Il comportamento dell'ottimizzatore: esempi e prestazioni

Antonio Albano Leonardo Candela
Dipartimento di Informatica - Università di Pisa

20 Settembre 2001

Sommario

Si presentano alcune interrogazioni con i rispettivi piani di accesso generati dal sistema per mostrare alcune caratteristiche dell'ottimizzatore come, ad esempio, l'eliminazione di operatori "inutili" dal piano di accesso prodotto, e come il risultato prodotto dall'ottimizzatore cambi al variare della strategia di ricerca e dello spazio degli alberi esplorato. Si mostrano infine i risultati di alcuni test per confrontare le prestazioni dell'ottimizzatore con quelle della versione precedente usata dal sistema JRS. I tempi di esecuzione si riferiscono ad una macchina dotata di processore Intel Pentium II a 400 MHz con 128 Mb di RAM e sistema operativo Linux.

1 La base di dati di riferimento

Lo schema della base di dati utilizzata per le interrogazioni mostrate in seguito è riportato in Tabella 1. Sono inoltre validi i parametri quantitativi riportati in Tabella 2.

2 Interrogazioni con una relazione

Negli esempi riportati di seguito e nelle sezioni succesive i piani di accesso mostrati sono ottenuti esplorando lo spazio di ricerca degli alberi profondi a sinistra con la strategia di ricerca *greedy*.

Interrogazione 2.1

Si mostra il comportamento dell'ottimizzatore per eseguire una semplice interrogazione su una relazione con restrizioni.

Relazione	Campi		
Progetti	Numero	INTEGER	
	Codice	VARCHAR(10)	NOT NULL
	GestitoDa	INTEGER	
ImpiegatiProgetti	Impiegato	INTEGER	NOT NULL
	Progetto	INTEGER	NOT NULL
	Impegno	INTEGER	
Impiegati	Codice	INTEGER	NOT NULL
	Cognome	VARCHAR(15)	NOT NULL
	Nome	VARCHAR(15)	NOT NULL
	AnnoNascita	INTEGER	NOT NULL
	Sesso	VARCHAR(1)	NOT NULL
	Dipartimento	INTEGER	NOT NULL
	Qualifica	VARCHAR(1)	NOT NULL
	Stipendio	INTEGER	NOT NULL
	Supervisore	INTEGER	
Dipartimenti	Numero	INTEGER	NOT NULL
	Nome	VARCHAR(15)	NOT NULL
	Citta	VARCHAR(10)	NOT NULL
	DirettoDa	INTEGER	
Familiari	Nome	VARCHAR(15)	NOT NULL
	CapoFamiglia	INTEGER	NOT NULL
	AnnoNascita	INTEGER	
	Parentela	VARCHAR(8)	NOT NULL
	Sesso	VARCHAR(1)	NOT NULL

Tabella 1: Schema della base di dati.

```
SELECT *
FROM   Impiegati
WHERE  Codice < 10
  AND  AnnoNascita > 1970;
```

In Figura 1 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	1369 *ms* $(min : 0, s : 1, ms : 369)$
`Result cardinality`	3 *Record*
`Cost`	2 *Logical Reads*

Si noti come la presenza di opportuni indici, nel caso particolare dell'indice `PIm`, permette la generazione di piani di accesso che ne fanno uso per l'esecuzione di restrizioni evitando di effettuare una scansione della relazione. Inoltre si noti l'assenza, nel piano di accesso, dell'operatore fisico di proiezione che in questo caso risulta essere superfluo.

Relazione	N_{pag}	N_{reg}	Indici	N_{leaf}	N_{key}
Progetti	2	40	*PPr* non di ordinamento attributi: Codice asc	4	40
			FGeDa non di ordinamento attributi: GestitoDa asc	2	11
ImpiegatiProgetti	13	477	*PImPr* non di ordinamento attributi: Impiegato asc, Progetto asc	29	477
Impiegati	16	169	*PIm* di ordinamento attributi: Codice asc	12	169
			FDi non di ordinamento attributi: Dipartimento asc	9	19
			FSu non di ordinamento attributi: Supervisore asc	12	26
Dipartimenti	1	20	*PDi* di ordinamento attributi: Numero asc	1	20
			uks0Di non di ordinamento attributi: Nome asc, Citta asc	2	20
			FDiDa non di ordinamento attributi: DirettoDa asc	1	20
Familiari	30	491	*PFa* non di ordinamento attributi: CapoFamiglia asc, Nome asc	36	491
			FCaFa non di ordinamento attributi: CapoFamiglia asc	25	167

Tabella 2: Parametri quantitativi di tabelle e indice della base di dati.

3 Interrogazioni con più relazioni

Si mostra il comportamento dell'ottimizzatore per eseguire interrogazione su più relazioni con giunzioni e restrizioni.

Interrogazione 3.1

```
SELECT   i.Cognome, i.Dipartimento, d.Nome
FROM     Impiegati i, Dipartimenti d
WHERE    i.Nome = 'Alessandro'
  AND    d.Citta = 'Pisa'
  AND    i.Dipartimento = d.Numero;
```

In Figura 2 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	5434 *ms* $(min : 0, s : 5, ms : 434)$
`Result cardinality`	2 *Record*
`Cost`	17 *Logical Reads*

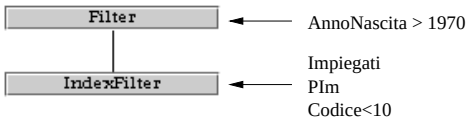

Figura 1: Piano di accesso dell'interrogazione 2.1.

