CURSO DE FUNDAMENTOS Y ADMINISTRACION DE DBMS

UBUNTU 10.4

vmware

Java

PostgreSQL

Windows

php

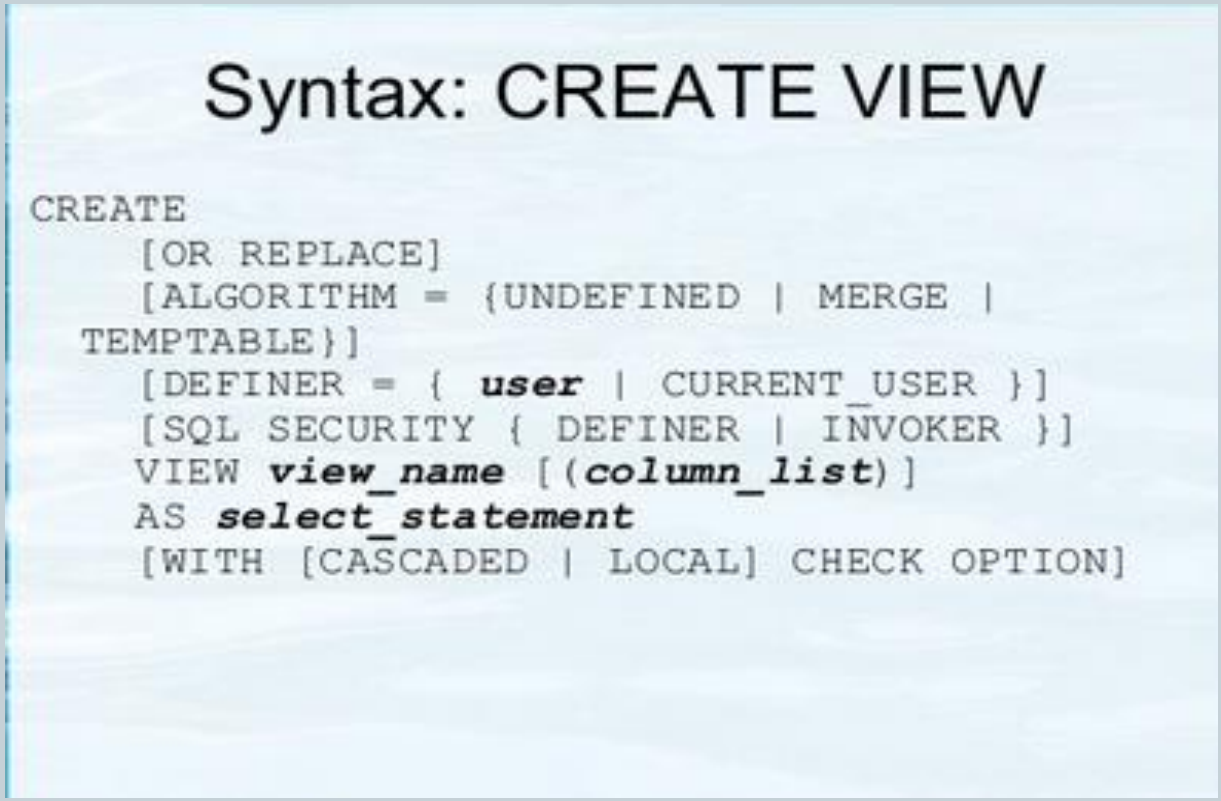
The Apache Software Foundation

SYBASE

INSTRUCTOR: RUDY SALVATIERRA RODRIGUEZ

SQL: CREACION DE VISTAS

- Representacion logica de un subconjunto de datos obtenidas de una o mas tablas.



