Base De Datos Actividad 1

Universidad De Cartagena

Adrian Elias Jadid Navarro

Jhon Arrieta

Cartagena de Indias 16/06/2023

Introduccion:

En el mundo de la gestión de bases de datos, Oracle SQL es una herramienta ampliamente utilizada y poderosa. Una de las características fundamentales de Oracle SQL es su capacidad para crear y utilizar funciones y procedimientos personalizados, que permiten ampliar la funcionalidad y adaptarla a las necesidades específicas de los usuarios.

En este trabajo, exploraremos las diferentes facetas de las funciones y procedimientos en Oracle SQL. Comenzamos por entender la diferencia entre una función y un procedimiento. Mientras que una función es una unidad de código reutilizable que devuelve un valor, un procedimiento es un conjunto de instrucciones que realiza una serie de tareas sin necesidad de devolver un resultado.

Posteriormente, nos adentramos en el mundo de las funciones personalizadas o definidas por el usuario en Oracle SQL. Estas funciones son creadas por los usuarios para abordar tareas específicas y ofrecen la posibilidad de encapsular lógica compleja o repetitiva en una única unidad de código. Su objetivo principal es facilitar la reutilización de código, simplificar consultas, mejorar la eficiencia y permitir la personalización de la funcionalidad.

Exploramos también algunas de las funciones predefinidas en Oracle SQL que nos permiten realizar operaciones sobre cadenas, cifrado y descifrado de datos, fechas, matemáticas, conversiones entre tipos de datos, manejo y obtención de información de errores, así como obtener información sobre el motor de base de datos.

Además, hemos discutido la sintaxis general para crear e invocar una función definida por el usuario en Oracle SQL. Esta sintaxis incluye la creación de la función con sus parámetros, tipo de dato de retorno, declaración de variables, lógica de la función y valor de retorno. También hemos aprendido cómo invocar una función en una consulta SQL utilizando la cláusula SELECT y el pseudo-tabla dual.

Objetivos:

El trabajo tiene como objetivo principal explorar y comprender las funciones y procedimientos en Oracle SQL, así como su utilidad y aplicaciones prácticas. Basándonos en la información proporcionada en este chat, los objetivos específicos del trabajo son los siguientes:

Comprender la diferencia entre una función y un procedimiento en Oracle SQL: Se busca tener claridad sobre las características y propósitos de cada uno de estos elementos fundamentales en el lenguaje de programación de Oracle SQL.

Explorar las funciones personalizadas o definidas por el usuario: Se pretende entender el proceso de creación de funciones personalizadas, incluyendo los parámetros de entrada, el tipo de valor de retorno y la lógica interna. Se busca comprender su importancia en la reutilización de código y la mejora de la eficiencia en la gestión de bases de datos.

Conocer las funciones predefinidas en Oracle SQL: Se desea obtener un conocimiento básico de las funciones predefinidas proporcionadas por Oracle SQL para realizar operaciones sobre cadenas, cifrado y descifrado de datos, fechas, matemáticas y conversiones entre tipos de datos.

Familiarizarse con la sintaxis general para crear e invocar una función definida por el usuario: Se busca entender la estructura y los elementos clave necesarios para crear una función personalizada en Oracle SQL, así como la forma correcta de invocarla en consultas y expresiones.

Comprender el uso de las funciones y procedimientos para obtener información sobre el motor de base de datos: Se pretende explorar las funciones predefinidas que permiten obtener información relevante sobre el motor de base de datos Oracle, como la versión, el nombre de la instancia, el nombre de la base de datos, entre otros detalles.

Justificacion:

La realización de este trabajo es justificada por varias razones importantes:

Relevancia y aplicabilidad: Las funciones y procedimientos son elementos fundamentales en Oracle SQL, utilizados ampliamente en el desarrollo y gestión de bases de datos. Comprender su funcionamiento, diferencias y aplicaciones prácticas es de gran relevancia para cualquier profesional que trabaje con Oracle SQL.

Mejora de habilidades técnicas: Al explorar las funciones personalizadas, predefinidas y la sintaxis de creación e invocación de funciones en Oracle SQL, se adquieren habilidades técnicas valiosas que pueden ser aplicadas en diversos proyectos y situaciones laborales.

Eficiencia y reutilización de código: Las funciones personalizadas permiten encapsular lógica compleja y repetitiva en una unidad de código reutilizable. Esto mejora la eficiencia y legibilidad del código, al tiempo que reduce la duplicación de código y facilita el mantenimiento de la base de datos.

