PL/SQL

Contenido

- Bloques
- Creación de Program Units
- Variables y Constantes
- Tipos de Datos
- Parámetros
- Estructuras de Control
- Manejo de Excepciones
- Manejo de Cursores

Contenido

- Packages
- Cursores Empaquetados
- Sobrecarga de Operadores
- Estructura de Datos
- Triggers de la Base de Datos

Bloques

- PL/SQL es un lenguaje estructurado por bloques
- Un bloque permite agrupar lógicamente declaraciones, sentencias y excepciones
- Un bloque puede ser:
  - anónimo
  - procedimiento
  - función

Bloques

- Se puede anidar subloques en:
  - área de sentencias
  - área de excepciones
- Definición de subprogramas locales en área de declaraciones
- Las declaraciones y las sentencias se separan por ;

Bloques

- Bloque anónimo

```
[DECLARE
        --declaraciones]
BEGIN
        --sentencias
[EXCEPTION
        --manejadores]
END;
```

Bloques

- Procedimiento:

```
PROCEDURE nombre (parámetros)
IS
        --declaraciones]
BEGIN
        --sentencias
[EXCEPTION
        --manejadores]
END;
```

Bloques

- Función:

```
FUNCTION nombre (parámetros) RETURN
  tipo
IS
    --declaraciones]
BEGIN
    --sentencias
[EXCEPTION
    --manejadores]
```

Creación de Program Units

- Para crear en la base de datos un procedimiento o función almacenado se debe prefijar con:

CREATE [OR REPLACE] texto

- Para ver los errores en SQL*Plus utilizar SHOW ERRORS

Variables y Constantes

- PL/SQL permite declarar variables y constantes
- Estas deben ser declaradas antes de ser referenciadas
- Las variables y constantes se declaran en el área de declaraciones de un bloque

Variables y Constantes

- Reglas de alcance:
  - Una variable o constante declarada en un bloque puede ser referenciada solamente en ese bloque y sus sub-bloques
  - Un procedimiento o función declarado en un bloque puede ser invocado solamente en ese bloque y sus sub-bloques

Variables y Constantes

- Para declarar una constante la sintaxis es:

  constant_name CONSTANT

  tipo_dato [NOT NULL](:=|DEFAULT) expression;

- Para declarar una variable la sintaxis es:

variable_name tipo_dato [[NOT NULL] (:=|DEFAULT) expression);

