Anexo B - Scripts SQL

Pepe Calderón, Fernando Sola, Daniel Ayala, Inma Hernández, Margarita Cruz, Carlos Arévalo, David Ruiz

Escuela Técnica Superior de
Ingeniería Informática

Índice

Este anexo contiene todos los scripts SQL necesarios para trabajar con la base de datos **GradesDB** utilizada en los laboratorios de IISSI-1.

1 Creación de la base de datos

Script principal para crear el esquema de la base de datos, incluyendo todas las tablas, claves, restricciones, funciones y disparadores (triggers).

createDB.sql

```
--
-- Autor: David Ruiz
-- Fecha: Noviembre de 2024
-- Descripción: Script para crear la BD de Grados
--

DROP DATABASE IF EXISTS GradesDB;
CREATE DATABASE GradesDB;
USE GradesDB;

--
================================================================================
-- Eliminamos tablas previas siguiendo el orden inverso de dependencias
--
================================================================================
SET FOREIGN_KEY_CHECKS = 0;
DROP TABLE IF EXISTS grades;
DROP TABLE IF EXISTS teaching_loads;
DROP TABLE IF EXISTS group_enrollments;
DROP TABLE IF EXISTS subject_enrollments;
DROP TABLE IF EXISTS groups;
DROP TABLE IF EXISTS subjects;
DROP TABLE IF EXISTS students;
```

```sql
DROP TABLE IF EXISTS professors;
DROP TABLE IF EXISTS degrees;
DROP TABLE IF EXISTS people;
SET FOREIGN_KEY_CHECKS = 1;

--
================================================================================
-- Tabla: people
--
================================================================================
CREATE TABLE people (
    person_id INT AUTO_INCREMENT,
    dni CHAR(9) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(150) NOT NULL,
    age TINYINT NOT NULL,
    email VARCHAR(255) NOT NULL,
    PRIMARY KEY (person_id)
);

--
================================================================================
-- Tabla: professors
--
================================================================================
CREATE TABLE professors (
    professor_id INT,
    category VARCHAR(30) NOT NULL,
    PRIMARY KEY (professor_id),
    FOREIGN KEY (professor_id) REFERENCES people(person_id)
);

--
================================================================================
-- Tabla: students
--
================================================================================
CREATE TABLE students (
    student_id INT,
    access_method VARCHAR(20) NOT NULL,
    PRIMARY KEY (student_id),
    FOREIGN KEY (student_id) REFERENCES people(person_id)
);

--
================================================================================
-- Tabla: degrees
--
================================================================================
CREATE TABLE degrees (
```

```
    degree_id INT AUTO_INCREMENT,
    degree_name VARCHAR(80) NOT NULL,
    duration_years TINYINT NOT NULL,
    PRIMARY KEY (degree_id)
);

--
================================================================================
-- Tabla: subjects
--
================================================================================
CREATE TABLE subjects (
    subject_id INT AUTO_INCREMENT,
    degree_id INT NOT NULL,
    subject_name VARCHAR(120) NOT NULL,
    acronym VARCHAR(12) NOT NULL,
    credits TINYINT NOT NULL,
    course TINYINT NOT NULL,
    subject_type VARCHAR(30) NOT NULL,
    PRIMARY KEY (subject_id),
    FOREIGN KEY (degree_id) REFERENCES degrees(degree_id)
);

--
================================================================================
-- Tabla: subject_enrollments
--
================================================================================
CREATE TABLE subject_enrollments (
    student_id INT,
    subject_id INT,
    PRIMARY KEY (student_id, subject_id),
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (subject_id) REFERENCES subjects(subject_id)
);

--
================================================================================
-- Tabla: groups
--
================================================================================
CREATE TABLE groups (
    group_id INT AUTO_INCREMENT,
    subject_id INT NOT NULL,
    group_name VARCHAR(40) NOT NULL,
    activity VARCHAR(15) NOT NULL,
    academic_year YEAR NOT NULL,
    PRIMARY KEY (group_id),
    FOREIGN KEY (subject_id) REFERENCES subjects(subject_id) ON DELETE
CASCADE
```

```
);

--
================================================================================
-- Tabla: group_enrollments
--
================================================================================
CREATE TABLE group_enrollments (
    student_id INT,
    group_id INT,
    PRIMARY KEY (student_id, group_id),
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (group_id) REFERENCES groups(group_id)
);

--
================================================================================
-- Tabla: teaching_loads
--
================================================================================
CREATE TABLE teaching_loads (
    professor_id INT,
    group_id INT,
    credits DECIMAL(4,1) NOT NULL,
    PRIMARY KEY (professor_id, group_id),
    FOREIGN KEY (professor_id) REFERENCES professors(professor_id),
    FOREIGN KEY (group_id) REFERENCES groups(group_id)
);

--
================================================================================
-- Tabla: grades
--
================================================================================
CREATE TABLE grades (
    grade_id INT AUTO_INCREMENT,
    student_id INT NOT NULL,
    group_id INT NOT NULL,
    grade_value DECIMAL(4,2) NOT NULL,
    exam_call VARCHAR(20) NOT NULL,
    with_honors BOOLEAN NOT NULL DEFAULT 0,
    PRIMARY KEY (grade_id),
    FOREIGN KEY (student_id) REFERENCES students(student_id),
    FOREIGN KEY (group_id) REFERENCES groups(group_id)
);

--
================================================================================
-- RESTRICCIONES (CHECK) SEGÚN EL MODELO
--
```

```
================================================================================


ALTER TABLE people
    ADD CONSTRAINT rn12_people_age CHECK (age BETWEEN 16 AND 70),
    ADD CONSTRAINT rn14_people_dni CHECK (dni REGEXP '^[0-9]{8}[A-Za-z]$');

ALTER TABLE professors
    ADD CONSTRAINT rn20_professors_category CHECK (
        category IN ('Ayudante','AyudanteDoctor','Titular','Catedrático')
    );

ALTER TABLE students
    ADD CONSTRAINT rn19_students_access_method CHECK (
        access_method IN
('Selectividad','Ciclo','Mayor','Titulado','Extranjero')
    );

ALTER TABLE degrees
    ADD CONSTRAINT rn13_degree_duration CHECK (duration_years BETWEEN 3 AND
6);

ALTER TABLE subjects
    ADD CONSTRAINT rn10_subjects_credits CHECK (credits IN (6, 12)),
    ADD CONSTRAINT rn16_subjects_course CHECK (course BETWEEN 1 AND 6),
    ADD CONSTRAINT ck_subjects_type CHECK (
        subject_type IN ('Formación Básica','Obligatoria','Optativa')
    );

ALTER TABLE groups
    ADD CONSTRAINT rn15_groups_year CHECK (academic_year BETWEEN 2000 AND
2100),
    ADD CONSTRAINT ck_groups_activity CHECK (
        activity IN ('Teoría','Laboratorio')
    );

ALTER TABLE teaching_loads
    ADD CONSTRAINT rn21_teaching_loads_credits CHECK (credits > 0);

ALTER TABLE grades
    ADD CONSTRAINT rn11_grades_value CHECK (grade_value BETWEEN 0 AND 10),
    ADD CONSTRAINT rn08_grades_with_honors CHECK (
        with_honors = 0 OR grade_value >= 9
    ),
    ADD CONSTRAINT rn18_grades_exam_call CHECK (
        exam_call IN ('Primera','Segunda','Tercera','Extraordinaria')
    );

--
================================================================================
```

```
-- RESTRICCIONES DE UNICIDAD (UNIQUE)
--
================================================================================
ALTER TABLE people
    ADD CONSTRAINT rn_uq_people_dni UNIQUE (dni),
    ADD CONSTRAINT rn_uq_people_email UNIQUE (email);

ALTER TABLE degrees
    ADD CONSTRAINT rn_uq_degrees_name UNIQUE (degree_name);

ALTER TABLE subjects
    ADD CONSTRAINT rn_uq_subjects_name UNIQUE (subject_name),
    ADD CONSTRAINT rn_uq_subjects_acronym UNIQUE (acronym);

ALTER TABLE groups
    ADD CONSTRAINT rn_uq_groups_name UNIQUE (subject_id, group_name,
academic_year);

--
================================================================================
-- RNF01 - Control de acceso: Añadir atributos role y password_hash a
people
--
================================================================================

-- Primero añadimos los atributos role y password_hash que son necesarios
-- para el control de acceso (RNF001) pero no forman parte del modelo
relacional básico
-- ALTER TABLE people
--     ADD COLUMN role ENUM('student','professor') NOT NULL AFTER email,
