**Information Integration**

- Large family of applications concerned with combining data from heterogeneous information sources
- Our goal: Get an overview of the
  - modes (or approaches, or architectures)
  - problems, and
  - some applications
  of information integration

---

### Modes of information integration

- **Federated databases**
  - collection of independent, co-operating sources
- **Warehousing**
  - single DB containing physical copies of data from several sources
- **Mediation**
  - system providing an integrated view to the sources (non-materialized, virtual database)

### Example

An integrated DB for dealer DBs of Aardvark Automobile Co, to support locating requested models and market analysis

**DB of Dealer1:**

```
Cars(serialNo, model, color, autoTrans, cdPlayer, ...)
-------------------------------
| 123 | BMV 1 | blue | yes | yes |
| 456 | MB 50 | red  | yes | no  |
```

**Options(serial, option)**

```
----------------------------------
| 323 | manualTrans |
| 323 | cdPlayer |
```

**DB of Dealer2:**

```
Autos(serial, model, color)
----------------------------
| 323 | MZ-6 | BL |
| 555 | MB 50 | Rd |
```

**Options(serial, option)**

```
----------------------------------
| 323 | manualTrans |
| 323 | cdPlayer |
```

### Problems of information integration

- Different names
  - Cars/Autos, serialNo/serial, ...
  - renaming straightforward
- Different data types
  - variable vs. fixed-length strings vs. integers vs ...
- Conceptual differences
  - Are trucks included in 'Cars'?
  - Are station wagons classified as minivans?
- Treatment of missing values
  - represent with NULLs or some default values?
Information integration architectures

1. Federated database
   - one-to-one connections between independent databases
   - Requires (n-1) interfaces to fully connect n systems together

Example (of federation)
Dealer1 wants to search requested cars also from the DB of Dealer2, for requests stored in relation

\[ \text{NeededCars (model, color, autoTrans)} \]

- Needs to query relations
  \[ \text{Autos(serial, model, color)} \]
  \[ \text{Options(serial, option)} \]
  of Dealer2 (with the following embedded SQL):

```sql
for (each tuple (:m, :c, :a) in NeededCars) {
  if (:a='yes') { // autoTrans wanted
    SELECT serial FROM Autos, Options
    WHERE Autos.serial = Options.serial AND
    model = :m AND color = :c AND
    option='autoTrans';
  } else { // manual transmission wanted
    SELECT serial FROM Autos
    WHERE model = :m AND color = :c AND
    NOT EXISTS (
      SELECT * FROM Options
      WHERE serial = Autos.serial AND
      option = 'autoTrans';
    )
  }
}
```

2. Data Warehouse
- Database formed by extracting and combining data from several sources
  - E.g., sales data for a supermarket chain
  - generally read-only

Example (Data warehouse of Aadvark Co.)
Create a warehouse of Dealer1 and Dealer2 databases with combined schema

\[ \text{CarWH(serNo, model, color, autoTrans, dealer)} \]

- Extract Dealer1’s data:
  ```sql
  INSERT INTO CarWH(serNo, model, color, autoTrans, dealer)
  SELECT serialNo, model, color, 'yes', 'dealer1'
  FROM Cars;
  ```

- Extract Dealer2’s data:
  ```sql
  INSERT INTO CarWH(serNo, model, color, autoTrans, dealer)
  SELECT serial, model, color, 'yes', 'dealer2'
  FROM Autos, Options
  WHERE Autos.serial = Options.serial AND
  option='autoTrans';

  INSERT INTO CarWH(serNo, model, color, autoTrans, dealer)
  SELECT serial, model, color, 'no', 'dealer2'
  FROM Autos
  WHERE NOT EXISTS (
    SELECT * FROM Options
    WHERE serial = Autos.serial AND
    option = 'autoTrans';
  );
  ```
3. Mediators

- Support views combining data from several sources

![Mediator Diagram]

**Example (mediation of dealer sources)**

Integrate Dealer1 and Dealer2 databases as a virtual database with schema

```
CarMed(serNo, model, color, autoTrans, dealer)
```

- Suppose user of the mediator gives query Q:
  ```sql
  SELECT serNo, model 
  FROM CarMed WHERE color='red'
  ```

- Translation by wrapper of Dealer1 easy:
  ```sql
  Q1: SELECT serialNo AS serNo, model 
  FROM Cars WHERE color='red'
  ```

- Query Q2 on relation `Autos` of Dealer2 similarly

- Return UNION of results to Q1 and Q2

**Extractors and wrappers**

- An *extractor* for a warehouse needs
  - one or more queries to get data from source
  - SQL, or other supported language
  - communication mechanisms for
    - passing queries to source
    - receiving responses from the source
    - passing data to the warehouse

- A *wrapper* for a mediator more complex
  - translates arbitrary queries for the source

**An aside: XML wrappers**

- What if the source does not provide a query language?
  - Need to translate a piece of extracted data
    (file/report/message/document/...)
  - developing and maintaining translation programs is tedious
  - => a wrapper specification language XW
    (XML Wrapper; Univ. of Kuopio, 2001-2002)

