

Basi di dati

Corso di laurea in Ingegneria Informatica
Scuola di Ingegneria - Università di Pisa

Oracle MySQL
A.A. 2016-2017

Ing. Francesco Pistolesi

Postdoctoral Researcher
Dipartimento di Ingegneria dell'Informazione
francesco.pistolesi@iet.unipi.it

Query optimization



Query optimization

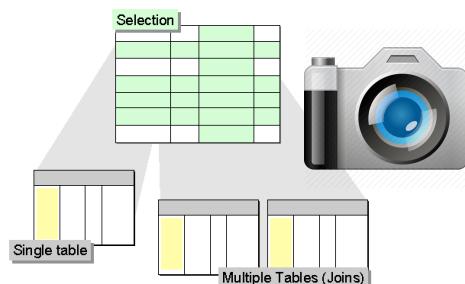


ridondanze



indici

Migliorare il livello di performance delle interrogazioni



materialized view

Performance



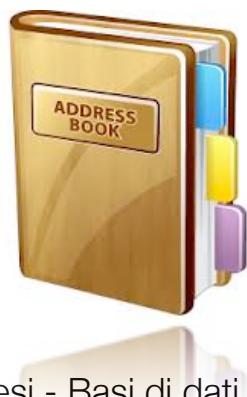
Una cosa alla volta



Indici

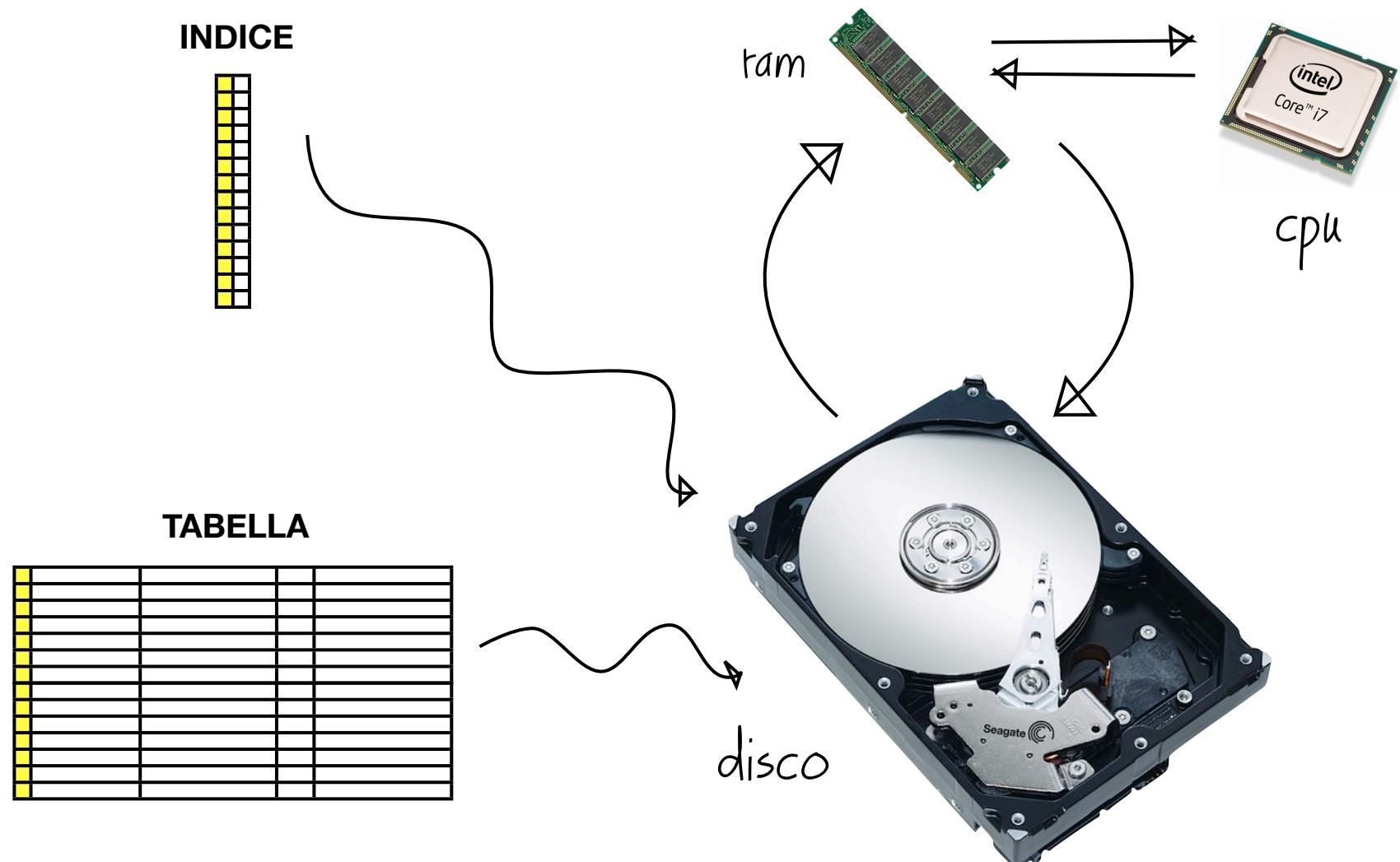
occupano poco spazio in memoria
spesso tree o hash table

