

Basi di dati cod. 861II [9 CFU]

Corso di Laurea in Ingegneria Informatica

Oracle MySQL
A.A. 2022-2023

Francesco Pistolesi
Dipartimento di Ingegneria dell'Informazione
Università di Pisa
francesco.pistolesi@unipi.it

Data analytics



Window functions [a.k.a. analytic functions]

Medico	Paziente	Data	Mutuata
001	slq6	2003-06-03	0
001	slq6	2004-04-05	0
002	slq6	2005-01-16	0
002	slq6	2007-11-30	0
003	slq6	2009-12-03	0
003	slq6	2012-07-21	0
004	slq6	2007-02-23	0
005	slq6	2005-02-22	0
005	slq6	2009-11-21	0
006	slq6	2006-01-24	0
006	slq6	2009-07-18	0
007	slq6	2004-11-30	0
007	slq6	2009-08-01	0
001	slplz	2003-08-01	0
001	slplz	2004-07-30	0
000	slplz	2005-07-05	0
000	slplz	2006-07-05	0
000	slplz	2007-07-05	0
000	slplz	2008-07-05	0

Medico	Paziente	Data	Mutuata
001	slq6	2003-06-03	0
001	slq6	2004-04-05	0
002	slq6	2005-01-16	0
002	slq6	2007-11-30	0
003	slq6	2009-12-03	0
003	slq6	2012-07-21	0
004	slq6	2007-02-23	0
005	slq6	2005-02-22	0
005	slq6	2009-11-21	0
006	slq6	2006-01-24	0
006	9bts	2009-07-18	0
007	9bts	2004-11-30	0
007	9bts	2009-08-01	0
008		2010-08-0005	0
008		0E-LL-4005	0
009		9bts	0
009		8L-TD-0005	0
009		9bts	0
009		PS-LD-5000	0
009		9bts	0

Medico	Paziente	Data	Mutua
001	slq6	2003-06-03	0
001	slq6	2004-04-05	0
002	slq6	2005-01-16	0
002	slq6	2007-11-30	0
003	slq6	2009-12-03	0
003	slq6	2012-07-21	0
004	slq6	2007-02-23	0
005	slq6	2005-02-22	0
005	slq6	2009-11-21	0
006	slq6	2006-01-24	0
006	slq6	2009-07-18	0
007	slq6	2004-11-30	0
007	slq6	2009-08-01	0
001	slq6	2006-08-05	0
001	slq6	2007-11-04	0
001	slq6	2008-01-25	0
001	slq6	2008-07-05	0
002	slq6	2009-01-15	0

Affiancano a ogni **record** r un valore ottenuto da un'operazione, eseguita su un **insieme di record logicamente connessi a r** .

conteggio, media, rank, ...

Window functions: un primo esempio

Scrivere una query che indichi, per ogni cardiologo, la matricola, la parcella, e la parcella media della sua specializzazione



ogni record del result set è un record della tabella Medico,
tutti i record relativi ai cardiologi devono essere mantenuti,
tuttavia, occorre anche applicare un operatore di aggregazione

Primo step

Scrivere una query che indichi, per ogni cardiologo, la matricola, la parcella, e la parcella media della sua specializzazione

1 `SELECT AVG(M.Parcella)` ← fa squash ed elimina i record originali!
2 `FROM Medico M`
3 `WHERE M.Specializzazione = 'Cardiologia';`

Matricola	Cognome	Nome	Specializzazione	Parcella	Città
014	Indachi	Loredana	Cardiologia	191	Pisa
015	Ciani	Gualtiero	Cardiologia	244	Pisa
016	Verdolini	Maria Paola	Cardiologia	233	Pisa
017	Amaranti	Adevane	Cardiologia	275	Pisa
018	Terra di Siena	Bruciata	Cardiologia	275	Siena



AVG(M.Parcella)
243.6000



per calcolare la parcella media, si perdono i record dei medici

Clausola OVER

le funzioni aggregate sono
sum, avg, count, max, min...

usano la partition ottenendo un valore
scalare che però non è il risultato di un
operatore di aggregazione dei record
che la compongono

Applica una funzione di tipo *aggregate* o *non-aggregate* a
un **insieme di record associati a un record di un result set**

detto "partition"



Soluzione

Scrivere una query che indichi, per ogni cardiologo, la matricola, la parcella, e la parcella media della sua specializzazione

```
1 SELECT
2     M.Matricola,
3     M.Parcella,
4     AVG(M.Parcella) OVER()
5 FROM
6     Medico M
7 WHERE
8     M.Specializzazione = 'Cardiologia';
```



Matricola	Parcella	AVG(M.Parcella)
014	180	230.0000
015	230	230.0000
016	220	230.0000
017	260	230.0000
018	260	230.0000

OVER()

se over non ha contenuto, la partition è sempre la stessa per ogni record, e coincide con il result set della query senza la parte arancio

Definizione della partition

È l'insieme di record a cui si applica una funzione aggregate/nonaggregate
e dipende dal record processato



Può essere definita anche su attributi non proiettati

Esempio con definizione della partition

Scrivere una query che indichi, **per ogni medico**, la matricola, la specializzazione, la parcella, e la parcella media della sua specializzazione