--     ADD COLUMN password_hash VARCHAR(255) NOT NULL AFTER role;

--
================================================================================
-- TRIGGERS PARA LAS RN COMPLEJAS
--
================================================================================

DELIMITER //

-- RN08: Un alumno no puede acceder por selectividad teniendo menos de 16
años
CREATE OR REPLACE TRIGGER t_biu_students_rn08
BEFORE INSERT OR UPDATE ON students
FOR EACH ROW
BEGIN
    DECLARE v_age TINYINT;
    IF NEW.access_method = 'Selectividad' THEN
        SELECT age INTO v_age FROM people WHERE person_id = NEW.student_id;
        IF v_age < 16 THEN
```

```
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'RN08: No se puede acceder por
Selectividad con menos de 16 años';
        END IF;
    END IF;
END//

-- RN02: Un alumno solo puede tener notas en grupos a los que pertenece
CREATE OR REPLACE TRIGGER t_biu_grades_rn02
BEFORE INSERT OR UPDATE ON grades
FOR EACH ROW
BEGIN
    DECLARE v_count INT;
    SELECT COUNT(*) INTO v_count
    FROM group_enrollments
    WHERE student_id = NEW.student_id
      AND group_id = NEW.group_id;

    IF v_count = 0 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'RN02: El alumno no pertenece al grupo
indicado';
    END IF;
END//

CREATE OR REPLACE TRIGGER t_biu_grades_rn01
BEFORE INSERT OR UPDATE ON grades
FOR EACH ROW
BEGIN
    DECLARE v_subject_id INT;
    DECLARE v_academic_year YEAR;
    DECLARE v_exists INT;

    SELECT subject_id, academic_year INTO v_subject_id, v_academic_year
    FROM groups
    WHERE group_id = NEW.group_id;

    SELECT COUNT(*) INTO v_exists
    FROM grades g
    JOIN groups gr ON gr.group_id = g.group_id
    WHERE g.student_id = NEW.student_id
      AND g.exam_call = NEW.exam_call
      AND gr.subject_id = v_subject_id
      AND gr.academic_year = v_academic_year
      AND (NEW.grade_id IS NULL OR g.grade_id <> NEW.grade_id);

    IF v_exists > 0 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'RN01: Ya existe una nota para la misma
asignatura, convocatoria y año académico';
```

```
    END IF;
END//

-- RN05: Una nota no puede modificarse en más de 4 puntos
CREATE OR REPLACE TRIGGER t_bu_grades_rn05
BEFORE UPDATE ON grades
FOR EACH ROW
BEGIN
    IF ABS(NEW.grade_value - OLD.grade_value) > 4 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'RN05: No se puede modificar la nota en más
de 4 puntos';
    END IF;
END//

-- RN03: Un grupo no puede tener más de 2 profesores
CREATE OR REPLACE TRIGGER t_bi_teaching_loads_rn03
BEFORE INSERT ON teaching_loads
FOR EACH ROW
BEGIN
    DECLARE v_count INT;

    SELECT COUNT(*) INTO v_count
    FROM teaching_loads
    WHERE group_id = NEW.group_id;

    IF v_count >= 2 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'RN03: El grupo ya tiene 2 profesores
asignados';
    END IF;
END//

CREATE OR REPLACE TRIGGER t_bu_teaching_loads_rn03
BEFORE UPDATE ON teaching_loads
FOR EACH ROW
BEGIN
    DECLARE v_count INT;

    SELECT COUNT(*) INTO v_count
    FROM teaching_loads
    WHERE group_id = NEW.group_id
      AND NOT (professor_id = OLD.professor_id AND group_id =
OLD.group_id);

    IF v_count >= 2 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'RN03: El grupo ya tiene 2 profesores
asignados';
    END IF;
```

```
END//

-- Funciones auxiliares para límites de grupos
CREATE OR REPLACE FUNCTION f_student_group_limit(
    p_student_id INT,
    p_subject_id INT,
    p_activity VARCHAR(15),
    p_excluded_group INT
)
RETURNS BOOLEAN
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE v_count INT;
    DECLARE v_result BOOLEAN DEFAULT FALSE;
    SELECT COUNT(*) INTO v_count
    FROM group_enrollments ge
    JOIN groups g ON g.group_id = ge.group_id
    WHERE ge.student_id = p_student_id
      AND g.subject_id = p_subject_id
      AND g.activity = p_activity
      AND (p_excluded_group IS NULL OR ge.group_id <> p_excluded_group);

    IF p_activity = 'Teoría' THEN
        SET v_result = (v_count >= 1);
    ELSEIF p_activity = 'Laboratorio' THEN
        SET v_result = (v_count >= 1);
    END IF;
    RETURN v_result;
END//

CREATE OR REPLACE FUNCTION f_is_student_enrolled(
    p_student_id INT,
    p_subject_id INT
)
RETURNS BOOLEAN
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE v_exists INT;
    SELECT COUNT(*) INTO v_exists
    FROM subject_enrollments
    WHERE student_id = p_student_id
      AND subject_id = p_subject_id;
    RETURN v_exists > 0;
END//

CREATE OR REPLACE FUNCTION f_count_subject_groups(
    p_subject_id INT,
    p_activity VARCHAR(15),
```

```
    p_academic_year YEAR,
    p_excluded_group INT
)
RETURNS BOOLEAN
DETERMINISTIC
READS SQL DATA
BEGIN
    DECLARE v_count INT;
    DECLARE v_result BOOLEAN DEFAULT FALSE;
    IF p_activity = 'Teoría' THEN
        SELECT COUNT(*) INTO v_count
        FROM groups
        WHERE subject_id = p_subject_id
          AND activity = 'Teoría'
          AND academic_year = p_academic_year
          AND (p_excluded_group IS NULL OR group_id <> p_excluded_group);
        SET v_result = (v_count >= 1);
    ELSEIF p_activity = 'Laboratorio' THEN
        SELECT COUNT(*) INTO v_count
        FROM groups
        WHERE subject_id = p_subject_id
          AND activity = 'Laboratorio'
          AND academic_year = p_academic_year
          AND (p_excluded_group IS NULL OR group_id <> p_excluded_group);
        SET v_result = (v_count >= 2);
    END IF;
    RETURN v_result;
END//

-- RN04: Límite de grupos de teoría y laboratorio por alumno y asignatura
CREATE OR REPLACE TRIGGER t_bi_group_enrollments_rn04
BEFORE INSERT ON group_enrollments
FOR EACH ROW
BEGIN
    DECLARE v_subject_id INT;
    DECLARE v_activity VARCHAR(15);
    SELECT subject_id, activity INTO v_subject_id, v_activity
    FROM groups
    WHERE group_id = NEW.group_id;

    IF NOT f_is_student_enrolled(NEW.student_id, v_subject_id) THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'RN07: El alumno debe estar matriculado en
la asignatura';
    END IF;

    IF f_student_group_limit(NEW.student_id, v_subject_id, v_activity,
NULL) THEN
        IF v_activity = 'Teoría' THEN
            SIGNAL SQLSTATE '45000'
```

```
                SET MESSAGE_TEXT = 'RN04: Solo puede haber un grupo de
teoría por asignatura y alumno';
        ELSE
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'RN04: Solo puede haber dos grupos de
laboratorio por asignatura y alumno';
        END IF;
    END IF;
END//

CREATE OR REPLACE TRIGGER t_bu_group_enrollments_rn04
BEFORE UPDATE ON group_enrollments
FOR EACH ROW
BEGIN
    DECLARE v_subject_id INT;
    DECLARE v_activity VARCHAR(15);
    SELECT subject_id, activity INTO v_subject_id, v_activity
    FROM groups
    WHERE group_id = NEW.group_id;

    IF NOT f_is_student_enrolled(NEW.student_id, v_subject_id) THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'RN07: El alumno debe estar matriculado en
la asignatura';
    END IF;

    IF f_student_group_limit(NEW.student_id, v_subject_id, v_activity,
OLD.group_id) THEN
        IF v_activity = 'Teoría' THEN
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'RN04: Solo puede haber un grupo de
teoría por asignatura y alumno';
        ELSE
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'RN04: Solo puede haber dos grupos de
laboratorio por asignatura y alumno';
        END IF;
    END IF;
END//

CREATE OR REPLACE TRIGGER t_bi_groups_rn06
BEFORE INSERT ON groups
FOR EACH ROW
BEGIN
    IF f_count_subject_groups(NEW.subject_id, NEW.activity,
NEW.academic_year, NULL) THEN
        IF NEW.activity = 'Teoría' THEN
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'RN06: Solo puede existir un grupo de
teoría por asignatura y año académico';
```

```
        ELSE
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'RN06: Solo pueden existir dos grupos de
laboratorio por asignatura y año académico';
        END IF;
    END IF;
END//

CREATE OR REPLACE TRIGGER t_bu_groups_rn06
BEFORE UPDATE ON groups
FOR EACH ROW
BEGIN
    IF f_count_subject_groups(NEW.subject_id, NEW.activity,
NEW.academic_year, OLD.group_id) THEN
        IF NEW.activity = 'Teoría' THEN
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'RN06: Solo puede existir un grupo de
teoría por asignatura y año académico';
        ELSE
            SIGNAL SQLSTATE '45000'
                SET MESSAGE_TEXT = 'RN06: Solo pueden existir dos grupos de
laboratorio por asignatura y año académico';
        END IF;
    END IF;
END//

DELIMITER ;
```

2

Carga inicial de datos

Script para poblar la base de datos con datos de ejemplo para las pruebas.

