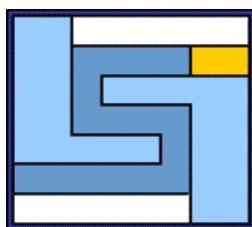




Escuela Técnica Superior de
Ingeniería Informática



Lab. 04: Instrucciones SQL I

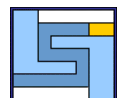
Introducción a la Ingeniería del Software y los Sistemas de
Información I

Curso 2020/21

David Ruiz, Inma Hernández, Agustín Borrego, Daniel Ayala

Índice

1	Objetivo	1
2	Preparación del entorno	1
3	INSERT	2
4	UPDATE	3
5	DELETE	4
6	SELECT	4
7	Vistas	6
8	Consultas varias	7
9	Ejercicios	9



1. Objetivo

El objetivo de esta práctica es manejar los datos almacenados en la base de datos mediante scripts SQL. El alumno aprenderá a:

- Usar INSERT para insertar filas.
- Usar UPDATE para actualizar filas.
- Usar DELETE para borrar filas.
- Usar SELECT para consultar filas.

Hasta ahora hemos creado las tablas que darán soporte a los datos, pero no hemos introducido datos en ellas, ni los hemos manejado. En esta práctica usaremos instrucciones para insertar, alterar, borrar y consultar las filas de cada tabla.

Durante la práctica se usará el siguiente modelo relacional:

Introducción a la Ingeniería del Software y Sistemas de Información I
Proyecto Curso 2019-20

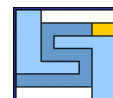
Modelo Entidad-Relación

Grado: Degrees(degreeld (PK), name, duration)
 Espacio::Despacho: Offices(officeld (PK), name, floor, capacity)
 Espacio::Aula: Classrooms(classroomld (PK), name, floor, capacity, hasProjector, hasLoudspeakers)
 Departamento: Departments(departmentld (PK), name)
 Persona::Alumno: Students(studentld (PK), accessMethod, dni, firstnamne, surname, birthDate, email)
 Persona::Profesor: Proffesors(proffesorld (PK), officeld(FK), departmentld (FK), category, dni, firstnamne, surname, birthDate, email)
 Tutoría: TutoringsHours(tutoringHoursld (PK), proffesorld(FK), dayOfWeek, startHour, endHour)
 Carga: TeachingLoads(teachingLoadld (PK), proffesorld(FK), groupld (FK), credits)
 Cita: Appointments(appointmentld (PK), tutoringHoursld (FK), studentld (FK), hour, date)
 Asignatura: Subjects(subjectld (PK), departmentld (FK), gradeld (FK), name, acronym, credits, year, type)
 Nota: Grades(gradeld (PK), studentld (FK), groupld (FK), value, call, withHonours)
 Grupo: Groups(groupld (PK), subjectld (FK), name, activity, academicYear)
 matriculadoEn: StudentsGroups(studentGroupld (PK), studentld (FK), groupld (FK))

Septiembre de 2019

2. Preparación del entorno

Conéctese a la base de datos y ejecute el archivo `tables.sql` contra la base de datos grados. Cree un archivo `queries.sql` en el que se escribirán las instrucciones que se irán desarrollando en esta práctica. Antes de cada ejecución de sus consultas, se recomienda ejecutar el script de creación de tablas, para que éstas se encuentren en un estado controlado antes de la consulta o modificación.



3. INSERT

Para insertar filas en una tabla, usamos INSERT de la siguiente manera:

```
14 INSERT INTO Degrees VALUES (NULL, 'Tecnologías Informáticas', 4);
```

Observe lo siguiente:

- Se indica primero el nombre de la tabla, y luego los valores de la fila separados por comas y entre paréntesis.
- Por defecto, hay que introducir los valores en el orden que tienen en la tabla.
- Al ID le damos valor NULL, para que se le de un valor automático con incremento. Si diéramos un número, se usaría ese en vez del generado automáticamente.
- Al escribir cadenas de texto deben usarse comillas simples. Aunque MariaDB técnicamente permite usar comillas dobles, no están aconsejadas ya que en lenguajes y SGBD similares causan errores.

