Universidad Mariano Gálvez de Guatemala

Boca del monte

Facultad de ingeniería en sistemas de la información

Curso: Base de datos I

Sección: A

Investigación proyecto final (individual)

Tedi Adolfo Alejandro Castellanos Amezquita 7690-23-3016

Guatemala, noviembre de 2025

Contenido

Resumen

El presente proyecto de investigación aborda el estudio de PL/SQL, un lenguaje de programación desarrollado por Oracle como una extensión procedimental de SQL. A diferencia del lenguaje SQL, que es declarativo y realiza una sola consulta a la vez , PL/SQL permite ejecutar bloques de código completos, combinando sentencias SQL con lógica procedimental.

La investigación detalla las características clave de PL/SQL, como su estructura en bloques (declarativa, de ejecución y de excepciones) , la integración estricta con SQL , el alto rendimiento al reducir el tráfico de red , y el soporte orientado a objetos. Un componente esencial son los cursores (implícitos y explícitos), que actúan como punteros al área de contexto de una declaración SQL para procesar filas de datos . El desarrollo de PL/SQL fue impulsado por la necesidad de una capacidad completa de programación procedimental dentro de la base de datos de Oracle, siendo Peter Clare su inventor principal. El conocimiento de PL/SQL es considerado una habilidad esencial para desarrolladores de bases de datos interesados en tecnologías RDBMS avanzadas.

Introduccion

El manejo eficiente de grandes volúmenes de datos es fundamental en la tecnología moderna, y para los sistemas de bases de datos relacionales (RDBMS) como Oracle, el lenguaje SQL ha sido el estándar para la gestión y manipulación de información. Sin embargo, SQL es un lenguaje declarativo que se centra en qué hacer, pero carece de la capacidad de programación procedimental necesaria para tareas complejas y la lógica de negocio.

Este proyecto se centra en PL/SQL (Procedural Language/SQL), la extensión de Oracle que resuelve esta limitación. PL/SQL complementa a SQL al introducir estructuras de control, variables, constantes y la organización en bloques de código. Esta integración permite a los desarrolladores especificar cómo llevar a cabo las operaciones de la base de datos , resultando en la creación de aplicaciones más eficientes, seguras y flexibles. El objetivo principal de esta investigación es explorar la naturaleza de PL/SQL, sus componentes fundamentales, como los cursores, y su relevancia como una habilidad esencial en el desarrollo de bases de datos.

Marco teórico

¿Qué es?

Es un lenguaje de programación que fue desarrollado directamente por Oracle, la cual se puede obtener por medio de una extensión de SQL, estos mismo son lenguajes de datos relacionales, pero tenemos unas diferencias:

- SQL es un lenguaje de consulta estructurado con el que podemos manejar la información de una base de datos, modificando esos datos, añadiendo o eliminando. Es un lenguaje declarativo.

- PL/SQL va más allá, es un lenguaje de programación por procedimientos, una ampliación de SQL que conserva sus sentencias en la sintaxis. Si con SQL solamente se realiza una consulta al mismo tiempo, con PL/SQL se ejecuta un bloque de código completo. Es uno de los lenguajes que se estudian en **FP**

Características:

La principal característica de PL/SQL es su capacidad para integrar construcciones de procedimientos con SQL. Al ejecutar un comando SQL, la base de datos recibe información sobre qué operación debe realizar, mientras que con PL/SQL se puede especificar cómo llevar a cabo esa operación. Si buscamos una definición debemos atender a sus particularidades.

- Se trata de un lenguaje de bases de datos sencillo y fácil de escribir y leer.
- Es un lenguaje de **procedimiento**, **estructurado en bloques** en el que son elementos fundamentales las variables, las constantes, la estructura de bucle y el cursor.
- Las **estructuras de control** son muy similares a la de otros lenguajes de programación.
- Permite **convertir problemas complejos en códigos de procedimiento** que pueden utilizarse en diferentes aplicaciones.
- A diferencia de otros lenguajes de bases de datos, PL/SQL **no distingue entre letras mayúsculas o minúsculas**.
- Puede guardar el contenido de una fila completa de una tabla en una única variable. Permite programar **funciones, triggers, procedimientos almacenados y scripts**.
- El usuario **puede utilizar las herramientas de Oracle**: Graphics, Reports, Application Server y Forms.

