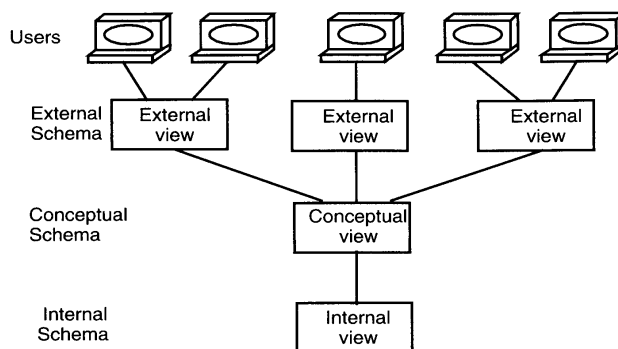


Information Integration

DBMS Standardization



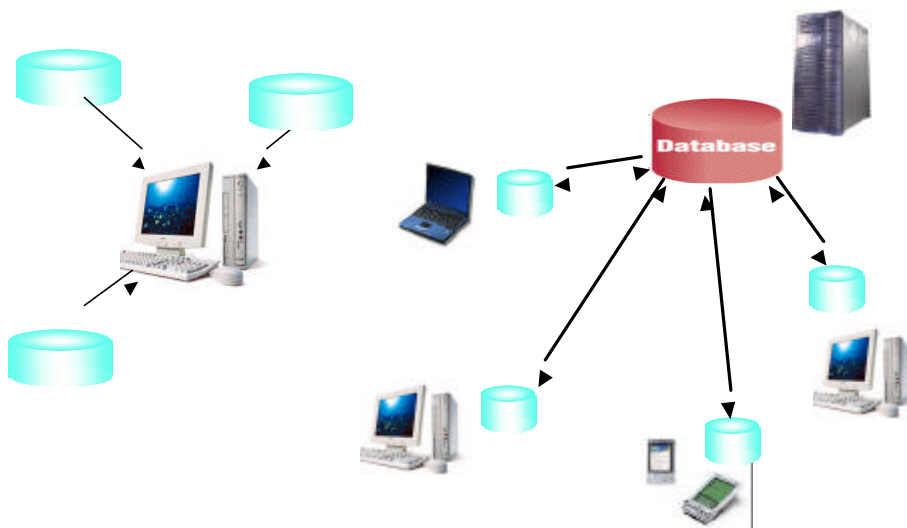
The ANSI/SPARC Architecture

DBMS Standardization

- At the lowest level of the architecture is the *internal view*, which deals with the physical definition and organization of data.
- At the other extreme is the *external view*, which is concerned with how users view the database.
- Between these two ends is the *conceptual schema*, which is an abstract definition of the database. It is the „real world” view of the enterprise being modeled in the database.

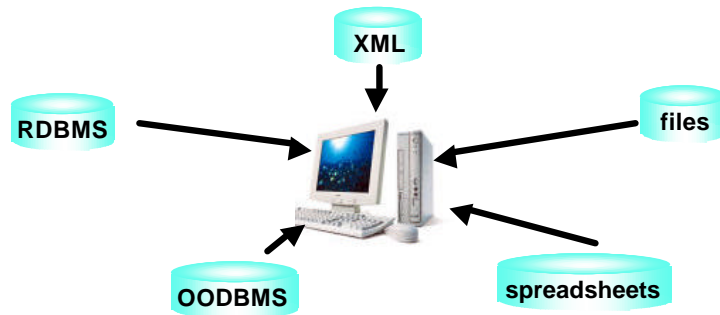
3

Integration and Distribution



4

Data Heterogeneity and Distribution



- autonomy
- distribution
- heterogeneity

integration

- federated systems
- mediated systems
- data warehouse systems

5

Autonomy

Tight integration

A single-image of the entire database is available to any user who wants to share the information, which may reside in multiple databases. From the users' perspective, the data is logically centralized in one database.

Semiautonomous systems

The DBMSs can operate independently. Each of these DBMSs determine what parts of their own database they will make accessible to users of other DBMSs.

Total isolation

The individual systems are stand-alone DBMSs, which know neither of the existence of the other DBMSs nor how to communicate with them.

