Práctica 2. Recuperación de datos sobre múltiples tablas. Uso de funciones en SQL

Índice

- Consultas multi-tabla
- Funciones específicas de SQL
- Consultas con cadenas
- Agrupación de resultados

Índice

- **Consultas multi-tabla**
- Funciones específicas de SQL
- Consultas con cadenas
- Agrupación de resultados

Consultas multi-tabla

- Vamos a complicar un poco las consultas SELECT vistas en la práctica anterior **aprendiendo a realizar consultas en varias tablas** de la base de datos al mismo tiempo
- Es habitual que queramos **acceder a datos que se encuentran en múltiples tablas** y mostrar información mezclada de todas ellas como resultado de una consulta: esto requiere una combinación de columnas de tablas diferentes
- **Solución:** especificando más de una tabla en la cláusula FROM de la instrucción SELECT

Consultas multi-tabla

- SELECT * FROM votantes;
  - Devuelve un total de 17 filas o registros
- SELECT * FROM localidades;
  - Devuelve un total de 16 filas o registros

- ¿Cómo podría obtenerse información de los votantes y las localidades a las que pertenece cada votante?

  - SELECT * FROM votantes, localidades;
    - Devuelve 272 filas o registros!!! Cada DNI aparece 16 veces, una por cada localidad
    - Se está realizando el producto cartesiano 17x16=272

Consultas multi-tabla

- Mostrar información de varias tablas mediante la **unión de filas de las tablas**
- Las columnas que se van a unir entre las dos tablas sean las mismas y contengan los mismos tipos de datos (**uso de una clave externa**):
  - SELECT * FROM votantes, localidades
    WHERE votantes.localidad=localidades.idlocalidad;
- Usando operaciones de unión JOIN (simplifica la lectura):
  - SELECT * FROM votantes JOIN localidades
    ON votantes.localidad=localidades.idlocalidad;

Consultas multi-tabla

- Uso de WHERE en consultas multi-tabla:
  - Usando JOIN:
    - SELECT * FROM votantes JOIN localidades ON votantes.localidad=localidades.idlocalidad WHERE votantes.localidad=1;
  - Sin usar JOIN:
    - SELECT * FROM votantes, localidades WHERE votantes.localidad=localidades.idlocalidad AND votantes.localidad=1;

Consultas multi-tabla

- Uso de alias en consultas multi-tabla:
  - Sin usar alias:
    - SELECT * FROM votantes JOIN localidades ON votantes.localidad=localidades.idlocalidad WHERE votantes.localidad=1;
    - SELECT * FROM votantes, localidades WHERE votantes.localidad=localidades.idlocalidad AND votantes.localidad=1;
  - Usando alias:
    - SELECT * FROM votantes a JOIN localidades b ON a.localidad=b.idlocalidad WHERE a.localidad=1;
    - SELECT * FROM votantes a, localidades b WHERE a.localidad=b.idlocalidad AND a.localidad=1;

UNIVERSIDAD D CÓRDOBA

Consultas multi-tabla

- Para trabajar sobre **más de dos tablas** hay que seguir la misma filosofía, pero teniendo en cuenta que las tres tablas deben estar unidas de alguna manera (**uso de claves externas**):
  - Sin usar JOIN
    - SELECT * FROM votantes a, localidades b, provincias c
      WHERE a.localidad=b.idlocalidad AND b.provincia=c.idprovincia AND a.localidad=1;
  - Usando JOIN
    - SELECT * FROM votantes a JOIN localidades b ON a.localidad=b.idlocalidad JOIN provincias c ON b.provincia=c.idprovincia WHERE a.localidad=1;

Índice

- Consultas multi-tabla
- **Funciones específicas de SQL**
- Consultas con cadenas
- Agrupación de resultados

Funciones específicas de SQL

- **AVG(col_name):** proporciona el valor medio de una columna numérica
- **COUNT(col_name):** proporciona el número de filas que cumplen un criterio
- **MAX(col_name):** devuelve el valor máximo de la columna
- **MIN(col_name):** devuelve el valor mínimo de la columna
- **SUM(col_name):** devuelve la suma de los valores númericos de la columna numérica

¡¡¡Estas funciones no pueden usarse en la clausula WHERE por lo que se ha añadido una nueva clausula **HAVING!!!**

Funciones específicas de SQL

- **AVG(col_name):** proporciona el valor medio de una columna numérica
  - SELECT AVG(presupuesto) FROM partidos;

- **COUNT(col_name):** proporciona el número de filas que cumplen un criterio
  - SELECT COUNT(presupuesto) FROM partidos WHERE presupuesto>100000;

  Importante observa la diferencia respecto a:

  SELECT presupuesto FROM partidos WHERE presupuesto>100000;

Funciones específicas de SQL

- **MAX(col_name):** devuelve el valor máximo de la columna
  - SELECT MAX(presupuesto) FROM partidos;

- **MIN(col_name):** devuelve el valor mínimo de la columna
  - SELECT MIN(presupuesto) FROM partidos;

- **SUM(col_name):** devuelve la suma de los valores númericos de la columna numérica
  - SELECT SUM(presupuesto) FROM partidos;

Índice

- Consultas multi-tabla
- Funciones específicas de SQL
- **Consultas con cadenas**
- Agrupación de resultados

Consultas con cadenas

- Se utiliza el operador LIKE en la clausula WHERE para especificar un patrón en una columna
- Dos comodines:
  - % indica cero o varios caracteres
  - _ indica un sólo carácter

  - WHERE nombrecompleto LIKE ‘a%’ Cualquier nombre que empiece por “a”
  - WHERE nombrecompleto LIKE ‘%a’ Cualquier nombre que acabe en “a”
  - WHERE nombrecompleto LIKE ‘%a%’ Cualquier nombre que contenga “a”
  - WHERE nombrecompleto LIKE ‘_a%’ Cualquier nombre que cuya segunda lentra sea “a”

Índice

- Consultas multi-tabla
- Funciones específicas de SQL
- Consultas con cadenas
- **Agrupación de resultados**

Agrupación de resultados

- Permite **agrupar resultados** dados por funciones de agregación tales como COUNT, MAX, MIN, SUM y AVG.
  - SELECT provincia, SUM(numerohabitantes) FROM localidades GROUP BY provincia;
    - En la tabla localidades, cada localidad pertenece a una provincia y, además, cada localidad tiene su propio numero de habitantes. Si queremos saber el numero total de habitantes de las localidades de cada provincia que hay ahora mismo con datos y agrupados por provincias, tenemos que indicar que la agrupación se hará en base a la provincia.