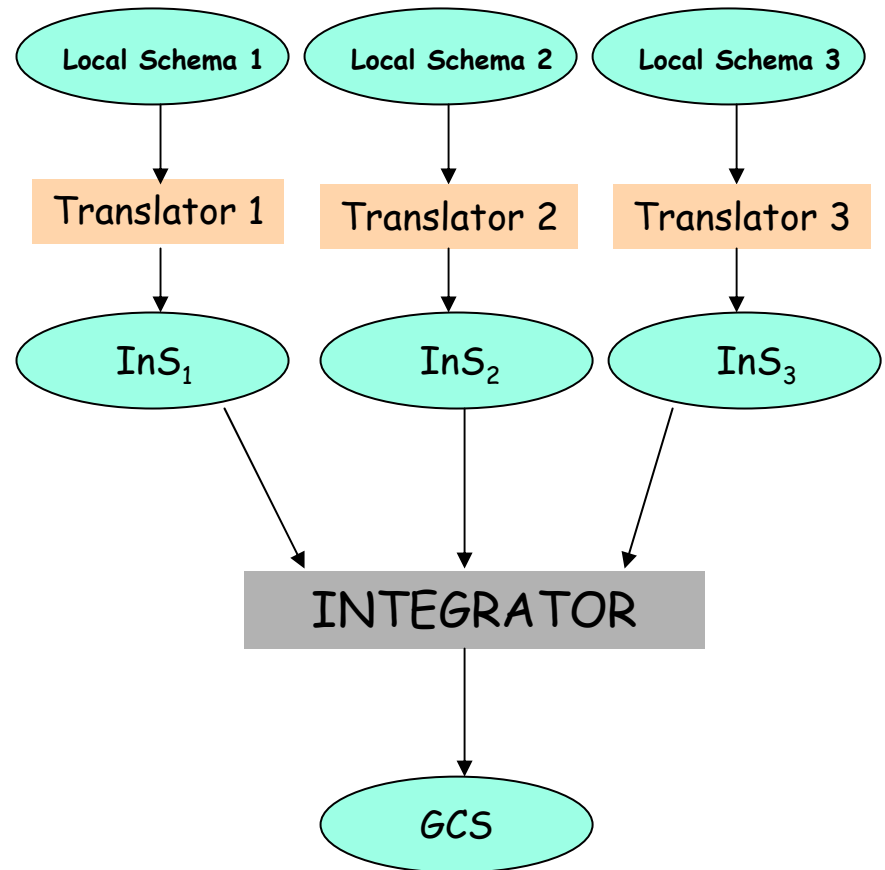


Query Processing in Multidatabase Systems

Query Processing in Three Steps

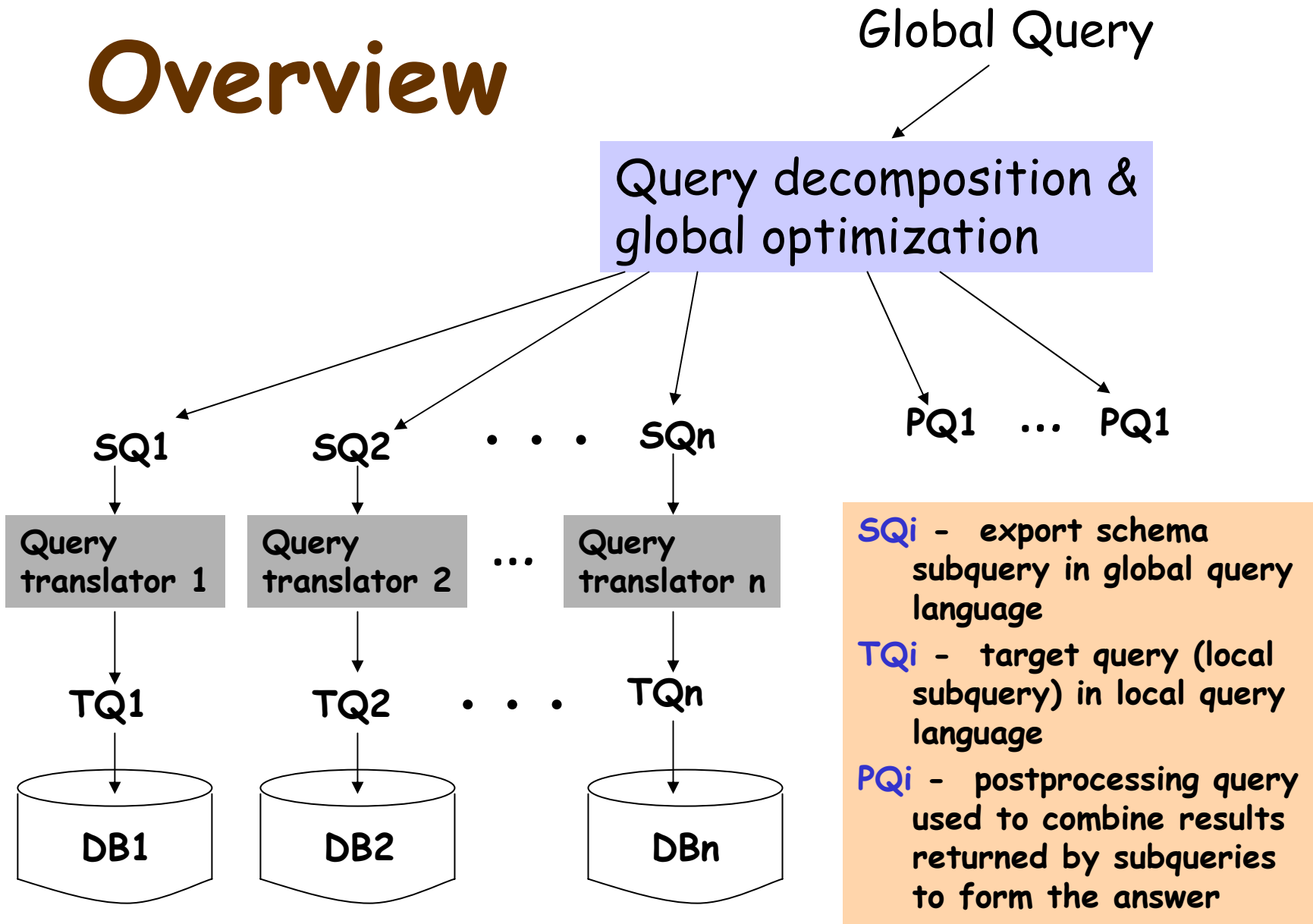
1. Global query is decomposed into local queries
2. Each local query is translated into queries over the corresponding local database system
3. Results of the local queries are combined into the answer



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



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 \mathcal{C} , the type of \mathcal{C} denoted by $\text{type}(\mathcal{C})$, is the set of attributes defined for \mathcal{C} and their corresponding domains.
- **extension**: the extension of \mathcal{C} , denoted by $\text{extension}(\mathcal{C})$, is the set of instances contained in \mathcal{C} .
- **world**: the world of \mathcal{C} , denoted by $\text{world}(\mathcal{C})$, is the set of real-world objects described by \mathcal{C} .

Review: *Outerjoin*

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

Emp1

OID	SSN	Name	Salary	Age
3	6789	Smith	90,000	40
4	4321	Chang	62,000	30
5	8642	Patel	75,000	35

Emp2

OID	SSN	Name	Salary	Rank
1	2222	Ahad	98,000	S. Mgr.
2	7531	Wang	95,000	S. Mgr.
3	6789	Smith	25,000	Mgr.

EmpO

OID	SSN	Name	Salary	Age	Rank
1	2222	Ahad	98,000	null	S. Mgr.
2	7531	Wang	95,000	null	S. Mgr.
3	6789	Smith	Incon-sistent	40	Mgr.
4	4321	Chang	62,000	30	null
5	8642	Patel	75,000	35	null

Schema Integration - *Outerjoin*

Two classes $C1$ and $C2$ can be integrated by equi-outerjoining the two classes on the OID to form a new class C .

