

# GUIA-EXAMEN-LABORATORIO-IISSI1.pdf



saracordobaa



Introducción a la Ingeniería del Software y los Sistemas de Información I



2º Grado en Ingeniería Informática - Tecnologías Informáticas



Escuela Técnica Superior de Ingeniería Informática  
Universidad de Sevilla

MÁSTER EN

Energías  
Renovables

MADRID

Ahora  
**25%**  
DE DESCUENTO

**EOI** Escuela de  
organización  
industrial

Estudia el máster líder en  
energías renovables según el

**Ranking 250  
Masters de:**  
EL MUNDO [Expansión](#)

[Info y descuentos](#)





# 10 AÑOS REUNIENDO AMIGOS



## Guía EXAMEN LABORATORIO IISSIA

### PREPARACIÓN DEL ENTORNO

- 1- Se inicia iissi-root contraseña: iissi \$ root
- 2- Se crea la base de datos y le damos permisos a iissi\_user: iissi \$ user
- 3- Iniciamos sesión iissi\_user. Cargar archivos SQL. Tenemos las tablas creadas y los datos.

### ESTRUCTURA EXAMEN:

- 1- Creación de tablas LAB2
  - 2- Consultas SQL LAB4
  - 3- Procedimiento LAB5
  - 4- Trigger LAB5/2
- Yo

### TABLAS

- \* PK: primary key → campo que identifica cada registro de forma única. No puede repetirse ni ser nula. Solo una PK por tabla.
- \* FK: foreign key → campo que apunta la PK de otra tabla. Se usa para relacionar tablas.
- \* AU: alternative key → otra clave que podría haber sido PK pero no fue elegida. Única y no nula.

#### • ASOCIACIÓN [1..N]



Departamentos (departamento\_id, departamento\_nombre, presupuesto, ciudad)

PK (departamento\_id)

AU (departamento\_nombre)

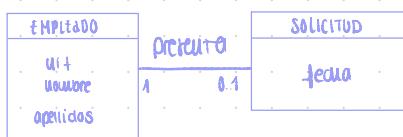
Empleados (empleado\_id, departamento\_id, ui, nombre, apellido)

PK (empleado\_id)

FU (departamento\_id)

AU (ui)

#### • ASOCIACION [1..1]



Empleados (empleado\_id, ui, nombre, apellido)

PK (empleado\_id)

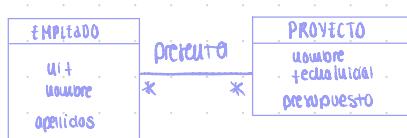
AU (ui)

Solicitudes (solicitud\_id, empleado\_id, fecha)

PK (solicitud\_id)

FU (empleado\_id)

#### • ASOCIACION [N..M]



Empleados (empleado\_id, ui, nombre, apellido)

PK (empleado\_id)

AU (ui)

Proyectos (proyecto\_id, nombre, fecha\_lanzamiento, presupuesto)

PK (proyecto\_id)

Empleados Proyectos (empleado\_proyecto\_id, empleado\_id, proyecto\_id)

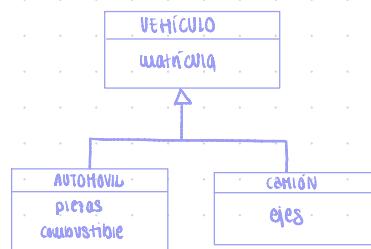
PK (empleado\_proyecto\_id)

FU (empleado\_id)

FU (proyecto\_id)

AU (empleado\_id; proyecto\_id)

#### • GENERALIZACIÓN:



Vehiculos (vehiculo\_id, matricula)

PK (vehiculo\_id)

AU (matricula)

Camiones (camion\_id, vehiculo\_id, ejes)

PK (camion\_id)

FU (vehiculo\_id)

Automoviles (automovil\_id, vehiculo\_id, piezas, combustible)

PK (automovil\_id)

FU (vehiculo\_id)

## ESTRUCTURA GENERAL

## CREAR TABLA

```

DROP TABLE IF EXISTS NombreTabla; ← en la cabecera
CREATE TABLE NombreTabla (
    nombre1 tipoDato restricciones
    • PK → nombre tipoDato PRIMARY KEY
    FOREIGN KEY ( nombre ) REFERENCES NombreTabla ( id )
    ON DELETE CASCADE → si se borra el nombre, se borra tambien en la tabla relacionada
    ON UPDATE CASCADE → si se actualiza en la tabla relacionada, se actualiza aqui tambien
    UNIQUE ( nombre FU )
);
  
```

- AUTO\_INCREMENT : se usa en campos numéricos (id) para que genere automáticamente un nuevo valor
- UNIQUE: los valores deben ser únicos
- NOT NULL: no puede estar vacío
- CHTC
- DEFAULT TRUE / FALSE para Booleano

- |                    |            |
|--------------------|------------|
| • INT              | • DATE     |
| • DATE             | • TIME     |
| • VARCHAR ( 255 )  | • DATETIME |
| • TEXT             | • YEAR     |
| • BOOLEAN          |            |
| • DECIMAL ( 10,2 ) |            |

# CONCEPTOS

## FUNCIONES

### 1. DE FECHA Y HORA

- CURDATE() → fecha actual (YYYY-MM-DD)
- NOW() → fecha y hora actual (YYYY-MM-DD HH:MM:SS)
- YEAR(fecha) → año de una fecha
- MONTH(), DAY(), HOUR(), MINUTE(), SECOND()
- TIMESTAMPDIFF(unidad, f1, f2) → diferencia de dos fechas en años, días, ... → TIMESTAMPDIFF(YEAR, y1, y2)

### 2. MATEMÁTICAS

- ABS(x) → valor absoluto
- ROUND(x, n) → redondeo a n decimales
- FLOOR(x) → redondea hacia abajo
- CEIL(x) → hacia arriba
- MOD(x,y) → resto de una división

### 3. TEXTO

- LENGTH(texto) → devuelve la longitud del texto en bytes
- CHAR\_LENGTH(t) → en caracteres (los que ves)
- UPPER(t) / LOWER(t) → a mayus. / minus.
- CONCAT(a,b) → une a y b

### DECLARE

A veces es necesario crear variables temporales para guardar datos.

