos binarios (hasta 4 GB)
BFILE	Datos binarios almacenados en un archivo externo (hasta 4 GB)
ROWID	Sistema numérico de base -64 que representa la dirección única de una fila en su tabla correspondiente.

ORACLE

Tipos de Dato

Al identificar una columna para una tabla, debe proporcionar un tipo de dato para la columna. Hay varios tipos de dato disponibles:

Tipo de Dato	Descripción
VARCHAR2(*size*)	Datos de caracteres de longitud variable (Se debe especificar un tamaño máximo: el tamaño mínimo es 1 y el máximo es 4.000).
CHAR [(*size*)]	Datos de caracteres de longitud fija de tamaño de bytes (El tamaño por defecto y mínimo es 1; el tamaño máximo es 2.000).
NUMBER [(*p*,*s*)]	Número con la precisión *p* y la escala *s* (La precisión es el número total de cifras decimales y la escala el número de dígitos a la derecha del punto decimal; la precisión puede oscilar entre 1 y 38 y la escala entre –84 y 127).
DATE	Valores de fecha y hora hasta el segundo más cercano entre el 1 de enero, 4712 a. C., y el 31 de diciembre, 9999 d. C.
LONG	Datos de caracteres de longitud variable (hasta 2 GB).
CLOB	Datos de caracteres (hasta 4 GB).

Tipos de Dato (continuación)

Tipo de Dato	Descripción
RAW(*size*)	Datos binarios raw de tamaño de longitud (Se debe especificar un tamaño máximo: El tamaño máximo es 2.000.)
LONG RAW	Datos binarios raw de longitud variable (hasta 2 GB).
BLOB	Datos binarios (hasta 4 GB)
BFILE	Datos binarios almacenados en un archivo externo (hasta 4 GB).
ROWID	Sistema numérico de base 64 que representa la dirección única de una fila en su tabla correspondiente.

Instrucciones

- Las columnas LONG no se copian al crear una tabla mediante una subconsulta.
- Las columnas LONG no se pueden incluir en una cláusula GROUP BYu ORDER BY.
- Sólo se puede utilizar una columna LONG por tabla.
- No se pueden definir restricciones en las columnas LONG.
- Puede que desee utilizar una columna CLOB en lugar de una columna LONG.

Tipos de Dato de Fecha y Hora

Puede utilizar varios tipos de dato de fecha y hora:

Tipo de Dato	Descripción
TIMESTAMP	Fecha con segundos fraccionarios
INTERVAL YEAR TO MONTH	Almacenados como un intervalo de años y meses
INTERVAL DAY TO SECOND	Almacenado como un intervalo de días, horas, minutos y segundos

ORACLE

Tipos de Dato de Fecha y Hora

Tipo de Dato	Descripción
TIMESTAMP	Permite almacenar los datos de tiempo como fecha con segundos fraccionarios. Almacena los valores de año, mes, día, hora, y segundos del tipo de dato DATE, así como el valor para segundos fraccionarios. Hay distintas variantes de este tipo de dato, entre otras, WITH TIMEZONE, WITH LOCALTIMEZONE.
INTERVAL YEAR TO MONTH	Permite almacenar el tiempo como un intervalo de años y meses. Utilizado para representar la diferencia entre dos valores de fecha y hora en los que las únicas partes significativas son el año y el mes.
INTERVAL DAY TO SECOND	Permite almacenar el tiempo como un intervalo de días, horas, minutos y segundos. Utilizado para representar la diferencia exacta entre dos valores de fecha y hora.

Nota: estos tipos de dato de fecha y hora están disponibles con las versiones Oracle9*i* y posteriores. Los tipos de dato de fecha y hora se tratan más detalladamente en la lección sobre gestión de datos situados en distintas zonas horarias del curso *Oracle Database: Conceptos Fundamentales de SQL II.*

Además, para obtener más información sobre los tipos de dato de fecha y hora, consulte las secciones sobre tipo de de dato TIMESTAMP, tipo de dato INTERVAL YEAR TO MONTH y tipo de dato INTERVAL DAY TO SECOND en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Agenda

- Objetos de base de datos
  - Reglas de nomenclatura
- Sentencia `CREATE TABLE`:
  - Acceso a otras tablas de usuario
  - Opción `DEFAULT`
- Tipos de dato
- Visión general de restricciones: Restricciones `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`
- Creación de una Tabla mediante una Subconsulta
- `ALTER TABLE`
  - Tablas de sólo lectura
- Sentencia `DROP TABLE`

ORACLE

Inclusión de Restricciones

- Las restricciones aplican reglas a nivel de tabla.
- Las restricciones impiden la supresión de una tabla si hay dependencias.
- Los siguientes tipos de restricciones son válidos:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

Restricciones

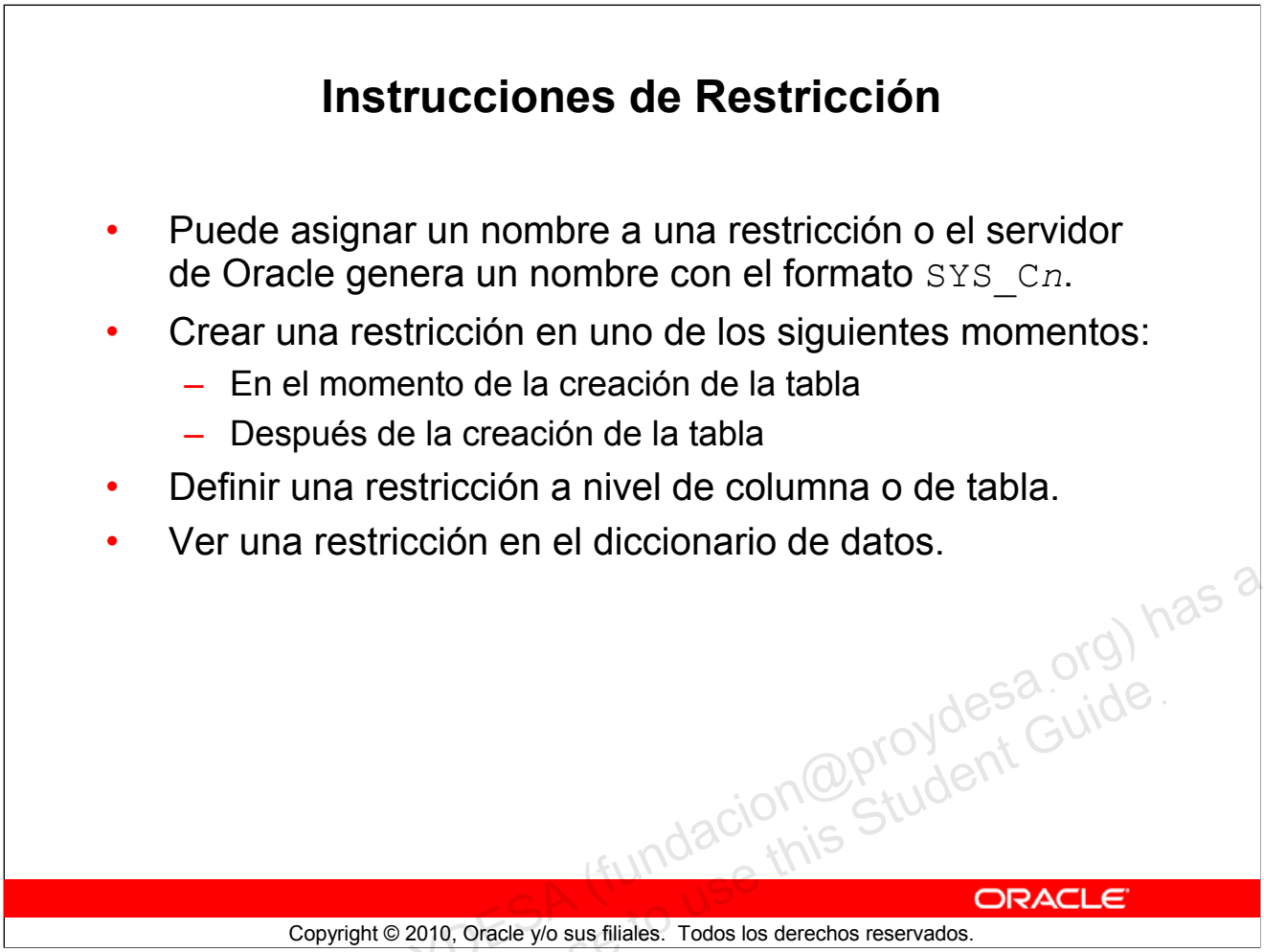
El servidor de Oracle utiliza restricciones para evitar la introducción de datos no válidos en las tablas.

Puede utilizar restricciones para realizar lo siguiente:

- Aplicar reglas a los datos de la tabla cuando se inserta, actualiza o suprime una fila de la misma. La restricción se debe cumplir para que la operación sea correcta.
- Evitar la supresión de una tabla si hay dependencias de otras tablas.
- Proporcionar reglas para las herramientas de Oracle, como Oracle Developer.

Restricciones de Integridad de Datos

Restricción	Descripción
NOT NULL	Especifica que la columna no puede contener un valor nulo.
UNIQUE	Especifica una columna o combinación de columnas cuyos valores deben ser únicos para todas las filas de la tabla.
PRIMARY KEY	Identifica de forma única cada fila de la tabla.
FOREIGN KEY	Establece y aplica la integridad referencial entre la columna y la columna de la tabla a la que se hace referencia; por ejemplo, busca los valores de una tabla que coinciden con los valores de la otra tabla.
CHECK	Especifica una condición que debe ser verdadera.



Instrucciones de Restricción

Todas las restricciones se almacenan en el diccionario de datos. Es fácil hacer referencia a las restricciones si les asigna un nombre significativo. Los nombres de las restricciones deben seguir las reglas de nomenclatura de objetos estándar, excepto que el nombre no puede ser el mismo que el de otro objeto propiedad del mismo usuario. Si no asigna ningún nombre a la restricción, el servidor de Oracle genera un nombre con el formato `SYS_Cn`, donde *n* es un entero para que el nombre de la restricción sea único.

Las restricciones se pueden definir en el momento de la creación de la tabla o después de haberla creado. Puede definir una restricción a nivel de columna o de tabla. Funcionalmente, una restricción a nivel de tabla es la mismo que una restricción a nivel de columna.

Para obtener más información, consulte la sección sobre funciones de restricciones en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Definición de Restricciones

- Sintaxis:

```
CREATE TABLE [schema.]table
      (column datatype [DEFAULT expr]
      [column_constraint],
      ...
      [table_constraint][,...]);
```

- Sintaxis de restricción a nivel de columna:

```
column [CONSTRAINT constraint_name] constraint_type,
```

- Sintaxis de restricción a nivel de tabla:

```
columna,...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

Definición de Restricciones

En la diapositiva se proporciona la sintaxis para la definición de restricciones al crear una tabla. Puede crear las restricciones a nivel de columna o de tabla. Las restricciones definidas a nivel de columna se incluyen al definir la columna. Las restricciones a nivel de tabla se definen al final de la definición de tabla y debe hacer referencia a la columna o las columnas a las que pertenece la restricción en un juego de paréntesis. Se trata principalmente de la sintaxis que diferencia a las dos; de lo contrario, funcionalmente, una restricción a nivel de columna es lo mismo que una restricción a nivel de tabla.

Las restricciones `NOT NULL` se deben definir a nivel de columna.

Las restricciones que se aplican a más de una columna se deben definir a nivel de tabla.

En la sintaxis:

`schema`	es el mismo nombre que el del propietario
`table`	es el nombre de la tabla
`DEFAULT expr`	especifica un valor por defecto para utilizarlo si se omite un valor en la sentencia `INSERT`
`column`	es el nombre de la columna
`datatype`	es el tipo de dato y la longitud de la columna
`column_constraint`	es una restricción de integridad como parte de la definición de columna
`table_constraint`	es una restricción de integridad como parte de la definición de tabla

Definición de Restricciones

- Ejemplo de una restricción a nivel de columna:

```
CREATE TABLE employees(
  employee_id  NUMBER(6)
    CONSTRAINT emp_emp_id_pk PRIMARY KEY,
  first_name    VARCHAR2(20),
  ...);
```

1

- Ejemplo de una restricción a nivel de tabla:

