**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:** 2024/04/25

# Autoestudio 5 – PL/SQL Básico

## INVESTIGACIÓN

1. **Transacciones**
2. ¿Cómo se define el comienzo y fin de una transacción en ORACLE?

**R/** Para definir el comienzo y final de una transacción en Oracle Database se utiliza la siguiente estructura:

BEGIN TRANSACTION;

-- OPERACIONES

COMMIT;

ROLLBACK;

El ***BEGIN*** funciona como el iniciador para indicar que las siguientes líneas hacen parte de la transacción. Después, se ponen las **OPERACIONES** de transacción que se deseen implementar. En estas pueden ir actualizaciones, declaración de variables, inserciones, eliminaciones, etc. Por último, el ***COMMIT*** funciona para confirmar los cambios y concretar la operación. Por el contrario, el comando ***ROLLBACK*** indica que hubo algún inconveniente durante el proceso de transacción y deshace los cambios realizados hasta dicho instante.

1. ¿Cuáles son los diferentes tipos de aislamiento que soporta ORACLE? ¿Cómo maneja los bloqueos? ¿Qué problemas resuelve?

**R/** Oracle Database ofrece varios niveles de aislamiento para manejar transacciones concurrentes de manera segura y eficiente. Los tipos de aislamiento más comunes son:

* ***Read Committed* (Lectura Comprometida):** Este es el nivel de aislamiento predeterminado en Oracle. En este nivel, las transacciones ven solo datos que han sido confirmados **(*commited*)**, es decir, los cambios realizados por otras transacciones que aún no se han confirmado **(*no commited*)** no son visibles. Sin embargo, dentro de una transacción, las filas pueden cambiar entre lecturas.
* ***Serializable* (Serializable):** En este nivel, Oracle garantiza que las transacciones no se intercalen, es decir, las transacciones se ejecutan como si fueran una después de la otra, sin que ninguna otra transacción intervenga. Esto evita problemas como la lectura sucia, la lectura fantasma y la intercalación.
* ***Repeatable Read* (Lectura Repetible):** En este nivel, Oracle garantiza que, dentro de una transacción, todas las lecturas de un conjunto de datos devolverán los mismos resultados, incluso si otras transacciones están realizando cambios en esos datos en paralelo. Esto previene la anomalía de lectura no repetible.
* ***Serializable Read* (Lectura Serializable):** Similar al nivel de aislamiento serializable, pero más flexible. Oracle permite que las transacciones lean los datos con un nivel de aislamiento más bajo, pero cuando necesitan escribir, se elevan al nivel serializable.

Oracle maneja bloqueos utilizando diferentes mecanismos de bloqueo, como bloqueos de fila ***(row locks),*** bloqueos de tabla ***(table locks)*** y bloqueos de recursos compartidos ***(share locks).*** Los bloqueos se adquieren automáticamente cuando una transacción modifica los datos y se liberan cuando se completa la transacción.

Algunos de los problemas que resuelve el control de aislamiento y bloqueo en Oracle incluye:

* **Conflictos de Concurrencia:** Evita que las transacciones concurrentes interfieran entre sí al garantizar que cada transacción tenga acceso coherente a los datos.
* **Lecturas Sucias:** Garantiza que las transacciones solo vean datos confirmados, evitando que lean datos que podrían revertirse más tarde.
* **Lecturas Fantasma:** Asegura que las transacciones no vean filas que no existían al principio de la transacción, pero que aparecieron durante la transacción debido a las operaciones de otras transacciones.
* **Anomalías de Lectura No Repetible:** Garantiza que una transacción no lea diferentes valores para una fila en diferentes momentos dentro de la misma transacción.

1. ¿Cuál es el tipo de aislamiento por defecto en ORACLE?

**R/** El tipo de aislamiento por defecto en Oracle Database es el nivel de aislamiento ***Read Committed*** (Lectura Comprometida). En este nivel, una transacción solo ve los datos que han sido confirmados ***(committed)*** por otras transacciones. Esto significa que los cambios realizados por otras transacciones que aún no se han confirmado ***(no committed)*** no son visibles para la transacción actual.

1. **Vistas**
2. ¿Cuáles son los mecanismos para la creación y borrado de vistas en ORACLE?