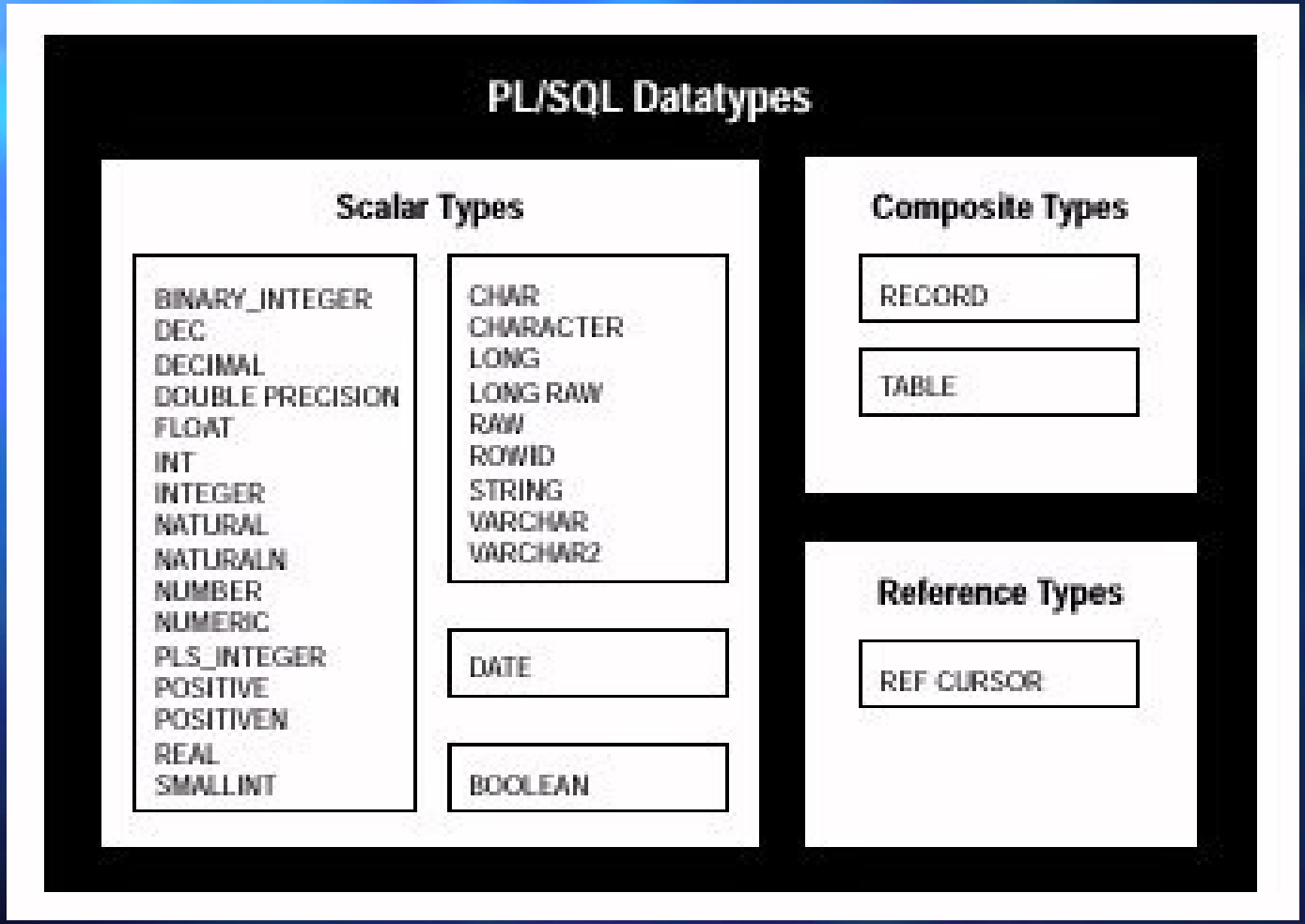
Variables y Constantes

- Hay diferentes formas de asignar valor a una variable:
  - asignación directa :=
  - SELECT.....INTO variable,..,variable
  - FETCH cursor INTO variable,..,variable

Tipos de Datos

- El tipo de dato de una variable o constante puede ser:
  - un tipo escalar
  - un tipo predefinido
  - el tipo de otra variable o columna de una tabla, registro o cursor (%type)
  - el tipo de la fila de una tabla, un registro o la fila de un cursor (%rowtype)

