**Escuela Colombiana de Ingeniería Julio Garavito**

**Carrera / Semestre:** Ingeniería de Sistemas / 5to Semestre

**Asignatura:** Modelos y Bases de Datos (MBDA)

**Nombre:** Andrés Chavarro, Jesús Pinzón

**Fecha:** 2024/03/21

# Autoestudio 4 - PL/SQL Básico

## INVESTIGACIÓN

1. **Acciones Referenciales**
2. ¿Para qué sirven las acciones referenciales?

**R/** En el contexto de las bases de datos, las **acciones referenciales** (también conocidas como acciones de integridad referencial o restricciones de clave externa) son **reglas que se aplican a las relaciones entre tablas** para mantener la coherencia de los datos. Estas acciones se utilizan para garantizar que las relaciones entre las tablas se mantengan válidas y que no se produzcan operaciones que puedan dejar los datos en un estado inconsistente.

1. ¿Qué acciones soporta ORACLE? ¿Qué permite hacer cada una de ellas?

**R/** Oracle Database soporta varias acciones referenciales que se pueden especificar al definir restricciones de clave externa (FKs). A continuación, se detallan las acciones referenciales admitidas por Oracle y lo que permite hacer cada una de ellas:

* **CASCADE:** Cuando se realiza una operación de actualización o eliminación en la tabla principal, Oracle automáticamente aplica la misma operación en las tablas secundarias relacionadas. Por ejemplo, si se actualiza o elimina una fila en la tabla principal, todas las filas relacionadas en las tablas secundarias también se actualizarán o eliminarán.
* **SET NULL:** Si se actualiza o elimina una fila en la tabla principal, Oracle establecerá los valores de los campos en las tablas secundarias relacionadas como **NULL**. Esto significa que la fila relacionada en la tabla secundaria no apuntará a ninguna fila en la tabla principal después de la operación.
* **NO ACTION:** Esta acción restringe la operación de actualización o eliminación si hay filas relacionadas en la tabla secundaria. Si se intenta actualizar o eliminar una fila en la tabla principal y hay filas relacionadas en las tablas secundarias, Oracle no permitirá que se realice la operación en la fila de la tabla principal.
* **SET DEFAULT:** Similar a **SET NULL**, esta acción establece los valores de los campos en las tablas secundarias relacionadas con la fila que se está actualizando o eliminando en la tabla principal en un valor predeterminado especificado.
* **RESTRICT:** Al igual que **NO ACTION**, esta acción restringe la operación de actualización o eliminación si hay filas relacionadas en la tabla secundaria. Oracle no permite que se realice la operación en la fila de la tabla principal si hay filas relacionadas en las tablas secundarias.

Estas acciones referenciales permiten definir el comportamiento deseado cuando se realizan operaciones de actualización o eliminación en tablas relacionadas, garantizando la integridad referencial en la base de datos Oracle.

1. **PL/SQL**
2. ¿Qué es PL/SQL?

**R/ PL/SQL** (***P****rocedural* ***L****anguage /* ***S****tructured* ***Q****uery* ***L****anguage*) es un **lenguaje de programación procedimental** y **extensión del lenguaje** **SQL** utilizado en la base de datos Oracle. Combina los aspectos de la programación procedural con las capacidades de consulta y manipulación de datos del SQL estándar. **PL/SQL** se utiliza principalmente para escribir procedimientos almacenados, funciones, disparadores (triggers) y paquetes que se ejecutan en el servidor de la base de datos Oracle. Estos objetos pueden realizar diversas operaciones, como realizar cálculos complejos, manipular datos en la base de datos, controlar el flujo de ejecución, interactuar con otros objetos de la base de datos y ejecutar lógica empresarial compleja.

1. ¿Qué motores lo soportan?

