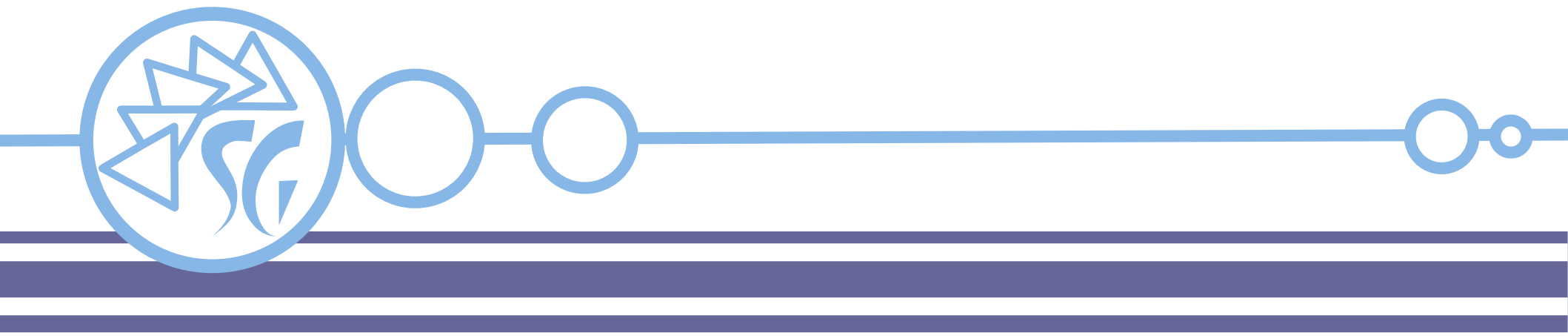

Il Linguaggio SQL
Data Manipulation Language

Ing. Simone Giustetti

www.giustetti.net

DML

È il sottoinsieme di SQL dedicato a gestire i dati

Le istruzioni sono ispirate al linguaggio inglese parlato

```
UPDATE impiegato
SET salario = 2000
WHERE id_impiegato = 511;
```

DML

Non fa differenza tra maiuscole e minuscole

```
Update impiegato
sEt salario = 2000
where id_impiegato = 511;
```

La prassi consiglia di usare il maiuscolo per le parole chiave

Ogni parte di una istruzione incomincia con una parola chiave

Le istruzioni devono essere terminate con “;”

DML

Esistono molti modi per ottenere il medesimo risultato

```
UPDATE impiegato
SET salario = ( salario * 120 ) / 100
WHERE salario <= 1200;

UPDATE impiegato
SET salario = ( salario * 120 ) / 100
WHERE salario NOT > 1200;

UPDATE impiegato
SET salario = ( salario * 120 ) / 100
WHERE salario BETWEEN 0 AND 1200;
```

Commento

Qualsiasi riga che incominci con la sequenza '--' è ignorata dal motore

```
-- This is a comment
```

Ottenere un elenco dei database / schema

- **MariaDB / MySQL**

```
SHOW DATABASES;
```

- **Ms SQL Server**

```
SELECT name
FROM   SYS.DATABASES;
GO

[ EXEC ] sp_databases
GO
```

DML

Ottenere un elenco delle tabelle

- **MariaDB / MySQL**

```
[ CONNECT | USE ] <db>;
SHOW TABLES;
```

- **Ms SQL Server** ≤ 2005

```
SELECT name, crdate
FROM   SYS.SYSOBJECTS
WHERE  xtype = 'U';  -- U = User Table
GO
```

Ms SQL Server ≥ 2005

```
SELECT *
FROM   <db>.INFORMATION_SCHEMA.TABLES
WHERE  TABLE_TYPE = 'BASE TABLE';
GO
```

Ottenere la definizione di una tabella

- **MariaDB / MySQL**

  DESC <table>;

  DESCRIBE <table>;

- **Ms SQL Server**

  [EXEC] sp_columns <table>;

  GO

Selezione di Righe

```
SELECT *
FROM <schema>.<table>;
```

```
SELECT * FROM anagrafica_mansioni;
```

IDAnagraficaMansione	AMANome	AMAAnnotazioni	AMAArchivio
1	PARTITA IVA		0
2	DIPENDENTE		0
3	SOCIO LAVORATORE		0
4	DIRIGENZA/PRESIDENZA		0
5	CONSULENTE		0

Filtrare le Colonne

```
SELECT [ <table>.<column>, ... ]
FROM <schema>.<table>;
```

```
SELECT IDAnagraficaMansione, AMANome
FROM anagrafica_mansioni;
```

```
+----------------------+----------------------+
| IDAnagraficaMansione | AMANome              |
+----------------------+----------------------+
|                    1 | PARTITA IVA          |
|                    2 | DIPENDENTE           |
|                    3 | SOCIO LAVORATORE     |
|                    4 | DIRIGENZA/PRESIDENZA |
|                    5 | CONSULENTE           |
+----------------------+----------------------+
```

Alias

SELECT <table>.<column> [[AS] <alias>], ...
FROM <schema>.<table> [AS <alias>];

SELECT IDAnagraficaMansione id_mansione, AMANome AS NOME
FROM anagrafica_mansioni AS Mansione;

```
+----------------------+----------------------+
| id_mansione          | NOME                 |
+----------------------+----------------------+
|                    1 | PARTITA IVA          |
|                    2 | DIPENDENTE           |
|                    3 | SOCIO LAVORATORE     |
|                    4 | DIRIGENZA/PRESIDENZA |
|                    5 | CONSULENTE           |
+----------------------+----------------------+
```