6

Software and Data Heterogeneity

- **Different functionality of programs managing data**
 - e.g., SQL enabled data sources, procedural processing, triggers, transactions
- **Different network protocols**
 - DecNET, TCP/IP, IPX/SPX, LU6.2
- **Different data models**
 - network, hierarchical, relational, object-oriented, object-relational, semistructured

7

Motivation of data integration

Aims of information technology:

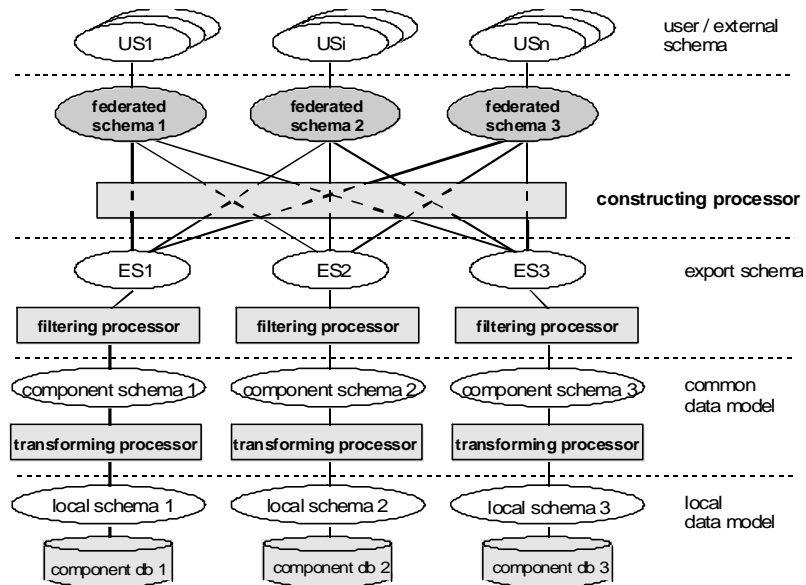
- To help workers in their everyday business activity and improve their productivity – clerical data processing tasks
- To help knowledge workers (executives, managers, analysts) make faster and better decisions – decision support systems
- **Two types of applications:**
 - Operational applications
 - Analytical applications

Motivation of data distribution

- To increase system's performance
- To increase data availability and safety

8

Federated DataBase System



9

Federated DataBase System

- **Transforming processor**
 - maintaining mappings between local and component schema elements
 - translation of commands from a federated query language to a query language of a component database
 - translation of data from a local to common data format
- **Filtering processor**
 - controls the set of operations that are issued for a component schema using the information about data visibility and access control specified in an export schema

10

Federated DataBase System

- **Constructing processor**
 - integrating different information sources by resolving inconsistencies and conflicts between them
 - determining the set of data sources capable to answer a given query that was issued and formulated in terms of a federated schema
 - decomposing, optimising, and transforming the query into local queries, that is queries for each of the data sources
 - sending each local query to appropriate data source
 - receiving query results from data sources, translating, filtering, and merging these results to form a global result
- **Loosely coupled FDBS**
- **Tightly coupled FDBS**

11

Loosely coupled FDBS

- User is responsible for creating its own federated schema
- Administrator of a FDBS does not have any control on a user's federated schema, such a schema is created as a view build on different export schemas
- In order to be able to define such a view, a user has to know which export schema are available and has to understand the structure of these export schemas
- Multiple federated schemas can exist at the same time, and they can be created and dropped at any time
- Two or more users may build their own views to access the same information from the same local databases, these users will not be aware of the existence of other views -> duplicate work

12

Loosely coupled FDBS

- Local databases have high degree of autonomy
 - changes in source schemas can be easily performed
 - it may be difficult to detect by a user that an export schema was changed
- Only querying data is allowed
- Updates of local data through a federated schema are not allowed because different users might define different mappings between views and export schemas and might define different update policies for the same data

13

Tightly coupled FDBS

- FDBS administrator is responsible for creation and maintenance of one or more global federated schemas.
- Such a global schema integrates all the available export schemas. The idea behind building a global federated schema is to provide location transparency. Users simply use the provided global schema without knowing the location of data sources.
- Local databases have less autonomy than in a loosely coupled FDBS. During the creation of a global federated schema, a FDBS administrator negotiates with local database administrators the structure of their export schemas.
- Changes in source schemas are difficult to propagate into a federated global schema as any change in a source and subsequently in an export schema results in the creation of a federated schema from scratch.
- Updates of local data through a federated schema are at least partially supported.

