Lab3 - Consultas SQL

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

Escuela Técnica Superior de
Ingeniería Informática

Índice

1. Objetivo 1
2. Preparación del entorno 2
3. Control de versiones 3
4. Estructura del archivo queries.sql 4
5. SELECT: La consulta básica 5
6. SELECT: La consulta básica 6
6.1. Consulta 1: Todas las asignaturas 6
6.2. Consulta 2: Asignatura con acrónimo específico 7
6.3. Consulta 3: Columnas específicas 7
6.4. Consulta 4: Filtrado por tipo 7
7. JOIN: Relacionando tablas 8
7.1. Consulta 5: Estudiantes con JOIN 8
7.2. Consulta 6: Profesores por categoría 8
8. Vistas: Simplificando consultas complejas 10
8.1. Vista v_professors 10
8.2. Vista v_students 11
9. Funciones agregadas 12
9.1. Consulta 7: Media de notas 12
9.2. Consulta 8: Suma de créditos 12
9.3. Consulta 9: Contar registros 13
9.4. Consulta 10: Contar con agrupación básica 13
9.5. Consulta 11: Valor máximo 13
9.6. Consulta 12: Valor mínimo 14
10. Condiciones múltiples 15
10.1. Consulta 13: OR lógico 15
10.2. Consulta 14: BETWEEN 15
10.3. Consulta 15: IN 16
10.4. Consulta 16: Rango de edad 16
11. DISTINCT: Eliminación de duplicados 17
11.1. Consulta 17: Nombres únicos de grupos 17
11.2. Consulta 18: Tipos de asignatura 17
11.3. Consulta 19: Años académicos 18
12. Subconsultas 19

12.1. Consulta 20: Agregado filtrado 19
12.2. Consulta 21: IN con subconsulta 19
12.3. Consulta 22: Comparación con agregado 20
13. Patrones con LIKE 21
13.1. Consulta 23: Terminación de cadena 21
13.2. Consulta 24: Inicio de cadena 21
13.3. Consulta 25: Longitud exacta 22
14. Filtrado por fechas y años 23
14.1. Consulta 26: Filtro por año 23
14.2. Consulta 27: Rango de años 23
15. Vistas complejas 24
15.1. Vista v_grades_with_honors 24
15.2. Vista v_student_grades 24
15.3. Vista v_professor_loads 25
15.4. Vista v_degree_students 26
15.5. Consulta 28: Uso de vistas 26
15.6. Consulta 29: Agregación sobre vista 26
16. ORDER BY, LIMIT y OFFSET: Control de resultados 27
16.1. Consulta 30: Ordenación ascendente 27
16.2. Consulta 31: Ordenación descendente 27
16.3. Consulta 32: Limitación de resultados 28
16.4. Consulta 33: Ordenación múltiple 28
16.5. Consulta 34: Paginación básica 28
16.6. Consulta 35: Tercera página 29
16.7. Consulta 36: Ordenación con criterios mixtos 29
17. JOIN: Consultas avanzadas con múltiples tablas 30
17.1. Consulta 37: JOIN con tabla intermedia 30
17.2. Consulta 38: JOIN uno a muchos 31
17.3. Consulta 39: JOIN múltiple 31
17.4. Consulta 40: JOIN con agregación 31
17.5. Consulta 41: DISTINCT con JOIN 32
17.6. Consulta 42: LEFT JOIN 32
18. GROUP BY: Agrupación y agregación 33
18.1. Consulta 43: Agregación por grupo 33
18.2. Consulta 44: Conteo por categoría 34
18.3. Consulta 45: Agregación con JOIN 34
18.4. Consulta 46: Conteo por entidad 34
18.5. Consulta 47: Promedio por categoría 34
18.6. Consulta 48: Agrupación múltiple 35
18.7. Consulta 49: COUNT DISTINCT 35
18.8. Consulta 50: Suma por grupo 36
19. HAVING: Filtrado de grupos 37
19.1. Consulta 51: Filtro por agregado 37
19.2. Consulta 52: Conteo con umbral 38
19.3. Consulta 53: Filtro de tamaño de grupo 38

