Integracja systemów transakcyjnych

Robert Wrembel
Politechnika Poznańska
Instytut Informatyki
`Robert.Wrembel@cs.put.poznan.pl`
`www.cs.put.poznan.pl/rwrembel`

Alokacja danych

- **Alokacja danych**
  - **umieszczanie danych w węzłach rozproszonej BD**
- **Wykorzystuje**
  - **fragmentację**
  - **replikację**
- **Uwzględnia**
  - **obciążenie węzłów**
    - **odczyty danych ⇨ dane powinny być "blisko" użytkownika**
    - **modyfikacje danych ⇨ należy synchronizować repliki**

Alokacja danych

- Wywodzi się z problemu alokacji plików w sieci komputerowej
- Problem jest NP-zupełny

  n fragmentów, m węzłów: $(2^m - 1)^n$
- Zysk z alokacji zależy od
  - "jakości alokacji"
  - możliwości optymalizatora zapytań

Algorytm Best Fit

- Podejście intuicyjne
- Fragment Fi pamiętamy na stanowisku Sj, na którym liczba referencji odczytu i modyfikacji Fi jest największa
- Ograniczenie: fragment alokuje się tylko w jednym węźle ⇨ brak replikacji
- Przykład
  - charakterystyka przetwarzania
  - z węzłów S1 i S4 jest realizowany dostęp do F1 i F2 z częstością 1
    - F1: 3 odczyty i 1 modyfikacja
    - F2: 2 odczyty
  - ...

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

Best Fit - Przykład

➲ **Charakterystyka przetwarzania**

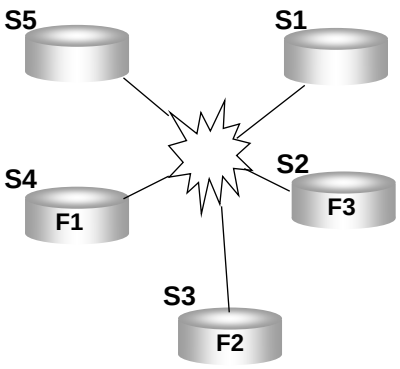
Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

Fragment	Węzeł	Trans. T1	Trans. T2	Trans. T3	Suma referenc.
F1	S1	1*(3r+1w)	0	0	4
	S2	0	2*2r	0	4
	S3	0	0	0	0
	S4	1*(3r+1w)	2*2r	0	8
	S5	0	0	0	0

Fragment	Węzeł	Trans. T1	Trans. T2	Trans. T3	Suma referenc.
F2	S1	1*2r	0	0	2
	S2	0	0	0	0
	S3	0	0	3*(3r+1w)	12
	S4	1*2r	0	0	2
	S5	0	0	3*(3r+1w)	12

Fragment	Węzeł	Trans. T1	Trans. T2	Trans. T3	Suma referenc.
F3	S1	0	0	0	0
	S2	0	2*(3r+1w)	0	8
	S3	0	0	3*2r	6
	S4	0	2*(3r+1w)	0	8
	S5	0	0	3*2r	6

Best Fit - Przykład



Best Fit

- Algorytm prosty obliczeniowo
- Nie uwzględnia replikacji
- Mała "dokładność"
  - uwzględnia tylko częstości odwołań do fragmentów

Algorytm optymalizacji zysku

- Algorytm określa wszystkie węzły, na których opłaca się umieścić fragment ⇨ wykorzystanie replik
- Fragment umieszczany w tych węzłach, dla których zysk z umieszczenia fragmentu jest większy niż koszt przechowywania i aktualizowania tego fragmentu
- Fragment Fi pamiętamy w węźle Sj, w którym koszt odczytu jest większy niż koszt zapisu Fi z dowolnego węzła w systemie

Algorytm optymalizacji zysku

- **Koszt utrzymywania dodatkowej kopii fragmentu Fi w węźle Sj:**
  - **sumaryczny czas wszystkich lokalnych aktualizacji fragmentu Fi przez transakcje inicjowane w węźle Sj plus sumaryczny czas wszystkich zdalnych aktualizacji fragmentu Fi przez transakcje inicjowane w pozostałych węzłach SRBD**
- **Zysk z utworzenia dodatkowej kopii fragmentu Fi w węźle Sj:**
  - **różnica między czasem wykonania zapytania zdalnego (bez replikacji) a czasem wykonania zapytania lokalnego pomnożona przez częstość zapytania do fragmentu Fi z węzła Sj**