SQL: CREACION DE VISTAS

```
CREATE OR REPLACE VIEW vista1 AS
 SELECT d.nombre_doc AS nombre,
   d.apellido_pat AS dasd, m.nombre_mat AS
   asdasd
 FROM docente d, doc_mat dm, materia m
 WHERE d.codigo_doc = dm.codigo_doc AND
dm.codigo_mat = m.codigo_mat AND
d.nombre_doc= 'Carlos‘ ;
```

SQL: CREACION DE PROCEDIMIENTOS

- **DEFINICION:**
  - Un procedimiento almacenado es un programa almacenado físicamente en una base de datos.
  - Habitualmente los procedimientos almacenados se escriben en un lenguaje de bases de datos del propietario.
  - No devuelven valor alguno
  - Generalmente usa variables de sesión, anda modificando dichas variables
  - Cuando se hace referencia a una variable de session se lo hace con @[nombre variable]=[valor]

SQL: CREACION DE PROCEDIMIENTOS

- Sintaxis de un procedimiento

```
CREATE PROCEDURE [nombre_procedimiento](parametros)
BEGIN
        [Instrucciones]
 END
```

En mysql se llama a un procedimiento con la function CALL

```
CALL mysp();
```

SQL: CREACION DE PROCEDIMIENTOS EJEMPLO

```
Set @variable=10;
CREATE PROCEDURE cambia_valor(in parametro
   integer)
  BEGIN
        set @variable= parametro;
 END

CALL cambia_valor (2) ;
```

SQL: CREACION DE PROCEDIMIENTOS EJEMPLO

Un procendimiento que cambie el valor variable var a uno si el parametro es 1 y si el valor de la variable var sera cero

Set @var=‘cero’;

CREATE PROCEDURE procedure1 (IN parametro TEXT)

BEGIN

IF parametro = ‘uno’ THEN

SET @var = ‘uno';

ELSE

SET @var = ‘cero';

END IF;

END

SQL: CREACION DE PROCEDIMIENTOS EJEMPLO

Procedimiento que inserte un parametro para poder introducir un dato en la tabla genero

```
CREATE PROCEDURE procedure1 (IN parametro
   TEXT)
BEGIN
        insert into genero(nombre_genero)
        values (parametro);
END
```

SQL: CREACION DE PROCEDIMIENTOS EJEMPLO

Procedimiento que genera un reporte que guarda el resultado en una variable out par

```
CREATE PROCEDURE simpleproc (OUT param1 INT)
BEGIN
SELECT COUNT(*) INTO param1 FROM t;
END

CALL simpleproc(@a);
SELECT @a;
```

SQL: CREACION DE PROCEDIMIENTOS EJEMPLO

- Estructura de control WHILE

```
CREATE PROCEDURE dowhile()
BEGIN
        DECLARE v1 INT DEFAULT 5;
        WHILE v1 > 0  DO
                SET v1 = v1 - 1;
                set @valor=v1;
        END WHILE;
END
```

SQL: CREACION DE PROCEDIMIENTOS EJEMPLO

- Ejercicio
  - Crear un procedimimiento que permita modificar la variable de session @resultado, este procedimiento reciba 2 parametros, uno de tipo entero y otro de tipo text, donde :
    - Si el parámetro 1 es mayor al parámetro 2 entonces con 1 while decremetar hasta llegar a ser igual y poner el valor final en la variable @resultado
    - Si el parametro 2 es mayor al parametro 1 entonces modificar el genero musical por musicales

SQL: CREACION DE FUNCIONES EN MYSQL

- Sintaxis de una funcion

```
CREATE FUNCTION [nombre_funcion]() RETURNS
   [tipo_de_retorno]
BEGIN
   [bloque declare]
   [bloque de codigo]
    RETURN  [dato_de_retorno]
END
```

SQL: CREACION DE FUNCIONES EN MYSQL EJEMPLO

- Función que me muestra el tradicional ‘Hola Mundo’

```
CREATE FUNCTION holaMundo() RETURNS
VARCHAR(20)
BEGIN

 RETURN 'HolaMundo';

END
```

SQL: CREACION DE FUNCIONES EN MYSQL EJEMPLO

- Función que concatena el nombre como parametro pasado dado con Hola

```
CREATE FUNCTION hola (s CHAR(20)) RETURNS CHAR(50)

BEGIN

   RETURN CONCAT('Hello, ',s,'!');

END
```

SQL: CREACION DE FUNCIONES EN MYSQL EJEMPLO

```
CREATE FUNCTION ContarLetra(letra char, cadena text) returns int
begin
        declare i int;
        declare contador int;
        declare car char;
        set contador=0;
        set i=length(cadena);
        while i>0 do
        set car=substring(cadena,i,1);
                if car=letra then
                        set contador=contador +1;
                end if;
    set i=i-1;
        end while;
RETURN contador;
end
```

