|  |
| --- |
|  |

Programación de Base de Datos (MDY3231)

**Experiencia 1**

**Semana 4: Trabajando con cursores explícitos en funciones y procedimientos almacenados.**

Temáticas

[Introducción a la semana 3](#_Toc126234226)

[Resultado de Aprendizaje (RA) – Indicador de Logro (IL) 3](#_Toc126234227)

[Palabras claves 4](#_Toc126234228)

[Preguntas gatillantes 4](#_Toc126234229)

[Conceptos previos 5](#_Toc126234230)

[Cursor explícito 5](#_Toc126234231)

[Comparación cursor implícito y explícito 6](#_Toc126234232)

[Funciones almacenadas con cursores explícitos 6](#_Toc126234233)

[Caso 1 6](#_Toc126234234)

[Procedimiento almacenado con cursores explícitos 10](#_Toc126234235)

[Caso 2 10](#_Toc126234236)

[Trabajando con sentencias DML 13](#_Toc126234237)

[Caso 3 13](#_Toc126234238)

[Cierre de la semana 17](#_Toc126234239)

[Referencias 18](#_Toc126234240)

[Bibliografía de la semana 18](#_Toc126234241)

[Apuntes 19](#_Toc126234242)

# **Introducción a la semana**

En esta semana deberás hacer la actividad sumativa llamada "Trabajando con cursores explícitos en funciones y procedimientos almacenados", que es la entrega de un producto a través de un encargo, el cual consta de construir funciones y procedimientos almacenados que utilicen variables compuestas, como lo son los cursores explícitos, utilizando el lenguaje de programación PL/SQL.

# **Resultado de Aprendizaje (RA) – Indicador de Logro (IL)**

|  |  |
| --- | --- |
| **Resultado de Aprendizaje (RA)** | **Indicador de Logro (IL)** |
| **RA1.** Construye el FrontEnd de una página web aplicando framework y lenguajes que respondan a la estructura visual y funcional requerida por el cliente. | **IL1.** Construye función almacenada con y sin parámetros utilizando sentencias y operadores SQL y PL/SQL para responder a los requerimientos de información y/o de la lógica de programación de acuerdo a reglas de negocio, sintaxis del lenguaje, restricciones del SGBD y buenas prácticas de la industria.  **IL2.** Construye procedimiento almacenado con y sin parámetros utilizando sentencias y operadores SQL y PL/SQL para responder a los requerimientos de información y/o de la lógica de programación de acuerdo a reglas de negocio, sintaxis del lenguaje, restricciones del SGBD y buenas prácticas de la industria. |

# **Palabras claves**

* Argumento o parámetro
* Argumento o parámetro de entrada
* Argumento o parámetro de salida
* Cursor implícito
* Cursor explícito
* Dato
* Función almacenada
* Procedimiento almacenado
* PL/SQL
* Tipo de dato
* Variable

# **Preguntas gatillantes**

* ¿Cómo puedo trabajar con un conjunto de registros?
* ¿Es posible que una unidad de programación: función o procedimiento, modifique el contenido de una tabla agregando o modificando registros?
* ¿Puede un procedimiento almacenado en la base de datos ser llamado por otras funciones y/o procedimientos?
* ¿De qué forma se logra recuperar información de la base de datos desde una función almacenada o un procedimiento almacenado?

# **Conceptos previos**

Las unidades de programación construidas hasta el momento son funciones y procedimientos almacenados que, como su nombre lo indica, permanecen guardadas en la base de datos junto al resto de los objetos (tablas, vistas, índices, secuencias, entre otros). Una de las características que tienen es que permiten modularizar la solución permitiendo de esta forma separar la solución en pequeñas soluciones que, en conjunto, logran el objetivo final (resolver el problema planteado).

Tal y como se da en otros lenguajes de programación, en donde es posible incorporar las estructuras de control: secuencia, condición y repetición; en las funciones y procedimientos almacenados también es posible incorporarlas; hasta ahora lo hemos realizado y hemos usando secuencia y condiciones; sin embargo, la pregunta es qué pasa con los ciclos repetitivos, la respuesta: uno de los usos es para recorrer una estructura usada en PL/SQL llamada cursores explícitos los cuales veremos a profundidad en esta semana y sabremos cómo se pueden incorporar a las unidades de programación que vamos a construir.