```
CREATE TABLE employees(
  employee_id  NUMBER(6),
  first_name   VARCHAR2(20),
  ...
  job_id       VARCHAR2(10) NOT NULL,
  CONSTRAINT emp_emp_id_pk
    PRIMARY KEY (EMPLOYEE_ID));
```

2

ORACLE

Definición de Restricciones (continuación)

Las restricciones se crean normalmente al mismo tiempo que la tabla. Las restricciones se pueden agregar a una tabla después de su creación y se pueden desactivar temporalmente.

Ambos ejemplos de la diapositiva crean una restricción de clave primaria en la columna `EMPLOYEE_ID` de la tabla `EMPLOYEES`.

1. En el primer ejemplo se utiliza la sintaxis de nivel de columna para definir la restricción.
2. En el segundo ejemplo se utiliza la sintaxis de nivel de tabla para definir la restricción.

Encontrará más información sobre la restricción de clave primaria más adelante en esta lección.

Restricción NOT NULL

Garantiza que no se permiten los valores nulos para la columna:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	24000	(null)	90	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	17000	(null)	90	NKOCHHAR	515.123.4568	21-SEP-89
102	Lex	De Haan	17000	(null)	90	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	9000	(null)	60	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	6000	(null)	60	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	4200	(null)	60	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	5800	(null)	50	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	3500	(null)	50	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	3100	(null)	50	CDAVIES	650.121.2994	29-JAN-97
143	Randall	Matos	2600	(null)	50	RMATOS	650.121.2874	15-MAR-98
144	Peter	Vargas	2500	(null)	50	PVARGAS	650.121.2004	09-JUL-98
149	Eleni	Zlotkey	10500	0.2	80	EZLOTKEY	011.44.1344.429018	29-JAN-00
174	Ellen	Abel	11000	0.3	80	EABEL	011.44.1644.429267	11-MAY-96
176	Jonathon	Taylor	8600	0.2	80	JTAYLOR	011.44.1644.429265	24-MAR-98
178	Kimberely	Grant	7000	0.15	(null)	KGRANT	011.44.1644.429263	24-MAY-99
200	Jennifer	Whalen	4400	(null)	10	JWHALEN	515.123.4444	17-SEP-87
201	Michael	Hartstein	13000	(null)	20	MHARTSTE	515.123.5555	17-FEB-96
202	Pat	Fay	6000	(null)	20	PFAY	603.123.6666	17-AUG-97
205	Shelley	Higgins	12000	(null)	110	SHIGGINS	515.123.8080	07-JUN-94
206	William	Gietz	8300	(null)	110	WGIETZ	515.123.8181	07-JUN-94

Restricción NOT NULL (La clave primaria aplica la restricción NOT NULL.)

NOT NULL restricción

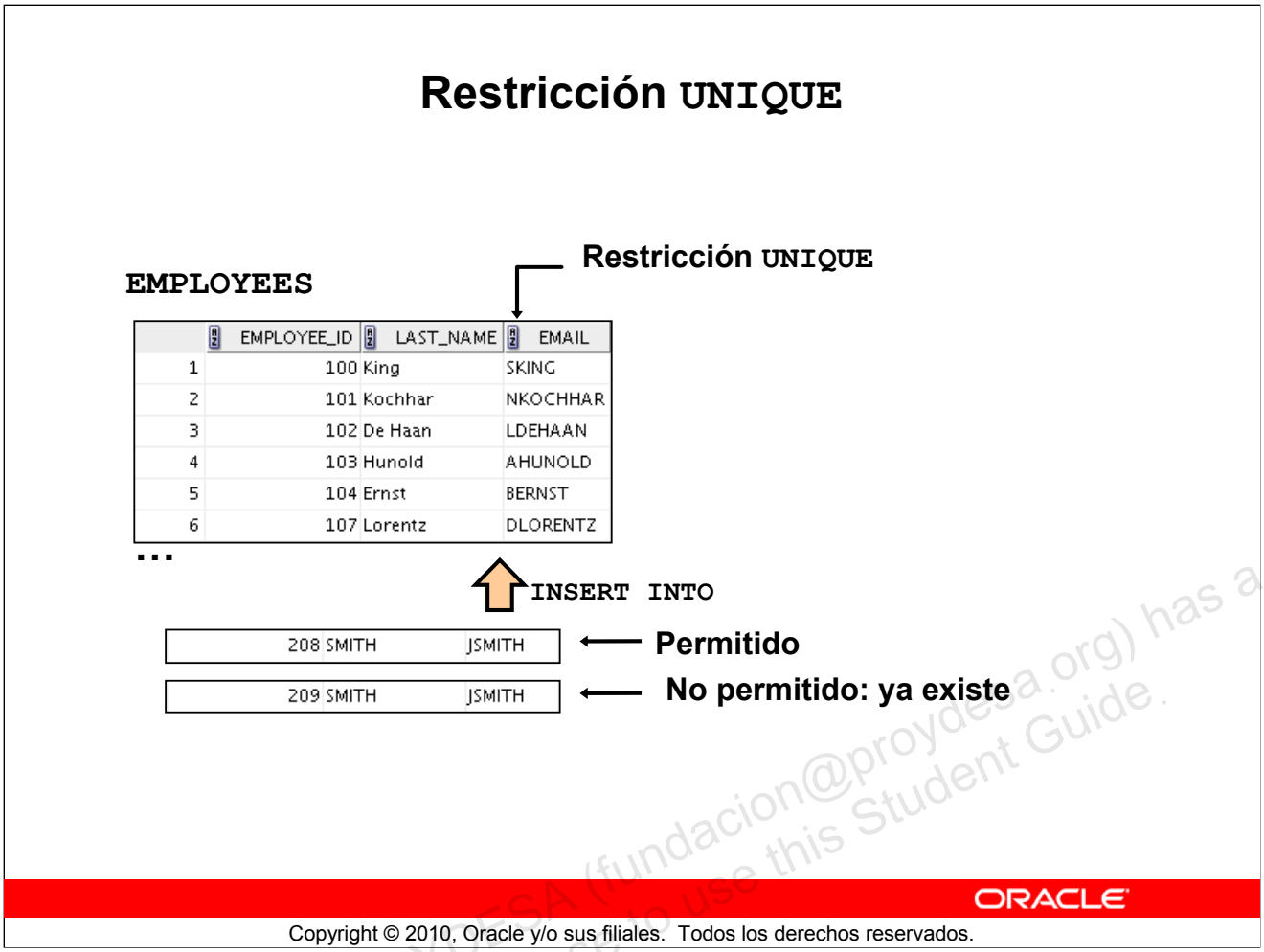
Ausencia de la restricción NOT NULL (Cualquier fila puede contener un valor nulo para esta columna.)

ORACLE

Restricción NOT NULL

La restricción `NOT NULL` garantiza que la columna no contiene ningún valor nulo. Las columnas sin la restricción `NOT NULL` pueden contener valores nulos por defecto. Las restricciones `NOT NULL` se deben definir a nivel de columna. En la tabla `EMPLOYEES`, la columna `EMPLOYEE_ID` hereda una restricción `NOT NULL` como se define en la clave primaria. De lo contrario, las columnas `LAST_NAME`, `EMAIL`, `HIRE_DATE` y `JOB_ID` tienen la restricción `NOT NULL` aplicada.

Nota: la restricción de clave primaria se trata con detalle más adelante en esta misma lección.



Restricción UNIQUE

Una restricción de integridad de clave UNIQUE necesita que todos los valores de una columna o juego de columnas (clave) sean únicos; es decir, que dos filas de una tabla no pueden tener valores duplicados en una columna o juego de columnas concreto. La columna (o juego de columnas) incluida en la definición de la restricción de clave UNIQUE se denomina *clave única.* Si la restricción UNIQUE comprende más de una columna, el grupo de columnas se denomina *clave única compuesta*.

Las restricciones UNIQUE permiten la entrada de valores nulos a menos que defina también las restricciones NOT NULL para las mismas columnas. De hecho, se puede incluir cualquier número de valores nulos para las columnas sin restricciones NOT NULL porque los valores nulos no se consideran iguales a ningún otro elemento. Un valor nulo en una columna (o en todas las columnas de una clave UNIQUE compuesta) cumple siempre una restricción UNIQUE.

Nota: debido al mecanismo de búsqueda de restricciones UNIQUE en una o más columnas, no puede tener valores idénticos en las columnas no nulas de una restricción de clave UNIQUE compuesta parcialmente nula.

Restricción UNIQUE

Definida a nivel de tabla o de columna:

```
CREATE TABLE employees(
    employee_id  NUMBER(6),
    last_name         VARCHAR2(25) NOT NULL,
    email             VARCHAR2(25),
    salary            NUMBER(8,2),
    commission_pct    NUMBER(2,2),
    hire_date         DATE NOT NULL,
...
    CONSTRAINT emp_email_uk UNIQUE(email));
```

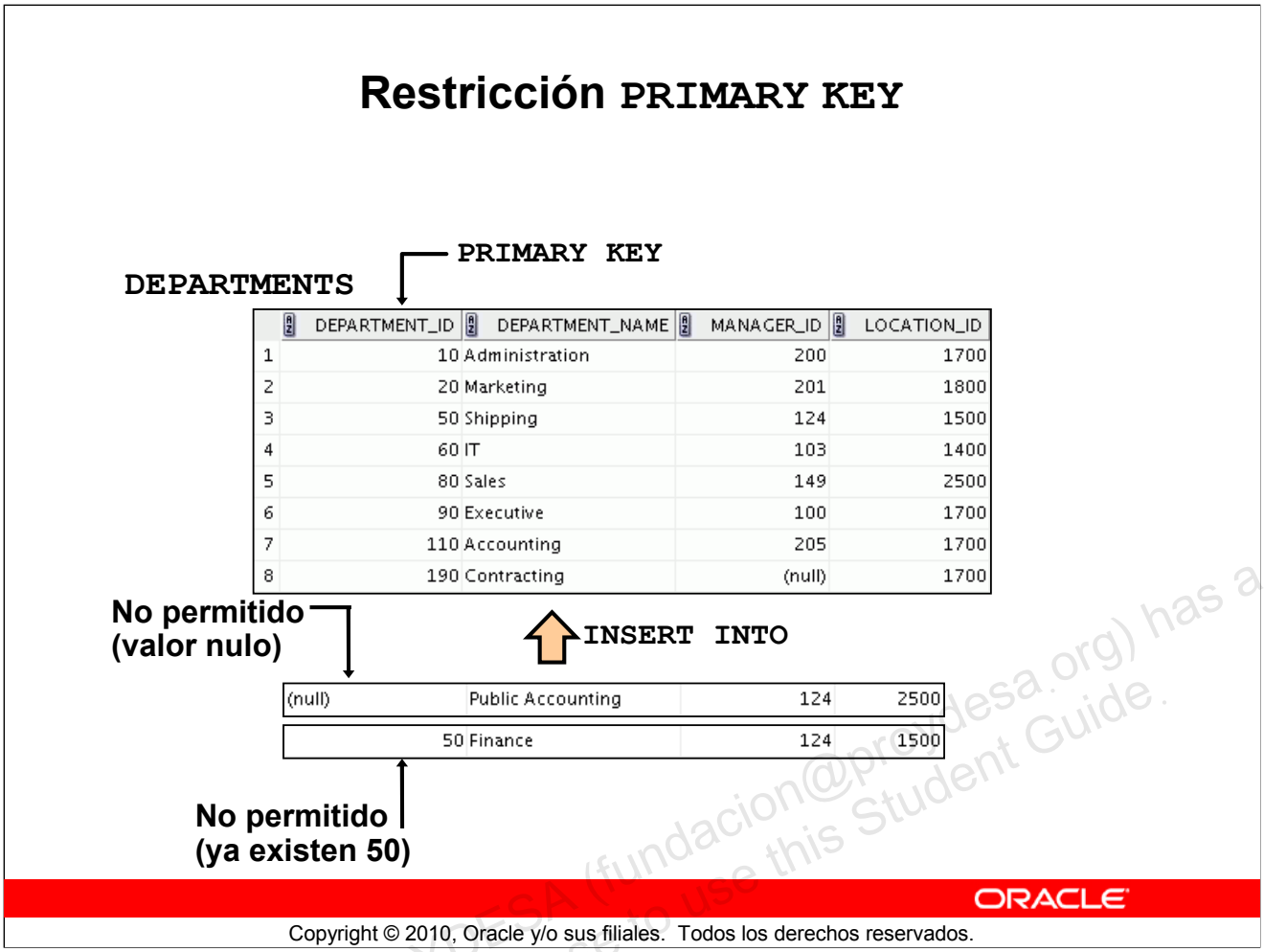
ORACLE

Restricción UNIQUE (continuación)

Las restricciones UNIQUE se pueden definir a nivel de tabla o de columna. Defina la restricción a nivel de tabla si desea crear una clave única compuesta. Una clave compuesta se define si no hay un único atributo que pueda identificar de forma única una fila. En ese caso, puede tener una clave única compuesta de dos o más columnas, el valor combinado de ambas es siempre único y puede identificar filas.

En el ejemplo de la diapositiva se aplica la restricción UNIQUE a la columna EMAIL de la tabla EMPLOYEES. El nombre de la restricción es EMP_EMAIL_UK.

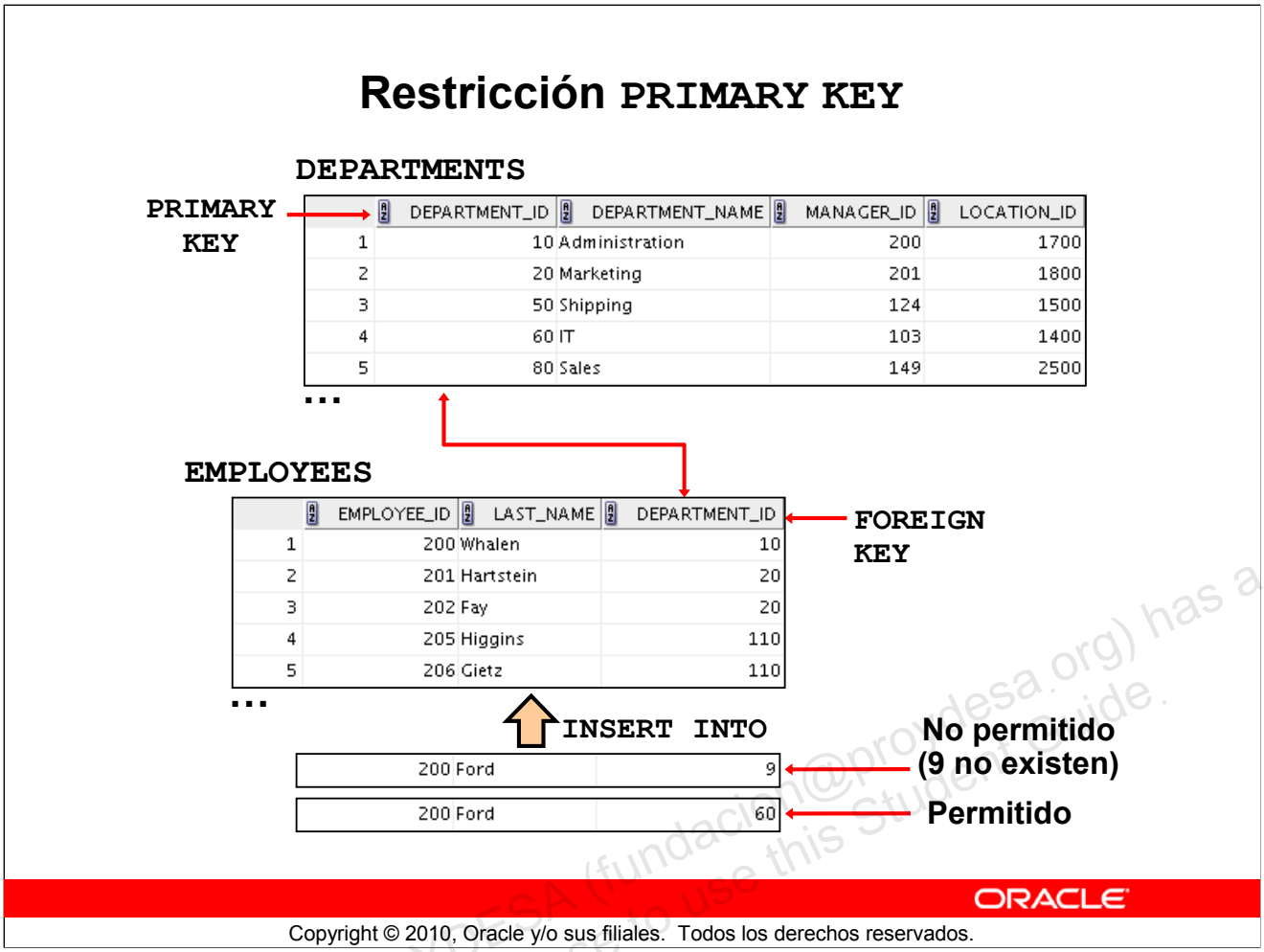
Nota: el servidor de Oracle aplica la restricción UNIQUE mediante la creación implícita de un índice único en la columna o columnas únicas.



Restricción PRIMARY KEY

La restricción PRIMARY KEY crea una clave primaria para la tabla. Sólo se puede crear una clave primaria para cada tabla. La restricción PRIMARY KEY es una columna o juego de columnas que identifica de forma única cada fila de tabla. Esta restricción aplica la unicidad de la columna o combinación de columnas y garantiza que ninguna columna que forme parte de la clave contenga un valor nulo.

Nota: puesto que la unicidad forma parte de la definición de restricción de clave primaria, el servidor de Oracle aplica la unicidad mediante la creación implícita de un índice único en la columna o columnas de clave primaria.



Restricción FOREIGN KEY

La restricción FOREIGN KEY (o integridad referencial) designa una columna o combinación de columnas como clave ajena y establece una relación entre una clave primaria o única en la misma tabla o en otra diferente.

En el ejemplo de la diapositiva, DEPARTMENT_ID se ha definido como clave ajena en la tabla EMPLOYEES (tabla dependiente o secundaria); hace referencia a la columna DEPARTMENT_ID de la tabla DEPARTMENTS (tabla a la que se hace referencia o principal).

Instrucciones

- El valor de clave ajena debe coincidir con un valor existente de la tabla principal o ser un valor NULL.
- Las claves ajenas se basan en los valores de datos y son punteros puramente lógicos, en lugar de físicos.

Restricción PRIMARY KEY

Definida a nivel de tabla o de columna:

```
CREATE TABLE employees(
    employee_id  NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
      REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

Restricción FOREIGN KEY (continuación)

Las restricciones FOREIGN KEY se pueden definir a nivel de restricción de columna o de tabla. Las claves ajenas compuestas se deben crear mediante la definición a nivel de tabla.

En el ejemplo de la diapositiva se define una restricción FOREIGN KEY en la columna DEPARTMENT_ID de la tabla EMPLOYEES, mediante la sintaxis a nivel de tabla. El nombre de la restricción es EMP_DEPT_FK.

