Data integration – general setting

- A source schema $S$:
  - relational schema XML Schema (DTD), etc.
- A global schema $G$:
  - could be of many different types too
- A mapping $M$ between $S$ and $G$:
  - many ways to specify it, e.g. by queries that mention $S$ and $T$
- A general condition: the source and our view of the global schema should satisfy the conditions imposed by the mapping $M$. 
Data integration – general setting cont’d

- Assume we have a source database $D$.
- We are interested in databases $D'$ over the global schema such that $(D, D')$ satisfies the conditions of the mapping $M$.
- There are many possible ways to specify the mapping.
- The set of such databases $D'$ is denoted by $[D]_M$.
- If we have a query $Q$, we want certain answers that are true in all possible databases $D'$:

$$\text{certain}_M(Q, D) = \bigcap_{D' \in [D]_M} Q(D').$$
Data integration – general setting cont’d

- Depending on a type of mapping $M$, the set $[D]_M$ could be very large — or even infinite.
- That makes certain$_M(Q, D)$ prohibitively expensive or even impossible to compute.
- Hence we need a rewriting $Q'$ so that

$$\text{certain}_M(Q, D) = Q'(D)$$

or even

$$\text{certain}_M(Q, D) = Q'(V)$$

if $V$ is the set of views that the database $D$ makes available.
Types of mappings: Two major parameters

- Source-central vs global schema-central:
  - Source is defined in terms of the global schema
    - Known as local-as-view (LAV)
  - The global schema is defined in terms of the source
    - Known as global-as-view (GAV)
  - Combinations are possible (GLAV, P2P, to be seen later)

- Exact vs sound definitions
  - Exact definition specify precise relationships that must hold between the source and the global schema database
  - Sound definitions leave that description potentially incomplete: we know some relationships but not all of them.
    - potentially many more instances in $[D]_M$
Example

• Source schema:
  - EM50(title,year,director)
    – meaning: European movies made since 1950
  - RV10(movie,review)
    – reviews for the past 10 years

• Global schema:
  - Movie(title,director,year)
  - ED(name,country,dob) (European directors)
  - RV(movie,review) (reviews)
Example – LAV setting

• We define the source (local) in terms of the global schema – hence local is a view.

• Two possibilities for $D' \in [D]_M$:
  ◦ Exact: $D = Q(D')$, where $Q$ is a query over the global schema.
  ◦ Sound: $D \subseteq Q(D')$.
  ◦ In other words, if a fact is present in $D$, it must be derivable from the global schema by means of $Q$.

• More generally, for each $n$-ary relation $R$ in the source schema, there is a query $Q_R$ over the global schema such that
  - $R = Q_R(D')$ (exact)
  - $R \subseteq Q_R(D')$ (sound)
Sound LAV setting

\[
\text{EM50}(T,Y,D) \subseteq \left\{ (t, y, d) \mid \exists c, \text{dob} \left( \begin{array}{l}
\text{Movie}(t, y, d) \\
\land \text{ED}(d, c, \text{dob}) \\
\land y \geq 1950
\end{array} \right) \right\}
\]

\[
\text{RV10}(t, r) \subseteq \left\{ (t, r) \mid \exists y, d \left( \begin{array}{l}
\text{Movie}(t, y, d) \\
\land \text{RV}(t, r) \\
\land y \geq 2000
\end{array} \right) \right\}
\]

Right-hand sides are simple SQL queries involving joins and simple selection predicates:

```
SELECT M.title, RV.review
FROM Movie M, RV
WHERE M.title=RV.title AND M.year >= 2000
```
Exact LAV setting

\[
EM50(T,Y,D) = \left\{ (t, y, d) \mid \exists c, dob \left( \begin{array}{c}
\text{Movie}(t, y, d) \\
\land \text{ED}(d, c, dob) \\
\land y \geq 1950
\end{array} \right) \right\}
\]

\[
RV10(t, r) = \left\{ (t, r) \mid \exists y, d \left( \begin{array}{c}
\text{Movie}(t, y, d) \\
\land \text{RV}(t, r) \\
\land y \geq 2000
\end{array} \right) \right\}
\]

All the data from the global database must be reflected in the source.
LAV setting – queries

Consider a global schema query

```
SELECT M.title, R.review
FROM Movie M, RV R
WHERE M.title=R.title AND M.year = 2005
```

(Movies from 2005 and their reviews)

This is rewritten as a relational calculus query:

```
\{t, r \mid \exists d, y \text{ Movie}(t, d, y) \land \text{RV}(t, r) \land y = 2005\}
```
LAV setting:
\{t, r \mid \exists d, y \text{ Movie}(t, d, y) \land \text{ RV}(t, r) \land y = 2005\} 