- $\text{extension}(C) = \text{extension}(C1) \bowtie_o \text{extension}(C2)$
- $\text{type}(C) = \text{type}(C1) \cup \text{type}(C2)$
- $\text{world}(C) = \text{world}(C1) \cup \text{world}(C2)$

Schema Integration - *Generalization*

Two classes $C1$ and $C2$ can be integrated by generalizing the two classes to form the superclass C .

- $\text{type}(C) = \text{type}(C1) \cap \text{type}(C2)$
- $\text{extension}(C) = \pi_{\text{type}(C)} [\text{extension}(C1) \cup_0 \text{extension}(C2)]$
- $\text{world}(C) = \text{world}(C1) \cup \text{world}(C2)$

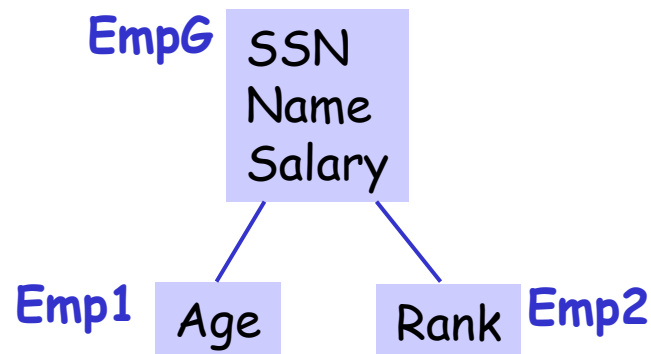
Generalization Example

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

Export Schemas

Emp1: SSN	Emp2: SSN
Name	Name
Salary	Salary
Age	Rank

Integrated Schemas

EmpG: SSN	EmpO: SSN
Name	Name
Salary	Salary
	Age
	Rank

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)

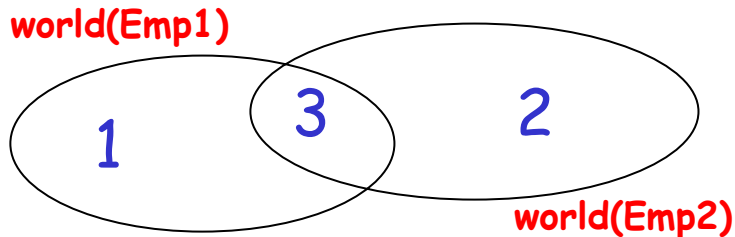
Global
Query

```
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.

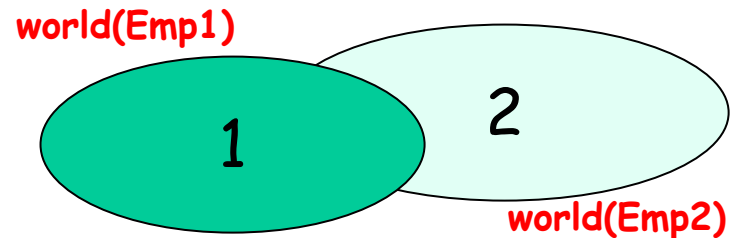
