




BASE DE DATOS II

FREDDY MACHACA MAMANI

The background is black with several thin white lines of varying lengths and orientations. A small cyan square is located in the upper right quadrant. The text '01' is rendered in a large, bold, cyan font, with the '0' having a pixelated or blocky appearance.

01

PARTE TEORICA

PROCEDURE MYSQL

Los procedimientos almacenados MySQL, también conocidos como Stored Procedure, se presentan como conjuntos de instrucciones escritas en el lenguaje SQL. Su objetivo es realizar una tarea determinada, desde operaciones sencillas hasta tareas muy complejas. Los procedimientos almacenados MySQL contienen una o más instrucciones SQL además de un procesamiento manipulador o lógico.



FUNCTION MYSQL

Las funciones almacenadas de MySQL nos permiten procesar y manipular datos de forma procedural de un modo muy eficiente. Podrás usarlas en las sentencias SQL independientemente del lenguaje de programación del servidor sobre el que se ejecuten las consultas.



FUNCTION STRUCTURE

```
create function suma_dos_numeros(num1 int,num2 int)
returns int as
begin
    declare respuesta int;
    set respuesta = num1+num2;

    return @respuesta;
end;
```

CREAR FUNCIÓN

```
create function suma_dos_numeros(num1 int,num2 int)
returns int as
begin
    declare respuesta int;
    set respuesta = num1+num2;
    return respuesta;
end;
```

MODIFICAR FUNCIÓN

```
create or replace function suma_dos_numeros(num1 int,num2 int)
returns int as
begin
    declare respuesta int;
    set respuesta = num1+num2;
    return respuesta;
end;
```

ELIMINAR FUNCIÓN

```
drop function edadMinima2;
```

CONCAT

La función CONCAT() suma dos o más expresiones juntas.

```
create function getParametros(par1 varchar(20), par2 varchar(20), par3 varchar(20))  
  returns varchar(60)  
begin  
  declare resultado varchar(60);  
  set resultado = CONCAT(par1, ' ', par2, ' ', par3);  
  return resultado;  
end;  
  
select getParametros( par1: 'pepito', par2: 'pep', par3: '50');
```

SUBSTRING

La función SUBSTRING() extrae una subcadena de una cadena (comenzando en cualquier posición).

```
CREATE OR REPLACE FUNCTION SUB_STRING (NOMBRE VARCHAR(50),POSICION INTEGER,LONGITUD INTEGER)
RETURNS TEXT
BEGIN
  DECLARE CADENA TEXT DEFAULT '';
  SET CADENA= SUBSTRING(NOMBRE,POSICION,LONGITUD);
  RETURN CADENA;
END;

SELECT SUBSTRING('DANIEL ORIVADO DA SILVA',5,10);
```

Output SUBSTRING('DANIEL OR... SILVA',5,10):varchar X

1 row

'SUBSTRING('DANIEL ORIVADO DA SILVA',5,10)'

1 EL ORIVADO

STRCMP

La función STRCMP() compara dos cadenas.

```
CREATE or replace FUNCTION CADENA(cadena TEXT,cadena1 text)
RETURNS text
```

```
BEGIN
```

```
    DECLARE mensaje text default '';
```

```
    CASE
```

```
        when mensaje=strcmp(cadena,cadena1) then
```

```
        if mensaje=0 then
```

```
            SET mensaje = ' cadenas iguales';
```

```
        end if;
```

```
        ELSE SET mensaje = 'cadenas distintas';
```

```
    END CASE;
```

```
    RETURN mensaje;
```

```
END;
```

```
SELECT CADENA( cadena: 'HOLA', cadena1: 'CHAU');
```

```
_STRING()
```

Output CADENA('HOLA','CHAU'):text ×

< < 1 row > > ↺ ■ ↻

SQL 'CADENA('HOLA','CHAU')'