non solo la cardiologia, ma tutte le specializzazioni

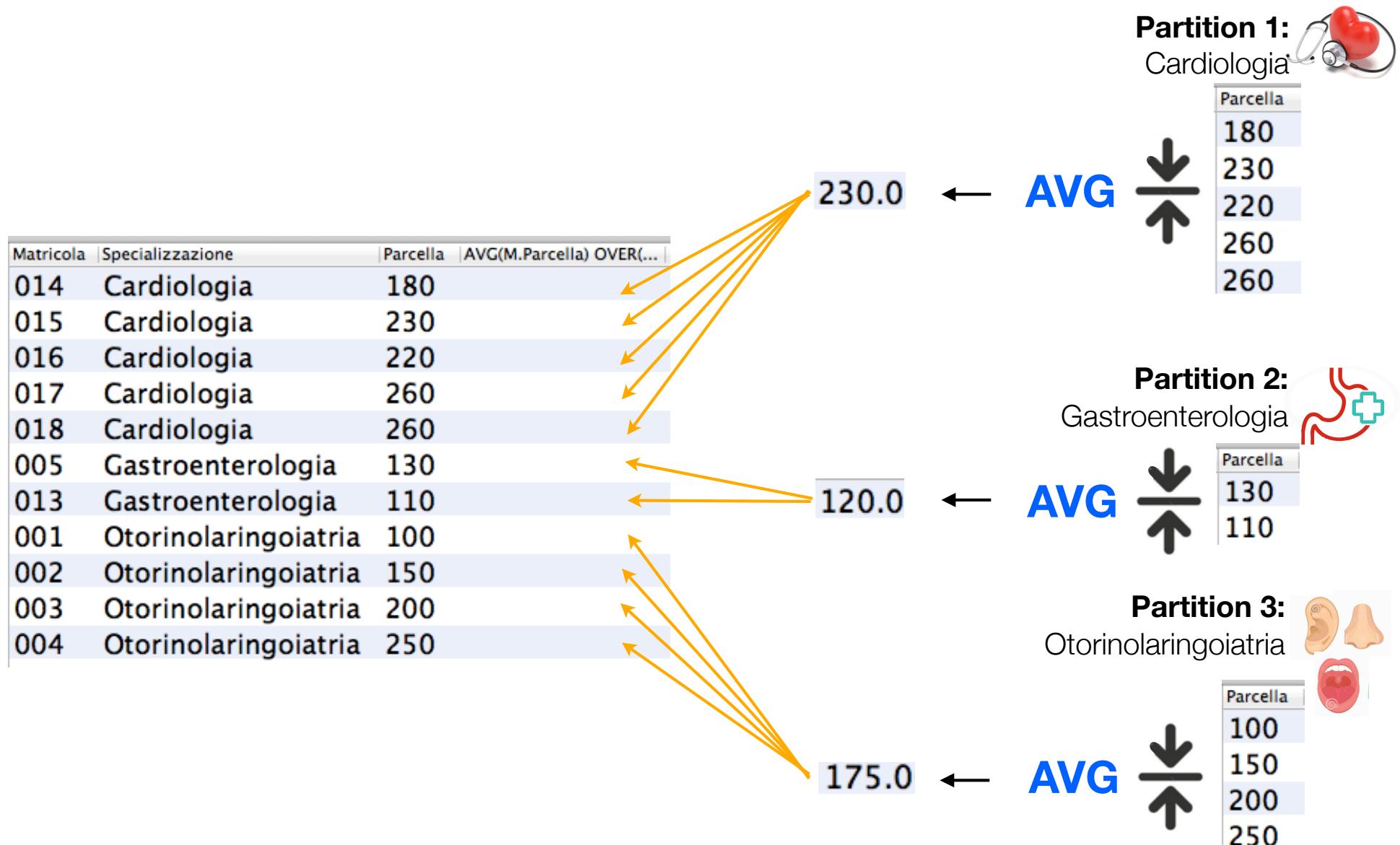
OVER con PARTITION BY

Scrivere una query che indichi, per ogni medico, la matricola, la specializzazione, la parcella, e la parcella media della sua specializzazione

```
1  SELECT
2      M.Matricola,
3      M.Specializzazione,
4      M.Parcella,
5      AVG(M.Parcella) OVER(
6          PARTITION BY
7              M.Specializzazione
8      )
9  FROM
10     Medico M;
```

avg() è applicato partizionando i record del result set della query per specializzazione; ad ogni record del result set si associa poi il valore avg() relativo alla corrispondente specializzazione medica

Funzionamento



Risultato

Scrivere una query che indichi, **per ogni medico**, la matricola, la parcella, e la parcella media della sua specializzazione

Matricola	Specializzazione	Parcella	AVG(M.Parcella) OVER(
017	Cardiologia	260	230.0000
014	Cardiologia	180	230.0000
016	Cardiologia	220	230.0000
018	Cardiologia	260	230.0000
015	Cardiologia	230	230.0000
005	Gastroenterologia	130	120.0000
013	Gastroenterologia	110	120.0000
001	Otorinolaringoiatria	100	175.0000
003	Otorinolaringoiatria	200	175.0000
002	Otorinolaringoiatria	150	175.0000
004	Otorinolaringoiatria	250	175.0000

NOTA: in questo esempio e nei prossimi, per ragioni di spazio, si è supposto che le specializzazioni dei medici della clinica siano solamente quelle mostrate.

Aggregate functions utilizzabili con OVER

Name	Description
<u>AVG ()</u>	Return the average value of the argument
<u>BIT_AND ()</u>	Return bitwise AND
<u>BIT_OR ()</u>	Return bitwise OR
<u>BIT_XOR ()</u>	Return bitwise XOR
<u>COUNT ()</u>	Return a count of the number of rows returned
<u>COUNT (DISTINCT)</u>	Return the count of a number of different values
<u>GROUP_CONCAT ()</u>	Return a concatenated string
<u>JSON_ARRAYAGG ()</u>	Return result set as a single JSON array
<u>JSON_OBJECTAGG ()</u>	Return result set as a single JSON object
<u>MAX ()</u>	Return the maximum value
<u>MIN ()</u>	Return the minimum value
<u>STD ()</u>	Return the population standard deviation
<u>STDDEV ()</u>	Return the population standard deviation
<u>STDDEV_POP ()</u>	Return the population standard deviation
<u>STDDEV_SAMP ()</u>	Return the sample standard deviation
<u>SUM ()</u>	Return the sum
<u>VAR_POP ()</u>	Return the population standard variance
<u>VAR_SAMP ()</u>	Return the sample variance
<u>VARIANCE ()</u>	Return the population standard variance

→ già viste

Non-aggregate functions utilizzabili con OVER

Name	Description
CUME_DIST()	Cumulative distribution value
DENSE_RANK()	Rank of current row within its partition, without gaps
FIRST_VALUE()	Value of argument from first row of window frame
LAG()	Value of argument from row lagging current row within partition
LAST_VALUE()	Value of argument from last row of window frame
LEAD()	Value of argument from row leading current row within partition
NTH_VALUE()	Value of argument from N-th row of window frame
NTILE()	Bucket number of current row within its partition.
PERCENT_RANK()	Percentage rank value
RANK()	Rank of current row within its partition, with gaps
ROW_NUMBER()	Number of current row within its partition

 usano l'**intera partition**

 lavorano **su frame**, cioè sottoinsiemi della partition (vediamo dopo)

Window functions non-aggregate



le funzioni non-aggregate che lavorano su frame (quelle) considerano, per default, i record dall'inizio della partition fino alla current row se si specifica una order by all'interno di over; senza order by, considerano tutti i record della partition

first_value, last_value, nth_value

Associano a ciascun record di un result set un valore ottenuto dalla partition **senza fonderne i record in un'informazione riepilogativa**



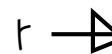
come invece fanno le funzioni aggregate (count, sum, avg...)

Funzione ROW_NUMBER

Assegnare un **numero** a ogni medico nella sua specializzazione

```
1 SELECT
2   M.Matricola,
3   M.Cognome,
4   M.Specializzazione,
5   ROW_NUMBER() OVER(PARTITION BY
6     M.Specializzazione)
7 FROM
8   Medico M;
```

row_number() non fa squash,
non le serve aggregate!