# **Cursor explícito**

Hasta el momento, hemos construido funciones y procedimientos que trabajan con cursores implícitos que están pensados para trabajar con sentencias que retornan a lo más un registro; sin embargo, cómo se hace cuando se quiere recuperar más de un registro; la respuesta: a través del uso de cursores explícitos.

## Comparación cursor implícito y explícito

Tabla 1: Comparación

|  |  |
| --- | --- |
| **CURSOR IMPLÍCITO** | **CURSOR EXPLÍCITO** |
| Requiere de una cláusula INTO para indicar donde debe guardar lo que se incluye en la sentencia SELECT. | NO utiliza la cláusula INTO |
| Se ubica entre las cláusulas BEGIN y END de una unidad de programación. | Se ubica entre las cláusulas IS y BEGIN de una función almacenada |
| Se utiliza cuando el conjunto de registros recuperados cuenta con, a lo más, un registro. | Se utilizan cuando el conjunto de registros recuperados contiene 2 o más registros; aunque de igual forma se puede usar cuando hay menos. |
| NO es necesario recorrerlo | DEBE ser recorrido a través de un ciclo repetitivo. |

Las sentencias SELECT que alimentan a un cursor pueden incluir todo lo revisado en el módulo de consultas de base de datos; es decir, uso de JOIN, funciones de grupo, funciones de fila; todo va a depender del nivel de complejidad dado por el requerimiento solicitado.

# **Funciones almacenadas con cursores explícitos**

## Caso 1

Para explicar el concepto vamos a resolver el siguiente requerimiento:

#### *Requerimiento*

Construir una función almacenada que imprima los datos de todos los cargos registrados y permita retornar si se logró o no procesar registros.

#### *Solución*

Figura 1: Función almacenada con cursor explícito

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Nota: Función almacenada que usa cursor explícito. Captura de pantalla, (2022). SQL Developer.

Analicemos el código:

* Las líneas 53 y 54 muestran la firma de la función: en este caso, la función no tiene argumentos y su valor de retorno es un BOOLEAN.
* Las líneas 57 a la 59 declaran un cursor explícito que permite guardar más de un registro (en este caso los cargos que están registrados).
* La línea 64 llama a la función creada en las secciones anteriores para poder, usando su valor de retorno, determinar si existen o no registros a procesar.
* La línea 65 evalúa (usando una estructura de control condicional) si existen o no registros a procesar.
* Las líneas 70 a la 72 establecen el recorrido del cursor; esto implica, acceder uno a uno a cada registro almacenado en el cursor; esto se logra utilizando una estructura de control repetitiva (en este caso el ciclo FOR).
* La línea 70 utiliza una variable compuesta llamada ***reg\_cargo*** que NO HA SIDO DECLARADA, esto se debe a que el FOR permite el uso de variables sin ser declaradas, que solo existirán mientras esté activo el ciclo (es lo que se conoce con el nombre de alcance o scope de una variable).
* La línea 71 permite imprimir los datos del registro que se está procesando en una determinada iteración del ciclo. Acá es relevante destacar el uso del operador punto (.) que permite acceder a los componentes de la variable compuesta reg\_cargo (que almacena un registro del cursor); los nombres que se utilizan después del operador punto (.) representa el campo que tiene el registro (en este caso: car\_id y nombre)
* Las líneas 67 y 74 indican el valor de retorno de la función; en este caso, son 2 porque dependiendo del resultado devuelto por la función será un verdadero o un falso. Anteriormente, se había mencionado que era posible encontrar más de una sentencia RETURN dentro de una función.

#### *Probando*

Nuevamente, como ya es costumbre, se debe construir un bloque anónimo para poder comprobar la ejecución de la función recién implementada.

Figura 2: Bloque anónimo

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Nota: Bloque anónimo para probar la función. Captura de pantalla, (2022). SQL Developer.

Acá se muestra la implementación del bloque anónimo para probar la función. En este caso se vuelve a utilizar la estructura condicional para decidir qué mensaje es el que se va a imprimir.

Al ejecutar el bloque el resultado es el que se muestra en la Figura 3, donde se aprecia el listado de todos los cargos registrados y la línea final indica el mensaje de acuerdo con lo retornado por la función.