Contexto de uso:

Como cualquier lenguaje de programación, PL/SQL cuenta con sus propias unidades léxicas. Son las que establecen los elementos más pequeños con significado, elementos que, al combinarlos, atendiendo a las reglas de sintaxis, se convierten en sentencias. En PL/SQL se clasifican en:

- **Delimitadores**. Representan operaciones entre diferentes clases de datos. Pueden ser operadores aritméticos, lógicos o relacionales.

- **Literales**. Se emplean en las comparaciones de valores o para asignar valores concretos a los identificadores que actúan como variables o constantes.

- **Comentarios**. No influyen en la estructura básica pero sirven de apoyo para saber qué se está haciendo.

- **Identificadores**. Se utilizan para nombrar elementos del programa (variables, constantes, excepciones, cursores, subprogramas, paquetes).

La **estructura de bloque** es una de las bases de PL/SQL, que se ejecuta a través de estructura en la que pueden distinguirse tres partes diferentes, que incluyen una palabra clave:

- La **parte declarativa** en la que se declaran todas las constantes y variables que se usarán durante la ejecución del proceso.

Comienza con la palabra ***DECLARE***.

- La **parte de ejecución** que incluye las instrucciones a llevar ejecutar en el bloque PL/SQL. Es la única obligatoria.

Comienza con ***BEGIN*** y finaliza con ***END***.

- La **parte de excepciones**. Las excepciones permiten detectar y gestionar errores durante la ejecución del proceso.

Comienza con la palabra ***EXCEPTION***.

PS/SQL permite manejar diferentes tipos de datos, la mayoría son los mismos que SQL. Los más utilizados son:

- ***CHAR*** (carácter): datos de tipo carácter que tengan un tamaño máximo de 32.767 bytes. Si no se especifica su valor de longitud, por defecto es 1.
- ***VARCHAR2*** (carácter de longitud variable): almacena datos de longitud variable.
- ***NUMBER*** (numérico): guarda números enteros o de punto flotante, especificando la precisión (número de dígitos) y la escala (número de decimales).
- ***BOOLEAN*** (lógico): almacena valores TRUE o FALSE.
- ***DATE*** (fecha): datos sobre fechas que se guardan como datos numéricos y, por lo tanto, pueden hacerse operaciones aritméticas con ellas.

En cuanto a los **procedimientos y funciones**, se emplean para ejecutar **tareas concretas**, la diferencia está en que las funciones siempre devuelven un valor, por lo que se emplean para realizar cálculos.

Como lenguaje de programación procedimental, PL/SQL permite **ejecutar procedimientos y funciones, declarar constantes y variables o emplear tablas** en las que almacenar varios valores del mismo tipo de dato.

Creación y gestión de unidades de programa PL/SQL

- Creación con bloques: PL/SQL es un lenguaje estructurado por bloques; la familiaridad con los bloques es fundamental para escribir un buen código.
- Control del flujo de ejecución: Bifurcación condicional y procesamiento iterativo en PL/SQL.
- Presenta tu código en un paquete limpio: los paquetes son los componentes clave fundamentales de cualquier aplicación basada en PL/SQL de alta calidad
- Selecciona tus paquetes: conceptos y ventajas de los paquetes PL/SQL
- Gestión de errores: Exploración de funciones de gestión de errores en PL/SQL
- The Data Dictionary: Logra que tus vistas resulten útiles para ti:Utiliza varias vistas clave del diccionario de datos para analizar y administrar tu código

PL/SQL SQL estático

Estas son las declaraciones SQL estáticas PL/SQL, que tienen la misma sintaxis que las declaraciones SQL correspondientes, excepto que se indique lo contrario:

- SELECT (esta afirmación también se llama a **consulta**)

  Para conocer la sintaxis PL/SQL, consulte "SELECCIONAR EN Declaración".

