University of Udine

Data Management for Big Data

SQL

Andrea Brunello
andrea.brunello@uniud.it

Introduction

SQL (Structured Query Language) is the (largely) most popular language used to interact with relational databases, supporting querying, definition, and data updating

- *Data Definition Language* (DDL)
  - Definition of the database schema
  - Integrity constraints
  - Views
- *Data Manipulation Language* (DML)
  - Inserts, updates, deletions of data
  - Queries
- Other stuff:
  - Users management, authorizations and permissions
  - Transactions and management of concurrency
  - Stored procedures and triggers
  - Embedding in OO or procedural languages
- Declarative flavour

History

The current SQL is the development of a query language for the relational DBMS **System R**, originally proposed at the IBM research laboratories in the latter half of the 1970s (SEQUEL - Structured English QUEry Language).

Since the 1980s, it has been subject to intense **standardization** activities, mainly within the ANSI (American National Standards Institute) and ISO (the international body coordinating national bodies). The standardization process is still ongoing.

Advantages of standardization: **portability** and **uniformity** in accessing different databases… though SQL implementations offered by different DBMS vendors are slightly different from one another, at least for what concerns advanced queries.

Evolution of the SQL Standard - 1

The main stages of the SQL standardization process can be summarized as follows:

(1) SQL-86 (Basic SQL), produced in 1986 by ANSI, contains all the basic constructs of the query language, but offers limited support for data definition and updating;

(2) SQL-89 (Basic SQL), a limited extension of SQL-86, issued in 1989, introduces the concept of referential integrity (foreign keys);

(3) SQL-92 (**SQL-2**), published in 1992, largely compatible with SQL-89, introduces many new features;

(4) SQL:1999 (**SQL-3**), fully compatible with SQL-2, introduces object extensions, various new constructs, and new services such as triggers and recursive views;

Evolution of the SQL Standard - 2

(5) SQL:2003, SQL:2006, and SQL:2011 introduce further object extensions, remove some unused constructs from previous standards, and include new parts such as SQL/JRT (integration of SQL with the Java language) and SQL/XML (XML data management);

(6) SQL:2016, SQL:2019, and SQL:2023 adds support and operations useful to deal with more complex data formats, like property graph data or JSON for (multidimensional) arrays.

Unlike SQL-2, SQL-3 is organized into several appropriately numbered and named parts (e.g., part 13 is related to SQL/JRT, part 14 to SQL/XML). Each part is produced by a specific technical committee and can be updated independently of the others.

Standard: Core and Extensions

Since SQL:1999, the standard is divided into a **core** and a set of **specialized extensions**.

All SQL-compliant DBMSs should provide the core; the extensions can be implemented as additional modules for specific applications, such as:

- data mining
- geographical and/or temporal data
- data warehousing
- OLAP (OnLine Analytical Processing)
- multimedia data
- ...

DBMS and SQL Standard

The various available DBMSs show small differences in the implementation of the SQL language, especially regarding the more innovative features.

Uniformity is much greater regarding the more established functionalities.

Many systems provide **user-friendly interfaces** for querying, defining, and updating data (for example, visual interfaces).

However, all allow for the **explicit** (and complete) use of the SQL language, which is something that has to be known by the advanced users of the system.

DDL and DML

- SQL is a **declarative** language composed of two parts:
  - DDL (Data Definition Language): that allows to specify the schema of the database (i.e., tables, constraints, ...)
  - DML (Data Manipulation Language): that allows to interact with the content of the database (i.e., the database instance) by means of *queries*

- SQL features a very intuitive syntax

The fundamental block of an SQL query is composed of **three clauses**

SELECT : list of attributes, aggregation functions, expressions
FROM : data sources, tables
WHERE : filtering conditions for the data that has to be extracted