Campi Calcolati

```
SELECT   Importo,
         ImponibilePostCambio,
         ImpostaArrotondata,
         ImponibilePostCambio + ImpostaArrotondata AS totale_calcolato,
         ( ImponibilePostCambio + ImpostaArrotondata ) / 2 AS totale_dimezzato
FROM tblfatturetctotali;
```

Importo	ImponibilePostCambio	ImpostaArrotondata	totale_calcolato	totale_dimezzato
219800.00	4396.00	879.20	5275.20	2637.600000
219800.00	4396.00	879.20	5275.20	2637.600000

Sopprimere le righe duplicate

```
SELECT DISTINCT [ * ] [ <table>.<column>, ... ]
FROM <schema>.<table>;
```

```
SELECT DISTINCT job
FROM scott.emp;
```

```
+--------------------+
| job                |
+--------------------+
| ANALYST            |
| CLERK              |
| MANAGER            |
| PRESIDENT          |
| SALESMAN           |
+--------------------+
```

Ordinare i dati

```
SELECT [ DISTINCT ] [ * ] [ <table>.<column>, ... ]
FROM <schema>.<table>
ORDER BY [ <table>.<column> [ ASC | DESC ], ... ]
  [ NUMBER [ ASC | DESC ], ... ];

SELECT 'Utente: ', first_name, last_name
FROM employees
ORDER BY first_name, last_name DESC;
```

Filtrare i dati

```
SELECT [ DISTINCT ] [ * ] [ <table>.<column>, ... ]
FROM <schema>.<table>
WHERE <table>.<column> <operator> <value>
ORDER BY [ <table>.<column> [ ASC | DESC ], ... ]
  [ NUMBER [ ASC | DESC ], ... ];
```

```
SELECT 'Utente: ', first_name, last_name
FROM employees
WHERE first_name != 'Aemer';
ORDER BY first_name, last_name DESC;
```

DML

SQL - OPERATORI DI CONFRONTO	
Operatore	**Descrizione**
=	Uguaglianza / Uguale a
<	Minore di
<=	Minore oppure uguale
>	Maggiore di
>=	Maggiore oppure uguale
!=, <>	Diverso da
BETWEEN ... AND	E' compreso tra 2 valori
IN(...,)	E' uno dei valori compresi nell'elenco
IS NULL	E' un valore nullo
IS NOT NULL	E' un valore qualsiasi diverso da nullo

DML

SQL - OPERATORI DI CONFRONTO	
Operatore	**Descrizione**
LIKE	Simile a

SQL - CARATTERI SPECIALI	
Carattere	**Descrizione**
%	Stringa generica
_	Carattere generico

DML

```
SELECT emp_no,
       last_name AS Cognome,
       first_name AS Nome
FROM employees
WHERE   emp_no <= 100050
AND     first_name LIKE '%Smi%'
AND     first_name LIKE '%h%'
ORDER BY first_name;
```

Funzioni

È possibile chiamare funzioni in diverse parti di un'istruzione SQL

La sintassi è analoga ai linguaggi di programmazione di alto livello

<function>(<parameter> [,] ...)

DML

MariaDB

https://mariadb.com/kb/en/built-in-functions/
https://mariadb.com/kb/it/funzioni-e-operatori/
https://www.techonthenet.com/mariadb/functions/index.php

Ms SQL Server

https://docs.microsoft.com/en-us/sql/t-sql/functions/functions?view=sql-server-ver15
https://docs.microsoft.com/it-it/sql/t-sql/functions/functions?view=sql-server-ver15
https://www.techonthenet.com/sql_server/functions/index.php