1 cadenas distintas

CHAR LENGTH - LOCATE

La función CHAR_LENGTH() devuelve la longitud de una cadena (en caracteres).

```
CREATE OR REPLACE FUNCTION CONTAR_CADENA(TEXT0 VARCHAR(20))  
RETURNS TEXT  
BEGIN  
    DECLARE RESPUESTA VARCHAR(20) DEFAULT '';  
    SET RESPUESTA=CHAR_LENGTH(TEXT0);  
    RETURN RESPUESTA;  
END;  
SELECT CONTAR_CADENA( TEXT0: 'HOLA');
```

Output CONTAR_CADENA('HOLA');t

	CONTAR_CADENA('HOLA')
1	4

La función LOCATE() devuelve la posición de la primera aparición de una subcadena en una cadena.

```
CREATE OR REPLACE FUNCTION LOCATE_CADENA(SUBCADENA VARCHAR(20),CADENA VARCHAR(20))  
RETURNS TEXT  
BEGIN  
    DECLARE RESP INTEGER DEFAULT 0;  
    SET RESP=LOCATE(SUBCADENA,CADENA);  
    RETURN RESP;  
END;  
SELECT LOCATE_CADENA( SUBCADENA: 'C', CADENA: 'SOFT.COM');
```

FUNCIONES DE AGREGACION Y FUNCIONES POR EL USUARIOS

Las funciones de agregación en SQL nos permiten efectuar operaciones sobre un conjunto de resultados, pero devolviendo un único valor agregado para todos ellos. Es decir, nos permiten obtener medias, máximos, sobre un conjunto de valores. Funciones creados por el BDA se utiliza para definir una función de tabla, fila o escalar de SQL definida por el usuario. Una función escalar devuelve un solo valor cada vez que se invoca y en general es válida cuando una expresión SQL es válida.



PARAMETROS

PARAMETRO DE ENTRADA “IN”


Es el tipo de procedimiento por default, el procedimiento utiliza parámetros que pasan el argumento por valor hacia dentro del mismo.

PARAMETRO DE SALIDA “OUT”

El valor del parámetro se regresa hacia afuera de este para utilizarse ya sea en una variable u otra consulta.

PARAMETRO DE SALIDA Y ENTRADA “INOUT”

Los parámetros definidos funcionan en ambas direcciones.

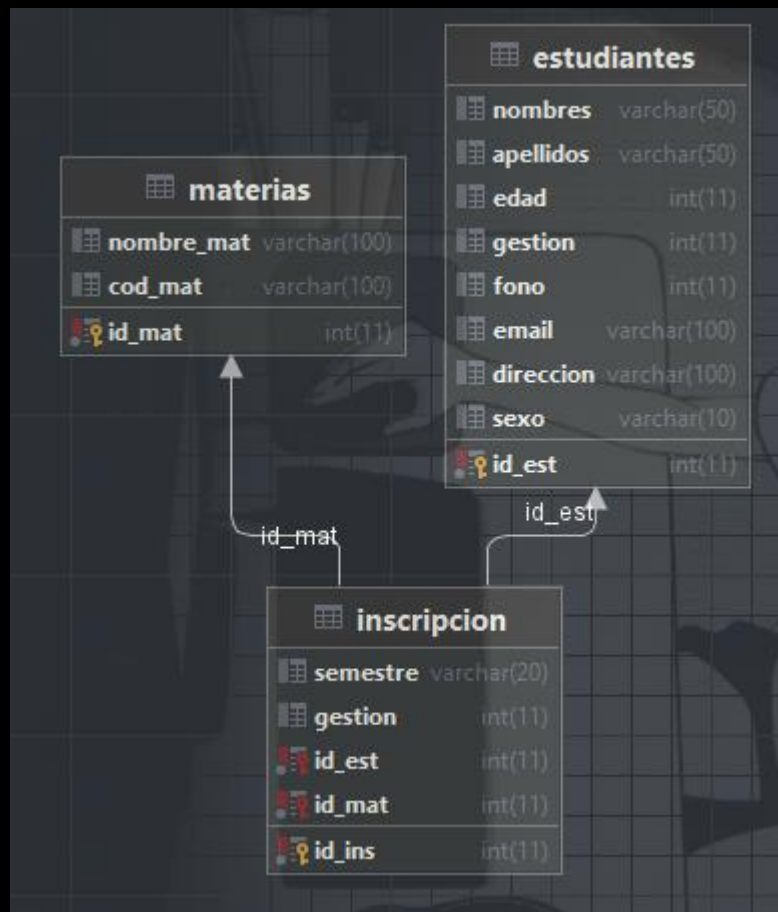
The background is black with several thin white lines of varying lengths and orientations. A small cyan square is located in the upper right quadrant. The text '02' is rendered in a large, cyan, pixelated font on the left side.