**R/** PL/SQL es un lenguaje de programación específico de Oracle y está integrado directamente en el motor de base de datos Oracle Database. Por lo tanto, **PL/SQL es compatible únicamente con el motor de base de datos Oracle Database.**

Otros motores de bases de datos, como MySQL, PostgreSQL, SQL Server, SQLite, entre otros, tienen sus propias extensiones de procedimientos almacenados y lenguajes de programación específicos, pero no son compatibles directamente con PL/SQL. Cada uno de estos motores de bases de datos tiene su propio lenguaje de procedimientos almacenados (como T-SQL en SQL Server, PL/pgSQL en PostgreSQL, etc.) con sintaxis y características específicas para ese motor en particular.

1. **Datos e Instrucciones en PL/SQL**
2. ¿Cuáles son los tipos de datos que ofrece?

**R/** PL/SQL ofrece una variedad de tipos de datos que pueden utilizarse para declarar variables, parámetros de procedimientos, funciones y otros elementos del código. A continuación, se enumeran los tipos de datos principales disponibles en PL/SQL:

* **Números:**
* **INTEGER:** Números enteros.
* **NUMBER (P, S):** Número de precisión P y escala S.
* **BINARY\_FLOAT:** Número de punto flotante de precisión simple.
* **BINARY\_DOUBLE:** Número de punto flotante de doble precisión.
* Caracteres y cadenas:
  + **CHAR(N):** Cadena de caracteres de longitud fija.
  + **VARCHAR2(N):** Cadena de caracteres de longitud variable.
  + **NCHAR(N):** Cadena Unicode de longitud fija.
  + **NVARCHAR2(N):** Cadena Unicode de longitud variable.
  + **CLOB:** Objeto grande de caracteres.
  + **NCLOB:** Objeto grande de caracteres Unicode.
* **Fecha y hora:**
  + **DATE:** Almacena fecha y hora.
  + **TIMESTAMP:** Almacena fecha y hora, incluidos segundos.
  + **INTERVAL YEAR TO MONTH:** Almacena intervalos de tiempo expresados en años y meses.
  + **INTERVAL DAY TO SECOND:** Almacena intervalos de tiempo expresados en días, horas, minutos y segundos.
* **Booleanos:**
  + **BOOLEAN:** Almacena valores de verdad (TRUE, FALSE, NULL).
* **Colecciones:**
  + **Arrays** (colecciones indexadas): Pueden ser de tipo varray (arrays de tamaño fijo) o nested table (tablas anidadas).
  + **Colecciones Asociativas** (también conocidas como tablas hash): Utilizan una clave para acceder a los elementos.
* **Binarios:**
  + **RAW:** Almacena datos binarios de longitud fija.
  + **BLOB:** Objeto grande binario.
* **Otros tipos de datos:**
  + **ROWID:** Identificador de fila único en una tabla.
  + **UROWID:** Identificador de fila universal.
  + **XMLType:** Almacena datos XML.

Estos son algunos de los tipos de datos más comunes que se utilizan en PL/SQL. Cada tipo de dato tiene sus propias características y restricciones, y pueden ser utilizados según las necesidades específicas de la aplicación que se está desarrollando.

1. ¿Cuál es la forma de definir constantes y variables?

**R/** En PL/SQL,las constantes y variables se definen utilizando declaraciones específicas**.** A continuación, se muestra cómo se definen las constantes y las variables en PL/SQL:

* **Definición de Constantes:** Las constantes en PL/SQL se definen utilizando la palabra clave **CONSTANT** dentro de un bloque PL/SQL o en la declaración de un objeto (como un paquete). Las constantes no pueden ser modificadas una vez que se han asignado.

Ejemplo de declaración de constante:

DECLARE

PI CONSTANT NUMBER := 3.14159;

MAX\_USERS CONSTANT INTEGER := 100;

BEGIN

-- Código PL/SQL aquí

END;

* **Definición de Variables:** Las variables en PL/SQL se definen utilizando la palabra clave **DECLARE** dentro de un bloque PL/SQL. Las variables pueden ser modificadas durante la ejecución del programa.