MySQL

https://dev.mysql.com/doc/refman/8.0/en/functions.html
https://www.techonthenet.com/mysql/functions/index.php

```
SELECT CONCAT( dept_name, ' Diretto da: ', manager )
   AS Reparto
FROM v_full_departments
ORDER BY Reparto;
```

```
+----------------------------------------------+
| Reparto                                      |
+----------------------------------------------+
| Customer Service Diretto da: Yuchang Weedman |
| Development Diretto da: Leon DasSarma        |
| Finance Diretto da: Isamu Legleitner         |
| Human Resources Diretto da: Karsten Sigstam  |
| Marketing Diretto da: Vishwani Minakawa      |
| Production Diretto da: Oscar Ghazalie        |
| Quality Management Diretto da: Dung Pesch    |
| Research Diretto da: Hilary Kambil           |
| Sales Diretto da: Hauke Zhang                |
+----------------------------------------------+
```

```
SELECT CONCAT( UCASE( dept_name ), ' Diretto da: ',
   LOWER( manager )) AS Reparto
FROM v_full_departments
ORDER BY Reparto;
```

```
+-----------------------------------------+
| Reparto                                 |
+-----------------------------------------+
| CUSTOMER SERVICE Diretto da: yuchang weedman |
| DEVELOPMENT Diretto da: leon dassarma   |
| FINANCE Diretto da: isamu legleitner    |
| HUMAN RESOURCES Diretto da: karsten sigstam  |
| MARKETING Diretto da: vishwani minakawa |
| PRODUCTION Diretto da: oscar ghazalie   |
| QUALITY MANAGEMENT Diretto da: dung pesch    |
| RESEARCH Diretto da: hilary kambil      |
| SALES Diretto da: hauke zhang           |
+-----------------------------------------+
```

```
SELECT LPAD(
   LEFT( manager, INSTR( manager, ' ' )), 50, ' ' ) AS Nome,
   RIGHT( manager, INSTR( manager, ' ' )) AS Cognome
FROM v_full_departments
ORDER BY Cognome, Nome;
```

```
+-----------+-----------+
| Nome      | Cognome   |
+-----------+-----------+
| Hilary    |  Kambil   |
| Vishwani  |  Minakawa |
| Karsten   |  Sigstam  |
| Yuchang   |  Weedman  |
| Hauke     |  Zhang    |
| Oscar     | azalie    |
| Isamu     | eitner    |
| Dung      | Pesch     |
| Leon      | Sarma     |
+-----------+-----------+
```

```
SELECT
   RPAD( LEFT( manager, INSTR( manager, ' ' )), 40, ' ' ) AS Nome,
   LPAD( RIGHT( manager, INSTR( manager, ' ' )), 40, ' ' ) AS Cognome
FROM v_full_departments
ORDER BY Cognome, Nome;
```

```
+----------------------------------------+----------------------------------------+
| Nome                                   | Cognome                                |
+----------------------------------------+----------------------------------------+
| Dung                                   |                                  Pesch |
| Leon                                   |                                  Sarma |
| Hauke                                  |                                  Zhang |
| Oscar                                  |                                 azalie |
| Isamu                                  |                                 eitner |
| Hilary                                 |                                 Kambil |
| Karsten                                |                                Sigstam |
| Yuchang                                |                                Weedman |
| Vishwani                               |                               Minakawa |
+----------------------------------------+----------------------------------------+
```

Ms SQL Server non riconosce LPAD e RPAD

Le funzioni possono essere incluse in diverse parti di una istruzione SQL

```
SELECT
   RPAD( LEFT( manager, INSTR( manager, ' ' )), 40, ' ' ) AS Nome,
   LPAD( RIGHT( manager, INSTR( manager, ' ' )), 40, ' ' ) AS Cognome
FROM v_full_departments
ORDER BY RIGHT( manager, INSTR( manager, ' ' )), Nome;

SELECT
   RPAD( LEFT( manager, INSTR( manager, ' ' )), 40, ' ' ) AS Nome,
   LPAD( RIGHT( manager, INSTR( manager, ' ' )), 40, ' ' ) AS Cognome
FROM v_full_departments
ORDER BY TRIM( RIGHT( manager, INSTR( manager, ' ' ))), Nome;
```

Esistono funzioni **per ogni tipologia di dato**

```
SELECT IdFattura, Quantity, FLOOR( Quantity ),
   CEIL( Quantity ), Valuta
FROM tblfatturevcorpo;
```

IdFattura	Quantity	FLOOR(Quantity)	CEIL(Quantity)	Valuta
2001#XXX#00000006	3147.870	3147	3148	USD
2001#XXX#00000007	3128.706	3128	3129	USD
2001#XXX#00000008	3147.025	3147	3148	USD
2001#XXX#00000009	4000.000	4000	4000	USD
2001#XXX#00000010	2099.914	2099	2100	USD
2001#XXX#00000011	2000.000	2000	2000	USD
2001#YYY#00000219	1940.420	1940	1941	USD
2001#YYY#00000218	2003.267	2003	2004	USD
2001#YYY#00000213	2500.000	2500	2500	USD
2001#XXX#00000084	1046.425	1046	1047	USD

DML

```
SELECT IdFattura, Quantity,
     ROUND( Quantity,  1 ) AS P1,
     ROUND( Quantity,  2 ) AS P2,
     ROUND( Quantity,  3 ) AS P3,
     ROUND( Quantity,  4 ) AS P4,
     ROUND( Quantity, -1 ) AS N1,
     ROUND( Quantity, -2 ) AS N2,
     ROUND( Quantity, -3 ) AS N3,
     ROUND( Quantity, -4 ) AS N4
  FROM tblfatturevcorpo;
```

IdFattura	Quantity	P1	P2	P3	P4	N1	N2	N3	N4
2001#XXX#00000006	3147.870	3147.9	3147.87	3147.870	3147.8700	3150	3100	3000	0
2001#XXX#00000007	3128.706	3128.7	3128.71	3128.706	3128.7060	3130	3100	3000	0
2001#XXX#00000008	3147.025	3147.0	3147.03	3147.025	3147.0250	3150	3100	3000	0
2001#XXX#00000009	4000.000	4000.0	4000.00	4000.000	4000.0000	4000	4000	4000	0
2001#XXX#00000010	2099.914	2099.9	2099.91	2099.914	2099.9140	2100	2100	2000	0

I valori **NULL** possono essere gestiti per sostituzione

```
SELECT IdFattura, IFNULL( Descrizione, '--' )
FROM tblfatturevcorpo
WHERE Descrizione IS NULL;
```

```
+--------------------+------------------------------+
| IdFattura          | IFNULL( Descrizione, '--' ) |
+--------------------+------------------------------+
| 2001#   H#00000006 | --                           |
| 2001#   H#00000007 | --                           |
| 2001#   H#00000008 | --                           |
| 2001#   H#00000009 | --                           |
| 2001#   H#00000010 | --                           |
+--------------------+------------------------------+
```

