Databases 2

Lecture VIII

Alessandro Artale

Faculty of Computer Science – Free University of Bolzano

Room: 221

artale@inf.unibz.it

http://www.inf.unibz.it/∼artale/

2003/2004 – First Semester

Summary of Lecture VIII

- **Information Integration**
  - Data Warehouse.
    - Data Extraction, Integration and Preprocessing
    - Warehouse Updates: View Maintenance
  - Mediator/Query-Driven Integration.
    - Data Extraction;
    - Queries Templates and Filtering
- Mediators Vs. Data Warehouses.

Why Information Integration

Problem: Need to access data stored in two or more databases (**Information Sources**).

Goal: Querying the sources as a unit providing an **Integrated View** of them (possibly virtual).

Type of Sources

The sources to be integrated may be:

- Conventional Databases;
- Collections of Web Pages;
- Textual documents;
- Digital Libraries;
- Scientific Databases;
- etc...

Problems: Heterogeneous Information Sources

Heterogeneous Sources. Sources dealing with the same kind of data but differ in:

Schema. Data can be structured in different Relations and equivalent Attributes can have different names.

E.g. let us take two car dealers of the same company. Dealer1 stores cars in a single relation with Boolean values for options:

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

Dealer2 separates `options` in a second Relation:

`Autos(serial, model, color)`

`Options(serial, option)`.

Data Type. Integer Vs. Strings; Variable Vs. Fixed lenght.

Value. Different constants for equivalent values (e.g, `Blak` Vs. `BK` for black).

Problems: Heterogeneous Information Sources (Cont.)

Data Semantics. Similar terms can have different meanings. E.g. Dealer1 can include trucks in the `Cars` Relation while Dealer2 considers only automobile in its `Autos` Relation.

Missing Values. A source might not record data needed at the integrated level. We can use `NULL`'s values. E.g., Dealer1 does not provide data for 'ABS' option.

Methods for Integrations

We consider two common approaches:

1. **Data-Warehouse.** A repository of information gathered from multiple sources, stored under a *Unified Schema* at a *Single Site*.
2. **Mediation/Query-Driven Integration.** A Mediator supports a *Virtual Database*: No data is stored, rather, data is the result of querying the sources (Integration on Demand).

Summary

- Information Integration
  - **Data Warehouse.**
    - Data Extraction, Integration and Preprocessing
    - Warehouse Updates: View Maintenance
  - Mediator/Query-Driven Integration.
    - Data Extraction;
    - Queries Templates and Filtering
- Mediators Vs. Data Warehouses.

What's a Data Warehouse?
A Practitioner Viewpoint

"*A data warehouse is simply a single, complete, and consistent store of data obtained from a variety of sources and made available to end users in a way they can understand and use it in a business context.*"

– Barry Devlin, IBM Consultant

What is a Data Warehouse?
An Alternative Viewpoint

”A Data Warehouse is a:

–subject-oriented,

–integrated,

–time-varying,

–non-volatile

collection of data that is used primarily in organizational decision making.”