Idea: re-express in terms of predicates of the source schema. The following seems to be the best possible way:

\{t, r \mid \exists d, y \text{ EM50}(t, y, d) \land \text{ RV10}(t, r) \land y = 2005\} 

and back to SQL:

SELECT EM50.title, RV10.review 
FROM EM50, RV10 
WHERE EM50.title=RV10.title AND EM50.year = 2005 

- Is this always possible? 
- In what sense is this the best way?
GAV settings

• Global schema is defined in terms of sources.
• Sound GAV:
  ○ $D' \supseteq Q(D)$
  ○ the global database contains the result of a query over the source
• Exact GAV:
  ○ $D' = Q(D)$
  ○ the global database is obtained as the result of a query over the source
• Note: in exact GAV, $[D]_M$ contains a unique database!
GAV example

- Change the schema slightly: $ED'(name)$ (i.e. we only keep names of European directors)
- A sound GAV setting:
  - $Movie \supseteq EM50$
  - $ED' \supseteq \{d \mid \exists t, y \ EM50(t, d, y)\}$
  - $RV \supseteq RV10$

Look at a SQL query:

```sql
SELECT M.title, RV.review
FROM Movie M, RV
WHERE M.title=RV.title AND M.year = 2005
```

(Movies from 2005 and their reviews)
GAV example

- Query: $\{t, r \mid \exists d, y \ M(t, d, y) \land RV(t, r) \land y = 2005\}$
- Substitute the definitions from the mapping and get:
  - $\{t, r \mid \exists d, y \ EM50(t, d, y) \land RV10(t, r) \land y = 2005\}$
- This is called unfolding.
- Does this always work? Can queries become too large?
Integration with views

- We have assumed that all source databases are available.
- But often we only get views that they publish.
- If only views are available, can queries be:
  - answered?
  - approximated?
- Assume that in EM50 directors are omitted. Then nothing is affected.
- But if titles are omitted in EM50, we cannot answer the query.
Towards view-based query answering

- Suppose only a view of the source is available. Can queries be answered?
- It depends on the query language.
- Start with relational algebra/calculus.
- Suppose we have either a LAV or a GAV setting, and we want to answer queries over the global schema using the view over the source.
- Problem: given the setting, and a query, can it be answered?
- This is **undecidable**!
- Two undecidable relational algebra problems:
  - If $e$ is a relational algebra expression, does it always produce $\emptyset$ (i.e., on every database)?
  - Closely related: if $e_1$ and $e_2$ are two relational algebra expressions, is it true that $e_1(D) = e_2(D)$ for every database?
Equivalence of relational algebra expressions

- A side note – this is the basis of query optimisation.
- But it can only be sound, never complete.
- Equivalence is undecidable for the full relational algebra
  - $\pi, \sigma, \bowtie, \cup, -$  
- The good news: it is decidable for $\pi, \sigma, \bowtie, \cup$
- And quite efficiently for $\pi, \sigma, \bowtie$
- And the latter form a very important class of queries, to be seen soon.
View-based query answering – relational algebra

• A very simple setting: exact LAV (and GAV)
  ○ the source schema and the target schema are identical (say, for each $R(A, B, C, \ldots)$ in the source there is $R'(A', B', C', \ldots)$ in the target)
  ○ The constraints in $M$ state that they are the same.
  ○ The source does not publish any views: i.e. $V = \emptyset$.

• If we can answer queries in this setting, it means they have to be answered independently of the data in the source.

• The only way it happens: $Q(D_1) = Q(D_2)$ for all databases $D_1, D_2$; we output this answer without even looking at the view $\emptyset$.

• But this $(Q(D_1) = Q(D_2)$ for all databases $D_1, D_2)$ is undecidable.
A better class of queries

- Conjunctive queries
- They are the building blocks for SQL queries:

  SELECT .....  
  FROM R1, ..., Rn  
  WHERE <conjunction of equalities>

- For example:

  SELECT M.title, RV.review  
  FROM Movie M, RV  
  WHERE M.title=RV.title AND M.year = 2005

- In relational calculus:

  \[ \{ t, r \mid \exists d, y \; \text{Movie}(t, d, y) \land \text{RV}(t, r) \land y = 2005 \} \]
Conjunctive queries

- \{ t, r \mid \exists d, y \text{ Movie}(t, d, y) \land \text{RV}(t, r) \land y = 2005 \}

- Written using only conjunction and existential quantification – hence the name.