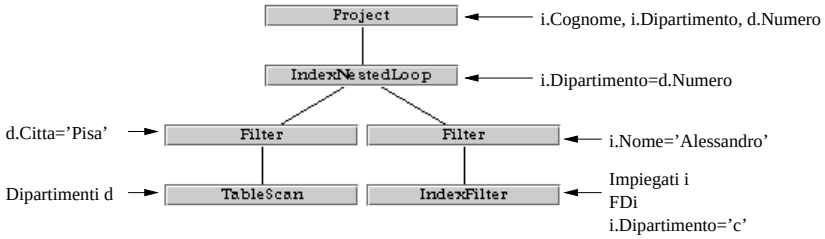

Figura 2: Piano di accesso dell'interrogazione 3.1.

Si noti come le due restrizione presenti sulle tabelle vengono anticipate rispetto l'operazione di giunzione, che viene eseguita con l'algoritmo *index nested loop* per la presenza dell'indice `FDi` sull'attributo `Dipartimento` della relazione `Impiegati`.

Interrogazione 3.2

La prossima interrogazione mostra il comportamento dell'ottimizzatore in presenza della clausola di ordinamento.

```
SELECT   i.Cognome, i.Dipartimento, d.Nome
FROM     Impiegati i, Dipartimenti d
WHERE    i.Dipartimento = d.Numero
ORDER BY I.Dipartimento;
```

In Figura 3 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	9484 ms $(min : 0, s : 9, ms : 484)$
`Result cardinality`	169 *Record*
`Cost`	101 *Logical Reads*

Si noti come in questo caso la giunzione venga eseguita usando l'algoritmo *page nested loop*.

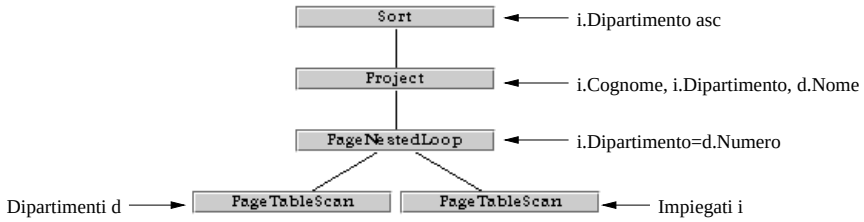

Figura 3: Piano di accesso dell'interrogazione 3.2 con strategia di ricerca *greedy*.

3.1 La strategia di ricerca *greedy*, di solito, non produce il piano di costo minimo

La strategia di ricerca *greedy* ha bassi tempi di ottimizzazione ma di solito non produce il piano di accesso di costo minimo che si ottiene usando la strategia di costo uniforme.

In Figura 4 è riportato il piano di accesso prodotto dall'ottimizzatore per l'interrogazione 3.2 usando la strategia di ricerca di costo uniforme.

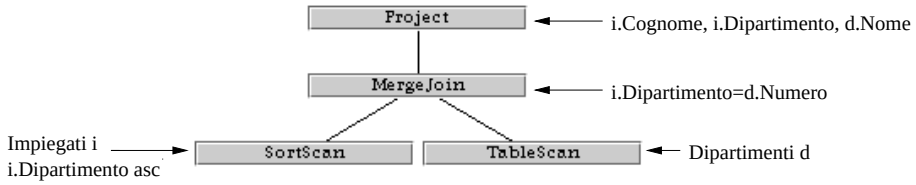

Figura 4: Piano di accesso dell'interrogazione 3.2 con strategia di ricerca di costo uniforme.

Il piano presenta le seguenti caratteristiche:

`Time`	17 195 ms $(min : 0, s : 17, ms : 195)$
`Result cardinality`	169 *Record*
`Cost`	81 *Logical Reads*

Si noti come la strategia di ricerca di costo uniforme produca un piano migliore ma con un tempo di ottimizzazione più alto.

Con il prossimo esempio si mostra il comportamento dell'ottimizzatore

con tre tipi di strategia di ricerca: *greedy*, costo uniforme limitata e costo uniforme.

Interrogazione 3.3

```
SELECT   D.Nome, D.Numero, I.Dipartimento, I.Cognome, I.Codice,
         DA.Nome, IP.Progetto, IP.Impiegato
FROM     Impiegati I, Dipartimenti D, Progetti P, ImpiegatiProgetti IP,
         Dipartimenti DA
WHERE    I.Dipartimento <> D.Numero
  AND    D.Numero = P.GestitoDa
  AND    P.Numero = IP.Progetto
  AND    IP.Impiegato = I.Codice
  AND    I.Dipartimento = DA.Numero;
```

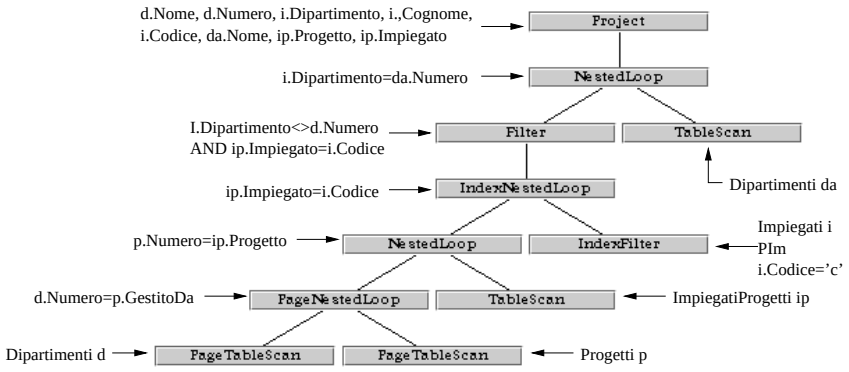

Figura 5: Piano di accesso dell'interrogazione 3.3 con strategia di ricerca *greedy*.

