Query Processing in Multidatabase Systems
Query Processing in Three Steps

1. Global query is decomposed into local queries

2. Each local query is translated into queries over the corresponding local database system

3. Results of the local queries are combined into the answer
Outline

• Overview of major query processing components in multidatabase systems:
  - Query Decomposition
  - Query Translation
  - Global Query Optimization

• Techniques for each of the above components
Overview

Query decomposition & global optimization

SQ1 SQ2 SQn

PQ1 PQ1

TQ1 TQn

DB1 DBn

SQi - export schema subquery in global query language

TQi - target query (local subquery) in local query language

PQi - postprocessing query used to combine results returned by subqueries to form the answer
Assumptions

• We use the object-oriented data model to present a query modification algorithm

• To simplify the discussion, we assume that there are only two export schemas:

  **ES1**
  Emp1: SSN
  Name
  Salary
  Age

  **ES2**
  Emp2: SSN
  Name
  Salary
  Rank
Definitions

• **type**: Given a class $C$, the type of $C$ denoted by $\text{type}(C)$, is the set of attributes defined for $C$ and their corresponding domains.

• **extension**: the extension of $C$, denoted by $\text{extension}(C)$, is the set of instances contained in $C$.

• **world**: the world of $C$, denoted by $\text{world}(C)$, is the set of real-world objects described by $C$. 
The **outerjoin** of relation $R1$ and $R2$ is the union of three components:

- the join of $R1$ and $R2$,
- dangling tuples of $R1$ padded with null values, and
- dangling tuples of $R2$ padded with null values.
## Outerjoin Example

<table>
<thead>
<tr>
<th>Emp1</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OID</td>
<td>SSN</td>
<td>Name</td>
<td>Salary</td>
<td>Age</td>
</tr>
<tr>
<td>3</td>
<td>6789</td>
<td>Smith</td>
<td>90,000</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>4321</td>
<td>Chang</td>
<td>62,000</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>8642</td>
<td>Patel</td>
<td>75,000</td>
<td>35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EmpO</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OID</td>
<td>SSN</td>
<td>Name</td>
<td>Salary</td>
<td>Age</td>
<td>Rank</td>
</tr>
<tr>
<td>1</td>
<td>2222</td>
<td>Ahad</td>
<td>98,000</td>
<td>null</td>
<td>S. Mgr.</td>
</tr>
<tr>
<td>2</td>
<td>7531</td>
<td>Wang</td>
<td>95,000</td>
<td>null</td>
<td>S. Mgr.</td>
</tr>
<tr>
<td>3</td>
<td>6789</td>
<td>Smith</td>
<td>Inconsistent</td>
<td>40</td>
<td>Mgr.</td>
</tr>
<tr>
<td>4</td>
<td>4321</td>
<td>Chang</td>
<td>62,000</td>
<td>30</td>
<td>null</td>
</tr>
<tr>
<td>5</td>
<td>8642</td>
<td>Patel</td>
<td>75,000</td>
<td>35</td>
<td>null</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Emp2</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OID</td>
<td>SSN</td>
<td>Name</td>
<td>Salary</td>
<td>Rank</td>
</tr>
<tr>
<td>1</td>
<td>2222</td>
<td>Ahad</td>
<td>98,000</td>
<td>S. Mgr.</td>
</tr>
<tr>
<td>2</td>
<td>7531</td>
<td>Wang</td>
<td>95,000</td>
<td>S. Mgr.</td>
</tr>
<tr>
<td>3</td>
<td>6789</td>
<td>Smith</td>
<td>25,000</td>
<td>Mgr.</td>
</tr>
</tbody>
</table>
Two classes $C_1$ and $C_2$ can be integrated by equi-outerjoining the two classes on the OID to form a new class $C$.

- $\text{extension}(C) = \text{extension}(C_1) \Join_{o} \text{extension}(C_2)$
- $\text{type}(C) = \text{type}(C_1) \cup \text{type}(C_2)$
- $\text{world}(C) = \text{world}(C_1) \cup \text{world}(C_2)$
Schema Integration - Generalization

Two classes $C_1$ and $C_2$ can be integrated by generalizing the two classes to form the superclass $C$.