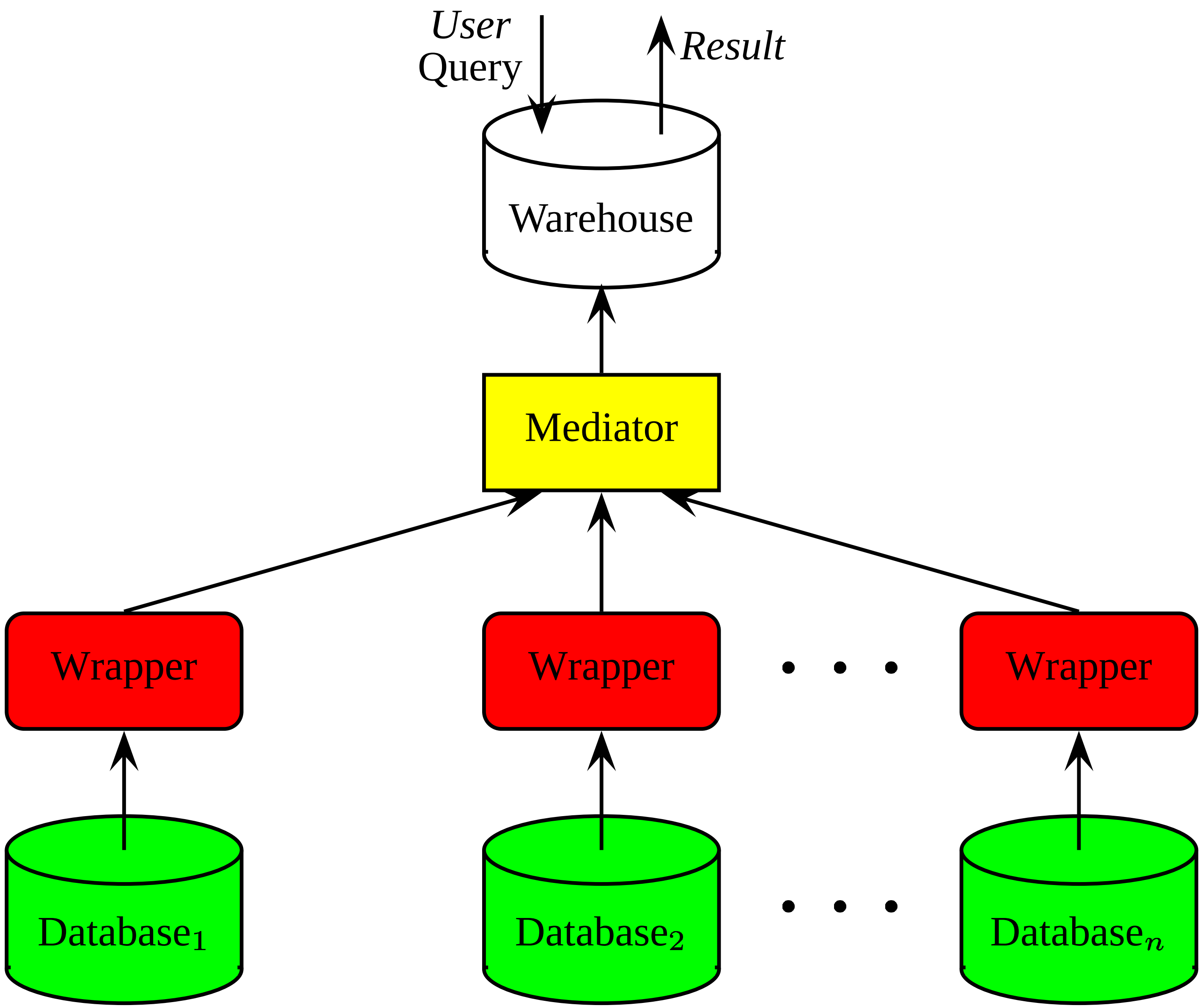
– W.H. Inmon, Building the Data Warehouse, 1992

Data Warehouse Main Features

A repository of information gathered from multiple sources, stored under a *Unified Schema* at a *Single Site*.

- **Subject-Oriented.** Organized by subject, not by application.
- **Used** for decision making, data analysis, data mining, historical data analysis (Time attributes are important).
- **Non-Volatile.** Data is persistently stored.
- **Updates** infrequent, at least with respect to Queries.
- **Optimized** differently from transaction-oriented DB's.
- **Examples.**
  - Summary of the transactions of a SuperMarket;
  - Complete client histories at insurance firm;
  - Financial information and portfolios.

Data Warehouse Architecture



Issues in Data Warehousing

1. Warehouse Design: Conceptual Models for Warehouse Specification
2. Data Extraction: Building Wrappers
3. Data Integration: Building Mediators
4. Data Preprocessing: Data Cleansing and Merging
5. Maintenance: How to Update a Warehouse
6. Optimization Techniques.

Summary

- Information Integration
  - **Data Warehouse.**
    - **Data Extraction, Integration and Preprocessing**
    - Warehouse Updates: View Maintenance
  - Mediator/Query-Driven Integration.
    - Data Extraction;
    - Queries Templates and Filtering
- Mediators Vs. Data Warehouses.