Przykład

- **Charakterystyki przetwarzania**

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

- **Koszty modyfikacji dla schematów alokacji fragmentu F1**
  - **F1 alokowany kolejno w S1, S2, ..., S5**

Fragment	Węzeł	Modyfikacje lokalne i zdalne	częstość * liczba mod. * czas	Koszt
F1	S1	zdalna z S4 (lokalna w S1)	1*1*600 (zdalna) + 1*1*150 (lokalna)	750
	S2	zdalna z S1 zdalna z S4	1*1*600 (zdalna) + 1*1*600 (zdalna)	1200
	S3	zdalna z S1 zdalna z S4	1*1*600 (zdalna) + 1*1*600 (zdalna)	1200
	S4	zdalna z S1 (lokalna w S4)	1*1*600 (zdalna) + 1*1*150 (lokalna)	750
	S5	zdalna z S1 zdalna z S4	1*1*600 (zdalna) + 1*1*600 (zdalna)	1200

Przykład

➲ Charakterystyki przetwarzania

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

➲ Koszty modyfikacji dla schematów alokacji fragmentu F2

- **F2 alokowany kolejno w S1, S2, ..., S5**

Fragment	Węzeł	Modyfikacje lokalne i zdalne	częstość * liczba mod. * czas	Koszt
F2	S1	zdalna z S3 zdalna z S5	3*1*700 (zdalna) + 3*1*700 (zdalna)	4200
	S2	zdalna z S3 zdalna z S5	3*1*700 (zdalna) + 3*1*700 (zdalna)	4200
	S3	zdalna z S5 lokalna w S3	3*1*700 (zdalna) + 3*1*200 (zdalna)	2700
	S4	zdalna z S3 zdalna z S5	3*1*700 (zdalna) + 3*1*700 (zdalna)	4200
	S5	zdalna z S3 lokalna w S5	3*1*700 (zdalna) + 3*1*200 (zdalna)	2700

Przykład

➲ Charakterystyki przetwarzania

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

➲ Koszty modyfikacji dla schematów alokacji fragmentu F3

- **F3 alokowany kolejno w S1, S2, ..., S5**

Fragment	Węzeł	Modyfikacje lokalne i zdalne	częstość * liczba mod. * czas	Koszt
F3	S1	zdalna z S2 zdalna z S4	2*1*1100 (zdalna) + 2*1*1100 (zdalna)	4400
	S2	zdalna z S4 lokalna w S2	2*1*1100 (zdalna) + 2*1*250 (lokalna)	2700
	S3	zdalna z S2 zdalna z S4	2*1*1100 (zdalna) + 2*1*1100 (zdalna)	4400
	S4	zdalna z S2 lokalna w S4	2*1*1100 (zdalna) + 2*1*250 (lokalna)	2700
	S5	zdalna z S2 zdalna z S4	2*1*1100 (zdalna) + 2*1*1100 (zdalna)	4400

Przykład

- **Charakterystyki przetwarzania**

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

- **Zysk dla schematów alokacji fragmentu F1**

Fragment	Węzeł	Zapytanie	częstość * liczba mod. * czas	Zysk
F1	S1	z S1	1*3*(500-100)	1200
	S2	z S2	2*2*(500-100)	1600
	S3	brak	0	0
	S4	z S4 z S4	1*3*(500-100) + 2*2*(500-100)	2800
	S5	brak	0	0

Przykład

- **Charakterystyki przetwarzania**

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

- **Zysk dla schematów alokacji fragmentu F2**

Fragment	Węzeł	Zapytanie	częstość * liczba mod. * czas	Zysk
F2	S1	z S1	1*2*(650-150)	1000
	S2	brak	0	0
	S3	z S3	3*3*(650-150)	4500
	S4	z S4	1*2*(650-150)	1000
	S5	z S5	3*3*(650-150)	4500

Przykład

➲ **Charakterystyki przetwarzania**

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

➲ **Zysk dla schematów alokacji fragmentu F3**

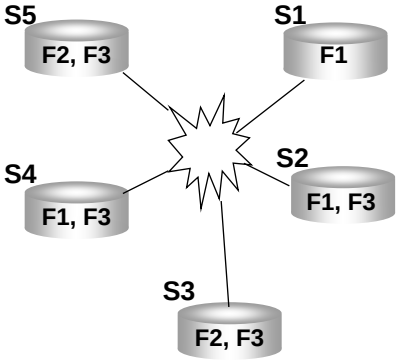
Fragment	Węzeł	Zapytanie	częstość * liczba mod. * czas	Zysk
F3	S1	brak	0	0
	S2	z S2	2*3*(1000-200)	4800
	S3	z S3	3*2*(1000-200)	4800
	S4	z S4	2*3*(1000-200)	4800
	S5	z S5	3*2*(1000-200)	4800