In Figura 5 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	$14\,227\,ms$ $(min:0, s:14, ms:227)$
`Result cardinality`	27 *Record*
`Cost`	1123 *Logical Reads*

In Figura 6 è riportato il piano di accesso prodotto dall'ottimizzatore usando la strategia di ricerca di costo uniforme limitata. Il piano presenta le seguenti caratteristiche:

`Time`	$60\,635$ ms $(min:1, s:0, ms:635)$
`Result cardinality`	28 *Record*
`Cost`	149 *Logical Reads*

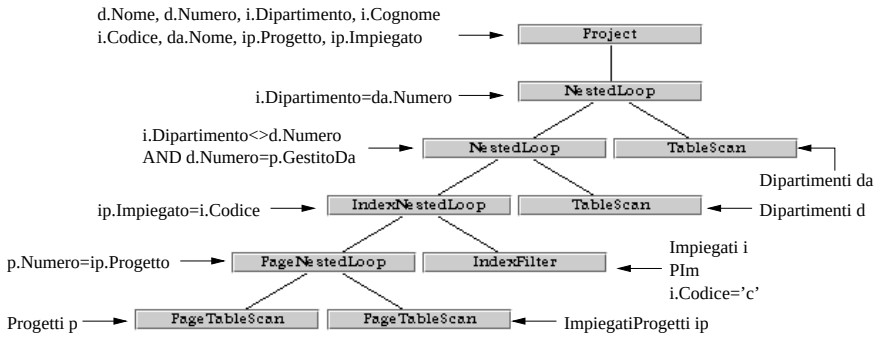

Figura 6: Piano di accesso dell'interrogazione 3.3 con strategia di ricerca di costo uniforme limitata.

Si noti come la strategia di ricerca di costo uniforme limitata ha prodotto un piano di costo inferiore impiegando però maggior tempo.

In Figura 7 è riportato il piano di accesso prodotto dall'ottimizzatore per l'interrogazione 3.3 usando la strategia di ricerca di costo uniforme.

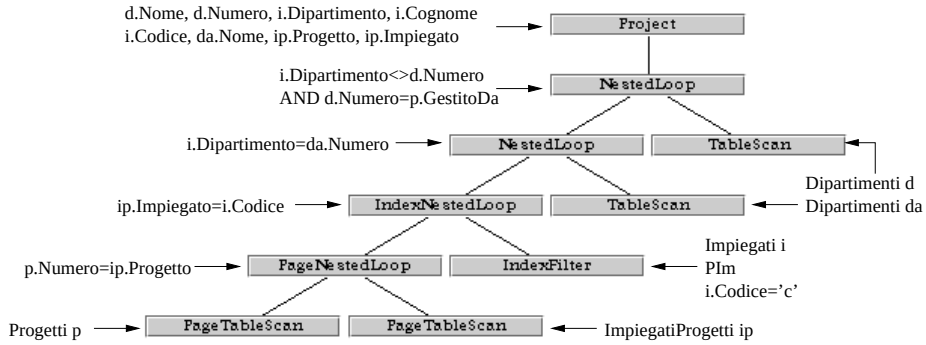

Figura 7: Piano di accesso dell'interrogazione 3.3 con strategia di ricerca di costo uniforme.

Tale piano presenta le seguenti caratteristiche:

`Time`	60 963 ms $(min : 1, s : 0, ms : 963)$
`Result cardinality`	28 *Record*
`Cost`	139 *Logical Reads*

Si noti come la strategia di ricerca di costo uniforme ha prodotto un piano

diverso e di costo inferiore sia rispetto a quello in Figura 5 prodotto con strategia *greedy*, sia rispetto a quello in Figura 6 prodotto con strategia di costo uniforme limitata, impiegando però maggior tempo. Questa osservazione è vera in generale, la strategia di ricerca di costo uniforme produce dei piani di accesso "migliori" rispetto alle altre due strategie a scapito di un maggior tempo di ottimizzazione.

3.2 Non sempre la soluzione migliore si trova nello spazio degli alberi profondi a sinistra

Negli esempi mostrati sino ad ora lo spazio di ricerca esplorato è stato sempre ridotto a quello degli alberi profondi a sinistra.

La scelta di limitare la ricerca allo spazio degli alberi profondi a sinistra è giustificata dal fatto che questa aumenta lo spazio dei possibili algoritmi per implementare l'operatore di giunzione. Ad esempio se l'operando interno di una giunzione fosse il risultato di un'altra giunzione non si potrebbe usare nessun indice per accedervi e quindi sarebbero esclusi dalla ricerca algoritmi che implementano la giunzione in modo efficiente usando indici come l'*index nested loop*.

Purtroppo la soluzione migliore non appartiene sempre allo spazio degli alberi profondi a sinistra, ma la si trova cercando nello spazio degli alberi generali.