Un istante della processazione:
la current row r (in rosso) e la
current partition (rosa).

Matricola	Cognome	Specializzazione	ROW_NUMBER
014	Indachi	Cardiologia	1
015	Ciani	Cardiologia	2
016	Verdolini	Cardiologia	3
017	Amaranti	Cardiologia	4
018	Terra di Siena	Cardiologia	5
005	Neri	Gastroenterologia	1
013	Blu	Gastroenterologia	2
001	Rossi	Otorinolaringoiatria	1
002	Verdi	Otorinolaringoiatria	2
003	Gialli	Otorinolaringoiatria	3
004	Turchesi	Otorinolaringoiatria	4

Funzione RANK

è spesso un attributo su cui si fa un sort

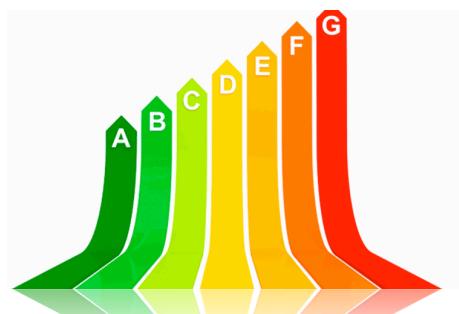
Serve per **stilare una classifica** dipendentemente da **un criterio**



il criterio permette di associare uno score a ogni record, più è alto lo score, migliore è il rank

Esempio di rank

Classificare i medici in base alla loro convenienza. Restituire matricola, cognome, specializzazione, parcella e posizione in classifica.



parcelle basse hanno maggior convenienza, quindi rank migliori

Soluzione

Effettuare una classifica della convenienza dei medici dipendentemente dalla loro parcella. Restituire matricola, cognome, specializzazione, parcella e posizione in classifica.

```
1  SELECT
2      M.Matricola,
3      M.Cognome,
4      M.Specializzazione,
5      M.Parcella,
6      RANK() OVER(
7          ORDER BY M.Parcella
8      )
9  FROM
10     Medico M;
```



per ogni row (current row), la partition è ottenuta calcolando il result set della query (righe da 1 a 5 e da 9 a 10), ordinandolo in modo crescente su Parcella, e selezionando infine le row dall'inizio fino alla current row

Risultato

Matricola	Cognome	Specializzazione	Parcella	RANK()
012	Grigi	Medicina generale	50	1
001	Rossi	Otorinolaringoiatria	100	2
019	Rossi	Oculistica	100	2
013	Blu	Gastroenterologia	110	4
020	Acquamarina	Oculistica	110	4
008	Gialli	Nefrologia	120	6
005	Neri	Gastroenterologia	130	7
002	Verdi	Otorinolaringoiatria	150	8
011	Marroni	Psichiatria	150	8
006	Bianchi	Ortopedia	160	10
009	Arancioni	Nefrologia	170	11
007	Rosi	Ortopedia	180	12
014	Indachi	Cardiologia	180	12
003	Gialli	Otorinolaringoiatria	200	14
010	Celesti	Neurologia	200	14
016	Verdolini	Cardiologia	220	16
015	Ciani	Cardiologia	230	17
004	Turchesi	Otorinolaringoiatria	250	18
017	Amaranti	Cardiologia	260	19
018	Terra di Siena	Cardiologia	260	19

LOOK
AT THIS



la posizione 3 non esiste,
come si vede, in caso di pari
merito, la funzione rank() salta
un numero di posizioni della
classifica pari al numero di ex
aequo trovati

Altro esempio: rank su partition

Effettuare una classifica dei medici **di ogni specializzazione** dipendentemente dalla loro parcella, partendo dalla più alta. Restituire matricola, cognome, specializzazione, parcella e posizione in classifica.



Il rank è tanto più piccolo (migliore) quanto più la parcella è alta

Soluzione

Effettuare una classifica dei medici **di ogni specializzazione** dipendentemente dalla loro parcella, partendo dalla più alta. Restituire matricola, cognome, specializzazione, parcella e posizione nella classifica.

```
1  SELECT
2      M.Matricola,
3      M.Cognome,
4      M.Specializzazione,
5      RANK() OVER(
6          PARTITION BY M.Specializzazione
7          ORDER BY M.Parcella DESC
8      )
9  FROM
10     Medico M;
```

occorre partizionare in questo caso, la partition è composta dai soli medici della specializzazione della current row



4

ordinamento decrescente dei medici in ogni specializzazione, dipendentemente dalla parcella

Sintassi alternativa con WINDOW

```
1 SELECT
2   M.Matricola,
3   M.Cognome,
4   M.Specializzazione,
5   RANK() OVER(
6     PARTITION BY M.Specializzazione
7     ORDER BY M.Parcella DESC
8   )
9 FROM
10 Medico M;
```

utile per non ripetere la definizione della partition quando si proiettano più attributi basati su di essa



```
1 SELECT
2   M.Matricola,
3   M.Cognome,
4   M.Specializzazione,
5   RANK() OVER w
6 FROM
7   Medico M
8   WINDOW w AS (PARTITION BY M.Specializzazione
9             ORDER BY M.Parcella DESC);
```



Come funziona?

Result set della query

Matricola	Cognome	Specializzazione	RANK()
014	Indachi	Cardiologia	
015	Ciani	Cardiologia	
016	Verdolini	Cardiologia	
017	Amaranti	Cardiologia	
018	Terra di Siena	Cardiologia	
005	Neri	Gastroenterologia	
013	Blu	Gastroenterologia	
012	Grigi	Medicina generale	
008	Gialli	Nefrologia	
009	Arancioni	Nefrologia	
010	Celesti	Neurologia	
019	Rossi	Oculistica	
020	Acquamarina	Oculistica	
006	Bianchi	Ortopedia	
007	Rosi	Ortopedia	
001	Rossi	Otorinolaringoiatria	
002	Verdi	Otorinolaringoiatria	
003	Gialli	Otorinolaringoiatria	
004	Turchesi	Otorinolaringoiatria	
011	Marroni	Psichiatria	

Partition [Cardiologia]

017	Amaranti	Cardiologia	260
018	Terra di Siena	Cardiologia	260
015	Ciani	Cardiologia	230
016	Verdolini	Cardiologia	220
014	Indachi	Cardiologia	180

per associare il rank a questa row, viene costruita la partition a essa associata (quella della Cardiologia, sopra, in viola), dopodiché si cerca la posizione che 015 ricopre nella partition. In questo caso, a 015 sarà associato rank=3.