Ms SQL Server riconosce la funzione equivalente **ISNULL**

È possibile tradurre i valori mediante la funzione **CASE**

```
SELECT DISTINCT Anno, NumDoc Numero,
   CASE TipoDoc
      WHEN 'F' THEN 'Fattura'
      WHEN 'T' THEN 'Nota di Credito'
      ELSE '--'  -- Optional
   END Tipo_Documento
FROM tblfatturetesta
ORDER BY Tipo_Documento, Anno, Numero;
```

```
+------+--------+----------------+
| Anno | Numero | Tipo_Documento |
+------+--------+----------------+
| 2001 |      6 | Fattura        |
| 2001 |      7 | Fattura        |
| 2001 |      8 | Fattura        |
| 2001 |      9 | Fattura        |
| 2001 |     10 | Fattura        |
| 2001 |     11 | Fattura        |
...
```

Funzioni di Raggruppamento

Consentono di eseguire operazioni su tutte le righe estratte da una o più tabelle

SQL - FUNZIONI DI RAGGRUPPAMENTO	
Funzione	**Descrizione**
AVG	Calcola il valore medio
COUNT	Conteggio delle righe
MAX	Rende il valore massimo
MIN	Rende il valore minimo
SUM	Somma dei valori

```
SELECT COUNT( * )
FROM salaries;
```

```
+------------+
| count( * ) |
+------------+
|    2844047 |
+------------+
1 row in set (1.63 sec)
```

```
SELECT COUNT( emp_no )
FROM salaries;
```

```
+-----------------+
| count( emp_no ) |
+-----------------+
|         2844047 |
+-----------------+
1 row in set (0.90 sec)
```

```
SELECT AVG( salary )
FROM salaries;
```

```
+---------------+
| AVG( salary ) |
+---------------+
|    63810.7448 |
+---------------+
1 row in set (1.16 sec)
```

```
SELECT SUM( salary )
FROM salaries;
```

```
+---------------+
| SUM( salary ) |
+---------------+
|  181480757419 |
+---------------+
1 row in set (0.90 sec)
```

L'istruzione **GROUP BY** è usata per raggruppare i risultati delle funzioni

```
SELECT Conto_Corrente, COUNT( IdCliente )
FROM sg_clienti_conticorrenti
GROUP BY Conto_Corrente;
```

Conto_Corrente	COUNT(IdCliente)
IT 12 A YYYYY 01400 XXXXXXXXXXX	270
IT 15 B YYYYY 01400 XXXXXXXXXXX	152
IT 22 R YYYYY 01480 XXXXXXXXXXX	118
IT 45 C YYYYY 01401 XXXXXXXXXXX	3
IT 64 M YYYYY 01400 XXXXXXXXXXX	61
IT 67 P YYYYY 01400 XXXXXXXXXXX	515
IT 70 N YYYYY 01400 XXXXXXXXXXX	223
IT 93 S YYYYY 01400 XXXXXXXXXXX	79
IT 95 T YYYYY 01480 XXXXXXXXXXX	26

DML

```
SELECT Conto_Corrente, COUNT( Id_Cliente ) AS Conteggio
FROM sg_clienti_conticorrenti
WHERE Id_Banca IN( 2, 4, 5 )
GROUP BY Conto_Corrente
ORDER BY Conteggio;
```

```
+-----------------------------------+-----------+
| Conto_Corrente                    | Conteggio |
+-----------------------------------+-----------+
| IT 45 C YYYYY 01401 XXXXXXXXXXX   |         3 |
| IT 95 T YYYYY 01480 XXXXXXXXXXX   |        26 |
| IT 22 R YYYYY 01480 XXXXXXXXXXX   |       118 |
| IT 12 A YYYYY 01400 XXXXXXXXXXX   |       270 |
| IT 67 P YYYYY 01400 XXXXXXXXXXX   |       515 |
+-----------------------------------+-----------+
```

È possibile filtrare i valori ottenuti dalle funzioni di raggruppamento

```
SELECT Conto_Corrente, COUNT( Id_Cliente ) AS Conteggio
FROM sg_clienti_conticorrenti
WHERE Id_Banca IN( 2, 4, 5 )
GROUP BY Conto_Corrente
HAVING Conteggio BETWEEN 5 AND 225
ORDER BY Conteggio;
```

```
+-----------------------------------+-----------+
| Conto_ Corrente                   | Conteggio |
+-----------------------------------+-----------+
| IT 95 T YYYYY 01480 XXXXXXXXXXX   |        26 |
| IT 22 R YYYYY 01480 XXXXXXXXXXX   |       118 |
+-----------------------------------+-----------+
```

Chiamare una funzione di raggruppamento dentro un'altra è un errore di sintassi

```
SELECT Conto_Corrente, MAX( COUNT( Id_Cliente )) AS MAX_Conteggio
FROM sg_clienti_conticorrenti
WHERE Id_Banca IN( 2, 4, 5 )
GROUP BY Conto_Corrente
ORDER BY MAX_Conteggio;
```

ERROR 1111 (HY000): Invalid use of group function

```
SELECT Conto_Corrente, MAX( COUNT( Id_Cliente )) AS MAX_Conteggio
FROM tblclienticonticorrenti
WHERE Id_Banca IN( 2, 4, 5 )
ORDER BY MAX_Conteggio;
```