Particolari **strutture dati** costruite per agevolare l'accesso ai record



funzionano come gli indici dei libri

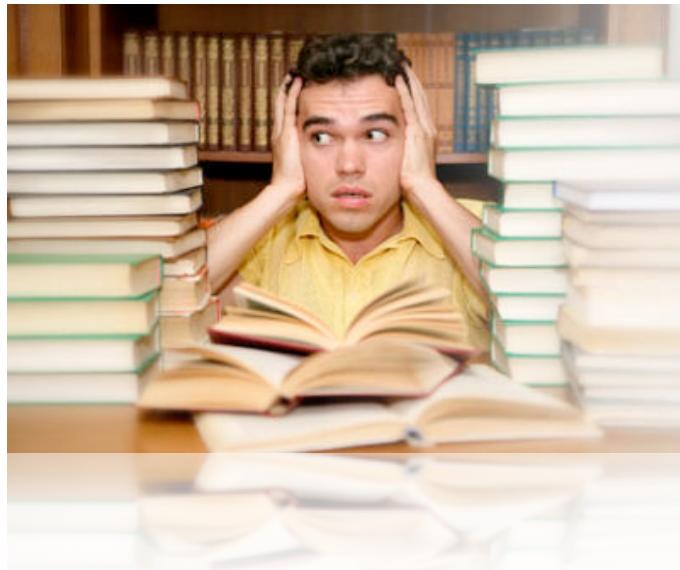
Indici e tavole



Ricerca

Trovare i paragrafi che trattano il query optimization in tutti i libri di
Basi di dati della biblioteca della scuola di Ingegneria di Pisa

