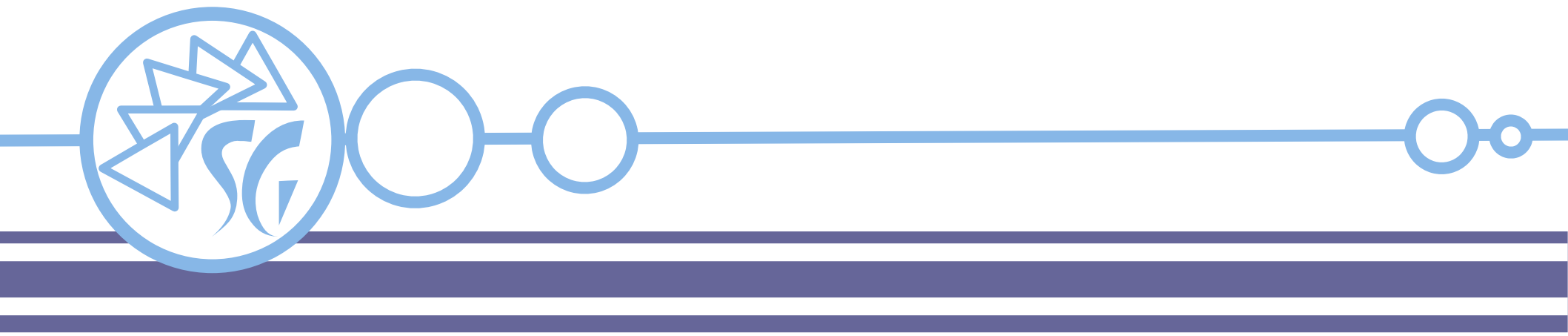

Funzionalità avanzate del linguaggio SQL

Ing. Simone Giustetti

www.giustetti.net

Transazioni

Nell’ambito dei database si chiama **transazione** un’operazione complessa, formata da una sequenza di molte operazioni semplici, coerente, affidabile e indipendente da altre transazioni

Le transazioni sono solitamente utilizzate per modificare i dati salvati in un database

Transazioni

Le transazioni sono state introdotte nei RDBMS con 2 finalità:

- Fornire unità di lavoro tanto affidabili da consentire il ripristino dei dati in caso di crash
- Garantire l'isolamento di diversi programmi o utenti che manipolino i dati in concorrenza, mantenendo nel contempo la consistenza degli stessi dati

Transazioni

Le transazioni devono essere:

- **A**tomiche: L'operazione deve completare nella sua interezza o essere ignorata
- **C**onsistente: Deve sottostare ai vincoli del database
- **I**solata: Non deve influenzare altre transazioni
- **D**urevole: Deve essere scritta su supporto fisico persistente

Transazioni

Ciclo di vita di una transazione

- Avvio della transazione
- Esecuzione di una sequenza di comandi SQL
- Se non ci sono errori la transazione è confermata ed i dati aggiornati di conseguenza
- Se ci sono errori la transazione è annullata riportando i dati allo stato precedente eventuali modifiche

Transazioni

Avvio di una transazione (MariaDB / MySQL)
{ START TRANSACTION | BEGIN }

Conferma di una transazione (MariaDB / MySQL)
COMMIT

Annullare una transazione (MariaDB / MySQL)
ROLLBACK

Impostare il comportamento standard del server
SET autocommit = { 0 | 1 }
Default = 1 (on)
Cambiare l'impostazione comporta il commit automatico delle transazioni attive

Transazioni

```
START TRANSACTION;

SELECT SUM(salary) AS total_expense
FROM salaries
WHERE salary_type = 1;

UPDATE employees
SET birth_date = ‘1973-11-24’
WHERE first_name = ‘simone’
AND last_name = ‘giustetti’;

COMMIT;
```

Transazioni

Le istruzioni DDL generano automaticamente un COMMIT implicito

Le istruzioni DCL generano automaticamente un COMMIT implicito

Le istruzioni di manutenzione generano automaticamente un COMMIT implicito

Le transazioni non possono essere dichiarate nel corpo di un Trigger o altro evento

Transazioni

Avvio di una transazione (Ms SQL SERVER)
BEGIN TRANSACTION [<transaction>]

Conferma di una transazione (Ms SQL SERVER)
COMMIT [<transaction>]

Annullare una transazione (Ms SQL SERVER)
ROLLBACK [<transaction>]

Impostare il comportamento standard del server
SET IMPLICIT_TRANSACTIONS { OFF | ON }
Default = ON

