

Parte III, punto 9

(a)

1. Determina tutte le raccolte dati con ID maggiore di 5

```
1. SELECT *  
2. FROM RaccoltaDati  
3. WHERE RaccoltaDati.ID > 5;
```

2. Determina le classi presenti in una singola scuola.

```
1. SELECT Classe.Anno, Classe.Sezione  
2. FROM Scuola JOIN Classe ON Classe.codiceMecc = Scuola.CodiceMeccanografico;
```

3. Identifica tutti i dispositivi Arduino utilizzati all'interno di un orto con attività di fitobonifica e piantati in pieno campo.

```
1. SELECT ID  
2. FROM Dispositivo JOIN Orto ON Dispositivo.OrtoDoverisiede = Orto.Nome AND  
Dispositivo.ScuolaDiRiferimento = Orto.CodiceMecc  
3. WHERE Dispositivo.TipoDispositivo = 'Arduino' AND Orto.Attività = 'Fitobonifica' AND  
Orto.Tipo = 'In pieno campo';
```

(b)

1. Dato che il carico di lavoro si basa su una ricerca per intervalli, è stato deciso di utilizzare un indice primario clusterizzato di tipo ordinato su ID che sarà la nostra chiave di ricerca per la relazione RaccoltaDati. Questa scelta mira a facilitare la ricerca e, al contempo, ridurre gli accessi ai blocchi per migliorare le prestazioni grazie all'utilizzo della clusterizzazione.
2. Considerando che il carico di lavoro coinvolge operazioni di confronto, abbiamo deciso di utilizzare un indice clusterizzato primario di tipo hash sull'attributo 'CodiceMeccanografico' della tabella 'Scuola'. Questa decisione è stata presa considerando che non sono necessarie ricerche per intervalli di valori. L'uso di un indice hash garantisce prestazioni ottimali con accessi in $O(1)$ (a meno che non si verifichino collisioni), consentendo un miglioramento delle prestazioni senza duplicare i dati come avviene con un indice ordinato.
3. Considerando il carico di lavoro che coinvolge operazioni di confronto, abbiamo deciso di utilizzare un indice clusterizzato primario di tipo Ordinato sulla chiave composta {Nome, CodiceMecc} nella tabella 'Orto'. Questa scelta è stata fatta per migliorare le prestazioni delle operazioni di ricerca, poiché il carico di lavoro si concentra principalmente su questi due elementi. Sebbene l'utilizzo di un indice clusterizzato ordinato comporti costi aggiuntivi in termini di archiviazione, modifiche ed eliminazioni, le ricerche beneficeranno notevolmente di questa scelta al posto di un indice hash.

Queste informazioni sono state raccolte usando delle tabelle leggermente modificate: ad alcune delle tabelle dello schema, infatti, è stato aggiunto un campo 'Dummy' (char(1800) in modo da raggiungere una dimensione maggiore per ogni tupla e valutare meglio il carico di lavoro.


	Tuple inserite	Dimensione in blocchi
Scuola	150	600 kB
Classe	150	304 kB
RaccoltaDati	150	296 kB
Orto	150	304 kB
Dispositivo	150	304 kB

Parte III, punto 9.d

Per ogni query svolta, abbiamo riportato solo un piano di esecuzione scelto dal sistema in quanto non è cambiato dopo la creazione dello schema fisico.

Prima query

Piano di esecuzione





 ortiscolastici/postgres@PostgreSQL 9.6


Query Editor Query History

```
1 SELECT *
2 FROM RaccoltaDati
3 WHERE RaccoltaDati.ID > 5;
```

Data Output Explain Messages Notifications

Graphical Analysis Statistics


ortiscolastici.ra-
ccoltadati

Tempi senza indici

 ortiscolastici/postgres@PostgreSQL 9.6

Query Editor Query History


```
1 SELECT *
2 FROM RaccoltaDati
3 WHERE RaccoltaDati.ID > 5;
```

Data Output Explain Messages Notifications

Graphical Analysis Statistics

#	Node	Timings		Rows				Loops
		Exclusive	Inclusive	Rows X	Actual	Plan		
1.	→ Seq Scan on ortiscolastici.raccoltadati as raccoltadati (cost=0..38.88 rows=150 width=1878) (actu... Filter: (raccoltadati.id > 5) Rows Removed by Filter: 1	0.043 ms	0.043 ms	1 1.01	149	150		1