Añadamos algunas filas a todas las tablas:

```
1 INSERT INTO Degrees (name, years) VALUES
2   ('Ingeniería del Software', 4),
3   ('Ingeniería del Computadores', 4),
4   ('Tecnologías Informáticas', 4);
5
6 INSERT INTO Subjects (name, acronym, credits, year, type, degreeId) VALUES
7   ('Fundamentos de Programación', 'FP', 12, 1, 'Formacion Basica', 3),
8   ('Lógica Informatica', 'LI', 6, 2, 'Optativa', 3);
9
10 INSERT INTO Groups (name, activity, year, subjectId) VALUES
11   ('T1', 'Teoria', 2019, 1),
12   ('L1', 'Laboratorio', 2019, 1),
13   ('L2', 'Laboratorio', 2019, 1);
14
15 INSERT INTO Students (accessMethod, dni, firstname, surname, birthdate, email) VALUES
16   ('Selectividad', '12345678A', 'Daniel', 'Pérez', '1991-01-01', 'daniel@alum.us.es'),
17   ('Selectividad', '22345678A', 'Rafael', 'Ramírez', '1992-01-01', 'rafael@alum.us.es'),
18   ('Selectividad', '32345678A', 'Gabriel', 'Hernández', '1993-01-01', 'gabriel@alum.us.es');
19
20 INSERT INTO GroupsStudents (groupId, studentId) VALUES
21   (1, 1),
22   (3, 1);
23
24 INSERT INTO Grades (value, gradeCall, withHonours, studentId, groupId) VALUES
25   (4.50, 1, 0, 1, 1);
```



Observe lo siguiente:

- Hemos añadido a las sentencias INSERT, antes de los valores, las columnas a las que vamos a dar valor. A las columnas no indicadas se les da valor NULL (o el default). El orden de las columnas especificadas no tiene por qué coincidir con el que tienen en la tabla, pero sí debe ser coherente con los valores que se indiquen a continuación.
- En un mismo INSERT a una tabla se pueden introducir varias filas separadas por comas.
- Las fechas se introducen como cadenas, siguiendo el formato YYYY-MM-DD.
- Los booleanos se introducen como valores numéricos 0 o 1.

4. UPDATE

Para modificar una o varias filas, usamos UPDATE de la siguiente manera:

```
42 UPDATE Students
43     SET birthdate='1998-01-01', surname='Fernández'
44     WHERE studentId=3;
```

Observe lo siguiente:

- Se pueden actualizar varios atributos a la vez.
- La query tiene tres partes: la tabla afectada, los atributos que van a ser modificados, y una condición limitando las filas afectadas.
- Si omitimos la cláusula WHERE, todas las filas serán actualizadas. Cuidado.

Las actualizaciones también pueden usar los valores antiguos al dar nuevos valores, por ejemplo, la siguiente actualización reduce a la mitad los créditos de todas las asignaturas:

```
46 UPDATE Subjects
47     SET credits = credits/2;
```



5. DELETE

Para borrar filas de una tabla se usa la instrucción `DELETE FROM`. Podemos borrar filas de la siguiente manera:

```
49 DELETE FROM Grades
50 WHERE gradeId = 1;
```

Observe lo siguiente:

- La query tiene dos partes: la tabla afectada, y una condición limitando las filas borradas.
- **¡Cuidado!** Si omitimos la cláusula `WHERE`, todas las filas de la tabla serán borradas.
- Por defecto, no se puede borrar una fila que esté referenciada por otra mediante una clave ajena. Habría primero que eliminar la referencia.

Si queremos que al borrar una fila se borren aquellas filas que la referencian mediante claves ajenas (en vez de producirse un error), tenemos que activar el borrado en cascada mediante `ON DELETE CASCADE`, como se explicó en el boletín anterior. Nótese que, en cualquier caso, se mantiene la integridad referencial de la base de datos, impidiéndose que existan referencias a filas que no existen.

6. SELECT

La consulta de datos de una base de datos es fundamental. El resultado de este tipo de consultas es siempre una tabla con filas y columnas determinadas por la consulta.