IMPORTANT

la partition cambia quando cambia la specializzazione

Risultato di RANK su partition

	Matricola	Cognome	Specializzazione	RANK()
A	017	Amaranti	Cardiologia	1
	018	Terra di Siena	Cardiologia	1
	015	Ciani	Cardiologia	3
	016	Verdolini	Cardiologia	4
	014	Indachi	Cardiologia	5
B	005	Neri	Gastroenterologia	1
	013	Blu	Gastroenterologia	2
C	012	Grigi	Medicina generale	1
	009	Arancioni	Nefrologia	1
D	008	Gialli	Nefrologia	2
	010	Celesti	Neurologia	1
E	020	Acquamarina	Oculistica	1
	019	Rossi	Oculistica	2
F	007	Rosi	Ortopedia	1
	006	Bianchi	Ortopedia	2
G	004	Turchesi	Otorinolaringoiatria	1
	003	Gialli	Otorinolaringoiatria	2
H	002	Verdi	Otorinolaringoiatria	3
	001	Rossi	Otorinolaringoiatria	4
I	011	Marroni	Psichiatria	1

Funzione DENSE_RANK

Classifica i record come la funzione *rank*, ma in caso di ex aequo, la row che segue i record a pari merito assume rank **immediatamente successivo**



il dense rank è anche detto "rank senza gap"

Stesso esempio, ma dense rank

Effettuare una classifica senza gap dei medici **di ogni specializzazione** dipendentemente dalla loro parcella, partendo dalla più alta. Restituire matricola, cognome, specializzazione, parcella e posizione nella classifica.

```
1  SELECT
2      M.Matricola,
3      M.Cognome,
4      M.Specializzazione,
5      DENSE_RANK() OVER(
6          PARTITION BY M.Specializzazione
7          ORDER BY M.Parcella DESC
8      )
9  FROM
10     Medico M;
```

Risultato di DENSE_RANK su partition

Matricola	Cognome	Specializzazione	DENSE_RANK()
017	Amaranti	Cardiologia	1
018	Terra di Siena	Cardiologia	1
015	Ciani	Cardiologia	2
016	Verdolini	Cardiologia	3
014	Indachi	Cardiologia	4
005	Neri	Gastroenterologia	1
013	Blu	Gastroenterologia	2
012	Grigi	Medicina generale	1
009	Arancioni	Nefrologia	1
008	Gialli	Nefrologia	2
010	Celesti	Neurologia	1
020	Acquamarina	Oculistica	1
019	Rossi	Oculistica	2
007	Rosi	Ortopedia	1
006	Bianchi	Ortopedia	2
004	Turchesi	Otorinolaringoiatria	1
003	Gialli	Otorinolaringoiatria	2
002	Verdi	Otorinolaringoiatria	3
001	Rossi	Otorinolaringoiatria	4
011	Marroni	Psichiatria	1

LOOK
AT THIS

Rank multipli

Stilare una classifica dei medici **in base al numero di visite effettuate**.

Restituire cognome, specializzazione, numero di viste effettuate, posizione nella **classifica generale**, e posizione nella **classifica per specializzazione**.

- 1 ogni medico è confrontato con tutti gli altri



- 2 ogni medico è confrontato con quelli della sua specializzazione



Soluzione

```
1 WITH visite AS
2 -( 
3   SELECT V.Medico,
4         M.Cognome,
5         M.Specializzazione,
6         COUNT(*) AS Visite
7   FROM Visita V
8     INNER JOIN Medico M ON V.Medico = M.Matricola
9   GROUP BY V.Medico
10 -)
11  SELECT VV.Cognome,
12        VV.Specializzazione,
13        VV.Visite,
14        RANK() OVER(ORDER BY VV.Visite DESC) AS GlobalRank,
15    - RANK() OVER(PARTITION BY VV.Specializzazione
16    -           ORDER BY VV.Visite DESC) AS SpecRank
17  FROM visite VV;
```

Risultato

Cognome	Specializzazione	Visite	GlobalRank	SpecRank
Indachi	Cardiologia	19	13	1
Terra di Siena	Cardiologia	19	13	1
Ciani	Cardiologia	17	15	3
Verdolini	Cardiologia	17	15	3
Amaranti	Cardiologia	11	18	5
Neri	Gastroenterologia	36	3	1
Blu	Gastroenterologia	17	15	2
Grigi	Medicina generale	25	11	1
Arancioni	Nefrologia	34	7	1
Gialli	Nefrologia	25	11	2
Celesti	Neurologia	36	3	1
Rossi	Oculistica	4	19	1
Acquamarina	Oculistica	3	20	2
Bianchi	Ortopedia	40	1	1
Rosi	Ortopedia	40	1	1
Verdi	Otorinolaringoiatria	35	6	1
Gialli	Otorinolaringoiatria	33	8	2
Rossi	Otorinolaringoiatria	28	9	3
Turchesi	Otorinolaringoiatria	27	10	4
Marroni	Psichiatria	36	3	1

rank calcolati su partition diverse, la 1 e la 2

Funzioni LEAD e LAG



Back



Forward

Lead & Lag

Ricavano il valore di un attributo di una row **posta k posizioni prima (lag) o dopo (lead)** la current row

che a sua volta può anche essere
argomento di altre function



si presuppone un ordinamento sensato nella partition
affinché il "balzo" in avanti o indietro abbia un significato

Funzione LAG

Permette di affiancare a ogni record i il valore di un record j posizionato
 k posizioni prima del record i all'interno della partition associata a i



Back

fa un balzo indietro nella partition di k rows, dove il valore di k deve essere passato come secondo argomento della funzione, mentre il primo argomento è il nome dell'attributo del record j da restituire in uscita

Esempio

Considerare le visite otorinolaringoiatriche dal 2010 al 2019, restituire, per ciascuna, matricola del medico, codice fiscale del paziente, data, e data della visita precedente del paziente con un medico della stessa specializzazione



per ogni visita √ bisogna creare una partition con le visite che il paziente ha effettuato in precedenza con medici della stessa specializzazione di quello che ha effettuato la visita √

Soluzione con LAG

```
1 SELECT
2   V.Medico,
3   V.Paziente,
4   V.`Data`,
5   LAG(V.`Data`, 1) OVER (
6     PARTITION BY V.Paziente
7
8     ORDER BY V.`Data`
9   )
10 FROM
11   Visita V
12   INNER JOIN
13     Medico M ON V.Medico = M.Matricola
14 WHERE
15   M.Specializzazione = 'Otorinolaringoiatria'
16   AND YEAR(V.`Data`) BETWEEN 2010 AND 2019;
```

significa che $k=1$, quindi si va indietro di una sola row
e si affianca alla current row (i) la data della visita j
posta nella posizione $(i-k)$

Esempi di partition

```
OVER (
    PARTITION BY V.Paziente
    ORDER BY V.`Data`
)
```