14

FDBS – Features

- User of a FDBS queries data that are always up-to-date, as queries operate directly on component databases, which are the subjects of integration
 - Only little additional storage for information is required in a FDBS
 - User can query any data that is accessible by a federated schema
 - ad-hoc queries
-
- The results of a query may arrive with a long delay caused by a slow network, or a low response time of data sources
 - The decomposition and translation of a query as well as merging the results of a query incur additional time overhead
 - Queries coming from a federated system may interfere with queries executed locally in component databases. In a consequence, federated queries may slow down the execution of the local queries
 - Some of the component databases may be temporarily unavailable, thus making the query results incomplete or unavailable

15

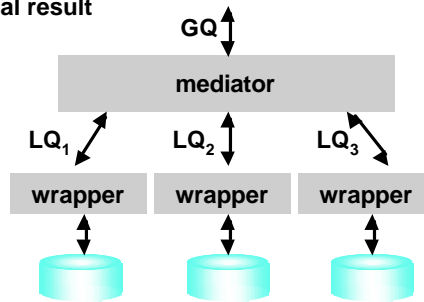
FDBS – Applicability

- source data in component databases change very often
- a user of an integrated system needs the information that is always up to date
- a user can afford the delay of receiving query results

16

Mediated System

- 1 global query is decomposed and transformed to local queries for each of the data sources
- 2 the result of each of the local queries is translated, filtered, and merged to form a global result



- **Mediation:** a mediator is a software component that supports a virtual database, which the user may query as if it were materialized (physically constructed like a warehouse). The mediator stores no data of its own. Rather, it translates the user's query into one or more queries to its sources. The mediator then synthesizes the answer to the user's query from the responses of those sources, and returns the answer to the user

17

Mediated System – Features

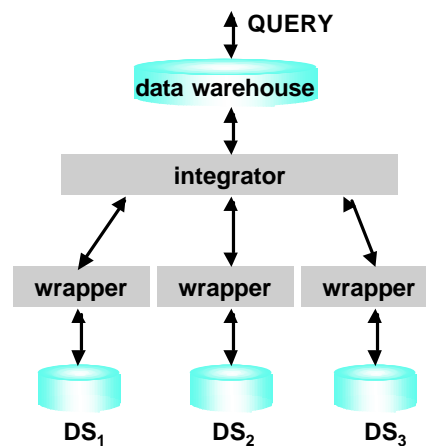
- A user queries data that are always up-to-date and only little additional storage for information is required in the system
- The delay in getting the result of a query, the interference of global queries with local queries in data sources, temporary unavailability of some data sources, like in a FDBS
- The set of possible queries that can be translated by a wrapper and sent to a data source is limited by the functionality of a wrapper, i.e. by the content of its mapping table

18

Data Warehouse System

- 1 data are extracted, filtered, merged, and stored in a central repository - data warehouse
- 2 queries are issued for data warehouse

- data redundancy \Rightarrow the content of a DW has to be kept up to date
- queries operate on a local, centralised repository \Rightarrow access time to data reduction
- DW is independent of data sources, that can be unavailable



19

Data Warehouse - Features

- Queries operate on a local centralised data repository, that reduces access time to data
- Queries need not be decomposed into different formats and their results need not be integrated because data are stored in a uniform format in a central repository
- Users execute queries in a data warehouse and their queries do not interfere with queries issued in data sources
- It is possible to query data that originally were not stored in a database, provided that these data have been previously integrated into the system and loaded into a warehouse
- Since a data warehouse usually makes available additional information not stored directly in data sources (e.g. summaries, averages), users can make use of this enriched information
- A data warehouse is independent of data sources. Consequently, even though data sources were unavailable, the users of a data warehousing system could get the information that has already been loaded into a warehouse

20

Data Warehouse - Features

- Because data from various sources are loaded to a warehouse, a warehouse becomes obsolete when source information change. In order to keep a warehouse up do date additional mechanisms must be implemented.
- A data warehouse administrator has to know in advance which sources of data should be integrated, which data should be extracted from these sources and loaded into a warehouse.
- A user can query only these data that have been previously loaded into a warehouse.
- Since a warehouse stores locally all its data, additional storage space is required. Moreover, the size of the storage space is very large as a warehouse contains data from a number of data sources

21

Problems of Information Integration

Example:

The AAAI Automobile Co. has 1000 dealers each of which maintains a database of their cars in stock. AAAI wants to create an integrated database containing the information of all 1000 sources. The integrated database will help dealers locate a particular model if they don't have one in stock. It also can be used by corporate analysts to predict the market and adjust production to provide the model most likely to sell¹