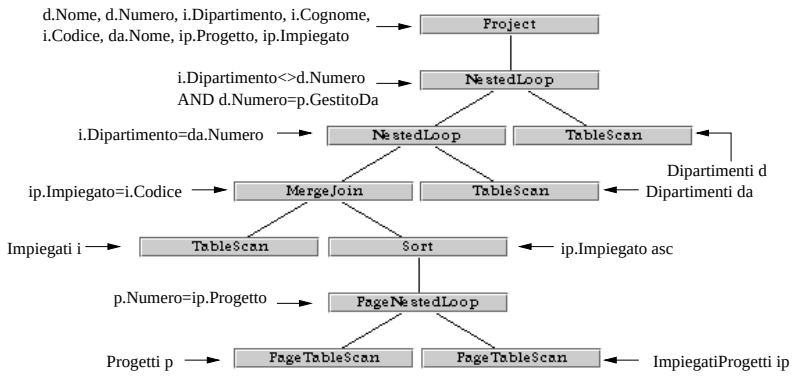

Figura 8: Piano di accesso dell'interrogazione 3.3 con strategia di ricerca di costo uniforme e esplorando lo spazio degli alberi generali.

In Figura 8 è riportato il piano di accesso prodotto dall'ottimizzato-

re per l'interrogazione 3.3 usando la strategia di ricerca di costo uniforme e esplorando lo spazio degli alberi generali. Il piano presenta le seguenti caratteristiche:

`Time`	66 757 ms $(min : 1, s : 6, ms : 757)$
`Result cardinality`	28 *Record*
`Cost`	105 *Logical Reads*

Le differenze tra il miglior piano dello spazio degli alberi profondi a sinistra e il miglior piano dello spazio degli alberi generali possono essere anche più evidenti come nell'interrogazione successiva.

Interrogazione 3.4

```
SELECT  *
FROM    Dipartimenti a, Dipartimenti b, Dipartimenti c, Dipartimenti d;
```

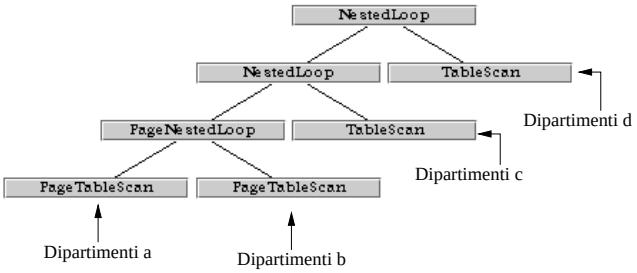

Figura 9: Piano di accesso dell'interrogazione 3.4 con strategia di ricerca di costo uniforme e esplorando lo spazio degli alberi profondi a sinistra.

In Figura 9 è riportato il piano di accesso prodotto dall'ottimizzatore usando la ricerca di costo uniforme con alberi profondi a sinistra. Il piano ha le seguenti caratteristiche:

`Time`	4084 ms $(min : 0, s : 4, ms : 84)$
`Result cardinality`	160 000 *Record*
`Cost`	8402 *Logical Reads*

In Figura 10 è riportato il piano di accesso prodotto dall'ottimizzatore usando ancora la strategia di ricerca di costo uniforme ma con alberi generali.

Il piano presenta le seguenti caratteristiche:

`Time`	4157 ms $(min : 0, s : 4, ms : 157)$
`Result cardinality`	160 000 *Record*
`Cost`	802 *Logical Reads*

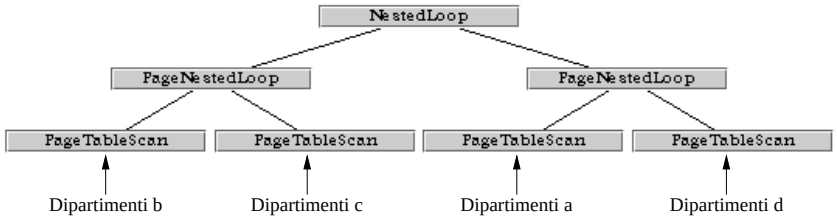

Figura 10: Piano di accesso dell'interrogazione 3.4 con strategia di ricerca di costo uniforme e esplorando lo spazio degli alberi generali.

Si noti come il costo del piano in Figura 10 sia minore di un fattore 10 rispetto a quello del piano in Figura 9.

4 Ordinamenti e raggruppamenti

Gli operatori fisici del JRS per l'eliminazione dei duplicati e per trattare la clausola `GROUP BY` sono realizzati con la tecnica dell'ordinamento ovvero si richiede che i dati su cui questi operatori lavorano siano ordinati sugli attributi di proiezione o di raggruppamento. Inoltre l'utente del sistema può imporre un ordinamento nei dati usando la clausola `ORDER BY`.

Negli esempi seguenti si mostra come l'ottimizzatore ha cura di produrre dei piani di accesso che tengono conto di questa caratteristica inserendo degli opportuni operatori di ordinamento nei piani di accesso prodotti. Successivamente si mostrano alcuni casi in cui gli operatori di ordinamento che andrebbero inseriti risultano superflui e quindi l'ottimizzatore li elimina.

Interrogazione 4.1

```
SELECT   DISTINCT Nome
FROM     Impiegati;
```

In Figura 11 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	278 ms $(min : 0, s : 0, ms : 278)$
`Result cardinality`	169 *Record*
`Cost`	52 *Logical Reads*

Si noti come la presenza della clausola `DISTINCT` porti all'inserimento di un opportuno operatore di ordinamento nel piano di accesso prodotto.

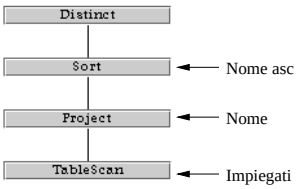

Figura 11: Piano di accesso dell'interrogazione 4.1.