La clave ajena se puede definir también a nivel de columna, siempre que la restricción está basada en una sola columna. La sintaxis difiere en que las palabras clave FOREIGN KEY no aparecen. Por ejemplo:

```
CREATE TABLE employees
(...
department_id NUMBER(4) CONSTRAINT emp_deptid_fk
REFERENCES departments(department_id),
...
)
```

Restricción FOREIGN KEY: Palabras Clave

- FOREIGN KEY: define la columna en la tabla secundaria a nivel de restricción de tabla
- REFERENCES: identifica la tabla y la columna en la tabla principal
- ON DELETE CASCADE: suprime las filas dependientes de la tabla secundaria cuando se suprime una fila de la tabla principal
- ON DELETE SET NULL: convierte los valores de clave ajena dependiente en nulos.

ORACLE

Restricción FOREIGN KEY: Palabras Clave

La clave ajena se define en la tabla secundaria y la tabla que contiene la columna a la que se hace referencia es la tabla principal. La clave ajena se define mediante una combinación de las siguientes palabras clave:

- FOREIGN KEY se utiliza para definir la columna en la tabla secundaria a nivel de restricción de tabla.
- REFERENCES identifica la tabla y la columna en la tabla principal.
- ON DELETE CASCADE indica que al suprimir una fila de la tabla principal, también se suprimen las filas dependientes de la tabla secundaria.
- ON DELETE SET NULL indica que al suprimir una fila de la tabla principal, los valores de clave ajena se definen como nulos.

El comportamiento por defecto se denomina *regla estricta*, que no permite la actualización o supresión de los datos a los que se hace referencia.

Sin las opciones ON DELETE CASCADE u ON DELETE SET NULL, la fila de la tabla principal no se puede suprimir si hace referencia a la misma en la tabla secundaria.

Restricción CHECK

- Define una condición que debe cumplir cada fila.
- Las siguientes expresiones no están permitidas:
  - Referencias a las pseudocolumnas CURRVAL, NEXTVAL, LEVEL y ROWNUM
  - Llamadas a las funciones SYSDATE, UID, USER y USERENV
  - Consultas que hagan referencia a otros valores en otras filas

```
..., salary   NUMBER(2)
     CONSTRAINT emp_salary_min
            CHECK (salary > 0),...
```


Restricción CHECK

La restricción CHECK define una condición que debe cumplir cada fila. La condición puede utilizar las mismas construcciones que las condiciones de consulta, con las siguientes excepciones:

- Referencias a las pseudocolumnas CURRVAL, NEXTVAL, LEVEL y ROWNUM
- Llamadas a las funciones SYSDATE, UID, USER y USERENV
- Consultas que hagan referencia a otros valores en otras filas

Una sola columna puede tener varias restricciones CHECK que hagan referencia a la columna en su definición. No hay ningún límite en cuanto al número de restricciones CHECK que puede definir en una columna.

Las restricciones CHECK se pueden definir a nivel de tabla o de columna.

```
CREATE TABLE employees
   (...
    salary NUMBER(8,2) CONSTRAINT emp_salary_min
                       CHECK (salary > 0),
   ...
```

Sentencia CREATE TABLE: Ejemplo

```
CREATE TABLE employees
    ( employee_id    NUMBER(6)
        CONSTRAINT     emp_employee_id   PRIMARY KEY
    , first_name     VARCHAR2(20)
    , last_name      VARCHAR2(25)
        CONSTRAINT     emp_last_name_nn  NOT NULL
    , email          VARCHAR2(25)
        CONSTRAINT     emp_email_nn      NOT NULL
        CONSTRAINT     emp_email_uk      UNIQUE
    , phone_number   VARCHAR2(20)
    , hire_date      DATE
        CONSTRAINT     emp_hire_date_nn  NOT NULL
    , job_id         VARCHAR2(10)
        CONSTRAINT     emp_job_nn        NOT NULL
    , salary         NUMBER(8,2)
        CONSTRAINT     emp_salary_ck     CHECK (salary>0)
    , commission_pct NUMBER(2,2)
    , manager_id     NUMBER(6)
         CONSTRAINT emp_manager_fk REFERENCES
          employees (employee_id)
    , department_id  NUMBER(4)
        CONSTRAINT     emp_dept_fk       REFERENCES
          departments (department_id));
```

ORACLE

Sentencia CREATE TABLE: Ejemplo

El ejemplo de la diapositiva muestra la sentencia que se utiliza para crear la tabla `EMPLOYEES` en el esquema `HR`.

Violación de Restricciones

El departamento 55 no existe.

```
UPDATE employees
SET    department_id = 55
WHERE  department_id = 110;
```

```
Error starting at line 1 in command:
UPDATE employees
SET    department_id = 55
WHERE  department_id = 110
Error report:
SQL Error: ORA-02291: integrity constraint (ORA1.EMP_DEPT_FK) violated - parent key not found
02291. 00000 - "integrity constraint (%s.%s) violated - parent key not found"
*Cause:    A foreign key value has no matching primary key value.
```

ORACLE

Violación de Restricciones

Si tiene restricciones en lugar de columnas, recibirá un error si intenta violar la regla de restricción. Por ejemplo, si intenta actualizar un registro con un valor ligado a una restricción de integridad, se devuelve un error.

En el ejemplo de la diapositiva, el departamento 55 no existe en la tabla principal, `DEPARTMENTS` y, por lo tanto, recibirá la violación de clave principal no encontrada `ORA-02291`.

Violación de Restricciones

No puede suprimir una fila que contenga una clave primaria que se utilice como clave ajena en otra tabla.

```
DELETE FROM departments
WHERE department_id = 60;
```

```
Error starting at line 1 in command:
DELETE FROM departments
WHERE department_id = 60
Error report:
SQL Error: ORA-02292: integrity constraint (ORA1.JHIST_DEPT_FK) violated - child record found
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"
*Cause:    attempted to delete a parent key value that had a foreign
           dependency.
*Action:   delete dependencies first then parent or disable constraint.
```


Violación de Restricciones (continuación)

Por ejemplo, si intenta suprimir un registro con un valor ligado a una restricción de integridad, se devuelve un error.

En el ejemplo de la diapositiva, se intenta suprimir el departamento 60 de la tabla `DEPARTMENTS`, pero se produce un error porque ese número de departamento se utiliza como clave ajena en la tabla `EMPLOYEES`. Si el registro principal que intenta suprimir tiene registros secundarios, recibirá la violación de registro secundario encontrado `ORA-02292`.

La siguiente sentencia funciona porque no hay ningún empleado en el departamento 70:

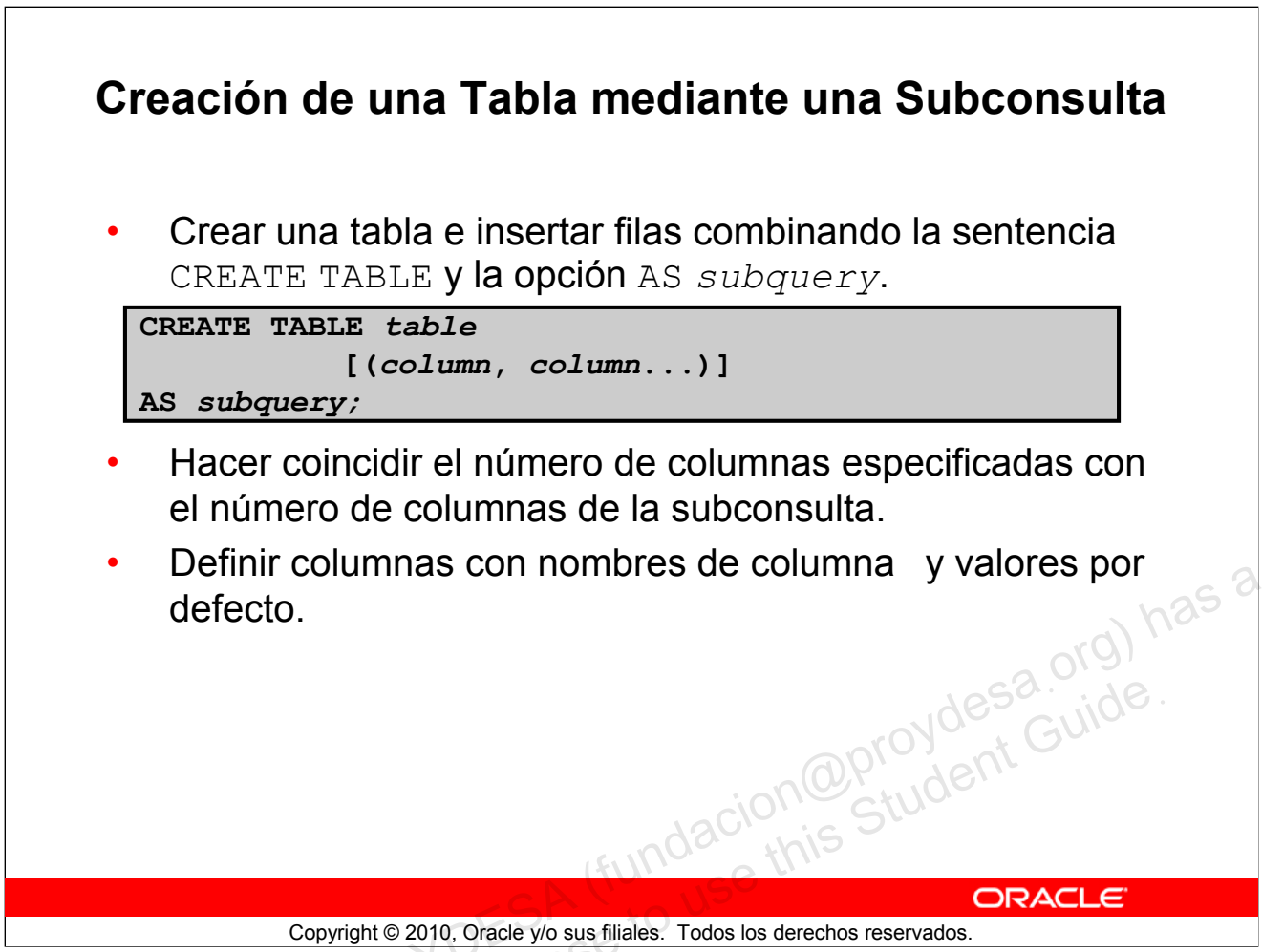
```
DELETE FROM  departments
WHERE department_id = 70;
```

```
1 rows deleted
```

Agenda

- Objetos de base de datos
  - Reglas de nomenclatura
- Sentencia `CREATE TABLE`:
  - Acceso a otras tablas de usuario
  - Opción `DEFAULT`
- Tipos de dato
- Visión general de restricciones: Restricciones `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`
- Creación de una Tabla mediante una Subconsulta
- `ALTER TABLE`
  - Tablas de sólo lectura
- Sentencia `DROP TABLE`

ORACLE



Creación de una Tabla mediante una Subconsulta

Un segundo método para crear tablas es aplicar la cláusula AS *subquery*, que crea la tabla e inserta las filas desvueltas de la subconsulta.

En la sintaxis:

table	es el nombre de la tabla
column	es el nombre de la columna, el valor por defecto y la restricción de integridad
subquery	es la sentencia SELECT que define el juego de filas que se va a insertar en la nueva tabla

Instrucciones

- La tabla se crea con los nombres de columna especificados y las filas recuperadas por la sentencia SELECT se insertan en la tabla.
- La definición de columna sólo puede contener el nombre de la columna y el valor por defecto.
- Si se proporcionan especificaciones de columna, el número de columnas debe ser igual al número de columnas de la lista SELECT de la subconsulta.
- Si no se proporcionan especificaciones de columna, los nombres de columna de la tabla son los mismos que los de la subconsulta.
- Las definiciones de tipo de dato de columna y de la restricción NOT NULL se transfieren a la nueva tabla. Tenga en cuenta que sólo la restricción explícita NOT NULL se heredará. La columna PRIMARY KEY no transferirá la función NOT NULL a la nueva columna. Las demás reglas de restricción no se transfieren a la nueva tabla. Sin embargo, puede agregar restricciones en la definición de columna.

Creación de una Tabla mediante una Subconsulta

```
CREATE TABLE  dept80
  AS
    SELECT  employee_id, last_name,
            salary*12 ANNSAL,
            hire_date
    FROM    employees
    WHERE   department_id = 80;
```

```
CREATE TABLE succeeded.
```

```
DESCRIBE dept80
```

Name	Null	Type
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE

ORACLE

Creación de una Tabla mediante una Subconsulta (continuación)

En el ejemplo de la diapositiva, se crea una tabla denominada `DEPT80`, que contiene los detalles de todos los empleados que trabajan en el departamento 80. Tenga en cuenta que los datos de la tabla `DEPT80` proceden de la tabla `EMPLOYEES`.

Puede verificar la existencia de una tabla de base de datos y comprobar las definiciones de columna mediante el comando `DESCRIBE`.

Sin embargo, asegúrese de proporcionar un alias de columna al seleccionar una expresión. La expresión `SALARY*12` recibe el alias `ANNSAL`. Sin el alias, se genera el siguiente error:

```
Error starting at line 1 in command:
CREATE TABLE  dept80
  AS     SELECT  employee_id, last_name,
            salary*12 ,
            hire_date    FROM    employees    WHERE   department_id = 80
Error at Command Line:3 Column:18
Error report:
SQL Error: ORA-00998: must name this expression with a column alias
00998. 00000 -  "must name this expression with a column alias"
*Cause:
*Action:
```

Agenda

- Objetos de base de datos
  - Reglas de nomenclatura
- Sentencia `CREATE TABLE`:
  - Acceso a otras tablas de usuario
  - Opción `DEFAULT`
- Tipos de dato
- Visión general de restricciones: Restricciones `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`
- Creación de una tabla mediante una subconsulta
- `ALTER TABLE`
  - Tablas de sólo lectura
- Sentencia `DROP TABLE`

ORACLE

Sentencia ALTER TABLE

Utilizar la sentencia ALTER TABLE para:

- Agregar una nueva columna
- Modificar una definición de columna existente
- Definir un valor por defecto para la nueva columna
- Borrar una columna
- Cambiar el nombre de una columna
- Cambiar la tabla al estado de sólo lectura

ORACLE

Sentencia ALTER TABLE

Después de crear una tabla, puede que necesite cambiar la estructura de la tabla por cualquiera de las siguientes razones:

- Ha omitido una columna.
- Debe cambiar la definición de columna o su nombre.
- Debe eliminar columnas.
- Desea definir la tabla en modo de sólo lectura.

Para ello, puede utilizar la sentencia ALTER TABLE.

Tablas de Sólo Lectura

Puede utilizar la sentencia ALTER TABLE para:

- Definir una tabla en modo de sólo lectura, que evita que DDL o DML cambie durante el mantenimiento de la tabla
- Volver a definir la tabla en modo de lectura/escritura

```
ALTER TABLE employees READ ONLY;

-- perform table maintenance and then
-- return table back to read/write mode

ALTER TABLE employees READ WRITE;
```


ORACLE

Tablas de Sólo Lectura

Con Oracle Database 11g, puede especificar READ ONLY para definir una tabla en modo de sólo lectura. Cuando la tabla esté en modo READ-ONLY, no puede ejecutar ninguna sentencia DML que afecte a la tabla o a cualquier sentencia SELECT ... FOR UPDATE. Puede ejecutar las sentencias DDL siempre y cuando no modifique los datos de la tabla. Se permiten operaciones sobre índices asociados a la tabla cuando la tabla esté en modo READ ONLY.

Especifique READ/WRITE para volver a definir una tabla de sólo lectura en modo de lectura/escritura.

