Data Integration and Data Exchange
Traditional approach to databases

- A single large repository of data.
- Database administrator in charge of access to data.
- Users interact with the database through application programs.
- Programmers write those (embedded SQL, other ways of combining general purpose programming languages and DBMSs)
- Queries dominate; updates less common.
- DBMS takes care of lots of things for you such as
  - query processing and optimisation
  - concurrency control
  - enforcing database integrity
Traditional approach to databases cont’d

• This model works very within a single organisation that either
  o does not interact much with the outside world, or
  o the interaction is heavily controlled by the DB administrators

• What do we expect from such a system?
  1. Data is relatively clean; little incompleteness
  2. Data is consistent (enforced by the DMBS)
  3. Data is there (resides on the disk)
  4. Well-defined semantics of query answering (if you ask a query, you know what you want to get)
  5. Access to data is controlled
The world is changing

- The traditional model still dominates, but the world is changing.
- Many huge repositories are publicly available
  - In fact many are well-organised databases, e.g., imdb.com, the CIA World Factbook, many genome databases, the DBLP server of CS publications, etc etc etc)
- Many queries cannot be answered using a single source.
- Often data from various sources needs to be combined, e.g.
  - company mergers
  - restructuring databases within a single organisation
  - combining data from several private and public sources
What industry offers now: ETL tools

• ETL stands for **Extract–Transform–Load**
  ○ Extract data from multiple sources
  ○ Transform it so it is compatible with the schema
  ○ Load it into a database

• Many self-built tools in the 80s and the 90s; through acquisition fewer products exist now

• The big players – IBM, Microsoft, Oracle – all have their ETL products; Microsoft and Oracle offer them with their database products.

• A few independent vendors, e.g. Informatica PowerCenter.

• Several open source products exist, e.g. Clover ETL.
ETL tools

- Focus:
  - Data profiling
  - Data cleaning
  - Simple transformations
  - Bulk loading
  - Latency requirements

- What they don’t do yet:
  - nontrivial transformations
  - query answering

- But techniques now exist for interesting data integration and for query answering – and we shall learn them.

- They soon will be reflected in products (IBM and Microsoft are particularly active in this area)
Data profiling/cleaning

- **Data profiling:** gives the user a view of data:
  - Samples over large tables
  - Statistics (how many different values etc)
  - Graphical tools for exploring the database

- **Cleaning:**
  - Same properties may have different names
    - e.g. Last_Name, L_Name, LastName
  - Same data may have different representations
    - e.g. (0131)555-1111 vs 01315551111,
    - George Str. vs George Street
  - Some data may be just wrong
Data transformation

- Most transformation rules tend to be simple:
  - Copy attribute LName to Last_Name
  - Set age to be current_year – DOB

- Heavy emphasis on industry specific formats

- For example, Informatica B2B Data Exchange product offers versions for Healthcare and Financial services as well as specialised tools for formats including:
  - MS Word, Excel, PDF, UN/EDIFACT (Data Interchange For Administration, Commerce, and Transport), RosettaNet for B2B, and many specialised healthcare and financial form.

- These are format/industry specific and have little to do with the general tasks of data integration.
Data integration, scenario 1

GLOBAL SCHEMA

QUERY: Q?
Data integration

GLOBAL SCHEMA

QUERY: Q?
Data integration

Answer to $Q$ is obtained by querying the views $V_1, \ldots, V_n$
Data integration, query answering

• We have our view of the world (the Global Schema)
• We can access (parts of) databases $DB_1, \ldots, DB_n$ to get relevant data.
• It comes in the form of views, $V_1, \ldots, V_n$
• Our query against the global schema must be reformulated as a query against the views $V_1, \ldots, V_n$
• The approach is completely virtual: we never create a database that conforms to the global schema.
Data integration, query answering, a toy example

- List courses taught by permanent teaching staff during Winter 2007
- We have two databases:
  - $D_1$ (name, age, salary) of permanent staff
  - $D_2$ (teacher, course, semester, enrollment) of courses
- $D_1$ only publishes the value of the name attribute
- $D_2$ does not reveal enrollments
- The views:
  \[
  V_1 = \pi_{\text{name}}(D_1)
  \]
  \[
  V_2 = \pi_{\text{teacher,course,semester}}(D_2)
  \]
- Next step: establish correspondence between attributes name of $V_1$ and teacher of $V_2$
To answer query, we need to import the following data:

\[ V_1 \]

\[ W_2 = \sigma_{\text{semester} = 'Winter 2007'}(V_2) \]

Answering query:

\[ \{ \text{course} \mid \exists \text{name, sem} \ V_1(\text{name}) \land W_2(\text{name, course, sem}) \} \]

Or, in relational algebra

\[ \pi_{\text{course}}(V_1 \bowtie_{\text{name}=\text{teacher}} W_2) \]
Toy example, lessons learned

- We don’t have access to all the data
- Some human intervention is essential (someone needs to tell us that teacher and name refer to the same entity)
- We don’t run a query against a single database. Instead, we
  - run queries against different databases based on restrictions they impose
  - get results to use them locally
  - run another query against those results
Toy example, things getting more complicated

- Find informatics permanent staff who taught during the Winter 2007 semester, and their phone numbers
- We have additional personnel databases:
  - an informatics database $D_3(employee, phone, office)$, and
  - a university-wide database $D_4(employee, school, phone)$
  - for simplicity, assume all this information is public
- Now we have a choice:
  - use $D_3$ to get information about phones
  - use $D_4$ to get information about phones
  - use both $D_3$ and $D_4$ to get information about phones
Toy example cont’d

• First, we need some human involvement to see that employee, name, and teacher refer to the same category of objects

• If one uses $D_3$, then the query is
  \[
  \{ \text{name, phone} \mid \exists \text{sem, course, office} \ V_1(\text{name}) \land \\
  W_2(\text{name, course, sem}) \land D_3(\text{name, phone, office}) \} 
  \]

• If one uses $D_4$, then the query is
  \[
  \{ \text{name, phone} \mid \exists \text{sem, course, school} \ V_1(\text{name}) \land \\
  W_2(\text{name, course, sem}) \land D_4(\text{name, school, phone}) \} 
  \]

• But what if one uses both $D_3$ and $D_4$?
Toy example cont’d

• We could insist on the phone number being:
  ◦ in either $D_3$ or $D_4$
  ◦ in both $D_3$ and $D_4$, but not necessarily the same
  ◦ in both $D_3$ and $D_4$, and the same in both databases

• One can write queries for all the cases, but which one should we use?

• New lessons:
  ◦ databases that are being integrated are often inconsistent
  ◦ query answering is by no means unique – there could be several ways to answer a query
  ◦ different possibilities for answering queries are a result of inconsistencies and incomplete information
Toy example cont’d

• Suppose phone numbers in $D_3$ and $D_4$ are different.

• What is a sensible query answer then?

• A common approach is to use certain answers – these are guaranteed to be true.

• Another question: what if there is no record at all for the phone number in $D_3$ and $D_4$?