19.4. Consulta 54: Promedio con filtro 38
20. Consultas complejas combinadas 39
20.1. Consulta 55: Top N con filtros 39
20.2. Consulta 56: Condición especial con agregado 40
20.3. Consulta 57: Ranking con múltiples JOIN 40
20.4. Consulta 58: Máximo por grupo 40
20.5. Consulta 59: Análisis de carga docente 41
20.6. Consulta 60: Subconsulta con GROUP BY 41
21. Push final 42
22. Ejercicios propuestos 43

1 Objetivo

El objetivo de esta práctica es aprender a realizar consultas SQL para extraer y analizar información de una base de datos relacional. El alumno aprenderá a:

- Realizar consultas básicas con `SELECT` y `WHERE`.
- Utilizar funciones agregadas (`AVG`, `SUM`, `COUNT`, `MAX`, `MIN`).
- Aplicar filtros con condiciones múltiples.
- Usar `DISTINCT` para eliminar duplicados.
- Implementar subconsultas.
- Aplicar patrones de texto con `LIKE`.
- Ordenar y paginar resultados con `ORDER BY`, `LIMIT` y `OFFSET`.
- Realizar consultas con `JOIN` (INNER y LEFT).
- Agrupar resultados con `GROUP BY`.
- Filtrar grupos con `HAVING`.
- Crear y utilizar vistas (`VIEW`).

Hasta ahora hemos creado la estructura de la base de datos y establecido las restricciones de integridad. En esta práctica aprenderemos a extraer información útil mediante consultas SQL, desde las más simples hasta consultas complejas que combinan datos de múltiples tablas.

2 Preparación del entorno

Abre HeidiSQL y conéctate con el usuario `iissi_user` a la base de datos `GradesDB`. Asegúrate de haber ejecutado previamente los scripts `createDB.sql` y `populateDB.sql` de los laboratorios anteriores para tener el esquema completo y datos de prueba.

Para este laboratorio necesitarás datos adicionales. Ejecuta también el script `populateDB2.sql` que añade más grupos, notas y asignaturas para cubrir todos los ejemplos de consultas:

```
-- En HeidiSQL, ejecuta:
SOURCE /ruta/a/tu/repositorio/populateDB2.sql;
-- O simplemente abre el archivo y ejecútalo (F9)
```

Crea un nuevo archivo `queries.sql` en tu repositorio donde irás escribiendo las consultas de este laboratorio.

3

Control de versiones

Continuaremos trabajando con el repositorio `GradesDB` creado en L1. En este laboratorio no es necesario hacer commits tras cada consulta individual. Puedes hacer commits por secciones completas:

```
git add queries.sql
git commit -m "Añadidas consultas básicas (SELECT, WHERE, funciones
agregadas)"
git commit -m "Añadidas consultas con JOIN y GROUP BY"
git commit -m "Añadidas vistas y consultas complejas"
```

O simplemente hacer un commit al finalizar el laboratorio completo:

```
git add queries.sql
git commit -m "Completado L3: Consultas SQL"
```

4 Estructura del archivo queries.sql

Al inicio del archivo, añade la instrucción para usar la base de datos correcta:

```
--
-- Autor: [Tu Nombre]
-- Fecha: Enero 2026
-- Descripción: Consultas SQL de ejemplo para GradesDB
--              Cubre conceptos de consultas simples y avanzadas
--

USE GradesDB;
```

5

SELECT: La consulta básica

La consulta de datos es la operación fundamental en bases de datos. El resultado de una consulta es siempre una tabla con filas y columnas determinadas por la consulta. La estructura básica de una consulta es:

6

SELECT: La consulta básica

La consulta de datos es la operación fundamental en bases de datos. El resultado de una consulta es siempre una tabla con filas y columnas determinadas por la consulta. La estructura básica de una consulta es:

```
SELECT columnas FROM tabla WHERE condiciones;
```

6.1. Consulta 1: Todas las asignaturas

Recupera todos los campos de todas las asignaturas en la base de datos.

```
SELECT *
FROM subjects;
```

Observe lo siguiente:

- El operador `*` selecciona todas las columnas de la tabla.
- Es útil para exploración inicial, pero en producción es mejor especificar las columnas necesarias.
- La consulta devuelve una tabla con todas las filas y columnas de `subjects`.

6.2. Consulta 2: Asignatura con acrónimo específico

Busca la asignatura con acrónimo 'IISSI-1'.

```
SELECT *
FROM subjects s
WHERE s.acronym = 'IISSI-1';
```

Observe lo siguiente:

- El alias `s` permite referirse a la tabla de forma más breve.
- La cláusula `WHERE` filtra los resultados según una condición.
- Solo se devuelven las filas que cumplan la condición.

6.3. Consulta 3: Columnas específicas

Recupera solo las columnas de nombre y acrónimo de todas las asignaturas.

```
SELECT s.subject_name, s.acronym
FROM subjects s;
```

Observe lo siguiente:

- Seleccionar solo las columnas necesarias mejora el rendimiento.
- Cada columna se separa por comas.
- El prefijo `s.` indica explícitamente de qué tabla proviene cada columna.

6.4. Consulta 4: Filtrado por tipo

Filtra las asignaturas que son de tipo 'Formación Básica'.

```
SELECT s.subject_name, s.acronym, s.subject_type
FROM subjects s
WHERE s.subject_type = 'Formación Básica';
```

7

JOIN: Relacionando tablas

Muchas veces necesitamos combinar información de varias tablas relacionadas. Para ello usamos `JOIN`.

7.1. Consulta 5: Estudiantes con JOIN

Recupera los estudiantes que accedieron por Selectividad, mostrando su información personal completa.

```sql
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';
```

Observe lo siguiente:

- `JOIN` combina dos tablas basándose en una condición de relación.
- `ON` especifica cómo se relacionan las tablas (clave ajena = clave primaria).
- Podemos acceder a columnas de ambas tablas en el `SELECT`.
- La condición del `WHERE` se evalúa después de realizar el `JOIN`.

7.2. Consulta 6: Profesores por categoría

Busca todos los profesores con categoría 'Catedrático'.

```sql
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';
```

8

Vistas: Simplificando consultas complejas

En ocasiones, es útil guardar los resultados de una consulta para luego utilizarlos en otras consultas como si fueran una tabla más. Esto se logra mediante **vistas** (`VIEW`), que dan un nombre a una consulta `SELECT` reutilizable.

8.1. Vista v_professors

Combina información de profesores con sus datos personales.

```
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;
```

Observe lo siguiente:

- `CREATE OR REPLACE VIEW` crea una nueva vista o reemplaza una existente.
- La vista ejecuta el `SELECT` cada vez que se consulta.
- Una vez creada, podemos usar `v_professors` como si fuera una tabla normal.

8.2. Vista v_students

Combina información de estudiantes con sus datos personales.

```
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;
```

Estas vistas evitan tener que escribir el `JOIN` con `people` repetidamente en consultas posteriores.

9 Funciones agregadas

Las funciones agregadas permiten calcular valores sobre conjuntos de filas: promedios, sumas, conteos, máximos y mínimos. Al usar estas funciones, normalmente se devuelve una sola fila con el resultado agregado.

9.1. Consulta 7: Media de notas

Calcula la nota media del grupo con ID 1.

```
SELECT AVG(g.grade_value) AS average_grade
FROM grades g
WHERE g.group_id = 1;
```

Observe lo siguiente:

- `AVG()` calcula el promedio aritmético de los valores.
- El alias `AS average_grade` renombra la columna en el resultado.
- La función agregada opera sobre todas las filas que cumplen el `WHERE`.
- El resultado es una única fila con un único valor.