Przykład

➲ **Uzyskany schemat alokacji**

- **fragment Fi jest alokowany we wszystkich węzłach dla których zysk>koszt**

Fragment	Węzeł	Koszt	Zysk
F1	**S1**	750	**1200**
	S2	1200	**1600**
	S3	1200	0
	S4	**750**	**2800**
	S5	1200	0
F2	S1	4200	1000
	S2	4200	0
	S3	**2700**	**4500**
	S4	4200	1000
	S5	**2700**	**4500**
F3	S1	4400	0
	S2	**2700**	**4800**
	S3	**4400**	**4800**
	S4	**2700**	**4800**
	S5	**4400**	**4800**



Sformułowanie problemu

- **Dane**
  - **charakterystyki i rozkłady zapytań**
  - **charakterystyki i rozkłady operacji DML**
  - **węzły sieci**
  - **przepustowość połączeń pomiędzy węzłami**
  - **moc przetwarzania w węzłach**
- **Poszukiwane**
  - **schemat rozmieszczenia fragmentów w węzłach (możliwe redundancje), tak by pewna funkcja kosztu była minimalna / maksymalna**

Niezbędne dane

- **Fragmenty F = {F_1, F_2, ..., F_j}**
  - **rozmiar każdego fragmentu (F_j) musi być określony ⇨ koszt komunikacji sieciowej**
- **Transakcje T = {T_1, T_2, ..., T_i}**
  - **rodzaje (odczyt, zapis)**
  - **częstotliwość uruchamiania**
  - **podzbiory adresowanych danych**
- **Węzły**
  - **pojemność dysków**
  - **charakterystyka I/O**
  - **CPU, ...**
- **Sieć komputerowa S = {S_1, S_2, ..., S_k}**
  - **przepustowość**

Informacje o transakcjach

- **RM (Retrieval Matrix)**

	F1	F2	F3	F4	F5
T1	2	3	0	0	0
T2	2	0	0	1	0
T3	0	0	3	0	0
T4	3	0	2	0	0

- **UM (Update Matrix)**

	F1	F2	F3	F4	F5
T1	0	0	0	1	2
T2	0	3	0	0	0
T3	2	1	0	1	0
T4	0	0	0	0	3

Informacje o transakcjach

- **Nie wszystkie rekordy muszą być odczytywane lub modyfikowane (Selectivity Matrix)**

SEL:(%)

	F1	F2	F3	F4	F5
T1	0.1	0.1	0	0.3	0.2
T2	0.1	0.3	0	1	0
T3	2	5	0.1	0.5	0
T4	0.5	0	10	0	4

- **Częstotliwość odwołań (Frequency Matrix)**

FRQ:

	S1	S2	S3	S4
T1	0	2	3	1
T2	0	3	0	0
T3	2	0	1	0
T4	0	0	4	0

Informacje o sieci komputerowej

- Węzły połączone medium komunikacyjnym o określonej przepustowości (Communication Cost Matrix)
- Uproszczenie: macierz komunikacji jest symetryczna

CCM

	S1	S2	S3	S4
S1	0	0.32	0.48	0.16
S2	0.32	0	0.64	0.32
S3	0.48	0.64	0	0.64
S4	0.16	0.32	0.64	0

Koszt

- Koszt może uwzględniać
  - liczbę operacji I/O (rozmiar danych)
  - czas procesora
  - koszt komunikacji sieciowej (przesyłania danych)
  - rozmiar bufora danych (cache)
  - ...