ERROR 1111 (HY000): Invalid use of group function

DML

Prestare sempre attenzione a quali campi inserire nell'istruzione GROU BY

```
SELECT Conto_Corrente, COUNT( Id_Cliente ) AS Conteggio
FROM sg_clienti_conticorrenti
GROUP BY Id_Banca
ORDER BY MAX_Conteggio;
```

```
+-----------------------------------+-----------+
| Conto_Corrente                    | Conteggio |
+-----------------------------------+-----------+
| IT 45 C YYYYY 01401 XXXXXXXXXXX   |         3 |
| IT 22 R YYYYY 01480 XXXXXXXXXXX   |       144 |
| IT 15 B YYYYY 01400 XXXXXXXXXXX   |       213 |
| IT 70 N YYYYY 01400 XXXXXXXXXXX   |       302 |
| IT 67 P YYYYY 01400 XXXXXXXXXXX   |       785 |
+-----------------------------------+-----------+
```

I conteggi sono effettuati per banca non per conto corrente pertanto i risultati sono casuali

DML

È possibile selezionare dati da molte tabelle con un’unica istruzione

Le tabelle devono essere legate usando relazioni o **JOIN**

I JOIN sono gestiti per valore
- Non sono orientati
- Non risentono del problema dei collegamenti morti
- Più lenti, ma più generici di un puntatore

DML

Per creare un JOIN si deve collegare uno o più campi di due tabelle

La tipologia e l'ordine dei dati devono coincidere

Le tabelle collegate generano un prodotto cartesiano delle righe. Le righe sono poi filtrate utilizzando le condizioni di collegamento

Nessuna condizione comporta la lettura di tutte le righe del prodotto cartesiano

INNER JOIN

Collegamento diretto tra 2 tabelle

1)Il motore esegue il prodotto cartesiano
2)Filtra le righe mantenendo solo quelle che rispettano la condizione con cui sono legate le tabelle

Esistono 2 forme di inner join:
- Implicita
- Esplicita

Forma Implicita

Il JOIN è imposto attraverso le condizioni del filtro

```
SELECT DEPT.dept_name, EMP.last_name,
    EMP.first_name, DM.from_date, DM.to_date
FROM dept_manager AS DM, departments AS DEPT,
    employees AS EMP
WHERE DM.dept_no = DEPT.dept_no
    AND  DM.emp_no = EMP.emp_no
ORDER BY DEPT.dept_name, DM.from_date, DM.to_date
```

Forma Esplicita

Il JOIN è imposto attraverso un'istruzione opportuna

```
SELECT DEPT.dept_name, EMP.last_name,
     EMP.first_name, DM.from_date, DM.to_date
FROM dept_manager AS DM
  INNER JOIN departments AS DEPT ON DM.dept_no = DEPT.dept_no
  INNER JOIN employees   AS EMP   ON DM.emp_no = EMP.emp_no
ORDER BY DEPT.dept_name, DM.from_date, DM.to_date
```

DML

Quando si estraggono i dati da molte tabelle bisogna verificare che i nomi delle colonne e gli alias siano univoci

```
SELECT last_name, first_name, title
FROM titles, employees
WHERE titles.emp_no = employees.emp_no
    AND emp_no = 19999;
```

ERROR 1052 (23000): Column 'emp_no' in where clause is ambiguous

LEFT / RIGHT JOIN

È un collegamento orientato tra due tabelle
1) Il motore esegue il prodotto cartesiano
2) Estrae tutte le righe della tabella di origine e solo quelle che rispettano la condizione della tabella destinazione

Le righe senza corrispondenza della tabella destinazione vengono riempite con campi **NULL**

Le righe risultanti sono in numero maggiore o uguale di quelle della tabella di origine

DML

```
SELECT Compagnia, TFT.Anno, TFT.NumDoc, DataFatt, ValutaFatturato,
    AC.Cambio
FROM   tblfatturetesta AS TFT
    LEFT JOIN _at_cambi AS AC ON TFT.DataFatt = AC.Data
                                 AND TFT.ValutaFatturato = AC.CodValuta
WHERE Anno = 2020
ORDER BY TFT.NumDoc;
```

```
+-----------+------+--------+---------------------+-----------------+--------------+
| Compagnia | Anno | NumDoc | DataFatt            | ValutaFatturato | Cambio       |
+-----------+------+--------+---------------------+-----------------+--------------+
|    C      | 2020 |      1 | 2020-01-16 00:00:00 | EUR             |         NULL |
|    C      | 2020 |      2 | 2020-01-16 00:00:00 | USD             | 0.8953000000 |
|    C      | 2020 |      3 | 2020-01-16 00:00:00 | USD             | 0.8953000000 |
|    C      | 2020 |      4 | 2020-01-16 00:00:00 | USD             | 0.8953000000 |
|    C      | 2020 |      5 | 2020-01-16 00:00:00 | USD             | 0.8953000000 |
|    C      | 2020 |      6 | 2020-01-16 00:00:00 | EUR             |         NULL |
|    C      | 2020 |      7 | 2020-01-17 00:00:00 | EUR             |         NULL |
|    C      | 2020 |      8 | 2020-01-17 00:00:00 | USD             | 0.9003000000 |
|    C      | 2020 |      9 | 2020-01-17 00:00:00 | EUR             |         NULL |
|    C      | 2020 |     10 | 2020-01-17 00:00:00 | EUR             |         NULL |
|    C      | 2020 |     11 | 2020-01-17 00:00:00 | EUR             |         NULL |
...
```