Tipos de Datos

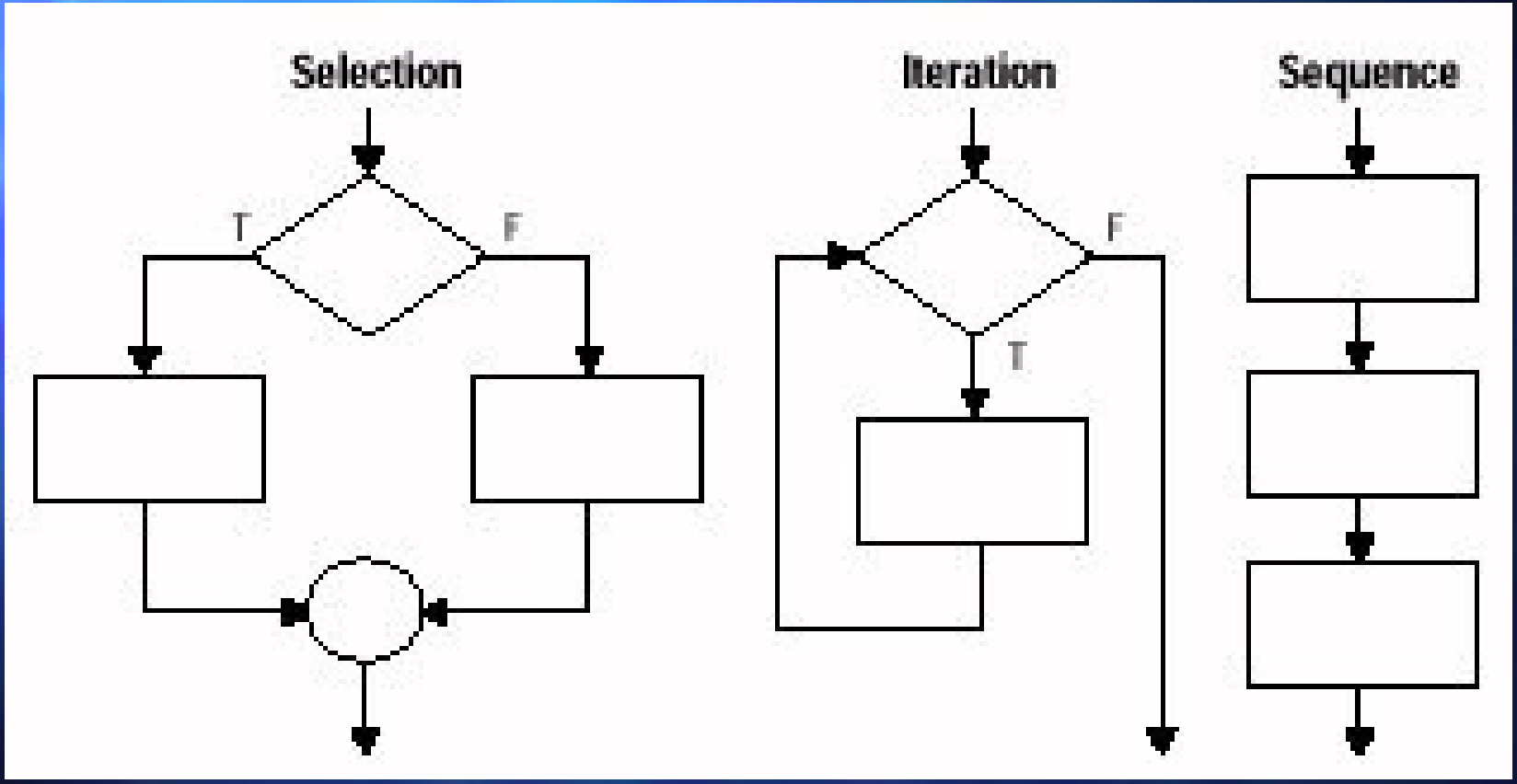


Parámetros

- Los subprogramas se pasan información mediante parámetros
- La sintaxis para definir un parámetro es

  parameter_name [IN|OUT|IN OUT] tipo [(:=|DEFAULT) expression]
- Los parámetros pueden pasarse
  - posicionalmente
  - por nombre

Estructuras de Control



Estructuras de Control

- IF-THEN

```
IF condicion THEN
    sequencia_de_sentencias;
END IF;
```

Estructuras de Control

- IF-THEN-ELSE

```
IF condicion THEN
    sequencia_de_sentencias1;
ELSE
    sequencia_de_sentencias2;
END IF;
```

Estructuras de Control

- IF-THEN-ELSIF

```
IF condicion THEN
    sequencia_de_sentencias1;
ELSIF
    sequencia_de_sentencias2;
ELSE
    sequencia_de_sentencias3;
END IF;
```

Estructuras de Control

- LOOP

```
<etiqueta>
  LOOP
      sequencia_de_sentencias;
  END LOOP; <etiqueta>
```

Estructuras de Control

- EXIT

```
EXIT [etiqueta];
```

- EXIT-WHEN

```
EXIT [etiqueta] WHEN condición;
```

Estructuras de Control

- WHILE-LOOP

```
WHILE condicion LOOP
        sequencia_de_sentencias;
END LOOP;
```

Estructuras de Control

- FOR-LOOP

```
    FOR contador IN [REVERSE]
min..max
    LOOP
          sequencia_de_sentencias;
    END LOOP;
```

Estructuras de Control

- GOTO

```
GOTO label;
```

- NULL

```
NULL;
```

Manejo de Excepciones

- En PL/SQL un error o advertencia se llama excepcion (exception)
- Pueden ser definidas internamente o por el usuario
- Cuando se produce un error una excepcion es disparada (raised)
- La ejecución se detiene y el control se transfiere al área de excepciones del bloque