Nota: si es necesario puede borrar una tabla con modo READ ONLY. El comando DROP se ejecuta sólo en el diccionario de datos, por lo que no es necesario el acceso al contenido de la tabla. El espacio utilizado por la tabla no se reclamará hasta que el tablespace se vuelva a definir en lectura/escritura y, a continuación, los cambios necesarios se podrán realizar en las cabeceras de segmento de bloque, etc.

Para obtener más información sobre la sentencia ALTER TABLE, consulte el curso titulado *Oracle Database: Conceptos Fundamentales de SQL II*.

Agenda

- Objetos de base de datos
  - Reglas de nomenclatura
- Sentencia `CREATE TABLE`:
  - Acceso a otras tablas de usuario
  - Opción `DEFAULT`
- Tipos de dato
- Visión general de restricciones: Restricciones `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK`
- Creación de una tabla mediante una subconsulta
- `ALTER TABLE`
  - Tablas de sólo lectura
- Sentencia `DROP TABLE`

ORACLE

Borrado de una Tabla

- Mueve una tabla a la papelera de reciclaje
- Elimina la tabla y todos sus datos completamente si se especifica la cláusula `PURGE`
- Invalida objetos dependientes y elimina privilegios de objeto en la tabla

```
DROP TABLE dept80;
DROP TABLE dept80 succeeded.
```


Borrado de una Tabla

La sentencia `DROP TABLE` mueve una tabla a la papelera de reciclaje o elimina la tabla y todos sus datos de la base de datos completamente. A menos que especifique la cláusula `PURGE`, la sentencia `DROP TABLE` no vuelve a liberar espacio en los tablespaces para que lo utilicen otros objetos y el espacio sigue contando en la cuota de espacio del usuario. El borrado de una tabla invalida objetos dependientes y elimina privilegios de objeto en la tabla.

Al borrar una tabla, la base de datos pierde todos los datos de la tabla y los índices asociados a los mismos.

Sintaxis

```
DROP TABLE table [PURGE]
```

En la sintaxis, *`table`* es el nombre de la tabla.

Instrucciones

- Se suprimen todos los datos de la tabla.
- Se mantienen las vistas y los sinónimos, pero no son válidos.
- Se confirman las transacciones pendientes.
- Sólo el creador de la tabla o un usuario con el privilegio `DROP ANY TABLE` puede eliminar una tabla.

Nota: utilice la sentencia `FLASHBACK TABLE` para restaurar una tabla borrada de la papelera de reciclaje. Esto se trata con más detalle en el curso titulado *Oracle Database: Conceptos Fundamentales de SQL II*.

Prueba

Puede utilizar restricciones para realizar lo siguiente:

1. Aplicar reglas a los datos de la tabla cuando se inserta, actualiza o suprime una fila de dicha tabla.
2. Evitar la supresión de una tabla.
3. Evitar la creación de una tabla.
4. Evitar la creación de datos en una tabla.

ORACLE

Respuesta: 1, 2, 4

Resumen

En esta lección, debe haber aprendido a utilizar la sentencia `CREATE TABLE` para crear una tabla e incluir restricciones.

- Clasificación de los Principales Objetos de Base de Datos
- Revisar la estructura de la tabla
- Mostrar los tipos de dato disponibles para columnas
- Crear una tabla simple
- Explicar cómo crear restricciones en el momento de la creación de la tabla
- Describir el funcionamiento de los objetos de esquema

Resumen

En esta lección debe haber aprendido a realizar lo siguiente:

CREATE TABLE

- Utilizar la sentencia `CREATE TABLE` para crear una tabla e incluir restricciones.
- Crear una tabla basada en otra tabla mediante una subconsulta.

DROP TABLE

- Eliminar filas y la estructura de una tabla.
- Una vez ejecutada, no se puede realizar un rollback de esta sentencia.

Práctica 10: Visión General

En esta práctica se abordan los siguientes temas:

- Creación de nuevas tablas
- Creación de una nueva tabla mediante la sintaxis `CREATE TABLE AS`
- Verificación de la existencia de las tablas
- Definición de una tabla al estado de sólo lectura
- Borrado de tablas

ORACLE

Práctica 10: Visión General

Cree nuevas tablas mediante la sentencia `CREATE TABLE`. Confirme que la nueva tabla se ha agregado a la base de datos. También aprenderá a definir el estado de una tabla como `READ ONLY` y, a continuación, revertir a `READ/WRITE`.

Nota: para todas las sentencias `DDL` y `DML`, haga clic en el icono Run Script (o pulse [F5]) para ejecutar la consulta en SQL Developer. De esta forma, ve los mensajes de comentarios en la página con separadores Script Output. Para consultas SELECT, siga haciendo clic en el icono Execute Statement o pulse [F9] para obtener la salida con formato en la página con separadores Results.

11

Creación de Otros Objetos de Esquema

ORACLE

Objetivos

Al finalizar esta lección, debería estar capacitado para lo siguiente:

- Crear vistas simples y complejas
- Recuperar datos de las vistas
- Crear, mantener y utilizar secuencias
- Crear y mantener índices
- Crear sinónimos privados y públicos

ORACLE

Objetivos

En esta lección, se ofrece una introducción a los objetos de vista, secuencia, sinónimo e índice. Aprenderá los conceptos básicos de la creación y el uso de vistas, secuencias e índices.

Agenda

- Visión general de la vistas:
  - Creación, modificación y recuperación de datos de una vista
  - Realización de operaciones de lenguaje de manipulación de datos (DML) en una vista
  - Borrado de una vista
- Visión general de secuencias:
  - Creación, uso y modificación de una secuencia
  - Valores de secuencia de caché
  - Pseudocolumnas `NEXTVAL` y `CURRVAL`
- Visión general de índices
  - Creación y borrado de índices
- Visión general de sinónimos
  - Creación y borrado de sinónimos

ORACLE

Objetos de Base de Datos

Objeto	Descripción
Tabla	Unidad básica de almacenamiento; compuesta por filas.
Vista	Representa de forma lógica subjuegos de datos de una o más tablas.
Secuencia	Genera valores numéricos.
Índice	Mejora el rendimiento de las consultas de recuperación de datos.
Sinónimo	Ofrece nombres alternativos para los objetos.

Objetos de Base de Datos

Existen otros objetos en la base de datos además de las tablas.

Con las vistas, puede presentar y ocultar datos de las tablas.

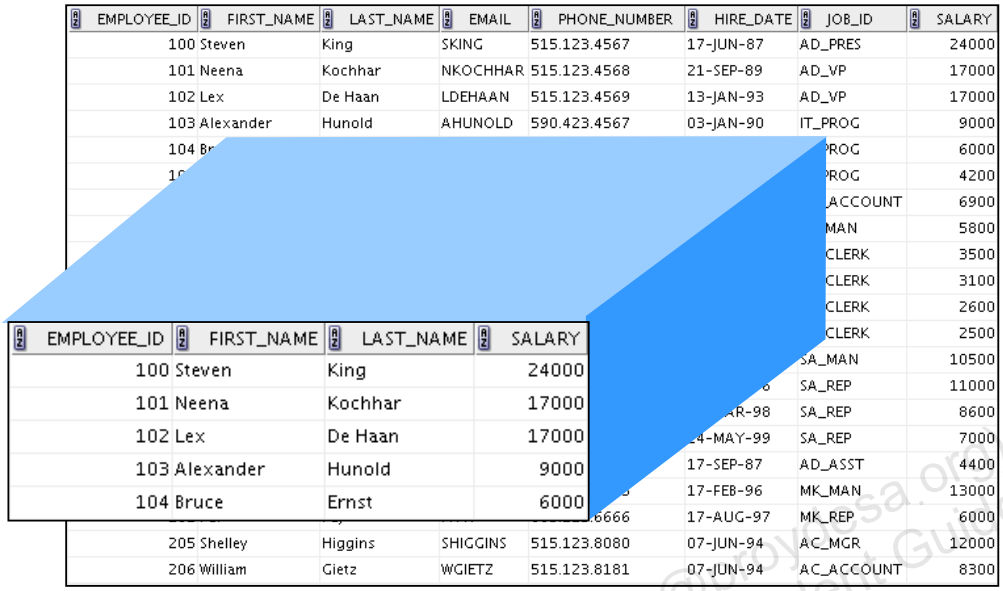
Muchas aplicaciones exigen el uso de números únicos como valores de clave primaria. Puede crear código en la aplicación para cumplir este requisito o utilizar una secuencia para generar números únicos.

Si desea mejorar el rendimiento de las consultas de recuperación de datos, debe crear un índice. También puede utilizar índices para reforzar la unicidad en una columna o recopilación de columnas.

Puede proporcionar nombres alternativos para los objetos mediante sinónimos.

¿Qué es una Vista?

Tabla **EMPLOYEES**



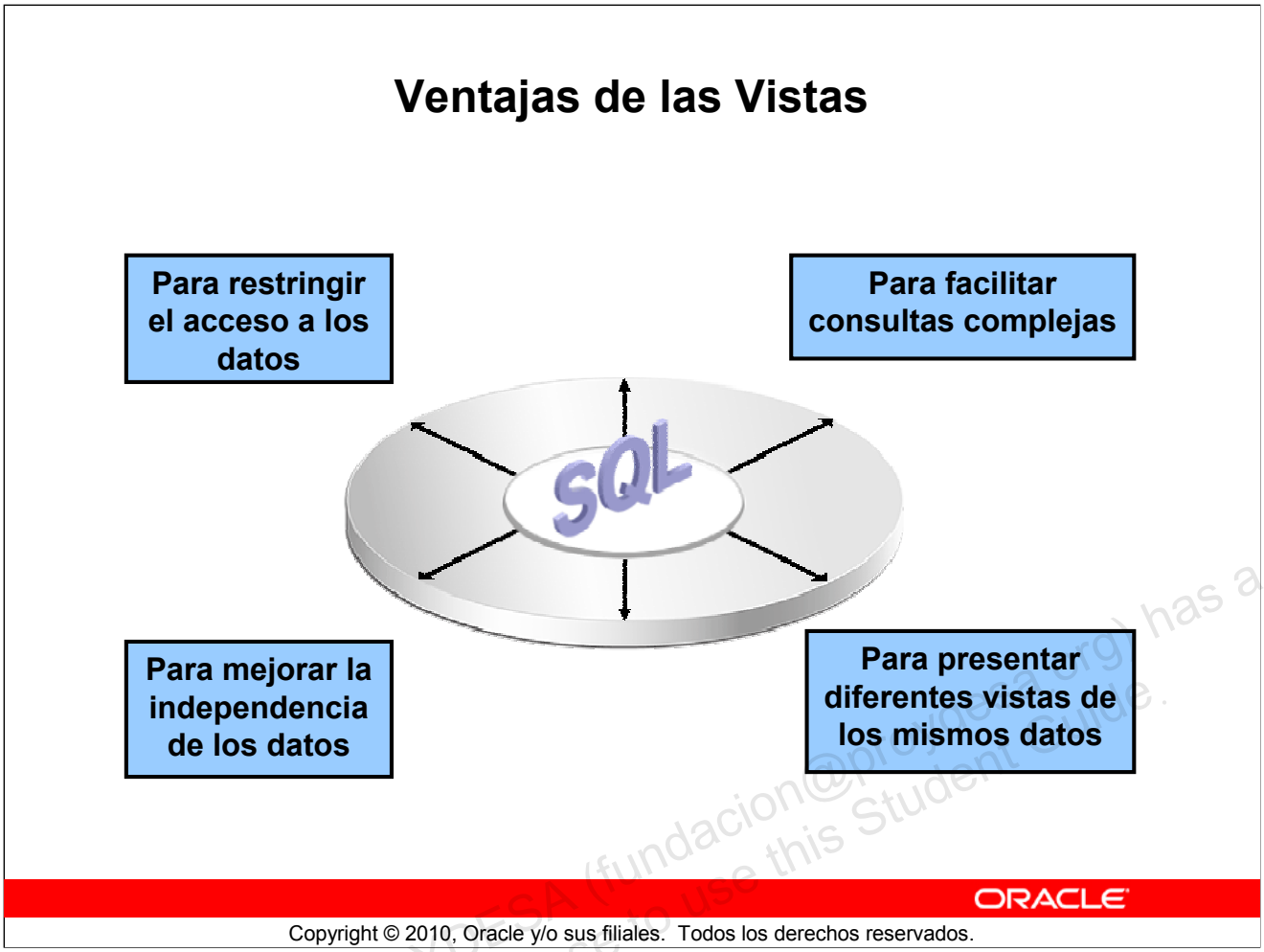
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
104	Br					ROG	6000
1						ROG	4200
						ACCOUNT	6900
						MAN	5800
						CLERK	3500
						CLERK	3100
						CLERK	2600
						CLERK	2500
						SA_MAN	10500
					6	SA_REP	11000
					AR-98	SA_REP	8600
					4-MAY-99	SA_REP	7000
					17-SEP-87	AD_ASST	4400
					17-FEB-96	MK_MAN	13000
				6666	17-AUG-97	MK_REP	6000
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
100	Steven	King	24000
101	Neena	Kochhar	17000
102	Lex	De Haan	17000
103	Alexander	Hunold	9000
104	Bruce	Ernst	6000

ORACLE

¿Qué es una Vista?

Puede representar combinaciones de datos o subjuegos lógicos mediante la creación de vistas de tablas. Una vista es una tabla lógica basada en una tabla o en otra vista. Una vista no contiene ningún dato propio, sino que se trata de una ventana a través de la que se pueden visualizar o cambiar los datos de las tablas. Las tablas en las que se basa la vista se denominan *tablas base*. Las vista se almacena como una sentencia `SELECT` en el diccionario de datos.



Ventajas de las Vistas

- Las vistas restringen el acceso a los datos porque muestra las columnas seleccionadas de la tabla.
- Las vistas se pueden utilizar para crear consultas simples para recuperar los resultados de consultas complicadas. Por ejemplo, las vistas se pueden utilizar para consultar información de varias tablas sin que el usuario sepa cómo escribir una sentencia de unión.
- Las vistas proporcionan independencia de los datos para usuarios ad-hoc y programas de aplicación. Las vistas se pueden utilizar para recuperar datos de varias tablas.
- Las vistas proporcionan a los grupos de usuarios acceso a los datos según sus criterios concretos.

Para obtener más información, consulte la sección sobre `CREATE VIEW` en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Vistas Simples y Complejas

Función	Vistas Simples	Vistas Complejas
Número de tablas	Una	Una o más
Contienen funciones	No	Sí
Contienen grupos de datos	No	Sí
Operaciones DML a través de una vista	Sí	No siempre

ORACLE

Vistas Simples y Complejas

Existen dos clasificaciones para las vistas: simples y complejas.. La principal diferencia está relacionada con las operaciones (`INSERT`, `UPDATE` y `DELETE`).

- Las vistas simples:
  - Derivan datos de una sola tabla.
  - No contienen funciones ni grupos de datos.
  - Pueden realizar operaciones DML a través de la vista.
- Las vistas complejas:
  - Derivan datos de varias tablas.
  - Contienen funciones o grupos de datos.
  - No siempre permiten operaciones DML a través de la vista.

Creación de Vistas

- Embeber una subconsulta en la sentencia `CREATE VIEW`:

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
  [(alias[, alias]...)]
 AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

- La subconsulta puede contener la sintaxis compleja `SELECT`.

ORACLE

Creación de Vistas

Para crear una vista, puede embeber una subconsulta en la sentencia `CREATE VIEW`.