9.2. Consulta 8: Suma de créditos

Suma todos los créditos de las asignaturas del grado con ID 3 (Tecnologías Informáticas).

```sql
SELECT SUM(s.credits) AS total_credits
FROM subjects s
WHERE s.degree_id = 3;
```

Observe lo siguiente:

- `SUM()` suma todos los valores numéricos de la columna especificada.
- Útil para calcular totales.

9.3. Consulta 9: Contar registros

Cuenta cuántos estudiantes hay en la base de datos.

```sql
SELECT COUNT(*) AS total_students
FROM students;
```

Observe lo siguiente:

- `COUNT(*)` cuenta el número de filas.
- `COUNT(columna)` contaría solo valores no nulos en esa columna.
- Sin `WHERE`, cuenta todas las filas de la tabla.

9.4. Consulta 10: Contar con agrupación básica

Cuenta cuántos grupos hay de cada tipo de actividad.

```sql
SELECT activity, COUNT(*) AS total_groups
FROM groups
GROUP BY activity;
```

Observe lo siguiente:

- `GROUP BY` agrupa las filas por el valor de `activity`.
- `COUNT(*)` cuenta filas en cada grupo.
- El resultado tiene una fila por cada valor distinto de `activity`.
- Las columnas en `SELECT` deben ser o agregadas o aparecer en `GROUP BY`.

9.5. Consulta 11: Valor máximo

Encuentra la nota más alta registrada en la base de datos.

```
SELECT MAX(g.grade_value) AS max_grade
FROM grades g;
```

9.6. Consulta 12: Valor mínimo

Encuentra la nota más baja registrada.

```
SELECT MIN(g.grade_value) AS min_grade
FROM grades g;
```

10

Condiciones múltiples

Las cláusulas `WHERE` pueden combinar múltiples condiciones con operadores lógicos (`AND` , `OR`) y operadores de rango (`BETWEEN` , `IN`).

10.1. Consulta 13: OR lógico

Busca notas que sean suspensos (< 5) o excelentes (> 9).

```
SELECT g.grade_id, g.grade_value, g.exam_call
FROM grades g
WHERE g.grade_value < 5 OR g.grade_value > 9;
```

Observe lo siguiente:

- El operador `OR` devuelve filas que cumplan al menos una de las condiciones.
- Se pueden combinar muchas condiciones con `AND` y `OR` .
- Es recomendable usar paréntesis para mayor claridad cuando se mezclan ambos operadores.

10.2. Consulta 14: BETWEEN

Busca notas entre 5 y 7 (aprobados y notables bajos).

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

Observe lo siguiente:

- `BETWEEN` incluye ambos extremos (5 y 7).
- Equivalente a `grade_value >= 5 AND grade_value <= 7`.
- Más legible para rangos numéricos o de fechas.

10.3. Consulta 15: IN

Busca asignaturas que tengan exactamente 6 o 12 créditos.

```sql
SELECT s.subject_name, s.credits
FROM subjects s
WHERE s.credits IN (6, 12);
```

Observe lo siguiente:

- `IN` comprueba si el valor está en una lista de valores.
- Equivalente a `credits = 6 OR credits = 12`.
- Más conciso cuando hay muchos valores posibles.

10.4. Consulta 16: Rango de edad

Busca personas entre 20 y 30 años.

```sql
SELECT p.first_name, p.last_name, p.age
FROM people p
WHERE p.age BETWEEN 20 AND 30;
```

11

DISTINCT: Eliminación de duplicados

`DISTINCT` elimina filas duplicadas del resultado. Es útil cuando un `JOIN` o agrupación puede producir repeticiones.