populateDB.sql

```
--
-- Autor: David Ruiz
-- Fecha: Noviembre 2024
-- Descripción: Procedimiento para poblar la BD de Grados
--
USE GradesDB;

DELIMITER //
CREATE OR REPLACE PROCEDURE p_populate()
BEGIN
    SET FOREIGN_KEY_CHECKS = 0;
    DELETE FROM grades;
    DELETE FROM teaching_loads;
    DELETE FROM group_enrollments;
    DELETE FROM subject_enrollments;
    DELETE FROM groups;
    DELETE FROM students;
    DELETE FROM professors;
    DELETE FROM people;
    DELETE FROM subjects;
    DELETE FROM degrees;
    SET FOREIGN_KEY_CHECKS = 1;

    INSERT INTO degrees (degree_id, degree_name, duration_years) VALUES
```

```
        (1, 'Ingeniería del Software', 4),
        (2, 'Ingeniería de Computadores', 4),
        (3, 'Tecnologías Informáticas', 4);

    INSERT INTO subjects (
        subject_id,
        degree_id,
        subject_name,
        acronym,
        credits,
        course,
        subject_type
    ) VALUES
        -- Primer curso (Tecnologías Informáticas)
        (1, 3, 'Fundamentos de Programación', 'FP', 12, 1, 'Formación
Básica'),
        (2, 3, 'Cálculo Infinitesimal y Numérico', 'CIN', 6, 1, 'Formación
Básica'),
        (3, 3, 'Circuitos Electrónicos Digitales', 'CED', 6, 1, 'Formación
Básica'),
        (4, 3, 'Fundamentos Físicos de la Informática', 'FFI', 6, 1,
'Formación Básica'),
        (5, 3, 'Introducción a la Matemática Discreta', 'IMD', 6, 1,
'Formación Básica'),
        (6, 3, 'Administración de Empresas', 'ADE', 6, 1, 'Formación
Básica'),
        (7, 3, 'Álgebra Lineal y Numérica', 'ALN', 6, 1, 'Formación
Básica'),
        (8, 3, 'Estadística', 'EST', 6, 1, 'Formación Básica'),
        (9, 3, 'Estructura de Computadores', 'EC', 6, 1, 'Formación
Básica'),
        -- Segundo curso (Tecnologías Informáticas)
        (10, 3, 'Análisis y Diseño de Datos y Algoritmos', 'ADDA', 12, 2,
'Obligatoria'),
        (11, 3, 'Introducción a la Ingeniería del Software y los Sistemas
de Información I', 'IISSI-1', 6, 2, 'Obligatoria'),
        (12, 3, 'Matemática Discreta', 'MD', 6, 2, 'Obligatoria'),
        (13, 3, 'Redes de Computadores', 'RC', 6, 2, 'Obligatoria'),
        (14, 3, 'Arquitectura de Computadores', 'AC', 6, 2, 'Obligatoria'),
        (15, 3, 'Introducción a la Ingeniería del Software y los Sistemas
de Información II', 'IISSI-2', 6, 2, 'Obligatoria'),
        (16, 3, 'Sistemas Operativos', 'SO', 6, 2, 'Obligatoria'),
        (17, 3, 'Inteligencia Artificial', 'IA', 6, 2, 'Obligatoria');

    INSERT INTO people (person_id, dni, first_name, last_name, age, email)
VALUES
        (1, '00000001A', 'David', 'Ruiz', 50, 'druiz@us.es'),
        (2, '00000002B', 'Inma', 'Hernández', 40, 'inmahernandez@us.es'),
        (3, '00000003C', 'Fernando', 'Sola', 28, 'fsola@us.es'),
        (4, '00000004D', 'Daniel', 'Ayala', 32, 'dayala1@us.es'),
```

```
        (5, '00000005E', 'Pepe', 'Calderón', 43, 'pepecalderon@us.es'),
        (6, '10000006F', 'David', 'Romero', 22, 'david.romero@alum.us.es'),
        (7, '10000007G', 'Lucía', 'Molina', 21, 'lucia.molina@alum.us.es'),
        (8, '10000008H', 'Hugo', 'Paredes', 20, 'hugo.paredes@alum.us.es'),
        (9, '10000009J', 'Sara', 'Campos', 21, 'sara.campos@alum.us.es'),
        (10, '10000010K', 'Mario', 'Galán', 22, 'mario.galan@alum.us.es'),
        (11, '10000011L', 'Elena', 'Torres', 21,
'elena.torres@alum.us.es'),
        (12, '10000012M', 'Rubén', 'Durán', 20, 'ruben.duran@alum.us.es'),
        (13, '10000013N', 'Claudia', 'Soto', 23,
'claudia.soto@alum.us.es'),
        (14, '10000014P', 'Iván', 'Cuesta', 22, 'ivan.cuesta@alum.us.es'),
        (15, '10000015Q', 'Noelia', 'Rey', 21, 'noelia.rey@alum.us.es'),
        (16, '10000016R', 'Pablo', 'Vidal', 22, 'pablo.vidal@alum.us.es'),
        (17, '10000017S', 'Alicia', 'Muñoz', 21,
'alicia.munoz@alum.us.es'),
        (18, '10000018T', 'Sergio', 'Izquierdo', 22,
'sergio.izquierdo@alum.us.es'),
        (19, '10000019U', 'Nerea', 'Saiz', 20, 'nerea.saiz@alum.us.es'),
        (20, '10000020V', 'Álvaro', 'León', 23, 'alvaro.leon@alum.us.es'),
        (21, '10000021W', 'Julia', 'Benito', 21,
'julia.benito@alum.us.es'),
        (22, '10000022X', 'Tomás', 'Rubio', 22, 'tomas.rubio@alum.us.es'),
        (23, '10000023Y', 'Irene', 'Salas', 21, 'irene.salas@alum.us.es'),
        (24, '10000024Z', 'Álex', 'Delgado', 22,
'alex.delgado@alum.us.es'),
        (25, '10000025A', 'Paula', 'Bermejo', 21,
'paula.bermejo@alum.us.es');

    -- Las credenciales se gestionan desde grants.sql

    INSERT INTO professors (professor_id, category) VALUES
        (1, 'Catedrático'),
        (2, 'Titular'),
        (3, 'AyudanteDoctor'),
        (4, 'Titular'),
        (5, 'Ayudante');

    INSERT INTO students (student_id, access_method) VALUES
        (6, 'Selectividad'),
        (7, 'Selectividad'),
        (8, 'Selectividad'),
        (9, 'Selectividad'),
        (10, 'Selectividad'),
        (11, 'Selectividad'),
        (12, 'Selectividad'),
        (13, 'Selectividad'),
        (14, 'Selectividad'),
        (15, 'Selectividad'),
        (16, 'Selectividad'),
```

```
        (17, 'Selectividad'),
        (18, 'Selectividad'),
        (19, 'Selectividad'),
        (20, 'Selectividad'),
        (21, 'Selectividad'),
        (22, 'Selectividad'),
        (23, 'Selectividad'),
        (24, 'Selectividad'),
        (25, 'Selectividad');

    INSERT INTO groups (group_id, subject_id, group_name, activity,
academic_year) VALUES
        (1, 11, 'T1', 'Teoría', 2024),
        (2, 11, 'L1', 'Laboratorio', 2024),
        (3, 11, 'L2', 'Laboratorio', 2024);

    INSERT INTO subject_enrollments (student_id, subject_id) VALUES
        (6, 11), (7, 11), (8, 11), (9, 11), (10, 11),
        (11, 11), (12, 11), (13, 11), (14, 11), (15, 11),
        (16, 11), (17, 11), (18, 11), (19, 11), (20, 11),
        (21, 11), (22, 11), (23, 11), (24, 11), (25, 11);

    INSERT INTO group_enrollments (student_id, group_id) VALUES
        -- Teoría
        (6, 1), (7, 1), (8, 1), (9, 1), (10, 1),
        (11, 1), (12, 1), (13, 1), (14, 1), (15, 1),
        (16, 1), (17, 1), (18, 1), (19, 1), (20, 1),
        (21, 1), (22, 1), (23, 1), (24, 1), (25, 1),
        -- Laboratorio L1
        (6, 2), (7, 2), (8, 2), (9, 2), (10, 2),
        (11, 2), (12, 2), (13, 2), (14, 2), (15, 2),
        -- Laboratorio L2
        (16, 3), (17, 3), (18, 3), (19, 3), (20, 3),
        (21, 3), (22, 3), (23, 3), (24, 3), (25, 3);

    INSERT INTO teaching_loads (professor_id, group_id, credits) VALUES
        (1, 1, 3.0),
        (2, 1, 3.0),
        (3, 2, 3.0),
        (5, 3, 1.5),
        (4, 3, 1.5);

    INSERT INTO grades (grade_id, student_id, group_id, grade_value,
exam_call, with_honors) VALUES
        -- Primera convocatoria: 10 aprobados (2 con MH), 10 suspensos
        (1, 6, 1, 9.8, 'Primera', 1),
        (2, 7, 1, 9.2, 'Primera', 1),
        (3, 8, 1, 5.4, 'Primera', 0),
        (4, 9, 1, 8.0, 'Primera', 0),
        (5, 10, 1, 6.0, 'Primera', 0),
```

```
        (6, 11, 1, 5.7, 'Primera', 0),
        (7, 12, 1, 7.0, 'Primera', 0),
        (8, 13, 1, 6.3, 'Primera', 0),
        (9, 14, 1, 5.8, 'Primera', 0),
        (10, 15, 1, 6.1, 'Primera', 0),
        (11, 16, 1, 4.2, 'Primera', 0),
        (12, 17, 1, 3.9, 'Primera', 0),
        (13, 18, 1, 4.5, 'Primera', 0),
        (14, 19, 1, 4.8, 'Primera', 0),
        (15, 20, 1, 4.4, 'Primera', 0),
        (16, 21, 1, 3.5, 'Primera', 0),
        (17, 22, 1, 3.8, 'Primera', 0),
        (18, 23, 1, 4.0, 'Primera', 0),
        (19, 24, 1, 4.3, 'Primera', 0),
        (20, 25, 1, 4.7, 'Primera', 0),
        -- Segunda convocatoria
        (21, 16, 1, 6.2, 'Segunda', 0),
        (22, 17, 1, 5.8, 'Segunda', 0),
        (23, 18, 1, 6.4, 'Segunda', 0),
        (24, 19, 1, 5.9, 'Segunda', 0),
        (25, 20, 1, 6.1, 'Segunda', 0),
        (26, 21, 1, 4.6, 'Segunda', 0),
        (27, 22, 1, 4.9, 'Segunda', 0),
        (28, 23, 1, 4.7, 'Segunda', 0),
        (29, 24, 1, 4.4, 'Segunda', 0),
        (30, 25, 1, 4.3, 'Segunda', 0),
        -- Tercera convocatoria
        (31, 21, 1, 5.6, 'Tercera', 0),
        (32, 22, 1, 5.9, 'Tercera', 0),
        (33, 23, 1, 5.5, 'Tercera', 0),
        (34, 24, 1, 6.0, 'Tercera', 0),
        (35, 25, 1, 5.7, 'Tercera', 0);
END //
DELIMITER ;

CALL p_populate();
```