Tempi con indici (dopo un run)

 ortiscolastici/postgres@PostgreSQL 9.6

Query Editor Query History

```
1 SELECT *
2 FROM RaccoltaDati
3 WHERE RaccoltaDati.ID > 5;
```

Data Output Explain Messages Notifications

Graphical Analysis Statistics

#	Node	Timings		Rows				Loops
		Exclusive	Inclusive	Rows X	Actual	Plan		
1.	→ Seq Scan on ortiscolastici.raccoltadati as raccoltadati (cost=0..38.88 rows=150 width=1878) (actu... Filter: (raccoltadati.id > 5) Rows Removed by Filter: 1	0.096 ms	0.096 ms	1 1.01	149	150		1

Tempi con indici (dopo vari run)

ortiscolastici/postgres@PostgreSQL 9.6									
Query Editor Query History									
<pre> 1 SELECT * 2 FROM RaccoltaDat1 3 WHERE RaccoltaDat1.ID > 5; </pre>									
Data Output Explain Messages Notifications									
Graphical Analysis Statistics									
#	Node	Timings		Rows				Loops	
		Exclusive	Inclusive	Rows X	Actual	Plan			
1.	→ Seq Scan on ortiscolastici.raccoltadati as raccoltadati (cost=0.38..88 rows=150 width=1878) (actu... Filter: (raccoltadati.id > 5) Rows Removed by Filter: 1	0.034 ms	0.034 ms	1 1.01	149	150		1	

Seconda query

Piano di esecuzione

ortiscolastici/postgres@PostgreSQL 9.6									
Query Editor Query History									
<pre> 1 SELECT Classe.Anno, Classe.Sezione 2 FROM ortiscolastici.Scuola JOIN Classe ON Classe.codiceMecc = ortiscolastici.Scuola.CodiceMeccanografico; </pre>									
Data Output Explain Messages Notifications									
Graphical Analysis Statistics									

Tempi senza indici

ortiscolastici/postgres@PostgreSQL 9.6									
Query Editor Query History									
<pre> 1 SELECT Classe.Anno, Classe.Sezione 2 FROM ortiscolastici.Scuola JOIN Classe ON Classe.codiceMecc = Scuola.codiceMeccanografico; </pre>									
Data Output Explain Messages Notifications									
Graphical Analysis Statistics									
#	Node	Timings		Rows				Loops	
		Exclusive	Inclusive	Rows X	Actual	Plan			
1.	→ Hash Inner Join (cost=41.38..122 rows=150 width=8) (actual=0.171..0.26 rows=150 loops=1) Hash Cond: (scuola.codicemeccanografico = classe.codicemecc)	0.096 ms	0.26 ms	1 1	150	150		1	
2.	→ Seq Scan on ortiscolastici.scuola as scuola (cost=0..78 rows=300 width=11) (actual=0.01..0...	0.069 ms	0.069 ms	1 1	300	300		1	
3.	→ Hash (cost=39.5..39.5 rows=150 width=19) (actual=0.095..0.095 rows=150 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 16 kB	0.04 ms	0.095 ms	1 1	150	150		1	
4.	→ Seq Scan on ortiscolastici.classe as classe (cost=0..39.5 rows=150 width=19) (actual=0...	0.055 ms	0.055 ms	1 1	150	150		1	

Tempi con indici (dopo vari run)

ortiscolastici/postgres@PostgreSQL 9.6									
Query Editor Query History									
<pre> 1 SELECT Classe.Anno, Classe.Sezione 2 FROM ortiscolastici.Scuola JOIN Classe ON Classe.codiceMecc = ortiscolastici.Scuola.CodiceMeccanografico; </pre>									
Data Output Explain Messages Notifications									
Graphical Analysis Statistics									
#	Node	Timings		Rows				Loops	
		Exclusive	Inclusive	Rows X	Actual	Plan			
1.	→ Hash Inner Join (cost=41.38..122 rows=150 width=8) (actual=0.177..0.25 rows=150 loops=1) Hash Cond: (scuola.codicemeccanografico = classe.codicemecc)	0.076 ms	0.25 ms	1 1	150	150		1	
2.	→ Seq Scan on ortiscolastici.scuola as scuola (cost=0.78 rows=300 width=11) (actual=0.009..0.009 rows=300 loops=1)	0.088 ms	0.088 ms	1 1	300	300		1	
3.	→ Hash (cost=39.5..39.5 rows=150 width=19) (actual=0.087..0.087 rows=150 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 16 kB	0.022 ms	0.087 ms	1 1	150	150		1	
4.	→ Seq Scan on ortiscolastici.classe as classe (cost=0.39..5 rows=150 width=19) (actual=0.004..0.004 rows=150 loops=1)	0.065 ms	0.065 ms	1 1	150	150		1	