Personalización de la funcionalidad: Las funciones personalizadas brindan la capacidad de personalizar y extender la funcionalidad de Oracle SQL según las necesidades específicas del usuario. Esto es especialmente útil cuando las funciones predefinidas no cubren todos los requisitos del proyecto.

Gestión de errores y obtención de información: La comprensión de las funciones predefinidas relacionadas con el manejo y obtención de información de errores, así como la obtención de información sobre el motor de base de datos, es esencial para el monitoreo y diagnóstico de problemas en la base de datos Oracle.

Potencial de optimización: Al conocer las funciones predefinidas de Oracle SQL para operaciones con cadenas, cifrado y descifrado de datos, fechas, matemáticas y conversiones de datos, se obtiene el conocimiento necesario para optimizar consultas y operaciones, mejorando así el rendimiento y la eficiencia de la base de datos.

Desarrollo:

Para el desarrollo de esta actividad académica, usted como estudiante del programa de Ingeniería de Software en **modalidad a distancia** de la Universidad de Cartagena debe desarrollar, presentar y sustentar en tutoría los siguientes puntos, de acuerdo a lo estipulado en tutorías sobre lo concerniente a las actividades académicas evaluativas, a continuación se describe el desarrollo de la primera actividad, donde ustedes como equipo deben tomar como base el mismo ejercicio desarrollado en B-1 (el caso práctico elegido para desarrollar las actividades propuestas en la asignatura BD-1), continuar con el desarrollo de la temática de la asignatura BD-2, por lo que para esta primera actividad (Unidad 1) deben resolver los siguiente puntos sobre Funciones, Procedimientos Almacenados y su aplicación en los motores de bases de datos relacionales OracleSQL, PostgreSQL y SQLServer.

1. Qué es y cual es el propósito de una Función en SQL

R/: El propósito principal de las funciones en SQL es permitir la reutilización de código y simplificar las consultas y operaciones en la base de datos. Al utilizar funciones, puedes encapsular lógica y operaciones comunes en un objeto que puede ser invocado desde diferentes partes de una consulta o de una instrucción SQL. Esto ayuda a mejorar la legibilidad del código y reduce la duplicación de código.

Algunos propósitos comunes de las funciones en SQL son:

Cálculos y transformaciones de datos: Las funciones pueden realizar cálculos matemáticos, conversiones de tipos de datos, manipulación de cadenas de texto, redondeo de valores, extracción de porciones de texto, entre otros. Por ejemplo, la función "SUM" se utiliza para calcular la suma de valores en una columna.

Validación y comprobación de datos: Las funciones pueden realizar validaciones en los datos para garantizar su integridad. Pueden verificar si un valor cumple ciertas condiciones o si se encuentra en un rango específico. Por ejemplo, la función "ISNULL" se utiliza para verificar si un valor es nulo o no.

Consultas más complejas: Las funciones pueden utilizarse para realizar consultas más complejas y realizar operaciones agregadas en los datos. Pueden agrupar, filtrar y resumir datos para proporcionar información más significativa. Por ejemplo, la función "AVG" se utiliza para calcular el promedio de valores en una columna.

Personalización y extensibilidad: Las funciones permiten la creación de nuevas funcionalidades en SQL al definir tus propias funciones personalizadas. Puedes implementar lógica específica de negocio y utilizarla en consultas y operaciones en la base de datos.

1. Cual es y cual es el propósito de un Procedimiento almacenado en SQL

R/: En SQL, un procedimiento almacenado es un conjunto de instrucciones SQL predefinidas y agrupadas en un objeto. Estas instrucciones pueden incluir consultas, actualizaciones, inserciones, eliminaciones y otras operaciones. El propósito principal de un procedimiento almacenado es encapsular una lógica de negocio compleja en la base de datos para su reutilización y ejecución eficiente.

Algunos aspectos importantes de los procedimientos almacenados son:

Reutilización de código: Al definir un procedimiento almacenado, puedes escribir una lógica compleja una vez y luego invocarla desde diferentes partes de una aplicación o sistema. Esto ayuda a evitar la duplicación de código y a mantener una base de código más limpia y organizada.

Mejora del rendimiento: Los procedimientos almacenados se compilan y almacenan en la base de datos, lo que permite un acceso más rápido y eficiente a la lógica y a los datos. Al ejecutarse repetidamente, los procedimientos almacenados pueden ofrecer un rendimiento superior en comparación con las consultas ad hoc.