**R/** En Oracle es posible crear y borrar vistas utilizando **sentencias SQL específicas**. A continuación, se muestra la sintaxis de los mecanismos para la creación y el borrado de vistas:

* **Creación de Vistas:** Para crear una vista en Oracle, se utiliza la sentencia ***SQL CREATE VIEW***. Aquí está la sintaxis básica:

CREATE [OR REPLACE] VIEW view\_name AS

SELECT column1, column2, ...

FROM table\_name

[WHERE conditions];

* + ***OR REPLACE*** es opcional y se utiliza para reemplazar una vista existente con el mismo nombre si ya existe.
  + ***view\_name*** es el nombre que se le va a asignar a la vista.
  + ***column1, column2, ...*** son las columnas que se desean incluir en la vista.
  + ***table\_name*** es el nombre de la tabla de la cual se creará la vista.
  + ***WHERE*** son condiciones opcionales que se pueden aplicar para filtrar los datos en la vista.
* **Borrado de Vistas:** Para borrar una vista en Oracle, se utiliza la sentencia ***SQL DROP VIEW***. Aquí está la sintaxis básica:

DROP VIEW view\_name;

* ***view\_name*** es el nombre de la vista que se desea eliminar.

Es importante tener en cuenta que, al borrar una vista, solo se elimina la definición de la vista y no los datos subyacentes en las tablas.

1. ¿Cuáles son las restricciones de las vistas en ORACLE?

**R/** En Oracle, las vistas son objetos de base de datos que ofrecen una representación lógica de los datos almacenados en una o más tablas. Aunque las vistas pueden proporcionar una forma conveniente de acceder y manipular datos, están sujetas a ciertas restricciones que deben tenerse en cuenta al diseñarlas y utilizarlas. Algunas de las restricciones más importantes de las vistas en Oracle son:

* **Restricciones de Modificación Directa:** Las vistas no pueden ser modificadas directamente si contienen lo siguiente:
* Funciones de grupo como ***SUM***, ***COUNT***, ***AVG***, etc.
* Funciones de agregado como ***MIN*** y ***MAX***.
* Operaciones ***DISTINCT***, ***GROUP BY***, ***CONNECT BY***.
* Pseudocolumnas (***ROWNUM***, ***ROWID***).

Sin embargo, se pueden actualizar a través de la vista si cumplen con ciertos criterios, como tener una clave primaria definida o ser vista de una sola tabla.

* **Restricciones de DML (*Data Manipulation Language*):** Las operaciones de modificación (***INSERT, UPDATE, DELETE***) en una vista están sujetas a ciertas restricciones, como:
  + La vista debe ser "actualizable", lo que significa que debe estar basada en una sola tabla o en una consulta SQL que se pueda mapear de manera unívoca a una sola tabla.
  + No puede contener expresiones complejas, subconsultas correlacionadas, uniones externas, etc.

.

* **Restricciones de Columnas Calculadas:** Las vistas no pueden contener columnas calculadas directamente en la definición de la vista. Sin embargo, pueden referirse a columnas calculadas en las tablas.
* **Restricciones de Seguridad:** Las vistas heredan los permisos de las tablas subyacentes. Si un usuario no tiene permiso para acceder a una tabla subyacente, tampoco podrá acceder a la vista que hace referencia a esa tabla.
* **Restricciones de Referencia de Objeto:** Si se utiliza una vista en la definición de otra vista, ambas vistas deben existir al mismo tiempo. No se puede eliminar una vista si hay otra vista que hace referencia a ella.
* **Restricciones de Operaciones DDL (*Data Definition Language*):** No se pueden realizar ciertas operaciones DDL directamente en una vista, como modificar la estructura de la vista o eliminar una tabla subyacente mientras existe una vista que hace referencia a ella.
* **Restricciones de Dependencias:** Una vista no puede depender de una tabla temporal global. Además, las vistas de base de datos distribuidas tienen ciertas restricciones adicionales en términos de cómo se pueden definir y utilizar.

Estas son algunas de las restricciones más comunes que se aplican a las vistas en Oracle. Es importante tener en cuenta estas restricciones al diseñar y trabajar con vistas para evitar problemas y garantizar un funcionamiento adecuado del sistema.