• Then we have an instance of incomplete information.
A different scenario

- So far we looked at virtual integration: no database of the global schema was created.
- Sometimes we need such a database to be created, for example, if many queries are expected to be asked against it.
- In general, this is a common problem with data integration: materialize vs federate.
- Materialize = create a new database based on integrating data from different sources.
- Federate = the virtual approach: obtain data from various sources and use them to answer queries.
Virtual vs Materialization

- A common situation for the materialization approach: merger of different organizations.
- A common situation for the federated approach: we don’t have full access to the data, and the data changes often.
Common tasks in data integration

• How do we represent information?
  ○ Global schema, attributes, constraints
  ○ data formats of attributes
  ○ reconciling data from different sources
  ○ abbreviations, terminology, ontologies

• How do we deal with imperfect information?
  ○ resolve overlaps
  ○ handling missing data
  ○ handling inconsistencies
Common tasks in data integration cont’d

• How do we answer queries?
  ◦ what information is available?
  ◦ Can we get the answer?
  ◦ if not, what is the semantics of query answering?
  ◦ Is query answering feasible?
  ◦ Is it possible to compute query answers at all?
  ◦ If now, how do we approximate?

• Materialize or federate?
Common tasks in data integration cont’d

- Do it from scratch or use commercial tools?
  - many are available (just google for “data integration”)
  - but do we fully understand them?
  - lots of them are very ad hoc, with poorly defined semantics
  - this is why it is so important to understand what really happens in data integration
Data Exchange

Source Schema $S$  

Target Schema $T$
Data Exchange

Source Schema $S$

Source Database

Target Schema $T$

Target Database
Data Exchange

Query over the target schema: $Q$

How to answer $Q$ so that the answer is consistent with the data in the source database?
Data exchange vs Data integration

Data exchange appears to be an easier problem:

- there is only one source database;
- and one has complete access to the source data.

But there could be many different target instances.

Problem: which one to use for query answering?
When do we have the need for data exchange

• A typical scenario:
  ○ Two organizations have their legacy databases, schemas cannot be changed.
  ○ Data from one organization 1 needs to be transferred to data from organization 2.
  ○ Queries need to be answered against the transferred data.
Query answering using views

- General setting: database relations $R_1, \ldots, R_n$.
- Several views $V_1, \ldots, V_k$ are defined as results of queries over the $R_i$’s.
- We have a query $Q$ over $R_1, \ldots, R_n$.
- Question: Can $Q$ be answered in terms of the views?
  - In other words, can $Q$ be reformulated so it only refers to the data in $V_1, \ldots, V_k$?
Query answering using views in data integration

- **LAV:**
  - $R_1, \ldots, R_n$ are global schema relations; $Q$ is the global schema query
  - $V_i$’s are the sources defined over the global schema
  - We must answer $Q$ based on the sources (virtual integration)

- **GAV:**
  - $R_1, \ldots, R_n$ are the sources that are not fully available.
  - $Q$ is a query in terms of the source (or a query that was reformulated in terms of the sources)
  - Must see if it is answerable from the available views $V_1, \ldots, V_k$.

- We know the problem is impossible to solve for full relational algebra, hence we concentrate on conjunctive queries.
Query answering using views: example

- Two relations in the database: \text{Cites(A,B)} (if A cites B), and \text{SameTopic(A,B)} (if A, B work on the same topic)

- Query \( Q(x, y) :\neg \text{SameTopic}(x, y), \text{Cites}(x, y), \text{Cites}(y, x) \)

- Two views are given:
  - \( V_1(x, y) :\neg \text{Cites}(x, y), \text{Cites}(y, x) \)
  - \( V_2(x, y) :\neg \text{SameTopic}(x, y), \text{Cites}(x, x'), \text{Cites}(y, y') \)

- Suggested rewriting: \( Q'(x, y) :\neg V_1(x, y), V_2(x, y) \)

- Why? Unfold using the definitions:
  \[ Q'(x, y) :\neg \text{Cites}(x, y), \text{Cites}(y, x), \text{SameTopic}(x, y), \text{Cites}(x, x'), \text{Cites}(y, y') \]

- Equivalent to \( Q \)
Query answering using views

• Need a formal technique (algorithm): cannot rely on the semantics.

• Query \( Q \):

\[
\text{SELECT } R1.A \\
\text{FROM } R R1, R R2, S S1, S S2 \\
\text{AND } R1.B=1 \text{ and } S2.B=1
\]

• \( Q(x) \) :– \( R(x, y), R(x, 1), S(x, z), S(x, 1) \)

• Equivalent to \( Q(x) :– R(x, 1), S(x, 1) \)

• So if we have a view

  \( V(x, y) :– R(x, y), S(x, y) \) (i.e. \( V = R \cap S \)), then
  \( Q = \pi_A(\sigma_{B=1}(V)) \)
  \( Q \) can be rewritten (as a conjunctive query) in terms of \( V \)
Query rewriting

- Setting:
  - Queries $V_1, \ldots, V_k$ over the same schema $\sigma$ (assume to be conjunctive; they define the views)
  - Each $Q_i$ is of arity $n_i$
  - A schema $\omega$ with relations of arities $n_1, \ldots, n_k$

- Given:
  - a query $Q$ over $\sigma$
  - a query $Q'$ over $\omega$

- $Q'$ is a rewriting of $Q$ if for every $\sigma$-database $D$,
  \[ Q(D) = Q'( V_1(D), \ldots, V_k(D) ) \]
Maximal rewriting

• Sometimes exact rewritings cannot be obtained

• $Q'$ is a maximally-contained rewriting if:
  
  ○ it is contained in $Q$:
    
    \[ Q'(V_1(D), \ldots, V_k(D)) \subseteq Q(D) \]
    
    for all $D$
  
  ○ it is maximal such: if
    
    \[ Q''(V_1(D), \ldots, V_k(D)) \subseteq Q(D) \]
    
    for all $D$, then
    
    \[ Q'' \subseteq Q' \]
Query rewriting: a naive algorithm

• Given:
  ○ conjunctive queries $V_1, \ldots, V_k$ over schema $\sigma$
  ○ a query $Q$ over $\sigma$

• Algorithm:
  ○ guess a query $Q'$ over the views
  ○ Unfold $Q'$ in terms of the views
  ○ Check if the unfolding is contained in $Q$

• If one unfolding is equivalent to $Q$, then $Q'$ is a rewriting

• Otherwise take the union of all unfoldings contained in $Q$
  – it is a maximally contained rewriting
Why is it not an algorithm yet?

- Problem: the guess stage.
  - There are infinitely many conjunctive queries.
  - We cannot check them all.
  - Solution: we only need to check a few.
Guessing rewritings

- A basic fact:
  - If there is a rewriting of $Q$ using $V_1, \ldots, V_k$, then there is a rewriting with no more conjuncts than in $Q$.
  - E.g., if $Q(x) := R(x, y), R(x, 1), S(x, z), S(x, 1)$, we only need to check conjunctive queries over $V$ with at most 4 conjuncts.

- Moreover, maximally contained rewriting is obtained as the union of all conjunctive rewritings of length of $Q$ or less.

- Complexity: enumerate all candidates (exponentially many); for each an NP (or exponential) algorithm. Hence exponential time is required.