Strategy 1 (based on Salary)

part. 1: world(Emp1) - world(Emp2)
part. 2: world(Emp2) - world(Emp1)
part. 3: world(Emp1) \cap world(Emp2)



Strategy 2 (based on Age)

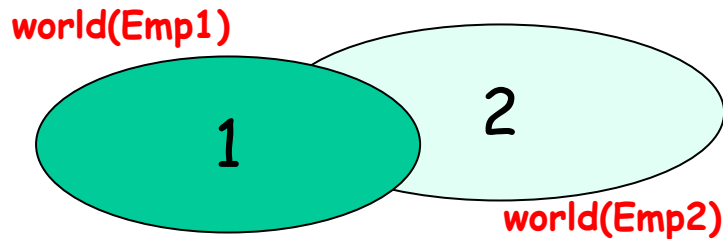
part. 1: world(Emp1)
part. 2: world(Emp2) - world(Emp1)



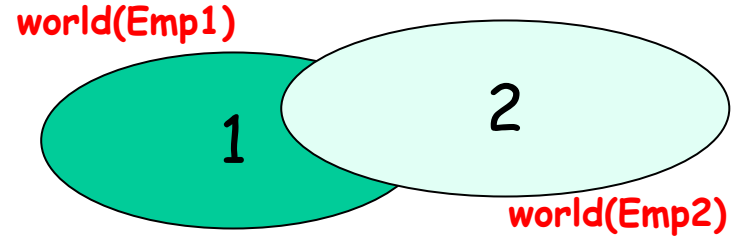
We use Strategy 1 since it is the finest partition among all the partitions.

Query Modification (2)

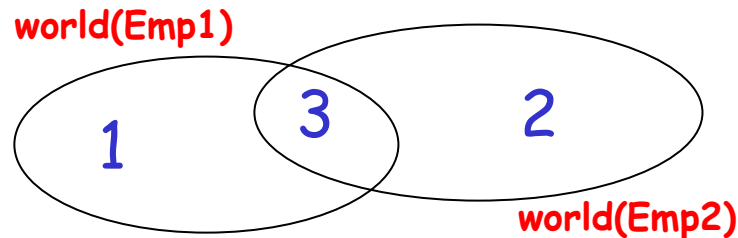
Strategy 1:



Strategy 2:



Use finer partition:



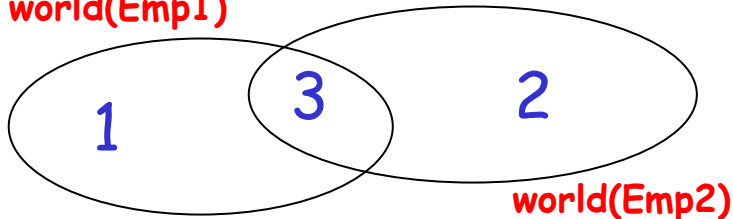
Query Modification (3)

Global Query:

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

Partition:

world(Emp1)



STEP 2: Obtain a query for each subset in the chosen partition.

part. 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)

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

part. 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

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:

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