Terza query

Piano di esecuzione

ortiscolastici/postgres@PostgreSQL 9.6									
Query Editor Query History									
<pre> 1 SELECT ID 2 FROM Dispositivo JOIN Orto ON Dispositivo.OrtoDoverisiede = Orto.Nome AND Dispositivo.ScuolaDiRiferimento = Orto.CodiceMecc 3 WHERE Dispositivo.TipoDispositivo = 'Arduino' AND Orto.Attività = 'Fitobonifica' AND Orto.Tipo = 'In pieno campo'; </pre>									
Data Output Explain Messages Notifications									
Graphical Analysis Statistics									
<pre> graph LR A[ortiscolastici.orto] -- Hash --> B[Hash] C[ortiscolastici.dispositivo] -- Hash Inner Join --> D[Hash Inner Join] B -- Hash Inner Join --> D </pre>									

Tempi senza indici

ortiscolastici/postgres@PostgreSQL 9.6									
Query Editor Query History									
<pre> 1 SELECT ID 2 FROM Dispositivo JOIN Orto ON Dispositivo.OrtoDoverisiede = Orto.Nome AND Dispositivo.ScuolaDiRiferimento = Orto.CodiceMecc 3 WHERE Dispositivo.TipoDispositivo = 'Arduino' AND Orto.Attività = 'Fitobonifica' AND Orto.Tipo = 'In pieno campo'; </pre>									
Data Output Explain Messages Notifications									
Graphical Analysis Statistics									
#	Node	Timings		Rows				Loops	
		Exclusive	Inclusive	Rows X	Actual	Plan			
1.	→ Hash Inner Join (cost=40.85..81.38 rows=17 width=4) (actual=0.077..0.115 rows=24 loops=1) Hash Cond: (((dispositivo.ortodoverisiede)::text = (orto.nome)::text) AND (dispositivo.scuoladiriferimento = orto.codicemecc))	0.02 ms	0.115 ms	1 1.42		24	17	1	
2.	→ Seq Scan on ortiscolastici.dispositivo as dispositivo (cost=0.39..88 rows=64 width=25) (actual=0.004..0.004 rows=64 loops=1) Filter: ((dispositivo.tipodispositivo)::text = 'Arduino'::text) Rows Removed by Filter: 86	0.038 ms	0.038 ms	1 1		64	64	1	
3.	→ Hash (cost=40.25..40.25 rows=40 width=24) (actual=0.058..0.058 rows=46 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 11 kB	0.01 ms	0.058 ms	1 1.15		46	40	1	
4.	→ Seq Scan on ortiscolastici.orto as orto (cost=0.40..25 rows=40 width=24) (actual=0.004..0.004 rows=40 loops=1) Filter: (((orto.attività)::text = 'Fitobonifica'::text) AND ((orto.tipo)::text = 'In pieno campo'::text)) Rows Removed by Filter: 104	0.049 ms	0.049 ms	1 1.15		46	40	1	

Tempi con indici (dopo 4 run)