Figura 3: Resultado

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Nota: Resultado de la ejecución del bloque. Captura de pantalla, (2022). SQL Developer.

# **Procedimiento almacenado con cursores explícitos**

Hasta el momento, al igual como sucedió con las funciones almacenadas, hemos construido procedimientos que trabajan con cursores implícitos que están pensados para trabajar con sentencias que retornan a lo más un registro; sin embargo, cómo se hace cuando se quiere recuperar más de un registro; la respuesta: a través del uso de cursores explícitos.

## Caso 2

Para continuar vamos a resolver el siguiente requerimiento:

#### *Requerimiento*

Construir un procedimiento almacenado que imprima el nombre de todas las especialidades registradas indicando el total de médicos con los que cuenta y permita retornarla cantidad de especialidades que fueron incluidas en el listado. El listado debe aparecer ordenado por total de médicos por especialidad de forma descendente y debe incluir solamente a las especialidades cuya cantidad de médicos asociados sea igual o superior a un cierto valor límite.

#### *Solución*

Figura 4: Procedimiento almacenado.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Nota: Procedimiento almacenado que usa cursor explícito. Captura de pantalla, (2022). SQL Developer.

La Figura 4, contiene la implementación del procedimiento que cumple con el requerimiento. Analicemos el código:

* La línea 44 define la firma del procedimiento: indicando su nombre y la lista de parámetros que el procedimiento requiere; en este caso, recibe 2 parámetros: uno de entrada (para recibir el límite) y uno de salida (para almacenar el total)
* Las líneas 46 a la 51 declaran el cursor explícito que permite recuperar los registros que serán incluidos en el listado; en este caso el cursor explícito recibe un argumento que representa el límite que se establece en el requerimiento.
* La línea 53 inicializa el parámetro de salida en CERO con el elemento neutro de la suma para poder contabilizar la cantidad de registros procesados. Acá es importante, considerar que una alternativa de solución puede ser obtener el total usando un cursor implícito (ver Figura 5 - Solución alternativa usando un cursor implícito)
* La línea 54 es la encargada de recorrer el cursor, registro a registro, como ya se mencionó en la guía anterior el contenido de cada uno de los registros contenidos en el cursor se almacena en la variable compuesta llamada r***eg\_especialidad*** que estará formada por las columnas que rescata la sentencia SELECT.

Figura 5: Solución alternativa.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Nota: Solución alternativa usando un cursor implícito. Captura de pantalla, (2022). SQL Developer.

#### *Probando*

Nuevamente, como ya es costumbre, se debe construir un bloque anónimo para poder comprobar la ejecución del procedimiento recién implementado.

Figura 6: Bloque anónimo

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Nota: Bloque anónimo para probar el procedimiento. Captura de pantalla, (2022). SQL Developer.

Al ejecutar el bloque e ingresar un 3 para el límite pedido, el resultado es el que se muestra en la Figura 7, se aprecia el listado de todas las especialidades que cuentan con más de 3 médicos asociados.

Figura 7: Resultado

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Nota: Resultado de la ejecución del bloque. Captura de pantalla, (2022). SQL Developer.

# **Trabajando con sentencias DML**

## Caso 3

Se requiere emitir un informe que incluya un listado de las especialidades incluyendo el ID de la especialidad, promedio de sueldos de los médicos de la especialidad y los médicos que pertenecen a esa especialidad cuyo sueldo sea superior al promedio de sueldos de la especialidad. Para cada médico se debe incluir su RUT, apellido paterno, materno y primer nombre, sueldo base y una observación que indique si la diferencia de su sueldo con el promedio es superior o igual al 50% del promedio o si es inferior.

El informe debe quedar almacenado de forma permanente en la tabla llamada INFORME\_E.

#### *Requerimiento*

**El diseño de su solución debe considerar:**

* Construir una función almacenada que, dado el ID de la especialidad, retorne el promedio de sueldos de los médicos de esa especialidad.
* Construir un procedimiento almacenado que permita generar el informe considerando a todas las especialidades. Este procedimiento debe truncar el contenido de la tabla que almacena el informe.

#### *Solución*

Un ejemplo de ejecución del procedimiento es el que se muestra en la Figura 8:

Figura 8: Vista (parcial) del informe

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

Nota: Vista (parcial) del informe. Captura de pantalla, (2022). SQL Developer.