Medico	Paziente	Data	LAG(V.`Data`, 1) OVER (
003	aaa1	2012-05-23	[Null]
001	aaa1	2012-12-01	2012-05-23
001	aaa1	2013-03-01	2012-12-01
004	bbc4	2010-10-03	[Null]
i1	004	bbe1	2010-05-30 [Null]
	003	bbe1	2011-11-27 2010-05-30
	001	ccc2	2012-07-27 [Null]
	002	ddd6	2010-10-16 [Null]
i2	004	eey9	2010-10-16 [Null]
	003	eey9	2011-01-23 2010-10-16
	002	eey9	2012-02-16 2011-01-23
	004	erv4	2012-06-07 [Null]
	004	ffq8	2012-09-03 [Null]
	002	ffq8	2013-01-26 2012-09-03
	001	ffq8	2013-02-01 2013-01-26
	002	fqa8	2010-11-16 [Null]
	001	gwp5	2010-01-19 [Null]
i3	004	hho1	2011-01-29 [Null]
	003	hho1	2011-08-31 2011-01-29
	002	hho1	2012-02-15 2011-08-31

Partition del record i1

004	bbe1	2010-05-30	[Null]
003	bbe1	2011-11-27	2010-05-30

Partition del record i2

004	eey9	2010-10-16	[Null]
003	eey9	2011-01-23	2010-10-16
002	eey9	2012-02-16	2011-01-23

Partition del record i3

004	hho1	2011-01-29	[Null]
003	hho1	2011-08-31	2011-01-29
002	hho1	2012-02-15	2011-08-31

order by Data

Risultato finale di LAG

Medico	Paziente	Data	LAG(V. `Data`, 1) OVER (
003	aaa1	2012-05-23	[Null]
001	aaa1	2012-12-01	2012-05-23
001	aaa1	2013-03-01	2012-12-01
004	bbc4	2010-10-03	[Null]
004	bbe1	2010-05-30	[Null]
003	bbe1	2011-11-27	2010-05-30
001	ccc2	2012-07-27	[Null]
002	ddd6	2010-10-16	[Null]
004	eyy9	2010-10-16	[Null]
003	eyy9	2011-01-23	2010-10-16
002	eyy9	2012-02-16	2011-01-23
004	erv4	2012-06-07	[Null]
004	ffq8	2012-09-03	[Null]
002	ffq8	2013-01-26	2012-09-03
001	ffq8	2013-02-01	2013-01-26
002	fqa8	2010-11-16	[Null]
001	gwp5	2010-01-19	[Null]
004	hho1	2011-01-29	[Null]
003	hho1	2011-08-31	2011-01-29
002	hho1	2012-02-15	2011-08-31

Funzione LEAD

Permette di affiancare a ogni record i il valore di un record j posizionato **k posizioni dopo** il record i all'interno della partition associata a i



Forward

fa un balzo in avanti nella partition di k rows, dove il valore di k deve essere passato come secondo argomento della funzione, mentre il primo argomento è il nome dell'attributo del record j da restituire in uscita

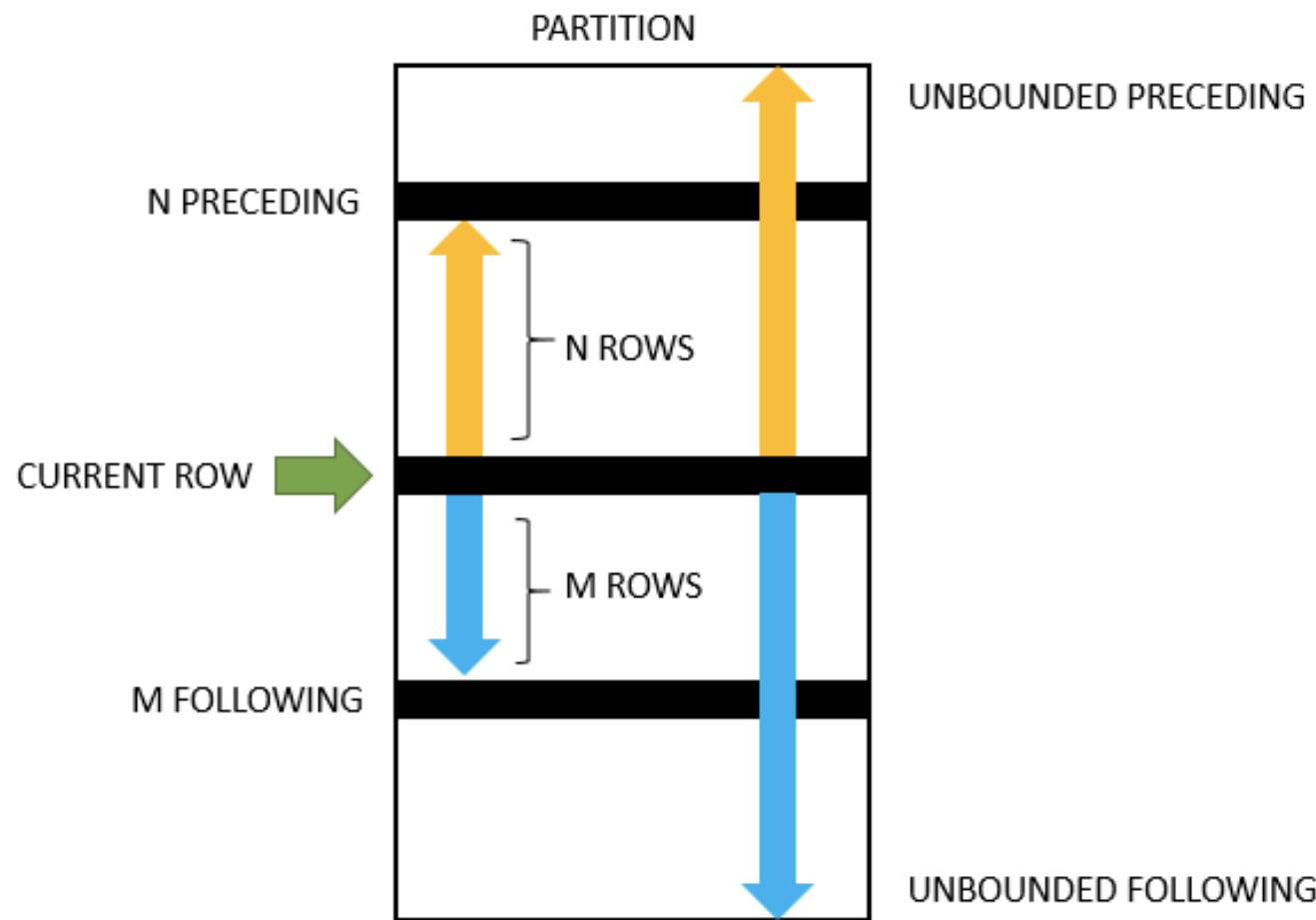
Esempio di lead

Considerare le visite otorinolaringoiatriche dal 2010 al 2019, restituire, per ciascuna, matricola del medico, codice fiscale del paziente, data, e data della visita successiva del paziente con un medico della stessa specializzazione