11.1. Consulta 17: Nombres únicos de grupos

Obtiene la lista de nombres de grupos sin repeticiones.

```
SELECT DISTINCT g.group_name
FROM groups g;
```

Observe lo siguiente:

- Sin `DISTINCT`, si varios grupos tienen el mismo nombre (pero diferente año o actividad), aparecerían múltiples veces.
- `DISTINCT` se aplica a toda la fila, no a columnas individuales.

11.2. Consulta 18: Tipos de asignatura

Lista los diferentes tipos de asignatura existentes.

```
SELECT DISTINCT s.subject_type
FROM subjects s;
```

11.3. Consulta 19: Años académicos

Lista los años académicos para los que hay grupos creados, ordenados descendentemente.

```
SELECT DISTINCT g.academic_year
FROM groups g
ORDER BY g.academic_year DESC;
```

Observe lo siguiente:

- `ORDER BY` ordena los resultados.
- `DESC` indica orden descendente (de mayor a menor).
- `ASC` sería ascendente (por defecto si se omite).

12 Subconsultas

Una subconsulta es una consulta dentro de otra consulta. Permite usar el resultado de una consulta como parte de la condición de otra.

12.1. Consulta 20: Agregado filtrado

Encuentra la nota más alta del estudiante con ID 6.

```
SELECT MAX(g.grade_value) AS max_grade
FROM grades g
WHERE g.student_id = 6;
```

12.2. Consulta 21: IN con subconsulta

Encuentra qué estudiantes están matriculados en grupos del año 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
);
```

Observe lo siguiente:

- La subconsulta (entre paréntesis) se ejecuta primero.
- Devuelve una lista de IDs de grupos del 2024.
- La consulta principal busca estudiantes en esos grupos.
- El `DISTINCT` evita repetir estudiantes que estén en varios grupos.

12.3. Consulta 22: Comparación con agregado

Encuentra asignaturas que tengan más créditos que el promedio.

```
SELECT s.subject_name, s.credits
FROM subjects s
WHERE s.credits > (SELECT AVG(credits) FROM subjects);
```

Observe lo siguiente:

- La subconsulta calcula el promedio de créditos.
- La consulta principal compara cada asignatura con ese promedio.
- La subconsulta debe devolver un único valor para poder usarse con `>`.

13

Patrones con LIKE

El operador `LIKE` permite buscar patrones de texto usando comodines:

- `%` representa cualquier secuencia de caracteres (incluyendo ninguno).
- `_` representa exactamente un carácter.

13.1. Consulta 23: Terminación de cadena

Busca personas cuyo DNI termina en la letra A.

```
SELECT p.first_name, p.last_name, p.dni
FROM people p
WHERE p.dni LIKE '%A';
```

Observe lo siguiente:

- `%A` significa “cualquier cosa seguida de A”.
- Es case-insensitive en MariaDB por defecto.

13.2. Consulta 24: Inicio de cadena

Filtra asignaturas cuyo nombre empieza con la palabra “Fundamentos”.

```sql
SELECT s.subject_name
FROM subjects s
WHERE s.subject_name LIKE 'Fundamentos%';
```

Observe lo siguiente:

- `Fundamentos%` significa “Fundamentos seguido de cualquier cosa”.

13.3. Consulta 25: Longitud exacta

Busca personas cuyo nombre tiene exactamente 5 letras.

```sql
SELECT p.first_name, p.last_name
FROM people p
WHERE p.first_name LIKE '_____';
```

Observe lo siguiente:

- Cada `_` representa exactamente un carácter.
- `_____` (cinco guiones bajos) = exactamente 5 caracteres.

14 Filtrado por fechas y años

Las consultas pueden filtrar valores numéricos que representan años o trabajar con fechas.

14.1. Consulta 26: Filtro por año

Busca grupos del año académico 2024.

```
SELECT g.group_name, g.activity, g.academic_year
FROM groups g
WHERE g.academic_year = 2024;
```

14.2. Consulta 27: Rango de años

Busca grupos entre 2020 y 2023, ordenados por año.