Manejo de Excepciones

- Para manejar excepciones se escriben rutinas separadas llamadas manejadores (handlers) de excepciones.
- Forma de un manejador de excepciones:

```
WHEN excepcion,[excepcion] THEN
    sequencia_de_sentencias;
```

Manejo de Excepciones

- Ventajas:
  - permite agrupar el manejo de errores y manejarlo separadamente
  - legibilidad de programas
  - robustez y confiabilidad
- Desventajas:
  - capturan sólo errores de ejecución
  - enmascaran la fuente del error

Manejo de Excepciones

Exception Name	*Oracle Error*	*SQLCODE Value*
CURSOR_ALREADY_OPEN	ORA–06511	–6511
DUP_VAL_ON_INDEX	ORA–00001	–1
INVALID_CURSOR	ORA–01001	–1001
INVALID_NUMBER	ORA–01722	–1722
LOGIN_DENIED	ORA–01017	–1017
NO_DATA_FOUND	ORA–01403	+100
NOT_LOGGED_ON	ORA–01012	–1012
PROGRAM_ERROR	ORA–06501	–6501
ROWTYPE_MISMATCH	ORA–06504	–6504
STORAGE_ERROR	ORA–06500	–6500
TIMEOUT_ON_RESOURCE	ORA–00051	–51
TOO_MANY_ROWS	ORA–01422	–1422
VALUE_ERROR	ORA–06502	–6502
ZERO_DIVIDE	ORA–01476	–1476

For a complete list of the messages that Oracle or PL/SQL might issue, see *Oracle7 Server Messages*.

Manejo de Excepciones

- Una excepción de usuario se declara:

```
excepcion EXCEPTION;
```

- Luego se dispara con:

```
RAISE excepcion;
```

Manejo de Excepciones

- Para definir el resto de las excepciones se utiliza la excepcion OTHERS
- Para asociar una excepción de usuario con un error de Oracle se debe declarar:

```
PRAGMA EXCEPTION_INIT
(nombre,nro_error_Oracle)
```

Manejo de Excepciones

- En el PL/SQL de la base de datos, para producir una excepción a ser capturada por una aplicación se debe usar:

  RAISE_APPLICATION_ERROR(nro_error, mensaje[,TRUE|FALSE]);

donde nro_error está entre -20000 y -20999

Manejo de Excepciones

- Cuando se dispara una excpeción en un bloque es capturada por el manejador de excepciones de ese bloque
- Si dicho manejador no maneja la excepión, la excepción se propaga

Manejo de Excepciones

```
EXCEPTION
  WHEN nombre_exception1 THEN
     sequencia_de_sentencias1;
  WHEN nombre_exception1 THEN
     sequencia_de_sentencias2;
  .....
  WHEN OTHERS THEN
     sequencia_de_sentencias3;
END;
```

Manejo de Excepciones

- En un manejador de excepciones es posible utilizar las funciones:
  - SQLCODE: código del error
  - SQLERRM: mensaje del error

Manejo de Cursores

- Hay dos tipos de cursores
  - Implícitos: sentencias de manipulación de SQL
  - Explícitos:proceso individual de filas en consultas que devuelven más de una fila.

Manejo de Cursores

- Un cursor se declara por medio de

```
CURSOR nombre[(parametros)] IS
sentencia_select;
```

donde un parámetro tiene la forma:

nombre [IN] tipo [{:=| DEFAULT}expr]

Manejo de Cursores

- Un cursor se maneja mediante las operaciones:
  - OPEN: abre el cursor y guarda memoria
  - FETCH: recupera la siguiente fila del cursor
  - CLOSE: cierra el cursor

Manejo de Cursores

- Y mediante los atributos:
  - %FOUND:TRUE si el último FETCH recupero filas y FALSE sino
  - %NOTFOUND: opuesto de %FOUND
  - %ISOPEN: TRUE si el cursor esta abierto y FALSE sino
  - %ROWCOUNT: cantidad de filas ya procesadas por el cursor.

Manejo de Cursores

- Cuando se abre un cursor se ejecuta la consulta y se identifica el conjunto de filas resultado.
- Para consultas definidas con FOR UPDATE se lockean las filas involucradas
- Un cursor mantiene consistencia de lectura al momento en que se abre
- Los parámetros se pasan al abrir el cursor.