En la sintaxis:

`OR REPLACE`	vuelve a crear la vista si ya existe.
`FORCE`	crea la vista independientemente de si existen o no las tablas base.
`NOFORCE`	crea la vista sólo si existen las tablas base (opción por defecto).
`view`	es el nombre de la vista.
`alias`	especifica los nombre de las expresiones seleccionadas por la consulta de la vista. (El numero de alias debe coincidir con el número de expresiones seleccionadas por la vista).
`subquery`	Es una sentencia `SELECT` completa (Puede utilizar alias para las columnas de la lista `SELECT`).
`WITH CHECK OPTION`	especifica que sólo las filas accesibles para la vista se puedan insertar o actualizar.
`constraint`	es el nombre asignado a la restricción `CHECK OPTION`.
`WITH READ ONLY`	garantiza que no se pueda realizar ninguna operación DML en esta vista.

Nota: en SQL Developer, haga clic en el icono Run Script o pulse [F5] para ejecutar las sentencias de lenguaje de definición de datos (DLL). Los comentarios se mostrarán en la página con separadores Script Output.

Creación de Vistas

- Crear la vista `EMPVU80`, que contiene los detalles de los empleados del departamento 80:

```
CREATE VIEW   empvu80
 AS SELECT  employee_id, last_name, salary
    FROM    employees
    WHERE   department_id = 80;
CREATE VIEW succeeded.
```

- Describir la estructura de la vista mediante el comando `DESCRIBE` de SQL*Plus:

```
DESCRIBE empvu80
```


Creación de Vistas (continuación)

En el ejemplo de la diapositiva se crea una vista que contiene el número de empleado, el apellido y el salario de cada empleado del departamento 80.

Puede visualizar la estructura de la vista mediante el comando `DESCRIBE`.

```
Name                          Null     Type
----------------------------- -------- ------------
EMPLOYEE_ID                   NOT NULL NUMBER(6)
LAST_NAME                     NOT NULL VARCHAR2(25)
SALARY                                 NUMBER(8,2)
```

Instrucciones

- La subconsulta que define una vista puede contener la sintaxis compleja `SELECT`, incluidas uniones, grupos y subconsultas.
- Si no especifica un nombre de restricción para la vista creada con `WITH CHECK OPTION`, el sistema asigna un nombre por defecto con el formato `SYS_Cn`.
- Puede utilizar la opción `OR REPLACE` para cambiar la definición de la vista sin borrarla y volver a crearla o sin necesidad de volver a otorgarle los privilegios de objeto otorgados previamente.

Creación de Vistas

- Crear una vista mediante alias de columna en la subconsulta:

```
CREATE VIEW   salvu50
 AS SELECT   employee_id ID_NUMBER, last_name NAME,
             salary*12 ANN_SALARY
    FROM     employees
    WHERE    department_id = 50;
CREATE VIEW succeeded.
```

- Seleccionar las columnas de esta vista según los nombres de alias proporcionados.

ORACLE

Creación de Vistas (continuación)

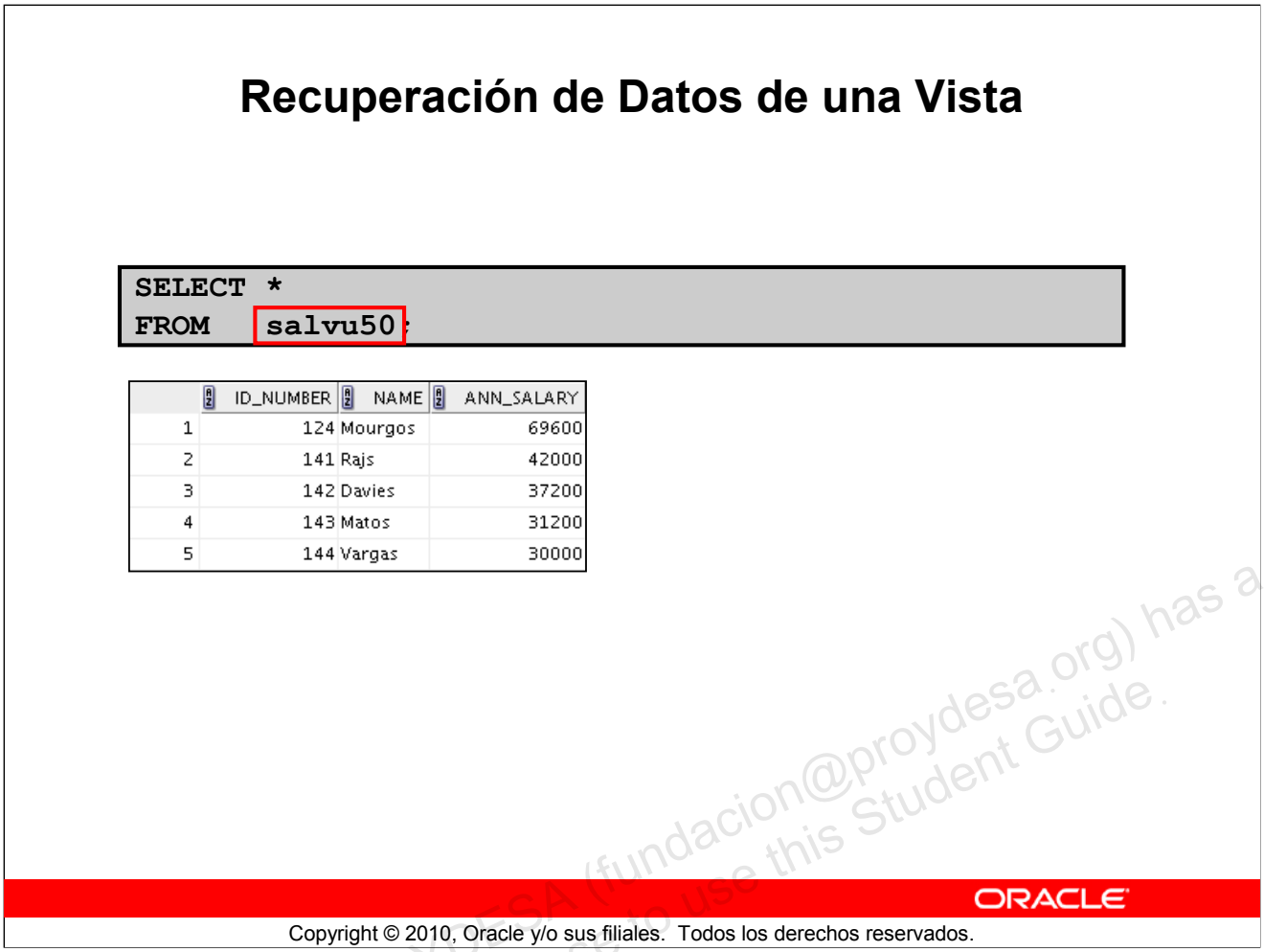
Puede controlar los nombres de columna incluyendo alias de columna en la subconsulta.

El ejemplo de la diapositiva crea una vista que contiene el número de empleado (`EMPLOYEE_ID`) con el alias `ID_NUMBER`, el nombre (`LAST_NAME`) con el alias `NAME` y el salario anual (`SALARY`) con el alias `ANN_SALARY` para cada empleado del departamento 50.

Como alternativa, puede utilizar un alias después de la sentencia `CREATE` y antes de la subconsulta `SELECT`. El numero de alias mostrado debe coincidir con el número de expresiones seleccionadas en la vista).

```
CREATE OR REPLACE VIEW   salvu50 (ID_NUMBER, NAME, ANN_SALARY)
  AS SELECT  employee_id, last_name, salary*12
     FROM    employees
     WHERE   department_id = 50;

CREATE VIEW succeeded.
```



Recuperación de Datos de una Vista

Puede recuperar datos de una vista al igual que de una tabla. Puede visualizar el contenido de toda la vista o sólo de filas y columnas específicas.

Modificación de Vistas

- Modificar la vista `EMPVU80` mediante una cláusula `CREATE OR REPLACE VIEW`. Agregar alias para cada nombre de columna:

```
CREATE OR REPLACE VIEW empvu80
  (id_number, name, sal, department_id)
AS SELECT  employee_id, first_name || ' '
           || last_name, salary, department_id
   FROM    employees
   WHERE   department_id = 80;
CREATE OR REPLACE VIEW succeeded.
```

- Los alias de columna de la cláusula `CREATE OR REPLACE VIEW` se muestran en el mismo orden que las columnas de la subconsulta.

ORACLE

Modificación de Vistas

Con la opción `OR REPLACE`, se puede crear una vista incluso si ya existe otra con este nombre, sustituyendo de esta forma la antigua versión de la vista para su propietario. Esto significa que la vista se puede modificar sin borrar, volver a crear y volver a otorgar los privilegios de objeto.

Nota: al asignar alias de columna en la cláusula `CREATE OR REPLACE VIEW`, recuerde que los alias se muestran en el mismo orden que las columnas de la subconsulta.

Creación de Vistas Complejas

Crear una vista compleja que contenga funciones de grupo para visualizar valores de dos tablas:

```
CREATE OR REPLACE VIEW dept_sum_vu
  (name, minsal, maxsal, avgsal)
AS SELECT   d.department_name, MIN(e.salary),
           MAX(e.salary),AVG(e.salary)
   FROM     employees e JOIN departments d
   ON       (e.department_id = d.department_id)
   GROUP BY d.department_name;
CREATE OR REPLACE VIEW succeeded.
```


Creación de Vistas Complejas

En el ejemplo de la diapositiva se crea una vista compleja de nombres de departamento, salarios mínimos y salarios medios por departamento. Tenga en cuenta que se han especificado nombres alternativos para la vista. Éste es un requisito necesario si alguna de las columnas de la vista se deriva de una función o expresión.

Puede visualizar la estructura de la vista mediante el comando `DESCRIBE`. Para visualizar el contenido de la vista, emita una sentencia `SELECT`.

```
SELECT *
FROM    dept_sum_vu;
```

	NAME	MINSAL	MAXSAL	AVGSAL
1	Administration	4400	4400	4400
2	Accounting	8300	12000	10150
3	IT	4200	9000	6400
4	Executive	17000	24000	19333.333333333...
5	Shipping	2500	5800	3500
6	Sales	8600	11000	10033.333333333...
7	Marketing	6000	13000	9500

Reglas para Realizar Operaciones DML en una Vista

- Normalmente, puede realizar operaciones DML en las vistas simples.
- No puede eliminar una fila si la vista contiene lo siguiente:
  - Funciones de grupo
  - Una cláusula `GROUP BY`
  - La palabra clave `DISTINCT`
  - La palabra clave `ROWNUM` de pseudocolumna

Reglas para Realizar Operaciones DML en una Vista

- Puede realizar operaciones DML en los datos a través de una vista si dichas operaciones siguen ciertas reglas.
- Puede eliminar una fila de una vista a menos que contenga alguna de las siguientes opciones:
  - Funciones de grupo
  - Una cláusula `GROUP BY`
  - La palabra clave `DISTINCT`
  - La palabra clave `ROWNUM` de pseudocolumna

Reglas para Realizar Operaciones DML en una Vista

No puede modificar datos de una vista si contiene:

- Funciones de grupo
- Una cláusula `GROUP BY`
- La palabra clave `DISTINCT`
- La palabra clave `ROWNUM` de pseudocolumna
- Columnas definidas por expresiones

ORACLE

Reglas para Realizar Operaciones DML en una Vista (continuación)

Puede modificar datos a través de una vista a menos que contenga cualquiera de las condiciones mencionadas en la diapositiva anterior o columnas definidas por expresiones (por ejemplo, `SALARY * 12`).

Reglas para Realizar Operaciones DML en una Vista

No puede agregar datos a través de una vista si ésta incluye:

- Funciones de grupo
- Una cláusula `GROUP BY`
- La palabra clave `DISTINCT`
- La palabra clave `ROWNUM` de pseudocolumna
- Columnas definidas por expresiones
- Columnas `NOT NULL` de las tablas base no seleccionadas por la vista

ORACLE

Reglas para Realizar Operaciones DML en una Vista (continuación)

Puede agregar datos a través de una vista a menos que contenga cualquiera de los elementos mostrados en la diapositiva. No puede agregar datos a una vista si ésta contiene columnas `NOT NULL` sin valores por defecto en la tabla base. Todos los valores necesarios deben estar presentes en la vista. Recuerde que va a agregar valores directamente a la tabla subyacente *a través* de la vista.

Para obtener más información, consulte la sección sobre `CREATE VIEW` en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Uso de la Cláusula WITH CHECK OPTION

- Puede asegurarse de que las operaciones DML realizadas en la vista permanecerán en el dominio de la vista mediante la cláusula WITH CHECK OPTION:

```
CREATE OR REPLACE VIEW empvu20
AS SELECT    *
   FROM      employees
   WHERE     department_id = 20
   WITH CHECK OPTION CONSTRAINT empvu20_ck ;
CREATE OR REPLACE VIEW succeeded.
```

- Cualquier intento de ejecutar INSERT con una fila con department_id distinto de 20, o ejecutar UPDATE en el número de departamento para cualquier fila de la vista fallará porque viola la restricción WITH CHECK OPTION.

ORACLE

Uso de la Cláusula WITH CHECK OPTION

Es posible realizar comprobaciones de integridad referencial a través de las vistas. También puede aplicar restricciones a nivel de base de datos. La vista se puede utilizar para proteger la integridad de los datos, pero su uso es muy limitado.

La cláusula WITH CHECK OPTION especifica que las operaciones INSERT y UPDATE realizadas a través de la vista no pueden crear filas que no pueda seleccionar la vista. Por lo tanto, permite aplicar restricciones de integridad y comprobaciones de validación de datos en los datos que se van a insertar o actualizar. Si se intenta realizar operaciones DML en filas que no haya seleccionado la vista, se muestra un error, junto con el nombre de la restricción si se ha especificado.

```
UPDATE empvu20
SET    department_id = 10
WHERE  employee_id = 201;
```

causa:

```
Error report:
SQL Error: ORA-01402: view WITH CHECK OPTION where-clause violation
01402. 00000 -  "view WITH CHECK OPTION where-clause violation"
```

Nota: no se ha actualizado ninguna fila porque si el número de departamento se cambiara a 10, la vista ya no podría ver ese empleado. Con la cláusula WITH CHECK OPTION, por tanto, la vista sólo puede ver los empleados del departamento 20 y no permite cambiar el número de departamento de dichos empleados a través de la vista.