senza indice

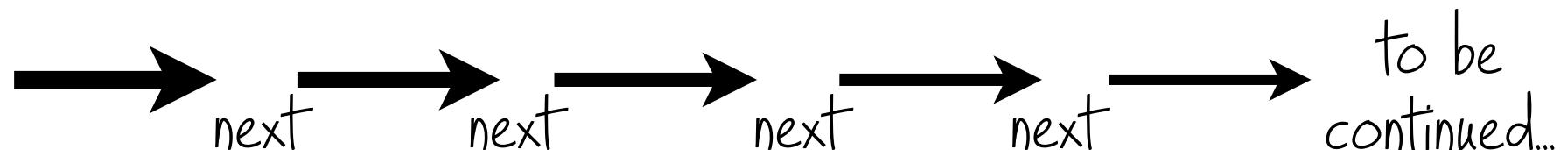


con indice



Full scan

Senza indici, occorre valutare una condizione **record per record**



Database di esempio

LIBRO

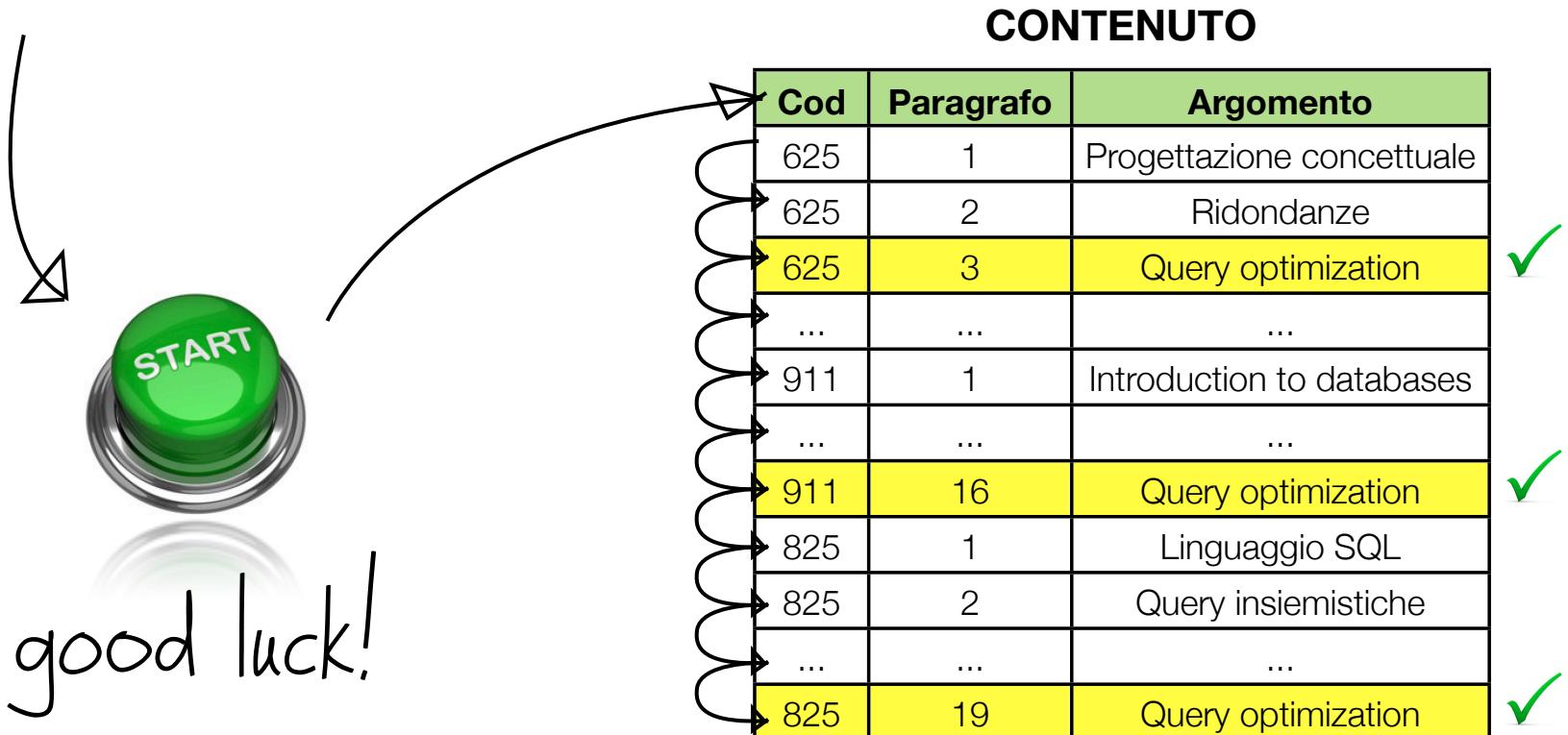
Cod	Autore	Titolo
625	Atzeni	Basi di dati
911	Silberschatz	Database system concepts
825	Paraboschi	Sistemi di basi di dati

CONTENUTO

Cod	Paragrafo	Argomento
625	1	Progettazione concettuale
625	2	Ridondanze
625	3	Query optimization
...
911	1	Introduction to databases
...
911	16	Query optimization
825	1	Linguaggio SQL
825	2	Query insiemistiche
...
825	19	Query optimization