Manejo de Cursores

- El fetch recupera las filas de la consulta de a una por vez y asigna el resultado en las variables indicadas
- Cerrar un cursor implica deshabilitarlo.

Manejo de Cursores

- Es posible simplificar la codificación utilizando un FOR de cursores
- La sintaxis es la siguiente:

```
FOR variable IN cursor LOOP
        sequencia_de_sentencias;
END LOOP;
```

Manejo de Cursores

- FOR de cursores declara implícitamente la variable de tipo cursor %ROWTYPE;
- Automáticamente:
  - abre el cursor
  - en cada paso asigna la fila actual a la variable
  - al terminar de recorrer las filas, cierra el cursor

Packages

- Un paquete (*package*) es un objeto de la base de datos que agrupa tipos, objetos y subprogramas PL/SQL lógicamente relacionados.
- Tienen dos partes:
  - Especificación(*specification*):interfase con las aplicaciones
  - Cuerpo (*body*): implementa la especificación

Packages

- Ventajas:
  - modularidad
  - diseño de aplicaciones más fácil
  - ocultamiento de información
  - mayor funcionalidad

Packages

- Los paquetes no pueden ser:
  - invocados
  - anidados
  - parametrizados
- Lo que se invoca, parametriza o anida son los procedimientos y funciones que contiene

Packages

- Especificación

```
CREATE PACKAGE nombre AS
  --declaraciones de objetos y tipo
        públicos
  --especificaciones de subprogramas
END [nombre]
```

Packages

- Cuerpo

```
CREATE PACKAGE BODY nombre AS
 --declaraciones de objetos y tipos
            privados
--cuerpos de subprogramas
[BEGIN
--sentencia de inicialización]
END [nombre]
```

Packages

- La especificación mantiene declaraciones públicas, visibles a las aplicaciones
- Los objetos declarados en la especificación son públicos:
  - pueden ser vistos por cualquier aplicación
  - pueden ser vistos por cualquier componente del paquete.

Packages

- El cuerpo mantiene declaraciones privadas y detalles de implementación,ocultas a las aplicaciones
- Es opcional: si el paquete solo define tipos y objetos es innecesario
- Contiene un bloque de sentencias de inicialización que se ejecuta por sesión, la primera vez que se invoca algo del paquete.

Packages

- La especificación y el cuerpo son objetos diferentes en la base de datos
- Es posible alterar e incluso borrar el cuerpo de un paquete sin alterar la especificación
- Las aplicaciones que invocan tipos, objetos y subprogramas siguen compilando aún cuando el cuerpo no este correcto

Packages

- Los objetos declarados en la especificación son públicos (*public*): pueden ser accedidos desde cualquier aplicación
- Los objetos declarados en el cuerpo son privados (*private*): sólo pueden ser accedidos por otras componentes del cuerpo del paquete

Packages

- Para referenciar tipos, objetos y subprogramas de un paquete se prefijan con el nombre del paquete y un punto:

```
nombre_paquete.nombre_tipo
nombre_paquete.nombre_objeto

nombre_paquete.nombre_subprograma
```

Packages

- Existe un paquete denominado *standard* que define el ambiente PL/SQL
- Este paquete declara tipos,objetos y subprogramas que están automáticamente disponibles a cualquier programa PL/SQL
- Por ejemplo, se define en este paquete la función ABS (valor absoluto)

Packages

- Los contenidos del paquete STANDARD son visibles directamente
- Por lo tanto, no es necesario prefijar a la componente con STANDARD
- Si se redefine alguna componente a nivel local, esto sobreescribe al comportamiento del paquete STANDARD

Packages

- Existen varios paquetes predefinidos por Oracle:
  - DBMS_STANDARD
  - DBMS_SQL
  - DBMS_ALERT
  - DBMS_OUTPUT
  - DBMS_PIPE
  - UTIL_FILE

Cursores empaquetados

- Es posible separar la especificación de un cursor de su implementación en un paquete
- De esta manera, es posible cambiar la implementación del cursor sin cambiar su interfase
- Esto da mayor flexibiliad a las aplicaciones

Cursores empaquetados

- Para declarar un cursor en la especificación:

```
CREATE PACKAGE nombre AS
    /*especificación del cursor/*
CURSOR nombre RETURN tipo_dato;
...
END nombre;
```