02

PARTE PRACTICA

02

MODELO E-R



02

CODIGO SQL

```
create table estudiantes
(
    id_est      int auto_increment primary key not null,
    nombres     varchar(50),
    apellidos   varchar(50),
    edad        int(11),
    gestion     int(11),
    fono        int(11),
    email       varchar(100),
    direccion   varchar(100),
    sexo        varchar(10)
);

create table materias
(
    id_mat      int auto_increment primary key not null,
    nombre_mat  varchar(100),
    cod_mat     varchar(100)
);

create table inscripcion
(
    id_ins      int auto_increment primary key not null,
    semestre    varchar(20),
    gestion     int(11),

    id_est      int not null,
    id_mat      int not null,

    foreign key (id_est) references estudiantes (id_est),
    foreign key (id_mat) references materias (id_mat)
);
```

02

ESTUDIANTES

REGISTROS MATERIA

INSCRIPCION

```
INSERT INTO estudiantes (nombres, apellidos, edad, fono, email, direccion, sexo)
VALUES ('Miguel', 'Gonzales Veliz', 20, 2832115, 'miguel@gmail.com', 'Av. 6 de Agosto', 'masculino'),
('Sandra', 'Mavir Uria', 25, 2832116, 'sandra@gmail.com', 'Av. 6 de Agosto', 'femenino'),
('Joel', 'Aubiriri Mondar', 30, 2832117, 'joel@gmail.com', 'Av. 6 de Agosto', 'masculino'),
('Andrea', 'Arias Ballesteros', 21, 2832118, 'andrea@gmail.com', 'Av. 6 de Agosto', 'femenino'),
('Santos', 'Montes Valenzuela', 24, 2832119, 'santos@gmail.com', 'Av. 6 de Agosto', 'masculino');
```

```
INSERT INTO materias (nombre_mat, cod_mat)
VALUES ('Introduccion a la Arquitectura', 'ARQ-101'),
('Urbanismo y Diseno', 'ARQ-102'),
('Dibujo y Pintura Arquitectonica', 'ARQ-103'),
('Matematica discreta', 'ARQ-104'),
('Fisica Basica', 'ARQ-105');
```

```
INSERT INTO inscripcion (semestre, gestion, id_est, id_mat)
values ('1er Semestre', 2018, 1, 1),
('2do Semestre', 2018, 1, 2),
('1er Semestre', 2019, 2, 4),
('2do Semestre', 2019, 2, 3),
('2do Semestre', 2020, 3, 3),
('3er Semestre', 2020, 3, 1),
('4to Semestre', 2021, 4, 4),
```

02

12. Crear una función que genere la serie Fibonacci.

- La función recibe un límite(number)
- La función debe de retornar una cadena.
- Ejemplo para **n=7. OUTPUT: 0, 1, 1, 2, 3, 5, 8,**
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
CREATE OR REPLACE FUNCTION fibonacci(limite INT)
RETURNS TEXT
DETERMINISTIC
BEGIN
    DECLARE fib1 INT DEFAULT 0;
    DECLARE fib2 INT DEFAULT 1;
    DECLARE fib3 INT DEFAULT 0;
    DECLARE str VARCHAR(255) DEFAULT '0,1,';

    IF limite = 1 THEN
        RETURN fib1;
    ELSEIF limite = 2 THEN
        RETURN CONCAT(fib1, fib2);
    ELSE
        WHILE limite > 2 DO
            SET fib3 = fib1 + fib2;
            SET fib1 = fib2;
            SET fib2 = fib3;
            SET limite = limite - 1;
            SET str = CONCAT(str, fib3, ',');
        END WHILE;
        RETURN str;
    END IF;
END;

select fibonacci( limite: 7);
```

Output fibonacci(7):text

1 row

fibonacci(7)

1 0,1,1,2,3,5,8,

02

13. Crear una variable global a nivel BASE DE DATOS.

- Crear una función cualquiera.
- La función debe retornar la variable global.
- Adjuntar el código SQL generado y una imagen de su correcto funcionamiento.