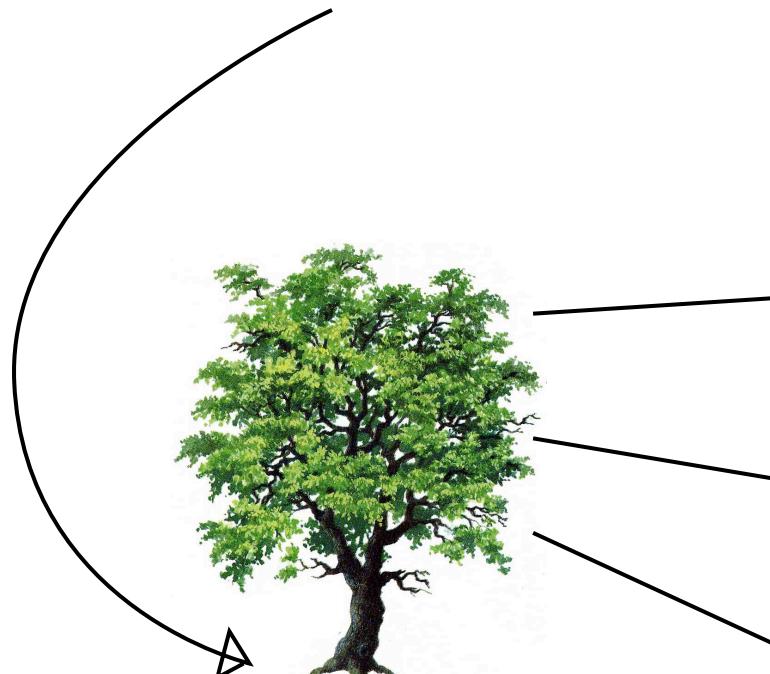
Ricerca senza indice

```
1 SELECT C.Cod, C.Paragrafo  
2 FROM Contenuto C  
3 WHERE C.Argomento = 'Query optimization'
```



Ricerca con indice

```
1 SELECT C.Cod, C.Paragrafo  
2 FROM Contenuto C  
3 WHERE C.Argomento = 'Query optimization'
```



it's a kind of magic

CONTENUTO

Cod	Paragrafo	Argomento
625	1	Progettazione concettuale
625	2	Ridondanze
625	3	Query optimization
...
911	1	Introduction to databases
...
911	16	Query optimization
825	1	Linguaggio SQL
825	2	Query insiemistiche
...
825	19	Query optimization

Tipologie di indici su MySQL

PRIMARY KEY

no ai valori null, no ai duplicati

UNIQUE

sì ai valori null, no ai duplicati

INDEX

sì ai valori null, sì ai duplicati

FULL TEXT

per ricerche nel testo

Query lente

è un semplice file di testo con
utente richiedente, data, ora, tempo di esecuzione...

Quelle che superano i 10 secondi di esecuzione sono visibili nello **slow-log**

valore default



Perché una query è lenta?...

è scritta male rispetto allo schema fisico dei dati



è eseguita durante il backup

è eseguita in momenti di sovraccarico

è eseguita mentre un'applicazione "spreme" il server

...oppure

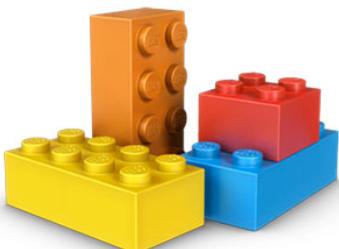
è lenta perché è complessa



Explain

è un modulo del DBMS che si occupa di ottimizzare il piano di esecuzione delle operazioni

Comando MySQL che, data una query, **consulta il query optimizer** e produce un result set avente un record per ogni tabella in gioco