- $\text{type}(C) = \text{type}(C_1) \cap \text{type}(C_2)$
- $\text{extension}(C) = \prod_{\text{type}(C)} [\text{extension}(C_1) \cup \text{extension}(C_2)]$
- $\text{world}(C) = \text{world}(C_1) \cup \text{world}(C_2)$
Generalization Example

Emp1: SSN
    Name
    Salary
    Age

Emp2: SSN
    Name
    Salary
    Rank

EmpG: SSN
    Name
    Salary

• Emp1 and Emp2 will also appear in the global schema since not all information in Emp1 and Emp2 is retained in EmpG
Inconsistency Resolution

- The schema integration techniques work as long as there is no data inconsistency.

- If data inconsistency may exist, then aggregate functions may be used to resolve the problem.
Inconsistency Resolution Example

<table>
<thead>
<tr>
<th>Export Schemas</th>
<th>Integrated Schemas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp1: SSN</td>
<td>EmpG: SSN</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Salary</td>
<td>Salary</td>
</tr>
<tr>
<td>Age</td>
<td>Rank</td>
</tr>
<tr>
<td>Emp2: SSN</td>
<td>EmpO: SSN</td>
</tr>
<tr>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Salary</td>
<td>Salary</td>
</tr>
<tr>
<td>Age</td>
<td>Age</td>
</tr>
<tr>
<td></td>
<td>Rank</td>
</tr>
</tbody>
</table>

Sample Aggregate Functions:

EmpG.Name = Emp1.Name, if EmpG is in world(Emp1)
   = Emp2.Name, if EmpG is in world(Emp2) - world(Emp1)

EmpG.Salary = Emp1.Salary, if EmpG is in world(Emp1) - world(Emp2)
   = Emp2.Salary, if EmpG is in world(Emp2) - world(Emp1)
   = Sum(Emp1.Salary, Emp2.Salary), if EmpG is in world(Emp1) \( \cap \) world(Emp2)

EmpO.Age = Emp1.Age, if EmpO is in world(Emp1)
   = Null, if EmpO is in world(Emp2) - world(Emp1)

EmpO.Rank = Emp2.Rank, if EmpO is in world(Emp2)
   = Null, if EmpO is in world(Emp1) - world(Emp2)
Query Modification (1)

Global Query
Select EmpO.Name, EmpO.Rank
From EmpO
Where EmpO.Salary > 80,000 AND EmpO.Age > 35

STEP 1: Obtain a partition of world(EmpO) based on the function used to resolve the data inconsistency.

Strategy 1 (based on Salary)
part. 1: world(Emp1) – world(Emp2)
part. 2: world(Emp2) – world(Emp1)
part. 3: world(Emp1) \cap world(Emp2)

Strategy 2 (based on Age)
part. 1: world(Emp1)
part. 2: world(Emp2) – world(Emp1)

We use Strategy 1 since it is the finest partition among all the partitions.
Query Modification (2)

**Strategy 1:**

1. world(Emp1)

2. world(Emp2)

**Strategy 2:**

1. world(Emp1)

2. world(Emp2)

**Use finer partition:**

1. world(Emp1)

2. world(Emp2)
**Global Query:**
Select EmpO.Name, EmpO.Rank
From EmpO
Where EmpO.Salary > 80,000 AND EmpO.Age > 35

**Partition:**

1. Select Emp1.Name
   From Emp1
   Where Emp1.Salary > 80,000 AND Emp1.Age > 35 AND Emp1.SSN NOT IN
   (Select Emp2.SSN
    From Emp2)

2. This subquery is discarded because EmpO.Age is Null.

3. Select Emp1.Name, Emp2.Rank
   From Emp1, Emp2
   Where Sum(Emp1.Salary, Emp2.Salary) > 80,000 AND Emp1.Age > 35 AND Emp1.SSN = Emp2.SSN

**STEP 2:** Obtain a query for each subset in the chosen partition.
**Query Modification (4)**

**STEP 3:** Some resulting query may still reference data from more than one database. They need to be further decomposed into subqueries and possibly also postprocessing queries.

**Before STEP 3:**

```sql
Select Emp1.Name
From Emp1
Where Emp1.Salary > 80,000 and
   Emp1.Age > 35 and
   Emp1.SSN NOT IN
   (Select Emp2.SSN
    From Emp2)
```

```sql
Insert INTO X
Select Emp2.SSN
From Emp2
```
Query Modification (5)

**STEP 4:** It may be desirable to reduce the number of subqueries by combining subqueries for the same database.
Query Translation (1)

IF  Global Query Language ≠  Local Query Language

THEN  Export  Schema  Subquery  Translator  Query  Language
Query Translation (2)

**IF** the source query language has a higher expressive power **THEN EITHER**

- Some source queries cannot be translated; or
- they must be translated using both
  - the syntax of the target query language, and
  - some facilities of a high-level programming language.