```
set @limit=7;
CREATE OR REPLACE FUNCTION fibonacci1()
RETURNS INT
DETERMINISTIC
BEGIN
    DECLARE fib1 INT DEFAULT 0;
    DECLARE fib2 INT DEFAULT 1;
    DECLARE fib3 INT DEFAULT 0;
    DECLARE str VARCHAR(255) DEFAULT '01';

    IF @limit = 1 THEN
        RETURN fib1;
    ELSEIF @limit = 2 THEN
        RETURN CONCAT(fib1, fib2);
    ELSE
        WHILE @limit > 2 DO
            SET fib3 = fib1 + fib2;
            SET fib1 = fib2;
            SET fib2 = fib3;
            SET @limit = @limit - 1;
            SET str = CONCAT(str, fib3);
        END WHILE;
        RETURN str;
    END IF;
END;

select fibonacci1();
```

Output fibonacci1():int(11) x

1 row

`fibonacci1()`

1 112358

02

14. Crear una función no recibe parámetros (Utilizar WHILE, REPEAT o LOOP).

- Previamente deberá de crear una función que obtenga la edad mínima de los estudiantes
 - La función no recibe ningún parámetro.
 - La función debe de retornar un número.(LA EDAD MÍNIMA).
- Si la edad mínima es **PAR** mostrar todos los pares empezando desde 0 a este ese valor de la edad mínima.

```
`paresImpares()`  
1 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24,
```

- Si la edad mínima es **IMPAR** mostrar descendentemente todos los impares hasta el valor 0.

```
create or replace function edadMinima2()  
returns TEXT  
begin  
    declare respuesta text default '';  
    declare limite int;  
    declare x int default 1;  
    select min(est.edad) into limite  
    from estudiantes as est;  
  
    if limite %2=0  
    then  
        set x=2;  
    end if;  
    while x<=limite do  
        set respuesta= concat(respuesta,x,',');  
        set x=x+2;  
    end while;  
    return respuesta;  
end;
```

```
select edadMinima2();
```

Output edadMinima2():text ×

< < 1 row > > | ↺ ■ ★

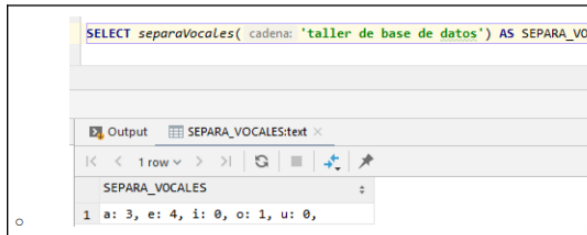
edadMinima2()

```
1 2, 4, 6, 8, 10, 12, 14, 16, 18, 20,
```

02

15. Crear una función que determina cuantas veces se repite las vocales.

- o La función recibe una cadena y retorna un TEXT.
- o Retornar todas las vocales ordenadas e indicando la cantidad de veces que se repite en la cadena.
- o Resultado esperado.

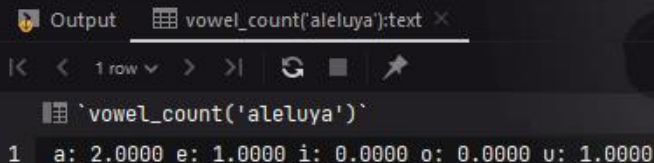


The screenshot shows a SQL query in a text editor: `SELECT separaVocales(cadena: 'taller de base de datos') AS SEPARA_VO`. Below the query, the output is displayed in a table with one row and one column named 'SEPARA_VOCALES'. The value in the row is '1 a: 3, e: 4, i: 0, o: 1, u: 0,'.

SEPARA_VOCALES
1 a: 3, e: 4, i: 0, o: 1, u: 0,