Seguridad y control de acceso: Los procedimientos almacenados pueden ayudar a garantizar la seguridad de los datos y controlar el acceso a ellos. Al utilizar procedimientos almacenados, se pueden definir permisos y roles específicos para limitar el acceso a la funcionalidad y a los datos subyacentes.

Modularidad y mantenibilidad: Al encapsular la lógica de negocio en procedimientos almacenados, se mejora la modularidad del sistema. Esto facilita el mantenimiento y la evolución del código, ya que los cambios y las actualizaciones se pueden realizar en un único lugar.

Transacciones y consistencia de datos: Los procedimientos almacenados se pueden utilizar dentro de transacciones, lo que garantiza la consistencia de los datos y la integridad referencial. Puedes realizar múltiples operaciones dentro de un procedimiento almacenado y confirmar o revertir la transacción completa.

1. Cuáles son las diferencias más representativas entre Función y Procedimiento almacenado.

R/: Valor de retorno: Una función siempre devuelve un valor como resultado, mientras que un procedimiento almacenado puede no devolver ningún valor o puede devolver múltiples valores utilizando parámetros de salida.

Uso en consultas: Una función se puede utilizar en una consulta como parte de una expresión o en la cláusula SELECT, WHERE o JOIN, por ejemplo. En cambio, un procedimiento almacenado no se puede utilizar directamente en una consulta, ya que su ejecución es independiente y no devuelve un conjunto de resultados.

Invocación: Una función se invoca como parte de una expresión, asignando su resultado a una variable o utilizando su valor directamente en una operación. Un procedimiento almacenado se invoca mediante una instrucción EXECUTE o CALL, y puede requerir la asignación de valores a sus parámetros de entrada.

Modificación de datos: Las funciones en SQL están diseñadas principalmente para realizar cálculos y transformaciones en los datos, pero no pueden modificar directamente el estado de la base de datos. Por otro lado, los procedimientos almacenados tienen la capacidad de realizar operaciones de modificación de datos, como inserciones, actualizaciones y eliminaciones.

Tratamiento de excepciones: En una función, solo se pueden manejar excepciones de forma limitada utilizando la cláusula TRY-CATCH. En cambio, en un procedimiento almacenado, se pueden implementar manejadores de excepciones más complejos y personalizados para controlar errores y realizar acciones específicas en caso de excepciones.

Transacciones: Los procedimientos almacenados pueden involucrar transacciones, lo que permite agrupar un conjunto de operaciones dentro de una transacción única, lo que garantiza la consistencia y la integridad de los datos. Las funciones, en general, no pueden iniciar ni confirmar transacciones.

1. Investigue cuales son las funciones predefinidas en OracleSQL, PostgreSQL y SQLServer que nos permiten realizar operaciones sobre:
   1. Cadenas
   2. Cifrado y descifrado de datos.
   3. Fechas
   4. Matemáticas
   5. Conversiones entre tipos de datos
   6. Manejo y obtener información de errores
   7. Obtener información sobre el Motor de BD
   8. entre otras

Debe mostrar el resultado de la investigación diligenciando el siguiente formato, por ejemplo:

|  |  |  |
| --- | --- | --- |
| NOMBRE FUNCIÓN | ACCIÓN | EJEMPLO |
| **CADENAS** | **CADENAS** | **CADENAS** |
| CONCAT | Combina dos o más cadenas de texto en una sola. | CONCAT('Hola', ' ', 'Mundo') devuelve 'Hola Mundo'. |
| SUBSTR | Devuelve una subcadena de una cadena más larga. Se especifica la cadena original, la posición de inicio y la longitud deseada. | SUBSTR('Hello World', 7, 5) devuelve 'World'. |
| LENGTH: | Devuelve la longitud de una cadena. | LENGTH('Hello') devuelve 5. |
| UPPER | Convierte una cadena a mayúsculas. | UPPER('hello') devuelve 'HELLO'. |
| LOWER | Convierte una cadena a minúsculas. | LOWER('WORLD') devuelve 'world'. |
| REPLACE | Reemplaza una subcadena dentro de una cadena por otra subcadena. Se especifica la cadena original, la subcadena a reemplazar y la subcadena de reemplazo. | REPLACE('Hello World', 'Hello', 'Hola') devuelve 'Hola World'. |
| TRIM | Elimina los espacios en blanco al principio y al final de una cadena. | TRIM(' Hola ') devuelve 'Hola'. |
| INSTR | Busca la posición de una subcadena dentro de una cadena. Se especifica la cadena original y la subcadena a buscar. | INSTR('Hello World', 'World') devuelve 7. |
| **Cifrado y descifrado de datos.** | **Cifrado y descifrado de datos.** | **Cifrado y descifrado de datos.** |
| DBMS\_CRYPTO.ENCRYPT | Esta función se utiliza para cifrar datos utilizando algoritmos de cifrado simétrico. Se especifica la clave de cifrado, el tipo de algoritmo y los datos a cifrar. | DBMS\_CRYPTO.ENCRYPT('Hello', 1, 'my\_key') devuelve el resultado cifrado. |
| DBMS\_CRYPTO.DECRYPT | Esta función se utiliza para descifrar datos previamente cifrados utilizando algoritmos de cifrado simétrico. Se especifica la clave de cifrado, el tipo de algoritmo y los datos cifrados. | DBMS\_CRYPTO.DECRYPT(ciphertext, 1, 'my\_key') devuelve el resultado descifrado. |
| DBMS\_CRYPTO.HASH | Esta función se utiliza para generar un resumen hash (digest) de una cadena de texto o datos. Se especifica el tipo de algoritmo hash y los datos de entrada | DBMS\_CRYPTO.HASH('Hello', 2) devuelve el resumen hash. |
| DBMS\_CRYPTO.HMAC | Esta función se utiliza para generar un código de autenticación de mensajes (HMAC) de una cadena de texto o datos utilizando una clave secreta. Se especifica el tipo de algoritmo HMAC, la clave secreta y los datos de entrada. | DBMS\_CRYPTO.HMAC('Hello', 3, 'my\_secret\_key') devuelve el código HMAC. |
| **FECHAS** | **FECHAS** | **FECHAS** |
| SYSDATE | Devuelve la fecha y hora actual del sistema. |  |
| TO\_DATE | Convierte una cadena de texto en formato específico a un valor de fecha. | TO\_DATE('2023-06-17', 'YYYY-MM-DD') devuelve la fecha correspondiente. |
| TO\_CHAR | Convierte un valor de fecha en una cadena de texto con un formato específico. | TO\_CHAR(SYSDATE, 'DD-MON-YYYY') devuelve la fecha en formato '17-JUN-2023'. |
| EXTRACT | Extrae una parte específica de una fecha, como el año, mes, día, hora, minuto, etc. | EXTRACT(YEAR FROM SYSDATE) devuelve el año actual. |
| ADD\_MONTHS | Agrega un número específico de meses a una fecha. | ADD\_MONTHS(SYSDATE, 3) devuelve la fecha que resulta de sumar 3 meses a la fecha actual. |
| MONTHS\_BETWEEN | Calcula el número de meses entre dos fechas. | MONTHS\_BETWEEN(SYSDATE, TO\_DATE('2023-01-01', 'YYYY-MM-DD')) devuelve la cantidad de meses entre la fecha actual y el 1 de enero de 2023. |
| NEXT\_DAY | Devuelve la fecha del próximo día de la semana especificado después de una fecha dada. | NEXT\_DAY(SYSDATE, 'SUNDAY') devuelve la próxima fecha de domingo después de la fecha actual. |
| LAST\_DAY | Devuelve la última fecha del mes para una fecha dada. | LAST\_DAY(SYSDATE) devuelve la última fecha del mes actual. |
| **Matemáticas** | **Matemáticas** | **Matemáticas** |
| ABS | Devuelve el valor absoluto de un número. | ABS(-5) devuelve 5. |
| ROUND | Redondea un número al número de decimales especificado. | ROUND(3.14159, 2) devuelve 3.14. |
| CEIL | Devuelve el entero más pequeño mayor o igual que un número. | CEIL(4.2) devuelve 5. |
| FLOOR | FLOOR(4.8) devuelve 4. | FLOOR(4.8) devuelve 4. |
| POWER | Eleva un número a una potencia especificada. | POWER(2, 3) devuelve 8 (2 elevado a la potencia de 3). |
| SQRT | Devuelve la raíz cuadrada de un número. | SQRT(25) devuelve 5. |
| MOD | Devuelve el resto de la división de un número por otro. | MOD(10, 3) devuelve 1 (10 dividido por 3 tiene un resto de 1). |
| EXP | Calcula el valor de e (número de Euler) elevado a la potencia especificada. | EXP(1) devuelve el valor aproximado de 2.71828. |