**Example:** A recursive OODB query may not be translated into a relational query using SQL alone.
**Translation Techniques (1)**

**CASE 1:** A single target query is generated

IF the target database system has a query optimizer

   THEN the query optimizer can be used to optimize the translated query

ELSE the translator has to consider the performance issues
Translation Techniques (2)

**CASE 2:** A set of target queries is needed.

- It might pay to have the minimum number of queries
  - It minimizes the number of invocations of the target system
  - It may also reduce the cost of combining the partial results
- It might pay for a set to contain target queries that can be well coordinated
  - The results or intermediate results of the queries processed earlier can be used to reduce the cost of processing the remaining queries
Relation-to-OO Translation

**OODB Schema:**

- **Auto**
  - OID
  - Color
  - Manufacturer

- **Company**
  - OID
  - Name
  - Profit
  - Headquarter
  - President

- **People**
  - OID
  - Name
  - Hometown
  - Automobile
  - Age

- **City**
  - OID
  - Name
  - State

**Equivalent Relational Schema:**

- **Auto**(Auto-OID, Color, Company-OID)
- **Company**(Company-OID, Name, Profit, City-OID, People-OID)
- **People**(People-OID, Name, Age, City-OID, Auto-OID)
- **City**(City-OID, Name, State)
Relational-to-OO Example (1)

Global Query:
Select Auto1.*
From Auto Auto1, Auto Auto2,
Company, People,
City City1, City City2
Where Auto1.Company-OID =
   Company,Company-OID AND
Company.People-OID =
   People.People-OID AND
People.Age = 52 AND
People.Auto-OID =
   Auto2.Auto-OID AND
Auto2.Color = “red” AND
People.City-OID =
   City1.City-OID AND
City1.Name = City2.Name AND
Company.City-OID =
   City2.City-OID

Relational Predicate Graph:
Relational-to-OO Example (2)

**OO Predicate Graph:**

**OO Query:**

Where  Auto.Manufacturer.President.Automobile.Color = red AND
      Auto.Manufacturer.President.Age = 52  AND
      Auto.Manufacturer.Headquarter.Name =
      Auto.Manufacturer.President.Hometown.Name
Global Query Optimization (1)

- A query obtained by the query modification process may still reference data from more than one database.

**Example:** part. 3 (i.e., world(Emp1) \( \cap \) world(Emp2)) on page 108

```
Select   Emp1.Name, Emp2.Rank
From     Emp1, Emp2      /* access two databases
Where   sum(Emp1.Salary, Emp2.Salary) > 80,000   AND
       Emp1.Age > 35   AND
       Emp1.SSN =  Emp2.SSN

→ Some global strategy is needed to process such queries
```
Global Query Optimization (2)

- Select Emp1.Name, Emp2.Rank
  From Emp1, Emp2 /* access two databases
  Where sum(Emp1.Salary, Emp2.Salary) > 80,000 AND
  Emp1.Age > 35 AND
  Emp1.SSN = Emp2.SSN

→ Some global strategy is needed to process such queries
Data Inconsistency

• If $C$ is integrated from $C1$ and $C2$ with no data inconsistency on attribute $A$, then

$$\delta_{A \text{ op } a} (C) = \delta_{A \text{ op } a} (C1) \cup \delta_{A \text{ op } a} (C2)$$

• If $A$ has data inconsistency, then the above equality may no longer hold.

**Example:** Consider the select operation

$$\delta_{\text{EmpO.Salary} > 100,000} (\text{EmpO})$$

the correct answer should have the record for Smith. However, the above equation will return an empty set!
Data Inconsistency - Solution

Express an outerjoin (or a generalization) as outer-unions as follows:

\[ C_1 \bowtie_o C_2 = C_1-O \cup_o C_2-O \cup_o (C_1-C \bowtie_{oID} C_2-C) \]

- **C1-O**: Those tuples of \( C_1 \) that have no matching tuples in \( C_2 \) (private part)
- **C1-C**: Those tuples of \( C_1 \) that have matching tuples in \( C_2 \) (overlap part)

\[ \delta_{A \, \text{op}_a} (C_1 \bowtie_o C_2) = \delta_{A \, \text{op}_a} (C_1-O) \cup_o \delta_{A \, \text{op}_a} (C_2-O) \]
\[ \quad \cup_o \delta_{A \, \text{op}_a} (C_1-C \bowtie C_2-C) \]
Distribution of Selections (1)

\[ \delta_{A \text{ op a}} (C1 \bowtie C2) = \delta_{A \text{ op a}} (C1-O) \cup_o \delta_{A \text{ op a}} (C2-O) \]

When can we distribute \( \delta \) over \( \bowtie \)?