Cursores empaquetados

- Para implementar el cursor en el cuerpo del paquete:

```
CREATE PACKAGE nombre AS
    /*cuerpo del cursor/*
CURSOR nombre RETURN tipo_dato IS
    sentencia_select;
...
END nombre;
```

Sobrecarga de Operadores

- PL/SQL permite tener varios procedimientos o funciones con el mismo nombre al mismo nivel
- Esto se denomina sobrecarga de operadores (*overloading*)
- Al mismo nivel significa
  - como procedimientos o funciones hermanos dentro del área de declaraciones del bloque
  - dentro de un paquete

Sobrecarga de Operadores

- PL/SQL determina cual de ellos ejecutar de acuerdo al tipo de los parámetros

- procedimiento p
  - procedure p(a number) is.....end;
  - procedure p(a varchar2) is.....end;

Sobrecarga de Operadores

- La sobrecarga se puede realizar a nivel de tipos diferentes, pero no de tipos del mismo conjunto (ej: NUMBER y FLOAT)
- En el paquete STANDARD existen operaciones sobreargadas (ej: TO_CHAR)

Estructuras de Datos

- PL/SQL permite la construcción de tipos definidos por el usuario por medio del constructor TYPE
- El nuevo tipo declarado pasa a ser un tipo más del lenguaje
- Es posible declarar variables de ese tipo de la misma forma en que se declara una variable de un tipo escalar

Estructuras de Datos

- La declaración de tipos se realiza en la zona de declaraciones, previamente a la declaración de variable
- PL/SQL provee tres constructores de nuevos tipos:
  - TABLE
  - RECORD
  - REF CURSOR

Estructuras de Datos

- Los objetos de tipo TABLE se denominan tablas PL/SQL
- Estan compuestas de filas y poseen una clave primaria
- No es posibe manipularlas a traves de sentencias SQL. Se acceden como arreglos a través de la clave primaria

Estructuras de Datos

- Una tabla es una colección ordenada de elementos del mismo tipo
- Cada elemento posee un índice numérico que indica su posesión exacta en la tabla
- No poseen tamaño fijo, cota inferior ni superior
- No requieren que las posiciones sean consecutivas (sparsity)

Estructuras de Datos

- Para declarar un tipo tabla, la sintaxis es:

```
TYPE nombre_tipo IS TABLE OF
   tipo_dato
[NOT NULL] INDEX BY BINARY INTEGER;
```

donde tipo_dato puede ser

- un tipo escalar

Estructuras de Datos

- Para referenciar un elemento de una tabla PL/SQL se debe acceder por el índice usando la sintaxis:

```
variable_de_tipo_tabla (posición);
```

Estructuras de Datos

- Es posible asignar una tabla a otra:

```
variable_de_tipo_tabla1:=variable_de_tipo_tabla2;
```

- También es posible asignar un valor a una posición determinada de la tabla:

```
variable_de_tipo_tabla(posicion):=expresion;
```

Estructuras de Datos

- Una tabla tiene los siguientes atributos:
  - EXISTS(n): devuelve TRUE si el n-ésimo elemento de la tabla existe
  - COUNT: devuelve la cantidad de elementos de la tabla
  - FIRST: devuelve el primer elemento de la tabla (aquel con menor índice)
  - LAST: devuelve el último elemento de la tabla

Estructuras de Datos

- PRIOR(n): devuelve el número de índice del elemento previo al n-ésimo elemento
- NEXT(n): devuelve el número de índice del elemento posterior al n-ésimo elemento
- DELETE: borra elementos de 3 formas:
  - DELETE: borra todos los elementos de la tabla
  - DELETE(n): borra el n-ésimo elemento
  - DELETE(m,n): borra todos los elementos en el rango de índices n..m

Estructuras de Datos

- Para aplicar un atributo a una tabla, se utiliza la notacion:

```
variable_de_tipo_tabla.atributo
```

- Si se accede por un índice a un elemento no definido se produce la excepcion

NO_DATA_FOUND

Estructuras de Datos

- Un registro es una colección de objetos de diferentes tipos agrupados bajo un mismo nombre
- Dos formas de definir variables de tipo registro:
  - implícitamente a traves de %ROWTYPE
  - explícitamente mediante la declaración de un tipo RECORD por medio de TYPE