- Cannot lower this due to NP-completeness.
Query rewriting

- Recall the algorithm, for a given $Q$ and view definitions $V_1, \ldots, V_k$:
  - Look at all rewritings that have as at most as many joins as $Q$
  - check if they are contained in $Q$
  - take the union of those that are
- This is the maximally contained rewriting
- There are algorithms that prune the search space and make looking for rewritings contained in $Q$ more efficient
  - the bucket algorithm
  - MiniCon
How hard is it to answer queries using views?

- Setting: we now have an actual content of the views.
- As before, a query is $Q$ posed against $D$, but must be answered using information in the views.
- Suppose $I_1, \ldots, I_k$ are view instances. Two possibilities:
  - Exact mappings: $I_j = V_j(D)$
  - Sound mappings: $I_j \subseteq V_j(D)$
- We need certain answers for given $\mathcal{I} = (I_1, \ldots, I_k)$:
  \[
  \text{certain}_{\text{exact}}(Q, \mathcal{I}) = \bigcap_{D: I_j=V_j(D) \text{ for all } j} Q(D)
  \]
  \[
  \text{certain}_{\text{sound}}(Q, \mathcal{I}) = \bigcap_{D: I_j\subseteq V_j(D) \text{ for all } j} Q(D)
  \]
How hard is it to answer queries using views?

• If \( \text{certain}_{\text{exact}}(Q, \mathcal{I}) \) or \( \text{certain}_{\text{sound}}(Q, \mathcal{I}) \) are impossible to obtain, we want maximally contained rewritings:
  - \( Q'(\mathcal{I}) \subseteq \text{certain}_{\text{exact}}(Q, \mathcal{I}) \), and
  - if \( Q''(\mathcal{I}) \subseteq \text{certain}_{\text{exact}}(Q, \mathcal{I}) \) then \( Q''(\mathcal{I}) \subseteq Q'(\mathcal{I}) \)
  - (and likewise for \text{sound})

• How hard is it to compute this from \( \mathcal{I} \)?
Complexity of query answering

• We want the complexity of finding

\[ \text{certain}_{\text{exact}}(Q, \mathcal{I}) \quad \text{or} \quad \text{certain}_{\text{sound}}(Q, \mathcal{I}) \]

in terms of the size of \( \mathcal{I} \)

• If all view definitions are conjunctive queries and \( Q \) is a relational algebra or a SQL query, then the complexity is \( \text{coNP} \).

• This is too high!

• If all view definitions are conjunctive queries and \( Q \) is a conjunctive query, then the complexity is \( \text{PTIME} \).
  
  ◦ Because: the maximally contained rewriting computes certain answers!
Complexity of query answering

<table>
<thead>
<tr>
<th>view language</th>
<th>CQ</th>
<th>CQ≠</th>
<th>relational calculus</th>
</tr>
</thead>
<tbody>
<tr>
<td>CQ</td>
<td>ptime</td>
<td>coNP</td>
<td>undecidable</td>
</tr>
<tr>
<td>CQ≠</td>
<td>ptime</td>
<td>coNP</td>
<td>undecidable</td>
</tr>
<tr>
<td>relational calculus</td>
<td>undecidable</td>
<td>undecidable</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

CQ – conjunctive queries

CQ≠ – conjunctive queries with inequalities
(for example, \( Q(x) :– R(x, y), S(y, z), x \neq z \) )
Data exchange

- Source schema, target schema; need to transfer data between them.
- A typical scenario:
  - Two organizations have their legacy databases, schemas cannot be changed.
  - Data from one organization 1 needs to be transferred to data from organization 2.
  - Queries need to be answered against the transferred data.
Data Exchange

Source Schema $S$  Target Schema $T$
Data Exchange

Source Schema $S$  \hspace{1cm}  Target Schema $T$
Data exchange: an example

• We want to create a target database with the schema

  \( \text{Flight}(\text{city}_1, \text{city}_2, \text{aircraft}, \text{departure}, \text{arrival}) \)
  \( \text{Served}(\text{city}, \text{country}, \text{population}, \text{agency}) \)

• We don't start from scratch: there is a source database containing relations

  \( \text{Route}(\text{source}, \text{destination}, \text{departure}) \)
  \( \text{BG}(\text{country}, \text{city}) \)

• We want to transfer data from the source to the target.
Data exchange – relationships between the source and the target

How to specify the relationship?
Relationships between the source and the target

- Formal specification: we have a relational calculus query over both the source and the target schema.
- The query is of a restricted form, and can be thought of as a sequence of rules:

  \( \text{Flight}(c1, c2, __, \text{dept}, __) \leftarrow \text{Route}(c1, c2, \text{dept}) \)

  \( \text{Served}(\text{city}, \text{country}, __, __) \leftarrow \text{Route}(\text{city}, __, __), \text{BG}(\text{country}, \text{city}) \)

  \( \text{Served}(\text{city}, \text{country}, __, __) \leftarrow \text{Route}(__, \text{city}, __), \text{BG}(\text{country}, \text{city}) \)
Data exchange – targets

- Target instances should satisfy the rules.
- What does it mean to satisfy a rule?
- Formally, if we take:

\[
\text{Flight}(c_1, c_2, \_, \text{dept}, \_) \Rightarrow \text{Route}(c_1, c_2, \text{dept})
\]

then it is satisfied by a source \( S \) and a target \( T \) if the constraint

\[
\forall c_1, c_2, d \left( \text{Route}(c_1, c_2, d) \rightarrow \exists a_1, a_2 \left( \text{Flight}(c_1, c_2, a_1, d, a_2) \right) \right)
\]

- This constraint is a relational calculus query that evaluates to true or false.
Data exchange – targets

• What happens if there no values for some attributes, e.g. *aircraft*, *arrival*?

• We put in *null values* or some real values.

• But then we may have multiple solutions!
Data exchange – targets

Source Database:

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Departure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edinburgh</td>
<td>Amsterdam</td>
<td>0600</td>
</tr>
<tr>
<td>Edinburgh</td>
<td>London</td>
<td>0615</td>
</tr>
<tr>
<td>Edinburgh</td>
<td>Frankfurt</td>
<td>0700</td>
</tr>
</tbody>
</table>

Look at the rule

\[
\text{Flight}(c1, c2, \_, \text{dept}, \_) :\neg \text{Route}(c1, c2, \text{dept})
\]

The right hand side is satisfied by

\[
\text{Route}(\text{Edinburgh, Amsterdam, 0600})
\]

But what can we put in the target?
Data exchange – targets

Rule:  \( \text{Flight}(c1, c2, \_\_, \text{dept}, \_\_) \leftarrow \text{Route}(c1, c2, \text{dept}) \)

Satisfied by:  \( \text{Route}(\text{Edinburgh}, \text{Amsterdam}, 0600) \)

Possible targets:

- \( \text{Flight}(\text{Edinburgh}, \text{Amsterdam}, \bot_1, 0600, \bot_2) \)
- \( \text{Flight}(\text{Edinburgh}, \text{Amsterdam}, B737, 0600, \bot) \)
- \( \text{Flight}(\text{Edinburgh}, \text{Amsterdam}, \bot, 0600, 0845) \)
- \( \text{Flight}(\text{Edinburgh}, \text{Amsterdam}, \bot, 0600, \bot) \)
- \( \text{Flight}(\text{Edinburgh}, \text{Amsterdam}, B737, 0600, 0845) \)

They all satisfy the constraints!
Which target to choose

• One of them happens to be right:
  – Flight(Edinburgh, Amsterdam, B737, 0600, 0845)
• But in general we do not know this; it looks just as good as
  – Flight(Edinburgh, Amsterdam, ’The Spirit of St Louis’, 0600, 1300), or
  – Flight(Edinburgh, Amsterdam, F16, 0600, 0620).
• Goal: look for the “most general” solution.
• How to define “most general”: can be mapped into any other solution.
• It is not unique either, but the space of solution is greatly reduced.
• In our case Flight(Edinburgh, Amsterdam, ⊥₁, 0600, ⊥₂) is most general as it makes no additional assumptions about the nulls.
Towards good solutions

A solution is a database with nulls. Reminder: every such database $T$ represents many possible complete databases, without null values:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>⊥</td>
</tr>
<tr>
<td>⊥</td>
<td>⊥</td>
<td>3</td>
</tr>
<tr>
<td>⊥</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>⊥</td>
<td>3</td>
</tr>
</tbody>
</table>

Example

$\begin{align*}
v(\perp_1) &= 4 \\
v(\perp_2) &= 3 \\
v(\perp_3) &= 5
\end{align*}$

$\Rightarrow$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>⊥</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

$[T]_{owa} = \{ R \mid v(T) \subseteq R \text{ for some valuation } v \}$
Good solutions

- An **optimistic** view – A good solution represents ALL other solutions:
  \[
  [T]_{\text{owa}} = \{ R \mid R \text{ is a solution without nulls} \}
  \]

- Shouldn’t settle for less than – A good solution is at least as general as others:
  \[
  [T]_{\text{owa}} \supseteq [T']_{\text{owa}} \text{ for every other solution } T'
  \]

- Good news: these two views are equivalent. Hence can take them as a definition of a good solutions.

- In data exchange, such solutions are called **universal solutions**.
Universal solutions: another description

• A homomorphism is a mapping $h : \text{Nulls} \rightarrow \text{Nulls} \cup \text{Constants}$.  

• For example, $h(\bot_1) = B737$, $h(\bot_2) = 0845$.  

• If we have two solutions $T_1$ and $T_2$, then $h$ is a homomorphism from $T_1$ into $T_2$ if for each tuple $t$ in $T_1$, the tuple $h(t)$ is in $T_2$.  

• For example, if we have a tuple  

$$t = \text{Flight}(\text{Edinburgh, Amsterdam, } \bot_1, 0600, \bot_2)$$  

then  

$$h(t) = \text{Flight}(\text{Edinburgh, Amsterdam, B737, 0600, 0845}).$$

• A solution is universal if and only if there is a homomorphism from it into every other solution.
Universal solutions: still too many of them

- Take any $n > 0$ and consider the solution with $n$ tuples:

  Flight(Edinburgh, Amsterdam, $\bot_1$, 0600, $\bot_2$)
  Flight(Edinburgh, Amsterdam, $\bot_3$, 0600, $\bot_4$)
  \ldots
  Flight(Edinburgh, Amsterdam, $\bot_{2n-1}$, 0600, $\bot_{2n}$)

- It is universal too: take a homomorphism

  $$h'(\bot_i) = \begin{cases} 
  \bot_1 & \text{if } i \text{ is odd} \\
  \bot_2 & \text{if } i \text{ is even}
  \end{cases}$$

- It sends this solution into

  Flight(Edinburgh, Amsterdam, $\bot_1$, 0600, $\bot_2$)
Universal solutions: cannot be distinguished by conjunctive queries

- There are queries that distinguish large and small universal solutions (e.g., does a relation have at least 2 tuples?)
- But these cannot be distinguished by conjunctive queries
- Because: if $\bot_{i_1}, \ldots, \bot_{i_k}$ witness a conjunctive query, so do $h(\bot_{i_1}), \ldots, h(\bot_{i_k})$ — hence, one tuple suffices
- In general, if we have
  - a homomorphism $h : T \to T'$,
  - a conjunctive query $Q$,
  - a tuple $t$ without nulls such that $t \in Q(T)$
- then $t \in Q(T')$
Universal solutions and conjunctive queries

• If
  ○ $T$ and $T'$ are two universal solutions
  ○ $Q$ is a conjunctive query, and
  ○ $t$ is a tuple without nulls,
then
  $$t \in Q(T) \iff t \in Q(T')$$
because we have homomorphisms $T \rightarrow T'$ and $T' \rightarrow T$.

• Furthermore, if
  ○ $T$ is a universal solution, and $T''$ is an arbitrary solution,
then
  $$t \in Q(T) \Rightarrow t \in Q(T'')$$
Now recall what we learned about answering conjunctive queries over databases with nulls:

- $T$ is a naive table
- the set of tuples without nulls in $Q(T)$ is precisely certain$(Q, T)$ – certain answers over $T$

Hence if $T$ is an arbitrary universal solution

$$\text{certain}(Q, T) = \bigcap \{Q(T') \mid T' \text{ is a solution}\}$$

$\bigcap \{Q(T') \mid T' \text{ is a solution}\}$ is the set of certain answers in data exchange under mapping $M$: certain$_M(Q, S)$. Thus

$$\text{certain}_M(Q, S) = \text{certain}(Q, T)$$

for every universal solution $T$ for $S$ under $M$. 
Universal solutions cont’d

• To answer conjunctive queries, one needs an arbitrary universal solution.

• We saw some; intuitively, it is better to have:

\[
\text{Flight(Edinburgh, Amsterdam, } \perp_1, \ 0600, \ \perp_2)\]

than

\[
\text{Flight(Edinburgh, Amsterdam, } \perp_1, \ 0600, \ \perp_2) \\
\text{Flight(Edinburgh, Amsterdam, } \perp_3, \ 0600, \ \perp_4) \\
\ldots \\
\text{Flight(Edinburgh, Amsterdam, } \perp_{2n-1}, \ 0600, \ \perp_{2n})
\]

• We now define a canonical universal solution.
Canonical universal solution

- Convert each rule into a rule of the form:

\[ \psi(x_1, \ldots, x_n, z_1, \ldots, z_k) \quad :\quad \varphi(x_1, \ldots, x_n, y_1, \ldots, y_m) \]

(for example,

\[ \text{Flight}(c1, c2, _, \text{dept}, _) \quad :\quad \text{Route}(c1, c2, \text{dept}) \]

becomes

\[ \text{Flight}(x_1, x_2, z_1, x_3, z_2) \quad :\quad \text{Route}(x_1, x_2, x_3) \]

- Evaluate \( \varphi(x_1, \ldots, x_n, y_1, \ldots, y_m) \) in \( S \).

