

Tutorial 2: Distributed Query Processing

INFS3200/7907
Advanced Database Systems

1

Overview

- Distributed query processing
 - How does DDBMS process a query?
 - Semi-join
- Distributed database design
 - Derived horizontal fragmentation

2

Questions

- Q2 – Distributed Query Optimisation
- Q3 – Query Processing using Semijoin
- Q5 (a, c, d, e and f) – Derived Horizontal Fragmentation
- Q1 – Primary Horizontal Fragmentation

3

Q2 – Distributed query Optimization

Question: Give operator trees for the following query

- After query decomposition using global schema
- After localization
- After reduction.

4

Question 2 (Continue)

Database Schemas

- Results (Olympiad, EventID, CompID, Pos)
 - *Olympiad* - one of the 4-year intervals between Olympic Games
 - *CompID* (Competitor ID) is a foreign key of table *Competitors*.
- Competitors (CompID, Name, Country, FirstOlympiad, Other)
 - *FirstOlympiad* is the first Olympiad in which the competitor competed.
 - *Other* – a group of other attributes irrelevant to this question.

5

Question 2 (Continue) - Fragmentation

BOC (Beijing Olympic Committee):

- $\sigma_{\text{Olympiad}=2008}$ Results
- $\Pi_{\text{CompID, Name, Country}}$ Competitors

IOC (International Olympic Committee):

- $\sigma_{\text{Olympiad}<2008}$ Results
- $\Pi_{\text{CompID, FirstOlympiad, Other}}$ Competitors

6

Question 2 (Continue) – Fragmentation

BOC

- $\sigma_{\text{Olympiad}=2008}$ Results (Horizontal)
 - Fragment **rb** (the rows in Beijing)
 - Select *
 - From Results
 - Where Olympiad = '2008'
- $\pi_{\text{CompID, Name, Country}}$ Competitors (Vertical)
 - Fragment **cb** (the columns in Beijing)
 - Select CompID, Name, Country
 - From Competitors

7

Question 2 (Continue) – Fragmentation

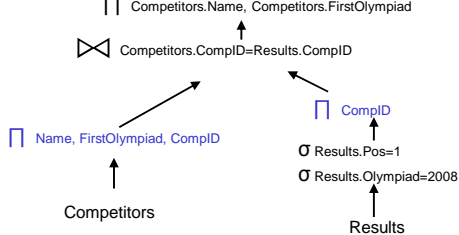
IOC

- $\sigma_{\text{Olympiad}<2008}$ Results (Horizontal)
 - Fragment **ri** (The rows in IOC)
 - Select *
 - From Results
 - Where Olympiad < '2008'
- $\pi_{\text{CompID, FirstOlympiad, Other}}$ Competitors (Vertical)
 - Fragment **ci** (The columns in IOC)
 - Select CompID, FirstOlympiad, Other
 - From Competitors

8

Question 2 (Continue)

- Query
 - SELECT c.Name, c.FirstOlympiad
 - FROM Competitors c, Results r
 - WHERE r.pos=1 AND r.Olympiad=2008 AND c.CompID=r.CompID
- The Query Tree with optimization using *global schema*



9

Question 2 (Continue)

After localization

- The global relations are replaced by their fragments
- Results is replaced by the UNION of

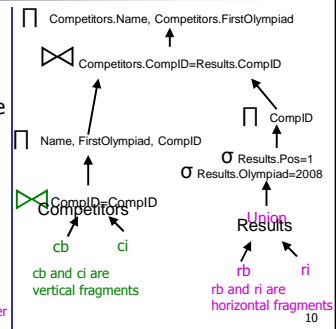
rb: $\sigma_{\text{Olympiad}=2008}$ Results

ri: $\sigma_{\text{Olympiad}<2008}$ Results

- Competitors by the JOIN of

cb: $\pi_{\text{CompID, Name, Country}}$ Competitors

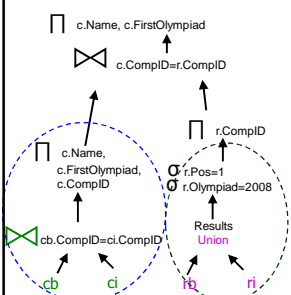
ci: $\pi_{\text{CompID, FirstOlympiad, Other}}$ Competitors