Interrogazione 4.2

```
SELECT   Nome, COUNT(*)
FROM     Impiegati
GROUP BY Nome;
```

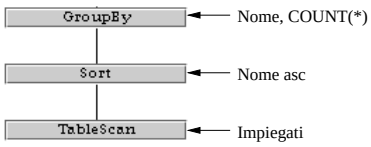

Figura 12: Piano di accesso dell'interrogazione 4.2.

In Figura 12 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	336	ms $(min : 0, s : 0, ms : 336)$
`Result cardinality`	169	*Record*
`Cost`	112	*Logical Reads*

Si noti come la presenza della clausola `GROUP BY` porti all'inserimento di un opportuno operatore di ordinamento nel piano di accesso prodotto.

4.1 Ordinamenti inutili

L'operatore di ordinamento risulta essere inutile nei seguenti casi:

- l'operando produce le ennuple nell'ordine richiesto;
- gli attributi di ordinamento risultano essere tutti legati in condizioni del tipo `Attributo=Costante` oppure `Attributo=Attributo2 AND Attributo2=Costante` nella clausola `WHERE`.

Interrogazione 4.3

```
SELECT   Nome
FROM     Impiegati
GROUP BY Nome
ORDER BY Nome asc;
```

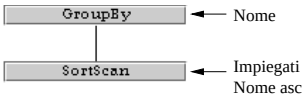

Figura 13: Piano di accesso dell'interrogazione 4.3.

In Figura 13 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	391 ms $(min : 0, s : 0, ms : 391)$
`Result cardinality`	169 *Record*
`Cost`	80 *Logical Reads*

Si noti come né la presenza della clausola `GROUP BY` né la presenza della clausola `ORDER BY` porta all'inserimento di un operatore di ordinamento nel piano di accesso prodotto in quanto l'ottimizzatore effettua una scansione ordinata della tabella e quindi le ennuple sono già nell'ordine voluto.

Interrogazione 4.4

```
SELECT   DISTINCT Nome
FROM     Impiegati
WHERE    Nome = 'Leonardo';
```

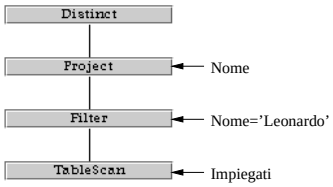

Figura 14: Piano di accesso dell'interrogazione 4.4.

In Figura 14 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	679 *ms* $(min : 0, s : 0, ms : 679)$
`Result cardinality`	17 *Record*
`Cost`	16 *Logical Reads*

Si noti come nonostante la presenza della clausola `DISTINCT` nel piano di accesso non è presente alcun operatore di ordinamento in quanto la presenza della condizione `Nome = 'Leonardo'` rende le ennuple che giungono all'operatore `Distinct` già ordinate sull'attributo `Nome`[1].

5 Piani di accesso che usano solo indici

In alcuni casi si può usare la sola informazione presente in un indice definito su una relazione per produrre i risultati di una interrogazione. Ciò è possibile se l'interrogazione non "usa" altri attributi della relazione se non quelli su cui è definito l'indice.

Questa proprietà è importante poiché permette di evitare l'accesso ai dati e quindi di generare dei piani di accesso di costo inferiore.

L'operatore fisico che rappresenta l'uso dei soli dati definiti nell'indice è l'operatore `IndexOnlyFilter` e nel seguito si mostrano dei piani di accesso che ne fanno uso per eseguire rispettivamente operazioni di proiezione, restrizione, giunzione e raggruppamento.

Interrogazione 5.1

```
SELECT   CapoFamiglia, Nome
FROM     Familiari
WHERE    CapoFamiglia <= 3
ORDER BY CapoFamiglia desc;
```

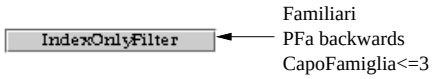

Figura 15: Piano di accesso dell'interrogazione 5.1.

In Figura 15 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	1920 *ms* $(min : 0, s : 1, ms : 920)$
`Result cardinality`	6 *Record*
`Cost`	1 *Logical Reads*

[1] Le ennuple assumono lo stesso valore su tale attributo.

Usando solo l'indice multiattributo `PFa` si riesce ad eseguire completamente l'interrogazione, in particolare ad eseguire la restrizione e quindi la proiezione. Infatti l'indice contiene i valori degli attributi chiave `(CapoFamiglia, Nome)` per ogni ennupla presente nella tabella e quindi tutta l'informazione necessaria all'esecuzione. Inoltre effettuando la scansione *backward* dell'indice si ottengono le ennuple ordinate nel modo voluto.

Interrogazione 5.2

```
SELECT   Familiari.Nome
FROM     Impiegati, Familiari
WHERE    Codice <= 3
  AND    Codice = CapoFamiglia;
```

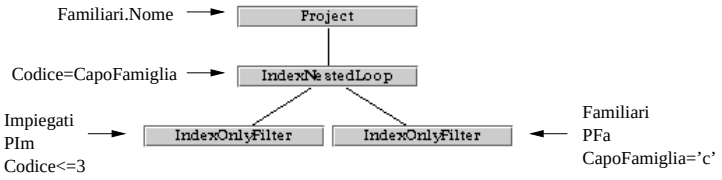

Figura 16: Piano di accesso dell'interrogazione 5.2.