| LOG | Calcula el logaritmo natural (base e) de un número. | LOG(10) devuelve el valor aproximado de 2.30259. |
| SIN, COS, TAN | Calculan el seno, coseno y tangente de un ángulo, respectivamente. | SIN(0) devuelve 0, COS(0) devuelve 1 y TAN(0) devuelve 0. |
| **Conversiones entre tipos de datos** | **Conversiones entre tipos de datos** | **Conversiones entre tipos de datos** |
| TO\_CHAR | Convierte un valor numérico, fecha o expresión en una cadena de caracteres. | TO\_CHAR(123.45) devuelve '123.45'. |
| TO\_NUMBER | Convierte una cadena de caracteres en un valor numérico. | TO\_NUMBER('123.45') devuelve el número 123.45. |
| TO\_DATE | Convierte una cadena de caracteres en un valor de fecha. Se especifica la cadena de caracteres y un formato de fecha opcional. | TO\_DATE('2023-06-17', 'YYYY-MM-DD') devuelve la fecha correspondiente. |
| TO\_TIMESTAMP | Convierte una cadena de caracteres en un valor de marca de tiempo. Se especifica la cadena de caracteres y un formato de marca de tiempo opcional. | TO\_TIMESTAMP('2023-06-17 12:34:56', 'YYYY-MM-DD HH24:MI:SS') devuelve el valor de marca de tiempo correspondiente. |
| CAST | Realiza una conversión explícita de un tipo de dato a otro. | CAST('123' AS NUMBER) convierte la cadena '123' en un valor numérico. |
| DECODE | Realiza una conversión condicional basada en un valor de entrada. Se especifican pares de valor-resultado y un valor predeterminado opcional. | DECODE(status, 'A', 'Activo', 'I', 'Inactivo', 'Desconocido') devuelve el resultado correspondiente según el valor de la variable status. |
| NVL | Realiza una conversión nula. Si la expresión de entrada es nula, se devuelve un valor alternativo. | NVL(salary, 0) devuelve el salario si no es nulo, de lo contrario, devuelve 0. |
| **Manejo y obtener información de errores** | **Manejo y obtener información de errores** | **Manejo y obtener información de errores** |
| SQLCODE | Devuelve el código numérico del error más reciente. | SQLCODE devuelve -1403 si ocurre un error de "NO DATA FOUND" (no se encontraron datos). |
| SQLERRM | Devuelve el mensaje de error asociado al código de error más reciente. | SQLERRM devuelve "ORA-01403: no data found" para el error de "NO DATA FOUND". |
| SQLERRM(valor) | Devuelve el mensaje de error asociado a un código de error específico. | SQLERRM(-1403) devuelve "ORA-01403: no data found". |
| DBMS\_UTILITY.FORMAT\_ERROR\_BACKTRACE | Devuelve una cadena que muestra la pila de llamadas (stack trace) cuando se produce un error. Puede ser útil para depurar y rastrear el origen de un error. |  |
| DBMS\_UTILITY.FORMAT\_CALL\_STACK | Devuelve una cadena que muestra la pila de llamadas actual. Puede ser útil para identificar el camino de ejecución que condujo a un punto específico del código. |  |
| DBMS\_UTILITY.FORMAT\_ERROR\_STACK | Devuelve una cadena que muestra la pila de errores (error stack) más reciente. Puede ser útil para analizar errores en un entorno de manejo de excepciones. |  |
| **Obtener información sobre el Motor de BD** | **Obtener información sobre el Motor de BD** | **Obtener información sobre el Motor de BD** |
| VERSION | Devuelve la versión del motor de base de datos Oracle instalado. | SELECT VERSION FROM V$INSTANCE devuelve la versión de Oracle. |
| INSTANCE\_NAME | Devuelve el nombre de la instancia de base de datos actual. | SELECT INSTANCE\_NAME FROM V$INSTANCE devuelve el nombre de la instancia. |
| DB\_NAME | Devuelve el nombre de la base de datos actual. | SELECT DB\_NAME FROM V$DATABASE devuelve el nombre de la base de datos. |
| HOST\_NAME | Devuelve el nombre del host en el que se ejecuta la instancia de la base de datos. | SELECT HOST\_NAME FROM V$INSTANCE devuelve el nombre del host. |
| SYS\_CONTEXT | Permite obtener información contextual específica del sistema, como el nombre de usuario actual, el nombre de la instancia, el nombre de la base de datos, etc. | SELECT SYS\_CONTEXT('USERENV', 'SESSION\_USER') FROM DUAL devuelve el nombre de usuario de la sesión actual. |
| DATABASE\_ROLE | Devuelve el rol de la base de datos, como PRIMARY o PHYSICAL STANDBY. | SELECT DATABASE\_ROLE FROM V$DATABASE devuelve el rol de la base de datos. |
| DBMS\_DB\_VERSION | Es un paquete de procedimientos y funciones que proporciona información sobre la versión de la base de datos, como el número de versión, la revisión, la actualización, etc. | DBMS\_DB\_VERSION.VERSION devuelve el número de versión de la base de datos. |