- For each tuple \((a_1, \ldots, a_n, b_1, \ldots, b_m)\) that belongs to the result (i.e.

\[ \varphi(a_1, \ldots, a_n, b_1, \ldots, b_m) \] holds in \( S \),

do the following:
Canonical universal solution cont’d

• ... do the following:
  ○ Create new (not previously used) null values \( \bot_1, \ldots, \bot_k \)
  ○ Put tuples in target relations so that

\[
\psi(a_1, \ldots, a_n, \bot_1, \ldots, \bot_k)
\]

holds.

• What is \( \psi \)?

• It is normally assumed that \( \psi \) is a conjunction of atomic formulae, i.e.

\[
R_1(\bar{x}_1, \bar{z}_1) \land \ldots \land R_l(\bar{x}_l, \bar{z}_l)
\]

• Tuples are put in the target to satisfy these formulae
Canonical universal solution cont’d

- Example: no-direct-route airline:

  \[ \text{Newroute}(x_1, z) \land \text{Newroute}(z, x_2) \leftarrow \text{Oldroute}(x_1, x_2) \]

- If \((a_1, a_2) \in \text{Oldroute}(a_1, a_2)\), then create a new null \(\bot\) and put:

  \[ \begin{align*}
  \text{Newroute}(a_1, \bot) \\
  \text{Newroute}(\bot, a_2)
  \end{align*} \]

  into the target.

- Complexity of finding this solution: polynomial in the size of the source \(S\):

  \[ O(\sum_{\text{rules } \psi \leftarrow \varphi} \text{Evaluation of } \varphi \text{ on } S) \]
Canonical universal solution and conjunctive queries

• Canonical solution: $\text{CanSol}_M(S)$.

• We know that if $Q$ is a conjunctive query, then $\text{certain}_M(Q, S) = \text{certain}(Q, T)$ for every universal solution $T$ for $S$ under $M$.

• Hence

\[
\text{certain}_M(Q, S) = \text{certain}(Q, \text{CanSol}_M(S))
\]

• Algorithm for answering $Q$:
  ○ Construct $\text{CanSol}_M(S)$
  ○ Apply naive evaluation to $Q$ over $\text{CanSol}_M(S)$
Beyond conjunctive queries

- Everything still works the same way for $\sigma, \pi, \Join, \cup$ queries of relational algebra. Adding union is harmless.
- Adding difference (i.e. going to the full relational algebra) is not.
- Reason: same as before, can encode validity problem in logic.
- Single rule, saying “copy the source into the target”
  $$T(x, y) :– S(x, y)$$

- If the source is empty, what can a target be? Anything!
- The meaning of $T(x, y) :– S(x, y)$ is
  $$\forall x \forall y \ (S(x, y) \rightarrow T(x, y))$$
Beyond conjunctive queries cont’d

• Look at $\varphi = \forall x \forall y (S(x, y) \rightarrow T(x, y))$

• $S(x, y)$ is always false ($S$ is empty), hence $S(x, y) \rightarrow T(x, y)$ is true ($p \rightarrow q$ is $\neg p \vee q$)

• Hence $\varphi$ is true.

• Even if $T$ is empty, $\varphi$ is true: universal quantification over the empty set evaluates to true:
  
  ○ Remember SQL’s ALL:

  ```sql
  SELECT * FROM R
  WHERE R.A > ALL (SELECT S.B FROM S)
  ```

  ○ The condition is true if SELECT S.B FROM S is empty.
Beyond conjunctive queries cont’d

• Thus if $S$ is empty and we have a rule $T(x, y) : S(x, y)$, then all $T$’s are solutions.

• Let $Q$ be a Boolean (yes/no) query. Then

$$\text{certain}_M(Q, S) = \text{true} \iff Q \text{ is valid}$$

• Valid = always true.

• Validity problem in logic: given a logical statement, is it:
  ○ valid, or
  ○ valid over finite databases

• Both are undecidable.
Beyond conjunctive queries cont’d

• If we want to answer queries by rewritings, i.e. find a query $Q'$ so that

\[
\text{certain}_M(Q, S) = Q'(\text{CAN}\text{SOL}_M(S))
\]

then there is no algorithm that can construct $Q'$ from $Q$!

• Hence a different approach is needed.
Key problem

- Our main problem:
  Solutions are open to adding new facts
- How to close them?
- By applying the CWA (Closed World Assumption) instead of the OWA (Open World Assumption)
More flexible query answering: dealing with incomplete information

- Key issue in dealing with incomplete information:
  - Closed vs Open World Assumption (CWA vs OWA)

- CWA: database is closed to adding new facts except those consistent with one of the incomplete tuples in it.

- OWA opens databases to such facts.

- In data exchange:
  - we move data from source to target;
  - query answering should be based on that data and not on tuples that might be added later.

- Hence in data exchange CWA seems more reasonable.
Solutions under CWA – informally

• Each null introduced in the target must be justified:
  - there must be a constraint \( \ldots T(\ldots, z, \ldots) \ldots \leftarrow \varphi(\ldots) \) with \( \varphi \) satisfied in the source.

• The same justification shouldn't generate multiple nulls:
  - for \( T(\ldots, z, \ldots) \leftarrow \varphi(\bar{a}) \) only one new null \( \perp \) is generated in the target.

• No unjustified facts about targets should be invented:
  - assume we have \( T(x, z) \leftarrow \varphi(x) \), \( T(z', x) \leftarrow \psi(x) \) and \( \varphi(a), \psi(b) \) are true in the source.
  - Then we put \( T(a, \perp) \) and \( T(\perp', b) \) in the target but not \( T(a, \perp), T(\perp, b) \) which would invent a new “fact”: \( a \) and \( b \) are connected by a path of length \( 2 \).
Solutions under the CWA: summary

- There are homomorphisms
  \[ h : \text{CanSol}(S) \rightarrow T \quad h' : T \rightarrow \text{CanSol}(S) \]

- so that \( T = h(\text{CanSol}(S)) \)
- \( T \) contains the core of \( \text{CanSol}(S) \)
- Core: the smallest \( C \subseteq \text{CanSol}(S) \) such that there is a homomorphism from \( \text{CanSol}(S) \) to \( C \).
- Often saves space, but takes time to compute.
- Data exchange systems try to move from \( \text{CanSol}(S) \) to the core but usually stop half-way due to the complexity of computation.
Query answering under the CWA

• Given
  ○ a source $S$,
  ○ a set of rules $M$,
  ○ a target query $Q$,

  A tuple $t$ is in $\text{certain}^M_{\text{CWA}}(Q, S)$ if it is in $Q(R)$ for every
  ○ solution $T$ under the CWA, and
  ○ $R \in [T]_{\text{owa}}$

• (i.e. no matter which solution we choose and how we interpret the nulls)
Query answering under the CWA – characterization

• Given a source $S$, a set of rules $M$, and a target query $Q$:
  $\text{certain}^\text{CWA}_M(Q, S) = \text{certain}(Q, \text{CanSol}(S))$

• That is, to compute the answer to query one needs to:
  ◦ Compute the canonical solution $\text{CanSol}(S)$ – which has nulls in it
  ◦ Find certain answers to $Q$ over $\text{CanSol}(S)$

• If $Q$ is a conjunctive query, this is exactly what we had before

• Under the CWA, the same evaluation strategy applies to all queries!
Data exchange and integrity constraints

- Integrity constraints are often specified over target schemas.
- In SQL’s data definition language one uses keys and foreign keys most often, but other constraints can be specified too.
- Adding integrity constraints in data exchange is often problematic, as some natural solutions – e.g., the canonical solution – may fail them.
Target constraints cause problems

- The simplest example:
  - Copy source to target
  - Impose a constraint on target not satisfied in the source
- Data exchange setting:
  - \( T(x, y) \leftarrow S(x, y) \) and
  - Constraint: the first attribute is a key
- Instance \( S: \begin{array}{cc}
  1 & 2 \\
  1 & 3 
\end{array} \)
- Every target \( T \) must include these tuples and hence violates the key.
Target constraints: more problems

- A common problem: an attempt to repair violations of constraints leads to an sequence of adding tuples.