1. **Modularidad – Paquetes**
2. ¿Para qué sirve un paquete?

**R/** Un paquete en una BD es una **colección lógica de objetos relacionados**, como procedimientos almacenados, funciones, tipos de datos definidos por el usuario y variables globales. Estos objetos están encapsulados en un solo contenedor lógico de la BD y son una característica importante en sistemas de BD como **Oracle, PL/SQL** y otros sistemas de gestión de BD **(DBMS).** Algunas razones importantes por las cuales se utilizan los paquetes son:

* **Encapsulación y Modularidad:** Los paquetes permiten agrupar lógicamente los objetos relacionados en un solo contenedor, lo que facilita su gestión y mantenimiento. Esto promueve la modularidad y facilita la comprensión del código.
* **Seguridad:** Los paquetes pueden ocultar los detalles de implementación de los procedimientos y funciones almacenados, lo que limita el acceso directo a los objetos internos del paquete. Esto ayuda a proteger la integridad y la seguridad de los datos.
* **Reutilización de Código:** Al agrupar procedimientos y funciones relacionados en un paquete, se promueve la reutilización del código. Otros programas o procedimientos pueden llamar a los objetos del paquete sin necesidad de volver a escribir el código.
* **Mejor Rendimiento:** Los paquetes pueden mejorar el rendimiento al reducir el tráfico de red y la sobrecarga del servidor. Esto se debe a que los objetos del paquete se almacenan en la base de datos una vez y se pueden reutilizar en múltiples llamadas, en lugar de enviar repetidamente el código al servidor.
* **Mantenimiento Simplificado:** Los paquetes facilitan el mantenimiento del código al proporcionar un único punto de acceso y modificación para los objetos relacionados. Esto ayuda a garantizar la coherencia y la consistencia en toda la aplicación.

En resumen, los paquetes son una herramienta poderosa en las BD que facilitan la organización, la seguridad, la reutilización y el mantenimiento del código, lo que contribuye a la eficiencia y la escalabilidad de las aplicaciones basadas en BD.

1. ¿Cuáles son los mecanismos para la creación, invocación, modificación y borrado de paquetes en ORACLE?

**R/** En Oracle, es posible crear, invocar, modificar y borrar paquetes utilizando **sentencias SQL específicas** y **comandos en PL/SQL**. A continuación, se muestran los mecanismos necesarios para realizar estas operaciones:

* **Creación de Paquetes:** Para crear un paquete en Oracle, primero hay que definir tanto las especificaciones como el cuerpo del paquete. La especificación contiene las declaraciones de procedimientos, funciones, tipos de datos y variables públicas, mientras que el cuerpo del paquete contiene las implementaciones reales de estos objetos.

La sintaxis para la creación de un paquete en Oracle es la siguiente:

CREATE [OR REPLACE] PACKAGE package\_name AS

-- Public object declarations

END package\_name;

CREATE [OR REPLACE] PACKAGE BODY package\_name AS

-- Public and private object implementations

END package\_name;

* **Invocación de Paquetes:** Una vez que el paquete está creado, se puede invocar los procedimientos y funciones definidos dentro de él utilizando su nombre cualificado.
* **Modificación de Paquetes:** Para modificar un paquete existente en Oracle, se puede utilizar la misma sintaxis de creación, pero omitiendo la palabra ***CREATE***. Oracle sobrescribirá el paquete existente con la nueva definición.
* **Borrado de Paquetes:** Para borrar un paquete en Oracle, se puede utilizar el comando ***DROP PACKAGE***. Aquí está la sintaxis básica:

DROP PACKAGE package\_name;

Es importante tener en cuenta que, al borrar un paquete, se eliminarán tanto la especificación como el cuerpo del paquete.

1. **SYS\_REFCURSOR**
2. ¿Qué es un *SYS\_REFCURSOR*?¿Para qué sirve?