Qué es y cuál el el objetivo de una Función personalizada o definida por el usuario

R/: En Oracle SQL, una función personalizada o definida por el usuario es una función creada por el usuario para realizar una tarea específica dentro de una base de datos Oracle. A diferencia de las funciones predefinidas en Oracle, que ya vienen incorporadas en el sistema, las funciones personalizadas se crean para adaptarse a las necesidades específicas del usuario.

El objetivo principal de una función personalizada es proporcionar una forma conveniente de encapsular una lógica compleja o repetitiva en una sola unidad de código reutilizable. Al crear una función personalizada, el usuario puede definir los parámetros de entrada, el tipo de valor de retorno y la lógica interna que debe ejecutarse cuando se invoca la función.

Algunos objetivos comunes de las funciones personalizadas en Oracle SQL incluyen:

Reutilización de código: Al encapsular una lógica de programación en una función personalizada, se puede utilizar en múltiples partes del código de la base de datos. Esto evita la duplicación de código y facilita el mantenimiento.

Simplificación de consultas: Las funciones personalizadas pueden utilizarse en consultas SQL para realizar cálculos complejos o transformaciones de datos. Esto permite simplificar las consultas y mejorar la legibilidad del código.

Mejora de la eficiencia: Al crear funciones personalizadas, se puede optimizar la lógica interna para mejorar el rendimiento de las operaciones de la base de datos. Esto puede incluir el uso de índices, operaciones eficientes de filtrado o cálculos optimizados.

Personalización de la funcionalidad: Las funciones personalizadas permiten al usuario agregar funcionalidades adicionales a la base de datos que no están disponibles en las funciones predefinidas. Esto permite adaptar la base de datos a los requisitos específicos del usuario y extender su capacidad.

Cual es la sintaxis general para crear e invocar una Función definida por el usuario.

R/: La sintaxis general para crear e invocar una función definida por el usuario en Oracle SQL es la siguiente:

Creación de la función: CREATE [OR REPLACE] FUNCTION nombre\_funcion

(parámetro1 tipo\_dato, parámetro2 tipo\_dato, ...)

RETURN tipo\_dato\_retorno

IS

declaración\_variables;

BEGIN

lógica\_de\_la\_función;

RETURN valor\_de\_retorno;

END;

nombre\_funcion es el nombre que se le da a la función personalizada.

parámetro1, parámetro2, ... son los parámetros de entrada de la función, especificando el tipo de dato de cada uno.

tipo\_dato\_retorno es el tipo de dato que la función devuelve como resultado.

IS es la palabra clave para iniciar la sección declarativa de la función.

declaración\_variables es opcional y se utiliza para declarar variables locales que se utilizarán en la lógica de la función.

BEGIN es la palabra clave para iniciar la sección ejecutable de la función.

lógica\_de\_la\_función es donde se escribe la lógica y las operaciones que realizará la función.

RETURN se utiliza para devolver un valor como resultado de la función.

valor\_de\_retorno es el valor que se devuelve como resultado de la función.

Invocación de la función:

Una vez creada la función, se puede invocar en una consulta SQL u otra expresión utilizando la siguiente sintaxis:

SELECT nombre\_funcion(argumento1, argumento2, ...)

FROM dual;

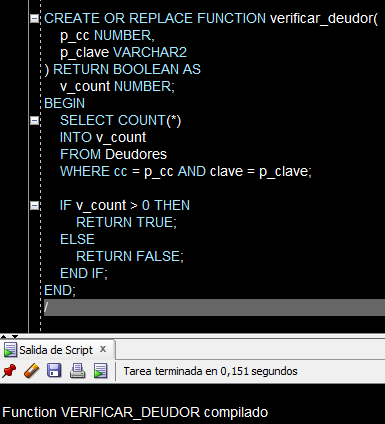
nombre\_funcion es el nombre de la función que se desea invocar.

argumento1, argumento2, ... son los valores de los argumentos que se pasan a la función.