3 Carga adicional de datos

Script con datos adicionales para ampliación de las pruebas.

populateDB2.sql

```
--
-- Autor: David Ruiz
-- Fecha: Enero 2026
-- Descripción: Datos adicionales para GradesDB para cubrir consultas de L3
--              Este script añade datos sin borrar los existentes de
populateDB.sql
--
USE GradesDB;

DELIMITER //
CREATE OR REPLACE PROCEDURE p_populate_grados_extra()
BEGIN
    -- Eliminar datos adicionales si ya existen (hacer el procedimiento
idempotente)
    DELETE FROM grades WHERE grade_id >= 36;
    DELETE FROM teaching_loads WHERE group_id >= 4;
    DELETE FROM group_enrollments WHERE group_id >= 4;
    DELETE FROM subject_enrollments WHERE subject_id >= 18 OR (student_id
>= 6 AND subject_id IN (1, 2, 10));
    DELETE FROM groups WHERE group_id >= 4;
    DELETE FROM subjects WHERE subject_id >= 18;

    -- Añadir más asignaturas a otros grados para consultas de agregación
    INSERT INTO subjects (
```

```
        subject_id,
        degree_id,
        subject_name,
        acronym,
        credits,
        course,
        subject_type
    ) VALUES
        -- Ingeniería del Software (degree_id = 1)
        (18, 1, 'Fundamentos de Programación Orientada a Objetos', 'FPOO',
12, 1, 'Formación Básica'),
        (19, 1, 'Bases de Datos', 'BD', 6, 2, 'Obligatoria'),
        (20, 1, 'Ingeniería del Software', 'IS', 6, 2, 'Obligatoria'),
        (21, 1, 'Sistemas de Información', 'SI', 6, 3, 'Obligatoria'),
        (22, 1, 'Gestión de Proyectos', 'GP', 6, 3, 'Obligatoria'),
        (23, 1, 'Calidad del Software', 'CS', 6, 3, 'Obligatoria'),
        -- Ingeniería de Computadores (degree_id = 2)
        (24, 2, 'Fundamentos de Computadores', 'FC', 12, 1, 'Formación
Básica'),
        (25, 2, 'Sistemas Digitales', 'SD', 6, 2, 'Obligatoria'),
        (26, 2, 'Arquitectura de Computadores Avanzada', 'ACA', 6, 3,
'Obligatoria'),
        -- Más asignaturas para TI (degree_id = 3)
        (27, 3, 'Fundamentos de Ingeniería del Software', 'FIS', 6, 3,
'Obligatoria'),
        (28, 3, 'Desarrollo de Sistemas de Información', 'DSI', 6, 3,
'Obligatoria'),
        (29, 3, 'Seguridad Informática', 'SEG', 6, 4, 'Optativa');

    -- Añadir más grupos para consultas de DISTINCT, COUNT, etc.
    -- IMPORTANTE: Respetando RN006 (máx. 1 grupo teoría y 2 laboratorio
por asignatura)
    INSERT INTO groups (group_id, subject_id, group_name, activity,
academic_year) VALUES
        -- Grupos para FP (subject_id = 1) - 1 teoría + 2 laboratorio
        (4, 1, 'T1', 'Teoría', 2024),
        (5, 1, 'L1', 'Laboratorio', 2024),
        (6, 1, 'L2', 'Laboratorio', 2024),
        -- Grupos para ADDA (subject_id = 10) - 1 teoría + 2 laboratorio
        (7, 10, 'T1', 'Teoría', 2024),
        (8, 10, 'L1', 'Laboratorio', 2024),
        (9, 10, 'L2', 'Laboratorio', 2024),
        -- Grupos para CIN (subject_id = 2) - 1 teoría + 2 laboratorio
        (10, 2, 'T1', 'Teoría', 2024),
        (11, 2, 'L1', 'Laboratorio', 2024),
        (12, 2, 'L2', 'Laboratorio', 2024),
        -- Grupos para MD (subject_id = 12) - 1 teoría + 2 laboratorio
        (13, 12, 'T1', 'Teoría', 2024),
        (14, 12, 'L1', 'Laboratorio', 2024),
        (15, 12, 'L2', 'Laboratorio', 2024),
```

```
        -- Grupos de años anteriores para consultas de rango
        (16, 11, 'T1', 'Teoría', 2023),
        (17, 11, 'L1', 'Laboratorio', 2023),
        (18, 1, 'T1', 'Teoría', 2023),
        (19, 10, 'T1', 'Teoría', 2023),
        (20, 11, 'T1', 'Teoría', 2022),
        (21, 1, 'T1', 'Teoría', 2022),
        (22, 11, 'T1', 'Teoría', 2021),
        (23, 1, 'T1', 'Teoría', 2020);

    -- Matricular estudiantes en asignaturas adicionales (ANTES de
añadirlos a grupos)
    -- RN07: Un alumno sólo puede pertenecer a grupos de asignaturas en las
que está matriculado
    INSERT INTO subject_enrollments (student_id, subject_id) VALUES
        (6, 1), (7, 1), (8, 1), (9, 1), (10, 1),
        (11, 1), (12, 1), (13, 1), (14, 1), (15, 1),
        (16, 1), (17, 1), (18, 1), (19, 1), (20, 1),
        (6, 10), (7, 10), (8, 10), (9, 10), (10, 10),
        (11, 10), (12, 10), (13, 10), (14, 10), (15, 10),
        (6, 2), (7, 2), (8, 2), (9, 2), (10, 2);

    -- Más matrículas en grupos para LEFT JOIN y análisis
    INSERT INTO group_enrollments (student_id, group_id) VALUES
        -- Estudiantes en grupos de FP (subject_id = 1)
        (6, 4), (7, 4), (8, 4), (9, 4), (10, 4),
        (11, 4), (12, 4), (13, 4), (14, 4), (15, 4),
        (16, 4), (17, 4), (18, 4), (19, 4), (20, 4),
        (6, 5), (7, 5), (8, 5), (9, 5), (10, 5),
        (16, 6), (17, 6), (18, 6), (19, 6), (20, 6),
        -- Estudiantes en grupos de ADDA (subject_id = 10)
        (6, 7), (7, 7), (8, 7), (9, 7), (10, 7),
        (11, 7), (12, 7), (13, 7), (14, 7), (15, 7),
        (6, 8), (7, 8), (8, 8), (9, 8), (10, 8),
        -- Estudiantes en grupos de CIN (subject_id = 2)
        (6, 10), (7, 10), (8, 10), (9, 10), (10, 10),
        (6, 11), (7, 11), (8, 11);

    -- Más cargas docentes (RN003: máximo 2 profesores por grupo)
    INSERT INTO teaching_loads (professor_id, group_id, credits) VALUES
        (1, 4, 3.0),
        (2, 5, 1.5),
        (3, 6, 1.5),
        (4, 7, 6.0),
        (1, 8, 1.5),
        (2, 9, 1.5),
        (3, 10, 3.0),
        (4, 11, 1.5);

    -- Más notas para consultas diversas
```

```
    INSERT INTO grades (grade_id, student_id, group_id, grade_value,
exam_call, with_honors) VALUES
        -- Notas en grupo 4 (FP Teoría 2024)
        (36, 6, 4, 7.5, 'Primera', 0),
        (37, 7, 4, 8.2, 'Primera', 0),
        (38, 8, 4, 6.5, 'Primera', 0),
        (39, 9, 4, 9.5, 'Primera', 1),
        (40, 10, 4, 9.8, 'Primera', 1),
        (41, 11, 4, 7.0, 'Primera', 0),
        (42, 12, 4, 8.5, 'Primera', 0),
        (43, 13, 4, 9.0, 'Primera', 0),
        (44, 14, 4, 9.2, 'Primera', 1),
        (45, 15, 4, 8.8, 'Primera', 0),
        (46, 16, 4, 7.8, 'Primera', 0),
        (47, 17, 4, 8.0, 'Primera', 0),
        (48, 18, 4, 9.1, 'Primera', 1),
        (49, 19, 4, 8.7, 'Primera', 0),
        (50, 20, 4, 9.3, 'Primera', 1),
        -- Notas en grupo 7 (ADDA Teoría 2024)
        (51, 6, 7, 6.0, 'Primera', 0),
        (52, 7, 7, 7.5, 'Primera', 0),
        (53, 8, 7, 5.5, 'Primera', 0),
        (54, 9, 7, 8.5, 'Primera', 0),
        (55, 10, 7, 7.8, 'Primera', 0),
        (56, 11, 7, 6.2, 'Primera', 0),
        (57, 12, 7, 7.0, 'Primera', 0),
        (58, 13, 7, 8.0, 'Primera', 0),
        (59, 14, 7, 7.2, 'Primera', 0),
        (60, 15, 7, 6.8, 'Primera', 0),
        -- Notas en grupo 10 (CIN Teoría 2024)
        (61, 6, 10, 5.5, 'Primera', 0),
        (62, 7, 10, 6.0, 'Primera', 0),
        (63, 8, 10, 7.5, 'Primera', 0),
        (64, 9, 10, 8.0, 'Primera', 0),
        (65, 10, 10, 6.5, 'Primera', 0);

END //
DELIMITER ;

-- Ejecutar el procedimiento
CALL p_populate_grados_extra();
```