Koszt

- Minimalizacja kosztu może dotyczyć
  - kosztu przechowywania każdego fragmentu Fj w węźle Sk
  - kosztu wykonywania zapytań Fj w węźle Sk
  - kosztu modyfikowania Fj we wszystkich węzłach, gdzie Fj jest przechowywany
  - kosztu komunikacji sieciowej
- Kryteria optymalizacji
  - minimalizacja czasu odpowiedzi
  - maksymalizacja przepustowości każdego węzła

Algorytmy statyczne

- Uruchamiane w czasie bezczynności systemu
- Wykonywane okresowo
- Złożone obliczeniowo
  - $O(jk^2i)$
    - j - liczba fragmentów tabeli
    - k - liczba węzłów SRBD, w których są alokowane dane
    - i - liczba transakcji

Przykładowy algorytm

- **Cel optymalizacji**
  - **rozmieszczenie fragmentów w węzłach w ten sposób, by koszt był minimalny**

$$\min(CC_{transfer} + CC_{transakcji})$$

$CC_{transfer}$ – koszt transferu danych
$CC_{transakcji}$ – koszt wykonania transakcji

- **Dane wejściowe**
  - **RM(Ti, Fj) – Retrieval matrix**
  - **UM(Ti, Fj) – Update matrix**
  - **SEL(Ti, Fj) – Selectivity matrix**
  - **FRQ(Ti, Sk) – Frequency matrix**
  - **CCM(Sk, Sm) – Communication cost matrix**
- **Dane wyjściowe**
  - **FAT(Fj, Sk) – Fragment Allocation Table**

Przykładowy algorytm

- **Inicjalizacja macierzy FAT (Fragment Allocation Table)**
- **Algorytm - krok 1**
  - **dla każdej transakcji Ti, fragmentu Fj, węzła Sk wykonaj:**
    - jeżeli
      - liczba dostępów transakcji Ti do Fj w Sk i
      - częstotliwość wywołania transakcji Ti w węźle Sk > 0
    - to pozostaw kopię fragmentu Fj w węźle Sk

Przykładowy algorytm

```
Krok 1
    For Ti in T, Fj in F, Sk in S do
        if (RM(Ti,Fj) * FREQ(Ti,Sk) > 0)
            FAT(Fj,Sk) := 1
```

RM

	F1	F2	F3	F4	F5
T1	2	3	0	0	0
T2	2	0	0	1	0
T3	0	0	3	0	0
T4	3	0	2	0	0

FREQ:

	S1	S2	S3	S4
T1	0	2	3	1
T2	0	3	0	0
T3	2	0	1	0
T4	0	0	4	0

FAT - krok 1

	F1	F2	F3	F4	F5
S1	0	0	0	0	0
S2	1	1	0	1	0
S3	1	1	1	0	0
S4	1	1	0	0	0

Przykładowy algorytm

- **Usunięcie kopii zaalokowanych fragmentów w celu redukcji kosztów odświeżania**
  - **usunięcie kopii dla których zysk < koszt**
- **Algorytm - krok 2**
  - **dla każdego Fj zaalokowanego w więcej niż 1 węźle, procedura usuwania jest powtarzana dopóki dla każdego fragmentu zysk > koszt lub pozostanie tylko 1 kopia Fj w SRBD**
  - **jeżeli zysk < koszt we wszystkich węzłach to Fj jest pozostawiany w tym węźle, w którym zysk-koszt jest największa**

```
Krok 2
For Fj in F do
   While (NumFragCopy(Fj) > 1)
     begin
       Let Sk be the site with FAT(Fj,Sk)=1
       and a minimum value of (Benefit(Fj,Sk)-Cost(Fj,Sk));
       if ((Benefit(Fj,Sk) - Cost(Fj,Sk)) < 0)
           FAT(Fj,Sk) = 0
     end;
```

Przykładowy algorytm

Cost(Fj,Sk)

	F1	...	F5
S1	0	...	0
S2	6	...	0
S3	4	...	0
S4	3	...	0

Benefit(Fj,Sk)

	F1	...	F5
S1	0	...	0
S2	4	...	0
S3	7	...	0
S4	9	...	0

FAT z kroku 1

	F1	F2	F3	F4	F5
S1	0	0	0	0	0
S2	1	1	0	1	0
S3	1	1	1	0	0
S4	1	1	0	0	0

FAT - krok 2

	F1	...	F5
S1	0	...	0
S2	0	...	0
S3	1	...	0
S4	1	...	0

Przykładowy algorytm

- **Sprawdzenie czy wszystkie fragmenty zostały zaalokowane**
- **Algorytm - krok 3**
  - **jeżeli któryś z fragmentów nie został jeszcze zaalokowany, ale są na nim wykonywane modyfikacje, to jako węzeł przechowywania należy wybrać ten z najmniejszym kosztem wykonania operacji**

```
For Fj in F do
    if (NumFragCopy(Fj) = 0 and UM(Ti,Fj) > 0)
      begin
        Sk = MinOperCost(Fj);
        FAT(Fj,Sk) = 1;
      end;
```