**XML in data exchange**

- XML an industry standard for textual representation of structured data (W3C Recommendation 1998)
  - XML-based protocols developed for e-business, medical messages, ...
  - old message formats need to be converted to XML
  - this translation addressed by XW

**XW wrapping**

```
<part-a>
  <e1>x1</e1>
  <e2>x2</e2>
</part-a>
<part-b>
  <line-1>
    <d1>y1</d1>
    <d2>y2</d2>
  </line-1>
  <d3>z2</d3>
</part-b>
```
Back to DB wrappers: Query templates

- Translating queries of mediator to queries on the source possible through **templates**
  - with parameters (place-holders for constants of the original query)
  - Notation $T \Rightarrow S$: template $T$ is translated to source query $S$
- Example: Translating color-based selections on mediated view to queries on Dealer1 DB

Example (of query templates)

```
SELECT *
FROM CarMed WHERE color = '$c' ;
=>
SELECT serialNo AS serNo, model, color, autoTrans, 'dealer1'
FROM Cars WHERE color = '$c' ;
```

- Similar templates for `model` and `autoTrans`
  - supporting all combinations of $n$ attributes requires $2^n$ templates!
  - Can do with fewer by applying filtering (see later)

Wrapper generators

- Templates defining a wrapper translated into code of a wrapper
  - by a **wrapper generator**
  - How?
- Two parts of a wrapper
  - table of query patterns specified by templates
  - a fixed **driver**

Example (of query templates)

```
SELECT * FROM CarMed WHERE color='blue' AND model='Gobi' ;
```

Filtering

- Often impossible to write templates for each possible form of mediator query
- Possible to wrap more complex queries
  - if some template returns a superset of the original query (by fixing only a part of conditions), then
  - filter the result of the template by the additional query conditions

Example (of filtering by wrapper)

```
SELECT * FROM CarMed WHERE color='blue' AND model='Gobi' ;
```

if we have only the below template?

```
SELECT *
FROM CarMed WHERE color = '$c' ;
=>
SELECT serialNo AS serNo, model, color, autoTrans, 'dealer1'
FROM Cars WHERE color = '$c' ;
```
Example (of filtering cont.)

1. Apply template with $c = 'blue'$
   and store the result:
   ```sql
   INSERT INTO Temp(serNo, model, color, autoTrans, dealer)
   SELECT serialNo AS serNo, model, color, autoTrans, 'dealer1'
   FROM Cars WHERE color = 'blue';
   ```

2. Filter out Gobis from relation Temp:
   ```sql
   SELECT * FROM Temp
   WHERE model = 'Gobi';
   ```

Warehouses and their applications

- **Warehouses**
  - aimed at executive, decision maker, analyst
  - often a copy of operational data

- **Advantages**
  - High query performance
  - Local processing at sources unaffected
  - Can operate when sources unavailable

Applications of warehouses (1)

- **OLAP (On-line Analytic Processing)**
  - finding important patterns or trends from data stored at the warehouse
  - involves complex queries on large parts of the database
  - Contrast: OLTP (On Line Transaction Processing)
    - Describes processing at operational sites, not at the warehouse

OLTP vs. OLAP

<table>
<thead>
<tr>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reads &amp; updates</td>
<td>Mostly reads</td>
</tr>
<tr>
<td>Many small transactions</td>
<td>Queries long, complex</td>
</tr>
<tr>
<td>MB-TB of data</td>
<td>GB-TB of data</td>
</tr>
<tr>
<td>Raw, up-to-date data</td>
<td>Summarized data,</td>
</tr>
<tr>
<td>Clerical users</td>
<td>updated periodically</td>
</tr>
<tr>
<td>Consistency,</td>
<td>Decision-makers,</td>
</tr>
<tr>
<td>recoverability critical</td>
<td>analysts as users</td>
</tr>
</tbody>
</table>

Applications of warehouses (2)

- **Data mining**
  - active area of R&D since 1990's
  - analysis of large amounts of data for finding interesting regularities or exceptions

- **Why?**
  - To get valuable information out of huge data collections of research commerce and industry

Example (data mining)

- **"Association rules" (IBM 1993)**
  - way to analyse co-occurrence of values
  - E.g. the market-basket analysis:
    
    $bread, cheese \Rightarrow butter (53\%, 74\%)$

<table>
<thead>
<tr>
<th>Support:</th>
<th>Confidence:</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many market_baskets contain bread, cheese and butter?</td>
<td>How many buyers of bread &amp; cheese also bought butter?</td>
</tr>
</tbody>
</table>
What’s new in Data Mining?

• Statistics:
  – methods to test hypotheses on data
• OLAP:
  – user-driven examination of data
• Data mining
  – automatic generation of hypotheses among hundreds or thousands of explaining variables

Summary

• We’ve had an overview of information integration
• Different approaches
  – federation, warehousing, mediation
• Applications of data warehouses
  – OLAP
  – data mining