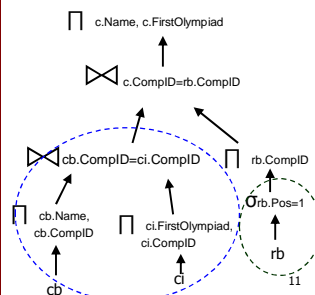
10

Question 2 (Continue)

After localization, but before reduction



After Reduction



11

Semijoin

- Semijoin

- The semijoin is joining similar to the natural join
- written as $R \bowtie S$ where R and S are relations, and
- Whereas the result of the semijoin is only the set of all tuples in R for which there is a tuple in S that is equal on their common attribute names (foreign keys).
- $R \bowtie S$ is not equivalent to $S \bowtie R$

12

Question 3 – Query Processing using semijoin

Schemas

- Results (EventID, CompID, Pos)
- Records (EventID, Time)

Results

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
SwM100Free	Iles	7
ShotputW	Cumba	1
ShotputW	Ostapchuk	4
ShotputW	Li	9

Records

EventID	Time
SwM100Free	0:47.84
SwW100Free	0:53.77

13

Q3a

Compute results $\triangleright <$ records.

- Can *Results* semi-joins *Records* (*Results* $\triangleright <$ *Records*)?
- Yes, *EventID* in *Results* is a foreign key of *Records*.
- From *Results* perspective, the outcome of *Results* $\triangleright <$ *Records* is a set of the tuples in *Results* including the identical values (*SwM100Free*) under the foreign key (*EventID*) in *Records*.

Results

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
SwM100Free	Iles	7
ShotputW	Cumba	1
ShotputW	Ostapchuk	4
ShotputW	Li	9

Records

EventID	Time
SwM100Free	0:47.84
SwW100Free	0:53.77

Results $\triangleright <$ Records

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
SwM100Free	Iles	7

Q3b

Compute records $\triangleright <$ results.

- Can *Records* semi-joins *Results* (*Records* $\triangleright <$ *Results*)?
- Yes, *EventID* in *Records* is a foreign key of *Results*.
- From *Records* perspective, the outcome of *Records* $\triangleright <$ *Results* is a set of the tuples in *Records* including the identical values (*SwM100Free*) under the foreign key (*EventID*) in *Results*.

Records

EventID	Time
SwM100Free	0:47.84
SwW100Free	0:53.77

Results

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
SwM100Free	Iles	7
ShotputW	Cumba	1
ShotputW	Ostapchuk	4
ShotputW	Li	9

Records $\triangleright <$ Results

EventID	Time
SwM100Free	0:47.84

15

Q3c –

Assume *Results* is at site 1 and *Records* is at site 2, and a query *Results* $\triangleright <$ *Records* has been issued at site 2. Give steps for a query processing strategy using *semijoin*, and check if the *semijoin* is a *beneficial* option in this case (ignore local processing cost)

SITE 1

Results

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
SwM100Free	Iles	7
ShotputW	Cumba	1
ShotputW	Ostapchuk	4
ShotputW	Li	9

SITE 2

Records

EventID	Time
SwM100Free	0:47.84
SwW100Free	0:53.77

Strategy 1 (naïve)

- Send the entire table (*Results*) from site 1 to site 2
- Cost: 21 cells (7 tuples * 3 attributes/tuple)

16

Q3c (Continues) –

Assume *Results* is at site 1 and *Records* is at site 2, and a query *Results* $\triangleright <$ *Records* has been issued at site 2. Give steps for a query processing strategy using *semijoin*, and check if the *semijoin* is a *beneficial* option in this case (ignore local processing cost)

SITE 1

Results

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
SwM100Free	Iles	7
ShotputW	Cumba	1
ShotputW	Ostapchuk	4
ShotputW	Li	9

SITE 2

Records

EventID	Time
SwM100Free	0:47.84
SwW100Free	0:53.77

Strategy 2 (semijoin)

- Send the cell values (2 cells) under *EventID* in *Records* from site 2 to site 1.
- In site 1, *Results* $\triangleright <$ *Records* with the cells, and send back the outcome of the semijoin (totally 12 cell values) to site 2.
- The total (transmission) cost is 14 cells that are, obviously, **CHEAPER** than the Strategy 1 which cost is 21 cells.