- The 1000 dealers do not all use the same database schema:

```
Cars (serialNo, model, color, autoTrans, cdPlayer, ...)
```

or

```
Autos (serialNo, model, color)
```

```
Options (serialNo, option)
```

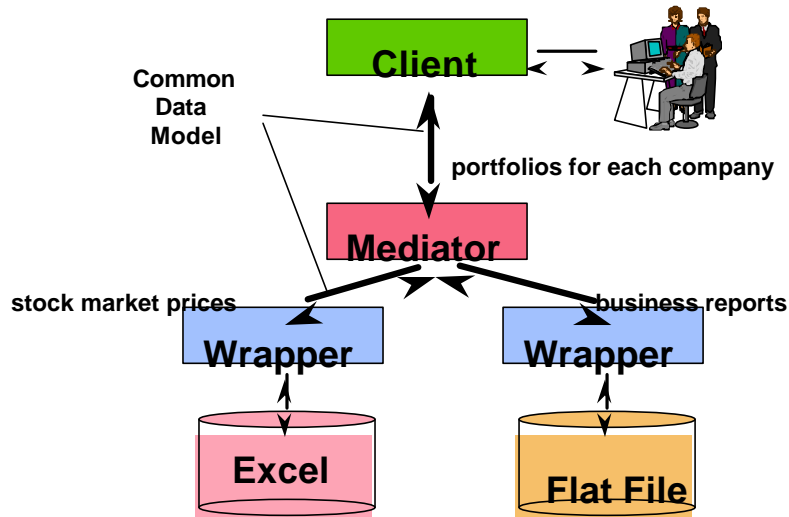
22

Problems of Information Integration

- **Schema difference**
- **Data type differences:** numbers may be represented by character strings of varying length at one source and fixed length at another
- **Value differences:** the same concept may be represented by different constants at different sources (BLACK, BL, 100, etc), salary= 1000 PLN or GBP?
- **Semantic differences:** Terms can be given different interpretations at different sources (Cars includes trucks or not)
- **Missing values:** a source may not record information of a type that all of the other sources provide
- **Different representation** of the same kind of information, even within the same data model
 - address_street, address_city, address_zip
 - address
- **Different values** of the same meaning
 - NY, New York
- **Different meaning** of the same value
 - prof. -> professional
 - prof. -> professor1

23

The Wrapper and Mediator Architecture



24

The Wrapper and Mediator Architecture

- A mediator supports a virtual view, or collection of views, that integrates several sources in much the same way that the materialized relation(s) in a data warehouse integrate sources.
- The mediator doesn't store any data!!!

Example:

Let us consider the same scenario. The mediator integrates the same two data sources into a view that is a single relation with the schema:

```
AutosMed(serialNo,model,color,autoTrans,dealer)
```

25

The Wrapper and Mediator Architecture

Assume the user asks the mediator about the red cars:

```
select serialNo, model from AutosMed  
where color = 'red';
```

The mediator forwards the same query to each of the two wrappers

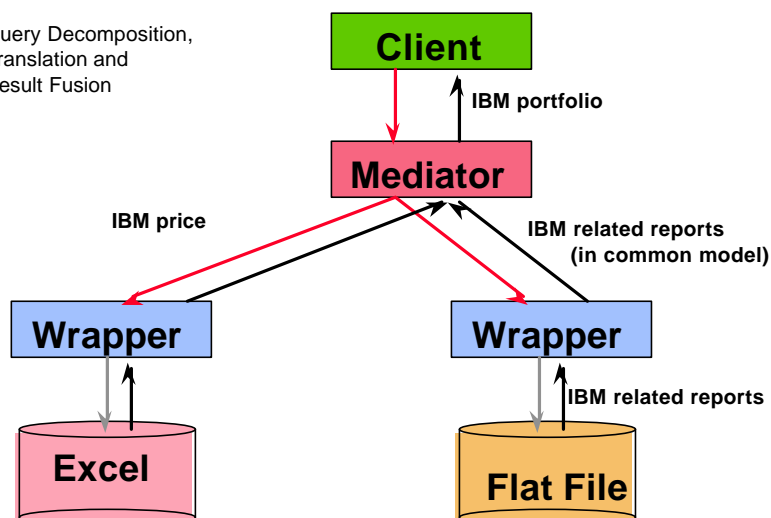
- (1) `select serialNo, model from Cars where color='red';`
- (2) `select serialNo, model from Autos where color='red';`

The mediator can take the union of answers and return the result to the user.

26

The Lazy Integration Approach

Query Decomposition,
Translation and
Result Fusion



27

Wrappers in Mediator-Based Systems

- In a data warehouse system, the source extractors consist of:
 - one or more queries built-in that are executed at the source to produce data for the data warehouse
 - communication mechanisms, so that wrapper can:
 - pass ad-hoc queries to the source
 - receive responses from the source
 - pass information to the warehouse
- Mediator systems require more complex wrappers - the wrapper must be able to accept a variety of queries from the mediator and translate any of them to the terms of the source.