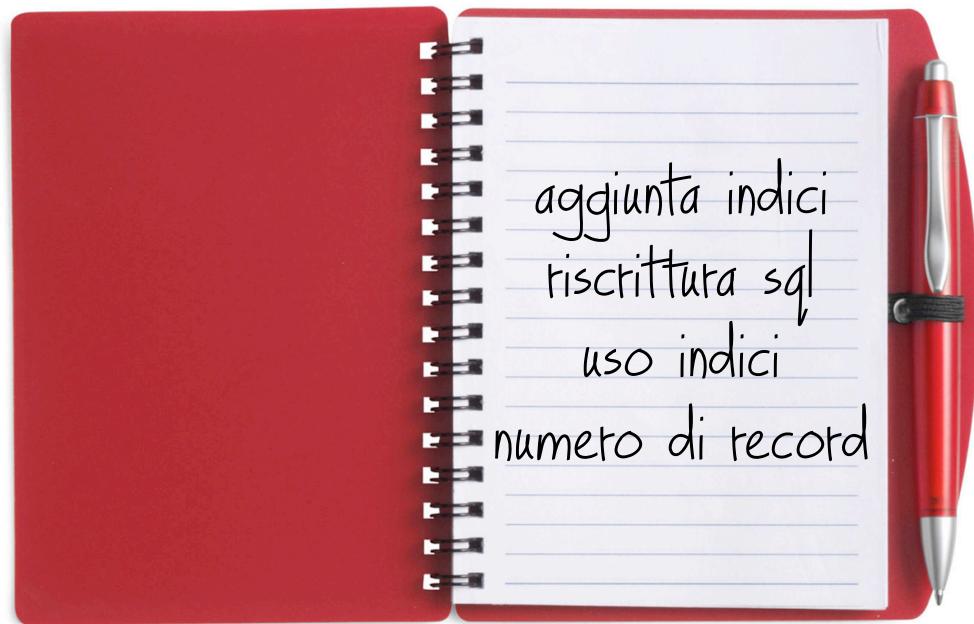


ci informa sul modo è "assemblata" e processata la query

Query optimizer



query optimizer



Esempio

Replicare le tabelle Medico, Paziente e Visita togliendo gli indici presenti. Ottimizzare una query che recupera nome e cognome del medico, nome e cognome del paziente, e la data di tutte le visite effettuate dopo il 10 Giugno 2005.

Replica tabelle senza indici

```
1 CREATE TABLE IF NOT EXISTS Medico2
2 SELECT * FROM Medico;
3
4 CREATE TABLE IF NOT EXISTS Paziente2
5 SELECT * FROM Paziente;
6
7 CREATE TABLE IF NOT EXISTS Visita2
8 SELECT * FROM Visita;
```

gli indici non sono copiati

Enjoy your query...

```
1 SELECT M2.Nome, M2.Cognome, P2.Nome, P2.Cognome, V2.`Data`  
2 FROM Medico2 M2 INNER JOIN Visita2 V2 ON M2.Matricola = V2.Medico  
3      INNER JOIN Paziente2 P2 ON V2.Paziente = P2.CodFiscale  
4 WHERE V2.`Data` > '2005-06-10'
```



12.4 sec elapsed

Explain...basta la parola

```
1 EXPLAIN SELECT M2.Nome, M2.Cognome, P2.Nome, P2.Cognome, V2.`Data`  
2 FROM Medico2 M2 INNER JOIN Visita2 V2 ON M2.Matricola = V2.Medico  
3           INNER JOIN Paziente2 P2 ON V2.Paziente = P2.CodFiscale  
4 WHERE V2.`Data` > '2005-06-10'
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	M2	ALL	[Null]	[Null]	[Null]	[Null]	18	
1	SIMPLE	P2	ALL	[Null]	[Null]	[Null]	[Null]	18	
1	SIMPLE	V2	ALL	[Null]	[Null]	[Null]	[Null]	5200	Using where

full scan

Significato dei campi

Column	Meaning
<u>id</u>	The SELECT identifier
<u>select_type</u>	The SELECT type
<u>table</u>	The table for the output row
<u>type</u>	The join type
<u>possible_keys</u>	The possible indexes to choose
<u>key</u>	The index actually chosen
<u>key_len</u>	The length of the chosen key
<u>ref</u>	The columns compared to the index
<u>rows</u>	Estimate of rows to be examined
<u>Extra</u>	Additional information

Inserimento indici per i join

```
1 ALTER TABLE Medico2 ADD INDEX(Matricola);  
2  
3 ALTER TABLE Visita2 ADD INDEX(Medico);
```



in Medico2 potrei usare anche "primary key"

Altro giro, altra corsa

```
1 EXPLAIN SELECT M2.Nome, M2.Cognome, P2.Nome, P2.Cognome, V2.`Data`  
2 FROM Medico2 M2 INNER JOIN Visita2 V2 ON M2.Matricola = V2.Medico  
3 INNER JOIN Paziente2 P2 ON V2.Paziente = P2.CodFiscale  
4 WHERE V2.`Data` > '2005-06-10'
```



questo indice non viene usato

questo è usato!

senza indice

id	select_type	table	type	possible keys	key	key_len	ref	rows	Extra
1	SIMPLE	M2	ALL	Matricola	[Null]	[Null]	[Null]	18	
1	SIMPLE	P2	ALL	[Null]	[Null]	[Null]	[Null]	18	
1	SIMPLE	V2	ref	Medico	Medico	50	salute2.M2.Matricola	27	Using where

Desideri...

secondo me l'esecuzione può
essere migliorata...



Creazione di indice sull'attributo *Data*

```
1 | ALTER TABLE Visita2 ADD INDEX(`Data`);
```

creando questo indice si velocizza la condizione where...

Riepilogo degli indici creati