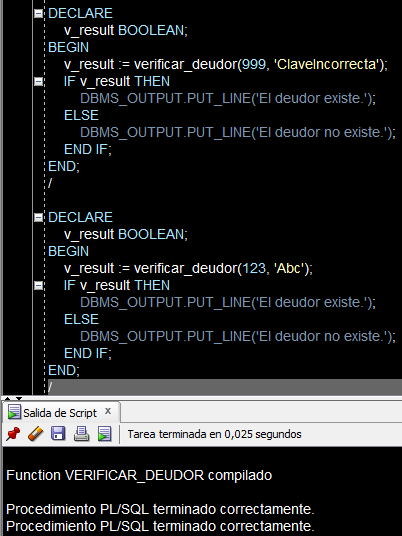
Es importante destacar que el FROM dual se utiliza porque Oracle SQL requiere que toda consulta tenga una cláusula FROM, y el pseudo-tabla dual se usa comúnmente para consultas que no necesitan acceder a una tabla real.

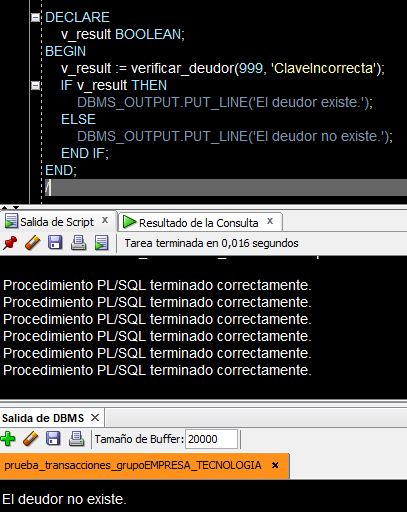
Usar la base de datos prueba\_transacciones\_grupoXYZ creada en la actividad de la Unidad 2 (punto 5 al punto 11), entonces deben hacer los siguiente:

1. Crear una función que reciba como parámetros de entrada la cc y la clave de un deudor, entonces la función debe retornar Verdadero si existe un usuario con esa misma CC y Clave registrados en esa tabla, o retornar FALSO en caso contrario.

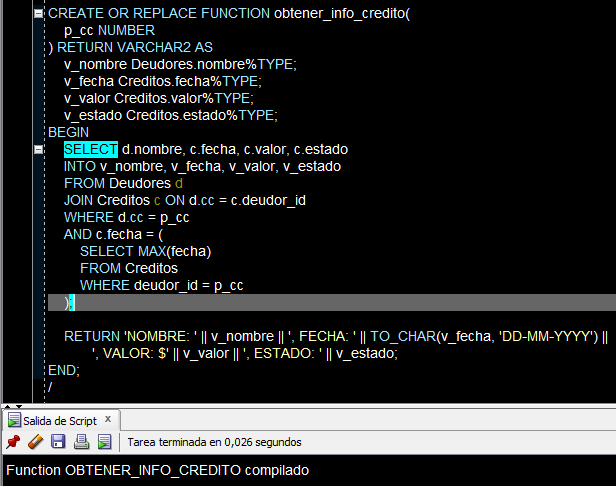


1. Invocar y probar la función con datos que permitan mostrar resultado Falso y otros que permitan retornar resultado Verdadero

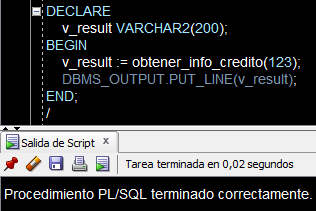


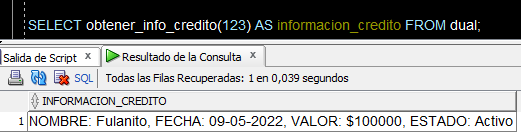


1. Crear una función que reciba como parámetros la cédula de un deudor y devuelva un solo dato con el nombre del deudor, valor del último, fecha y estado del último crédito que tiene en el sistema. El resultado debe ser una cadena de texto en el siguiente formato: por ejemplo. NOMBRE: FULANITO DE TAL, FECHA: 12-04-2022, VALOR: $100000, ESTADO: ACTIVO