28

Wrappers in Mediator-Based Systems

- A systematic way to design a wrapper that connects a mediator to a source is to classify the possible queries that the mediator can ask into templates, which are queries with parameters that represent constants.
- The mediator can provide the constants, and the wrapper executes the query with the given constants.
- **T \rightarrow S** the template T is turned into the source query S

Example:

The source of dealer1:

```
Cars (serialNo, model,color,autoTrans, cdPlayer, ...)
```

Assume we use the mediator with schema:

```
AutosMed(serialNo,model,color,autoTrans, dealer)
```

How the mediator could ask the wrapper for cars of a given color?

The template:

```
select * from AutosMed where color= '$c';  $\rightarrow$ 
```

```
select serialNo, model color, autoTrans, 'dealer1'  
from Cars where color='$c';
```

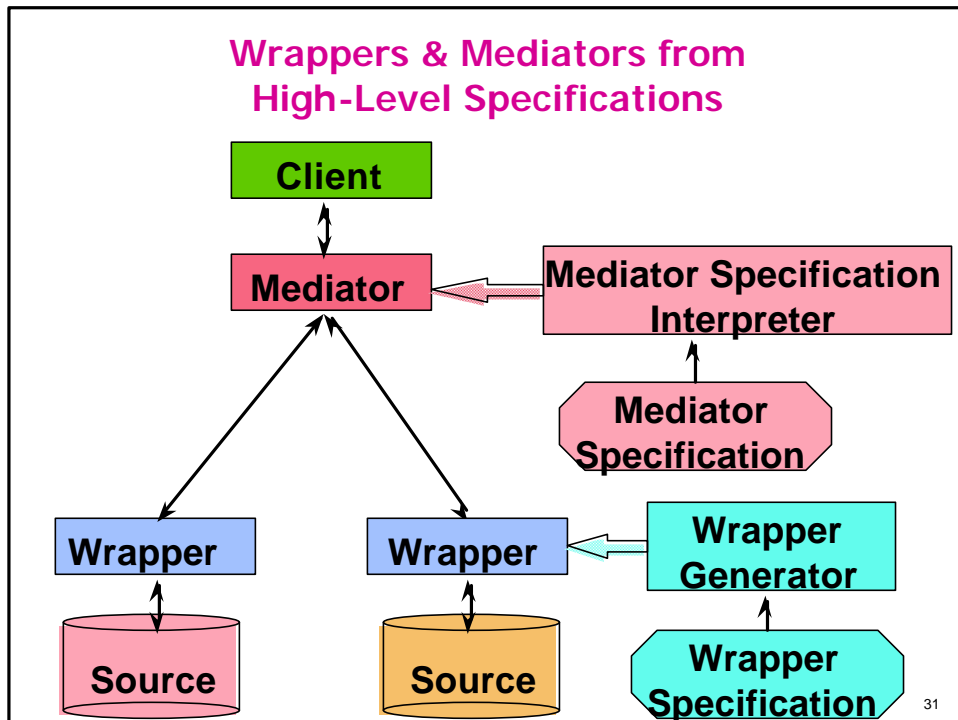
29

Wrapper Generators

- The template defining a wrapper must be turned into code for the wrapper itself - the software that creates the wrapper is called a wrapper generator
- The wrapper generator creates a table that holds the various query patterns contained in the templates, and the source queries that are associated with each.
- A driver is used in each wrapper. The task of the driver is to:
 - accept a query from the mediator
 - search the table for a template that match the query
 - send the query to the source using a communication mechanism
 - process the result, if necessary, and then returned to the mediator

30

Wrappers & Mediators from High-Level Specifications



Filters

- **Complex template**

```
select * from AutoMed
where color= '$c' and model = '$m'; P
select serialNo, model color, autoTrans, 'dealer1'
from Cars where color='$c' and model ='$m';
```
- **Wrapper filter** approach - if the wrapper has a template that returns a superset of what the query wants then it is possible to filter the result at the wrapper

Example:

Given the template

```
select * from AutoMed where color= '$c';
```

The mediator needs to find blue Gobi model car:

```
select * from AutoMed where color= 'blue' and model='Gobi';
```

- use the template with \$c=blue to find all blue cars
- store the result in the temporary relation Temp
- select from Temp the Gobi's and return the result