Data Extraction: An Example

Given the car's dealers example using the schemas:

$\texttt{Cars}(\texttt{serialNo}, \texttt{model}, \texttt{color}, \texttt{autoTransmission}, \texttt{cdPlayer}, \ldots)$

while Dealer2 separates `options` in a second Relation:

$\texttt{Autos}(\texttt{serial}, \texttt{model}, \texttt{color})$

$\texttt{Options}(\texttt{serial}, \texttt{option})$.

We wish to create a Warehouse with the schema:

$\texttt{AutosWhse}(\texttt{serialNo}, \texttt{model}, \texttt{color}, \texttt{autoTransmission}, \texttt{dealer})$

Data Extraction: An Example (Cont.)

To populate the Warehouse the Wrapper for Dealer1 issues the SQL query:

```
INSERT INTO AutosWhse(serialNo,model,color,autoTransmission,dealer)
            SELECT serialNo,model,color,autoTransmission,'dealer1'
            FROM   Cars;
```

Data Extraction: An Example (Cont.)

To extract data from Dealer2 the Wrapper needs to run two SQL's:

```
INSERT INTO AutosWhse(serialNo,model,color,autoTransmission,dealer)
          SELECT serial,model,color,'yes','dealer2'
          FROM   Autos, Options
          WHERE  Autos.serial = Options.serial AND
                 option = 'autoTransmission';

INSERT INTO AutosWhse(serialNo,model,color,autoTransmission,dealer)
          SELECT serial,model,color,'no','dealer2'
          FROM   Autos
          WHERE  NOT EXISTS (
                 SELECT *
                 FROM   Options
                 WHERE  serial = Autos.serial AND
                        option = 'autoTransmission');
```

Data Integration: The Mediator

A **Mediator** performs various operations on the relations extracted from the sources:

- Relation could be Joined or the Union of the different queries is taken as result.
- Various **Aggregations** could be computed on the resulting queries.

In our example the job of the Mediator is to compute the Union of the tuples extracted from the two sources.

Data Preprocessing: Cleansing and Merging

Data Preprocessing is done both by wrappers and mediators.

- **Data Cleansing** is the task of correcting the data.
  - Misspelled data;
  - Differences in either data types or values;
  - Dealing with NULL values;
  - etc...
- **Data Merging** is the task of recognizing whether tuples from different sources represent the same object.

Summary

- Information Integration
  - **Data Warehouse.**
    - Data Extraction, Integration and Preprocessing
    - **Warehouse Updates: View Maintenance**
  - Mediator/Query-Driven Integration.
    - Data Extraction;
    - Queries Templates and Filtering
- Mediators Vs. Data Warehouses.

Warehouse Update

Problem. Updates on source relations must propagate to the Warehouse.

- **Reconstruction.** The system is *periodically* shut down to reconstruct the Warehouse from the sources. *Disadvantages:* Long time to reconstruct the Warehouse; the Warehouse could become out-of-date.
- **Incremental Update.** The Warehouse is *periodically* updated based on the changes since the last update. *Advantage:* Short time required by the update. *Disadvantage:* complex processing.
- **Immediate Change.** The Warehouse is updated in response to each change of the sources. *Advantage:* Warehouse always up-to-date. *Disadvantage:* Too much communication and processing.

Warehouse as View

The mapping between the Warehouse and the sources is given in terms of a set of views.

- **GAV (Global-As-View) Approach:** *The Warehouse relations are Materialized Views over the information sources.*

The task of keeping up-to-date a materialized view is known as **View Maintenance**.

A *View* is a **virtual/materialized relation** derived from other relations (either base relations or views) and specified by the following command:

CREATE VIEW $<$ *view name* $>$ **AS** $<$ *view definition* $>$

where

- $<$ *view name* $>$ is the name of the view;
- $<$ *view definition* $>$ is an SQL query.

Incremental View Maintenance

- Technique adopted by current commercial DBMS. Once a materialized view is declared the DBMS is able to incrementally update the view instances when the underlying source date changes.
- We consider only *Insertions* and *Deletions*. *Updates* can be considered as deletion of the tuple followed by insertion of the updated tuple.
- Incremental view maintenance is based on the relational algebra operations that define the view.
- To handle complex relational algebra expressions we apply incremental view maintenance recursively to sub-expressions.