Ejemplo de declaración de variables:

DECLARE

nombre VARCHAR2(50);

edad NUMBER := 30;

BEGIN

-- Código PL/SQL aquí

END;

En el ejemplo anterior, nombre es una variable de tipo VARCHAR2 que puede contener una cadena de hasta 50 caracteres, mientras que edad es una variable de tipo NUMBER que se inicializa con el valor 30.

* **Algunos puntos importantes a tener en cuenta:**
* Las constantes y variables deben declararse antes de ser utilizadas en un bloque PL/SQL.
* Se puede especificar un tipo de datos explícito para las variables o dejar que PL/SQL infiera el tipo de datos en función del valor proporcionado durante la inicialización.
* Las variables se pueden inicializar con un valor específico durante la declaración, como se muestra en los ejemplos anteriores. Si no se proporciona un valor de inicialización, la variable se inicializará con NULL.
* Las constantes deben inicializarse con un valor durante la declaración y no pueden ser modificadas posteriormente en el código.

Estas son las formas básicas de definir constantes y variables en PL/SQL. Dependiendo de la situación, también hay otros detalles y técnicas avanzadas que se pueden utilizar para gestionar constantes y variables en PL/SQL.

1. ¿Cómo se define una variable con un tipo tomado de la base de datos?

**R/** En PL/SQL, puedes definir una variable utilizando un tipo de dato que se tome de la base de datos utilizando la cláusula **%TYPE**. Esto permite que la variable tenga el mismo tipo de datos que una columna específica en una tabla de la base de datos, lo que puede ser útil para mantener consistencia y evitar errores de tipo de datos.

La sintaxis para definir una variable utilizando **%TYPE** es la siguiente:

nombre\_variable tabla.columna%TYPE;

1. ¿Cuál es la forma de los diferentes tipos de asignación?

**R/** En muchos lenguajes de programación, incluido PL/SQL, existen varias formas de asignar valores a variables. Aquí están los tres tipos principales de asignación que se encuentran comúnmente:

1. **Asignación Directa o Simple:** Esta es la forma más básica de asignación, donde se asigna directamente un valor a una variable utilizando el operador de asignación (**:=**). Es útil cuando se sabe exactamente qué valor se desea asignar y no se necesita ningún cálculo adicional.

Ejemplo en PL/SQL:

v\_nombre := 'Juan';

1. **Asignación por Expresión:** En este tipo de asignación, se utiliza una expresión para calcular el valor que se asignará a la variable. La expresión puede implicar variables existentes, operaciones aritméticas, llamadas a funciones, o cualquier combinación de estas.

Ejemplo en PL/SQL:

v\_resultado := v\_numero1 + v\_numero2;

1. **Asignación por Selección o Consulta:** Esta forma de asignación implica seleccionar un valor de una consulta (**SELECT**) y asignarlo a una variable. Esto es comúnmente utilizado cuando se desea asignar un valor procedente de una tabla en la base de datos a una variable.

Ejemplo en PL/SQL:

SELECT nombre INTO v\_nombre

FROM empleados

WHERE id = 1;

Estos son los tres tipos principales de asignación que se utilizan en la mayoría de los lenguajes de programación, incluido PL/SQL. Cada uno tiene su propio propósito y casos de uso específicos, y su elección depende de los requisitos y la lógica del programa que estés escribiendo.

1. **Cursores**
2. ¿Qué es un cursor implícito? ¿Para qué sirve?

**R/** En el contexto de las bases de datos, un **"cursor implícito"** se refiere a un tipo de cursor que se utiliza de manera implícita por el sistema de gestión de bases de datos (SGBD) para **manejar el resultado de una consulta SQL**. A diferencia de los cursores explícitos, que son definidos y manejados por el programador, los cursores implícitos son manejados automáticamente por el motor de la base de datos sin necesidad de que el usuario los declare explícitamente.

Un cursor implícito generalmente se utiliza en una instrucción **SQL** que devuelve un conjunto de resultados, como una consulta **SELECT**, la cual se encarga de procesar un conjunto de datos para devolver los resultados. Este cursor implícito se mueve a través de las filas del conjunto de resultados a medida que se recuperan y procesan.