In Figura 16 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	6628 *ms* $(min : 0, s : 6, ms : 628)$
`Result cardinality`	3 *Record*
`Cost`	4 *Logical Reads*

Usando l'indice multiattributo `PFa` per la relazione `Familiari` e l'indice `PIm` per la relazione `Impiegati` si riesce ad eseguire completamente l'interrogazione. Si noti (a) l'uso dell'indice `PFa` e non dell'indice `FCaFa`, entrambi contengono l'attributo `CapoFamiglia` ma il secondo non contiene l'attributo `Nome` usato nella clausola `SELECT` e (b) l'operatore `IndexOnlyFilter` che usa l'indice `PIm` esegue la condizione di restrizione sulla relazione `Impiegati`.

Interrogazione 5.3

```
SELECT   CapoFamiglia, MAX(Nome), MIN(Nome)
FROM     Familiari
WHERE    CapoFamiglia <= 3
GROUP BY CapoFamiglia;
```

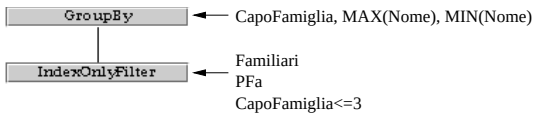

Figura 17: Piano di accesso dell'interrogazione 5.3.

In Figura 17 è riportato il piano di accesso che presenta le seguenti caratteristiche:

`Time`	2238 ms $(min : 0, s : 2, ms : 381)$
`Result cardinality`	6 *Record*
`Cost`	1 *Logical Reads*

I dati dell'indice multiattributo `PFa` permettono di eseguire completamente l'interrogazione, infatti scandendo l'indice si ottengono i valori degli attributi chiave `(CapoFamiglia, Nome)` di ogni ennupla presente nella tabella `Familiari`. Inoltre tali ennuple sono ordinate sull'attributo di raggruppamento e quindi l'ordinamento richiesto dall'operatore di raggruppamento è superfluo.

6 Le prestazioni del sistema

Si presentano i risultati di alcuni test per valutare le prestazioni delle strategie di ricerca realizzate e per confrontare l'ottimizzatore con la versione precedente del sistema JRS.

6.1 L'ambiente sperimentale

Per stimare le prestazioni dell'ottimizzatore si sono utilizzate interrogazioni di due tipi: *a catena* e *a stella*.

```
SELECT *
FROM   R0, R1, ..., R9
WHERE  R0.a0 = R1.i ∧ R1.i = R2.a1 ∧
       R2.a2 = R3.i ∧ R3.i = R4.a4 ∧
       R4.a5 = R5.i ∧ R5.i = R6.a5 ∧
       R6.a6 = R7.a7 ∧ ... ∧ R8.a9 = R9.a9;
```

Esempio di *interrogazione a catena* da 10 tabelle.

```
SELECT *
FROM   R0, R1, ..., R9
WHERE  R0.i = R1.i ∧
       R0.i = R2.i ∧
       R0.i = R3.i ∧
       R0.a4 = R4.a7 ∧ ... ∧ R0.a9 = R9.a9;
```

Esempio di *interrogazione a stella* da 10 tabelle.

In entrambi i tipi di interrogazioni, tre delle nove giunzioni sono eseguite su uno stesso attributo (i) mentre le altre su attributi diversi. La presenza di questo attributo *condiviso* favorisce l'uso dell'algoritmo *sort merge* e

inoltre rende l'ottimizzazione di queste interrogazioni più complessa poiché l'ordinamento sull'attributo $R_0.i$ è *interessante* per tutti i sottopiani che includono R_0 e non coinvolgono R_1, R_2 e R_3.

Si sono eseguite 5 interrogazioni per ogni tipo di interrogazione, facendo variare il numero di relazioni coinvolte tra 2 e 10, nonché le relazioni in esame. Ossia si sono eseguite 5 interrogazioni a stella con 2 relazioni, 5 con 3 tabelle e via di seguito per un totale di 90 interrogazioni diverse. I parametri quantitativi delle relazioni utilizzate sono stati generati in modo casuale così come le ennuple inserite nelle stesse.

Si è utilizzata una macchina dotata di processore Intel Pentium III a 800 MHz con 256 Mb di RAM e sistema operativo Microsoft Windows Me.

6.2 I risultati ottenuti

6.2.1 Le strategie di ricerca

In Figura 18 sono riportati i valori medi dei tempi di generazione del piano di accesso delle tre strategie di ricerca implementate.

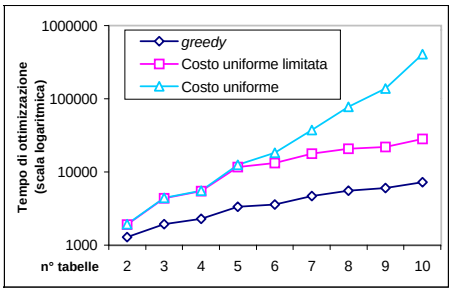

Figura 18: Valori medi dei tempi di generazione del piano di accesso.

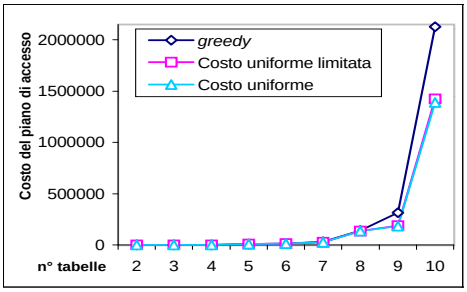

Figura 19: Media dei costi del piano di accesso generato.