```
1 SHOW INDEX  
2 FROM Visita2;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
visita2	1	Medico	1	Medico	A	25	[Null]	[Null]		BTREE
visita2	1	Data	1	Data	A	2670	[Null]	[Null]		BTREE

posizione della colonna nell'indice
(in un indice multi-column
specifica la posizione dell'attributo)

può contenere duplicati
(se vale 0 non può)

stima dei record univoci
(si aggiorna con 'analyze table')

ordine ascendente

tutto l'attributo è indicizzato
(l'indice può essere creato anche
su un sottoinsieme del valore dell'attributo)

cella vuota se l'attributo
non è nullable

Niente miglioramento...

```
1 EXPLAIN SELECT M2.Nome, M2.Cognome, P2.Nome, P2.Cognome, V2.`Data`  
2 FROM Visita2 V2 INNER JOIN Medico2 M2 ON M2.Matricola = V2.Medico  
3      INNER JOIN Paziente2 P2 ON V2.Paziente = P2.CodFiscale  
4 WHERE V2.`Data` > '2005-06-10'
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	M2	ALL	Matricola	[Null]	[Null]	[Null]	18	
1	SIMPLE	P2	ALL	[Null]	[Null]	[Null]	[Null]	18	
1	SIMPLE	V2	ref	Medico,Data	Medico	50	salute2.M2.Matricola	27	Using where

l'indice sulla data non viene utilizzato...ma perché?

Ricorda



Questione di convenienza

```
1 EXPLAIN SELECT M2.Nome, M2.Cognome, P2.Nome, P2.Cognome, V2.`Data`  
2 FROM Visita2 V2 INNER JOIN Medico2 M2 ON M2.Matricola = V2.Medico  
3 INNER JOIN Paziente2 P2 ON V2.Paziente = P2.CodFiscale  
4 WHERE V2.`Data` > '2005-06-10'
```

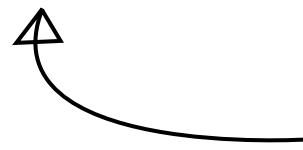
probabilmente poche visite antecedenti

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	M2	ALL	Matricola	[Null]	[Null]	[Null]	18	
1	SIMPLE	P2	ALL	[Null]	[Null]	[Null]	[Null]	18	
1	SIMPLE	V2	ref	Medico,Data	Medico	50	salute2.M2.Matricola	27	Using where

due indici mono-attributo

Cambio di condizione

```
1 EXPLAIN SELECT M2.Nome, M2.Cognome, P2.Nome, P2.Cognome, V2.`Data`  
2 FROM Visita2 V2 INNER JOIN Medico2 M2 ON M2.Matricola = V2.Medico  
3 INNER JOIN Paziente2 P2 ON V2.Paziente = P2.CodFiscale  
4 WHERE V2.`Data` > '2013-06-10'
```



condizione più restrittiva

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	V2	range	Medico,Data	Data	3	[Null]	2	Using where
1	SIMPLE	M2	ref	Matricola	Matricola	50	salute2.V2.Medico	2	
1	SIMPLE	P2	ALL	[Null]	[Null]	[Null]	[Null]	18	Using where

cambia l'ordine di processazione e l'indice su Data è usato

Straight join

Forza MySQL a eseguire il join **nell'ordine in cui si è scritto**

sempre con tecnica nested loop



col join classico, l'ordine varia in base alle scelte
dell'ottimizzatore, basate sulle statistiche

Let's try again!

```
1 EXPLAIN SELECT M2.Nome, M2.Cognome, P2.Nome, P2.Cognome, V2.`Data`  
2 FROM Visita2 V2 STRAIGHT_JOIN Medico2 M2 ON M2.Matricola = V2.Medico  
3 STRAIGHT_JOIN Paziente2 P2 ON V2.Paziente = P2.CodFiscale  
4 WHERE V2.`Data` > '2005-06-10'
```

ordine di esecuzione forzato

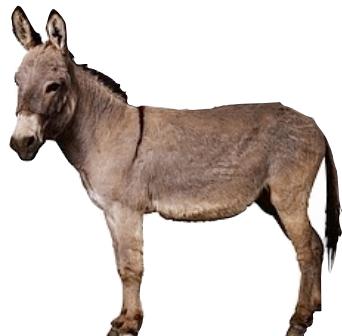
id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	V2	ALL	Medico,Data	[Null]	[Null]	[Null]	5200	Using where
1	SIMPLE	M2	ref	Matricola	Matricola	50	salute2.V2.Medico	2	
1	SIMPLE	P2	ALL	[Null]	[Null]	[Null]	[Null]	18	Using where



continua a preferire il full scan... masochismo?!

Indice forzato

È possibile forzare l'ottimizzatore a usare un **indice ben preciso**



l'ottimizzatore potrebbe rifiutarsi lo stesso di utilizzarlo

Indice forzato

