Lab 1 - Consultas básicas y Funciones SQL

David Moreno Lumbreras & Daniela Patricia Feversani
GSyC, ETSIT. URJC.

Laboratorio de Bases de Datos (BBDD)

Curso 24-25

Contenidos

L1.1 Origen de SQL

Cronología

- E.F Codd de IBM publica, en 1970, un artículo sobre el modelo relacional de datos.
- D.Chamberlin de IBM crea SEQUEL(Structured English QUery Language), un lenguaje para interactuar con bases de datos.
- Posteriormente evolucionó hasta SQL: Structured Query Language.
- Primera versión estándar de SQL (ANSI): SQL-86 (1986). Provoca la **interoperabilidad, portabilidad, crecimiento y difusión** del SQL.
- Posteriormente otras versiones, hasta la más reciente SQL:2016.

Partes de SQL

- Incluye un DDL (Data Definition Language).
  - `CREATE`, `DROP`, `TRUNCATE`...
- Incluye un DML (Data Manipulation Language).
  - `SELECT`, `FROM`, `WHERE`, `JOIN`, `GROUP BY`, `ORDER BY`, `LIMIT`...
- Elementos adicionales:
  - Comandos para especificar restricciones de integridad de datos (e.g. cláves foráneas). `CHECK`, `FOREIGN KEY`, `UNIQUE`
  - Posibilidad de definir *vistas*. `CREATE VIEW`
  - Posibilidad de definir los límites de una transacción (operaciones realizadas atómicamente) `BEGIN TRANSACTION`, `COMMIT`, `ROLLBACK`.
  - Integración con otros lenguajes de programación (e.g. C, C++ o Java).
  - El DDL incluye comandos para especificar privilegios de acceso de usuarios a relaciones y vistas, crear y borrar usuarios, etc.

Tipos de datos

- El estándar SQL define una serie de tipos de datos básicos.
  - **char**(n): String de longitud fija.
  - **varchar**(n): String de longitud variable.
  - **int**: Conjunto finito de enteros (dependiente de la máquina).
  - **smallint**: Conjunto de enteros de menor rango.
  - **numeric**(p, d): Números con decimales (*fixed-point*), especifican número exacto de dígitos y dígitos decimales.
  - **real**, **double precision**: Números en coma flotante, con diferente grado de precisión (dependiente de la máquina).
  - **float**(n): Número en coma flotante, con al menos *n* dígitos de precisión.
- Más adelante veremos cómo usarlos en la definición y alteración de tablas en PostgreSQL.
- Por supuesto, cada base de datos define muchos tipos adicionales, no estándares, para aplicaciones específicas (e.g. `cidr`, `inet`, `macaddr`).

Funciones y operadores

- Contamos con numerosas funciones que permiten realizar todo tipo de operaciones sobre los valores de atributos incluidos en las cláusulas de nuestras consultas (`SELECT`, `WHERE`, `GROUP BY`, `ORDER BY`, etc.).
- También existen operadores lógicos y de comparación (`AND`, `OR`, `BETWEEN`...).
  - Operadores lógicos.
  - Funciones y operadores de comparación.
  - Funciones y operadores matemáticos.
  - Funciones y operadores para *strings*.
  - Búsqueda de patrones (en strings).
- Por último, tenemos funciones para tipos especiales de datos.
  - Fecha y hora, enumerados, funciones y operadores geométricos, para direcciones de red, para JSON, etc.

L1.2 Consultas básicas

Consulta básica, una tabla

- Se recuperan datos de una sola tabla, no hay filtros ni condiciones adicionales (agrupamiento, orden, etc.).

```
SELECT attr1, attr2, ..., attrN
FROM table_name;
```

- Un * en SELECT indica todos los atributos de una tabla.

```
SELECT * FROM item;
```

- Se admiten operaciones básicas sobre los valores de columnas (usualmente numéricos).

```
SELECT item_id, description, sell_price, sell_price * 0.9 AS dc_price
FROM item;
```