El cursor implícito proporciona una interfaz conveniente para recorrer los resultados de la consulta, ya que el SGBD se encarga de la gestión de este cursor automáticamente. El programador no necesita preocuparse por la implementación detallada del cursor; simplemente puede iterar sobre las filas devueltas por la consulta utilizando el mecanismo proporcionado por el lenguaje de programación o la herramienta de acceso a la base de datos que esté utilizando.

1. ¿Qué es un cursor explícito? ¿Para qué sirve?

**R/** Un **“cursor explícito**” es un **objeto de base de datos** que permite a los programadores **manejar y procesar filas de datos individualmente** en un conjunto de resultados devuelto por una consulta SQL. A diferencia de los cursores implícitos, que son manejados automáticamente por el sistema de gestión de bases de datos (SGBD), los cursores explícitos deben ser definidos, abiertos, recorridos y cerrados explícitamente por el programador.

Para utilizar un cursor explícito, el programador debe seguir estos pasos:

1. **Definición del Cursor**: El programador declara y define el cursor, especificando la consulta SQL que desea ejecutar.
2. **Apertura del Cursor**: Después de definir el cursor, se debe abrir explícitamente para ejecutar la consulta y recuperar los resultados.
3. **Recorrido del Cursor**: Una vez abierto, el cursor puede recorrer fila por fila para procesar los datos utilizando bucles o estructuras de control.
4. **Procesamiento de los Datos**: Durante el recorrido del cursor, el programador puede acceder a los datos de cada fila y realizar cualquier operación o cálculo necesario.
5. **Cierre del Cursor**: Después de haber terminado de procesar los datos, el cursor debe ser cerrado para liberar los recursos del sistema.
6. ¿Cuáles son las excepciones propias de uso de estos cursores?

**R/** En el contexto de PL/SQL y el manejo de cursores, existen varias excepciones que pueden ocurrir durante su uso. Estas excepciones pueden ocurrir durante la apertura, el recorrido o el cierre del cursor, así como durante el acceso a los datos dentro del cursor. Algunas de las excepciones más comunes relacionadas con el uso de cursores son:

1. **NO\_DATA\_FOUND**: Esta excepción se produce cuando se intenta recuperar una fila de un cursor, pero no se encuentra ninguna fila que cumpla con los criterios de búsqueda.
2. **TOO\_MANY\_ROWS**: Esta excepción ocurre cuando se espera recuperar una sola fila, pero el cursor devuelve más de una.
3. **CURSOR\_ALREADY\_OPEN**: Se produce cuando se intenta abrir un cursor que ya está abierto.
4. **CURSOR\_NOT\_FOUND**: Esta excepción ocurre cuando se intenta cerrar un cursor que no está abierto o no existe.
5. **INVALID\_CURSOR**: Se produce cuando se intenta operar en un cursor que no es válido, como cerrar un cursor que ya ha sido cerrado.
6. **CURSOR\_ALREADY\_OPEN**: Se produce cuando se intenta abrir un cursor que ya está abierto.
7. **CURSOR\_IS\_CLOSED**: Esta excepción ocurre cuando se intenta operar en un cursor que ya ha sido cerrado.
8. **FETCH\_OUTSIDE\_CURSOR**: Se produce cuando se intenta realizar una operación **FETCH** en un cursor que no está abierto.

Estas son algunas de las excepciones más comunes relacionadas con el manejo de cursores en PL/SQL. Es importante manejar estas excepciones adecuadamente en el código para garantizar un comportamiento robusto y evitar interrupciones no deseadas en la ejecución del programa. Esto generalmente se hace mediante bloques **TRY-CATCH** (o bloques **EXCEPTION** en PL/SQL) para capturar y manejar las excepciones de manera adecuada.

1. **Modularidad**
2. ¿Cuál es la estructura general de un bloque PL/SQL?