4 Consultas de ejemplo

Ejemplos de consultas SQL sobre la base de datos GradesDB.

queries.sql

```
--
-- Autor: David Ruiz
-- Fecha: Enero de 2026
-- Descripción: Consultas SQL de ejemplo para GradesDB
--              Cubre conceptos de consultas simples y avanzadas
--

USE GradesDB;

--
================================================================================
-- CONSULTAS SIMPLES (SELECT básico)
--
================================================================================

-- Consulta 1: Todas las asignaturas
SELECT *
FROM subjects;

-- Consulta 2: Asignatura con acrónimo 'IISSI-1'
SELECT *
FROM subjects s
WHERE s.acronym = 'IISSI-1';
```

```sql
-- Consulta 3: Nombres y acrónimos de todas las asignaturas
SELECT s.subject_name, s.acronym
FROM subjects s;

-- Consulta 4: Asignaturas de formación básica
SELECT s.subject_name, s.acronym, s.subject_type
FROM subjects s
WHERE s.subject_type = 'Formación Básica';

-- Consulta 5: Estudiantes con método de acceso por Selectividad
SELECT p.first_name, p.last_name, st.access_method
FROM students st
JOIN people p ON st.student_id = p.person_id
WHERE st.access_method = 'Selectividad';

-- Consulta 6: Profesores con categoría Catedrático
SELECT p.first_name, p.last_name, pr.category
FROM professors pr
JOIN people p ON pr.professor_id = p.person_id
WHERE pr.category = 'Catedrático';

--
================================================================================
-- VISTAS BASE: Profesores y Estudiantes con información de persona
--
================================================================================

-- Vista: Profesores con información completa
CREATE OR REPLACE VIEW v_professors AS
SELECT
    pr.professor_id,
    pr.category,
    p.person_id,
    p.dni,
    p.first_name,
    p.last_name,
    p.age,
    p.email
FROM professors pr
JOIN people p ON pr.professor_id = p.person_id;

-- Vista: Estudiantes con información completa
CREATE OR REPLACE VIEW v_students AS
SELECT
    st.student_id,
    st.access_method,
    p.person_id,
    p.dni,
    p.first_name,
    p.last_name,
```

```
    p.age,
    p.email
FROM students st
JOIN people p ON st.student_id = p.person_id;

--
================================================================================
-- CONSULTAS CON FUNCIONES AGREGADAS
--
================================================================================

-- Consulta 7: Media de las notas del grupo con ID 1
SELECT AVG(g.grade_value) AS average_grade
FROM grades g
WHERE g.group_id = 1;

-- Consulta 8: Total de créditos de las asignaturas del grado con ID 1
SELECT SUM(s.credits) AS total_credits
FROM subjects s
WHERE s.degree_id = 1;

-- Consulta 9: Número total de estudiantes
SELECT COUNT(*) AS total_students
FROM students;

-- Consulta 10: Número de grupos por tipo de actividad
SELECT activity, COUNT(*) AS total_groups
FROM groups
GROUP BY activity;

-- Consulta 11: Máxima nota obtenida
SELECT MAX(g.grade_value) AS max_grade
FROM grades g;

-- Consulta 12: Mínima nota obtenida
SELECT MIN(g.grade_value) AS min_grade
FROM grades g;

--
================================================================================
-- CONSULTAS CON CONDICIONES MÚLTIPLES
--
================================================================================

-- Consulta 13: Notas con valor menor que 5 o mayor que 9
SELECT g.grade_id, g.grade_value, g.exam_call
FROM grades g
WHERE g.grade_value < 5 OR g.grade_value > 9;

-- Consulta 14: Notas entre 5 y 7 (inclusive)
```

```sql
SELECT g.grade_id, g.grade_value, g.exam_call
FROM grades g
WHERE g.grade_value BETWEEN 5 AND 7;

-- Consulta 15: Asignaturas de 6 o 12 créditos
SELECT s.subject_name, s.credits
FROM subjects s
WHERE s.credits IN (6, 12);

-- Consulta 16: Personas con edad entre 20 y 30 años
SELECT p.first_name, p.last_name, p.age
FROM people p
WHERE p.age BETWEEN 20 AND 30;

--
================================================================================
-- CONSULTAS CON DISTINCT
--
================================================================================

-- Consulta 17: Nombres de grupos diferentes
SELECT DISTINCT g.group_name
FROM groups g;

-- Consulta 18: Tipos de asignatura diferentes
SELECT DISTINCT s.subject_type
FROM subjects s;

-- Consulta 19: Años académicos registrados
SELECT DISTINCT g.academic_year
FROM groups g
ORDER BY g.academic_year DESC;

--
================================================================================
-- CONSULTAS CON SUBCONSULTAS
--
================================================================================

-- Consulta 20: Máxima nota del estudiante con ID 1
SELECT MAX(g.grade_value) AS max_grade
FROM grades g
WHERE g.student_id = 1;

-- Consulta 21: IDs de estudiantes del año académico 2024
SELECT DISTINCT ge.student_id
FROM group_enrollments ge
WHERE ge.group_id IN (
    SELECT g.group_id
    FROM groups g
```

```
    WHERE g.academic_year = 2024
);

-- Consulta 22: Asignaturas con más créditos que la media
SELECT s.subject_name, s.credits
FROM subjects s
WHERE s.credits > (SELECT AVG(credits) FROM subjects);

--
================================================================================
-- CONSULTAS CON LIKE Y PATRONES
--
================================================================================

-- Consulta 23: Personas con DNI terminado en 'A'
SELECT p.first_name, p.last_name, p.dni
FROM people p
WHERE p.dni LIKE '%A';

-- Consulta 24: Asignaturas cuyo nombre empieza por 'Fundamentos'
SELECT s.subject_name
FROM subjects s
WHERE s.subject_name LIKE 'Fundamentos%';

-- Consulta 25: Personas con nombres de 5 letras (usar 5 guiones bajos)
SELECT p.first_name, p.last_name
FROM people p
WHERE p.first_name LIKE '_____';

--
================================================================================
-- CONSULTAS CON FUNCIONES DE FECHA
--
================================================================================

-- Consulta 26: Grupos del año 2024
SELECT g.group_name, g.activity, g.academic_year
FROM groups g
WHERE g.academic_year = 2024;

-- Consulta 27: Grupos entre 2020 y 2023
SELECT g.group_name, g.activity, g.academic_year
FROM groups g
WHERE g.academic_year BETWEEN 2020 AND 2023
ORDER BY g.academic_year;

--
================================================================================
-- VISTAS
--
```

```
================================================================================

-- Vista 1: Notas con matrícula de honor
CREATE OR REPLACE VIEW v_grades_with_honors AS
SELECT g.grade_id, g.student_id, g.group_id, g.grade_value, g.exam_call
FROM grades g
WHERE g.with_honors = 1;

-- Vista 2: Notas de todos los estudiantes con información completa
CREATE OR REPLACE VIEW v_student_grades AS
SELECT
    p.person_id,
    p.first_name,
    p.last_name,
    p.dni,
    g.grade_value,
    g.exam_call,
    g.with_honors,
    gr.group_name,
    gr.activity,
    gr.academic_year,
    s.subject_name,
    s.acronym,
    s.credits
FROM people p
JOIN students st ON p.person_id = st.student_id
JOIN grades g ON st.student_id = g.student_id
JOIN groups gr ON g.group_id = gr.group_id
JOIN subjects s ON gr.subject_id = s.subject_id;

-- Vista 3: Información de profesores con sus cargas docentes
CREATE OR REPLACE VIEW v_professor_loads AS
SELECT
    p.person_id,
    p.first_name,
    p.last_name,
    pr.category,
    tl.credits AS teaching_credits,
    gr.group_name,
    gr.activity,
    gr.academic_year,
    s.subject_name,
    s.acronym
FROM people p
JOIN professors pr ON p.person_id = pr.professor_id
JOIN teaching_loads tl ON pr.professor_id = tl.professor_id
JOIN groups gr ON tl.group_id = gr.group_id
JOIN subjects s ON gr.subject_id = s.subject_id;

-- Vista 4: Estudiantes por grado
```

```sql
CREATE OR REPLACE VIEW v_degree_students AS
SELECT DISTINCT
    d.degree_id,
    d.degree_name,
    p.person_id,
    p.first_name,
    p.last_name,
    st.access_method,
    gr.academic_year
FROM degrees d
JOIN subjects s ON d.degree_id = s.degree_id
JOIN groups gr ON s.subject_id = gr.subject_id
JOIN group_enrollments ge ON gr.group_id = ge.group_id
JOIN students st ON ge.student_id = st.student_id
JOIN people p ON st.student_id = p.person_id;

-- Uso de vistas: Consultas sobre las vistas creadas
-- Consulta 28: Total de matrículas de honor
SELECT COUNT(*) AS total_honors
FROM v_grades_with_honors;

-- Consulta 29: Nota media de cada estudiante
SELECT v.first_name, v.last_name, AVG(v.grade_value) AS average_grade
FROM v_student_grades v
GROUP BY v.person_id, v.first_name, v.last_name;

--
================================================================================
-- CONSULTAS AVANZADAS: ORDER BY, LIMIT, OFFSET
--
================================================================================

-- Consulta 30: Notas ordenadas por valor (de menor a mayor)
SELECT g.grade_id, g.student_id, g.grade_value, g.exam_call
FROM grades g
ORDER BY g.grade_value ASC;

-- Consulta 31: Notas ordenadas por valor descendente