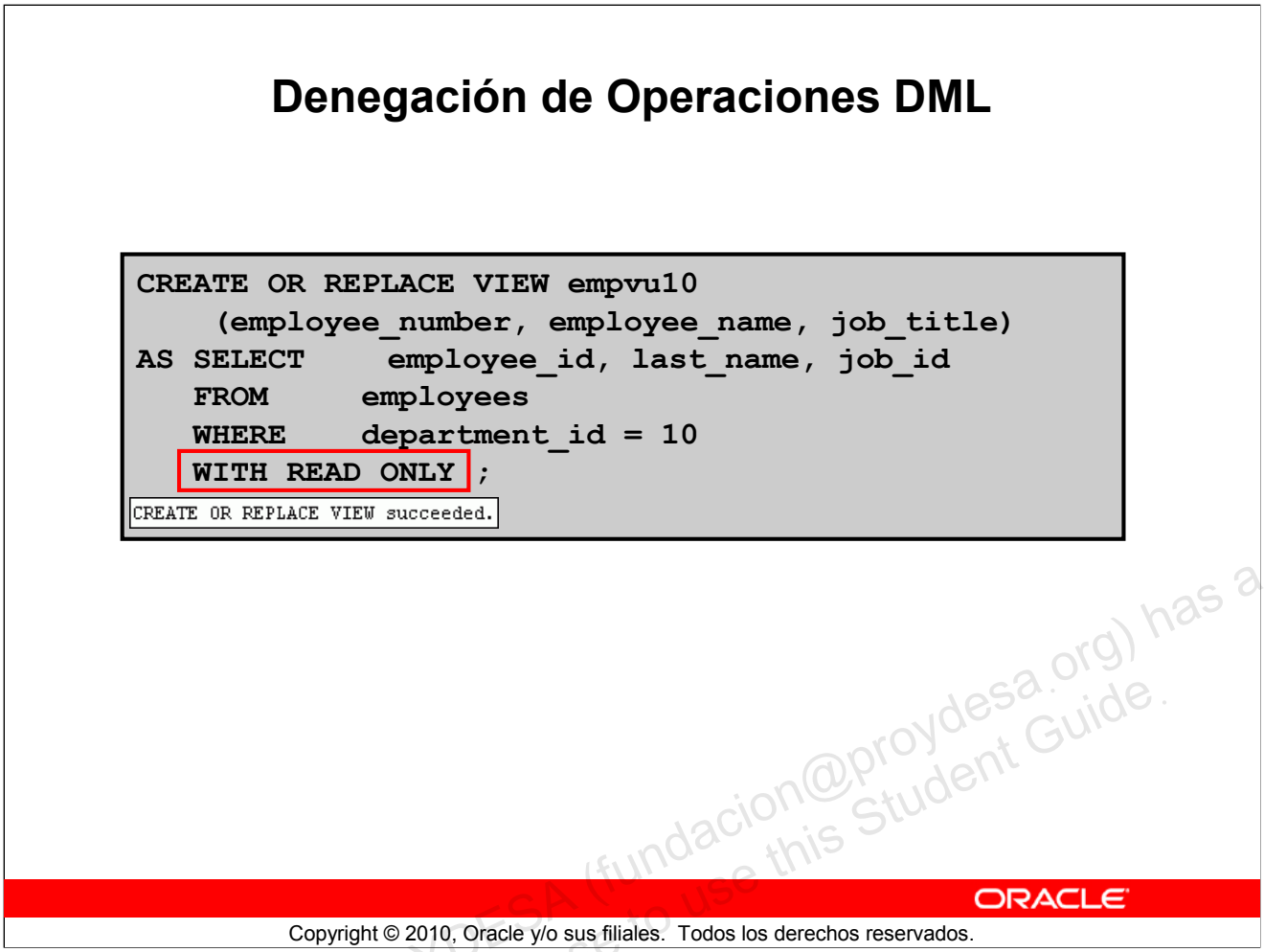
Denegación de Operaciones DML

- Para asegurarse de que no se realice ninguna operación DML, agregue la opción `WITH READ ONLY` a la definición de vista.
- Cualquier intento de realizar una operación DML en cualquier fila de la vista provocará un error del servidor de Oracle.

ORACLE

Denegación de Operaciones DML

Puede asegurarse de que no se realice ninguna operación DML en la vista, créela con la opción `WITH READ ONLY`. En el ejemplo de la siguiente diapositiva se modifica la vista `EMPVU10` para evitar operaciones DML en la vista.



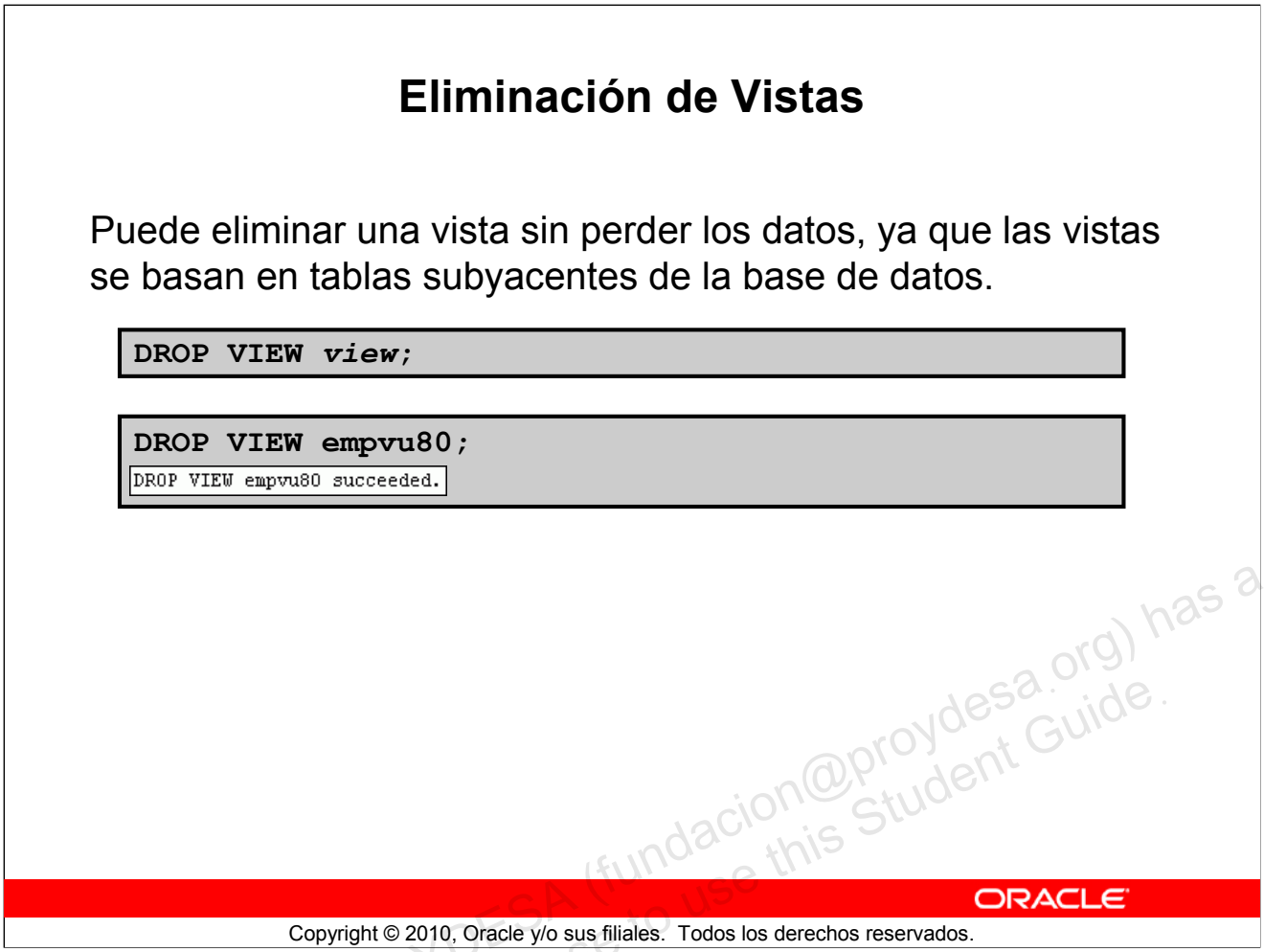
Denegación de Operaciones DML (continuación)

Cualquier intento de eliminar una fila de una vista con una restricción de sólo lectura provocará un error:

```
DELETE FROM empvu10
WHERE  employee_number = 200;
```

Igualmente, cualquier intento de insertar o modificar una fila mediante una vista con una restricción de sólo lectura provocará el mismo error.

```
Error report:
SQL Error: ORA-42399: cannot perform a DML operation on a read-only view
```



Eliminación de Vistas

Puede utilizar la sentencia `DROP VIEW` para eliminar una vista. La sentencia elimina la definición de vista de la base de datos. Sin embargo, el borrado de vistas no tiene ningún efecto en las tablas en las que se basan las vistas. Por otro lado, las vistas y otras aplicaciones basadas en las vistas suprimidas se convierten en no válidas. Sólo el creador o un usuario con el privilegio `DROP ANY VIEW` puede eliminar una vista.

En la sintaxis, *`view`* es el nombre de la vista.

Práctica 11: Visión General de la Parte 1

En esta práctica se abordan los siguientes temas:

- Creación de una vista simple
- Creación de una vista compleja
- Creación de una vista con restricción de comprobación
- Intento de modificar datos de la vista
- Eliminación de vistas

ORACLE

Práctica 11: Visión General de la Parte 1

La Parte 1 de la práctica de esta lección ofrece varios ejercicios de creación, uso y eliminación de vistas. Complete las preguntas de la 1 a la 6 al final de esta lección.

Agenda

- Visión general de la vistas:
  - Creación, modificación y recuperación de datos de una vista
  - Operaciones DML en una vista
  - Borrado de una vista
- Visión general de secuencias:
  - Creación, uso y modificación de una secuencia
  - Valores de secuencia de caché
  - Pseudocolumnas `NEXTVAL` y `CURRVAL`
- Visión general de índices
  - Creación y borrado de índices
- Visión general de sinónimos
  - Creación y borrado de sinónimos

ORACLE

Secuencias

Objeto	Descripción
Tabla	Unidad básica de almacenamiento; compuesta por filas.
Vista	Representa de forma lógica subjuegos de datos de una o más tablas.
Secuencia	Genera valores numéricos.
Índice	Mejora el rendimiento de algunas consultas.
Sinónimo	Ofrece nombres alternativos para los objetos.

ORACLE

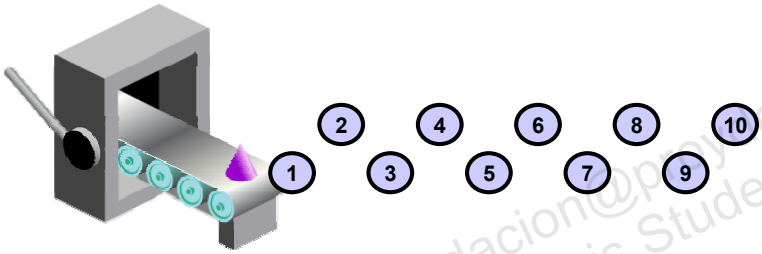
Secuencias

Una secuencia es un objeto de base de datos que crea valores enteros. Puede crear secuencias y, a continuación, utilizarlas para generar números.

Secuencias

Una secuencia:

- Puede generar automáticamente números únicos
- Es un objeto que se puede compartir
- Se puede utilizar para crear un valor de clave primaria
- Sustituye el código de aplicación
- Acelera la eficacia del acceso a los valores de secuencia cuando están almacenados en caché



ORACLE

Secuencias (continuación)

Una secuencia es un objeto de base de datos creado por el usuario que pueden compartir varios usuarios para generar enteros.

Puede definir una secuencia para generar valores únicos o reciclar y volver a utilizar los mismos números.

El uso normal de las secuencias es la creación de un valor de clave primaria, que debe ser único para cada fila. Una secuencia se genera y aumenta (o disminuye) mediante una rutina interna de Oracle. Éste puede ser un objeto de ahorro de tiempo, ya que reduce la cantidad de código de aplicación necesario para escribir una rutina de generación de secuencia.

Los números de secuencia se almacenan y generan independientemente de las tablas. Por lo tanto, la misma secuencia se puede utilizar para varias tablas.

Sentencia CREATE SEQUENCE: Sintaxis

Definir una secuencia para generar números secuenciales automáticamente:

```
CREATE SEQUENCE sequence
       [INCREMENT BY n]
       [START WITH n]
       [{MAXVALUE n | NOMAXVALUE}]
       [{MINVALUE n | NOMINVALUE}]
       [{CYCLE | NOCYCLE}]
       [{CACHE n | NOCACHE}];
```

ORACLE

Sentencia CREATE SEQUENCE: Sintaxis

Genera números secuenciales automáticamente mediante la sentencia CREATE SEQUENCE.

En la sintaxis:

sequence	es el nombre del generador de secuencias.
INCREMENT BY *n*	especifica el intervalo entre números de secuencia, donde *n* es un entero (Si se omite esta cláusula, la secuencia aumenta en 1).
START WITH *n*	especifica el primer número de secuencia que se va a generar (Si se omite esta cláusula, la secuencia empieza con 1).
MAXVALUE *n*	especifica el valor máximo que puede generar la secuencia.
NOMAXVALUE	especifica un valor máximo de 10^27 para una secuencia ascendente y –1 para una secuencia descendente (Ésta es la opción por defecto).
MINVALUE *n*	especifica el valor mínimo de secuencia.
NOMINVALUE	especifica un valor mínimo de 1 para una secuencia ascendente y –(10^26) para una secuencia descendente (Ésta es la opción por defecto).

Creación de Secuencias

- Crear una secuencia con el nombre `DEPT_DEPTID_SEQ` que se utilizará para la clave primaria de la tabla `DEPARTMENTS`.
- No utilizar la opción `CYCLE`.

```
CREATE SEQUENCE dept_deptid_seq
                INCREMENT BY 10
                START WITH 120
                MAXVALUE 9999
                NOCACHE
                NOCYCLE;
```

```
CREATE SEQUENCE succeeded.
```


Creación de una Secuencia (continuación)

`CYCLE | NOCYCLE`	especifica si la secuencia sigue generando valores después de alcanzar su valor máximo o mínimo (`NOCYCLE` es la opción por defecto).
`CACHE n | NOCACHE`	especifica cuántos valores preasigna el servidor de Oracle el servidor de Oracle en la memoria (por defecto, el servidor de Oracle almacena en caché 20 valores).

En el ejemplo de la diapositiva se crea una secuencia con el nombre `DEPT_DEPTID_SEQ` que se utilizará para la columna `DEPARTMENT_ID` de la tabla `DEPARTMENTS`. La secuencia se inicia en 120, no permite el almacenamiento en caché y no sigue ningún ciclo.

No utilice la opción `CYCLE` si la secuencia se utiliza para generar valores de clave primaria, a menos que disponga de un mecanismo fiable que depure las filas antiguas más rápido que los ciclos de secuencia.

Para obtener más información, consulte la sección sobre `CREATE SEQUENCE` en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Nota: la secuencia no está ligada a una tabla. Como norma general, debe asignar un nombre a la secuencia según el uso que desee darle. Sin embargo, la secuencia se puede utilizar en cualquier lugar, independientemente de su nombre.

Pseudocolumnas NEXTVAL y CURRVAL

- NEXTVAL devuelve el siguiente valor de secuencia disponible. Devuelve un valor único cada vez que se hace referencia a dicha columna, incluso para diferentes usuarios.
- CURRVAL obtiene el valor de secuencia actual.
- NEXTVAL se debe emitir para dicha secuencia antes de que CURRVAL contenga un valor.

ORACLE

Pseudocolumnas NEXTVAL y CURRVAL

Después de crear la secuencia, se generan números secuenciales para utilizarlos en las tablas. Haga referencia a los valores de secuencia mediante las pseudocolumnas NEXTVAL y CURRVAL.

La pseudocolumna NEXTVAL se utiliza para extraer números de secuencia sucesivos de una secuencia especificada. Debe cualificar a NEXTVAL con el nombre de secuencia. Al hacer referencia a *sequence*.NEXTVAL, se genera un nuevo número de secuencia y el actual se sustituye en CURRVAL.

La pseudocolumna CURRVAL se utiliza para hacer referencia al número de secuencia que acaba de generar el usuario actual. Sin embargo, NEXTVAL se debe utilizar para generar un número de secuencia en la sesión del usuario actual antes de que se pueda hacer referencia a CURRVAL. Debe cualificar a CURRVAL con el nombre de secuencia. Al hacer referencia a *sequence*.CURRVAL, se muestra el último valor devuelto al proceso de ese usuario.

Pseudocolumnas NEXTVAL y CURRVAL (continuación)

Reglas para el Uso de NEXTVAL y CURRVAL

Puede utilizar NEXTVAL y CURRVAL en los siguientes contextos:

- La lista SELECT de una sentencia SELECT que no forme parte de una subconsulta
- La lista SELECT de una subconsulta de la sentencia INSERT
- La cláusula VALUES de una sentencia INSERT
- La cláusula SET de la sentencia UPDATE

Puede utilizar NEXTVAL y CURRVAL en los siguientes contextos:

- La lista SELECT de una vista
- Una sentencia SELECT con la palabra clave DISTINCT
- Una sentencia SELECT con las cláusulas GROUP BY, HAVING o ORDER BY
- Una subconsulta en una sentencia SELECT, DELETE o UPDATE
- La expresión DEFAULT en una sentencia CREATE TABLE o ALTER TABLE

Para obtener más información, consulte las secciones sobre pseudocolumnas y sobre CREATE SEQUENCE en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10*g* u 11*g*.