32

Challenge: Sources Without a Well-Structured Schema

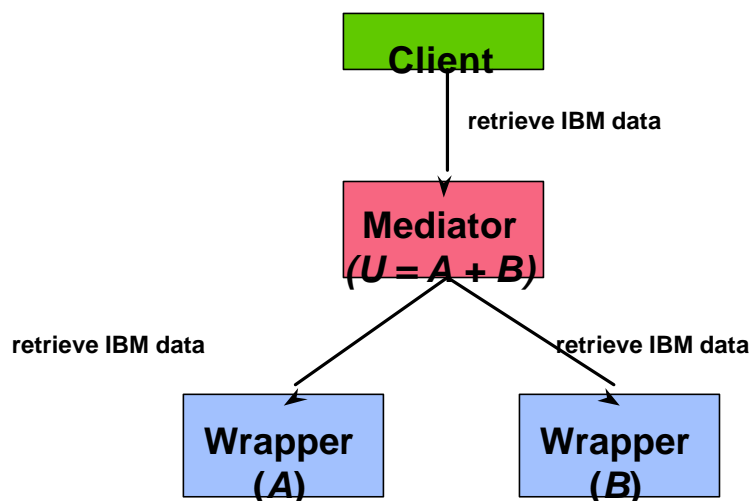
- semistructured
 - irregular
 - deeply nested
 - cross-referenced
- incomplete schema knowledge
 - autonomous
 - dynamic

Examples

- HTML pages
- SGML documents
- chemical structures
- bibliographic information
- results of the integration process