• DECLARE edadCliente INT; → crea una variable llamada edadCli que puedes usar dentro del trigger para guardar su número.

### INNER JOIN Y FROM

- FROM → indica de qué tablas sacar los datos
- INNER JOIN → para unir dos tablas cuando tienen una relación (por fu p ej)

SELECT Clientes.usuNombre, Pedidos.id

FROM Clientes

INNER JOIN Pedidos ON Clientes.id = Pedidos.clienteId

} devuelve los nombres de los clientes junto con sus pedidos.



# 10 AÑOS REUNIENDO AMIGOS



DESCÁRGATE BIWENGER



**CONSULTAS:** instrucciones que se hacen a la base de datos para obtener o modificar información.

## TIPOS DE INSTRUCCIÓN

- SELECT → leer datos. `SELECT * FROM Clientes;`
- INSERT → insertar nuevos datos `INSERT INTO Clientes (...). VALUES (...);`
- UPDATE → modificar datos existentes `UPDATE Clientes SET edad = 30 WHERE id=1`
- DELETE → borrar datos `DELETE FROM Clientes WHERE id=1;`

## → MÉTODO PARA HACER CONSULTAS →

1- LEER ENUNCIADO → - Que información tengo que mostrar  
↳ de qué tabla/s viene

↳ hay condiciones?  
↳ agrupa, ordena, cuenta?

2- IDENTIFICAR LAS TABLAS IMPLICADAS → - Me encanta más de una tabla? (JOIN?)  
↳ donde están los datos que me piden?

3- ESTRUCTURA BÁSICA → `SELECT ... (tipo INSTRUCCIÓN)`

FROM ...

LEFT JOIN ... ON ... = ... ↳ si tiene right

WHERE ... ↳ los cumplen

GROUP BY ... ↳ todos por el LEFT

ORDER BY ... ↳ null

GROUP BY → para agrupar datos para hacer cálculos y luego aplicarles funciones como COUNT, SUM

ORDER BY → ordena los resultados por una o más columnas ASC / DESC

# PROCEDURE

como una función que se guarda en la base de datos y se puede ejecutar cuando lo quiera

## ESTRUCTURA GENERAL

DELIMITER //

CREATE PROCEDURE nombreProcedure ( IN param1 TIPO, IN param2 TIPO,  
OUT paramSalida TIPO )

BEGIN  
    **NO** siempre tiene parámetro de salida

DECLARE ← si hay que declarar variables

// manejo de errores

DECLARE EXIT HANDLER FOR SQLCEPTION

BEGIN

ROLLBACK;

SIGNAL SQLSTATE '4500' SET MESSAGE\_TEXT = 'Error'

END;

// iniciamos la transacción

START TRANSACTION;

SET ... = ... ← le damos valores a las variables

// se lanza aquí normalmente el IF con el ERROR

// se ejecuta a realizar lo pedido, lo importante a modificar

COMMIT;

END //

DELIMITER;

IF ... THEN ... ENDIF → para condicionales

WHILE ... DO ... END WHILE → para bucles

SET var = ... → para asignar valores a variables

Para llamarla:

CALL nombreProcedure ( a,b,c )

lo ponemos antes de iniciar la transacción  
siempre igual. Viene en (DB5: SQL AVANZADO)

tu el examen pedirás tu propio control de errores, solo hay que  
IF condición de error THEN

SIGNAL SQLSTATE '45001' SET MESSAGE\_TEXT = 'enviado'



# 10 AÑOS

## REUNIENDO AMIGOS

**DESCÁRGATE  
BIWENGER**



# TRIGGERS

bloque de código que se ejecuta automáticamente en la base de datos cuando ocurre un evento  
sirve para validar datos antes de insertarlos, restringir ciertas operaciones, automatizar cálculos,...

## TIPOS DE EVENTOS QUE PUEDEN ACTIVAR UN TRIGGER

- BEFORE / AFTER INSERT → antes / después de insertar un nuevo dato.
- BEFORE / AFTER UPDATE → " " de modificar un dato existente.
- BEFORE / AFTER DELETE → " " de eliminar un dato.

## SINTAXIS BÁSICA

DELIMITER //

CREATE OR REPLACE TRIGGER nombreTrigger  
[ TIPO EVENTO ] ON nombreTabla

FOR EACH ROW

BEGIN

DECLARE

SELECT ... ← para referirnos a la nueva columna que entra y comprueba usamos NEW. → uso de los valores de la tabla donde lo vamos a meter (nombreTabla)

// código a ejecutar → { IF ... THEN  
END // SIGNAL SQLSTATE '4500'  
DECLARE;  
SET MESSAGE\_TEXT = 'restricción';  
END IF



**CLAUSULAZO A TU PORTERO EL JUEVES**  
»»»  
**ESTUDIAR EN VERANO**

**PT**

**DESCÁRGATE BIWENGER**

**DL**



Made with Goodnotes

**WUOLAH**