ortiscolastici/postgres@PostgreSQL 9.6									
Query Editor Query History									
<pre>1 SELECT ID 2 FROM Dispositivo JOIN Orto ON Dispositivo.OrtoDoverIsiede = Orto.Nome AND Dispositivo.ScuolaDiRiferimento = Orto.CodiceMecc 3 WHERE Dispositivo.TipoDispositivo = 'Arduino' AND Orto.Attività = 'Fitobonifica' AND Orto.Tipo = 'In pieno campo'; 4</pre>									
Data Output Explain Messages Notifications									
Graphical Analysis Statistics									
#	Node	Timings		Rows				Loops	
		Exclusive	Inclusive	Rows X	Actual	Plan			
1.	→ Hash Inner Join (cost=40.85..81.38 rows=17 width=4) (actual=0.067..0.103 rows=24 loops=1) Hash Cond: (((dispositivo.ortodoverisiede)::text = (orto.nome)::text) AND (dispositivo.scuoladiriferimento = orto.codicemecc))	0.015 ms	0.103 ms	↓ 1.42	24	17		1	
2.	→ Seq Scan on ortiscolastici.dispositivo as dispositivo (cost=0..39.88 rows=64 width=25) (actual=0.034..0.034 rows=64 loops=1) Filter: ((dispositivo.tipodispositivo)::text = 'Arduino'::text) Rows Removed by Filter: 86	0.034 ms	0.034 ms	↑ 1	64	64		1	
3.	→ Hash (cost=40.25..40.25 rows=40 width=24) (actual=0.054..0.054 rows=46 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 11 kB	0.008 ms	0.054 ms	↓ 1.15	46	40		1	
4.	→ Seq Scan on ortiscolastici.orto as orto (cost=0..40.25 rows=40 width=24) (actual=0.004..0.004 rows=40 loops=1) Filter: (((orto.attività)::text = 'Fitobonifica'::text) AND ((orto.tipo)::text = 'In pieno campo'::text)) Rows Removed by Filter: 104	0.046 ms	0.046 ms	↓ 1.15	46	40		1	

Tempi con indici (dopo 4 ulteriori run)

ortiscolastici/postgres@PostgreSQL 9.6									
Query Editor Query History									
<pre>1 SELECT ID 2 FROM Dispositivo JOIN Orto ON Dispositivo.OrtoDoverIsiede = Orto.Nome AND Dispositivo.ScuolaDiRiferimento = Orto.CodiceMecc 3 WHERE Dispositivo.TipoDispositivo = 'Arduino' AND Orto.Attività = 'Fitobonifica' AND Orto.Tipo = 'In pieno campo'; 4</pre>									
Data Output Explain Messages Notifications									
Graphical Analysis Statistics									
#	Node	Timings		Rows				Loops	
		Exclusive	Inclusive	Rows X	Actual	Plan			
1.	→ Hash Inner Join (cost=40.85..81.38 rows=17 width=4) (actual=0.06..0.095 rows=24 loops=1) Hash Cond: (((dispositivo.ortodoverisiede)::text = (orto.nome)::text) AND (dispositivo.scuoladiriferimento = orto.codicemecc))	0.017 ms	0.095 ms	↓ 1.42	24	17		1	
2.	→ Seq Scan on ortiscolastici.dispositivo as dispositivo (cost=0..39.88 rows=64 width=25) (actual=0.031..0.031 rows=64 loops=1) Filter: ((dispositivo.tipodispositivo)::text = 'Arduino'::text) Rows Removed by Filter: 86	0.031 ms	0.031 ms	↑ 1	64	64		1	
3.	→ Hash (cost=40.25..40.25 rows=40 width=24) (actual=0.046..0.047 rows=46 loops=1) Buckets: 1024 Batches: 1 Memory Usage: 11 kB	0.01 ms	0.047 ms	↓ 1.15	46	40		1	
4.	→ Seq Scan on ortiscolastici.orto as orto (cost=0..40.25 rows=40 width=24) (actual=0.003..0.003 rows=40 loops=1) Filter: (((orto.attività)::text = 'Fitobonifica'::text) AND ((orto.tipo)::text = 'In pieno campo'::text)) Rows Removed by Filter: 104	0.038 ms	0.038 ms	↓ 1.15	46	40		1	

Considerazioni finali

Abbiamo notato un pattern con tutti gli indici, quindi indipendentemente dall'uso di indici ordinati e indici hash: subito dopo la creazione dell'indice, la prima esecuzione della query richiede molto tempo, ma questa quantità va scemando con i run successivi. L'interpretazione che abbiamo dato è la seguente: la prima volta che eseguiamo la query dopo la creazione dell'indice, il sistema carica le tuple nella cache, quindi impiega molto tempo. Nei run successivi, le tuple non devono essere caricate, ma semplicemente consultate, quindi impiega meno tempo.

	nomeutente5 (GestoreGlobale Progetto)	nomeutente4 (ReferenteIstituto)	nomeutente3 (ReferenteScuola)	nomeutente2 (Insegnante)	nomeutente1 (Studente)
Scuola	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	/	/	/
Classe	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	/	/
Persona	SELECT, INSERT, DELETE, UPDATE	/	/	/	/
Orto	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	/
Pianta	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	/
Specie	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	/
Gruppo	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	/
RaccoltaDati	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE
Dispositivo	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE	SELECT, INSERT, DELETE, UPDATE