SQL: CREACION DE FUNCIONES EN MYSQL EJEMPLO

- select ord('A'); //muestra el codigo ASCII del caracter
- select char(65,66,67); //muestra la letra de esos valores
- select concat('Hola,',' ','como esta?'); //concatena 2 cadenas
- select concat_ws('-','Juan','Pedro','Luis');//concatena
- select find_in_set('hola','como esta,hola,buen dia'); //muestra el numero de palabra donde esta hola
- select length('Hola'); //muestra el tamano del texto
- select replace('xxx.mysql.com','x','w');//reemplaca x por w
- select lower('HOLA ESTUDIAnte'); //convierte la cadena a minisculas
- select upper('HOLA ESTUDIAnte');//convierte la cadena a myusculas

PL/PGSQL: LENGUAJE PROCEDURAL DE POSTGRES

- PL/pgSQL provee un lenguaje de programación especial dentro el SQL.
- Este lenguaje procedural ofrece la potencia a los desarrolladores en ves de solo consultas **select**, **insert**, **delete**, etc.
- Resuelve problemas complejos que solo con sentencias SQL no podrían resolverse.
- Con PL/pgSQL es posible construir funciones, operadores y triggers (disparadores).

PL/PGSQL: LENGUAJE PROCEDURAL DE POSTGRES

- **COMO HABILITAR EL LENGUAJE**

Para Habilitar el lenguaje PL/pgSQL en una base de datos

1. Primero Verificamos si el lenguaje plpgsql esta habilitado en nuestra BD, en este caso prueba. Con el siguiente comando

```
prueba=> select * from pg_language;
```

2. Si no existe añadir con el siguiente comando desde la session del usuario postgres con el siguiente comando

a. $createlang plpgsql –d BD_a_Habilitar_PL/pgSQL
b. prueba=>select * from pg_language;

PL/PGSQL: ESTRUCTURA DEL LENGUAJE

- **Funciones en PL/pgSQL**

```
CREATE  [OR REPLACE]  FUNCTION nombre_funcion
(tipo_argumento, ... ) RETURNS tipo_retorno AS
‘
        CUERPO DE LA FUNCION
’
LANGUAGE lang_name;
```

nombre_funcion = cualquier nombre. Menos palabras reservadas.

tipo_agumento = cualquier tipo de dato, solo poner el tipo; int, char, text, etc.

tipo_retorno = cualquier tipo de dato, solo poner el tipo; int, char, text, etc.

lang_name = ***plpgsql***

PL/PGSQL: CUERPO DE LA FUNCION

- **Estructura del CUERPO DE LA FUNCION**

```
[ DECLARE
        … Declaración de variables …
]
BEGIN
        [ DECLARE
                … Declaración de variables …
        ]
        BEGIN
                … Instrucciones …
        END;
        … Instrucciones …
        RETURN { tipo_retorno | NULL };
END;
```

Bloque Opcional

Bloque Opcional

Sub bloque principal

Bloque principal

PL/PGSQL: CUERPO DE LA FUNCION

- **Retorno en una función**
  - Cada función definida en PL/pgSQL debe terminar con la sentencia RETURN.
  - Pueden existir varios **return**'s, pero solo uno de ellos se ejecutara

Sintaxis:

RETURN expresión;

Ej.:

Ejemplos anteriores

PL/PGSQL: CUERPO DE LA FUNCION

- **Borrando funciones**
  - Para borrar funciones se debe especificar aparte del nombre de la función los argumentos que recibe, pero solo los tipos de los argumentos.