- Lenguaje de manipulación de datos (Declaraciones DML):
  - INSERT

Para conocer la sintaxis PL/SQL, consulte "Extensión de declaración INSERT"

- UPDATE

Para conocer la sintaxis PL/SQL, consulte "ACTUALIZAR extensiones de declaración"

- DELETE

Para conocer la sintaxis PL/SQL, consulte "Eliminar extensión de declaración"

- MERGE (para la sintaxis, consulte *Referencia de lenguaje SQL de Oracle Database*)

Por que aprender PL/SQL

- Aprender PL/SQL es una habilidad esencial para las personas interesadas en bases de datos y otras tecnologías RDBMS avanzadas. PL/SQL ofrece varios beneficios, lo que lo convierte en una habilidad esencial para los desarrolladores **de bases de datos**
- Facilidad de uso: PL/SQL es sencillo de escribir y leer y presenta una sintaxis estructurada en bloques que simplifica la programación y la depuración.
- Portabilidad: Los programas escritos en PL/SQL son totalmente portables en diferentes bases de datos Oracle, lo que garantiza la coherencia y la facilidad de migración.
- Integración SQL estricta: PL/SQL está estrechamente integrado con SQL, lo que permite consultar, transformar y actualizar datos de manera eficiente dentro de una base de datos.
- Alto rendimiento: Reduce el tráfico de red enviando bloques enteros de declaraciones a la base de datos a la vez, mejorando así el rendimiento.
- Seguridad: Incluye sólidas funciones de seguridad para proteger la integridad de la base de datos.
- Soporte orientado a objetos: Admite programación orientada a objetos y le permite definir tipos de objetos que se pueden utilizar en diseños orientados a objetos.

Cursores:

Oracle crea un área de memoria, conocida como área de contexto, para procesar una declaración SQL, que contiene toda la información necesaria para procesar la declaración; por ejemplo, la cantidad de filas procesadas, etc.

A cursor es un puntero a esta área de contexto. PL/SQL controla el área de contexto a través de un cursor. Un cursor contiene las filas (una o más) devueltas por una declaración SQL. El conjunto de filas que contiene el cursor se denomina conjunto activo.

Puede nombrar un cursor para que pueda ser mencionado en un programa para recuperar y procesar las filas devueltas por la declaración SQL, una a la vez. Hay dos tipos de cursores:

- Cursores implícitos
- Cursores explícitos

Cursores Implicitos

Oracle crea automáticamente cursores implícitos cada vez que se ejecuta una declaración SQL, cuando no hay un cursor explícito para la declaración. Los programadores no pueden controlar los cursores implícitos ni la información que contienen.

Cada vez que se emite una declaración DML (INSERTAR, ACTUALIZAR y ELIMINAR), se asocia un cursor implícito con esta declaración. Para las operaciones INSERT, el cursor contiene los datos que deben insertarse. Para las operaciones ACTUALIZAR y ELIMINAR**,** el cursor identifica las filas que se verían afectadas.

En PL/SQL, puede referirse al cursor implícito más reciente como **Cursor SQL**, que siempre tiene atributos como **%ENCONTRADO, %ISOPEN, %NOENCONTRADO**, y **%RECUENTO DE FILAS**. El cursor SQL tiene atributos adicionales, **%CUENTO_FILA A GRANEL** y **%EXCEPCIONES_A GRANEL**, diseñado para usarse con el **FORALL** declaración. La siguiente tabla proporciona la descripción de los atributos más utilizados

S.No	Atributo y descripción
1	**%ENCONTRADO** Devuelve VERDADERO si una declaración INSERT, UPDATE o DELETE afectó una o más filas o una declaración SELECT INTO devolvió una o más filas. De lo contrario, devuelve FALSO.
2	**%NO ENCONTRADO** El opuesto lógico de %FOUND. Devuelve VERDADERO si una declaración INSERT, UPDATE o DELETE no afectó ninguna fila, o una declaración SELECT INTO no devolvió ninguna fila. De lo contrario, devuelve FALSO.
3	**%ISOPEN** Siempre devuelve FALSO para cursores implícitos, porque Oracle cierra el cursor SQL automáticamente después de ejecutar su declaración SQL asociada.
4	**%RECUENTO DE FILAS** Devuelve la cantidad de filas afectadas por una declaración INSERT, UPDATE o DELETE, o devueltas por una declaración SELECT INTO.

Se accederá a cualquier atributo del cursor SQL como **sql%atributo_nombre** como se muestra a continuación en el ejemplo.