Przykładowy algorytm

UM

	F1	F2	F3	F4	F5
T1	0	0	0	1	2
T2	0	3	0	0	0
T3	2	1	0	1	0
T4	0	0	0	0	3

FREQ:

	S1	S2	S3	S4
T1	0	2	3	1
T2	0	3	0	0
T3	2	0	1	0
T4	0	0	4	0
suma:	0	2	7	1

FAT - krok 2

	F1	...	F5
S1	0	...	0
S2	0	...	0
S3	1	...	0
S4	1	...	0

FAT - krok 3

	F1	...	F5
S1	0	...	0
S2	1	...	0
S3	0	...	1
S4	0	...	0

Przykładowy algorytm - modyfikacja

- Kroki 1 i 3 są takie same jak w poprzednim algorytmie
- Krok 2 - usuwanie fragmentów
  - przejrzenie wszystkich fragmentów z kroku 1 w porządku malejących wag
  - waga jest wyliczana na podstawie liczby aktualizacji zrealizowanych przez transakcje pochodzące z innych węzłów
  - kopia jest usuwana z węzła na podstawie wag (więcej aktualizacji – fragment szybciej usunięty z węzła)

Algorytmy dynamiczne

- Rozkład obciążenia węzłów i charakterystyki dostępu do danych zmieniają się w czasie
  - algorytmy statyczne zmniejszają wydajność baz danych
  - konieczność relokacji fragmentów ⇨ algorytmy dynamiczne
    - Simple Counter
    - Load Sensitive Counter

Simple Counter

- Jeden dedykowany węzeł SRBD utrzymuje liczniki dostępów z każdego węzła do każdego fragmentu (bloku danych)
- Proces systemowy okresowo sprawdza liczniki dla każdego fragmentu (bloku danych) w regularnych odstępach czasowych
- Relokacja rekordów ma miejsce, gdy węzeł z najwyższą wartością licznika jest inny niż ten, gdzie dane są aktualnie składowane

Simple Counter

- **Proces systemowy**
  - **gromadzi statystyki takie jak: przepustowość, średni czas odpowiedzi, liczba transakcji wymagających dostępu do danych**
- **Okres sprawdzania liczników**
  - **musi zapewnić szybką reakcję na zmieniające się obciążenie**
  - **musi zapewnić relokację danych po ustabilizowaniu się zmian obciążenia**

Simple Counter

- **Zalety:**
  - **zapewnia zadowalający schemat alokacji, gdy**
    - **obciążenie systemu jest niskie**
    - **węzły są w miarę równomiernie obciążone**
    - **zmiany obciążenia stabilizują się w czasie**
- **Wady:**
  - **jeżeli większość zapytań pochodzi z jednego węzła, to wszystkie rekordy mogą znaleźć się w tym węźle ⇨ przeciążenie węzła**

Load Sensitive Counter

- Monitoruje obciążenie systemu i częstotliwość dostępu do danych
- Relokacja danych wykonywana jest jako Simple Counter Algorithm
- Relokacja wykonywana jest jeśli tylko nie powoduje przekroczenia pewnego poziomu obciążenia węzła
- Parametry algorytmu
  - maksymalny procent danych przechowywanych w węźle
  - maksymalny poziom obciążenia

Algorytmy dynamiczne

- Cechy
  - nieefektywne, gdy częstotliwość sprawdzania liczników jest za duża lub gdy obciążenie systemu zmienia się szybko
  - najgorsza wydajność, gdy algorytm próbuje nadążyć za zmieniającym się obciążeniem (dużo relokacji danych)

Bibliografia

- M. T. Özsu, P. Valduriez, Distributed and Parallell Database Systems, ACM Computing Surveys, vol. 28, no. 1, 1996, pp.125-128
- Y-F. Huang, J-H. Chen, Fragment Allocation in Distributed Database Design, July 2000, pp. 491-506
- A. Brunstrom, S. T. Leutenegger, R. Simha, Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database With Changing Workloads, no. TR-95-2, 1995, pp. 1-15
- P. M. G. Apers, Data Allocation in Distributed Database Systems, ACM Transactions on Database Systems, vol. 13, no. 3, September 1988, pp. 263-304