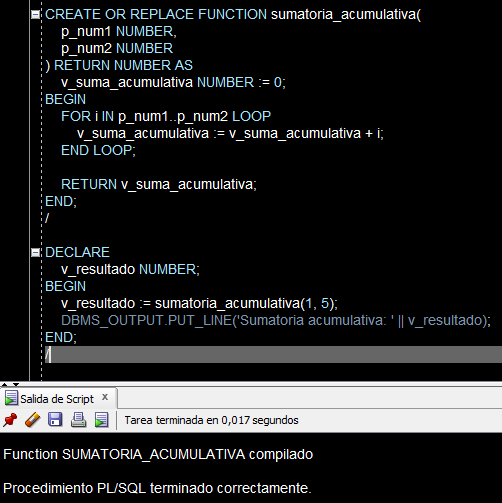


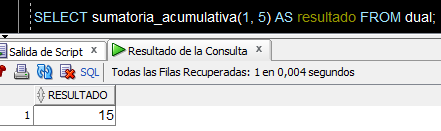
1. Invocar y probar la función



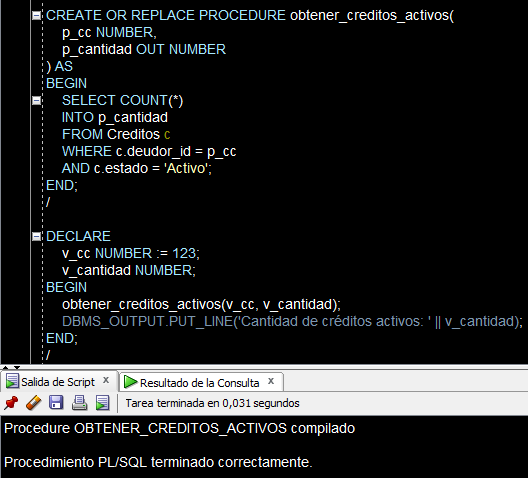


1. Crear una que reciba 2 números y revuelva la sumatoria acumulativa entre ellos.

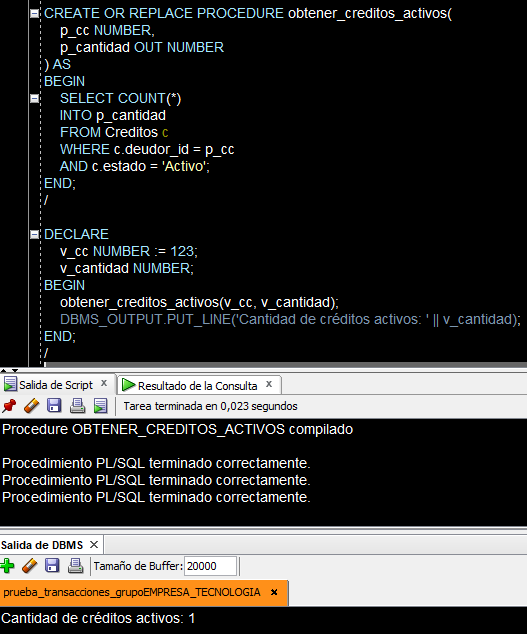




1. Crear un procedimiento almacenado con las siguientes características:
   1. debe tener un parámetro de **entrada** que reciba el código de un deudor
   2. debe tener un parámetro de **salida** que devuelva la cantidad (cuantos) de créditos Activos que tiene ese deudor



1. Invocar y probar el procedimiento almacenado.



ENLACE GIT HUB: https://github.com/AEJN/UDC-BD2-actividad-3-2023.git

Bibliografia:

Oracle Database SQL Language Reference: Este es el manual de referencia oficial proporcionado por Oracle, que contiene información completa y detallada sobre el lenguaje SQL en Oracle, incluyendo funciones y procedimientos. Puedes encontrar la documentación actualizada en el sitio web oficial de Oracle.

Oracle PL/SQL Programming: Este libro escrito por Steven Feuerstein es una guía completa sobre programación en PL/SQL, el lenguaje de programación de Oracle. Ofrece explicaciones detalladas sobre funciones, procedimientos y otros aspectos del desarrollo en Oracle SQL.

Oracle PL/SQL Best Practices: Este libro, escrito por Steven Feuerstein y Bill Pribyl, proporciona consejos y prácticas recomendadas para desarrollar aplicaciones eficientes y de alta calidad en Oracle PL/SQL. Incluye información relevante sobre el uso de funciones y procedimientos.

Oracle Magazine: Oracle Magazine es una revista oficial publicada por Oracle que cubre una amplia gama de temas relacionados con la tecnología de Oracle, incluyendo bases de datos, desarrollo de aplicaciones y SQL. Puedes encontrar artículos y tutoriales sobre funciones y procedimientos en Oracle SQL en los archivos de la revista.

Blogs y comunidades en línea: Hay numerosos blogs y comunidades en línea dedicados a Oracle SQL y PL/SQL, donde los expertos comparten sus conocimientos y experiencias. Algunos blogs populares incluyen AskTom, Oracle Developers y Oracle Base. También puedes participar en foros como Oracle Community o Stack Overflow para buscar respuestas a tus preguntas y encontrar información relevante.