Ejemplo

Utilizaremos la tabla CLIENTES que habíamos creado y utilizado en los capítulos anteriores.

```
Select * from customers;
```

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

El siguiente programa actualizará la tabla y aumentará el salario de cada cliente en 500 y utilizará el **SQL%ROWCOUNT** atributo para determinar el número de filas afectadas –

```
DECLARE
   total_rows number(2);
BEGIN
   UPDATE customers
   SET salary = salary + 500;
   IF sql%notfound THEN
      dbms_output.put_line('no customers selected');
   ELSIF sql%found THEN
      total_rows := sql%rowcount;
      dbms_output.put_line( total_rows || ' customers selected ');
   END IF;
END;
/
```

Cursores explícitos:

Los cursores explícitos son cursores definidos por el programador para obtener más control sobre el área de contexto. Se debe definir un cursor explícito en la sección de declaración del bloque PL/SQL. Se crea en una declaración SELECT que devuelve más de una fila.

La sintaxis para crear un cursor explícito es:

CURSOR cursor_name IS select_statement;

Trabajar con un cursor explícito incluye los siguientes pasos:

- Declarar el cursor para inicializar la memoria
- Abrir el cursor para asignar la memoria
- Obtener el cursor para recuperar los datos
- Cerrar el cursor para liberar la memoria asignada

Ventajas:

Permite escribir código procedimental que incluye SQL tan fácilmente, como si fuera un solo lenguaje. PL/SQL está estrechamente integrado con SQL, el lenguaje más popular y ampliamente utilizado para la manipulación de bases de datos. Esta integración permite a los programadores utilizar todas las declaraciones, funciones y operadores SQL. Rara vez es necesario realizar conversiones de tipos de datos con PL/SQL, ya que el lenguaje admite todos los tipos de datos SQL. Además, los programadores pueden utilizar ambos estático y dinámico SQL para crear aplicaciones más flexibles y versátiles.

Interfaz de programación de aplicaciones específicas (API) no se necesitan conocimientos para mapear tipos de datos, preparar declaraciones o incluso procesar conjuntos de resultados. Los desarrolladores pueden utilizar API en PL/SQL para abstraer estructuras de datos complejas e implementaciones de seguridad de aplicaciones cliente. Por ejemplo, al agregar código - inserciones, actualizaciones, eliminaciones - a una API, pueden envolver el procesamiento transaccional en esa API y evitar tener que crear activadores, por ejemplo, para implementar una funcionalidad comercial específica en una aplicación. Una vez configurada la implementación de la API, se puede ajustar según sea necesario. La aplicación cliente permanece intacta durante el ajuste, lo que reduce la necesidad de redistribuir la aplicación.

Con PL/SQL, la lógica empresarial se construye en la base de datos mediante el uso de bloques, y cada bloque contiene múltiples declaraciones SQL. Como resultado, el código del cliente solo necesita una única llamada a la base de datos por transacción. La aplicación envía un único bloque PL/SQL a la base de datos y la base de datos procesa todo este bloque en una sola pasada. Este enfoque reduce sobrecarga de red.

PL/SQL permite almacenar la lógica de la aplicación en la propia base de datos. Las declaraciones SQL y otras construcciones PL/SQL se agrupan y almacenan como esquema objeto (procedimiento o función PL/SQL). Estas declaraciones y construcciones se ejecutan como una unidad para resolver problemas o realizar tareas

relacionadas. La centralización de la lógica de la aplicación en la base de datos puede aumentar la seguridad del programa y la productividad del desarrollador.

Otro beneficio de PL/SQL se relaciona con la codificación de lógica empresarial. En aplicaciones de nivel medio, pueden ocurrir múltiples interacciones entre el servidor de aplicaciones y la base de datos para procesar y ejecutar una única transacción comercial. Esto aumenta la sobrecarga del tráfico de red entre la aplicación y la base de datos.

Finalmente, PL/SQL es una buena opción cuando se trabaja con Oracle Database.

Oracle Database es un sistema de gestión de bases de datos relacionales (RDBMS) que controla el almacenamiento, la organización y la recuperación de datos en una base de datos relacional. Implementa características orientadas a objetos, como tipos definidos por el usuario, herencia y polimorfismo, y garantiza que el almacenamiento físico de datos sea independiente de las estructuras de datos lógicas. Dado que amplía el modelo relacional de a DBMS a un modelo relacional de objetos, admite el almacenamiento y manipulación de modelos de negocio complejos en una base de datos relacional.