Soluzione

```
1 | SELECT
2 |     V.Medico,
3 |     V.Paziente,
4 |     V.`Data`,
5 |     LEAD(V.`Data`, 1) OVER (
6 |                             PARTITION BY V.Paziente
7 |
8 |                             ORDER BY V.`Data`
9 |
10| FROM
11|     Visita V
12|     INNER JOIN
13|         Medico M ON V.Medico = M.Matricola
14| WHERE
15|     M.Specializzazione = 'Otorinolaringoiatria'
16|     AND YEAR(V.`Data`) BETWEEN 2010 AND 2019;
```

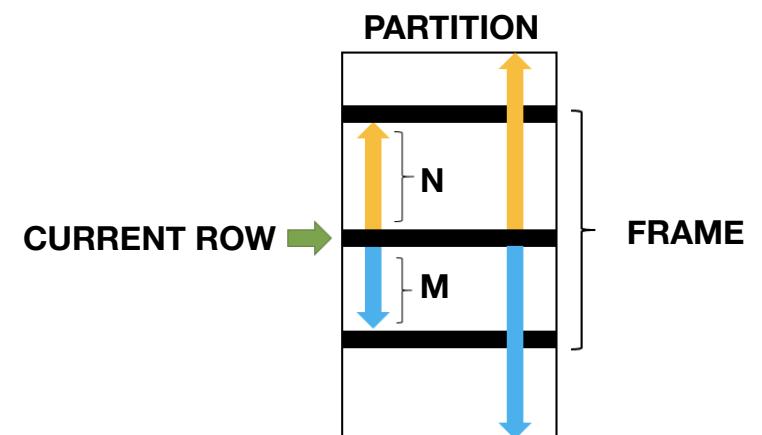
Frame



Frame

È un **sottoinsieme della partition dipendente dalla current row**, definito dal programmatore per formulare alcune query in modo rapido

Contiene N row precedenti ed M row successive alla current row, all'interno della partition.
 N ed M dipendono dal contesto



Window functions su frame

Processano un result set e calcolano valori sulla base di record
“adiacenti” alla current row



il contorno della current row è
definito riducendo la partition



Aggregate functions su frame

Se usate su frame, le aggregate functions calcolano il risultato **usando solo i record del frame**



Se il programmatore non definisce il frame, usano tutta la partition

Non-aggregate functions che lavorano SOLO su frame

- FIRST_VALUE()
- LAST_VALUE()

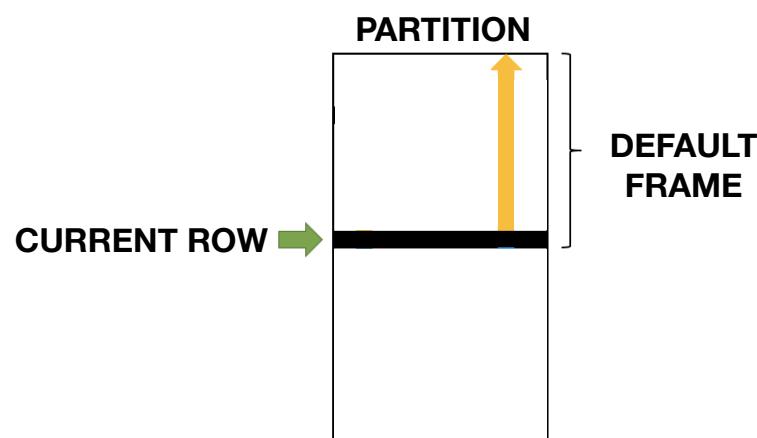
funzioni non-aggregate che
lavorano solo su frame

Usano i record del frame, che, se non viene definito dal programmatore,
è impostato automaticamente a un **default frame**

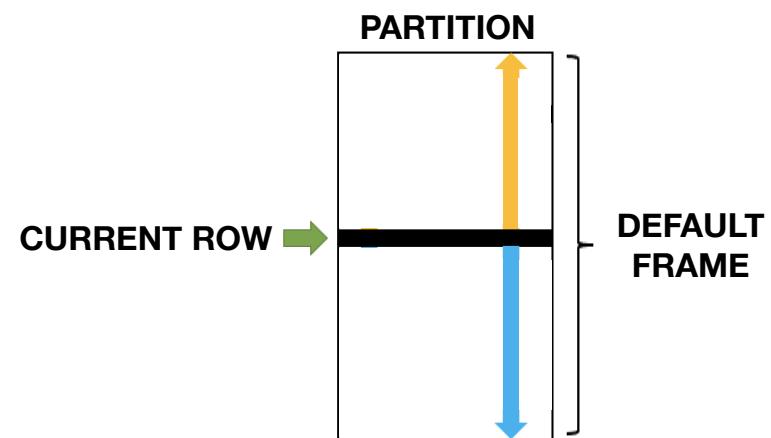
cambia dipendentemente dal
contenuto di over()

Default frame

OVER() con order by



OVER() senza order by



N.B. Il default frame viene automaticamente usato solo nelle non-aggregate function `first_value()` e `last_value()`, qualora il programmatore non definisca il frame

Default frame: da ricordare

In assenza di specifica
di frame, se over() contiene
order by, le window functions aggregate
e le non aggregate che lavorano su
frame usano i record dall'inizio della
partition fino alla current row. Se
over() non contiene order by, tali
funzioni considerano tutta la partition

Funzione FIRST_VALUE

Date le visite cardiologiche dei pazienti ‘aaa1’, ‘bbc4’ e ‘ccc2’ nel triennio 2012-2014, restituirne, per ciascuna, matricola del medico, codice fiscale del paziente, data, e data della prima visita effettuata dal paziente con quel medico

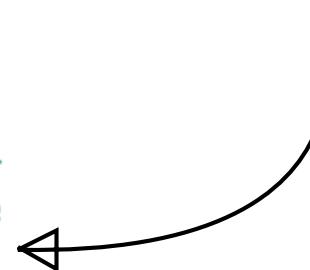
Soluzione

```

1 SELECT V.Medico,
2      V.Paziente,
3      V.`Data`,
4      FIRST_VALUE(V.`Data`) OVER w
5 FROM Visita V
6 WHERE V.Paziente IN ('aaa1','bbc4','ccc2')
7       AND YEAR(V.`Data`) BETWEEN 2012 AND 2014
8 ▼WINDOW w AS (PARTITION BY V.Medico, V.Paziente
9 ▲          ORDER BY V.`Data`)

```

Non importa specificare il frame,
il default frame va bene perché tanto ci
interessa il primo valore della partition!