DML

```
SELECT Compagnia, TFT.Anno, TFT.NumDoc, DataFatt, ValutaFatturato,
    IFNULL( AC.Cambio, '' ) AS Cambio
FROM   tblfatturetesta AS TFT
    LEFT JOIN _at_cambi AS AC ON TFT.DataFatt = AC.Data
                                 AND TFT.ValutaFatturato = AC.CodValuta
WHERE Anno = 2020
ORDER BY TFT.NumDoc;
```

```
+-----------+------+--------+---------------------+-----------------+--------------+
| Compagnia | Anno | NumDoc | DataFatt            | ValutaFatturato | Cambio       |
+-----------+------+--------+---------------------+-----------------+--------------+
|    C      | 2020 |      1 | 2020-01-16 00:00:00 | EUR             |              |
|    C      | 2020 |      2 | 2020-01-16 00:00:00 | USD             | 0.8953000000 |
|    C      | 2020 |      3 | 2020-01-16 00:00:00 | USD             | 0.8953000000 |
|    C      | 2020 |      4 | 2020-01-16 00:00:00 | USD             | 0.8953000000 |
|    C      | 2020 |      5 | 2020-01-16 00:00:00 | USD             | 0.8953000000 |
|    C      | 2020 |      6 | 2020-01-16 00:00:00 | EUR             |              |
|    C      | 2020 |      7 | 2020-01-17 00:00:00 | EUR             |              |
|    C      | 2020 |      8 | 2020-01-17 00:00:00 | USD             | 0.9003000000 |
|    C      | 2020 |      9 | 2020-01-17 00:00:00 | EUR             |              |
|    C      | 2020 |     10 | 2020-01-17 00:00:00 | EUR             |              |
...
```

DML

```
SELECT Compagnia, TFT.Anno, TFT.NumDoc, DataFatt, ValutaFatturato,
    IFNULL( AC.Cambio, '' ) AS Cambio
FROM   tblfatturetesta AS TFT
    RIGHT JOIN _at_cambi AS AC ON TFT.DataFatt = AC.Data
                                  AND TFT.ValutaFatturato = AC.CodValuta
WHERE Compagnia IS NULL;
```

```
+-----------+------+--------+----------+-----------------+--------------+
| Compagnia | Anno | NumDoc | DataFatt | ValutaFatturato | Cambio       |
+-----------+------+--------+----------+-----------------+--------------+
| NULL      | NULL |   NULL | NULL     | NULL            | 0.0005164600 |
| NULL      | NULL |   NULL | NULL     | NULL            | 1.0612000000 |
| NULL      | NULL |   NULL | NULL     | NULL            | 1.0493000000 |
| NULL      | NULL |   NULL | NULL     | NULL            | 1.0573000000 |
| NULL      | NULL |   NULL | NULL     | NULL            | 1.0477000000 |
| NULL      | NULL |   NULL | NULL     | NULL            | 1.0530000000 |
| NULL      | NULL |   NULL | NULL     | NULL            | 1.0637000000 |
| NULL      | NULL |   NULL | NULL     | NULL            | 1.0625000000 |
| NULL      | NULL |   NULL | NULL     | NULL            | 1.0501000000 |
| NULL      | NULL |   NULL | NULL     | NULL            | 1.0477000000 |
...
```

FULL OUTER JOIN

Include tutte le righe di entrambe le tabelle

Non supportato da MariaDB / MySQL

```
SELECT CodIva, Articolo, CodBanca, Banca
FROM _at_iva AS AI FULL OUTER JOIN _at_banche AS AB
    ON AI.CodIva = AB.CodBanca
ORDER BY CodIva, CodBanca;
```

```
SELECT CodIva, Articolo, CodBanca, Banca
FROM _at_iva AS AI FULL JOIN _at_banche AS AB
    ON AI.CodIva = AB.CodBanca
ORDER BY CodIva, CodBanca;
```

LIMITARE LE RIGHE RESE DA UNA QUERY

Esistono occasioni in cui è opportuno limitare o scaglionare le righe ottenute interrogando un Database

Troppe righe potrebbero comportare un eccessivo consumo di risorse oppure un tempo di elaborazione troppo prolungato

Non esiste una soluzione stndardizzata, ma solo estensioni proprietarie del linguaggio SQL