Historia de PL/SQL

La carrera de Peter Clare comenzó en 1972 en Corporación de datos de control (CDC). Aunque CDC ya no existe, durante las décadas de 1960 y 1970 fue una de las principales empresas de mainframes y supercomputadoras de Estados Unidos. En términos de software, la División de Productos Compiladores de CDC creó el primer comercial Pascal compilador. El papel de Peter en esta división consistió en el desarrollo de lenguajes y compiladores centrados en construir los procesadores de lenguajes informáticos más rápidos del mundo. Durante sus seis años de mandato allí, Peter no sólo mejoró los compiladores de CDC sino que también aprendió mucho sobre el desarrollo general del lenguaje y, lo que es más importante, las cuestiones prácticas relacionadas con él. Con este conocimiento, Peter dejó los CDC y se convirtió en consultor.

Años más tarde, en 1984, Peter revisó una de las publicaciones tecnológicas locales del área de la bahía en busca de trabajo adicional relacionado con consultoría. Dentro de esa publicación había un anuncio de Oráculo. A través de los rumores, Peter había oído que Oracle estaba en proceso de portar la base de datos a muchas plataformas diferentes, una de las cuales conocía bastante bien. Pensando que podría ser una excelente oportunidad para trabajar por contrato, Peter llamó a Oracle. Unos días después, Peter se reunió con Bob minero y Bill Friend para una entrevista.

Inicialmente, el trío revisó la experiencia de Peter relacionada con la portabilidad. Como Oracle también estaba trabajando en SQL precompiladores y un interfaz basada en formularios a la base de datos, el grupo también discutió la experiencia de Peter en el desarrollo de lenguajes y compiladores. Al final de esa reunión, Bob y Bill convencieron a Peter de aceptar un trabajo de tiempo completo; Peter se convirtió en el decimotercer desarrollador de Oracle.

Peter no era el típico desarrollador de Oracle. Descrito por sus compañeros de equipo como un arquitecto amigable, tranquilo, competente y muy capaz, Peter no solo era conocido por ser inteligente, sino también por su amor por el agua y windsurf, lo cual hacía casi todos los días. Peter tomó en serio el viejo lema "trabajar duro, jugar duro", esforzándose por lograr un equilibrio satisfactorio entre el trabajo y la vida personal – que perfeccionó a lo largo de su carrera.

En sus primeros seis meses en Oracle, las responsabilidades de Peter estaban más relacionadas con la gestión de proyectos que con la técnica. Bob quería centrarse en el desarrollo y confiaba en Peter para gestionar las tareas diarias del proyecto. Aunque a Peter le gustaba el trabajo, la comunicación se volvió cada vez más desafiante. "En ese momento, Bob y Larry habían estado trabajando juntos durante bastante tiempo", recordó Peter, "era natural que se comunicaran directamente entre sí y, como el hombre del medio, no pasó mucho tiempo antes de que todos se dieran cuenta de que yo era la tercera rueda" Para reducir esta fricción, todos coincidieron en que sería mejor que Peter se centrara en el trabajo técnico.

Dada su experiencia en lenguajes de programación, Peter comenzó a trabajar en los precompiladores de Oracle. No pasó mucho tiempo hasta que surgió otro problema: la competencia.

En aquellos días, un par de proveedores de bases de datos tenían soluciones simples desencadenantes. De manera similar, la única forma en que la base de datos podía ejecutar una función procedimental era escribirla en otro lenguaje, compilarla y escribir una interfaz. Ken Rudin, gerente de producto inicial de PL/SQL, recuerda: "como estos enfoques eran algo limitantes, en lugar de implementar una funcionalidad comparable, Oracle quería superar a la competencia y ofrecer una capacidad completa de programación procedimental que viviera dentro de la base de datos. Afortunadamente, Oracle se dio cuenta desde el principio de que un lenguaje procedimental consistente sería beneficioso en múltiples herramientas" Ese lenguaje era PL/SQL y la mejor persona para inventarlo fue Peter Clare.

Metodologia

La metodología PL/SQL se basa en integrar lógica procedimental con sentencias SQL para crear programas más complejos y eficientes dentro de la base de datos, mejorando la modularidad y la reutilización del código. Esta metodología incluye la definición de bloques de código con nombre, como procedimientos y funciones, que pueden incluir variables, estructuras de control (como bucles IF y WHILE), cursores y el manejo de colecciones.