```
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;
```

15

Vistas complejas

Podemos crear vistas que combinan información de múltiples tablas para consultas frecuentes.

15.1. Vista v_grades_with_honors

Filtra solo las 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;
```

15.2. Vista v_student_grades

Información completa de notas de estudiantes.

```
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;
```

Observe lo siguiente:

- Esta vista realiza múltiples `JOIN` en cascada.
- Cada `JOIN` conecta dos tablas mediante su relación.
- Una vez creada, podemos consultar `v_student_grades` como una tabla normal.
- Simplifica enormemente consultas que necesiten esta información combinada.

15.3. Vista v_professor_loads

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;
```

15.4. Vista v_degree_students

Estudiantes por grado con su información completa.

```
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;
```

15.5. Consulta 28: Uso de vistas

Cuenta cuántas matrículas de honor hay en total usando la vista creada.

```
SELECT COUNT(*) AS total_honors
FROM v_grades_with_honors;
```

Observe lo siguiente:

- Usar vistas simplifica las consultas.
- No necesitamos recordar la condición `with_honors = 1`.
- La vista se comporta como una tabla normal en las consultas.

15.6. Consulta 29: Agregación sobre vista

Calcula la nota media de cada estudiante usando `v_student_grades`.

```
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;
```

16

ORDER BY, LIMIT y OFFSET: Control de resultados

Estas cláusulas controlan el orden y paginación de los resultados.

16.1. Consulta 30: Ordenación ascendente

Ordena todas las notas de menor a mayor valor.

```
SELECT g.grade_id, g.student_id, g.grade_value, g.exam_call
FROM grades g
ORDER BY g.grade_value ASC;
```

Observe lo siguiente:

- `ORDER BY` ordena los resultados según una o más columnas.
- `ASC` (ascending) es el orden por defecto.
- Se escribe explícitamente por claridad.

16.2. Consulta 31: Ordenación descendente

Ordena las notas de mayor a menor.

```
SELECT g.grade_id, g.student_id, g.grade_value, g.exam_call
FROM grades g
ORDER BY g.grade_value DESC;
```

16.3. Consulta 32: Limitación de resultados

Muestra solo las 5 notas más altas.

```
SELECT g.grade_id, g.student_id, g.grade_value, g.exam_call
FROM grades g
ORDER BY g.grade_value DESC
LIMIT 5;
```

Observe lo siguiente:

- `LIMIT` restringe el número de filas devueltas.
- Muy útil para "top N" consultas.
- Siempre se usa con `ORDER BY` para resultados consistentes.

16.4. Consulta 33: Ordenación múltiple

Lista notas aprobadas ordenadas por apellido y nombre del 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;
```

Observe lo siguiente:

- Se puede ordenar por múltiples columnas separadas por comas.
- Primero ordena por apellido; si hay empate, entonces por nombre.

16.5. Consulta 34: Paginación básica

Implementa paginación mostrando las notas 6-10 (segunda página de 5 elementos).

```
SELECT g.grade_id, g.student_id, g.grade_value
FROM grades g
ORDER BY g.grade_value DESC
LIMIT 5 OFFSET 5;
```

Observe lo siguiente:

- `OFFSET 5` salta las primeras 5 filas.
- `LIMIT 5` devuelve las siguientes 5 filas.
- Para página N de tamaño P: `LIMIT P OFFSET (N-1)*P`.

16.6. Consulta 35: Tercera página

Muestra las notas 11-15 (tercera página).

```
SELECT g.grade_id, g.student_id, g.grade_value
FROM grades g
ORDER BY g.grade_value DESC
LIMIT 5 OFFSET 10;
```

16.7. Consulta 36: Ordenación con criterios mixtos

Ordena grupos por año (descendente) y luego por nombre (ascendente).

```
SELECT g.group_name, g.activity, g.academic_year
FROM groups g
ORDER BY g.academic_year DESC, g.group_name ASC;
```

17

JOIN: Consultas avanzadas con múltiples tablas

Los `JOIN` permiten combinar filas de diferentes tablas basándose en relaciones entre ellas.