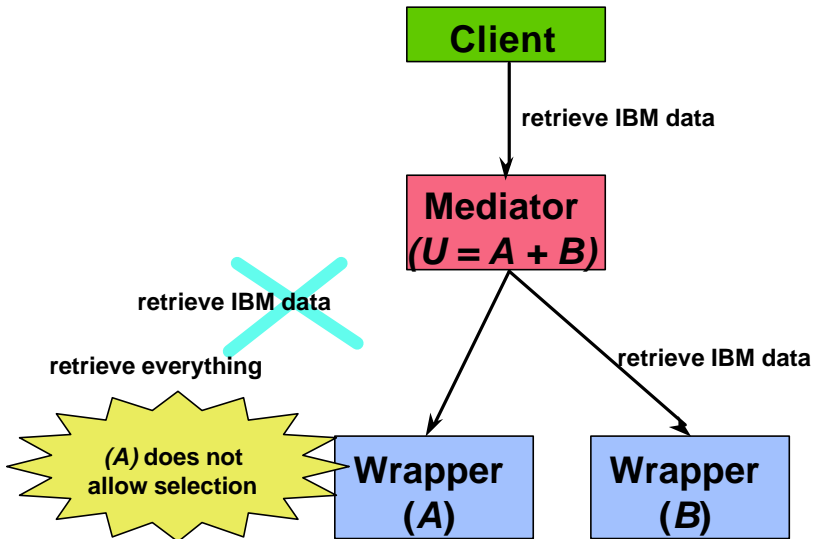
33

Challenge: Different and Limited Source Capabilities



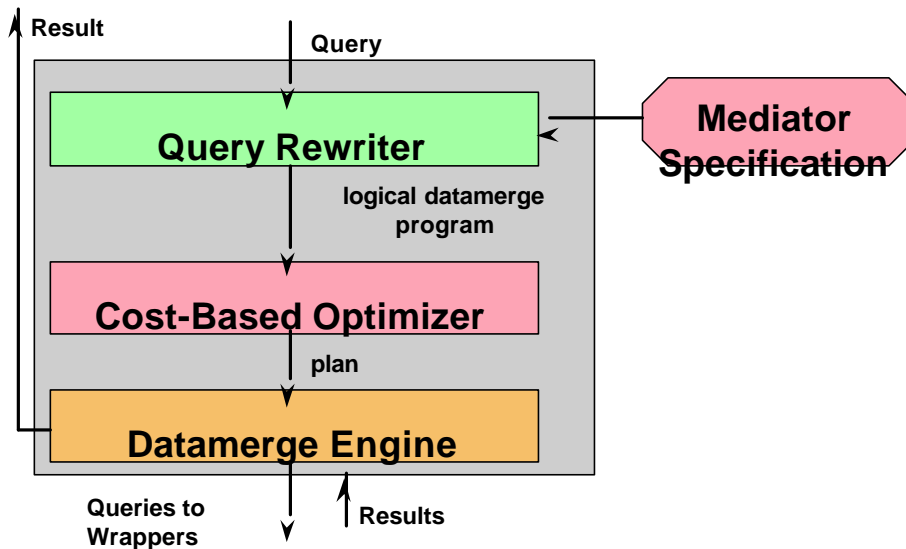
34

Mediator has to Adapt to Query Capabilities of Sources



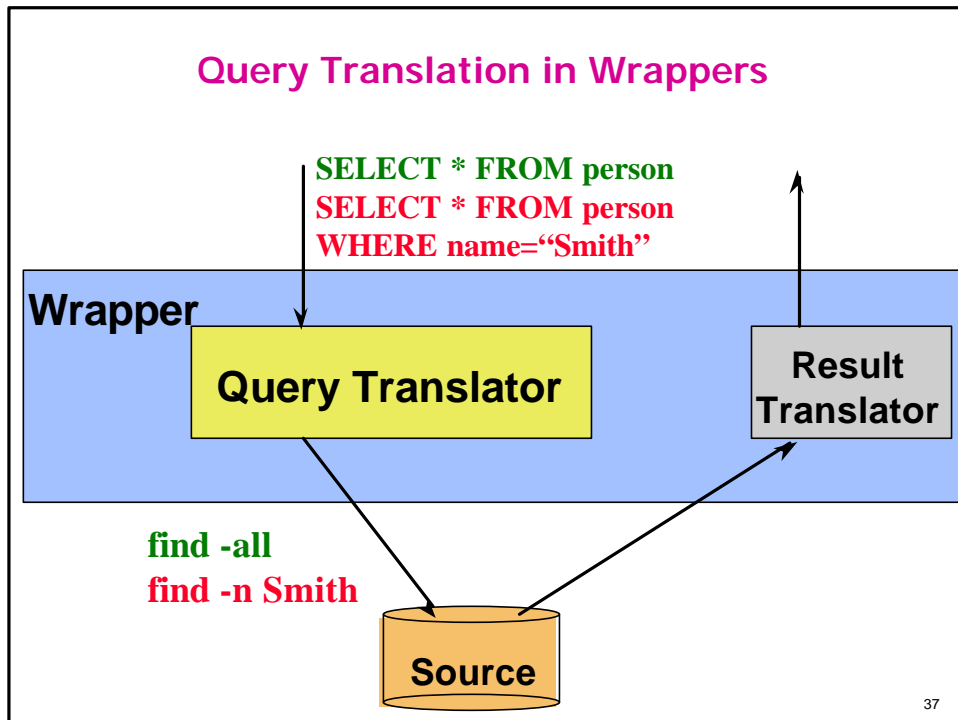
35

Mediator Specification Interpreter Architecture

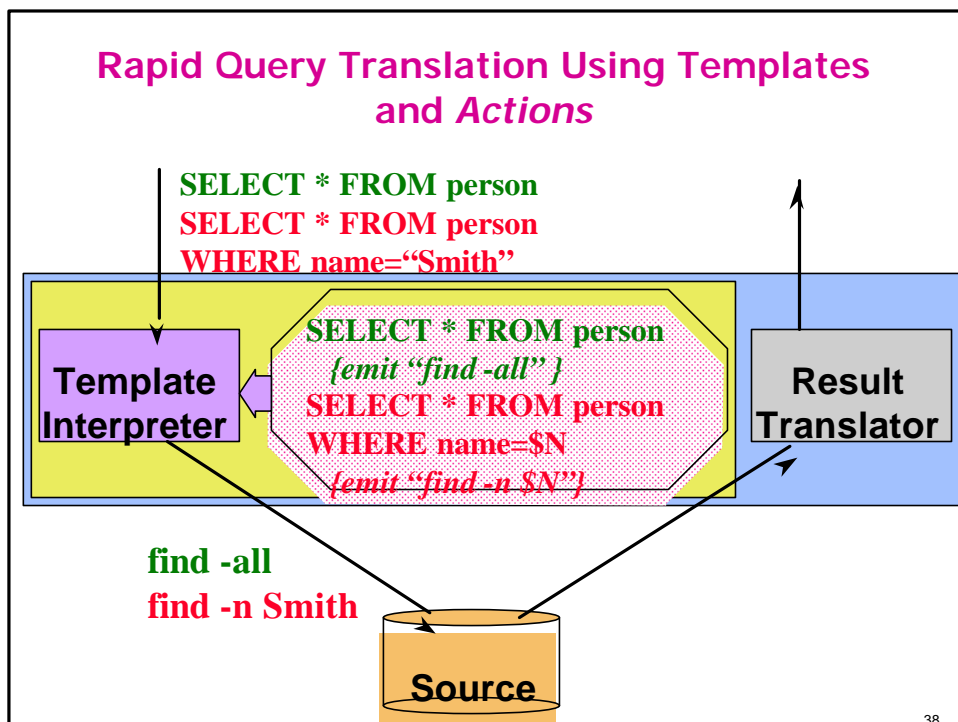


36

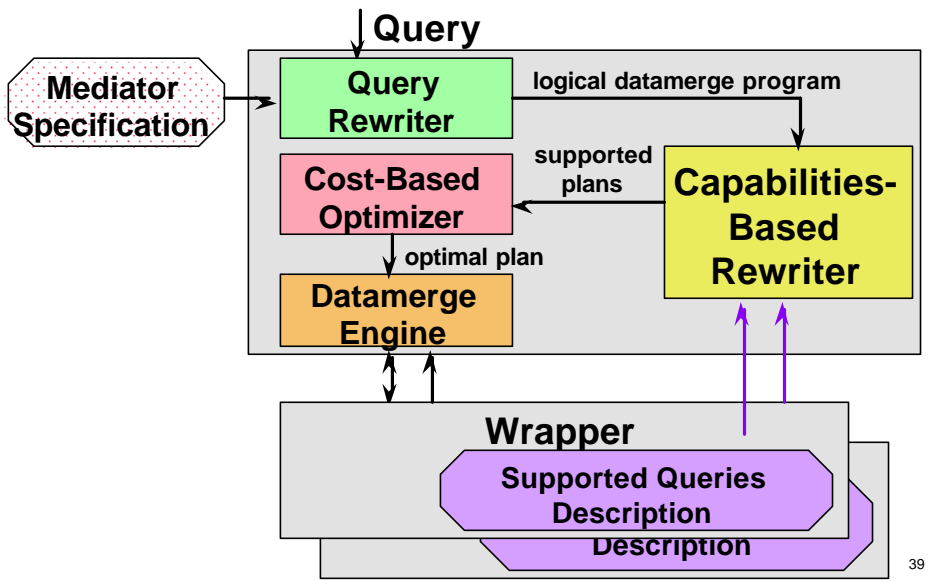
Query Translation in Wrappers