```
Select Emp1.Name
From Emp1
Where Emp1.Salary > 80,000 and
      Emp1. Age > 35 and
      Emp1.SSN NOT IN X
```

```
Insert INTO X
Select Emp2.SSN
From Emp2)
```



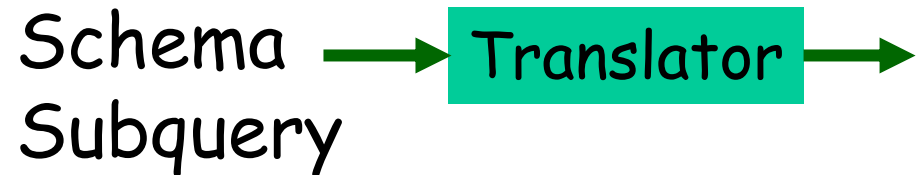
Query Modification (5)

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 \neq
Local Query Language

THEN Export
Schema \longrightarrow Translator \longrightarrow Local
Subquery Query
Language



```
graph LR; A[Export Schema Subquery] --> B[Translator]; B --> C[Local 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

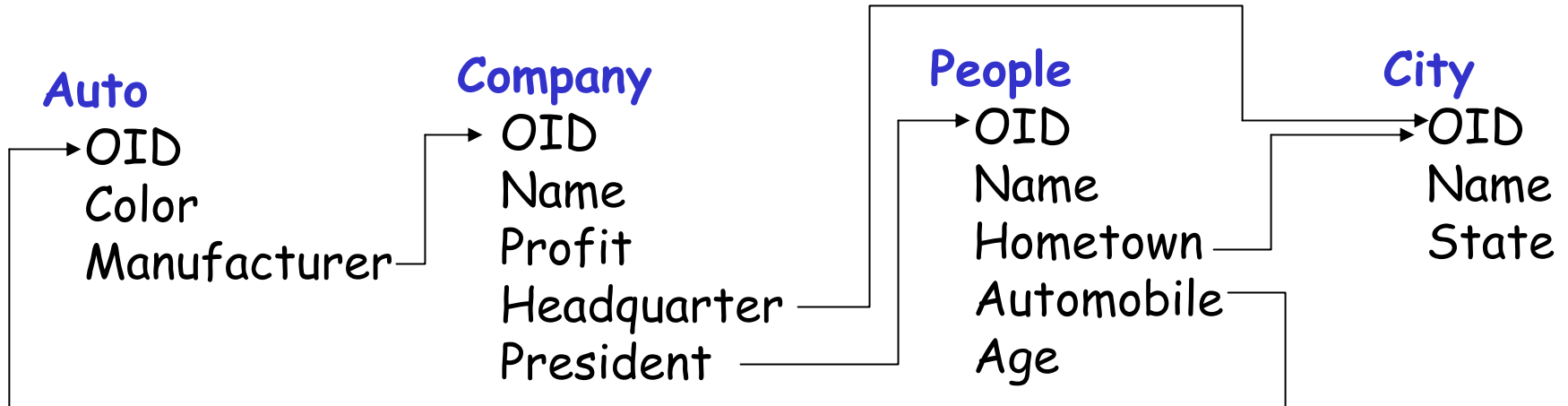
Translation Techniques (2)

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:



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)

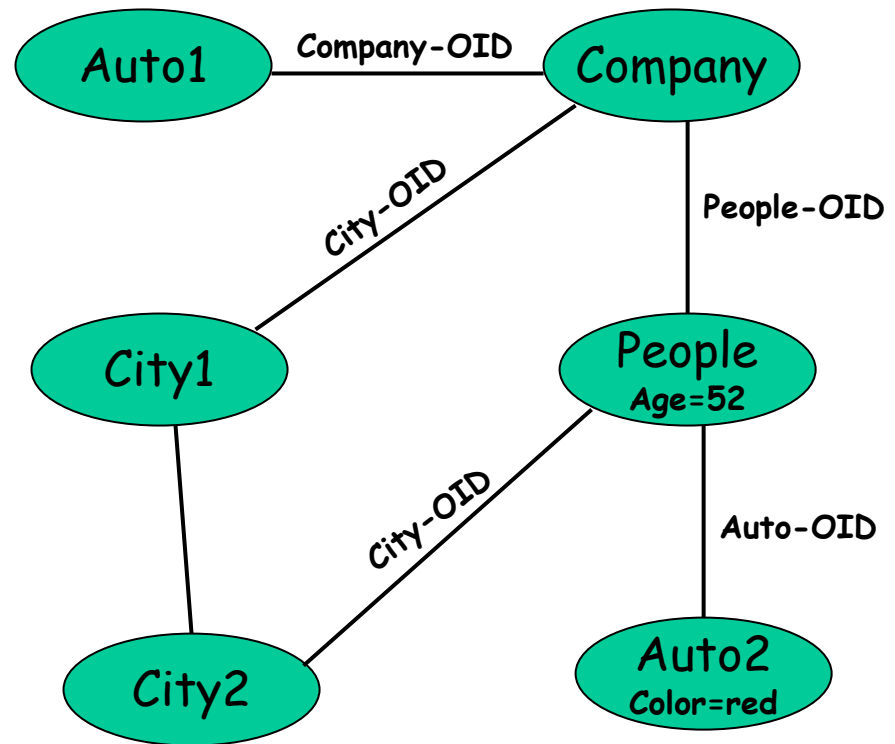
City(City-OID, Name, State)

Relational-to-OO Example (1)

Global Query:

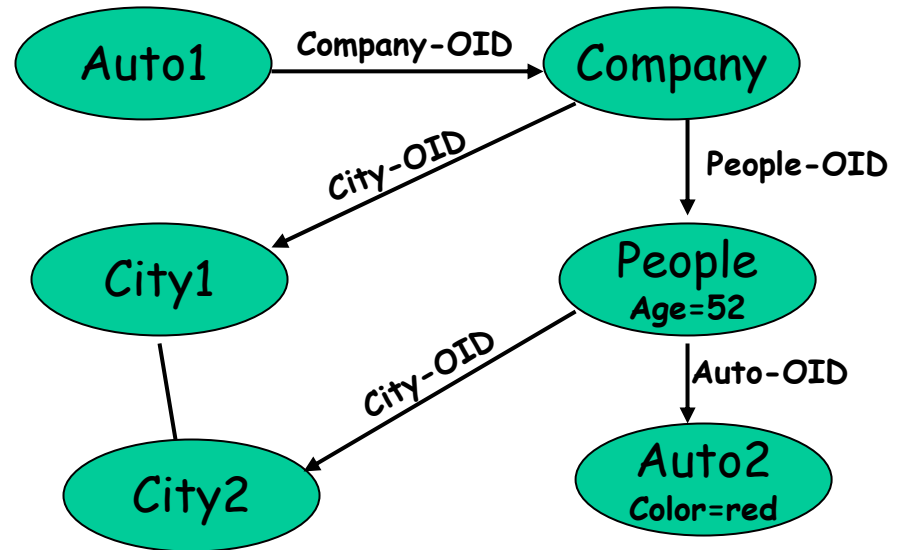
```
Select Auto1.*
From Auto Auto1, Auto Auto2,
Company, People,
City City1, City City2
Where Auto1.Company-OID =
Company.Company-OID AND
Company.People-OID =
People.People-OID AND
People.Age = 52 AND
People.Auto-OID =
Auto2.Auto-OID AND
Auto2.Color = "red" AND
People.City-OID =
City1.City-OID AND
City1.Name = 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., $\text{world}(\text{Emp1}) \cap \text{world}(\text{Emp2})$)
on page 108

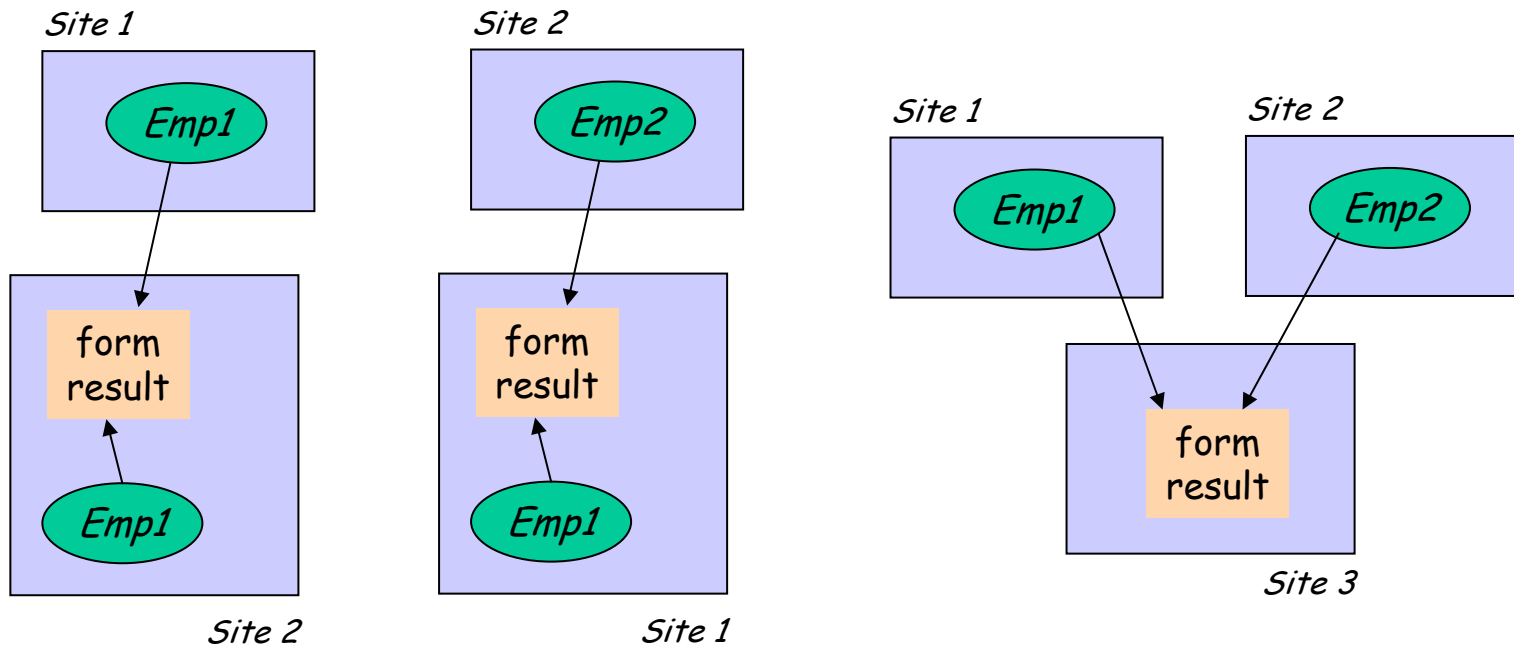
```
Select  Emp1.Name, Emp2.Rank
From    Emp1, Emp2    /* access two databases
Where   sum(Emp1.Salary, Emp2.Salary) > 80,000  AND
        Emp1.Age > 35  AND
        Emp1.SSN = Emp2.SSN
```

→ 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

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

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

Example: Consider the select operation

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

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:

$$C1 \bowtie_o C2 = C1-O \cup_o C2-O \cup_o (C1-C \bowtie_{OID} C2-C)$$

C1-O: Those tuples of C1 that have no matching tuples in C2 (private part)

C1-C: Those tuples of C1 that have matching tuples in C2 (overlap part)

$$\sigma_{A \text{ op } a} (C1 \bowtie_o C2) = \sigma_{A \text{ op } a} (C1-O) \cup_o \sigma_{A \text{ op } a} (C2-O) \cup_o \sigma_{A \text{ op } a} (C1-C \bowtie C2-C)$$

Distribution of Selections (1)

$$\bar{b}_{A \text{ op } a} (C1 \bowtie_o C2) = \bar{b}_{A \text{ op } a} (C1-O) \cup_o \bar{b}_{A \text{ op } a} (C2-O)$$