- Example:
  
  - Source `DeptEmpl(dept_id,manager_name,empl_id)`
  - Target
    - `Dept(dept_id,manager_id,manager_name),`  
      - `Empl(empl_id,dept_id)`
  - Rule `Dept(d, z, n), Empl(e, d) :- DeptEmpl(d, n, e)`
  - Target constraints:
    - `Dept[manager_id] ⊆ Empl[empl_id]`
    - `Empl[dept_id] ⊆ Dept[dept_id]`
Target constraints: more problems cont’d

- Start with (CS, John, 001) in DeptEmpl.
- Put Dept(CS, ⊥₁, John) and Empl(001, CS) in the target
- Use the first constraint and add a tuple Empl(⊥₁, ⊥₂) in the target
- Use the second constraint and put Dept(⊥₂, ⊥₃, ⊥₃’) into the target
- Use the first constraint and add a tuple Empl(⊥₃, ⊥₄) in the target
- Use the second constraint and put Dept(⊥₄, ⊥₅, ⊥₅’) into the target
- this never stops....
Target constraints: avoiding this problem

• Change the target constraints slightly:
  ○ Target constraints:
    - Dept[dept_id, manager_id] ⊆ Empl[empl_id, dept_id]
    - Empl[dept_id] ⊆ Dept[dept_id]

• Again start with (CS, John, 001) in DeptEmpl.
• Put Dept(CS, ⊥₁, John) and Empl(001, CS) in the target
• Use the first constraint and add a tuple Empl(⊥₁, CS)
• Now constraints are satisfied – we have a target instance!

• What’s the difference? In our first example constraints are very cyclic causing an infinite loop. There is less cyclicity in the second example.

• Bottom line: avoid cyclic constraints.
Schema mappings

- Rules used in data exchange specify mappings between schemas.
- To understand the evolution of data one needs to study operations on schema mappings.
- Most commonly we need to deal with two operations:
  - composition
  - inverse
Composition and inverse

S1 \( \Sigma \) S2 \( \Delta \) S3
Composition and inverse

S1 \xrightarrow{\Sigma} S2 \xrightarrow{\Delta} S3

\Sigma \circ \Delta
Composition and inverse

\[ \Sigma \circ \Delta \]

\[ \Gamma \]

S1 → S2 → S3

S1'
Composition and inverse

\[ \Gamma^{-1} \circ (\Sigma \circ \Delta) \]

\[ \Sigma \circ \Delta \]
Mappings

- Schema mappings are typically given by rules

\[ \psi(\bar{x}, \bar{z}) \; :\; \exists \bar{u} \; \varphi(\bar{x}, \bar{y}, \bar{u}) \]

where

- \( \psi \) is a conjunction of atoms over the target:

\[ T_1(\bar{x}_1, \bar{z}_1) \land \ldots \land T_m(\bar{x}_m, \bar{z}_m) \]

- \( \varphi \) is a conjunction of atoms over the source:

\[ S_1(\bar{x}_1', \bar{y}_1, \bar{u}_1) \land \ldots \land S_k(\bar{x}_k', \bar{y}_k, \bar{u}_k) \]

- Example: \( \text{Served}(x_1, x_2, z_1, z_2) \; :\; \exists u_1, u_2 \; \text{Route}(x_1, u_1, u_2) \land \text{BG}(x_1, x_2) \)
The closure problem

- Are mappings closed under
  - composition?
  - inverse?
- If not, what needs to be added?
- It turns out that mappings are not closed under inverses and composition.
- We next see what might need to be added to them.
Skolem functions

- Source: \( EP(\text{empl\_name}, \text{dept}, \text{project}) \);
  Target: \( EDPH(\text{empl\_id}, \text{dept}, \text{phone}), \text{DP(dept,project)} \)

- A natural mapping is:
  \[
  EDPH(z_1, x_2, z_3) \land DP(x_2, x_3) \quad \leftarrow \quad EP(x_1, x_2, x_3)
  \]

- This is problematic: if we have tuples
  \[(\text{John, CS, } P_1) \quad (\text{John, CS, } P_2)\]
  in \( EP \), the canonical solution would have
  \[
  \begin{array}{ccc}
  \bot_1 & \text{CS} & \bot'_1 \\
  \bot_2 & \text{CS} & \bot'_2 \\
  \end{array}
  \]

  corresponding to two projects \( P_1 \) and \( P_2 \).

- So \( \text{empl\_id} \) is hardly an id!
Skolem functions cont’d

• Solution: make empl_id a function of empl_name.

• Such “invented” functions are called Skolem functions (see Logic 001 for a proper definition)

• Source: \( \text{EP}(\text{empl\_name,dept,project}) \);
  Target: \( \text{EDPH}(\text{empl\_id,dept,phone}), \text{DP}(\text{dept,project}) \)

• A new mapping is:

\[
\text{EDPH}(f(x_1), x_2, z_3) \land \text{DP}(x_2, x_3) \leftarrow \text{EP}(x_1, x_2, x_3)
\]

• \( f \) assigns a unique id to every name.
Other possible additions

- One can look at more general queries used in mappings.
- Most generally, relational algebra queries, but to be more modest, one can start with just adding inequalities.
- One may also **disjunctions**: for example, if we want to invert

\[
T(x) \leftarrow S_1(x) \\
T(x) \leftarrow S_2(x)
\]

it seems natural to introduce a rule

\[
S_1(x) \lor S_2(x) \leftarrow T(x)
\]
Composition: definition

- Recall the definition of composition of binary relations $R$ and $R'$:

$$(x, z) \in R \circ R' \iff \exists y : (x, y) \in R \text{ and } (y, z) \in R'$$

- A schema mapping $\Sigma$ for two schemas $\sigma$ and $\tau$ is viewed as a binary relation

$$\Sigma = \{(S, T) \mid S \text{ is a } \sigma\text{-instance} \quad T \text{ is a } \tau\text{-instance} \quad T \text{ is a solution for } S\}$$