Uso de una Secuencia

- Insertar un nuevo departamento denominado “Support” en la ubicación con el ID 2500:

```
INSERT INTO departments(department_id,
            department_name, location_id)
VALUES      (dept_deptid_seq.NEXTVAL,
            'Support', 2500);
```

```
1 rows inserted
```

- Consultar el valor actual de la secuencia DEPT_DEPTID_SEQ:

```
SELECT   dept_deptid_seq.CURRVAL
FROM     dual;
```

ORACLE

Uso de una Secuencia

En el ejemplo de la diapositiva se inserta un nuevo departamento en la tabla DEPARTMENTS. Utiliza la secuencia DEPT_DEPTID_SEQ para generar un nuevo número de departamento como se muestra a continuación.

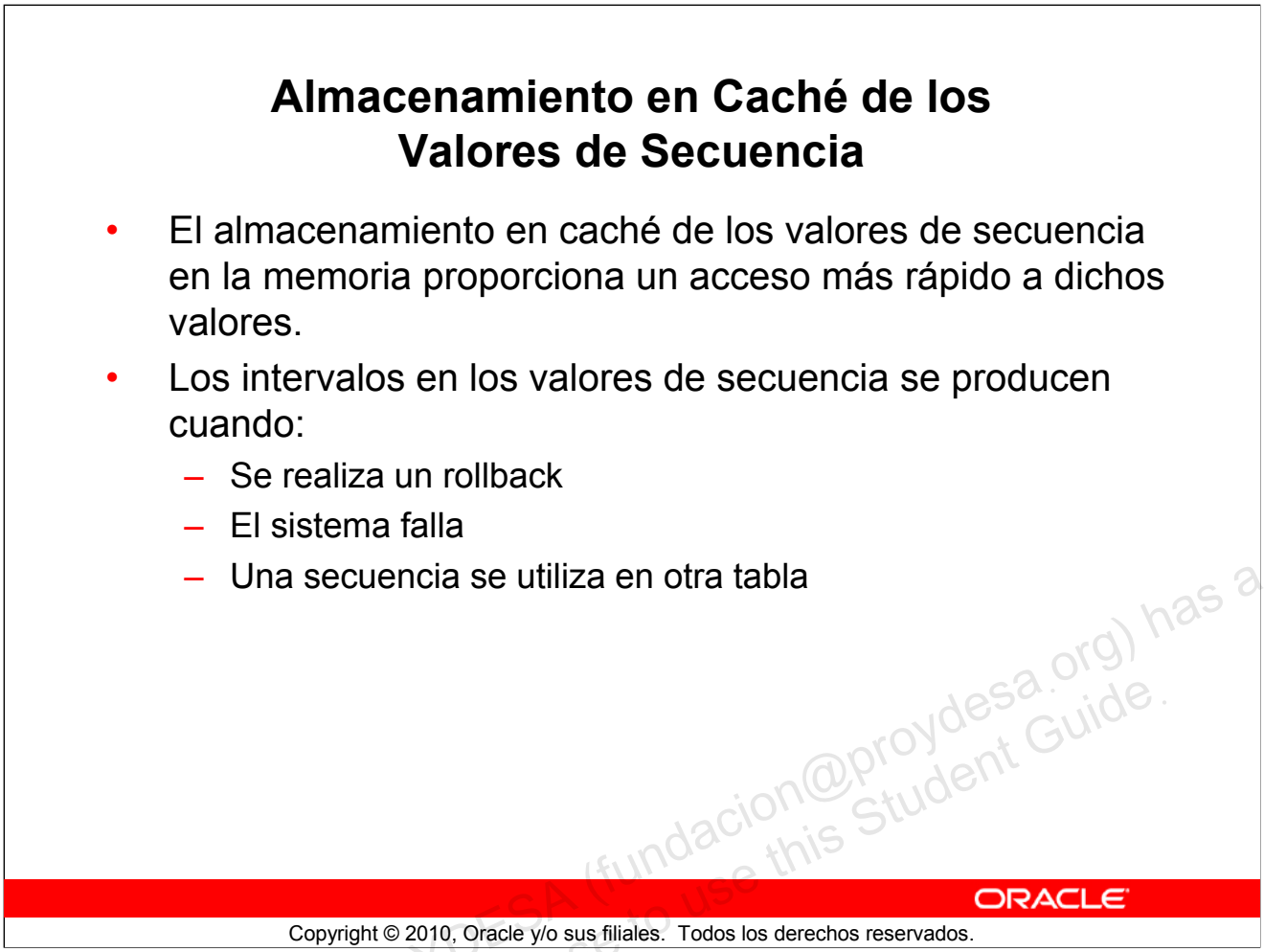
Puede visualizar el valor actual de la secuencia mediante *sequence_name*.CURRVAL, se muestra en el ejemplo de la diapositiva. La salida de la consulta se muestra a continuación:

	CURRVAL
1	120

Supongamos que ahora desea contratar empleados para el nuevo departamento. La sentencia INSERT que se debe ejecutar para los nuevos empleados puede incluir el siguiente código:

```
INSERT INTO employees (employee_id, department_id, ...)
VALUES (employees_seq.NEXTVAL, dept_deptid_seq .CURRVAL, ...);
```

Nota: en el ejemplo anterior, se asume que la secuencia denominada EMPLOYEE_SEQ ya se ha creado para generar los números de los nuevos empleados.



Almacenamiento en Caché de los Valores de Secuencia

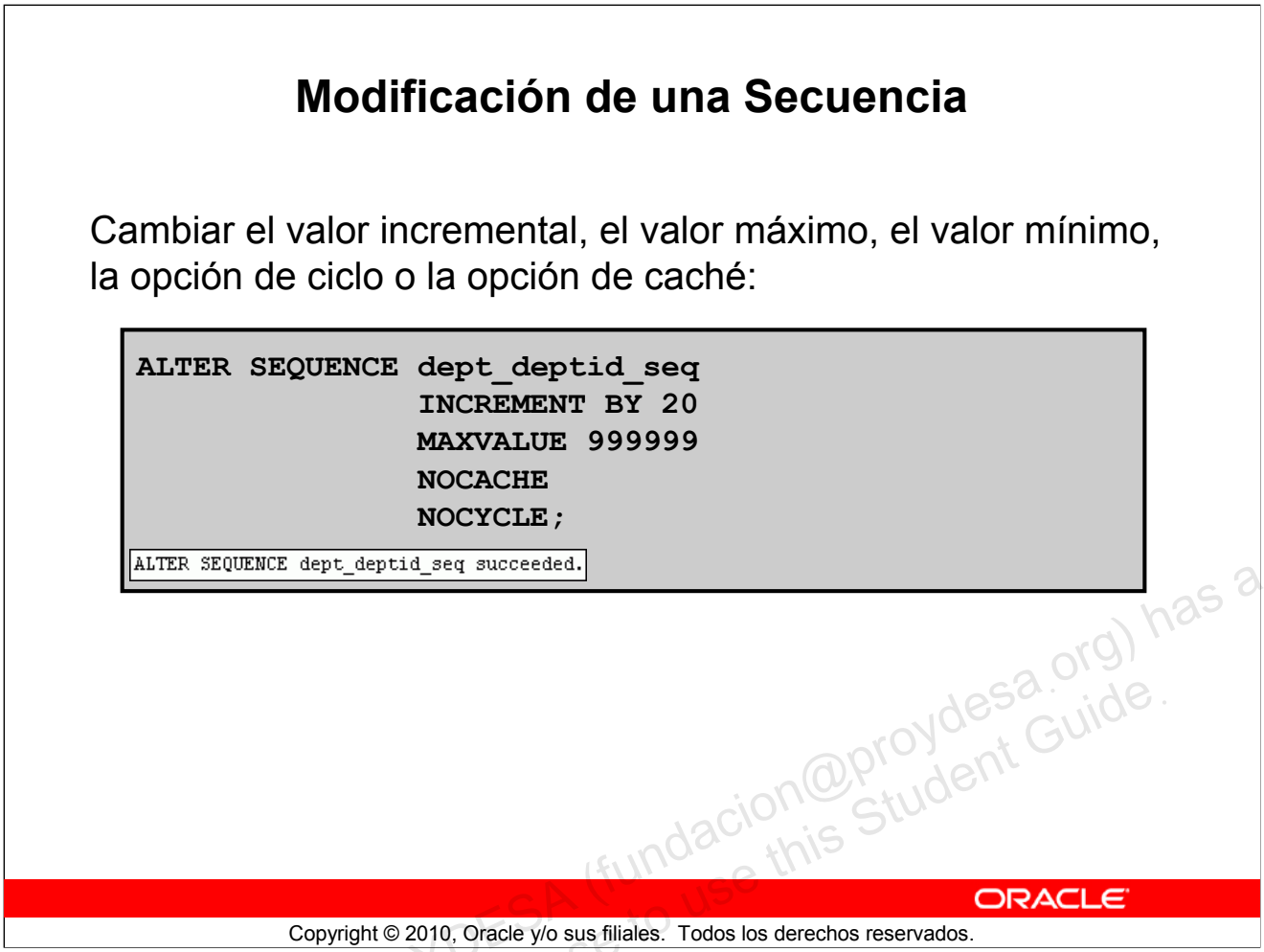
Puede almacenar las secuencias en caché en la memoria para proporcionar un acceso más rápido a los valores de dichas secuencias. La caché se rellena la primera vez que hace referencia a la secuencia. Las solicitudes del siguiente valor de secuencia se recuperan de la secuencia almacenada en caché. Después de utilizar el último valor de secuencia, la siguiente solicitud de la secuencia introduce otra caché de secuencias en la memoria.

Intervalos en la Secuencia

Aunque los generadores de secuencias emiten números secuenciales sin intervalos, esta acción se realiza independientemente de que se realice una confirmación o un rollback. Por lo tanto, si realiza un rollback de una sentencia que contiene una secuencia, se pierde el número.

Otro evento que puede provocar intervalos en la secuencia es un fallo del sistema. Si la secuencia almacena los valores en caché en la memoria, estos valores se pierden si se produce un fallo del sistema.

Puesto que las secuencias no están ligadas directamente a las tablas, se puede utilizar la misma secuencia para varias tablas. Si lo hace así, cada tabla puede contener intervalos en los números secuenciales.



Modificación de una Secuencia

Si alcanza el límite `MAXVALUE` para la secuencia, no se asignan valores adicionales de la secuencia y recibirá un error que indica que la secuencia excede el valor `MAXVALUE`. Para seguir utilizando la secuencia, puede modificarla mediante la sentencia `ALTER SEQUENCE`.

Sintaxis

```
ALTER  SEQUENCE    secuencia
      [INCREMENT BY n]
      [{MAXVALUE n | NOMAXVALUE}]
      [{MINVALUE n | NOMINVALUE}]
      [{CYCLE | NOCYCLE}]
      [{CACHE n | NOCACHE}];
```

En la sintaxis, *sequence* es el nombre del generador de secuencias.

Para obtener más información, consulte la sección sobre `ALTER SEQUENCE` en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10g u 11g.

Instrucciones para Modificar una Secuencia

- Debe ser el propietario o tener el privilegio `ALTER` para la secuencia.
- Sólo se ven afectados los números de secuencia futuros.
- La secuencia se debe borrar y volver a crear para reiniciar la secuencia en un número diferente.
- Se realiza alguna validación.
- Para eliminar una secuencia, utilice la sentencia `DROP`:

```
DROP SEQUENCE dept_deptid_seq;
DROP SEQUENCE dept_deptid_seq succeeded.
```


Instrucciones para Modificar una Secuencia

- Debe ser el propietario o tener el privilegio `ALTER` para la secuencia si desea modificarla. Debe ser el propietario o tener el privilegio `DROP ANY SEQUENCE` para eliminarla.
- Sólo se ven afectados por la sentencia `ALTER SEQUENCE` los números de secuencia futuros.
- La opción `START WITH` no se puede cambiar mediante `ALTER SEQUENCE`. La secuencia se debe borrar y volver a crear para reiniciar la secuencia en un número diferente.
- Se realiza alguna validación. Por ejemplo, no se puede imponer un nuevo valor `MAXVALUE` menor que el número de secuencia actual.

```
ALTER SEQUENCE dept_deptid_seq
      INCREMENT BY 20
      MAXVALUE 90
      NOCACHE
      NOCYCLE;
```

- El error:

```
Error report:
SQL Error: ORA-04009: MAXVALUE cannot be made to be less than the current value
04009. 00000 -  "MAXVALUE cannot be made to be less than the current value"
*Cause:    the current value exceeds the given MAXVALUE
*Action:   make sure that the new MAXVALUE is larger than the current value
```

Agenda

- Visión general de la vistas:
  - Creación, modificación y recuperación de datos de una vista
  - Operaciones DML en una vista
  - Borrado de una vista
- Visión general de secuencias:
  - Creación, uso y modificación de una secuencia
  - Valores de secuencia de caché
  - Pseudocolumnas `NEXTVAL` y `CURRVAL`
- Visión general de índices
  - Creación y borrado de índices
- Visión general de sinónimos
  - Creación y borrado de sinónimos

ORACLE

Índices

Objeto	Descripción
Tabla	Unidad básica de almacenamiento; compuesta por filas.
Vista	Representa de forma lógica subjuegos de datos de una o más tablas.
Secuencia	Genera valores numéricos.
Índice	Mejora el rendimiento de algunas consultas.
Sinónimo	Ofrece nombres alternativos para los objetos.

ORACLE

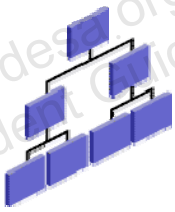
Índices

Los índices son objetos de base de datos que puede crear para mejorar el rendimiento de algunas consultas. Los índices también los puede crear automáticamente el servidor al crear una restricción de clave primaria o única.

Índices

Un índice:

- Es un objeto de esquema
- Lo puede utilizar el servidor de Oracle para acelerar la recuperación de filas mediante un puntero
- Puede reducir la entrada/salida (E/S) de disco mediante un método de ruta de acceso rápido para buscar datos de forma rápida
- Es independiente de la tabla que indexa
- Lo utiliza y mantiene automáticamente el servidor de Oracle



Índices (continuación)

Un índice de servidor de Oracle es un esquema de objeto que puede acelerar la recuperación de filas mediante un puntero. Los índices se pueden crear explícita o automáticamente. Si no hay un índice en la columna, se produce una exploración de tabla completa.

Un índice proporciona acceso directo y rápido a las filas de una tabla. Su finalidad es reducir la E/S de disco mediante una ruta de acceso indexada para buscar datos de forma rápida. Un índice lo utiliza y mantiene automáticamente el servidor de Oracle. Después de crear un índice, no será necesaria ninguna intervención directa por parte del usuario.

Los índices son lógica y físicamente independientes de la tabla que indexan. Esto significa que se pueden crear o borrar en cualquier momento sin que afecten a las tablas base o a otros índices.

Nota: al borrar una tabla, se borran también los índices correspondientes.

Para obtener más información, consulte la sección sobre objetos de esquema: índices en *Oracle Database Concepts* (Conceptos de Oracle Database) *para la base de datos 10g u 11g*.

¿Cómo se Crean los Índices?

- Automáticamente: al definir una restricción `PRIMARY KEY` o `UNIQUE` en una definición de tabla, se crea un índice único automáticamente.