Por ejemplo, podríamos seleccionar los nombres y apellidos de alumnos de acceso por selectividad de la siguiente manera:

```
55 SELECT firstname, surname
56 FROM Students
57 WHERE accessMethod = 'Selectividad';
```

firstname	surname
Daniel	Pérez
Rafael	Ramírez
Gabriel	Fernández



Observe lo siguiente:

- La query tiene tres partes: las columnas a obtener, las tablas implicadas (puede haber varias), y una condición (opcional) limitando las filas seleccionadas.
- Si queremos obtener todas las columnas presentes en la tabla seleccionada, podemos usar *: `SELECT * FROM...`

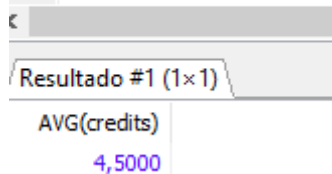
Las columnas a seleccionar no tienen por qué ser sólo las ya existentes en las tablas. Podemos definir cálculos a devolver como columnas. Por ejemplo:

```
59 SELECT credits > 3
60 FROM Subjects;
```

En ese caso, el valor devuelto será un boolean, indicando si en cada fila el número de créditos es mayor que 3.

También podemos pedir valores agregados (sumas, medias, etc.). Pedir estos valores implica que solo se devolverá una fila. Si además solo se pide una columna, se devolverá un valor único:

```
62 SELECT AVG(credits)
63 FROM Subjects;
```

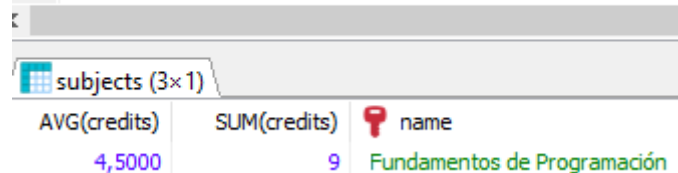


Resultado #1 (1x1)

AVG(credits)
4,5000

Observe lo que ocurre cuando se piden como columnas valores agregados junto con una columna de la tabla:

```
65 SELECT AVG(credits), SUM(credits), name
66 FROM Subjects;
```



subjects (3x1)

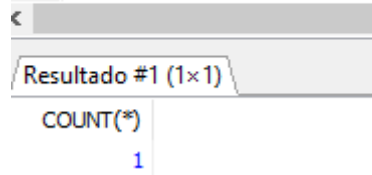
AVG(credits)	SUM(credits)	name
4,5000	9	Fundamentos de Programación



Al pedirse valores agregados, solo se devuelve una fila, que corresponde al valor en cuestión calculado para toda la tabla. El nombre devuelto es simplemente el de la primera de las filas. No tiene sentido pedir valores agregados junto con atributos que cambian de fila a fila.

Uno de los valores agregados más útiles es COUNT. En su variante más común, cuenta el número de filas devueltas:

```
65 SELECT COUNT(*)
66     FROM Subjects
67     WHERE credits > 4;
```

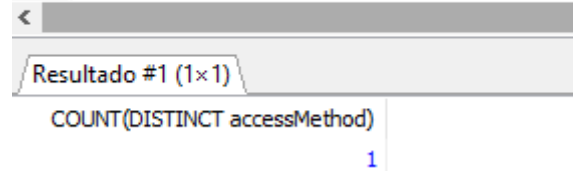


Resultado #1 (1x1)

COUNT(*)
1

Sin embargo, podemos incluir una expresión que limite las filas que se están contando:

```
70 SELECT COUNT(DISTINCT accessMethod)
71     FROM Students;
```



Resultado #1 (1x1)

COUNT(DISTINCT accessMethod)
1

7. Vistas

En ocasiones, es útil guardar los resultados de una consulta para luego utilizarlos en otras consultas como si fueran una tabla más. De esta manera se simplifica en gran medida la creación de consultas anidadas complejas. Esto se puede hacer mediante vistas, en las que damos un nombre a una consulta SELECT que luego se puede usar como si fuera una tabla.