Consulta básica, renombrado

- Se pueden establecer nuevos alias para el nombre de atributos (columnas) o tablas en la consulta, usando la cláusula **AS**. Suele ser habitual en:
  - Consultas largas o con nombres de tablas/atributos largos.
  - Cuando calculamos operaciones (funciones) sobre las columnas, para renombrar la columna generada. Especialmente útil cuando queremos reutilizar esa consulta dentro de otra y referirnos a esa columna.

```
SELECT attr1, attr2, ..., attrN AS new_name
FROM table1 t1;
```

```
SELECT t1.attr1, t1.attr2, t2.attr1
FROM table1 AS t1, table2 AS t2 ON t1.attr1 = t2.attr2;
```

```
SELECT i.description, i.sell_price - i.cost_price as benefits
FROM item AS i
LIMIT 5 --Sirve para limitar el número de resultados a mostrar
```

L1.3 Operadores Lógicos y Comparación

Operadores lógicos

- Se emplean para realizar filtros en las filas recuperadas, reteniendo solo las que cumpla determinadas condiciones (impuestas como expresiones booleanas: **AND**, **OR**, **NOT**).

```
SELECT attr1, attr2, ..., attrN
FROM table
WHERE condition;
```

```
SELECT lname, fname
FROM customer
WHERE town = 'Welltown';
```

```
SELECT description, cost_price
FROM item
WHERE cost_price >=5 AND cost_price <= 10;
```

- https://www.postgresql.org/docs/10/functions-logical.html
- Se pueden mezclar las condiciones en la cláusula **WHERE**, pero es conveniente usar paréntesis para garantizar que la precedencia de evaluación es la esperada.

Operadores y Predicados de comparación

- Los operadores de comparación en SQL, como `=`, `<`, y `>`, se utilizan para comparar valores y establecer condiciones en las consultas de bases de datos. Ayudan a filtrar y relacionar datos según criterios específicos.

```
SELECT *
FROM customer
WHERE zipcode = 'BG4 2WE'
```

- Los predicados de comparación en SQL, como `BETWEEN`, `IS DISTINCT FROM` y `IS NULL`, permiten realizar comparaciones más complejas y condicionales en las consultas de bases de datos, ampliando las opciones para filtrar y recuperar datos
- Tabla 9.1 y Tabla 9.2 en https://www.postgresql.org/docs/10/functions-comparison.html.

```
SELECT item_id, description
FROM item
WHERE cost_price BETWEEN 4.5 AND 12.3
```

```
SELECT *
FROM orderinfo
WHERE customer_id IS DISTINCT FROM 8
```

L1.4 Operadores y Funciones para strings

Funciones con strings

- Funciones con Strings:
  - **upper**(s): A mayúsculas.
  - **lower**(s): A minúsculas.
  - **trim**(s): Elimina espacios en blanco (comienzo y final).

```
SELECT LOWER(TRIM(zipcode))
FROM customer
```

```
SELECT fname, lname
FROM customer
WHERE UPPER(fname) LIKE '%A%';
```

Operaciones con strings

- Operaciones con Strings:
  - **Búsqueda de patrones**: % → cualquier substring; _ → cualquier carácter.
  - Para la comparación de patrones se emplea el operador `LIKE`.

```
SELECT fname, lname
FROM customer
WHERE lname LIKE '%es';  -- Apellidos que terminen en "es".
```

```
SELECT fname, lname
FROM customer
WHERE lname LIKE '__tt___'; -- Dos chars, dos 't', tres chars.
```

L1.6 Funciones y operadores matemáticos

Funciones de agregación básicas

- Existen varias funciones básicas de agregación:
  - Promedio: `AVG` (solo val. numéricos).
  - Mínimo: `MIN`.
  - Máximo: `MAX`.
  - Suma: `SUM` (solo val. numéricos).
  - Conteo: `COUNT`.
  - Discriminar valores distintos: `DISTINCT`.

```sql
SELECT MAX(cost_price)
FROM item
WHERE sell_price > 10.5;

SELECT COUNT(DISTINCT town)
FROM customer;
```

Operadores matemáticos

- Listado de operadores matemáticos más completo que en otras bases de datos relacionales.
  - Tabla 9.4 en https://www.postgresql.org/docs/10/functions-math.html.
- A destacar la disponibilidad de operadores unarios para:
  - Cálculo de potencias: `^`
  - Raíces cuadradas y cúbicas: `|/` y `||/`
  - Valor absoluto: `@`

```
SELECT description, @(sell_price - cost_price) AS Diff_price
FROM item
```