 CURRENT ROW
 PARTITION
 DEFAULT FRAME

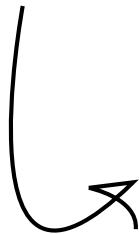
Medico	Paziente	Data	FIRST_VALUE(V.Data)
001	aaa1	2012-12-01	2012-12-01
001	aaa1	2013-03-01	2012-12-01
001	aaa1	2013-12-11	2012-12-01
001	aaa1	2014-01-23	2012-12-01
001	ccc2	2012-07-27	2012-07-27
007	bbc4	2012-09-25	2012-09-25
007	bbc4	2014-05-20	2012-09-25
007	ccc2	2012-01-04	2012-01-04
010	bbc4	2012-01-25	2012-01-25
010	ccc2	2012-06-27	2012-06-27
011	bbc4	2012-02-23	2012-02-23

Medico	Paziente	Data	FIRST_VALUE(V.Data)
001	aaa1	2012-12-01	2012-12-01
001	aaa1	2013-03-01	2012-12-01
001	aaa1	2013-12-11	2012-12-01
001	aaa1	2014-01-23	2012-12-01
001	ccc2	2012-07-27	2012-07-27
007	bbc4	2012-09-25	2012-09-25
007	bbc4	2014-05-20	2012-09-25
007	ccc2	2012-01-04	2012-01-04
010	bbc4	2012-01-25	2012-01-25
010	ccc2	2012-06-27	2012-06-27
011	bbc4	2012-02-23	2012-02-23

Medico	Paziente	Data	FIRST_VALUE(V.Data)
001	aaa1	2012-12-01	2012-12-01
001	aaa1	2013-03-01	2012-12-01
001	aaa1	2013-12-11	2012-12-01
001	aaa1	2014-01-23	2012-12-01
001	ccc2	2012-07-27	2012-07-27
007	bbc4	2012-09-25	2012-09-25
007	bbc4	2014-05-20	2012-09-25
007	ccc2	2012-01-04	2012-01-04
010	bbc4	2012-01-25	2012-01-25
010	ccc2	2012-06-27	2012-06-27
011	bbc4	2012-02-23	2012-02-23

Funzione LAST_VALUE

Considerate le visite cardiologiche dei pazienti ‘aaa1’, ‘bbc4’ e ‘ccc2’, nel triennio 2012-2014 e restituirne matricola del medico, codice del paziente, data, e data dell’ultima visita del paziente con quel medico nel triennio.



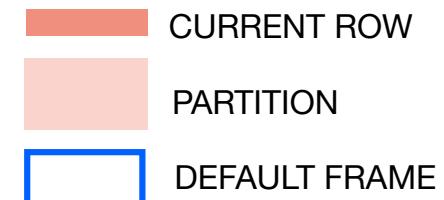
vogliamo la più recente, qui il default frame non va bene, perché si fermerebbe alla current row!

we must go on and on and on, in the partition

E se usassi LAST_VALUE col default frame?

```
1 SELECT V.Medico,  
2     V.Paziente,  
3     V.`Data`,  
4     LAST_VALUE(V.`Data`) OVER w  
5 FROM Visita V  
6 WHERE V.Paziente IN ('aaa1','bbc4','ccc2')  
7     AND YEAR(V.`Data`) BETWEEN 2012 AND 2014  
8 ▾WINDOW w AS (PARTITION BY V.Medico, V.Paziente  
9 ▾          ORDER BY V.`Data`)  
10 ORDER BY V.Paziente, V.Medico;
```

MA
ANCHE
No!!



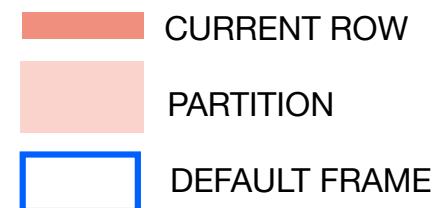
È questa l'ultima visita di aaa1 con 001!
Ed è la sua data (2014-01-23) che
dovrebbe essere affiancata a ogni record
della partition per 001 e aaa1!
Col default frame, mi fermo sempre alla
current row. Non va bene

Medico	Paziente	Data	FIRST_VALUE(V.Data)
001	aaa1	2012-12-01	2012-12-01
001	aaa1	2013-03-01	2013-03-01
001	aaa1	2013-12-11	2013-12-11
001	aaa1	2014-01-23	2014-01-23
001	ccc2	2012-07-27	2012-07-27
007	bbc4	2012-09-25	2012-09-25
007	bbc4	2014-05-20	2014-05-20
007	ccc2	2012-01-04	2012-01-04
010	bbc4	2012-01-25	2012-01-25
010	ccc2	2012-06-27	2012-06-27
011	bbc4	2012-02-23	2012-02-23

LAST_VALUE con definizione del frame

```
1 SELECT V.Medico,  
2     V.Paziente,  
3     V.`Data`,  
4     LAST_VALUE(V.`Data`) OVER w  
5 FROM Visita V  
6 WHERE V.Paziente IN ('aaa1', 'bbc4', 'ccc2')  
7     AND YEAR(V.`Data`) BETWEEN 2012 AND 2014  
8 WINDOW w AS (PARTITION BY V.Medico, V.Paziente  
9                 ORDER BY V.`Data`  
10                ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING)
```

il frame va dalla current row
fino alla fine della partition



Adesso la data dell'ultima visita di aaa1 con 001 (cioè 2014-01-23) viene affiancata a ogni visita che li coinvolge! Tutto torna anche per le altre coppie paziente-medico :-)

Medico	Paziente	Data	LAST_VALUE(V.Data)
001	aaa1	2012-12-01	2014-01-23
001	aaa1	2013-03-01	2014-01-23
001	aaa1	2013-12-11	2014-01-23
001	aaa1	2014-01-23	2014-01-23
001	ccc2	2012-07-27	2012-07-27
007	bbc4	2012-09-25	2014-05-20
007	bbc4	2014-05-20	2014-05-20
007	ccc2	2012-01-04	2012-01-04
010	bbc4	2012-01-25	2012-01-25
010	ccc2	2012-06-27	2012-06-27
011	bbc4	2012-02-23	2012-02-23

Moving average



Scrivere una funzione analytics che, per ogni terapia conclusa del paziente ‘ttw2’, restituisca il farmaco, la durata e la durata media rispetto alla terapia precedente e successiva con lo stesso farmaco

Soluzione

```
1 WITH durate AS
2 (
3     SELECT T.Farmaco,
4            T.DataInizioTerapia,
5            DATEDIFF(
6                T.DataFineTerapia,
7                T.DataInizioTerapia
8            ) AS Durata
9     FROM Terapia T
10    WHERE T.Paziente = 'ttw2'
11        AND T.DataFineTerapia IS NOT NULL
12 )
13     SELECT D.Farmaco,
14            D.Durata,
15            D.DataInizioTerapia,
16            AVG(D.Durata) OVER w
17     FROM durate D
18     WINDOW w AS (ORDER BY D.DataInizioTerapia
19             ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING);
```

Traduzione dal MySQL[LESE] by Olga...