Estructuras de Datos

- Para declarar un tipo registro se utiliza la sintaxis:

```
TYPE nombre_tipo IS RECORD
  (campo[,campo....]);
```

donde campo tiene la forma

```
nombre tipo_dato[[NOT NULL](:=|
  DEFAULT) expr
```

Estructuras de Datos

- Es posible construir registros anidados: un campo puede ser a su vez un registro
- Para referenciar un campo de registros se utiliza la sintaxis:

variable_de_tipo_registro.nombre_campo

Estructuras de Datos

- Se puede asignar un registro a otro:

```
variable_de_tipo_registro1:=variable_de_tipo_registro2;
```

- Se puede asignar un valor a un campo determinado del registro:

```
variable_de_tipo_registro.campo:=expresion;
```

Estructuras de Datos

- Se pueden declarar variables de cursor
- Estas variables, como los cursores, apuntan a la fila actual del área de trabajo asociada
- Diferencia con los cursores:
  - los cursores son estáticos
  - las variables son dinámicas no estan asociadas a una consulta específica
- Es posible abrir una variable de cursor

Estructuras de Datos

- Las variables de cursor son como punteros: guardan la dirección de memoria de un cursor
- Una variable de cursor tiene tipo REF CURSOR
- Están disponibles en cualquier máquina PL/SQL, ya sea un cliente o servidor

Estructuras de Datos

- Se utilizan para pasar resultados de consultas entre el servidor y diferentes aplicaciones cliente
- Un área de trabajo permanece accesible mientras haya una variable de cursor apuntando hacia ella
- No hay limitaciones de uso entre clientes y servidor

Estructuras de Datos

- Para declarar un tipo REF CURSOR se utiliza la sintaxis:

```
TYPE nombre IS REF CURSOR [RETURN tipo]
```

donde tipo representa un tipo registro

- declarado implícitamente por %ROWTYPE
- declarado explícitamente

Estructuras de Datos

- Si se declara un tipo de retorno el tipo es fuerte (strong), en otro caso es débil (weak)
- Un tipo fuerte da mayor seguridad y un tipo débil da mayor flexibilidad
- Si se producen errores de tipos se dispara la excepción INVALID_CURSOR

Estructuras de Datos

- Es posible manipular una variable de tipo cursor con las sentencias:
  - OPEN cursor FOR sentencia
  - FETCH cursor INTO variable,..,variable
  - CLOSE cursor
- A una variable de tipo cursor no se le pueden pasar parámetros

Estructuras de Datos

- Es posible manipular una variable de tipo cursor con las sentencias:
  - OPEN cursor FOR sentencia
  - FETCH cursor INTO variable,..,variable
  - CLOSE cursor
- A una variable de tipo cursor no se le pueden pasar parámetros