Incremental View Maintenance: Join

Consider the materialized view: $\mathbf{V} = \mathbf{R} \bowtie \mathbf{S}$. If

- I_R denotes the set of new tuples *Inserted* in R;
- R^{old} denotes the old instances of R;
- R^{new} denotes the new instances of R, i.e., $R^{new} = R^{old} \bigcup I_R$;
- V^{new} denotes the new instances of the view V, i.e., $V^{new} = R^{new} \bowtie S$.

Then

$$\begin{aligned} V^{new} &= (R^{old} \bigcup I_R) \bowtie S \\ &= (R^{old} \bowtie S) \bigcup (I_R \bowtie S) \\ &= \mathbf{V}^{old} \bigcup (\mathbf{I_R} \bowtie \mathbf{S}) \end{aligned}$$

Thus, to incrementally update V we need to add $(I_R \bowtie S)$ to the old content of the view. Insertions to S are handled in a similar way.

Now, assume that we *Delete* the set D_R from R. Then:

$$\mathbf{V}^{new} = \mathbf{V}^{old} - (\mathbf{D_R} \bowtie \mathbf{S}).$$

Incremental View Maintenance: Selection

Consider the materialized view: $\mathbf{V} = \sigma_\theta(\mathbf{R})$. If I_R denotes the set of new tuples *Inserted* in R, then:

$$\mathbf{V}^{new} = \mathbf{V}^{old} \bigcup \sigma_\theta(\mathbf{I_R})$$

If we *Delete* the set D_R from R, then:

$$\mathbf{V}^{new} = \mathbf{V}^{old} - \sigma_\theta(\mathbf{D_R})$$

Incremental View Maintenance: Projection

Projection is a more difficult operation to handle.

Problem Example. Consider a materialized view: $\mathbf{V} = \pi_{\mathbf{A}}(\mathbf{R})$. Let $R = R(A, B)$ with just two tuples: $(a, 2)$ $(a, 3)$. Then, $\pi_A(R) = (a)$.

If we delete the tuple $(a, 2)$ from R, then $V^{new} = V^{old} = (a)$. Thus, we **cannot** delete the tuple (a) from the view.

The reason is that the tuple (a) in the view is derived from *two* tuples in R.

Solution.

- For each tuple in a Projection we keep a count of how many times it was derived.
- If we *Delete* the set D_R from R, then, denoting by $t.A$ the projection of the tuple t along A, we find $(t.A)$ in the view and decrease the count.
- If the count for $(t.A)$ is zero the tuple is deleted from the materialized view.

Incremental View Maintenance: Projection (Cont.)

The case for *Insertion* is similar. Let I_R be the set of tuples inserted in R.

- If $(t.A)$ is already in the view: Increase the counter;
- Otherwise, insert $(t.A)$ in the view, with the counter set to 1.