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

*When can we distribute
b over \bowtie ?*

Expensive operation

Distribution of Selection (2)

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

1. **$f(A1, A2) \text{ op } a \equiv A1 \text{ op } a \text{ AND } A2 \text{ op } a$** :

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

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

2. **$f(A1, A2) \text{ op } a \equiv f(A1 \text{ op } a, A2 \text{ op } a) \text{ op } a$** :

$$\bar{b}_{A \text{ op } a}(C1-C \bowtie C2-C) = \bar{b}_{A \text{ op } a}(\bar{b}_{A1 \text{ op } a}(C1-C) \bowtie \bar{b}_{A2 \text{ op } a}(C2-C))$$

Example: $\text{sum}(\text{Emp1-C.Salary}, \text{Emp2-C.Salary}) < 30K$
 $\equiv \text{sum}(\text{Emp1-C.Salary} < 30K,$
 $\text{Emp2-C.Salary} < 30K) < 30K$

Distribution of Selection (3)

3. $f(A1, A2) \text{ op } a \equiv f(A1 \text{ op}' a, A2 \text{ op}' a) \text{ op } a$:

$$\bar{\sigma}_{A \text{ op } a}(C1 - C \bowtie C2 - C) = \bar{\sigma}_{A \text{ op } a}(\bar{\sigma}_{A1 \text{ op}' a}(C1 - C) \bowtie \bar{\sigma}_{A2 \text{ op}' a}(C2 - C))$$

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

4. **No improvement is possible:**

Example: $\text{sum}(\text{Emp1-C.Salary}, \text{Emp2-C.Salary}) > 30\text{K}$

Distribution Rules for δ over \bowtie

$$\delta_{A \text{ op } a}(C1-C \bowtie C2-C)$$

A \ op	>	\geq	\leq	<	=	\neq	in	Not in
sum(A1, A2)	4	4	2	2	3	4	4	4
avg(A1, A2)	4	4	2	2	3	4	4	4
max(A1, A2)	4	4	1	1	3	4	4	4
min(A1, A2)	1	1	4	4	3	4	4	4

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.