**R/** Un ***SYS\_REFCURSOR*** es un **tipo de dato especial en Oracle** que se utiliza para **representar un cursor** de resultado o un puntero a un conjunto de filas devuelto por una consulta SQL. Es un tipo de cursor que puede ser definido y utilizado en programas PL/SQL para manejar resultados de consultas dinámicas o para pasar conjuntos de filas entre procedimientos o funciones. Al contrario de un cursor explícito, que debe ser declarado y abierto explícitamente, un **SYS\_REFCURSOR** se define como un parámetro de salida en un procedimiento o función y se abre implícitamente durante la ejecución del procedimiento o función. Un **SYS\_REFCURSOR** sirve para:

* **Retorno de Resultados Dinámicos:** Permite a un procedimiento almacenado o función devolver conjuntos de resultados cuya estructura o contenido no se conoce de antemano.
* **Comunicación entre Procedimientos:** Puede ser utilizado para pasar conjuntos de filas de un procedimiento a otro. Por ejemplo, un procedimiento puede abrir un cursor, recorrer sus resultados y luego devolver otro cursor para que una función lo procese más adelante.
* **Iteración sobre Resultados:** Un ***SYS\_REFCURSOR*** puede ser recorrido utilizando un bucle para procesar cada fila de resultados individualmente.
* **Flexibilidad en el Desarrollo:** Proporciona una forma flexible de manejar resultados de consultas SQL en programas PL/SQL, lo que permite construir aplicaciones que pueden adaptarse dinámicamente a diferentes escenarios.

1. ¿Cómo se define, se asigna y se retorna?

**R/** Para definir, asignar y retornar un ***SYS\_REFCURSOR*** en Oracle, generalmente se sigue el siguiente proceso:

1. **Definición del *SYS\_REFCURSOR*:** Se puede definir un ***SYS\_REFCURSOR*** como un parámetro de salida en un procedimiento almacenado o función. La definición básica sería:

PROCEDURE procedure\_name (ref\_cursor

OUT SYS\_REFCURSOR);

1. **Asignación del *SYS\_REFCURSOR*:** Dentro del cuerpo del procedimiento, se puede abrir un cursor y asignarlo al parámetro ***SYS\_REFCURSOR*** utilizando la cláusula ***OPEN FOR***. Por ejemplo:

OPEN ref\_cursor FOR

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

1. **Retorno del *SYS\_REFCURSOR*:** Si el procedimiento es una función y se requiere devolver el ***SYS\_REFCURSOR***, simplemente se incluye en la cláusula ***RETURN*** de la función. Por ejemplo:

FUNCTION function\_name RETURN SYS\_REFCURSOR IS

ref\_cursor SYS\_REFCURSOR;

BEGIN

OPEN ref\_cursor FOR

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

RETURN ref\_cursor;

END function\_name;

Si el procedimiento es un procedimiento almacenado, simplemente se ejecuta y el ***SYS\_REFCURSOR*** se pasa como parámetro de salida. Por ejemplo:

DECLARE

ref\_cursor SYS\_REFCURSOR;

BEGIN

procedure\_name (ref\_cursor);

-- You can use the resulting ref\_cursor here

END;

1. **Cierre del *SYS\_REFCURSOR*:** Es importante recordar que después de utilizar un ***SYS\_REFCURSOR***, se debe cerrar para liberar los recursos. Esto se hace utilizando la sentencia ***CLOSE***. Por ejemplo:

CLOSE ref\_cursor;

En resumen, para definir, asignar y retornar un `SYS\_REFCURSOR` en Oracle, primero se define como un parámetro de salida en un procedimiento almacenado o función, luego se asigna un cursor a este parámetro dentro del procedimiento o función y, finalmente, se puede devolver si es necesario o utilizar directamente en el bloque PL/SQL donde se invoca el procedimiento.

## PRÁCTICA

|  |  |
| --- | --- |
|  |  |

1. **Ofreciendo Servicios**
2. Implemente los paquetes de componentes necesario para ofrecer las funciones básicas y consultas del ciclo actual del sistema (CRUD).