```
1 EXPLAIN SELECT M2.Nome, M2.Cognome, P2.Nome, P2.Cognome, V2.`Data`  
2 FROM Visita2 V2 FORCE INDEX (`Data`) STRAIGHT_JOIN Medico2 M2 ON M2.Matricola = V2.Medico  
3 STRAIGHT_JOIN Paziente2 P2 ON V2.Paziente = P2.CodFiscale  
4 WHERE V2.`Data` > '2005-06-10'
```

senza usare l'indice

rows
5200

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	V2	range	Data	Data	3	[Null]	5097	Using where
1	SIMPLE	M2	ref	Matricola	Matricola	50	salute2.V2.Medico	2	
1	SIMPLE	P2	ALL	[Null]	[Null]	[Null]	[Null]	18	Using where

filtraggio molto basso
(per questo viene preferito un full scan)

Migliorare ancora?

...tornando all'esempio senza ordine forzato...

```
1 EXPLAIN SELECT M2.Nome, M2.Cognome, P2.Nome, P2.Cognome, V2.`Data`  
2 FROM Visita2 V2 INNER JOIN Medico2 M2 ON M2.Matricola = V2.Medico  
3       INNER JOIN Paziente2 P2 ON V2.Paziente = P2.CodFiscale  
4 WHERE V2.`Data` > '2005-06-10'
```

$18 \times 18 \times 27 = 8748$ record

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	M2	ALL	Matricola	[Null]	[Null]	[Null]	18	
1	SIMPLE	P2	ALL	[Null]	[Null]	[Null]	[Null]	18	
1	SIMPLE	V2	ref	Medico,Data	Medico	50	salute2.M2.Matricola	27	Using where

Paziente2 potrebbe beneficiare di un indice sull'attributo CodFiscale, evitando il full scan

Inserimento degli indici

```
1 ALTER TABLE Visita2 ADD INDEX(Paziente);  
2  
3 ALTER TABLE Paziente2 ADD INDEX(CodFiscale);
```

NB: creo anche l'indice su Visita2
perché devo effettuare un join

Explain again!