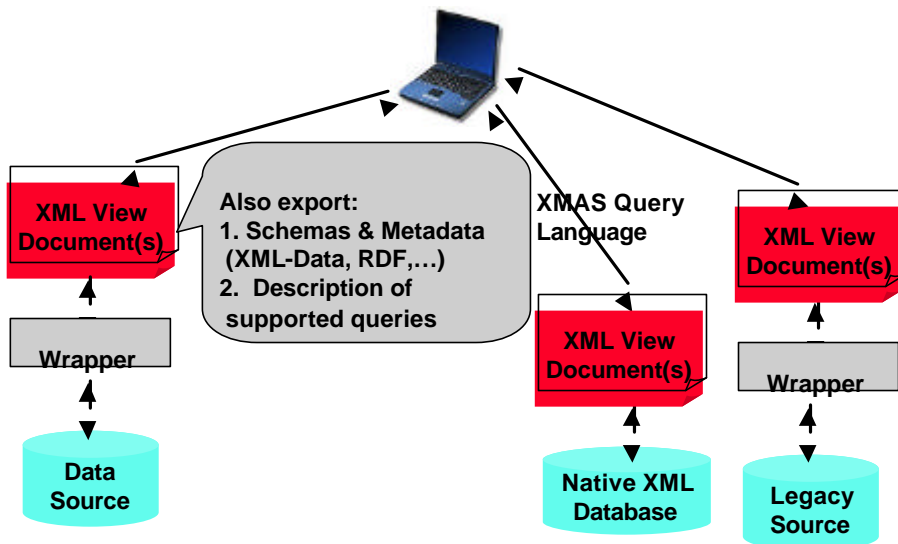
Rapid Query Translation Using Templates and Actions



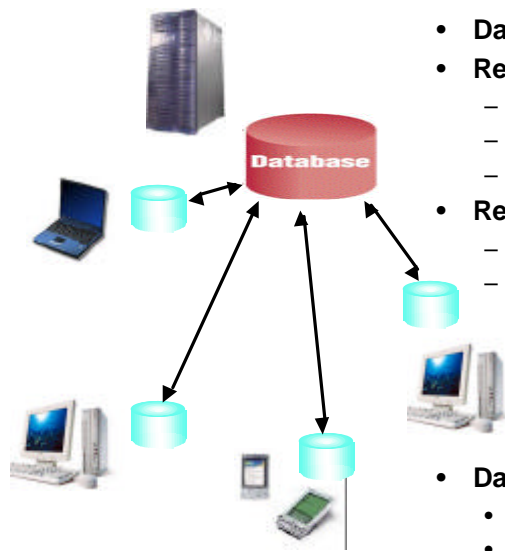
Capabilities-Based Rewriter in Mediator Architecture



Web emerges as a Distributed DB and XML as its Data Model



Data distribution



- Data synchronization problem
- Replication
 - immediate
 - lazy
 - on demand
- Refreshing
 - full
 - incremental
- Data partitioning problem
 - where
 - how (vertical, horizontal)