Si noti come i tempi ottenuti con la strategia di ricerca *greedy* siano sempre inferiori, anche notevolmente, rispetto alle altre due strategie validando ulteriormente le osservazioni riportate in precedenza. Confrontando i tempi delle strategie di costo uniforme e di costo uniforme limitata notiamo come i due grafici procedano appaiati per interrogazioni che coinvolgono fino 5 tabelle, infatti la strategia di costo uniforme limitata implementata segue un approccio di costo uniforme per generare piani sino a quattro tabelle quindi procede con approccio *greedy*.

Tali differenze aumentano all'aumentare del numero di tabelle coinvolte nell'interrogazione: si passa dall'ordine dei millisecondi nel caso di interro-

gazioni che coinvolgono due tabelle[2] a differenze dell'ordine dei minuti per interrogazioni che coinvolgono dieci tabelle[3].

In Figura 19 sono riportati i valori medi dei costi, stimati come numero di letture logiche di pagine, del piano di accesso delle tre strategie di ricerca implementate.

Ancora una volta i dati ottenuti confermano come la strategia di ricerca di costo uniforme produca i piani migliori. La strategia di ricerca di costo uniforme limitata produce piani meno buoni ma comunque migliori, anche molto, rispetto alla strategia *greedy*.

Le differenze di costo tra i piani prodotti dalle tre strategie variano al variare del numero di tabelle coinvolte nell'interrogazione: si passa dall'ordine delle decine di letture logiche per interrogazioni che coinvolgono tre tabelle[4](per interrogazioni che coinvolgono due tabelle le tre strategie hanno generato sempre lo stesso piano) a differenze dell'ordine del milione per interrogazioni che coinvolgono dieci tabelle[5].

Dai dati raccolti si evince come la strategia di ricerca di costo uniforme limitata possa rappresentare il giusto compromesso tra tempo di ottimizzazione e qualità della soluzione trovata quando il numero di tabelle coinvolte risulta essere maggiore di sette. Ad esempio, per interrogazioni che coinvolgono dieci tabelle il tempo di ottimizzazione risulta essere maggiore di quello di una ricerca *greedy* ($28s$ contro $7s$), ma il costo del piano generato risulta essere notevolmente inferiore avvicinandosi a quello della ricerca di costo uniforme (1 423 909 letture logiche contro 2 127 519). Il maggior tempo di ottimizzazione viene compensato da un minor tempo di esecuzione del piano di accesso: stimando a $20ms$ il tempo medio di una lettura logica, e considerando interrogazioni che coinvolgono 10 tabelle, si ottiene che il tempo complessivo di risposta all'interrogazione risulta essere di circa 5 ore con ricerca di costo uniforme limitata contro le 11 ore con ricerca *greedy* e le 7 ore della ricerca di costo uniforme.

[2] $1291ms$ con ricerca *greedy*, $1917ms$ con ricerca di costo uniforme limitata e $1926ms$ con ricerca di costo uniforme.

[3] $7s, 278ms$ con ricerca *greedy*, $28s, 363ms$ con ricerca di costo uniforme limitata e $6min, 46s, 503ms$ con ricerca di costo uniforme.

[4] 445 letture logiche con ricerca *greedy*, 412 letture logiche con ricerca di costo uniforme limitata e 412 letture logiche con ricerca di costo uniforme.

[5] 2 127 519 letture logiche con ricerca *greedy*, 1 423 909 letture logiche con ricerca di costo uniforme limitata e 1 388 681 letture logiche con ricerca di costo uniforme.

6.2.2 Lo spazio di ricerca

In Figura 20 sono riportati i valori medi dei tempi di generazione del piano di accesso esplorando lo spazio degli alberi profondi a sinistra e lo spazio degli alberi generali usando sempre la strategia di ricerca di costo uniforme.

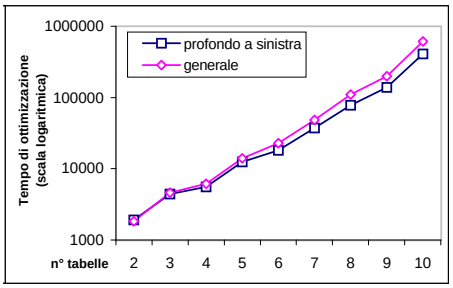

Figura 20: Valori medi dei tempi di generazione del piano di accesso.

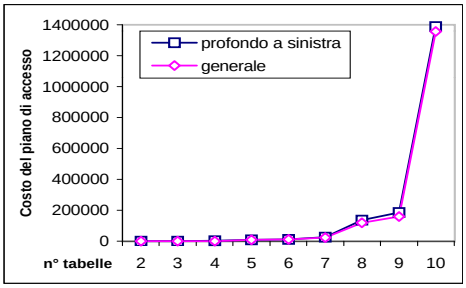

Figura 21: Media dei costi del piano di accesso generato.

Come era facile prevedere, poiché bisogna esplorare un numero minore di alberi logici, il tempo impiegato nel caso di ricerca nello spazio dei soli alberi profondi a sinistra risulta essere sempre inferiore rispetto a quello con alberi generali. Tali differenze aumentano all'aumentare del numero di tabelle coinvolte nell'interrogazione: si passa da $1843ms$ (alberi profondi a sinistra) contro $1926ms$ (alberi generali) per interrogazioni con due tabelle a $6min, 46s, 503ms$ (alberi profondi a sinistra) contro $10min, 8s, 591ms$ (alberi generali) per interrogazioni che coinvolgono dieci tabelle.