MariaDB / MySQL usa le istruzioni **LIMIT** e **OFFSET**

```
SELECT *
FROM employees
LIMIT 10;
```

```
+--------+------------+------------+-----------+--------+------------+
| emp_no | birth_date | first_name | last_name | gender | hire_date  |
+--------+------------+------------+-----------+--------+------------+
|  10001 | 1953-09-02 | Georgi     | Facello   | M      | 1986-06-26 |
|  10002 | 1964-06-02 | Bezalel    | Simmel    | F      | 1985-11-21 |
|  10003 | 1959-12-03 | Parto      | Bamford   | M      | 1986-08-28 |
|  10004 | 1954-05-01 | Chirstian  | Koblick   | M      | 1986-12-01 |
|  10005 | 1955-01-21 | Kyoichi    | Maliniak  | M      | 1989-09-12 |
|  10006 | 1953-04-20 | Anneke     | Preusig   | F      | 1989-06-02 |
|  10007 | 1957-05-23 | Tzvetan    | Zielinski | F      | 1989-02-10 |
|  10008 | 1958-02-19 | Saniya     | Kalloufi  | M      | 1994-09-15 |
|  10009 | 1952-04-19 | Sumant     | Peac      | F      | 1985-02-18 |
|  10010 | 1963-06-01 | Duangkaew  | Piveteau  | F      | 1989-08-24 |
+--------+------------+------------+-----------+--------+------------+
```

```
SELECT *
FROM employees
LIMIT 10
OFFSET 10;
```

```
+--------+------------+------------+-------------+--------+------------+
| emp_no | birth_date | first_name | last_name   | gender | hire_date  |
+--------+------------+------------+-------------+--------+------------+
|  10011 | 1953-11-07 | Mary       | Sluis       | F      | 1990-01-22 |
|  10012 | 1960-10-04 | Patricio   | Bridgland   | M      | 1992-12-18 |
|  10013 | 1963-06-07 | Eberhardt  | Terkki      | M      | 1985-10-20 |
|  10014 | 1956-02-12 | Berni      | Genin       | M      | 1987-03-11 |
|  10015 | 1959-08-19 | Guoxiang   | Nooteboom   | M      | 1987-07-02 |
|  10016 | 1961-05-02 | Kazuhito   | Cappelletti | M      | 1995-01-27 |
|  10017 | 1958-07-06 | Cristinel  | Bouloucos   | F      | 1993-08-03 |
|  10018 | 1954-06-19 | Kazuhide   | Peha        | F      | 1987-04-03 |
|  10019 | 1953-01-23 | Lillian    | Haddadi     | M      | 1999-04-30 |
|  10020 | 1952-12-24 | Mayuko     | Warwick     | M      | 1991-01-26 |
+--------+------------+------------+-------------+--------+------------+
```

Oracle < 12G

```
SELECT *
FROM employees
WHERE ROWNUM BETWEEN 11 AND 20;
```

Oracle ≥ 12G

```
SELECT *
FROM employees
OFFSET 10 ROWS
    FETCH NEXT 10 ROWS ONLY;
```

Ms SQL Server < 2012

```
SELECT TOP 10 *
FROM ( SELECT TOP 20 *
            FROM employees
           ORDER BY emp_id
         )
ORDER BY emp_id DESC;
```

Ms SQL Server ≥ 2012

```
SELECT *
FROM employees
ORDER BY emp_id
OFFSET 10 ROWS
    FETCH NEXT 10 ROWS ONLY;
```

Aggiunta di Nuovi Dati

Le tabelle di un database devono essere popolate ⇒
Esiste un'istruzione specifica per il compito: **INSERT**

```
INSERT INTO <schema>.<table> [ ( <column> [, ... ] ) ]
VALUES( <value> [, ... ] );
```

Consente di inserire una singola riga di nuovi dati in una tabella

```
INSERT INTO departments
VALUES( 'd010', 'Information Technology' );
```

```
Query OK, 1 row affected (0.08 sec)
```

I nomi delle colonne sono obbligatori quando si inseriscono dati solo in alcune di esse

```
INSERT INTO departments( dept_no, dept_name )
VALUES( 'd011', 'Software Development' );
```

```
Query OK, 1 row affected (0.25 sec)
```

```
INSERT INTO departments( dept_no )
VALUES( 'd012' );
```

```
Query OK, 1 row affected, 1 warning (0.07 sec)
```

Nelle colonne non in elenco vengono inseriti i valori di default

DML

È necessario usare tutti i campi della chiave primaria ed evitare duplicazioni:

```
INSERT INTO titles( emp_no, title )
VALUES( '499999', 'Stagista' );
```

```
Query OK, 1 row affected, 1 warning (0.14 sec)
```

```
SELECT * FROM titles WHERE emp_no = '499999';
```

```
+--------+----------+------------+------------+
| emp_no | title    | from_date  | to_date    |
+--------+----------+------------+------------+
| 499999 | Engineer | 1997-11-30 | 9999-01-01 |
| 499999 | Stagista | 0000-00-00 | NULL       |
+--------+----------+------------+------------+
2 rows in set (0.00 sec)
```

DML

Ripetendo l’inserimento si verrebbe a creare una duplicazione ed il conseguente errore

```
INSERT INTO titles( emp_no, title )
VALUES( '499999', 'Stagista' );
```

```
ERROR 1062 (23000): Duplicate entry '499999-Stagista-0000-00-00' for key
'PRIMARY'
```

DML

È possibile utilizzare **NULL** come valore per gli inserimenti

Non deve violare i vincoli imposti sui dati

```
INSERT INTO titles( emp_no, title, from_date, to_date )
VALUES( '499999', 'Commesso', NULL, NULL );
```

```
ERROR 1048 (23000): Column 'from_date' cannot be null
```

DML

È possibile combinare una query INSERT con SELECT per aggiungere molte righe con un solo comando. Il numero delle colonne impostate nelle 2 query deve coincidere

INSERT INTO <table_name> [(<column> [, ...])]
SELECT ...;

In caso di duplicazioni l'operazione fallisce rendendo un errore

- Transazione attiva ⇒ Rimuove tutte le righe aggiunte sino al momento dell'errore
- Commit automatico ⇒ Le righe inserite rimangono nella tabella