17

A Quick Recap of Fragmentation

Types of Fragmentation

- Horizontal Fragmentation
- Vertical Fragmentation
- Hybrid fragmentation

18

Types of Fragmentation

- Horizontal Fragmentation
 - Characteristics
 - A relation (table) is divided **along tuples**.
 - The fragments are mutually exclusive/disjointed.
 - How?
 - **SELECT** Operation in Relational Algebra
 - The reconstruction of the fragmentation is by **UNION**.

19

Types of Fragmentation

- Vertical Fragmentation
 - Characteristics
 - A relation is partitioned **with attributes**.
 - Each fragment must contain primary keys.
 - How?
 - **Project** Operation in Relational Algebra
 - The reconstruction of the fragmentation is by **JOIN**.

20

Types of Fragmentation

- Hybrid fragmentation
 - The combination of Horizontal and Vertical Fragmentation

21

Question 5 Derived Horizontal Fragmentation

- **Q (a):** Write an SQL query to find the **CompID** and **Pos** of competitors in **swimming events**.
 - Database Schemas
 - Events (**EventID**, **EName**, Venue, Indoors?, **Sport**)
 - Results (**EventID**, **CompID**, **Pos**)
- **Solution:**

```
SELECT Results.CompID, Results.Pos
FROM Results, Events
WHERE Events.EventID=Results.EventID AND
Events.Sport = "Swim"
```

22

Question 5 (continue)

- **Q (c):** How many fragments of **Events** are generation from applying the predicates
 - P1: Sport = 'Swim'
 - P2: Sport = 'Track'
- **Solution:**
 - there are totally two fragments:
 - one with Sport = 'Swim',

```
SELECT *
FROM Events
WHERE Sport = 'Swim'
```
 - the other with Sport = 'Track'.

```
SELECT *
FROM Events
WHERE Sport = 'Track'
```

23

Question 5 (continue)

- **Q (d):** What is the relationship between Events and Results, that is which is owner and which is member?
 - *Derived Horizontal Fragmentation* is defined on a **member** relation of a link according to a selection operation specified on its **owner**.
 - Database Schema
 - Events (**EventID**, **EName**, Venue, Indoors?, **Sport**)
 - Results (**EventID**, **CompID**, **Pos**)
- **Solution:**
 - **Events** is the **owner** and **Results** is the **member** because the **Results** depends on the **Events**.
 - Weak-Entity Relationship

24

Question 5 (continue)

- Q (e): What is the *join predicate* of the relations **Events** and **Results**
 - Database
 - Events (EventID, EName, Venue, Indoors?, Sport)
 - Results (EventID, CompID, Pos)
- Solution:
 - Events.EventID=Results.EventID

25

Question 5 (continue)

- Q (f): Find the derived horizontal fragments of the **Results** relation on the basis of the predicates used in (c) (P_1 : Sport = 'Swim', P_2 : Sport = 'Track')

26

Question 5 (continue)

Solution for Q (f):-

Events (owner)		Results (member)		
EventID	Sport	EventID	CompID	Pos
M100FS	Swim	M100FS	Thorpe	3
M100Sp	Track	M100FS	Hoogenband	1
W200B	Swim	M100FS	Schoeman	2
W1500	Track	M100FS	Iles	7
MSP	Track	WSP	Cumba	1
WSP	Track	WSP	Ostapchuk	4
MMar	Track	WSP	Li	9

Fragment 1 (Sport="Swim")		
EventID	CompID	Pos
M100FS	Thorpe	3
M100FS	Hoogenband	1
M100FS	Schoeman	2
M100FS	Iles	7

Fragment 1 (Sport="Track")		
EventID	CompID	Pos
WSP	Cumba	1
WSP	Ostapchuk	4
WSP	Li	9

Join Predicate:
Events.EventID = Results.EventID

29

Q1- Primary horizontal fragmentation

- Primary Horizontal Fragmentation (of a relation) is performed using **predicates** that are defined on that **relation**
- For example of relation S

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
SwM100Free	Iles	7
ShotputW	Cumba	1
ShotputW	Ostapchuk	4
ShotputW	Li	9

Predicates:
 p_1 : Pos ≤ 3
 p_2 : Pos > 3

28

Question 1a (Continue)

Question (a): How do we check if Pr (P_1, P_2) is complete and minimal.