Para lograr cumplir con el requerimiento se debe agregar la tabla que aparece en la Figura 9 :

Figura 9: DDL de la tabla para guardar el informe

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Nota: DDL de la tabla para guardar el informe. Captura de pantalla, (2022). SQL Developer.

Ahora construimos las unidades de programación solicitadas.

Figura 10: Código de la función almacenada.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Nota: Código de la función almacenada. Captura de pantalla, (2022). SQL Developer.

#### Desafío

#### La propuesta de solución utiliza una subconsulta para obtener el resultado pedido. ¿Puede usted, con sus conocimientos del módulo anterior (Consulta de base de datos), generar una versión alternativa de esta función?



La propuesta para el procedimiento es la que se indica en la Figura 11 :

Figura 11: Propuesta de implementación

Escala de tiempo

Descripción generada automáticamente

Nota: Propuesta de implementación del procedimiento. Captura de pantalla, (2022). SQL Developer.

Analicemos el código:

* El procedimiento no tiene parámetros
* Define 2 cursores explícitos, uno con parámetros y otro sin parámetros
* La línea 27 muestra una sentencia que permite ejecutar una sentencia del tipo DDL que trunca el contenido de la tabla de acuerdo con lo que se indica en el requerimiento.
* Las líneas 37 a la 39 contienen la sentencia DML que permite insertar un registro en la tabla donde debe quedar el informe.
* Finalmente, la línea 41 considera el caso en el cual la especialidad NO registra médicos por lo que su promedio es cero y se verifica esa situación y en ese caso se inserta de acuerdo con lo que aparece en la Figura 8 .

#### Desafío

#### Intente obtener una solución alternativa considerando, quizás, el uso de otros cursores y “desarmando” los que se entregan en esta propuesta.



# **Cierre de la semana**

Durante esta semana has logrado avanzar en el uso de cursores explícitos; en esta oportunidad la sentencia SELECT es algo más elaborada y te diste cuenta de que un cursor puede recibir parámetros los que, normalmente, se usan como filtros de los registros que se deben considerar.

Hasta el momento tus unidades de programación tendrán la capacidad de recuperar muchos registros y procesarlos; lo logran a través del uso de un cursor explícito: estructura que se define en función de una sentencia SELECT y que se comporta como una variable más dentro de la unidad de programación (función o procedimiento) y que necesita ser tratada a través del uso de un ciclo repetitivo para poder acceder uno a uno a cada registro que se almacena en el cursor.

Finalmente, aprendiste que las unidades de programación igual pueden ejecutar sentencias DML sobre las tablas de la base de datos.

Con esto ya estás mejorando tus habilidades para construir soluciones modulares que permitan dar respuesta a ciertos requerimientos planteados.

# **Referencias**

Jayapalan, L., Morin, L., Alpern, D., Belden, E., Agrawal, S., Baer, H., … Yang, M. (n.d.). Oracle database database PL/SQL language reference, 18c. Recuperado el 30 de noviembre, 2022 desde: <https://docs.oracle.com/en/database/oracle/oracle-database/18/lnpls/database-pl-sql-language-reference.pdf>

Murray, C., Belden, E., Engsig, B., Greenberg, N., Huey, P., Jones, C., … Moore, S. (n.d.). Oracle database 2 day developer’s guide, 21. Recuperado el 30 de noviembre, 2022 desde: <https://docs.oracle.com/en/database/oracle/oracle-database/21/tdddg/2-day-developers-guide.pdf>

# **Bibliografía de la semana**

Designing PL / SQL Code.

**Fuente:** Gupta, S. (2016). Advanced oracle PL/SQL developer’s guide. Segunda edición. Packt Publishing*.* Páginas:75 - 90.

Chapter - Control Structures.

**Fuente:** McLaughlin, M. (2008). Oracle Database 11g PL/SQL Programming. McGraw-Hill*.* Páginas: 91 - 129.

Chapter - Functions and Procedures.

**Fuente:** McLaughlin, M. (2008). Oracle Database 11g PL/SQL Programming. McGraw-Hill*.* Páginas: 163 - 205.

Dibujo de la tierra

Descripción generada automáticamente con confianza baja

# **Apuntes**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Facilitador disciplinar:*** Jazna Meza  
***Asesor par:*** Ricardo Pino

Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.