```
1 EXPLAIN SELECT M2.Nome, M2.Cognome, P2.Nome, P2.Cognome, V2.`Data`  
2 FROM Visita2 V2 INNER JOIN Medico2 M2 ON M2.Matricola = V2.Medico  
3 INNER JOIN Paziente2 P2 ON V2.Paziente = P2.CodFiscale  
4 WHERE V2.`Data` > '2005-06-10'
```

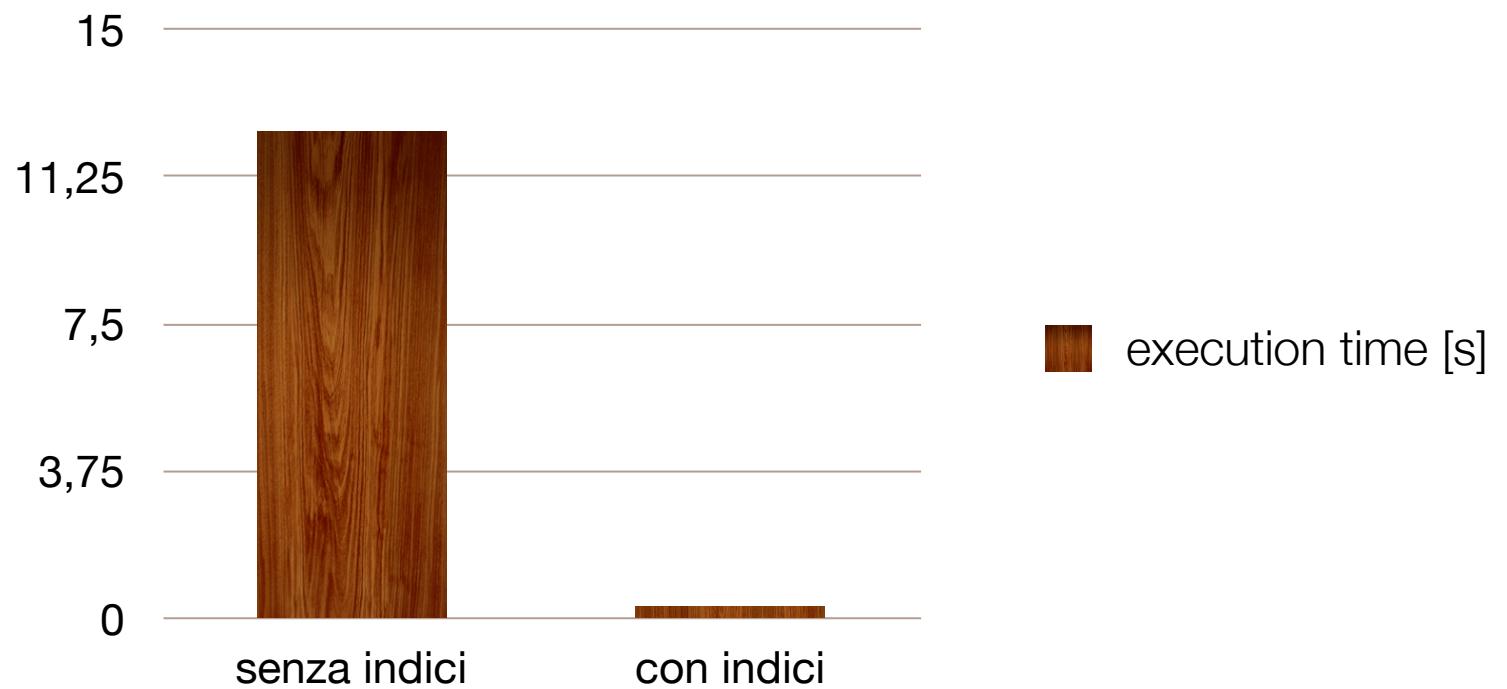
18X27=486 record

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	M2	ALL	Matricola	[Null]	[Null]	[Null]	18	
1	SIMPLE	V2	ref	Medico,Data,Paziente	Medico	50	salute2.M2.Matricola	27	Using where
1	SIMPLE	P2	ref	CodFiscale	CodFiscale	50	salute2.V2.Paziente	1	



0.03 sec elapsed

Miglioramento



Rimozione di un indice

modo 1

```
1 | ALTER TABLE nomeTabella  
2 | DROP INDEX nomeIndice;
```

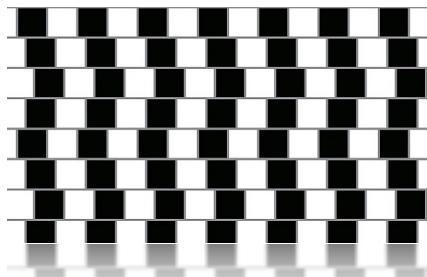
modo 2

```
1 | DROP INDEX nomeIndice ON nomeTabella;
```

l'indice è immediatamente distrutto

Refresh delle statistiche

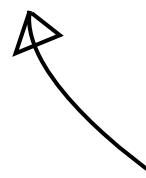
Se si effettuano operazioni di INSERT/UPDATE/DELETE le statistiche potrebbero divenire **molto obsolete**



necessità di refresh delle statistiche per evitare che l'ottimizzatore prenda degli abbagli

Refresh delle statistiche

```
1 ANALYZE TABLE Medico2;
```



aggiorna indici e le statistiche
da fare con cadenza giornaliera o comunque
dipendente dalla mole di aggiornamenti subiti dalla
tabella nel frattempo

Table	Op	Msg_type	Msg_text
salute2.medico2	analyze	status	OK

Refresh e ottimizzazione

```
1 OPTIMIZE TABLE Medico2;
```



aggiorna gli indici ed
effettua ottimizzazioni varie come
il compattamento su disco

Table	Op	Msg_type	Msg_text
salute2.medico2	optimize	status	OK



è più lento di analyze, e prende un lock di tabella

...e se l'esecuzione resta lenta?

Se una query complessa non trae vantaggio dagli indici, occorre adottare soluzioni diverse come quella della **replicazione**



materialized view e database dedicati
(si velocizzano le query lente scendendo a
comptomessi sui dati aggiornati)