Other clauses: *ORDER BY*, *HAVING*, *GROUP BY*

Let us consider the following two tables

Table "Employee"

	ssn [PK] text	first_name text	last_name text	gender text	birth_date date	salary numeric	department text
1	386-64-8608	Lorrie	Gillaspy	Female	1968-04-04	68490	sales
2	258-77-4074	Lynde	Kitchenside	Female	1987-03-16	76273	market
3	682-79-1556	Johnny	Binder	Male	1969-02-28	43672	[null]
4	622-17-9461	Mark	Robinson	Male	1960-08-17	84741	[null]
5	205-35-1353	John	Smith	Male	1984-02-20	70825	sales
6	678-24-1113	Veronique	Pauleit	Female	1987-01-28	58604	prod
7	210-35-1000	Frank	Doyle	Male	1994-02-20	70825	c_service

Table "Department"

	code [PK] text	name text	budget numeric	address text
1	prod	Product development	513086	68152 Schmedeman Junction
2	market	Marketing	715091	763 Shoshone Avenue
3	sales	Sales	748657	25370 Tennyson Drive
4	c_service	Customer service	780377	61 Dorton Park
5	res	Research and development	981007	57 Cucumber Street

Extract the name, surname, and monthly salary all employees of gender female and with a total salary greater than 60000

SELECT first_name, last_name, salary/12
FROM employee
WHERE gender = 'Female' **AND** salary > 60000;

	ssn [PK] text	first_name text	last_name text	gender text	birth_date date	salary numeric	department text
1	386-64-8608	Lorrie	Gillaspy	Female	1968-04-04	68490	sales
2	258-77-4074	Lynde	Kitchenside	Female	1987-03-16	76273	market
3	682-79-1556	Johnny	Binder	Male	1969-02-28	43672	[null]
4	622-17-9461	Mark	Robinson	Male	1960-08-17	84741	[null]
5	205-35-1353	John	Smith	Male	1984-02-20	70825	sales
6	678-24-1113	Veronique	Pauleit	Female	1987-01-28	58604	prod
7	210-35-1000	Frank	Doyle	Male	1994-02-20	70825	c_service

	first_name text	last_name text	round numeric
1	Lorrie	Gillaspy	5707.50
2	Lynde	Kitchenside	6356.08

Extract the average salary of employees working in the sales department

```
SELECT AVG(salary)
FROM employee
WHERE department = 'sales';
```

	ssn [PK] text	first_name text	last_name text	gender text	birth_date date	salary numeric	department text
1	386-64-8608	Lorrie	Gillaspy	Female	1968-04-04	68490	sales
2	258-77-4074	Lynde	Kitchenside	Female	1987-03-16	76273	market
3	682-79-1556	Johnny	Binder	Male	1969-02-28	43672	[null]
4	622-17-9461	Mark	Robinson	Male	1960-08-17	84741	[null]
5	205-35-1353	John	Smith	Male	1984-02-20	70825	sales
6	678-24-1113	Veronique	Pauleit	Female	1987-01-28	58604	prod
7	210-35-1000	Frank	Doyle	Male	1994-02-20	70825	c_service

	avg numeric
1	69657.500000000000

Extract the name and surname of employees working at a department with a budget greater than 720000

```
SELECT employee.first_name, employee.last_name
FROM employee
        JOIN department ON (employee.department
                            = department.code)
WHERE department.budget > 720000;
```

Each row of employee is combined with every other row of department; then, only the resulting rows that satisfy the JOIN condition are kept