17.1. Consulta 37: JOIN con tabla intermedia

Lista estudiantes y los grupos en los que están matriculados.

```
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;
```

Observe lo siguiente:

- Esta consulta realiza dos `JOIN` consecutivos.
- Primero conecta estudiantes con la tabla intermedia de matrículas.
- Luego conecta matrículas con grupos.
- Esto es típico en relaciones N:M.

17.2. Consulta 38: JOIN uno a muchos

Muestra asignaturas junto con el nombre de su 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;
```

17.3. Consulta 39: JOIN múltiple

Muestra notas con nombres de 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;
```

17.4. Consulta 40: JOIN con agregación

Lista profesores y los grupos que enseñan.

```
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;
```

17.5. Consulta 41: DISTINCT con JOIN

Muestra qué estudiantes están matriculados en qué asignaturas.

```
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;
```

Observe lo siguiente:

- `DISTINCT` elimina duplicados.
- Un estudiante podría estar en múltiples grupos de la misma asignatura.
- Sin `DISTINCT`, aparecería una fila por cada grupo.

17.6. Consulta 42: LEFT JOIN

Muestra todos los estudiantes, incluso si no tienen notas registradas.

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

Observe lo siguiente:

- `LEFT JOIN` incluye todas las filas de la tabla izquierda (`v_students`).
- Aunque no haya coincidencias en la tabla derecha (`grades`).
- Las columnas de `grades` serán `NULL` para estudiantes sin notas.
- `INNER JOIN` (o simplemente `JOIN`) solo incluye filas con coincidencia en ambas tablas.

18

GROUP BY: Agrupación y agregación

`GROUP BY` agrupa filas con valores iguales en columnas especificadas, permitiendo calcular agregados por grupo.

18.1. Consulta 43: Agregación por grupo

Calcula la nota media y total de notas por 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;
```

Observe lo siguiente:

- Todas las columnas no agregadas deben estar en `GROUP BY`.
- La consulta devuelve una fila por cada combinación única de valores en `GROUP BY`.
- Las funciones agregadas calculan valores para cada grupo.

18.2. Consulta 44: Conteo por categoría

Cuenta cuántos estudiantes hay por cada método de acceso.

```
SELECT
    st.access_method,
    COUNT(*) AS total_students
FROM students st
GROUP BY st.access_method;
```

18.3. Consulta 45: Agregación con JOIN

Cuenta cuántas asignaturas tiene cada 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;
```

18.4. Consulta 46: Conteo por entidad

Cuenta cuántos grupos tiene cada 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;
```

18.5. Consulta 47: Promedio por categoría

Calcula la nota media en cada convocatoria de examen.

```
SELECT
    g.exam_call,
    AVG(g.grade_value) AS average_grade,
```

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

18.6. Consulta 48: Agrupación múltiple

Desglosa la nota media por asignatura y 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;
```

Observe lo siguiente:

- Agrupamos por dos dimensiones: asignatura y convocatoria.
- Obtenemos estadísticas separadas para cada combinación.

18.7. Consulta 49: COUNT DISTINCT

Cuenta cuántos estudiantes distintos hay en cada 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;
```

Observe lo siguiente:

- `COUNT(DISTINCT ...)` cuenta valores únicos.
- Evita contar dos veces un estudiante con múltiples grupos.
- Sin `DISTINCT`, contaría matrículas, no estudiantes únicos.

18.8. Consulta 50: Suma por grupo

Suma los créditos que imparte cada 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;
```

19

HAVING: Filtrado de grupos

`HAVING` filtra grupos después de `GROUP BY`. A diferencia de `WHERE`, que filtra filas antes de agrupar, `HAVING` filtra grupos después de calcular agregados.