- The composition of mappings $\Sigma$ from $\sigma$ to $\tau$ and $\Delta$ from $\tau$ to $\omega$ is now

$$\Sigma \circ \Delta$$

- Question (closure): is there a mapping $\Gamma$ between $\sigma$ and $\omega$ so that

$$\Gamma = \Sigma \circ \Delta$$
Composition: when it works

- If \( \Sigma \)
  - does not generate any nulls, and
  - no variables \( \bar{u} \) for source formulas

- Example:
  
  \[
  \Sigma : \quad T(x_1, x_2) \land T(x_2, x_3) \quad \text{:=} \quad S(x_1, x_2, x_3) \\
  \Delta : \quad W(x_1, x_2, z) \quad \text{:=} \quad T(x_1, x_2)
  \]

- First modify into:
  
  \[
  \Sigma : \quad T(x_1, x_2) \quad \text{:=} \quad S(x_1, x_2, x_3) \\
  \Sigma : \quad T(x_2, x_3) \quad \text{:=} \quad S(x_1, x_2, x_3) \\
  \Delta : \quad W(x_1, x_2, z) \quad \text{:=} \quad T(x_1, x_2)
  \]

- Then substitute in the definition of \( W \):
Composition: when it cont’d

\[
\begin{align*}
W(x_1, x_2, z) & := S(x_1, x_2, y) \\
W(x_1, x_2, z) & := S(y, x_1, x_2)
\end{align*}
\]

to get \( \Sigma \circ \Delta \).

Explaining the second rule:

\[
\begin{align*}
W(x_1, x_2, z) \\
\rightarrow T(x_1, x_2) \quad \text{using } T(var_1, var_2) & := S(var_3, var_1, var_2) \\
\rightarrow S(y, x_1, x_2)
\end{align*}
\]
Composition: when it doesn’t work

- Schema $\sigma$: Takes(st_name, course)
- Schema $\tau$: Takes’(st_name, course), Nameld(st_name, st_id)
- Schema $\omega$: Enroll(st_id, course)
- Mapping $\Sigma$ from $\sigma$ to $\tau$:
  
  \[
  \begin{align*}
  \text{Takes'}(s, c) & : \leftarrow \text{Takes}(s, c) \\
  \text{Nameld}(s, i) & : \leftarrow \exists c \text{Takes}(s, c)
  \end{align*}
  \]

- Mapping $\Delta$ from $\tau$ to $\omega$:
  
  \[
  \begin{align*}
  \text{Enroll}(i, c) & : \leftarrow \text{Nameld}(s, i) \land \text{Takes'}(s, c)
  \end{align*}
  \]

- A first attempt at the composition: $\text{Enroll}(i, c) : \leftarrow \text{Takes}(s, c)$
Composition: when it doesn’t work cont’d

• What’s wrong with $\Gamma$: $\text{Enroll}(i, c) :- \text{Takes}(s, c)$?
• Student id $i$ depends on both name and course!

\[
\begin{align*}
\begin{array}{c|c}
\text{Takes:} & \text{John} & \text{CS1} \\
& \text{John} & \text{CS2} \\
\end{array}
& \xrightarrow{\Sigma} \\
\begin{array}{c|c}
\text{Takes':} & \text{John} & \text{CS1} \\
& \text{John} & \text{CS2} \\
\end{array}
& \xrightarrow{\Delta} \\
\begin{array}{c|c}
\text{Nameld:} & \text{John} & \bot \\
\end{array}
& \xrightarrow{} \\
\begin{array}{c|c}
\text{Enroll:} & \bot & \text{CS1} \\
& \bot & \text{CS2} \\
\end{array}
\end{align*}
\]

But:

\[
\begin{align*}
\begin{array}{c|c}
\text{Takes:} & \text{John} & \text{CS1} \\
& \text{John} & \text{CS2} \\
\end{array}
& \xrightarrow{\Gamma} \\
\begin{array}{c|c}
\text{Enroll:} & \bot_1 & \text{CS1} \\
& \bot_2 & \text{CS2} \\
\end{array}
\end{align*}
\]
Composition: when it doesn’t work cont’d

- Solution: Skolem functions.
- $\Gamma'$: $\text{Enroll}(f(s), c) \leftarrow \text{Takes}(s, c)$
- Then:

\[
\begin{array}{c|c}
\text{Takes} & \text{Enroll} \\
\hline
\text{John} & \text{CS1} \\
\text{John} & \text{CS2} \\
\hline
& \exists \\
\end{array}
\]

- where $\bot = f(\text{John})$
Composition: another example

- Schema $\sigma$: $\text{Empl}(\text{eid})$
- Schema $\tau$: $\text{Mngr}(\text{eid},\text{mngid})$
- Schema $\omega$: $\text{Mngr}'(\text{eid},\text{mngid}), \text{SelfMng}(\text{id})$
- Mapping $\Sigma$ from $\sigma$ to $\tau$:
  $$\text{Mngr}(e, m) :\leftarrow \text{Empl}(e)$$

- Mapping $\Delta$ from $\tau$ to $\omega$:
  $$\text{Mngr}'(e, m) :\leftarrow \text{Mngr}(e, m)$$
  $$\text{SelfMng}(e) :\leftarrow \text{Mngr}(e, e)$$

- Composition:
  $$\text{Mngr}'(e, f(e)) :\leftarrow \text{Empl}(e)$$
  $$\text{SelfMng}(e) :\leftarrow \text{Empl}(e) \land e = f(e)$$
Composition and Skolem functions

• Schema mappings with Skolem functions compose!

• Algorithm:
  ○ replace all nulls by Skolem functions
    - \( \text{Mngr}(e, f(e)) \vdash \text{Empl}(e) \)
    - \( \Delta \) stays as before
  ○ Use substitution:
    - \( \text{Mngr}'(e, m) \vdash \text{Mngr}(e, m) \) becomes
      \[ \text{Mngr}'(e, f(e)) \vdash \text{Empl}(e) \]
    - \( \text{SelfMng}(e) \vdash \text{Mngr}(e, e) \) becomes
      \[ \text{SelfMng}(e) \vdash \text{Empl}(e) \land e = f(e) \]
Inverting mappings

- Harder than composition.
- Intuition: $\Sigma \circ \Sigma^{-1} = \text{ID}$.
- But even what $\text{ID}$ should be is not entirely clear.
- Some intuitive examples will follow.
Examples of inversion

- The inverse of projection is null invention:
  - $T(x) \leftarrow S(x, y)$
  - $S(x, y) \leftarrow T(x)$

- Inverse of union requires disjunction:
  - $T(x) \leftarrow S(x)$  $T(x) \leftarrow S'(x)$
  - $S(x) \lor S'(x) \leftarrow T(x)$

- So reversing the rules doesn’t always work.
Examples of inversion cont’d

- Inverse of decomposition is join:
  - $T(x_1, x_2) \land T'(x_2, x_3) \iff S(x_1, x_2, x_3)$
  - $S(x_1, x_2, x_3) \iff T(x_1, x_2) \land T'(x_2, x_3)$

- But this is also an inverse of $T(x_1, x_2) \land T'(x_2, x_3) \iff S(x_1, x_2, x_3)$:
  - $S(x_1, x_2, z) \iff T(x_1, x_2)$
  - $S(z, x_2, x_3) \iff T'(x_2, x_3)$
Examples of inversion cont’d

• One may need to distinguish nulls from values in inverses.