Utilizzando anche il nome dello schema, è possibile copiare dati da tabelle di altri database gestiti dalla medesima istanza del RDBMS

Modifica di Dati Esistenti

I valori contenuti nei campi delle righe possono essere modificati mediante l'istruzione **UPDATE**

```
UPDATE [ <schema>. ]<table_name>
SET <field> = <value> [ , <field> = <value> ]
[ WHERE <condition> ];
```

```
UPDATE departments
SET dept_name = 'Education'
WHERE dept_no = 'd011';
```

```
Query OK, 1 row affected (0.57 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

DML

È possibile specificare più colonne in un'unica istruzione di modifica. L'ordine delle assegnazioni può differire da quello delle colonne della tabella

```
+--------+-----------------+------------+------------+
| emp_no | title           | from_date  | to_date    |
+--------+-----------------+------------+------------+
```

```
UPDATE titles
SET title = 'Staff Junior', to_date = '1995-07-09',
    from_date = '1988-05-13'
WHERE emp_no = 10007
    AND  from_date = '1989-02-10';
```

```
Query OK, 1 row affected (0.13 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

DML

È possibile specificare il valore **DEFAULT** per impostare il valore predefinito di un campo, configurato durante la creazione di una tabella

L’istruzione UPDATE Consente di aggiornare molte righe con un'unica operazione. Le righe interessate dipendono dalle condizioni nella clausola **WHERE**

Se non fosse presente una condizione **WHERE** verrebbero aggiornate tutte le righe di una tabella

Modifica di Dati Contenuti in Più Tabelle

È possibile impostare più tabelle nella prima parte della istruzione **UPDATE**

Vengono modificate le righe di tutte le tabelle elencate

La sintassi non è standard, ma dipende dal RDBMS

MariaDB / MySQL

```
UPDATE vtiger_sginvoice AS VSGI
    INNER JOIN vtiger_sginvoicecf AS VSGIC
        ON VSGI.sginvoiceid = VSGIC.sginvoiceid
     LEFT  JOIN vtiger_sgvoyagecf  AS VSGV
        ON VSGV.voyage_gescon_code =
        VSGIC.invoice_voyage_gescon_code
SET VSGI.entity_origin = VSGV.sgvoyageid
WHERE VSGI.sginvoice_expense_type != 'time_charter'
AND      VSGIC.invoice_voyage_gescon_code != ''
AND     VSGIC.invoice_gescon_code != ''
AND     VSGI.entity_origin = 0;
```

Ms SQL Server

```
UPDATE dbo.Table_2
SET dbo.Table_2.Col_B =
    dbo.Table_2.Col_B + dbo.Table_1.Col_B
FROM dbo.Table_2
    INNER JOIN dbo.Table_1 ON
        ( dbo.Table_2.Col_A = dbo.Table_1.Col_A );
```

Le sotto-query possono essere utilizzate in un'istruzione **UPDATE**

```
UPDATE vtiger_sginvoice
SET vat_exchange_rate = (
    SELECT DISTINCT VSGER.sgexchangerateid
    FROM  vtiger_sgexchangerate AS VSGER
    WHERE VSGER.currencyid =
        vtiger_sginvoice.invoice_currencyid
    AND VSGER.exchange_date =
        vtiger_sginvoice.invoice_date
  )
  WHERE vat_exchange_rate = 0
  AND sginvoiceid NOT IN( 66849, 66920, 66925 );
```

DML

Le funzioni ed i valori calcolati possono essere impiegati in un'istruzione **UPDATE**

```
UPDATE sgimport_invoice_dhodetail AS SIIDD
SET SIIDD.expense_summary = CONCAT(
      SIIDD.sgexpense_type,        '-',
      SIIDD.expense_description, '-',
      SIIDD.quantity );

UPDATE vtiger_sginvoice AS VSGI
      INNER JOIN vtiger_sginvoicedhototal VSGIDHOT
            ON VSGI.sginvoiceid = VSGIDHOT.sginvoiceid
      LEFT JOIN vtiger_sgvat AS VSGV
            ON VSGI.sgvatid = VSGV.sgvatid
SET VSGIDHOT.tax = IFNULL(( VSGIDHOT.taxable_post_exchange *
       VSGV.vat_rate ) / 100, 0 );
```

Cancellazione di Dati

I dati obsoleti, errati, superflui, non più necessari possono essere rimossi da una tabella mediante l'istruzione **DELETE**

Rimuove le righe salvandone una copia nei segmenti di rollback, nel transaction log o in altro sistema di preservazione dei dati

DELETE FROM [<schema>.]<table_name>
[WHERE <condition>];

DML

```
DELETE FROM departments
WHERE dept_no = 'd011';
```

```
Query OK, 1 row affected (0.16 sec)
```

```
DELETE FROM departments
WHERE dept_name = '';
```

```
Query OK, 1 row affected (0.03 sec)
```

Quando non viene specificata una condizione, vengono rimossi tutti i dati di una tabella

```
DELETE FROM employees;
```

```
Query OK, 300024 rows affected (2 min 17.15 sec)
```

DML

L'istruzione **DELETE** attiva tutti i vincoli di integrità referenziale:

- Foreign Key
- Trigger
- …

La violazione di un vincolo di integrità può comportare il fallimento di una cancellazione di dati

Informazioni & Licenze