

Sistemas de Información II – Práctica III: Consultas en SQL

Carlos Castillo – carlos.castillo@upf.edu

1. Uso de su propia base de datos

Es importante que cada uno use su propia base de datos, para ello, desde **sauron** o **saruman** hay que conectarse de esta manera al mysql en **atari**:

```
mysql si2_XX -h atari.aules.upf.edu -u si2_XX -p
```

Lo primero es verificar que los datos de la práctica anterior sigan ahí:

```
mysql> SELECT * FROM comunitat;
```

El objetivo de esta práctica es aprender más comandos SQL relacionados con seleccionar datos. Además se incluyen algunas actividades menos guiadas que en la práctica anterior, la idea es intentar inferir cómo hacer algunas operaciones, siguiendo la lógica de los comandos SQL que hemos visto hasta ahora.

2. SQL IN

La instrucción SQL IN corresponde a la relación matemática de pertenencia \in . Ejemplo:

```
mysql> SELECT m_id,nom from municipi WHERE nom IN ('Biel','Zuera');
+-----+-----+
| m_id | nom   |
+-----+-----+
| 1499 | Zuera |
| 1500 | Biel  |
+-----+-----+
```

Lo interesante es que lo que se encuentra dentro de la cláusula IN (...) puede ser a su vez una consulta SELECT, ejemplo:

```
SELECT DISTINCT nom
FROM comunitat
WHERE ca_id IN
(
  SELECT ca_id
  FROM municipi
  WHERE nom LIKE 'Santa Mar_a del %'
);
```

Como sabemos, los espacios y saltos de línea no son relevantes.

Actividad: para cada uno de los siguientes requerimientos, escribir una consulta que usando **IN**, entregue:

- Todas las comunidades con un municipio de menos de 5.000 habitantes.

- Todas las comunidades con un municipio que tenga superficie entre 1.000 y 3.000.
- Todas las comunidades con un municipio que termine en la letra z.
- Reescriba también esta última consulta sin usar IN, el resultado debe ser el mismo.

3. SQL HAVING

La instrucción SQL HAVING se usa para especificar una condición, y por tanto funciona igual que WHERE, pero se utiliza después de haber ejecutado una función de agregación, ejemplo:

```
mysql> SELECT comunitat.nom, SUM(municipio.poblacio2003)
FROM municipio,comunitat
WHERE comunitat.ca_id=municipio.ca_id
GROUP BY comunitat.nom
HAVING SUM(municipio.poblacio2003) > 5000000;
```

nom	SUM(municipio.poblacio2003)
Andalucía	7606848
Catalunya	6704146
Comunidad de Madrid	5718942

Actividad: escribir una consulta que, usando **HAVING**, entregue:

- El nombre de la comunidad y el número de municipios de las comunidades con más de 1000 municipios (es sólo uno).
- El nombre de la comunidad y la superficie promedio de sus municipios de las comunidades con entre 500 y 1000 municipios.
- El nombre de la comunidad y la densidad de población de las comunidades con más de 1000 habitantes por unidad de superficie en promedio (son sólo dos, y no son Madrid y Catalunya).

4. SQL UPDATE

Para actualizar datos en una tabla, se utiliza la instrucción UPDATE. La sintaxis es la siguiente:

```
UPDATE nombretabla
SET columna=valor
WHERE condición
```

Primero, vamos a crear una tabla comunitat2 para no dañar los datos originales.

Actividad: crear una tabla comunitat2. La operación SHOW COLUMNS FROM comunitat2 debe entregar un resultado idéntico a SHOW COLUMNS FROM comunitat.

Ahora, para copiar datos de una tabla a otra, se usa:

```
INSERT INTO comunitat2 (ca_id,nom)
(
SELECT ca_id,nom FROM comunitat
);
```

En cualquier caso puede ser útil saber que para borrar una tabla se usa:

```
DROP TABLE comunitat2;
```

Y para **vaciar** una tabla (borrar los datos sin borrar el esquema de la tabla), se usa:

```
TRUNCATE comunitat2;
```

Una vez que tenemos una tabla `comunitat2` con los mismos datos que `comunitat`, podemos actualizar datos.

Actividad: crear una tabla `municipi2` con los mismos datos que `municipi`, a continuación, realizar las siguientes actualizaciones (en las copias, no en los originales):

- Poner la población1991 en 0 para todos los municipios que empiecen con la letra “a”.
- Poner la superficie a la mitad en todos los municipios que tengan más de 1000 habitantes.

Actividad: crear una tabla `municipi_cat` que tenga todos los datos de los municipios de Catalunya.

5. SQL DELETE

Por último, para borrar filas de una tabla, se utiliza `DELETE`. Esta instrucción funciona igual que `UPDATE` y es incluso más sencilla.

```
DELETE FROM tabla  
WHERE condición
```

Actividad: escribir una consulta que haga lo siguiente (en la tabla `municipi2`):

- Borrar un municipio específico de la tabla (cualquiera, pero que borre sólo uno).
- Borrar los municipios que no tengan habitantes en el 2003.

6. Actividad difícil

Esta actividad es guardar un dato derivado en la tabla `comunitat2`.

Actividad: agregar a la tabla `comunitat2` una columna con población y una columna con superficie, y llenar esa columna con los datos reales obtenidos de sumar la superficie y población total de los municipios. Se puede hacer con 4 instrucciones:

- Una instrucción para agregar la columna de población.
- Una instrucción para agregar la columna de superficie.
- Una operación `UPDATE` que se combina con un `SELECT` para copiar el dato de población total.
- Una operación `UPDATE` que se combina con un `SELECT` para copiar el dato de superficie total.

Hint: en el `SELECT` sólo debe mencionar una tabla en la cláusula `FROM`, y debe incluir una cláusula `WHERE` para que correspondan correctamente las tablas `municipi` y `comunitat2`.

Los datos deben extraerse de la tabla `municipi` porque la tabla `municipi2` contiene información falsa después de las actividades de `UPDATE`.

7. RESULTADO

Lo que se debe entregar son todos los comandos SQL que ejecuten todas las actividades de esta sesión (no la salida de `mysql`).

El resultado de esta práctica **debe entregarse hoy Lunes** vía Campus Global.