Predicates:
 p_1 : Pos ≤ 3 p_2 : Pos > 3

29

Question 1a (Continue)

Primary Horizontal Fragmentation is defined by a **selection operation** on the owner **relations** of a databases schema.

- In this Question: there are a couple of selection operations :-
 - Database Schema
 - Relation Results (EventID, CompID, Pos)
- SELECT *
 - FROM Results
 - WHERE Pos ≤ 3
- SELECT *
 - FROM Results
 - WHERE Pos > 3

30

Question 1a (Continue)

COM_MIN Algorithm

Given: a relation **Results** and a set of simple predicates Pr (p1: Pos≤3, p2: Pos>3)

Output: a complete and minimal set of simple predicates Pr' for Pr

Rule 1: a relation or fragment is partitioned into at least two parts which are accessed differently by at least one application.

31

Question 1a (Continue)

COM_MIN Algorithm

Given: relation S and Pr= {p1, p2}

Output: Pr' = Pr

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
SwM100Free	Iles	7
ShotputW	Cumba	1
ShotputW	Ostapchuk	4
ShotputW	Li	9

p₁: Pos≤3

OR

p₂: Pos>3

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
ShotputW	Cumba	1

EventID	CompID	Pos
SwM100Free	Iles	7
ShotputW	Ostapchuk	4
ShotputW	Li	9

32

Question 1b (Continue)

- Question (b), apply the PHORIZONTAL algorithm.
 - Sometimes, the resultant set of predicates from COM_MIN algorithm could be large and redundant. Thus, how we can minimize the number of predicates, but maintain the completeness.

33

Question 1b (Continue)

- PHORIZONTAL Algorithm
 - Makes use of COM_MIN to preform fragmentation.
 - Input: a relation S and a set of simple predicates Pr
 - Output: a set of minterm predicates M according to which relation S is to be fragmented

34

Question 1b (Continue)

- Workings
 - Step 1: Pr' ← COM_MIN(S, Pr)
 - According to the Q2(a) result, Pr' = Pr = {p1, p2}

35

Question 1b (Continue)

- Step 2: determine the set M of minterm predicates.
 - Minterm - as a logical expression of *n* variables consisting of only the logical and (∧) operator and complements.
 - m1: (Pos≤3) ∧ (Pos>3) = Φ
 - m2: ¬(Pos≤3) ∧ (Pos>3) = (Pos>3)
 - m3: (Pos≤3) ∧ ¬(Pos>3) = (Pos≤3)
 - m4: ¬(Pos≤3) ∧ ¬(Pos>3) = Φ

36

Question 1b (Continue)

- Step 3: determine the set I of implications among $p_i \in Pr$

$$i_1: (Pos \leq 3) \rightarrow \neg (Pos > 3)$$

$$i_2: \neg (Pos \leq 3) \rightarrow (Pos > 3)$$

$$i_3: (Pos > 3) \rightarrow \neg (Pos \leq 3)$$

$$i_4: \neg (Pos > 3) \rightarrow (Pos \leq 3)$$

37

Question 1b (Continue)

Final step: according to I,

- $m1 ((Pos \leq 3) \wedge (Pos > 3))$ AND
- $m4 (\neg (Pos \leq 3) \wedge \neg (Pos > 3))$

are contradictory and eliminated from M.

Thus, we are left with $M = \{m_2, m_3\}$ and define two fragments $F_S = \{S_1, S_2\}$ according to M.

38

Question 1b (Continue)

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
SwM100Free	Iles	7
ShotputW	Cumba	1
ShotputW	Ostapchuk	4
ShotputW	Li	9

$m_2: Pos > 3$

OR

$m_3: Pos \leq 3$

EventID	CompID	Pos
SwM100Free	Thorpe	3
SwM100Free	Hoogenband	1
SwM100Free	Schoman	2
ShotputW	Cumba	1

EventID	CompID	Pos
SwM100Free	Iles	7
ShotputW	Ostapchuk	4
ShotputW	Li	9

39

Thank You

40