Sintaxis:

DROP FUNCTION funcion(tipo_argumento, …);

Ej.(drop_funciones.txt)

```
drop function nada();
drop function hoy();
drop function retorno(integer);
drop function llamada(integer);
drop function multiplicar(integer, integer);
```

PL/PGSQL: CUERPO DE LA FUNCION

- Llamando a una función

  1). Desde un sentencia ***select***

  Sintaxis:

  prueba=>select nombre_funcion(argumentos);

  2). Desde otra función

  Sintaxis 1:

  begin

  PERFORM nada(); --esto para obviar resultados por la sentencia ***select***

  end;

  Sintaxis 2:

  begin

  resultado := (sumar(2,5)); --Recuperando el resultado

  end;

PL/PGSQL: CUERPO DE LA FUNCION

- Ejemplo de llamadas a funciones

  1. funcion llamada(integer);
  2. funcion calcular(integer, integer);

- Comentarios en PL/pgSQL:

  1. --Comentario de una linea simple

  2.
  /*
  Comentario de bloque
  */

PL/PGSQL: CUERPO DE LA FUNCION

- Definición de variables, ALIAS FOR

```
DECLARE
        … Declaraciones …        --Toda variable debe declararse dentro
                                   el bloque DECLARE
BEGIN
        … Sentencias …
END;
```

Sintaxis:

```
nombre_variable tipo_dato;
nombre_variable tipo_dato := valor_inicial;
```

Ej. (declaracion_variables.txt)

```
nombre text := '';
PI CONSTANT real := 2.14;
parametro1 ALIAS FOR $1;
```

PL/PGSQL: CUERPO DE LA FUNCION

- **Pasando parámetros a las funciones**

  Los parámetros de las funciones se los accede de la forma $X, donde X es el numero de argumento comenzando dese 1 hasta el total de argumentos enviados a la función.

  EJ.(parametro.txt)

```
CREATE or replace FUNCTION PARAMETRO(int, int, text, char, int, text)
returns text as
‘
DECLARE
        ARG1 ALIAS FOR $1;
        ARG2 ALIAS FOR $2;
        ……
BEGIN
```

PL/PGSQL: CUERPO DE LA FUNCION

- **Sentencias de control**

```
IF condiciones THEN
        instrucciones;
END IF;
IF condiciones THEN
        sentencias;
ELSE
        sentencias;
END IF;

IF condiciones THEN
        sentencias;
ELSE IF condiciones THEN
                sentencias;
END IF;
```

PL/PGSQL: CUERPO DE LA FUNCION

- Ejemplo de una función que compara el mayor de 2 números

```
CREATE OR REPLACE FUNCTION mayor(numero1 integer,numero2 integer)
RETURNS text AS $$
DECLARE
        resultado text;

BEGIN
        resultado:=‘el resultado es:’;
        if numero1 > numero2 then
                raise notice 'el numero1 es el mayor';
                resultado:=resultado||numero1;
        ELSE
                raise notice 'el numero2 es el mayor';
                resultado:=resultado||numero2;
        end if;

        RETURN resultado;
END;
$$
LANGUAGE plpgsql;
```

PL/PGSQL: CUERPO DE LA FUNCION

- **Ejercicios de sentencias de control**

  Crear una función en PL/pgSQL para:

  1. Obtener el mayor de tres numero dados como parámetros

  2. Obtener la cadena mas grande dados dos cadenas como parámetros

  3. Que retorne verdadero/falso si un parámetro enviado a esa función es IS NULL

PL/PGSQL: CUERPO DE LA FUNCION

- **Bucles**

```
WHILE condicion LOOP
        instrucciones;
END LOOP;

FOR ind IN [ REVERSE ] desde..hasta LOOP
        Instrucciones;
END LOOP;
```

PL/PGSQL: CUERPO DE LA FUNCION

- **Ejercicios con bucles**

Crear una función para:

1. Verificar si un numero es primo
2. Realizar la misma función que el factorial de un numero dado como parámetro.
3. Una ves verificado insertar ese numero en una tabla llamado primo(integer), que antes deben crearlo

PL/PGSQL: CUERPO DE LA FUNCION

- **Manejo de mensajes/errores**

  Sintaxis:

  RAISE nivel-mensaje 'mensaje';

  3 niveles existentes:

  debug = Es registrado en los logs del servidor y la transaccion u operación no es afectado,

  notice = Es registrado en los logs del servidor y tambien enviado al cliente, solo es un mensaje de notificacion, no es consideardoun problema

  exception = Es registrado en los logs del servidor y la transaccion es abortado o interrumpido