	ssn text	first_name text	last_name text	gender text	birth_date date	salary numeric	department text	code text	name text	budget numeric	address text
1	205-35-1353	John	Smith	Male	1984-02-20	70825	sales	prod	Product development	513086	68152 Schmedeman Junction
2	205-35-1353	John	Smith	Male	1984-02-20	70825	sales	market	Marketing	715091	763 Shoshone Avenue
3	205-35-1353	John	Smith	Male	1984-02-20	70825	sales	sales	Sales	748657	25370 Tennyson Drive
4	205-35-1353	John	Smith	Male	1984-02-20	70825	sales	c_service	Customer service	780377	61 Dorton Park
5	205-35-1353	John	Smith	Male	1984-02-20	70825	sales	res	Research and development	981007	57 Cucumber Street
6	210-35-1000	Frank	Doyle	Male	1994-02-20	70825	c_service	prod	Product development	513086	68152 Schmedeman Junction
7	210-35-1000	Frank	Doyle	Male	1994-02-20	70825	c_service	market	Marketing	715091	763 Shoshone Avenue
8	210-35-1000	Frank	Doyle	Male	1994-02-20	70825	c_service	sales	Sales	748657	25370 Tennyson Drive
9	210-35-1000	Frank	Doyle	Male	1994-02-20	70825	c_service	c_service	Customer service	780377	61 Dorton Park
10	210-35-1000	Frank	Doyle	Male	1994-02-20	70825	c_service	res	Research and development	981007	57 Cucumber Street
11	258-77-4074	Lynde	Kitchenside	Female	1987-03-16	76273	market	prod	Product development	513086	68152 Schmedeman Junction

SQL, Data Manipulation Language
Example 3

At this point, the WHERE condition is applied over the remaining rows

	ssn text	first_name text	last_name text	gender text	birth_date date	salary numeric	department text	code text	name text	budget numeric	address text
1	205-35-1353	John	Smith	Male	1984-02-20	70825	sales	sales	Sales	748657	25370 Tennyson Drive
2	210-35-1000	Frank	Doyle	Male	1994-02-20	70825	c_service	c_service	Customer service	780377	61 Dorton Park
3	258-77-4074	Lynde	Kitchenside	Female	1987-03-16	76273	market	market	Marketing	715091	763 Shoshone Avenue
4	386-64-8608	Lorrie	Gillaspy	Female	1968-04-04	68490	sales	sales	Sales	748657	25370 Tennyson Drive
5	678-24-1113	Veronique	Pauleit	Female	1987-01-28	58604	prod	prod	Product development	513086	68152 Schmedeman Junction

We then obtain the final result, keeping just the attributes specified in the SELECT clause

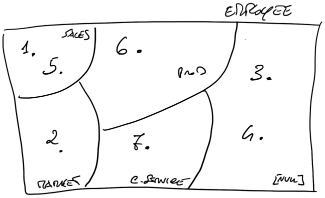
	first_name text	last_name text
1	Lorrie	Gillaspy
2	John	Smith
3	Frank	Doyle

For each department, extract the minimum and maximum salary of its employees

SELECT department, **MIN**(salary), **MAX**(salary)
FROM employee
GROUP BY department;

	ssn [PK] text	first_name text	last_name text	gender text	birth_date date	salary numeric	department text
1	386-64-8608	Lorrie	Gillaspy	Female	1968-04-04	68490	sales
2	258-77-4074	Lynde	Kitchenside	Female	1987-03-16	76273	market
3	682-79-1556	Johnny	Binder	Male	1969-02-28	43672	[null]
4	622-17-9461	Mark	Robinson	Male	1960-08-17	84741	[null]
5	205-35-1353	John	Smith	Male	1984-02-20	70825	sales
6	678-24-1113	Veronique	Pauleit	Female	1987-01-28	58604	prod
7	210-35-1000	Frank	Doyle	Male	1994-02-20	70825	c_service

The table rows are first partitioned according to the value of *department*



For each partition, its name, minimum and maximum salary is returned

	department character varying (20)	min numeric	max numeric
1	[null]	43672	84741
2	prod	58604	58604
3	sales	68490	70825
4	c_service	70825	70825
5	market	76273	76273

References

A. Silberschatz, H.F. Korth, S. Sudarshan *Database system concepts*, 7th Edition, 2020.

PostgreSQL's website: `https://www.postgresql.org/`