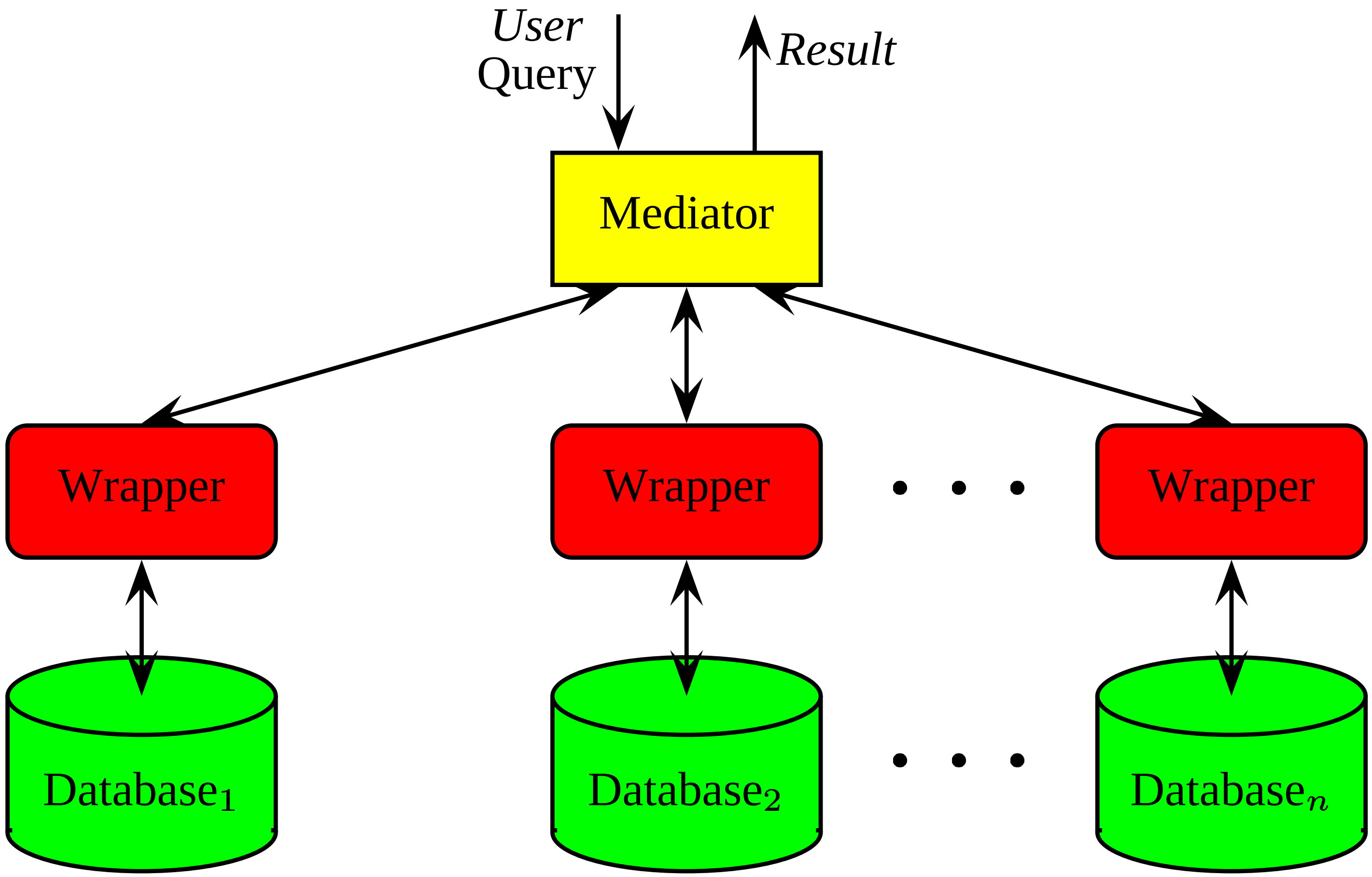
Summary

- Information Integration
  - Data Warehouse.
    - Data Extraction, Integration and Preprocessing
    - Warehouse Updates: View Maintenance
  - **Mediator/Query-Driven Integration.**
    - Data Extraction;
    - Queries Templates and Filtering
- Mediators Vs. Data Warehouses.

Mediators: A Query-Driven Integration

- **A *Mediator* supports a *Virtual Database*: No data is stored, rather, data is the result of querying the sources (*Integration on Demand*).**
- The user query is delivered by the Mediator to one or more sources.
- The Wrapper *Rewrites* the user query and get back the results to the Mediator which finally combines and integrates the results.

Mediators Architecture



Issues in Mediators-based Integration

1. Data Extraction: Building Wrappers—much more difficult than in Warehouses.
2. Data Integration: Building Mediators
3. Data Preprocessing: Data Cleansing and Merging
4. Optimization Techniques—mostly query optimization.

Summary

- Information Integration
  - Data Warehouse.
    - Data Extraction, Integration and Preprocessing
    - Warehouse Updates: View Maintenance
  - **Mediator/Query-Driven Integration.**
    - **Data Extraction;**
    - Queries Templates and Filtering
- Mediators Vs. Data Warehouses.

Data Extraction: An Example

Given the car's dealers example using the schemas:

$$\texttt{Cars}(\texttt{serialNo}, \texttt{model}, \texttt{color}, \texttt{autoTransmission}, \texttt{cdPlayer}, \ldots)$$

while Dealer2 separates `options` in a second Relation:

$$\texttt{Autos}(\texttt{serial}, \texttt{model}, \texttt{color})$$
$$\texttt{Options}(\texttt{serial}, \texttt{option}).$$

and the Mediator virtual view is:

$$\texttt{AutosMed}(\texttt{serialNo}, \texttt{model}, \texttt{color}, \texttt{autoTransmission}, \texttt{dealer})$$

We wish to answer the following user query:

```
SELECT serialNo,model
FROM   AutosMed
WHERE  color = 'red';
```

Data Extraction: An Example (Cont.)