WINDOW w AS (ORDER BY D.DataInizioTerapia
ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING);



dichiarazione di frame con una riga prima e una riga dopo
(in questo caso, il frame è centrato sulla current row)



considera il result set R della query, ordinato cronologicamente per DataInizioTerapia (dalle terapie più vecchie alle più recenti). Per ogni row r di R , crea una partition ordinando cronologicamente le terapie sulla base di DataInizioTerapia, e poi considera un frame f (sottoinsieme della partition) contenente r , la row precedente e la row successiva

$r \rightarrow f$

Medico	Paziente	Data	Mutuata
001	slq6	2003-06-03	0
001	slq6	2004-04-05	0
002	slq6	2005-01-16	0
002	slq6	2007-11-30	0
003	slq6	2009-12-03	0
003	slq6	2012-07-21	0
004	slq6	2007-02-23	0
005	slq6	2005-02-22	0
005	slq6	2009-11-21	0
006	slq6	2006-01-24	0
006	slq6	2009-07-18	0

Risultato

Farmaco	Durata	DataInizioTerapia	AVG(D.Durata) OVER w
Gaviscon	23	1994-12-03	23.0000
Seglor	23	1995-10-19	24.3333
Efferalgan	27	1997-11-03	34.3333
Inderal	53	1998-12-14	35.3333
Protargolo	26	1999-02-23	44.0000
Quait	53	2002-02-21	28.0000
EN	5	2004-02-10	28.0000
Gaviscon	26	2008-01-15	27.0000
Aprovel	50	2008-03-09	44.0000
Lobivon	56	2009-04-06	42.3333
Gaviscon	21	2009-10-20	38.5000

Altro esempio

Considerate le visite ortopediche di ogni paziente, scrivere una query analytics che restituisca codice fiscale del paziente, matricola del medico, la sua parcella, il numero di visite ortopediche effettuate fino a quel momento, e la spesa sostenuta dal paziente per tali visite



significa che la partition associata alla row r è composta dalle sole visite che precedono r

Soluzione

```
1  SELECT V.Paziente,
2      V.Medico,
3      M.Parcella,
4      COUNT(*) OVER w,
5      SUM(M.Parcella) OVER w
6  FROM
7      Visita V
8      INNER JOIN
9      Medico M ON V.Medico = M.Matricola
10 WHERE
11     M.Specializzazione = 'Ortopedia'
12 ▾WINDOW w AS (
13         PARTITION BY V.Paziente
14         ORDER BY V.`Data`
15         ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
16 ▾);
```



Il frame considera le row fino alla current row, andando all'indietro illimitatamente, finché ce ne sono

Risultato

Paziente	Medico	Parcella	COUNT(*) OVER w	SUM(M.Parcella)
aaa1	006	160	0	160
aaa1	007	180	1	340
aaa1	006	160	2	500
aaa1	007	180	3	680
bbc4	006	160	0	160
bbc4	006	160	1	320
bbc4	007	180	2	500
bbc4	007	180	3	680
bbe1	007	180	0	180
bbe1	007	180	1	360
bbe1	006	160	2	520
bbe1	006	160	3	680
ccc2	007	180	0	180
ccc2	006	160	1	340
ccc2	006	160	2	500
ccc2	007	180	3	680
ccc2	007	180	4	860
ddd6	006	160	0	160
ddd6	006	160	1	320
ddd6	007	180	2	500
ddd6	007	180	3	680
ddd6	007	180	4	860

$$500 = 160 + 180 + 160$$

row di esempio

frame

partition

Una variante con frame temporale

Considerate le visite ortopediche di ogni paziente, scrivere una query analytics che restituisca codice fiscale del paziente, matricola del medico, la sua parcella, il numero di visite ortopediche effettuate nei sei mesi precedenti, e la spesa sostenuta dal paziente per tali visite



la partition associata alla row t comprende le row relative alle visite fino a 6 mesi prima

Soluzione con dichiarazione di range

```
1 SELECT V.Paziente,
2       V.Medico,
3       M.Parcella,
4       COUNT(*) OVER w - 1,
5       SUM(M.Parcella) OVER w
6 FROM
7       Visita V
8       INNER JOIN
9       Medico M ON V.Medico = M.Matricola
10 WHERE
11     M.Specializzazione = 'Ortopedia'
12 ▶WINDOW w AS (
13     PARTITION BY V.Paziente
14     ORDER BY V.`Data`
15     RANGE BETWEEN
16         INTERVAL 6 MONTH PRECEDING
17         AND CURRENT ROW
18 ▷ );
```

con la keyword 'range' si
possono dichiarare frame
basati su intervalli di valori
numerici, date o timestamp

Ricapitolando

Paziente	Medico	Parcella	COUNT(*) OVER w	SUM(M.Parcella)
aaa1	006	160	0	160
aaa1	007	180	0	180
aaa1	006	160	0	160
aaa1	007	180	1	340
bbc4	006	160	0	160
bbc4	006	160	1	320
bbc4	007	180	0	180
bbc4	007	180	0	180
bbe1	007	180	0	180
bbe1	007	180	1	360
bbe1	006	160	0	160
bbe1	006	160	1	320
ccc2	007	180	0	180
ccc2	006	160	0	160
ccc2	006	160	0	160
ccc2	007	180	0	180
ccc2	007	180	0	180
ddd6	006	160	0	160
ddd6	006	160	0	160
ddd6	007	180	0	180
ddd6	007	180	0	180
ddd6	007	180	1	360

Se voglio escludere il mese attuale?

```
1 SELECT V.Paziente,
2       V.Medico,
3       M.Parcella,
4       COUNT(*) OVER w - 1,
5       SUM(M.Parcella) OVER w
6 FROM
7       Visita V
8       INNER JOIN
9       Medico M ON V.Medico = M.Matricola
10 WHERE
11     M.Specializzazione = 'Ortopedia'
12 ▶WINDOW w AS (
13     PARTITION BY V.Paziente
14     ORDER BY V.`Data`
15     RANGE BETWEEN
16         INTERVAL 6 MONTH PRECEDING
17         AND INTERVAL 1 MONTH PRECEDING
18 ▷ );
```

Attenzione a *rows* e *range*!