Características principales

- Combinación de lenguajes: Fusiona la sintaxis de SQL con la lógica procedimental para procesar datos de manera más avanzada que solo con SQL.
- Bloques de código: Se estructura en bloques que pueden ser almacenados en la base de datos, como procedimientos almacenados, funciones, triggers y paquetes.
- Eficiencia: La ejecución de PL/SQL y SQL se realiza en el mismo proceso de servidor, lo que proporciona un rendimiento óptimo.
- Manejo de errores: Permite incluir manejo de errores para gestionar de forma controlada las excepciones.
- Herencia de la base de datos: Hereda la robustez, seguridad y portabilidad de la base de datos subyacente.
- Componentes clave
- Variables y constantes: Para almacenar datos y valores.
- Estructuras de control: Permiten controlar el flujo de ejecución con sentencias como IF...THEN...ELSE y bucles LOOP, WHILE, FOR.
- Cursores: Permiten procesar un conjunto de resultados de una consulta, fila por fila.
- Colecciones: Permiten almacenar y manipular múltiples elementos, como arrays anidados y tablas.
- Procedimientos y funciones: Bloques de código con nombre que realizan una tarea específica y se pueden llamar varias veces desde otros programas.

Resultados

La investigación detallada de PL/SQL arrojó los siguientes resultados clave sobre su estructura, características y componentes esenciales:

1. **Naturaleza y Diferencia con SQL:** Se confirmó que PL/SQL es un **lenguaje de programación procedimental** que amplía a SQL. La diferencia operativa es crucial: SQL ejecuta una consulta a la vez (lenguaje declarativo) , mientras que PL/SQL ejecuta un **bloque de código completo**, lo que mejora la eficiencia.

2. **Estructura de Bloque:** Se identificó la **estructura de bloque** como la base de PL/SQL, compuesta por tres partes:
    - **DECLARE (Parte Declarativa):** Para definir variables y constantes.
    - **BEGIN...END (Parte de Ejecución):** La única parte obligatoria, contiene las instrucciones a ejecutar.
    - **EXCEPTION (Parte de Excepciones):** Para detectar y gestionar errores durante la ejecución.

3. **Beneficios Clave:** Se validaron múltiples beneficios de su uso:
    - **Portabilidad** entre diferentes bases de datos Oracle.
    - **Alto rendimiento** por la reducción de sobrecarga de red, al enviar bloques completos en una sola llamada.
    - Soporte para **programación orientada a objetos**.
    - Capacidad de **centralizar la lógica de negocio** en la base de datos (procedimientos y funciones almacenadas).

4. **Componentes de Cursors:** Se describieron los cursores como **punteros al área de contexto** (memoria) que Oracle crea para procesar sentencias SQL.
    - **Cursores Implícitos:** Creados automáticamente por Oracle para cada sentencia SQL sin un cursor explícito. Son referenciados a través del **Cursor SQL** y sus atributos (%FOUND, %ROWCOUNT, etc.).
    - **Cursores Explícitos:** Definidos por el programador para un **mayor control** en consultas que devuelven múltiples filas. Requieren pasos explícitos: **Declarar, Abrir, Obtener y Cerrar** .

Discusión

Los resultados de esta investigación confirman la posición de PL/SQL no solo como un complemento de SQL, sino como un **lenguaje procedimental robusto** que ofrece una ventaja significativa en el desarrollo de aplicaciones de bases de datos Oracle. La integración de la lógica de programación (variables, estructuras de control, bucles) directamente con las sentencias SQL estáticas (SELECT, INSERT, UPDATE, DELETE) permite a los desarrolladores **convertir problemas complejos en códigos de procedimiento**.

La **centralización de la lógica empresarial** en la base de datos mediante paquetes, funciones y procedimientos almacenados es quizás el argumento más fuerte a favor de PL/SQL, ya que esto:

- **Minimiza la sobrecarga de red** , dado que la aplicación cliente solo necesita realizar una única llamada a la base de datos por transacción.
- **Aumenta la seguridad** y la **productividad del desarrollador**.
- Permite la implementación de **API** que abstraen estructuras de datos y seguridad , permitiendo ajustes sin la necesidad de redistribuir la aplicación cliente.

La gestión de datos fila por fila mediante **cursores explícitos** es un mecanismo clave que contrarresta la naturaleza de conjunto (set-based) de SQL. Aunque los cursores son esenciales para el control de procesamiento , requieren un manejo cuidadoso (declaración, apertura, obtención y cierre) . En contraste, los cursores implícitos ofrecen simplicidad para las operaciones DML, permitiendo a los desarrolladores acceder a métricas transaccionales cruciales como %ROWCOUNT (conteo de filas afectadas).