• $\Sigma$ given by

\[
\begin{align*}
T_1(x) & : \ S(x, x) \\
T_2(x, z) & : \ S(x, y) \land S(y, x) \\
T_3(x_1, x_2, z) & : \ S(x_1, x_2)
\end{align*}
\]

• Its inverse $\Sigma^{-1}$ requires:
  
  o a predicate $\text{NotNull}$ and
  
  o inequalities:

\[
\begin{align*}
S(x, x) & : \ T_1(x) \land T_2(x, y_1) \land T_3(x, x, y_2) \land \text{NotNull}(x) \\
S(x_1, x_2) & : \ T_3(x_1, x_2, y) \land (x_1 \neq x_2) \land \text{NotNull}(x_1) \land \text{NotNull}(x_2)
\end{align*}
\]
Integrating preferences/rankings

Problem statement

- Each object has \( m \) grades, one for each of \( m \) criteria.
- The grade of an object for field \( i \) is \( x_i \).
- Normally assume \( 0 \leq x_i \leq 1 \).
  - Typically evaluations based on different criteria
  - The higher the value of \( x_i \), the better the object is according to the \( i \)th criterion
- The objects are given in \( m \) sorted lists
  - the \( i \)th list is sorted by \( x_i \) value
  - These lists correspond to different sources or to different criteria.
- Goal: find the top \( k \) objects.
Example

<table>
<thead>
<tr>
<th>Grade 1</th>
<th>Grade 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(17, 0.9936)</td>
<td>(235, 0.9996)</td>
</tr>
<tr>
<td>(1352, 0.9916)</td>
<td>(12, 0.9966)</td>
</tr>
<tr>
<td>(702, 0.9826)</td>
<td>(8201, 0.9926)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(12, 0.3256)</td>
<td>(17, 0.406)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Aggregation Functions

• Have an aggregation function $F$.

• Let $x_1, \ldots, x_m$ be the grades of object $R$ under the $m$ criteria.

• Then $F(x_1, \ldots, x_m)$ is the overall grade of object $R$.

• Common choices for $F$:
  
  o min
  o average or sum

• An aggregation function $F$ is monotone if
  
  $$F(x_1, \ldots, x_m) \leq F(x'_1, \ldots, x'_m)$$

  whenever $x_i \leq x'_i$ for all $i$. 

Other Applications

- Information retrieval
- Objects $R$ are documents.
- The $m$ criteria are search terms $s_1, \ldots, s_m$.
- The grade $x_i$: how relevant document $R$ is for search term $s_i$.
- Common to take the aggregation function $F$ to be the sum

$$F(x_1, \ldots, x_m) = x_1 + \cdots + x_m.$$
Modes of Access

- **Sorted** access
  - Can obtain the next object with its grade in list $L_i$
  - cost $c_S$.

- **Random** access
  - Can obtain the grade of object $R$ in list $L_i$
  - cost $c_R$.

- **Middleware cost:**
  
  $$c_S \cdot (\# \text{ of sorted accesses}) + c_R \cdot (\# \text{ of random accesses}).$$
Algorithms

- Want an algorithm for finding the top $k$ objects.
- Naive algorithm:
  - compute the overall grade of every object;
  - return the top $k$ answers.
- Too expensive.
Fagin’s Algorithm (FA)

1. Do sorted access in parallel to each of the $m$ sorted lists $L_i$.
   - Stop when there are at least $k$ objects, each of which have been seen in all the lists.

2. For each object $R$ that has been seen:
   - Retrieve all of its fields $x_1, \ldots, x_m$ by random access.
   - Compute $F(R) = F(x_1, \ldots, x_m)$.

3. Return the top $k$ answers.
Fagin’s algorithm is correct

- Assume object $R$ was not seen
  - its grades are $x_1, \ldots, x_m$.
- Assume object $S$ is one of the answers returned by FA
  - its grades are $y_1, \ldots, y_m$.
- Then $x_i \leq y_i$ for each $i$
- Hence

$$F(R) = F(x_1, \ldots, x_m) \leq F(y_1, \ldots, y_m) = F(S).$$
Fagin’s algorithm: performance guarantees

• Typically probabilistic guarantees
• Orderings are independent
• Then with high probability the middleware cost is

\[ O\left( N \cdot \sqrt[k]{k/N} \right) \]

• i.e., sublinear
• But may perform poorly
  ◦ e.g., if \( F \) is constant:
  ◦ still takes \( O\left( N \cdot \sqrt[k]{k/N} \right) \) instead of a constant time algorithm
Optimal algorithm: The Threshold Algorithm

1. Do sorted access in parallel to each of the $m$ sorted lists $L_i$. As each object $R$ is seen under sorted access:
   - Retrieve all of its fields $x_1, \ldots, x_m$ by random access.
   - Compute $F(R) = F(x_1, \ldots, x_m)$.
   - If this is one of the top $k$ answers so far, remember it.
   - Note: buffer of bounded size.

2. For each list $L_i$, let $\hat{x}_i$ be the grade of the last object seen under sorted access.

3. Define the threshold value $t$ to be $F(\hat{x}_1, \ldots, \hat{x}_m)$.

4. When $k$ objects have been seen whose grade is at least $t$, then stop.

5. Return the top $k$ answers.
Threshold Algorithm: correctness and optimality

- The Threshold Algorithm is correct for every monotone aggregate function $F$.
- Optimal in a very strong sense:
  - it is as good as any other algorithm on every instance
  - any other algorithm means: except pathological algorithms
  - as good means: within a constant factor
  - pathological means: making wild guesses.
Wild guesses can help

- An algorithm “makes a wild guess” if it performs random access on an object not previously encountered by sorted access.
- Neither FA nor TA make wild guesses, nor does any “natural” algorithm.
- Example: The aggregation function is $\text{min}$; $k = 1$.

$$\begin{align*}
\text{LIST } L_1 & \\
(1, 1) & \\
(2, 1) & \\
(3, 1) & \\
\ldots & \\
(n+1, 1) & \\
(n+2, 0) & \\
(n+3, 0) & \\
\ldots & \\
(2n+1, 0) & \\
\text{LIST } L_2 & \\
(2n+1, 1) & \\
(2n, 1) & \\
(2n-1, 1) & \\
\ldots & \\
(n+1, 1) & \\
(n, 0) & \\
(n-1, 0) & \\
\ldots & \\
(1, 0) &
\end{align*}$$
Threshold Algorithm as an approximation algorithm

- Approximately finding top $k$ answers.
- For $\varepsilon > 0$, an $\varepsilon$-approximation of top $k$ answers is a collection of $k$ objects $R_1, \ldots, R_k$ so that
  - for each $R$ not among them,
    $$(1 + \varepsilon) \cdot F(R_i) \geq F(R)$$

- Turning TA into an approximation algorithm:
  - Simply change the stopping rule into:
    - When $k$ objects have been seen whose grade is at least
      $$\frac{t}{1 + \varepsilon}$$
    then stop.