Por ejemplo, podríamos crear una vista que contenga las notas del grupo con ID 18:

```
123 CREATE OR REPLACE VIEW ViewGradesGroup18 AS
124     SELECT * FROM Grades WHERE groupId = 18;
```

Y a continuación usarla en diferentes consultas:




```
126 SELECT MAX(value) FROM ViewGradesGroup18;  
127 SELECT COUNT(*) FROM ViewGradesGroup18;  
128 SELECT * FROM ViewGradesGroup18 WHERE gradeCall = 2;
```

También podemos usar una vista dentro de otra:

```
130 CREATE OR REPLACE VIEW ViewGradesGroup18Call11 AS  
131     SELECT * FROM ViewGradesGroup18 WHERE gradeCall = 1;  
132  
133 SELECT * FROM ViewGradesGroup18Call11;
```

8. Consultas varias

Las posibilidades a la hora de realizar consultas son casi ilimitadas. A continuación se realizarán una serie de consultas que cubren muchos de los casos posibles. Antes de ejecutar las consultas, ejecute el archivo `populate.sql`, que inserta una mayor cantidad de datos en las tablas.

- Todas las asignaturas.

```
SELECT * FROM Subjects;
```

- Asignatura con acrónimo 'FP'.

```
SELECT * FROM Subjects WHERE acronym='FP';
```

- Nombres y acrónimos de todas las asignaturas.

```
SELECT name, acronym FROM Subjects;
```

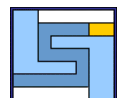
- Media de las notas del grupo con ID 18.

```
SELECT AVG(value) FROM Grades WHERE groupId=18;
```

- Total de créditos de las asignaturas del grado de Tecnologías Informáticas (ID 3).

```
SELECT SUM(credits) FROM Subjects WHERE degreeId=3;
```

- Notas con valor menor que 4 o mayor que 6.



```
SELECT * FROM Grades WHERE value < 4 OR value > 6;
```

- Nombres de grupos diferentes.

```
SELECT DISTINCT name FROM Groups;
```

- Máxima nota del alumno con ID 1.

```
SELECT MAX(VALUE) FROM Grades WHERE studentId=1;
```

- Alumnos con un apellido igual al acrónimo de alguna asignatura.

```
SELECT * FROM Students WHERE surname IN (SELECT acronym FROM Subjects);
```

- IDs de alumnos del curso 2019.

```
SELECT DISTINCT(StudentId)
FROM GroupsStudents
WHERE groupId IN (SELECT groupId FROM Groups WHERE year = 2019);
```

- Alumnos con un DNI terminado en la letra C. Observe cómo % representa cualquier cantidad de caracteres.

```
SELECT *
FROM Students
WHERE dni LIKE('%C')
```

- Alumnos con un nombre de 6 letras. Observe cómo _ representa un caracter cualquiera.

```
SELECT *
FROM Students
WHERE firstName LIKE('_____') -- 6 guiones bajos
```

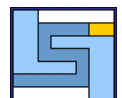
- Alumnos nacidos antes de 1995.

```
SELECT *
FROM Students
WHERE YEAR(birthdate) < 1995
```

- Alumnos nacidos entre enero y febrero.

```
SELECT *
FROM Students
WHERE (MONTH(birthdate) >= 1 AND MONTH(birthdate) <= 2)
```

Observe cómo en las dos últimas consultas es posible incluir otra consulta dentro de la condición de la primera, en lugar de usar valores establecidos a mano.



9. Ejercicios

A las consultas anteriores, añada otras que obtengan lo siguiente:

- Nombre de las asignaturas que son obligatorias.
- Media de las notas del grupo con ID 19, usando el agregador AVG.
- La misma consulta anterior, sin usar AVG.
- Cantidad de nombres de grupo diferentes.
- Notas entre 4 y 6, inclusive.
- Notas con valor igual o superior a 9, pero que no son matrícula de honor. Cree una vista para las notas que son matrícula de honor.

Cree un script SQL que genere las tablas del sistema de gestión de tutorías y les añada datos, los modifique, los borre, y los consulte. Debe incluir:

- Al menos tres filas insertadas en cada tabla.
- La actualización de varios atributos en una fila concreta.
- La actualización de un solo atributo en varias filas a la vez.
- El borrado de una fila concreta.
- El borrado de varias filas a la vez (sin llegar a eliminar todas las de la tabla).

Para realizar el ejercicio, cree con su usuario de GitHub una copia del repositorio correspondiente a esta sesión. Realice las modificaciones pertinentes en los archivos suministrados, creando nuevos archivos si es necesario, y suba esos cambios a su copia en GitHub. Recuerde establecer la privacidad de su repositorio como “Private” y dar acceso como colaborador al usuario **iissi**.