**R/** Un bloque PL/SQL sigue una estructura general que consiste en varias secciones. A continuación, se presenta la estructura básica de un bloque PL/SQL:

DECLARE

-- Declaraciones de variables locales

BEGIN

-- Código PL/SQL

EXCEPTION

-- Manejo de excepciones

END;

* **DECLARE:** Esta sección es opcional y se utiliza para declarar variables, cursores, tipos de datos, constantes y otros elementos que se utilizarán dentro del bloque PL/SQL. Las variables declaradas aquí son locales al bloque y no son visibles fuera de él.
* **BEGIN:** Esta sección marca el inicio del bloque PL/SQL real donde se escribe el código que realiza las operaciones deseadas. Aquí se colocan las instrucciones SQL, estructuras de control (como IF-THEN-ELSE, LOOP, etc.), llamadas a procedimientos y funciones, y cualquier otra lógica de programación necesaria.
* **EXCEPTION:** Esta sección es opcional y se utiliza para manejar excepciones que pueden ocurrir durante la ejecución del bloque PL/SQL. Aquí se pueden capturar excepciones específicas y tomar medidas para manejarlas adecuadamente, como registrar un mensaje de error, realizar operaciones de limpieza o lanzar una nueva excepción personalizada.
* **END:** Marca el final del bloque PL/SQL.

La estructura general de un bloque PL/SQL puede variar según el contexto y los requisitos específicos del programa, pero esta estructura básica proporciona un marco común para escribir código PL/SQL.

1. ¿Para qué sirven las diferentes estructuras modulares? (bloque anónimo, procedimiento, función y disparador)

**R/** Las diferentes **estructuras modulares** en PL/SQL sirven para **organizar y modularizar el código**, lo que facilita su mantenimiento, reutilización y comprensión. A continuación, se describe cada una de estas estructuras:

* **Bloque Anónimo:** Bloque de código PL/SQL que no tiene un nombre asociado y se ejecuta una sola vez cuando se llama. Se utiliza para escribir y ejecutar un conjunto de instrucciones PL/SQL de forma ad hoc, sin la necesidad de definir un procedimiento o función. Son útiles para tareas simples que no necesitan ser reutilizadas.
* **Procedimiento:** Conjunto de instrucciones PL/SQL que realiza una tarea específica. Puede aceptar parámetros de entrada y/o salida y puede ser llamado varias veces desde diferentes partes del código. Se utilizan para encapsular la lógica de negocio o funcionalidades complejas que necesitan ser ejecutadas repetidamente en diferentes partes de la aplicación.
* **Función:** Una función es similar a un procedimiento, pero devuelve un valor de resultado. Puede aceptar parámetros de entrada y/o salida y puede ser llamada desde otros bloques PL/SQL o sentencias SQL. Se utilizan para calcular y devolver valores específicos basados en los parámetros de entrada. Las funciones son útiles cuando se necesita realizar cálculos repetitivos y obtener un resultado específico.
* **Disparador (Trigger):** Bloque de código PL/SQL que se ejecuta automáticamente en respuesta a ciertos eventos en la base de datos, como inserciones, actualizaciones o eliminaciones en una tabla. Se utilizan para implementar lógica empresarial o aplicar restricciones de integridad de datos de forma automática cuando se produce un evento específico en la base de datos. Los disparadores son útiles para mantener la consistencia y la integridad de los datos en la base de datos.

En resumen, las diferentes estructuras modulares en PL/SQL proporcionan formas de organizar y encapsular la lógica de programación en bloques de código que pueden ser reutilizados, mantenidos y ejecutados de manera modular. Cada estructura tiene su propio propósito y casos de uso específicos, lo que permite a los desarrolladores escribir código eficiente y modular en la base de datos Oracle.