**PC\_EXTRAS [Consultar Diseño al Final]**

(CRUDE (la especificación), CRUDI (la implementación))

1. Proponga un caso de prueba exitoso por subprograma.

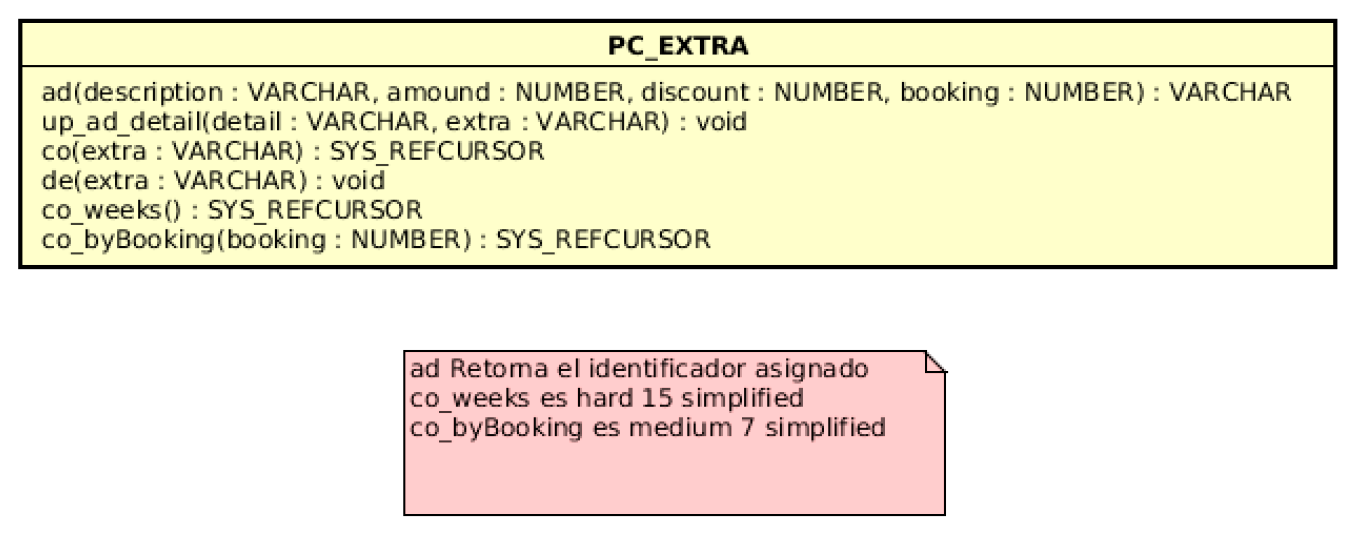
(CRUDOK)

1. Proponga tres casos en los que el subprograma no se puede ejecutar.

(CRUDNoOK)

1. Escriba las instrucciones necesarias para eliminar los paquetes.

(CRUDX)



## Bibliografía

Campus MVP. (2014). *Fundamentos de SQL: Transacciones.* Consultado de: <https://www.campusmvp.es/recursos/post/Fundamentos-de-SQL-Transacciones.asp>

El Baúl del Programador. (2017). *Introducción a PL/SQL. Transacciones*. Consultado de: <https://elbauldelprogramador.com/introduccion-plsql-transacciones/>

El Baúl del Programador. (2017). *PL/SQL. Paquetes (Packages).* Consultado de: <https://elbauldelprogramador.com/plsql-paquetes-packages/>

IBM. (2022). *Paquetes (PL/SQL).* Consultado de: <https://www.ibm.com/docs/es/db2/11.1?topic=support-packages-plsql>

Microsoft. (2023). *Esquemas de mensaje para cursores REF CURSOR.* Consultado de: <https://learn.microsoft.com/es-es/biztalk/adapters-and-accelerators/adapter-oracle-database/message-schemas-for-ref-cursors>

Oracle Help Center. (2024*). Creación de Vistas*. Consultado de: <https://docs.oracle.com/es-ww/iaas/database-tools/doc/creating-view.html>

Oracle Help Center. (2024*). Trabajar con Transacciones*. Consultado de: [https://docs.oracle.com/cloud/help/es/account-reconcile-cloud/RAARC/ admin\_recons\_transactions\_108xd7653e36.htm#RAARC-GUID-7C584869-3ABC-4EF8-B240-1EF1BB2D1798](https://docs.oracle.com/cloud/help/es/account-reconcile-cloud/RAARC/%20admin_recons_transactions_108xd7653e36.htm#RAARC-GUID-7C584869-3ABC-4EF8-B240-1EF1BB2D1798)