The Wrapper for Dealer1 rewrites the query as:

SELECT serialNo,model
FROM Cars
WHERE color = 'red';

The Wrapper for Dealer2 rewrites the query as:

SELECT serial,model
FROM Autos
WHERE color = 'red';

The Mediator can take the union of these sets and return the result to the user.

Automatic Wrapper Generators

- Mediator-based Integrated systems require more complex wrappers than Warehouses.
- The wrapper must accept various queries and rewrite them into the terms of the source.
- It's quite important to build *Flexible* wrappers able to cope with different queries.
- A technique used to generate a wrapper is to classify the possible queries into **Templates**.

Summary

- Information Integration
  - Data Warehouse.
    - Data Extraction, Integration and Preprocessing
    - Warehouse Updates: View Maintenance
  - **Mediator/Query-Driven Integration.**
    - Data Extraction;
    - **Queries Templates and Filtering**
- Mediators Vs. Data Warehouses.

Query Templates

- ***Templates*** are query patterns with *parameters* that represent constants provided by the Mediator.
- Templates are described by the notation `T` $\Rightarrow$ `S`, meaning that the template T is rewritten to the source query S.
- **Example.** Consider the car dealers example and a wrapper for Dealer1 source. Suppose the wrapper should answer queries about colors. Then the template could be:

```
SELECT *
FROM   AutosMed
WHERE  color = '$c';
       ⇒
SELECT serialNo,model,color,autoTransmission,'dealer1'
FROM   Cars
WHERE  color = '$c';
```

Filtering Query Templates

Problem. Suppose that the user needs to look for cars of a particular *Color and Model*. Should the wrapper be extended with a new template or can we take advantage from the current template?

Solution. If the wrapper template is able to return a *Superset* of the current user query, then the wrapper can **Filter** the template query to obtain the desired result.

Example. Let the user query be:

SELECT *
FROM AutosMed
WHERE color = 'blue' **AND** model='FIAT Stilo';

Then, a possible way to answer the query would be to:

1. Use the color template with $\$c = 'blue'$ to find all the *blue* cars;
2. Filter from the above result the tuples such that `model=`*'FIAT Stilo'*.

Filtering Query Templates (Cont.)

- Checking whether a user query is a subset of a template query is a hard task and still a matter of Research (*Query Containment*).
- Filtering can involve more complex operations than further SELECTION. In particular, columns may be PROJECTED, different templates can be JOINED, attributes can be AGGREGATED, etc...

Summary

- Information Integration
  - Data Warehouse.
    - Data Extraction, Integration and Preprocessing
    - Warehouse Updates: View Maintenance
  - Mediator/Query-Driven Integration.
    - Data Extraction;
    - Queries Templates and Filtering
- **Mediators Vs. Data Warehouses.**

Query-Driven Approaches Vs. Warehouses

Query-Driven approaches are Inefficient and potentially Expensive for frequent queries.

- Delay in query processing
  - Slow or unavailable information sources;
  - Complex filtering and integration: non accurate queries;
  - Query Optimization at run-time;
  - Competes with local processing at sources.

The possible *Advantages* are:

- No *Space* needed to store a materialized view;
- No problems related to view maintenance;
- Data always up-to-date.

Advantages of Warehousing Approach

- High query performance
- Doesn't interfere with local processing at sources: Complex queries at warehouse while OLTP at information sources
- Information materialized at Warehouse
  - Can modify, annotate, summarize, restructure, etc...
  - Can store historical information
  - Security issues
- Has caught on in industry

Summary of Lecture VIII

- Information Integration
  - Data Warehouse.
    - Data Extraction, Integration and Preprocessing
    - Warehouse Updates: View Maintenance
  - Mediator/Query-Driven Integration.
    - Data Extraction;
    - Queries Templates and Filtering
- Mediators Vs. Data Warehouses.