In Figura 21 sono riportati i valori medi dei costi dei piani di accesso con alberi profondi a sinistra e alberi generali, usando sempre la strategia di ricerca di costo uniforme.

I dati ottenuti confermano come il piano migliore sia presente sempre nello spazio degli alberi generali. Le differenze di costo tra i piani prodotti esplorando i due spazi variano al variare del numero di tabelle coinvolte nell'interrogazione: si passa dall'ordine delle centinaia di letture logiche per interrogazioni che coinvolgono tre tabelle[6] (per interrogazioni che coinvolgono due tabelle non c'è differenza, lo spazio è lo stesso) a differenze dell'ordine delle decine di migliaia di letture logiche per interrogazioni che coinvolgono dieci tabelle[7].

[6] 412 letture logiche con alberi profondi a sinista e 217 letture logiche con alberi generali.

[7] 1 388 681 letture logiche con alberi profondi a sinista e 1 357 347 letture logiche con alberi generali.

Dai dati raccolti si può dedurre come per ottenere quindi la soluzione migliore bisogna indagare lo spazio degli alberi generali infatti il maggior tempo di ottimizzazione viene compensato da un minor tempo di esecuzione del piano di accesso trovato.

6.2.3 Il confronto con la versione precedente dell'ottimizzatore

In Figura 22 sono riportati i valori medi dei tempi di generazione del piano di accesso della versione precedente dell'ottimizzatore del sistema JRS e della nuova versione realizzata.

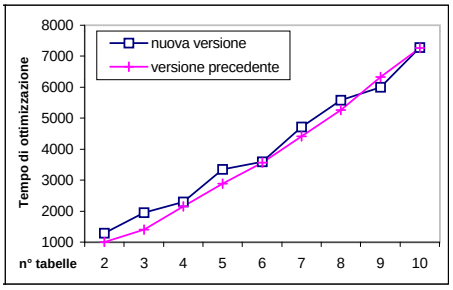

Figura 22: Valori medi dei tempi di generazione del piano di accesso.

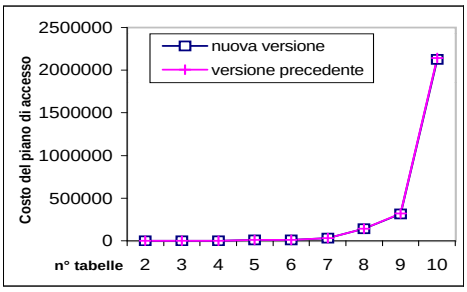

Figura 23: Media dei costi del piano di accesso generato.

Tali tempi così come i valori riportati in Figura 23 fanno riferimento alla sola versione della strategia di ricerca *greedy* in quanto la strategia di ricerca di costo uniforme della versione precedente dell'ottimizzatore risulta essere notevolmente più lenta rispetto alla nuova versione[8], mentre la strategia di ricerca di costo uniforme limitata non era prevista.

Si noti come i due ottimizzatori presentino dei tempi confrontabili, le differenze si limitano ad essere dell'ordine dei millisecondi nonostante la presenza, nella nuova versione dell'ottimizzatore, di altri operatori fisici e di accorgimenti che nella versione precedente non erano implementati.

In Figura 23 sono riportati i valori medi dei costi del piano di accesso della versione precedente dell'ottimizzatore del sistema JRS e della nuova versione realizzata.

I costi dei piani di accesso generati dalla versione precedente risultano essere sempre superiori rispetto a quelli generati dalla nuova versione anche se tali differenze risultano essere poco eclatanti tanto da non risultare nel

[8] Per un'interrogazione con sei tabelle la versione precedente impiega $2min, 27sec, 200ms$ contro $2sec, 910ms$ della nuova versione.

grafico: si passa dall'ordine delle poche unità di letture logiche per interrogazioni che coinvolgono tre tabelle[9](per interrogazioni che coinvolgono due tabelle non c'è differenza) a ordine delle decine di migliaia per interrogazioni che coinvolgono dieci tabelle[10].

Concludendo, la nuova versione dell'ottimizzatore risulta essere "migliore" di quella precedente nonostante sia stata utilizzata un'architettura generale per avere un ottimizzatore estendibile.

7 Conclusioni

Si sono presentate alcune interrogazioni, con i relativi piani di accesso generati, per mostrare alcune caratteristiche della nuova versione dell'ottimizzatore del sistema JRS. Sono stati inoltre presentati alcuni dati sperimentali che hanno permesso di osservare che:

- per la ricerca del piano di accesso migliore bisogna usare la strategia di ricerca di costo uniforme ed esplorare lo spazio degli alberi generali;
- la strategia di ricerca di costo uniforme limitata risulta essere un'alternativa valida nella ricerca del piano di accesso per interrogazioni che coinvolgono un numero di tabelle maggiore o uguale a sette, in quanto richiede un tempo di calcolo inferiore rispetto alla ricerca di costo uniforme e genera, in generale, un piano di costo poco superiore a quello generato con ricerca di costo uniforme;
- la strategia di ricerca *greedy* è la più rapida nella generazione di un piano di accesso anche se il piano generato non è quello migliore.

[9] 445 letture logiche della nuova versione e 449 letture logiche della versione precedente.
[10] 2 127 519 letture logiche della nuova versione e 2 142 342 letture logiche della versione precedente.