SELECT g.grade_id, g.student_id, g.grade_value, g.exam_call
FROM grades g
ORDER BY g.grade_value DESC;

-- Consulta 32: Las 5 mejores notas
SELECT g.grade_id, g.student_id, g.grade_value, g.exam_call
FROM grades g
ORDER BY g.grade_value DESC
LIMIT 5;

-- Consulta 33: Notas aprobadas ordenadas por estudiante
SELECT
```

```
    st.first_name,
    st.last_name,
    g.grade_value,
    g.exam_call
FROM grades g
JOIN v_students st ON g.student_id = st.student_id
WHERE g.grade_value >= 5
ORDER BY st.last_name ASC, st.first_name ASC;

-- Consulta 34: Paginación - Segunda página de 5 notas
SELECT g.grade_id, g.student_id, g.grade_value
FROM grades g
ORDER BY g.grade_value DESC
LIMIT 5 OFFSET 5;

-- Consulta 35: Tercera página de 5 notas
SELECT g.grade_id, g.student_id, g.grade_value
FROM grades g
ORDER BY g.grade_value DESC
LIMIT 5 OFFSET 10;

-- Consulta 36: Grupos ordenados por año académico descendente
SELECT g.group_name, g.activity, g.academic_year
FROM groups g
ORDER BY g.academic_year DESC, g.group_name ASC;

--
================================================================================
-- CONSULTAS AVANZADAS: JOIN
--
================================================================================

-- Consulta 37: Estudiantes con sus grupos (INNER JOIN)
SELECT
    st.first_name,
    st.last_name,
    gr.group_name,
    gr.activity,
    gr.academic_year
FROM v_students st
JOIN group_enrollments ge ON st.student_id = ge.student_id
JOIN groups gr ON ge.group_id = gr.group_id;

-- Consulta 38: Asignaturas con información del grado
SELECT
    d.degree_name,
    s.subject_name,
    s.acronym,
    s.credits,
    s.course,
```

```
    s.subject_type
FROM degrees d
JOIN subjects s ON d.degree_id = s.degree_id
ORDER BY d.degree_name, s.course;

-- Consulta 39: Notas con nombre del estudiante y asignatura
SELECT
    st.first_name,
    st.last_name,
    s.subject_name,
    s.acronym,
    g.grade_value,
    g.exam_call,
    gr.academic_year
FROM grades g
JOIN v_students st ON g.student_id = st.student_id
JOIN groups gr ON g.group_id = gr.group_id
JOIN subjects s ON gr.subject_id = s.subject_id;

-- Consulta 40: Profesores con grupos que imparten
SELECT
    pr.first_name,
    pr.last_name,
    pr.category,
    s.subject_name,
    gr.group_name,
    gr.activity,
    tl.credits AS teaching_credits
FROM v_professors pr
JOIN teaching_loads tl ON pr.professor_id = tl.professor_id
JOIN groups gr ON tl.group_id = gr.group_id
JOIN subjects s ON gr.subject_id = s.subject_id;

-- Consulta 41: Estudiantes matriculados en asignaturas (a través de
grupos)
SELECT DISTINCT
    st.first_name,
    st.last_name,
    s.subject_name,
    s.acronym,
    gr.academic_year
FROM v_students st
JOIN group_enrollments ge ON st.student_id = ge.student_id
JOIN groups gr ON ge.group_id = gr.group_id
JOIN subjects s ON gr.subject_id = s.subject_id
ORDER BY st.last_name, s.subject_name;

-- Consulta 42: LEFT JOIN - Todos los estudiantes, con o sin notas
SELECT
    st.first_name,
```

```sql
    st.last_name,
    g.grade_value,
    g.exam_call
FROM v_students st
LEFT JOIN grades g ON st.student_id = g.student_id;

--
================================================================================
-- CONSULTAS AVANZADAS: GROUP BY
--
================================================================================

-- Consulta 43: Nota media de cada estudiante
SELECT
    st.first_name,
    st.last_name,
    AVG(g.grade_value) AS average_grade,
    COUNT(*) AS total_grades
FROM v_students st
JOIN grades g ON st.student_id = g.student_id
GROUP BY st.student_id, st.first_name, st.last_name;

-- Consulta 44: Número de estudiantes por método de acceso
SELECT
    st.access_method,
    COUNT(*) AS total_students
FROM students st
GROUP BY st.access_method;

-- Consulta 45: Número de asignaturas por grado
SELECT
    d.degree_name,
    COUNT(*) AS total_subjects
FROM degrees d
JOIN subjects s ON d.degree_id = s.degree_id
GROUP BY d.degree_id, d.degree_name;

-- Consulta 46: Número de grupos por asignatura
SELECT
    s.subject_name,
    s.acronym,
    COUNT(*) AS total_groups
FROM subjects s
JOIN groups gr ON s.subject_id = gr.subject_id
GROUP BY s.subject_id, s.subject_name, s.acronym;

-- Consulta 47: Nota media por convocatoria
SELECT
    g.exam_call,
    AVG(g.grade_value) AS average_grade,
```

```
    COUNT(*) AS total_grades
FROM grades g
GROUP BY g.exam_call;

-- Consulta 48: Nota media por asignatura en cada convocatoria
SELECT
    s.subject_name,
    g.exam_call,
    AVG(g.grade_value) AS average_grade,
    COUNT(*) AS total_grades
FROM grades g
JOIN groups gr ON g.group_id = gr.group_id
JOIN subjects s ON gr.subject_id = s.subject_id
GROUP BY s.subject_id, s.subject_name, g.exam_call
ORDER BY s.subject_name, g.exam_call;

-- Consulta 49: Número de estudiantes por año académico
SELECT
    gr.academic_year,
    COUNT(DISTINCT ge.student_id) AS total_students
FROM groups gr
JOIN group_enrollments ge ON gr.group_id = ge.group_id
GROUP BY gr.academic_year
ORDER BY gr.academic_year DESC;

-- Consulta 50: Carga docente total por profesor
SELECT
    pr.first_name,
    pr.last_name,
    pr.category,
    SUM(tl.credits) AS total_credits
FROM v_professors pr
JOIN teaching_loads tl ON pr.professor_id = tl.professor_id
GROUP BY pr.professor_id, pr.first_name, pr.last_name, pr.category;

--
================================================================================
-- CONSULTAS AVANZADAS: HAVING
--
================================================================================

-- Consulta 51: Estudiantes con nota media superior a 7
SELECT
    st.first_name,
    st.last_name,
    AVG(g.grade_value) AS average_grade
FROM v_students st
JOIN grades g ON st.student_id = g.student_id
GROUP BY st.student_id, st.first_name, st.last_name
HAVING AVG(g.grade_value) > 7;
```

```sql
-- Consulta 52: Asignaturas con más de 5 grupos
SELECT
    s.subject_name,
    COUNT(*) AS total_groups
FROM subjects s
JOIN groups gr ON s.subject_id = gr.subject_id
GROUP BY s.subject_id, s.subject_name
HAVING COUNT(*) > 5;

-- Consulta 53: Grupos con más de 10 estudiantes
SELECT
    gr.group_name,
    gr.activity,
    COUNT(*) AS total_students
FROM groups gr
JOIN group_enrollments ge ON gr.group_id = ge.group_id
GROUP BY gr.group_id, gr.group_name, gr.activity
HAVING COUNT(*) > 10;

-- Consulta 54: Asignaturas con nota media superior a 6
SELECT
    s.subject_name,
    AVG(g.grade_value) AS average_grade,
    COUNT(*) AS total_grades
FROM subjects s
JOIN groups gr ON s.subject_id = gr.subject_id
JOIN grades g ON gr.group_id = g.group_id
GROUP BY s.subject_id, s.subject_name
HAVING AVG(g.grade_value) > 6;

--
================================================================================
-- CONSULTAS COMPLEJAS COMBINADAS
--
================================================================================

-- Consulta 55: Top 3 asignaturas con más grupos de teoría en 2024
SELECT
    s.subject_name,
    COUNT(*) AS total_theory_groups
FROM subjects s
JOIN groups gr ON s.subject_id = gr.subject_id
WHERE gr.activity = 'Teoría' AND gr.academic_year = 2024
GROUP BY s.subject_id, s.subject_name
ORDER BY COUNT(*) DESC
LIMIT 3;

-- Consulta 56: Estudiantes con más de 3 matrículas de honor
SELECT
```

```
    st.first_name,
    st.last_name,
    COUNT(*) AS total_honors
FROM v_students st
JOIN grades g ON st.student_id = g.student_id
WHERE g.with_honors = 1
GROUP BY st.student_id, st.first_name, st.last_name
HAVING COUNT(*) > 3;

-- Consulta 57: Grados con más estudiantes en 2024
SELECT
    d.degree_name,
    COUNT(DISTINCT ge.student_id) AS total_students
FROM degrees d
JOIN subjects s ON d.degree_id = s.degree_id
JOIN groups gr ON s.subject_id = gr.subject_id
JOIN group_enrollments ge ON gr.group_id = ge.group_id
WHERE gr.academic_year = 2024
GROUP BY d.degree_id, d.degree_name
ORDER BY COUNT(DISTINCT ge.student_id) DESC;

-- Consulta 58: Nota máxima de cada estudiante con nombre y apellidos
SELECT
    st.first_name,
    st.last_name,
    MAX(g.grade_value) AS max_grade
FROM v_students st
JOIN grades g ON st.student_id = g.student_id
GROUP BY st.student_id, st.first_name, st.last_name
ORDER BY MAX(g.grade_value) DESC;

-- Consulta 59: Profesores que imparten en más de 2 grupos
SELECT
    pr.first_name,
    pr.last_name,
    COUNT(DISTINCT tl.group_id) AS total_groups
FROM v_professors pr
JOIN teaching_loads tl ON pr.professor_id = tl.professor_id
GROUP BY pr.professor_id, pr.first_name, pr.last_name
HAVING COUNT(DISTINCT tl.group_id) > 2;

-- Consulta 60: Asignaturas que pertenecen a grados con más de 10
asignaturas
SELECT DISTINCT s.subject_name, d.degree_name
FROM subjects s
JOIN degrees d ON s.degree_id = d.degree_id
WHERE d.degree_id IN (
    SELECT degree_id
    FROM subjects
    GROUP BY degree_id
```

```
    HAVING COUNT(*) > 10
);

--
================================================================================
-- FIN DEL SCRIPT
--
================================================================================
```