Funciones matemáticas

- Listado de funciones matemáticas.
  - https://www.postgresql.org/docs/10/functions-math.html
  - Funciones básicas: Tabla 9.5.
  - Funciones aleatorias (básicas): Tabla 9.6.
  - Funciones trigonométricas: Tabla 9.7.
- Incluye la constante π, distintos logaritmos, potencias, raíces cuadradas y cúbicas...
- A destacar las funciones `width_bucket(...)`, para cálculos relacionados con la construcción de histogramas, en análisis exploratorio de datos.

```
SELECT description, cost_price,
WIDTH_BUCKET(cost_price, 0, 20, 3) AS GamaCalidad
FROM item;
```

- **Cuidado**: Las funciones aleatorias no tienen propiedades robustas como para satisfacer los estándares criptográficos. Para aplicaciones de seguridad mejor usar el módulo **pgcrypto**.

L1.7 Funciones para fecha y hora

Funciones de fecha y hora

- Formato estándar de fecha y hora: ISO 8601.
- Funciones de formateado de fecha y hora:
  - https://www.postgresql.org/docs/10/functions-formatting.html.
- Funciones de fecha y hora:
  - https://www.postgresql.org/docs/10/functions-datetime.html.
- La función para extraer partes de un dato de fecha/hora es `EXTRACT(field FROM timestamp)` o `EXTRACT(field FROM interval)`.
- La función `AGE(timestamp1, timestamp2)` calcula automáticamente la edad de una persona entre la fecha considerada (primer argumento) y su fecha de nacimiento (segundo argumento).

```
SELECT orderinfo_id, customer_id,
EXTRACT(YEAR FROM date_placed) AS year_placed,
EXTRACT(MONTH FROM date_placed) AS month_placed
FROM orderinfo;

SELECT AGE(date_shipped, date_placed) AS Tiempo_Entrega
FROM orderinfo
```

L1.8 Ordenación y Agrupamiento

Ordenación de resultados

- Se puede agregar la cláusula `ORDER BY ... [DESC]`, para ordenar los resultados en función del valor de una columna (`DESC` es opcional).
- Las columnas que usamos para ordenar deben aparecer en el `SELECT` o en la cláusula `GROUP BY`.
- Por defecto la ordenación es ascendente (numérico: menor a mayor; string: orden alfabético). Añadiendo la cláusula `DESC` se invierte el orden de los resultados.

```
SELECT description, cost_price * 1.1 AS IncrementoPrecio
FROM item
ORDER BY IncrementoPrecio DESC;
```

Agrupamientos y agregaciones

- La cláusula `GROUP BY` permite agrupar resultados en función de los valores de una o varias columnas.
- Podemos calcular operaciones básicas de agregación por cada uno de los grupos que se han formado.

```
SELECT item_id, COUNT(quantity)
FROM orderline
GROUP BY item_id

SELECT item_id, COUNT(DISTINCT(quantity))
FROM orderline
GROUP BY item_id
```

Agrupamientos y agregaciones: filtrado

- La cláusula **HAVING** permite filtrar los resultados de una operación de agrupamiento, imponiendo condiciones sobre los valores calculados por grupo.

```
SELECT item_id, COUNT(quantity)
FROM orderline
GROUP BY item_id
HAVING item_id >=4;
```

L1.9 Funciones para tipos de datos específicos

Direcciones de red

- Funciones para direcciones de red: disponibles para los tipos de datos `cidr` e `inet`.
  - https://www.postgresql.org/docs/10/functions-net.html.
- Destacan varias funciones que automatizan cálculos habituales al trabajar con estos tipos de datos:
  - Identificación de versión del protocolo IP de la dirección.
  - Construir máscara de red.
  - Cálculo automático de subredes (mínima subred que incluye a dos rangos de red dados).
  - Extraer dirección de host, de red, *broadcast*...

Bibliografía I


[Silberschatz et al., 2019] Silberschatz, A., Korth, Henry F. Sudarshan, S.
Database System Concepts.
McGraw-Hill, 7th Ed. 2019.


[Connolly & Begg, 2015] Connolly, T., Begg, C.
Database Systems.
Pearson, 6th Global Edition. 2015.