## PRÁCTICA

En este autoestudio vamos a implementar un caso de uso de la BD [GuestHouse](https://sqlzoo.net/wiki/Guest_House).

|  |  |
| --- | --- |
|  | **TExtra**  Entero(11) Positivo inicia en 9  **TDetail**  Cadena(50) No vacía  **TPrice**  Entero(9) Positivo |
|  | **Register an Extra (Ad,Co,El)**  **Ad**  - El id se genera automáticamente: 9AAAAMMDDmmss  - El descuento no puede ser mayor al 50% del precio. Si tiene un precio mayor a 100000 y no se indica descuento, se aplica un descuento del 10% y el primer detalle es ‘CON 10% de descuento automático’.  **Mo**  - Sólo se pueden adicionar detalles a la descripción, no modificar ni eliminar.  **Co**  - Consulta todos los datos del extra  **El**  - Si no tiene descripción, se pude eliminar |

1. **Adicionando Restricciones Declarativas**

Para proteger la BD vamos a implementar las siguientes restricciones que se pueden garantizar usando restricciones declarativas:

1. Cree la nueva tabla

(Tablas, XTablas)

Código presente en el archivo [*[auto4.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Auestudios%20•/auto4.sql)

1. Definir las restricciones declarativas

(Atributos, Tuplas)

Código presente en el archivo [*[auto4.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Auestudios%20•/auto4.sql)

1. Validar con casos significativos la protección de la BD. Adicione comentarios que expliquen la condición que están validando.

(AtributosOK, AtributosNoOK, TuplasOK, TuplasNoOK)

Código presente en el archivo [*[auto4.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Auestudios%20•/auto4.sql)

1. **Adicionando Acciones de Referencia**

Para proteger la BD vamos a implementar las siguientes acciones de referencia necesarias para el caso de uso:

1. Definir las acciones de referencia.

(Eliminar y volver a crear las FK afectadas. Justificar la decisión)

(Acciones)

Código presente en el archivo [*[auto4.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Auestudios%20•/auto4.sql)

1. Validar con casos significativos las condiciones definidas.

(AccionesOK)

Código presente en el archivo [*[auto4.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Auestudios%20•/auto4.sql)

1. **Adicionando Disparadores**

Para preparar las acciones asociadas a los diferentes casos de uso vamos a implementar los siguientes disparadores (CRUD):

1. Escriba las instrucciones necesarias para crear y eliminar el disparador

(Disparadores - XDisparadores)

Código presente en el archivo [*[auto4.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Auestudios%20•/auto4.sql)

1. Escriba 3 instrucciones que permitan probar la actualización de la BD haciendo uso de cada disparador cuando sea pertinente.

(DisparadoresOK)

Código presente en el archivo [*[auto4.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Auestudios%20•/auto4.sql)

1. Escriban 3 instrucciones para validar que protege la BD impidiendo cambios cuando sea pertinente.

(DisparadoresNoOK)

Código presente en el archivo [*[auto4.sql]*](https://d.docs.live.net/35ab795c14f570bc/Compartidos/MBDA/Auestudios%20•/auto4.sql)

## Bibliografía

Moodle ECI. (2024). *Requisitos de Entrega*. Consultado de: https://campusvirtual. escuelaing.edu.co/moodle/mod/wiki/view.php?id=59242

Gavilán, F. (2015). *Acciones Referenciales.* Consultado de: https://gavilanch. wordpress.com/2018/05/06/entity-framework-core-2-acciones-referenciales

Microsoft. (2023). *¿Qué es un cursor?.* Consultado de: <https://learn.microsoft.com/es-es/sql/ado/guide/data/what-is-a-cursor?view=sql-server-ver16>

Oracle. (2024). *PL/SQL para Desarrolladores.* Consultado de: <https://www.oracle.com/co/database/technologies/appdev/plsql.html>

Oracle. (2024). *Trabajar con Acciones.* Consultado de: <https://docs.oracle.com/middleware/12213help/biee/es/BIEUG/GUID-63576665-7BFE-4D1E-ABB9-51B8AAFBB940.htm#BIEUG644>

SQLZoo. (2017). *Guest House*. Consultado de: <https://sqlzoo.net/wiki/Guest_House>

Wikipedia. (2023). *PL/SQL.* Consultado de: <https://es.wikipedia.org/wiki/PL/SQL>