5 Permisos y usuarios

Script para configurar los permisos y crear los usuarios necesarios para trabajar con la base de datos.

grants.sql

```
--
-- Usuarios, roles y permisos para la BD de Grados
--
USE GradesDB;

CREATE TABLE IF NOT EXISTS credentials (
    credential_id INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(255) NOT NULL UNIQUE,
    role ENUM('admin','teacher','student') NOT NULL DEFAULT 'student',
    password_hash VARCHAR(256) NOT NULL DEFAULT '',
    CONSTRAINT ak_email UNIQUE(email)
);

-- INSERT INTO credentials (email, role)
-- VALUES
--     ('druiz@us.es', 'admin'),
--     ('inmahernandez@us.es', 'teacher'),
--     ('david.romero@alum.us.es', 'student');

-- -- Comentado para Silence
-- -- ROLES
```

```
-- -- Admin: todos los privilegios sobre la BD
-- CREATE ROLE IF NOT EXISTS role_admin;

-- -- Teacher: lectura de toda la BD y escritura limitada en 'grades'
-- CREATE ROLE IF NOT EXISTS role_teacher;

-- -- Student: lectura de toda la BD
-- CREATE ROLE IF NOT EXISTS role_student;

-- -- 3) PRIVILEGIOS A LOS ROLES (en este orden)
-- -- Admin (total sobre la BD)
-- GRANT ALL PRIVILEGES ON GradesDB.* TO role_admin;

-- -- Teacher:
-- --   - Lectura de toda la BD (para que pueda navegar y hacer SELECT en
todas las tablas)
-- --   - Modificaciones SOLO en 'grades'
-- GRANT SELECT ON GradesDB.* TO role_teacher;
-- GRANT INSERT, UPDATE ON GradesDB.grades TO role_teacher;

-- -- Student: solo lectura en toda la BD
-- GRANT SELECT ON GradesDB.* TO role_student;

-- -- USUARIOS y asignación de roles

-- CREATE USER IF NOT EXISTS 'admin_grados'@'%' IDENTIFIED BY 'druiz';
-- GRANT role_admin TO 'admin_grados'@'%';

-- CREATE USER IF NOT EXISTS 'teacher_grados'@'%' IDENTIFIED BY
'inmahernandez';
-- GRANT role_teacher TO 'teacher_grados'@'%';

-- CREATE USER IF NOT EXISTS 'student_grados'@'%' IDENTIFIED BY
'david.romero';
-- GRANT role_student TO 'student_grados'@'%';

-- -- Activar ROL por defecto (clave para que funcione al iniciar sesión)
-- -- Al iniciar sesión desde HeidiSQL odesde la línea de comando, debe
activarse el rol
-- -- por defecto para que el usuario tenga los permisos asignados al rol.

-- -- Ejecuta los siguientes escenarios de test
-- -- SET ROLE role_admin TO 'admin_grados'@'%';
-- -- SOURCE test_admin.sql;

-- -- SET ROLE role_teacher TO 'teacher_grados'@'%';
-- -- SOURCE test_teacher.sql;

-- -- SET ROLE role_student TO 'student_grados'@'%';
```

```
-- -- SOURCE test_student.sql;
```

6 Tests

Script con procedimientos de prueba para validar el funcionamiento de la base de datos.