- Manualmente: los usuarios pueden crear índices no únicos en las columnas para acelerar el acceso a las filas.

¿Cómo se Crean los Índices?

Puede crear dos tipos de índices.

- **Índice único:** el servidor de Oracle crea automáticamente este índice al definir una columna en la tabla para tener una restricción `PRIMARY KEY` o `UNIQUE`. El nombre del índice es el nombre proporcionado a la restricción.
- **Índice no único:** se trata de un índice que puede crear el usuario. Por ejemplo, puede crear un índice de columna `FOREIGN KEY` para una unión en una consulta para mejorar la velocidad de recuperación.

Nota: puede crear manualmente un índice único, pero se recomienda crear una restricción única, que implícitamente crea el índice único.

Creación de Índices

- Crear un índice en una o más columnas:

```
CREATE [UNIQUE][BITMAP]INDEX index
ON table (column[, column]...);
```

- Mejorar la velocidad de acceso de consulta a la columna LAST_NAME de la tabla EMPLOYEES:

```
CREATE INDEX  emp_last_name_idx
ON            employees(last_name);
CREATE INDEX succeeded.
```

Creación de Índices

Para crear un índice en una o más columnas, emita la sentencia CREATE INDEX.

En la sintaxis:

- index es el nombre del índice.
- table es el nombre de la tabla.
- column es el nombre de la columna de la tabla que se debe rellenar.

Especifique UNIQUE para indicar que el valor de la columna (o columnas) en la que se basa el índice debe ser único. Especifique BITMAP para indicar que el índice se debe crear con un bitmap para cada clave distinta, en lugar de indexar cada fila por separado. Los índices de bitmap almacenan los rowids asociados con un valor de clave como bitmap.

Para obtener más información, consulte la sección sobre CREATE INDEX en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database).

Instrucciones de Creación de Índices

Más No Siempre Es Sinónimo de Mejor

El hecho de tener más índices en una tabla no significa que las consultas sean más rápidas. Cada operación DML que se confirma en una tabla con índices implica la actualización de los índices. Cuanto mayor sea el número de índices asociados a una tabla, mayor será el esfuerzo que debe realizar el servidor de Oracle para actualizar todos los índices después de la operación DML.

Cuándo Crear un Índice

Por lo tanto, sólo debe crear índices si:

- La columna contiene una amplia variedad de valores
- La columna contiene un gran número de valores nulos
- Una o más columnas se utilizan con frecuencia en conjunto en una cláusula `WHERE` o una condición de unión
- La tabla es grande y se espera que la mayoría de las consultas recuperen menos del 2% al 4% de las filas

Recuerde que si desea reforzar la unicidad, debe definir una restricción única en la definición de tabla. En ese caso, se crea un índice único automáticamente.

Eliminación de Índices

- Eliminar un índice del diccionario de datos mediante el comando `DROP INDEX`:

```
DROP INDEX index;
```

- Eliminar el índice `emp_last_name_idx` del diccionario de datos:

```
DROP INDEX emp_last_name_idx;
DROP INDEX emp_last_name_idx succeeded.
```

- Para borrar un índice, debe ser el propietario del mismo o tener el privilegio `DROP ANY INDEX`.

ORACLE

Eliminación de Índices

No puede modificar los índices. Para cambiar un índice, debe borrarlo y volver a crearlo.

Eliminar una definición de índice del diccionario de datos mediante la emisión de la sentencia `DROP INDEX`. Para borrar un índice, debe ser el propietario del mismo o tener el privilegio `DROP ANY INDEX`.

En la sintaxis, *`índice`* es el nombre del índice.

Nota: si borra una tabla, los índices y restricciones se borran automáticamente, pero permanecen las vistas y secuencias.

Agenda

- Visión general de la vistas:
  - Creación, modificación y recuperación de datos de una vista
  - Operaciones DML en una vista
  - Borrado de una vista
- Visión general de secuencias:
  - Creación, uso y modificación de una secuencia
  - Valores de secuencia de caché
  - Pseudocolumnas `NEXTVAL` y `CURRVAL`
- Visión general de índices
  - Creación y borrado de índices
- Visión general de sinónimos
  - Creación y borrado de sinónimos

ORACLE

Sinónimos

Objeto	Descripción
Tabla	Unidad básica de almacenamiento; compuesta por filas.
Vista	Representa de forma lógica subjuegos de datos de una o más tablas.
Secuencia	Genera valores numéricos.
Índice	Mejora el rendimiento de algunas consultas.
Sinónimo	Ofrece nombres alternativos para los objetos.

ORACLE

Sinónimos

Los sinónimos son objetos de base de datos que permiten llamar a una tabla por otro nombre. Puede crear sinónimos para proporcionar un nombre alternativo a la tabla.

Creación de un Sinónimo para un Objeto

Simplificar el acceso a los objetos mediante la creación de un sinónimo (otro nombre para un objeto). Con los sinónimos, puede:

- Crear una referencia más sencilla a una tabla propiedad de otro usuario
- Acortar nombres de objetos largos

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```


ORACLE

Creación de un Sinónimo para un Objeto

Para hacer referencia a una tabla propiedad de otro usuario, debe anteponer al nombre de la tabla el nombre del usuario que la haya creado, seguido de un punto. Con la creación de un sinónimo se elimina la necesidad de cualificar el nombre del objeto con el esquema y se ofrece nombres alternativos para tablas, vistas, secuencias, procedimientos u otros objetos. Este método es especialmente útil con nombres de objetos largos, como las vistas.

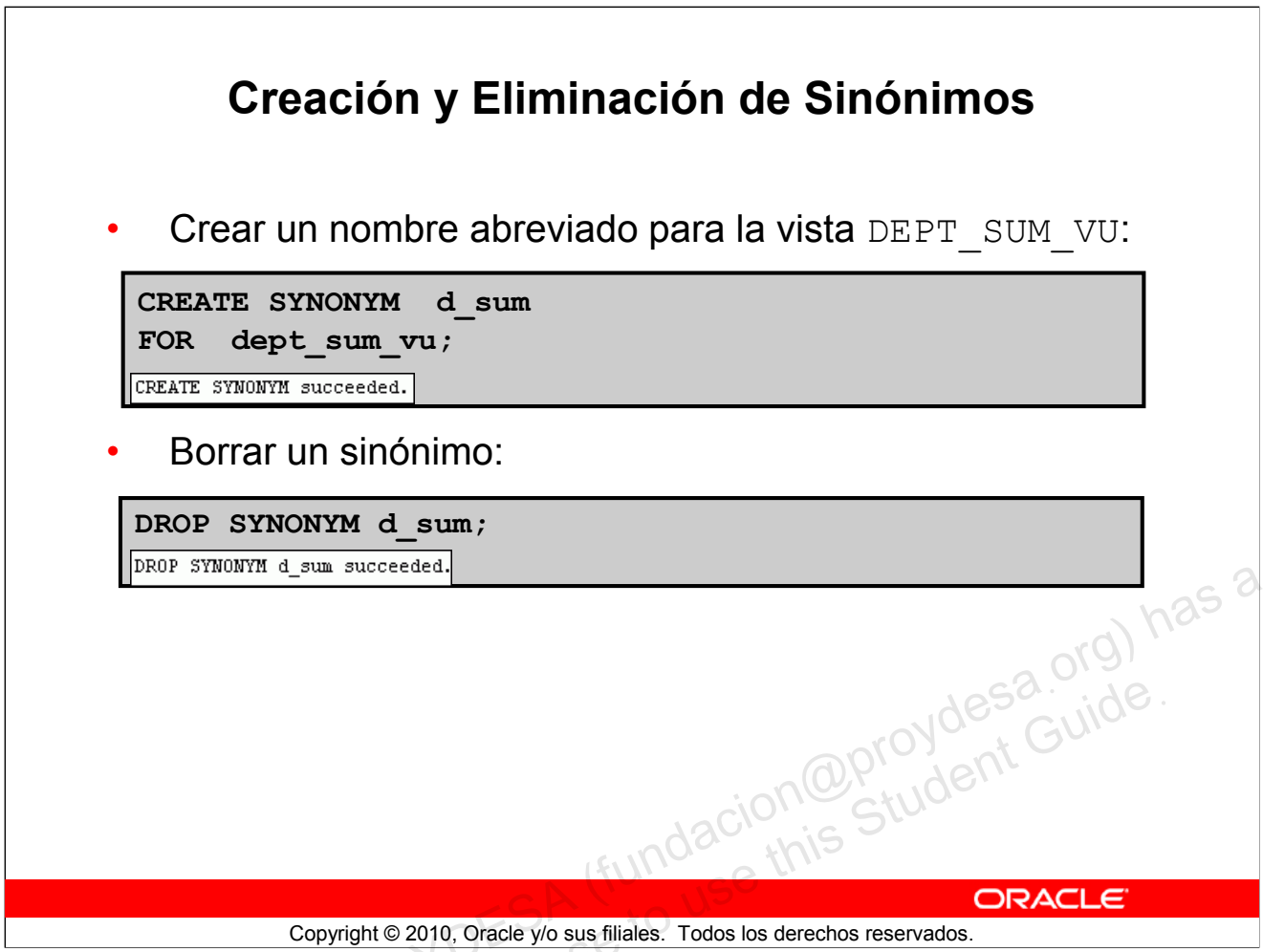
En la sintaxis:

`PUBLIC`	crea un sinónimo al que pueden acceder todos los usuarios.
`synonym`	es el nombre del sinónimo que se va a crear.
`object`	identifica el objeto para el que se crea el sinónimo.

Instrucciones

- El objeto no puede estar en un paquete.
- Un nombre sinónimo privado debe ser distinto de todos los demás objetos propiedad del mismo usuario.

Para obtener más información, consulte la sección sobre "`CREATE SYNONYM`" en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10g u 11g.



Creación y Eliminación de Sinónimos

Creación de Sinónimos

En la diapositiva del ejemplo se crea un sinónimo para la vista DEPT_SUM_VU para una referencia más rápida.

El administrador de la base de datos puede crear un sinónimo público al que puedan acceder todos los usuarios. En el siguiente ejemplo se crea un sinónimo público con el nombre DEPT para la tabla DEPARTMENTS de Alice:

```
CREATE PUBLIC SYNONYM  dept
FOR    alice.departments;
```

```
CREATE SYNONYM succeeded.
```

Eliminación de Sinónimos

Para eliminar un sinónimo, utilice la sentencia DROP SYNONYM. Sólo el administrador de la base de datos puede borrar un sinónimo público.

```
DROP PUBLIC SYNONYM  dept;
```

Para obtener más información, consulte la sección sobre DROP SYNONYM en *Oracle Database SQL Language Reference* (Referencia sobre Lenguaje SQL de Oracle Database) para la base de datos 10g u 11g.

Prueba

Los índices se deben crear manualmente y sirven para acelerar el acceso a las filas de la tabla.

1. Verdadero
2. Falso

Respuesta: 2

Nota: los índices se diseñan para acelerar el rendimiento de las consultas. Sin embargo, no todos los índices se crean manualmente. El servidor de Oracle crea automáticamente este índice al definir una columna en la tabla para tener una restricción `PRIMARY KEY` o `UNIQUE`.

Resumen

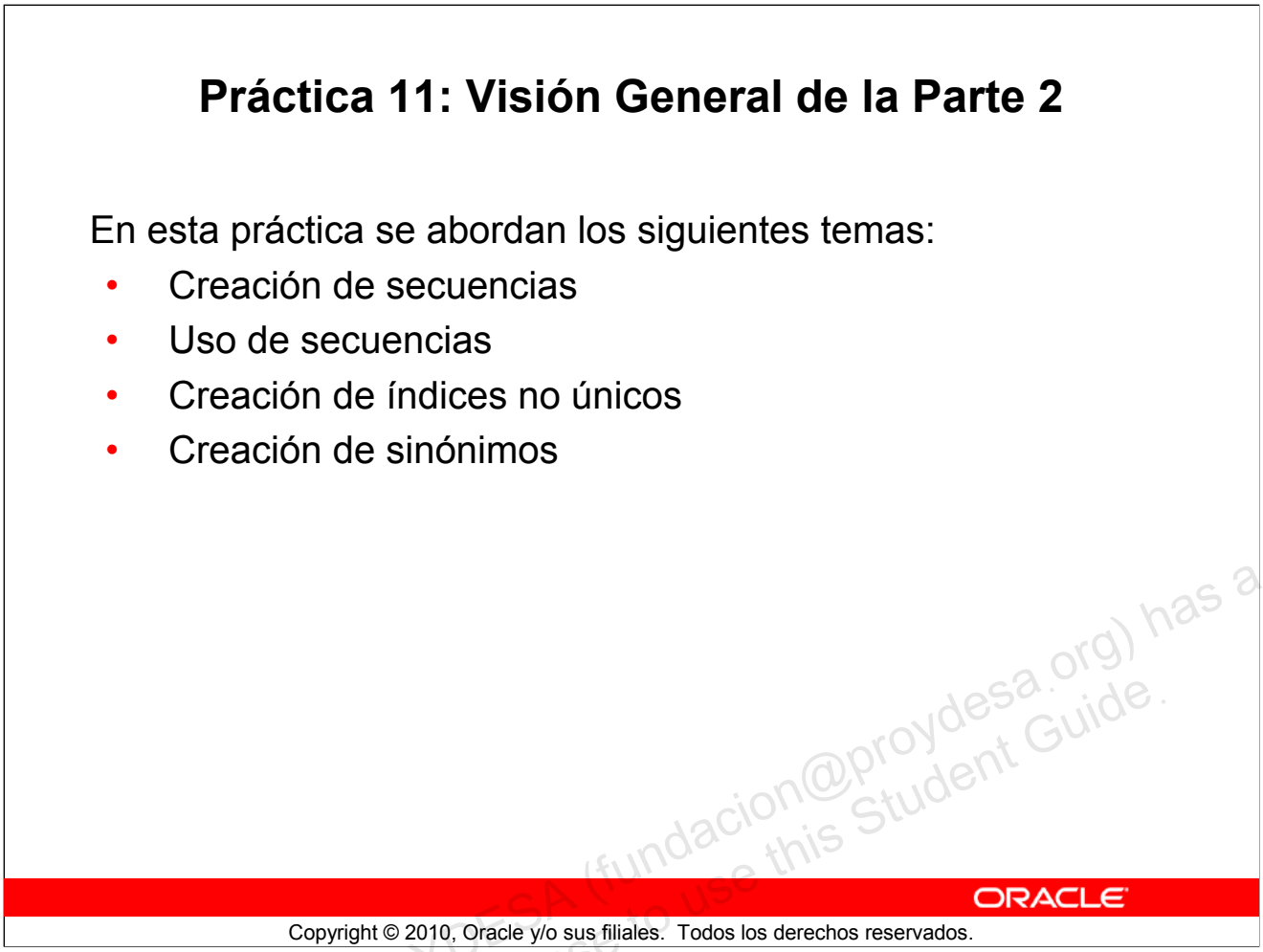
En esta lección, debe haber aprendido lo siguiente:

- Crear, utilizar y eliminar vistas
- Generar números de secuencia automáticamente con el generador de secuencias
- Crear índices para mejorar la velocidad de recuperación de las consultas
- Utilizar sinónimos para proporcionar nombres alternativos para los objetos

ORACLE

Resumen

En esta lección, debe haber aprendido acerca de los objetos de base de datos como las vistas, secuencias, índices y sinónimos.



Práctica 11: Visión General de la Parte 2

La Parte 2 de la práctica de esta lección ofrece varios ejercicios de creación y uso de una secuencia, un índice y un sinónimo.

Complete las preguntas de la 7 a la 10 al final de esta lección.