19.1. Consulta 51: Filtro por agregado

Muestra solo estudiantes cuya nota media supera el 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;
```

Observe lo siguiente:

- `HAVING` se evalúa después de calcular `AVG` para cada grupo.
- Solo se devuelven grupos que cumplan la condición del `HAVING`.
- No se puede usar `WHERE` para filtrar por agregados.

19.2. Consulta 52: Conteo con umbral

Filtra asignaturas que tienen más de 3 grupos creados.

```
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(*) > 3;
```

19.3. Consulta 53: Filtro de tamaño de grupo

Identifica grupos con más de 15 estudiantes matriculados.

```
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(*) > 15;
```

19.4. Consulta 54: Promedio con filtro

Filtra asignaturas cuya nota media es superior a 5.5.

```
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) > 5.5;
```

20 Consultas complejas combinadas

Estas consultas combinan múltiples conceptos vistos: `JOIN`, `GROUP BY`, `HAVING`, `ORDER BY`, `LIMIT`, subconsultas.

20.1. Consulta 55: Top N con filtros

Encuentra las tres asignaturas con más grupos de teoría en el año 2024.

```sql
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;
```

Observe lo siguiente:

- `WHERE` filtra antes de agrupar (solo grupos de teoría de 2024).
- `GROUP BY` agrupa por asignatura.
- `ORDER BY` ordena por el conteo.
- `LIMIT` devuelve solo las 3 primeras.
- Secuencia: FROM → WHERE → GROUP BY → ORDER BY → LIMIT.

20.2. Consulta 56: Condición especial con agregado

Identifica estudiantes con más de 1 matrícula 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(*) > 1;
```

20.3. Consulta 57: Ranking con múltiples JOIN

Ordena grados por número de estudiantes matriculados 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;
```

20.4. Consulta 58: Máximo por grupo

Muestra la nota más alta de cada estudiante, ordenada descendentemente.

```
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;
```

20.5. Consulta 59: Análisis de carga docente

Identifica profesores que imparten en más de 1 grupo.

```
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) > 1;
```

20.6. Consulta 60: Subconsulta con GROUP BY

Muestra asignaturas que pertenecen a grados con más de 15 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(*) > 15
);
```

Observe lo siguiente:

- La subconsulta encuentra grados grandes usando `GROUP BY` y `HAVING`.
- La consulta principal filtra asignaturas de esos grados.
- Combina subconsultas con agregación.

21

Push final

Antes de finalizar el laboratorio, verifica que tu archivo `queries.sql` esté completo y funcionando:

1. **Ejecuta todas las consultas** en HeidiSQL para verificar que no hay errores de sintaxis
2. **Revisa las vistas creadas**: Asegúrate de que todas las vistas se han creado correctamente
3. **Prueba las consultas complejas**: Verifica que las consultas con múltiples JOIN devuelven resultados coherentes

Una vez verificado todo:

```
git add -A
git commit -m "Completado L3: Consultas SQL con vistas y agregaciones"
git push
```

22

Ejercicios propuestos

Para practicar, intenta crear las siguientes consultas adicionales:

1. Estudiantes que nunca han suspendido (todas sus notas ≥ 5).
2. Profesores que solo imparten teoría (no laboratorios).
3. Asignaturas sin grupos creados (usar LEFT JOIN).
4. Nota media de cada grado en el año 2024.
5. Estudiantes con la mejor nota de cada asignatura.
6. Grupos ordenados por número de estudiantes matriculados.
7. Profesores con carga docente superior a 15 créditos.
8. Año académico con más matrículas de honor.
9. Asignaturas con mayor dispersión de notas (usando STD o VARIANCE).
10. Estudiantes matriculados en todas las asignaturas de primer curso.

> **Info**
>
> Versión PDF disponible