tests.sql

```
--
-- Autor: David Ruiz
-- Fecha: Noviembre 2024
-- Descripción: Tests negativos para la BD de Grados
--
USE GradesDB;

-- ============================================================
-- TABLA DE RESULTADOS
-- ============================================================
CREATE OR REPLACE TABLE test_results (
    test_id VARCHAR(20) PRIMARY KEY,
    test_name VARCHAR(200) NOT NULL,
    test_message VARCHAR(500) NOT NULL,
    test_status ENUM('PASS','FAIL','ERROR') NOT NULL,
    execution_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- ============================================================
-- PROCEDIMIENTO AUXILIAR
-- ============================================================
DELIMITER //
CREATE OR REPLACE PROCEDURE p_log_test(
    IN p_test_id VARCHAR(20),
```

```
    IN p_message VARCHAR(500),
    IN p_status ENUM('PASS','FAIL','ERROR')
)
BEGIN
    INSERT INTO test_results(test_id, test_name, test_message, test_status)
    VALUES (p_test_id, SUBSTRING_INDEX(p_message, ':', 1), p_message,
p_status);
END //
DELIMITER ;

-- ============================================================
-- TESTS (RN001 - RN016)
-- ============================================================

-- Test RN001: Matrícula de honor requiere nota >= 9
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn001_mh_requirement()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN001', 'RN001: La MH requiere nota >= 9',
'PASS');

    CALL p_populate();

    UPDATE grades SET with_honors = 1 WHERE grade_id = 21;

    CALL p_log_test('RN001', 'ERROR: Se permitió MH con nota inferior a 9',
'FAIL');
END //
DELIMITER ;

-- Test RN002: No se permiten notas duplicadas por asignatura/convocatoria
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn002_duplicate_grade()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN002', 'RN002: No se permiten notas duplicadas
por asignatura/convocatoria', 'PASS');

    CALL p_populate();

    INSERT INTO grades (grade_id, student_id, group_id, grade_value,
exam_call, with_honors)
        VALUES (101, 6, 1, 6.0, 'Primera', 0);

    CALL p_log_test('RN002', 'ERROR: Se permitió duplicar nota en la misma
convocatoria', 'FAIL');
END //
DELIMITER ;
```

```
-- Test RN003: Un grupo no puede tener más de 2 profesores
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn003_professors_per_group()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN003', 'RN003: No se permite añadir un tercer
profesor al grupo', 'PASS');

    CALL p_populate();

    INSERT INTO teaching_loads (professor_id, group_id, credits) VALUES (5,
1, 1.0);

    CALL p_log_test('RN003', 'ERROR: Se asignó un tercer profesor al
grupo', 'FAIL');
END //
DELIMITER ;

-- Test RN004: Un alumno no puede pertenecer a más de un grupo de teoría y
uno de laboratorio por asignatura
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn004_student_group_limit()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN004', 'RN004: Un alumno no puede pertenecer a
más grupos de los permitidos', 'PASS');

    CALL p_populate();

    INSERT INTO group_enrollments (student_id, group_id) VALUES (6, 3);

    CALL p_log_test('RN004', 'ERROR: Se permitió un alumno en más grupos de
los permitidos', 'FAIL');
END //
DELIMITER ;

-- Test RN005: Una nota no puede alterarse en más de 4 puntos
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn005_grade_delta()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN005', 'RN005: No se puede modificar la nota en
más de 4 puntos', 'PASS');

    CALL p_populate();

    UPDATE grades SET grade_value = 0.5 WHERE grade_id = 1;

    CALL p_log_test('RN005', 'ERROR: Se permitió modificar la nota en más
de 4 puntos', 'FAIL');
```

```
END //
DELIMITER ;

-- Test RN006: No puede haber más de un grupo de teoría y dos de
laboratorio por asignatura
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn006_extra_group()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN006', 'RN006: No se permite crear un segundo
grupo de teoría para la asignatura', 'PASS');

    CALL p_populate();

    INSERT INTO groups (group_id, subject_id, group_name, activity,
academic_year)
        VALUES (11, 11, 'T2', 'Teoría', 2024);

    CALL p_log_test('RN006', 'ERROR: Se creó un segundo grupo de teoría
para la misma asignatura', 'FAIL');
END //
DELIMITER ;

-- Test RN007: Un alumno sólo puede pertenecer a grupos de asignaturas en
las que está matriculado
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn007_subject_enrollment()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN007', 'RN007: No se puede añadir a un grupo sin
matrícula en la asignatura', 'PASS');

    CALL p_populate();

    INSERT INTO people (person_id, dni, first_name, last_name, age, email,
role, password_hash)
        VALUES (102, '20000002B', 'Test', 'Alumno', 20, 'nuevo@alum.us.es',
'student', 'pbkdf2_sha256$1$00$00');
    INSERT INTO students (student_id, access_method) VALUES (102,
'Selectividad');
    INSERT INTO group_enrollments (student_id, group_id) VALUES (102, 1);

    CALL p_log_test('RN007', 'ERROR: Se permitió unir a un grupo sin
matrícula en la asignatura', 'FAIL');
END //
DELIMITER ;

-- Test RN008: Un alumno no puede acceder por selectividad con menos de 16
años
DELIMITER //
```

```
CREATE OR REPLACE PROCEDURE p_test_rn008_min_age()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN008', 'RN008: No se permite Selectividad con
menos de 16 años', 'PASS');

    CALL p_populate();

    INSERT INTO people (person_id, dni, first_name, last_name, age, email,
role, password_hash)
        VALUES (104, '20000004D', 'Test', 'Menor', 15, 'menor@alum.us.es',
'student', 'pbkdf2_sha256$1$00$00');
    INSERT INTO students (student_id, access_method) VALUES (104,
'Selectividad');

    CALL p_log_test('RN008', 'ERROR: Se permitió Selectividad para un
menor', 'FAIL');
END //
DELIMITER ;

-- Test RN009: Los atributos obligatorios no pueden quedar a NULL
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn009_not_null_attributes()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN009', 'RN009: Los atributos obligatorios no
pueden quedar a NULL', 'PASS');

    CALL p_populate();

    INSERT INTO people (person_id, dni, first_name, last_name, age, email,
role, password_hash)
        VALUES (103, '20000003C', NULL, 'Campos', 22, 'null@alum.us.es',
'student', 'pbkdf2_sha256$1$00$00');

    CALL p_log_test('RN009', 'ERROR: Se permitió dejar atributos
obligatorios a NULL', 'FAIL');
END //
DELIMITER ;

-- Test RN010: Los créditos de una asignatura pueden ser 6 o 12
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn010_subject_credits()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN010', 'RN010: Los créditos de una asignatura
pueden ser 6 o 12', 'PASS');

    CALL p_populate();
```

```
    INSERT INTO subjects (subject_id, degree_id, subject_name, acronym,
credits, course, subject_type)
        VALUES (31, 3, 'Asignatura Créditos Inválidos', 'ACI', 8, 2,
'Obligatoria');

    CALL p_log_test('RN010', 'ERROR: Se permitió una asignatura con
créditos inválidos', 'FAIL');
END //
DELIMITER ;

-- Test RN011: El valor de la nota está comprendido entre 0 y 10
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn011_grade_range()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN011', 'RN011: La nota debe estar entre 0 y 10',
'PASS');

    CALL p_populate();

    INSERT INTO grades (grade_id, student_id, group_id, grade_value,
exam_call, with_honors)
        VALUES (201, 6, 1, 11.0, 'Primera', 0);

    CALL p_log_test('RN011', 'ERROR: Se permitió una nota fuera de rango',
'FAIL');
END //
DELIMITER ;

-- Test RN012: La edad de las personas está entre 16 y 70 años
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn012_people_age()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN012', 'RN012: La edad de las personas debe estar
entre 16 y 70', 'PASS');

    CALL p_populate();

    INSERT INTO people (person_id, dni, first_name, last_name, age, email,
role, password_hash)
        VALUES (105, '20000005E', 'Edad', 'Fuera', 80, 'edad@us.es',
'student', 'pbkdf2_sha256$1$00$00');

    CALL p_log_test('RN012', 'ERROR: Se permitió una edad fuera de rango',
'FAIL');
END //
DELIMITER ;

-- Test RN013: Los años de un grado están entre 3 y 6
```

```
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn013_degree_years()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN013', 'RN013: Los grados deben tener entre 3 y 6
años', 'PASS');

    CALL p_populate();

    INSERT INTO degrees (degree_id, degree_name, duration_years)
        VALUES (10, 'Grado Experimental', 2);

    CALL p_log_test('RN013', 'ERROR: Se permitió un grado con años fuera de
rango', 'FAIL');
END //
DELIMITER ;

-- Test RN014: El DNI está formado por 8 números y una letra
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn014_dni_format()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN014', 'RN014: El DNI debe tener 8 dígitos y una
letra', 'PASS');

    CALL p_populate();

    INSERT INTO people (person_id, dni, first_name, last_name, age, email,
role, password_hash)
        VALUES (106, 'INVALIDO', 'DNI', 'Incorrecto', 30, 'dni@us.es',
'student', 'pbkdf2_sha256$1$00$00');

    CALL p_log_test('RN014', 'ERROR: Se permitió un DNI con formato
inválido', 'FAIL');
END //
DELIMITER ;

-- Test RN015: El año académico está entre 2000 y 2100
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn015_academic_year()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN015', 'RN015: El año académico debe estar entre
2000 y 2100', 'PASS');

    CALL p_populate();

    INSERT INTO groups (group_id, subject_id, group_name, activity,
academic_year)
        VALUES (20, 12, 'MD-T2025', 'Teoría', 1999);
```

```
    CALL p_log_test('RN015', 'ERROR: Se permitió un año académico fuera de
rango', 'FAIL');
END //
DELIMITER ;

-- Test RN016: El curso de una asignatura está entre 1 y 6
DELIMITER //
CREATE OR REPLACE PROCEDURE p_test_rn016_course_range()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
        CALL p_log_test('RN016', 'RN016: El curso de una asignatura debe
estar entre 1 y 6', 'PASS');

    CALL p_populate();

    INSERT INTO subjects (subject_id, degree_id, subject_name, acronym,
credits, course, subject_type)
        VALUES (30, 3, 'Asignatura Fuera de Curso', 'AFC', 6, 0,
'Obligatoria');

    CALL p_log_test('RN016', 'ERROR: Se permitió un curso fuera de rango',
'FAIL');
END //
DELIMITER ;

-- ============================================================
-- ORQUESTADOR
-- ============================================================
DELIMITER //
CREATE OR REPLACE PROCEDURE p_run_grados_tests()
BEGIN
    DELETE FROM test_results;

    CALL p_test_rn001_mh_requirement();
    CALL p_test_rn002_duplicate_grade();
    CALL p_test_rn003_professors_per_group();
    CALL p_test_rn004_student_group_limit();
    CALL p_test_rn005_grade_delta();
    CALL p_test_rn006_extra_group();
    CALL p_test_rn007_subject_enrollment();
    CALL p_test_rn008_min_age();
    CALL p_test_rn009_not_null_attributes();
    CALL p_test_rn010_subject_credits();
    CALL p_test_rn011_grade_range();
    CALL p_test_rn012_people_age();
    CALL p_test_rn013_degree_years();
    CALL p_test_rn014_dni_format();
    CALL p_test_rn015_academic_year();
    CALL p_test_rn016_course_range();
```

```
    SELECT * FROM test_results ORDER BY execution_time, test_id;
    SELECT test_status, COUNT(*) AS total FROM test_results GROUP BY
test_status;
END //
DELIMITER ;

CALL p_run_grados_tests();
```

> **Info**
>
> Versión PDF disponible