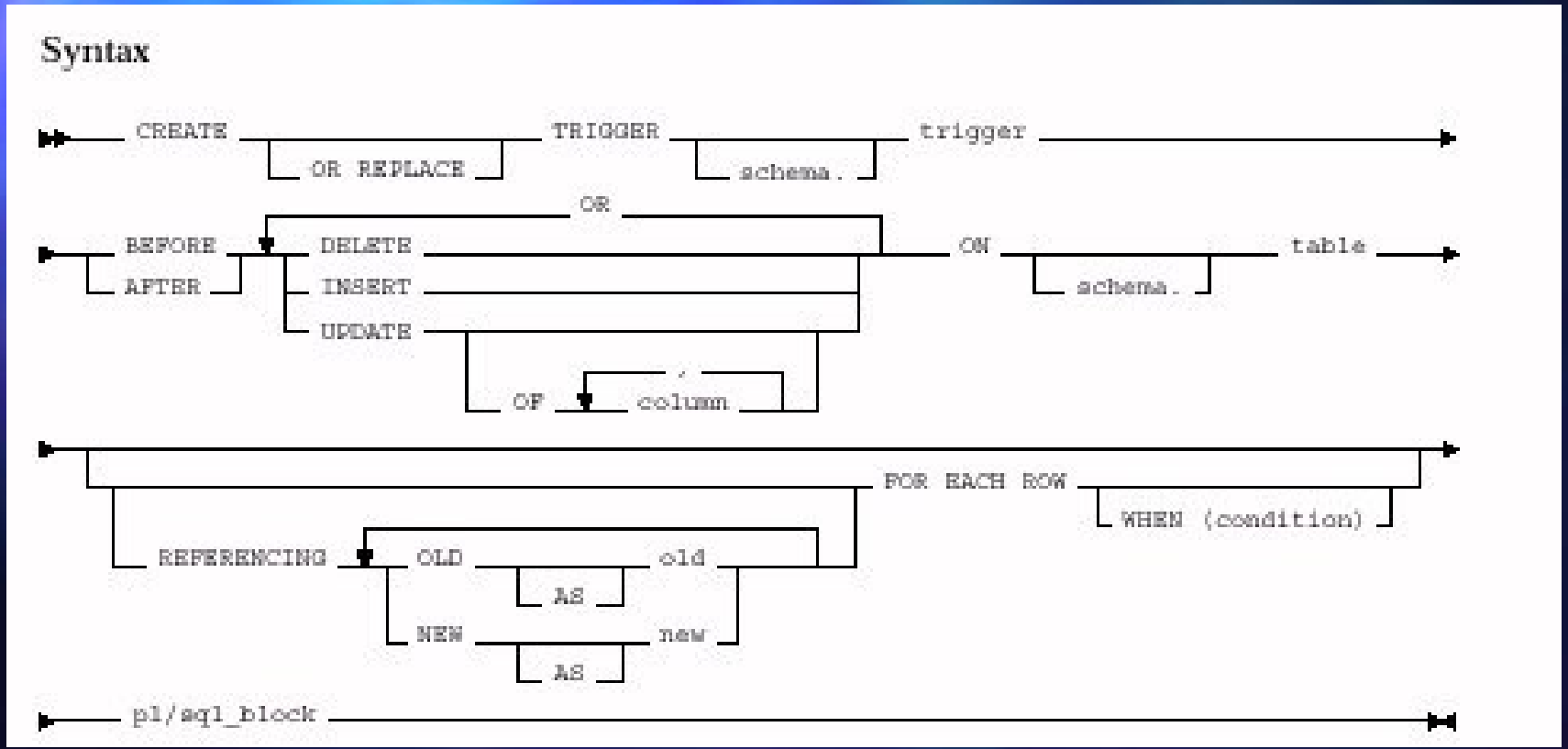
Triggers de la Base de datos

- Un trigger es un bloque PL/SQL asociado con una tabla
- Oracle ejecuta automáticamente un trigger cuando se realiza una operación SQL específica en la tabla
- Un trigger es un objeto almacenado por lo que debe ser creado usando la sentencia:

  CREATE TRIGGER

Triggers de la Base de datos

- La sintaxis de la sentencia CREATE TRIGGER es:



Triggers de la Base de datos

- BEFORE: indica que el trigger se dispara antes de ser realizada la operación
- AFTER: indica que el trigger se dispara después de realizada la operación
- La operación puede ser DELETE,INSERT o UPDATE
- En UPDATE se puede pedir que se dispare solo cuando cambian las columnas dadas

Triggers de la Base de datos

- Para referenciar a los valores nuevos de la fila que está siendo modificada o insertada se utiliza:NEW.columna
- Para referenciar a los valores viejos de la fila que está siendo modificada o borrada se utiliza :OLD.columna
- Tanto NEW como OLD se pueden nombrar por medio de REFERENCING

Triggers de la Base de datos

- Un trigger puede ser disparado:
  - una vez para toda la sentencia (default)
  - una vez para cada fila involucrada en la sentencia (FOR EACH ROW)
- OLD y NEW se utilizan en triggers de fila
- Es posible pedir que el trigger se dispare solo cuando se cumple una condición mediante la cláusula WHEN

Triggers de la Base de datos

- Aplicaciones de triggers:
  - Auditoría sofisticada
  - Validaciones no estructurales
  - Derivación automática de valores de columnas
  - Seguridad más compleja
  - Tablas replicadas
- Un trigger puede ser habilitado(ENABLE) o deshabilitado (DISABLE)