```
create or replace function vowel_count(str varchar(1024))
returns text
begin
    return concat(
        concat(' a: ', (LENGTH(str) - LENGTH(REPLACE(str, 'a', '')))/LENGTH('a')),
        concat(' e: ', (LENGTH(str) - LENGTH(REPLACE(str, 'e', '')))/LENGTH('e')),
        concat(' i: ', (LENGTH(str) - LENGTH(REPLACE(str, 'i', '')))/LENGTH('i')),
        concat(' o: ', (LENGTH(str) - LENGTH(REPLACE(str, 'o', '')))/LENGTH('o')),
        concat(' u: ', (LENGTH(str) - LENGTH(REPLACE(str, 'u', '')))/LENGTH('u'))
    );
end;

select vowel_count( str: 'aleluya');
```



The screenshot shows the output of the SQL function in a table with one row and one column named 'vowel_count('aleluya')'. The value in the row is '1 a: 2.0000 e: 1.0000 i: 0.0000 o: 0.0000 u: 1.0000'.

vowel_count('aleluya')
1 a: 2.0000 e: 1.0000 i: 0.0000 o: 0.0000 u: 1.0000

02

16. Crear una función que recibe un parámetro INTEGER.

- La función debe de retornar un texto(**TEXT**) como respuesta.
- El parámetro es un valor numérico **credit_number**.
- Si es mayor a 50000 es **PLATINIUM**.
- Si es mayor igual a 10000 y menor igual a 50000 es **GOLD**.
- Si es menor a 10000 es **SILVER**
- La función debe retornar indicando si ese cliente es PLATINIUM, GOLD o SILVER en base al valor del credit_number.

```
create or replace function function_credit1(creditNumber int)
returns text
begin
    declare respuesta text default '';
    case
        when creditNumber > 50000 then set respuesta = 'PLATINIUM';
        when creditNumber <= 50000 and @creditNumber >= 10000 then set respuesta = 'GOLD';
        when creditNumber < 10000 then set respuesta = 'SILVER';
        else set respuesta = 'caso desconocido';
    end case;
    return respuesta;
end;

select function_credit1( creditNumber: 50001);
```

Output	
function_credit1(50001):text	
1 row	
`function_credit1(50001)`	
1	PLATINIUM

02

17. Crear una función que reciba un parámetro TEXT

- En donde este parámetro deberá de recibir una **cadena** cualquiera y retorna un **TEXT** de respuesta.
- Concatenar **N** veces la misma cadena reduciendo en uno en cada iteración hasta llegar a una sola letra.
- Utilizar REPEAT y retornar la nueva cadena concatenada.
- Considerar la siguiente imagen:

LETTERS	
1	dbaii, baii, aii, ii, i,

```
CREATE OR REPLACE FUNCTION REDUCIR_CADENA (PALABRA TEXT)
RETURNS TEXT
BEGIN
    DECLARE RESPUESTA TEXT DEFAULT '';
    DECLARE CONTADOR INTEGER DEFAULT CHAR_LENGTH(PALABRA);
    DECLARE AUX INTEGER DEFAULT 1;
    DECLARE AUX2 INTEGER DEFAULT CHAR_LENGTH(PALABRA);

    REPEAT
        SET RESPUESTA= CONCAT(RESPUESTA,',',SUBSTRING(PALABRA,AUX,AUX2));
        SET CONTADOR=CONTADOR-1;
        SET AUX=AUX+1;

    until CONTADOR <=0 end repeat;

    RETURN RESPUESTA;

end;

SELECT REDUCIR_CADENA( PALABRA: 'ABRAHAM');
```

UCIR_CADENA()

Output	
REDUCIR_CADENA('ABRAHAM'):text	
1	,ABRAHAM,BRAHAM,RAHAM,AHAM,HAM,AM,M