- In relational algebra:

\[
\pi_{t,r} \left( \sigma_{y=2005} \left( \text{Movie} \Join_{\text{Movie}.t = \text{RV}.t} \text{RV} \right) \right)
\]

- Also called SPJ-queries (Select-Project-Join)

- These are all equivalent (exercise – why?)
 Conjunctive queries: good properties

- **QUERY CONTAINMENT**:  
  
  Input: two queries $Q_1$ and $Q_2$  
  Output: true if $Q_1(D) \subseteq Q_2(D)$ for all databases $D$

- **QUERY EQUIVALENCE**:  
  
  Input: two queries $Q_1$ and $Q_2$  
  Output: true if $Q_1(D) = Q_2(D)$ for all databases $D$

- For relational algebra queries, both are undecidable.
- For conjunctive queries, both are decidable.
- Complexity: NP. This gives an $2^{O(n)}$ algorithm.
- Can often be reasonable in practice – queries are small.
Conjunctive queries: good properties

- For each conjunctive query, one can find an equivalent query with the minimum number of joins.

- SELECT R2.A
  FROM R R1, R R2

- In relational algebra: $\pi_{\ldots}(\sigma_{\ldots}(R \times R))$

- $\{x \mid \exists y, z \ R(x, 2, 1) \land R(x, y, z)\}$

- Looking at it carefully, this is equivalent to $\{x \mid R(x, 2, 1)\}$, or $\pi_A(\sigma_{B=2 \land C=1}(R))$

- The join is saved:

  SELECT R.A
  FROM R WHERE R.B=2 AND R.C=1
Conjunctive queries: complexity

• Can one find a polynomial algorithm? Unlikely.
• Reminder: NP-completeness.
• Take a graph $G = (V, E)$:
  - $V = \{a_1, \ldots, a_n\}$ the set of vertices;
  - $E$ is the set of edges $(a_i, a_j)$
• and define a conjunctive query
  $$Q_G = \exists x_1, \ldots x_n \bigwedge_{(a_i, a_j) \in E} E(x_i, x_j)$$
• Then $G'$ satisfies $Q_G$ iff there is a homomorphism from $G$ to $G'$.
• A homomorphism from $G$ to $\{(r, b), (r, g), (g, b), (g, r), (b, r), (b, g)\}$
  $\iff$ the graph is 3-colourable.
Conjunctive queries: summary

- A nicely-behaved class
- Basic building blocks of SQL queries
- Easy to reason about
  - Another important property: monotonicity:
    - if $D_1 \subseteq D_2$ then $Q(D_1) \subseteq Q(D_2)$
- Heavily used in data integration/exchange
**GAV-exact with conjunctive queries**

- **Source:** $R_1(A, B), R_2(B, C)$
- **Global schema:** $T_1(A, B, C), T_2(B, C)$
- **Exact GAV mapping:**
  - $T_1 = \{ x, y, z \mid R_1(x, y) \land R_2(y, z) \}$ (or $R_1 \Join_B R_2$)
  - $T_2 = \{ x, y \mid R_2(x, y) \}$
- **Query $Q$:**

  ```sql
  SELECT T1.A, T1.B, T2.C
  FROM T1, T2
  ```

- **As conjunctive query:** $\{ x, y, z \mid T_1(x, y, z) \land T_2(y, z) \}$
GAV-exact with conjunctive queries cont’d

• Take \( \{ x, y, z \mid T_1(x, y, z) \land T_2(y, z) \} \) and unfold:

• \( \{ x, y, z \mid R_1(x, y) \land R_2(y, z) \land R_2(y, z) \} \)

• or \( R_1 \Join R_2 \Join R_2 \)

• This is of course \( R_1 \Join R_2 \).

• Bottom line: optimise after unfolding – save joins.
GAV-sound with conjunctive queries

- Source and global schema as before:
  - source $R_1(A, B), R_2(B, C)$
  - Global schema: $T_1(A, B, C), T_2(B, C)$

- GAV mappings become sound:
  - $T_1 \supseteq \{ x, y, z | R_1(x, y) \land R_2(y, z) \}$
  - $T_2 \supseteq R_2$

- Let $D_{exact}$ be the unique database that arises from the exact setting (with $\supseteq$ replaced by $=$)

- Then every database $D_{sound}$ that satisfies the sound setting also satisfies

$$D_{exact} \subseteq D_{sound}$$
GAV-sound with conjunctive queries cont’d

- Conjunctive queries are monotone:
  \[ D_1 \subseteq D_2 \implies Q(D_1) \subseteq Q(D_2) \]

- Exact solution is a sound solution too, and is contained in every sound solution.

- Hence certain answers for each conjunctive query
  \[
  \text{certain}(D, Q) = \bigcap_{D_{\text{sound}}} Q(D_{\text{sound}}) = Q(D_{\text{exact}})
  \]

- The solution for GAV-exact gives us certain answers for GAV-sound, for conjunctive (and more generally, monotone) queries.