Transazioni

```
BEGIN TRANSACTION;

DELETE FROM salaries
WHERE salary = 66961;

COMMIT;

BEGIN TRANSACTION;

INSERT INTO employees
VALUES( 200000, 'simone', 'giustetti', 'M', '2020-04-15' );
INSERT INTO employees
VALUES( 200001, 'stefano', 'gustinetti', 'M',
        '2020-04-15' );

ROLLBACK;
```

Sotto-query

Sotto-query / query annidate / query ricorsive

Sono query eseguite all'interno di parti di una istruzione SQL **SELECT**

Possono essere usate nelle clausole
- SELECT (Ms SQL Server)
- FROM (MariaDB / MySQL)
- WHERE

Possono penalizzare le prestazioni, ma la maggior parte delle query che contengono sotto-query possono essere riscritte con i join

Sotto-query

Le sotto query devono essere racchiuse tra parentesi tonde

È consentito interrogare più colonne in una singola sotto-query. In caso di confronto anche i campi confrontati devono essere racchiusi tra parentesi tonde

Una sotto-query può rendere un numero di righe superiore a 1

Sotto-query

```
SELECT *
FROM employees
WHERE ( last_name, birth_date ) =
    ( SELECT last_name, birth_date
      FROM employees
      WHERE emp_no = 10001
    );
```

```
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello   | M      | 1986-06-26 |
+--------+------------+------------+-----------+--------+------------+
```

Sotto-query

```
SELECT *
FROM employees
WHERE ( last_name, birth_date ) =
    ( SELECT last_name, birth_date
      FROM employees
      WHERE first_name  = 'Georgi'
    );
```

```
ERROR 1242 (21000): Subquery returns more than 1 row
```

Sotto-query

```
SELECT *
FROM employees
WHERE ( last_name, birth_date ) IN
    ( SELECT last_name, birth_date
      FROM employees
      WHERE first_name  = 'Georgi'
    );
```

```
+--------+------------+------------+---------------+--------+------------+
| emp_no | birth_date | first_name | last_name     | gender | hire_date  |
+--------+------------+------------+---------------+--------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello       | M      | 1986-06-26 |
|  10909 | 1954-11-11 | Georgi     | Atchley       | M      | 1985-04-21 |
|  11029 | 1962-07-12 | Georgi     | Itzfeldt      | M      | 1992-12-27 |
|  11430 | 1957-01-23 | Georgi     | Klassen       | M      | 1996-02-27 |
|  12157 | 1960-03-30 | Georgi     | Barinka       | M      | 1985-06-04 |
...
```

Sotto-query

```
SELECT EMP_MASTER.*
FROM employees AS EMP_MASTER,
        employees AS EMP_SLAVE
WHERE EMP_MASTER.last_name = EMP_SLAVE.last_name
AND EMP_MASTER.birth_date = EMP_SLAVE.birth_date
AND EMP_SLAVE.first_name = 'Georgi';
```

```
+--------+------------+------------+-------------+--------+------------+
| emp_no | birth_date | first_name | last_name   | gender | hire_date  |
+--------+------------+------------+-------------+--------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello     | M      | 1986-06-26 |
|  10909 | 1954-11-11 | Georgi     | Atchley     | M      | 1985-04-21 |
|  11029 | 1962-07-12 | Georgi     | Itzfeldt    | M      | 1992-12-27 |
|  11430 | 1957-01-23 | Georgi     | Klassen     | M      | 1996-02-27 |
|  12157 | 1960-03-30 | Georgi     | Barinka     | M      | 1985-06-04 |
|  15220 | 1957-08-03 | Georgi     | Panienski   | F      | 1995-07-23 |
|  15660 | 1956-01-13 | Georgi     | Hartvigsen  | M      | 1994-10-13 |
...
```

Sotto-query

Le sotto-query contenenti funzioni di raggruppamento **non possono** essere riscritte usando i join

```
SELECT ST.last_name, ST.first_name
FROM student AS ST INNER JOIN marks AS MA
WHERE ST.student_id = MA.student_id
AND MA.date = '2020-03-14'
AND MA.score > ( SELECT AVG( score )
                 FROM marks
                 WHERE date = '2020-03-14'
               )
ORDER BY ST.last_name, ST.first_name;
```

Sotto-query

Sotto-query nell'istruzione FROM (MariaDB / MySQL):

```
SELECT AVG( score_sum )
FROM ( SELECT SUM( score ) AS score_sum
         FROM student
         GROUP BY name
       ) AS temp_table;
```

Sotto-query

Sotto-query nell'istruzione SELECT (Ms SQL Server):

```
SELECT sales_order_id, total, (
    SELECT AVG( total )
    FROM sales_order_detail ) AS average_total
FROM sales_order_detail;

SELECT AVG(
        ( SELECT SUM( score )
          FROM student
          GROUP BY name
         )) AS average_score
FROM dual;
```

Informazioni & Licenze