En esencia, la historia de PL/SQL, impulsada por la necesidad de superar las limitaciones de la competencia en cuanto a funcionalidades de programación dentro de la base de datos , subraya su valor permanente como una **habilidad esencial** para el desarrollo de Oracle.

Conclusion

PL/SQL es un **lenguaje de programación procedimental** desarrollado por Oracle que sirve como una potente extensión de SQL. Su diseño logra una **integración estricta** entre la sintaxis SQL declarativa y la lógica de programación procedimental, permitiendo la creación de soluciones complejas directamente en la base de datos.

Las conclusiones fundamentales de este estudio son:

- PL/SQL se basa en una **estructura de bloques modular** (`DECLARE`, `BEGIN...END`, `EXCEPTION`), que promueve un código organizado y facilita el manejo de excepciones.
- El uso de PL/SQL ofrece **ventajas significativas en el rendimiento** de las aplicaciones al reducir el tráfico de red y permite la **centralización eficiente de la lógica de negocio**.
- Los **cursores** (implícitos y explícitos) son mecanismos esenciales para el procesamiento controlado de conjuntos de resultados de consultas, con los cursores explícitos ofreciendo al programador el máximo nivel de control.

En definitiva, dada su **portabilidad, seguridad y alto rendimiento** , el conocimiento de PL/SQL es indispensable para cualquier desarrollador que trabaje con Oracle Database, asegurando la creación de sistemas de gestión de datos robustos y eficientes.

Referencias

1. https://www.unir.net/revista/ingenieria/que-es-plsql/ UNIR Revista. (21 de septiembre de 2021). *¿Qué es PL/SQL? Descripción, características y contexto de uso*. UNIR. https://www.unir.net/revista/ingenieria/que-es-plsql/

2. https://docs.oracle.com/cd/E11882_01/appdev.112/e25519/static.htm#LNPLS99878 Oracle. (2009). *Overview of PL/SQL static SQL*. En *PL/SQL User's Guide and Reference (11g Release 2 (11.2))*. https://docs.oracle.com/cd/E11882_01/appdev.112/e25519/static.htm#LNPLS99878 *Nota: La fecha se toma del lanzamiento general de Oracle 11g Release 2 (11.2).*

3. https://docs.oracle.com/en/database/oracle/oracle-database/18/arpls/ Oracle. (2018). *PL/SQL Packages and Types Reference (Oracle Database 18c)*. https://docs.oracle.com/en/database/oracle/oracle-database/18/arpls/ *Nota: La fecha se toma del lanzamiento general de Oracle Database 18c.*

4. https://www.tutorialspoint.com/plsql/plsql_cursors.htm TutorialsPoint. (s.f.). *PL/SQL - Cursors*. https://www.tutorialspoint.com/plsql/plsql_cursors.htm *Nota: Al no tener una fecha de publicación específica, se usa **(s.f.)** que significa "sin fecha".*

5. https://blogs.oracle.com/connect/post/object-oriented-plsql Feuerstein, S. (17 de mayo de 2016). *Object-Oriented PL/SQL*. Oracle Blogs. https://blogs.oracle.com/connect/post/object-oriented-plsql

6. https://www.techtarget.com/searchoracle/definition/PL/SQL TechTarget. (s.f.). *What is PL/SQL?*. https://www.techtarget.com/searchoracle/definition/PL/SQL *Nota: Al no tener una fecha de publicación específica ni un autor individual, se usa **(s.f.)** y el nombre de la organización como autor.*

7. https://www.ibm.com/docs/es/db2/12.1.0?topic=support-functions-plsql IBM. (2024). *Funciones de soporte PL/SQL*. En *IBM Documentation (Db2 12.1.0)*. https://www.ibm.com/docs/es/db2/12.1.0?topic=support-functions-plsql *Nota: La fecha se toma del lanzamiento general de Db2 12.1.0.*

8. https://oracle-internals.com/blog/2020/04/29/a-not-so-brief-but-very-accurate-history-of-pl-sql/ Harris, J. (29 de abril de 2020). *A (Not So) Brief But (Very) Accurate History of PL/SQL*. Oracle Internals. https://oracle-internals.com/blog/2020/04/29/a-not-so-brief-but-very-accurate-history-of-pl-sql/