Expensive operation
Distribution of Selection (2)

Four cases were identified when all arguments of the aggregate function are non-negative

1. \( f(A_1, A_2) \) \( \text{op a} \equiv A_1 \text{ op a AND A}_2 \text{ op a} \):

   \[ \delta_{A \text{ op a}} (C_1 - C \bowtie C_2 - C) = \delta_{A \text{ op a}} (C_1 - C) \bowtie \delta_{A \text{ op a}} (C_2 - C) \]

   **Example:** \( \max(Emp1-C.Salary, Emp2-C.Salary) < 30K \)
   \[ \equiv \max(Emp1-C.Salary < 30K, \text{AND} \quad \max(Emp2-C.Salary < 30K) \]

2. \( f(A_1, A_2) \) \text{op a} \equiv f(A_1 \text{ op a}, A_2 \text{ op a}) \text{ op a}:

   \[ \delta_{A \text{ op a}} (C_1 - C \bowtie C_2 - C) = \delta_{A \text{ op a}} (\delta_{A_1 \text{ op a}} (C_1 - C) \bowtie \delta_{A_2 \text{ op a}} (C_2 - C)) \]

   **Example:** \( \sum(Emp1-C.Salary, Emp2-C.Salary) < 30K \)
   \[ \equiv \sum(Emp1-C.Salary < 30K, \text{AND} \quad \sum(Emp2-C.Salary < 30K) < 30K \]
Distribution of Selection (3)

3. \( f(A_1, A_2) \text{ op a} \equiv f(A_1 \text{ op’ a}, A_2 \text{ op’ a}) \text{ op a} \):

\[
\bar{\delta}_A \text{ op a}(C_1 - C \bowtie C_2 - C) = \bar{\delta}_A \text{ op a}(\bar{\delta}_{A_1} \text{ op’ a}(C_1 - C) \bowtie \bar{\delta}_{A_2} \text{ op’ a}(C_2 - C))
\]

**Example**: \( \text{sum(Emp1-C.Salary, Emp2-C.Salary)} = 30K \)
\( \equiv \text{sum(Emp1-C.Salary} \leq 30K, \text{Emp2-C.Salary} \leq 30K) = 30K \)

4. **No improvement is possible**:

**Example**: \( \text{sum(Emp1-C.Salary, Emp2-C.Salary)} > 30K \)
Distribution Rules for $\bar{b}$ over $\bowtie$

$\bar{b}_A \text{ op } a(C1-C \bowtie C2-C)$

<table>
<thead>
<tr>
<th>A</th>
<th>sum(A1, A2)</th>
<th>avg(A1, A2)</th>
<th>max(A1, A2)</th>
<th>min(A1, A2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>&gt;</td>
<td>≥</td>
<td>≤</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>≥</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>≤</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>&lt;</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>=</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>≠</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>in</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Not in</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Problem in Global Query Optimization (1)

Important information about local entity sets that is needed to determine global query processing plans may not be provided by the local database systems.

- **Example**: cardinalities
  availability of fast access paths

- **Techniques**:
  - Sampling queries may be designed to collect statistics about the local databases.
  - A monitoring system can be used to collect the completion time for subqueries. This can be used to better estimate subsequent subqueries.
Problems in Global Query Optimization (2)

• Different query processing algorithms may have been used in different local database systems.
  → Cooperation across different systems difficult
    Examples: Semijoin may not be supported on some local systems.

• Data transmission between different local database systems may not be fully supported.
  Examples:
  – A local database system may not allow update operations
  – For many nonrelational systems, the instances of one entity set are more likely to be clustered with the instances of other entity sets. Such clustering makes it very expensive to extract data for one entity set.
  → Need more sophisticated decomposition algorithms.