2. Query Evaluation
The evaluation strategy of SQL appears to be top-down.

And yet, SQL expressions are not evaluated top-down but rather bottom-up because the data is located on the leaves of the corresponding operator trees.

Some of the algebraic optimization steps, however, work top-down such as pushing selections or projections.

To this end, SQL queries (or views) are first translated into relational algebra expressions which are transformed into more efficient ones.

algebraic optimization
Example for Algebraic Optimization (1)

SQL View:

```
SELECT DISTINCT s.Name
FROM Students s,
     Professors p,
     Lectures l
WHERE l.Title = 'Intelligent IS'
  AND s.Supervisor = p.ID
  AND p.ID = l.Given;
```

Canonical translation into a relational algebra expression:

```
\pi_{s.Name}(\sigma_{l.Title = 'Intelligent IS' \land s.Supervisor = p.ID \land p.ID = l.Given} ((Students \times Professors) \times Lectures))
```
Example for Algebraic Optimization (2)

Operator tree of the algebra expression:

\[ \pi \sigma \times p \times s \]

1. \( \pi \) s.Name
2. \( \sigma \) I.Title = 'Intelligent IS' \( \land \) s.Supervisor = p.ID \( \land \) p.ID = I.Given
3. \( \times \) 320.000
4. \( \times \) 16.000
5. \( \times \) 1.000
6. \( \times \) 16
7. \( \times \) 20
Example for Algebraic Optimization (3)

Splitting conditions and pushing selections:
Example for Algebraic Optimization (4)

Selections applied to Cartesian product replaced by joins:

\[ s \text{.Name} \]

\[ p.\text{ID} = l.\text{Given} \]

\[ s.\text{Supervisor} = p.\text{ID} \]

\[ l.\text{Title} = 'Intelligent IS' \]
Example for Algebraic Optimization (5)

Change the order of joins:

π

s.Name

10

s.Supervisor = p.ID

p.ID = l.Given

σ

l.Title = 'Intelligent IS'

s

1.000

p

16

l

20
Example for Algebraic Optimization (6)

Still many unneeded columns:

- s.Name
- s.Supervisor = p.ID
- p.ID = l.Given
- l.Title = 'Intelligent IS'
Introducing projections helps but how many are feasible?

Example for Algebraic Optimization (7)
Example for Algebraic Optimization (8)

Refined expression by using implicit projections within semi joins:

Probably best evaluation plan!

1.000 s

16 p

20 l

1. Title = 'Intelligent IS'

s.Name

s.Supervisor = p.ID

p.ID = l.Given

l.Title = 'Intelligent IS'
Some Rules for Pushing Selections - 1

- Whenever the selection condition is a conjunction, selections can be cut off and their order can be swapped:

\[ \sigma_{C_1 \land C_2} (R) = \sigma_{C_1} (\sigma_{C_2} (R)) = \sigma_{C_2} (\sigma_{C_1} (R)) \]

- Push selections inside union, difference and intersection:

\[
\begin{align*}
\sigma_{C} (R_1 \cup R_2) & = \sigma_{C} (R_1) \cup \sigma_{C} (R_2) \\
\sigma_{C} (R_1 - R_2) & = \sigma_{C} (R_1) - \sigma_{C} (R_2) \\
\sigma_{C} (R_1 \cap R_2) & = \sigma_{C} (R_1) \cap \sigma_{C} (R_2)
\end{align*}
\]
Some Rules for Pushing Selections - 2

- Push selection inside a join, i.e. to a join argument,
  \[ \sigma_C ( R_1 \bowtie_{c2} R_2 ) = \sigma_C ( R_1 \bowtie_{c2} R_2 ) \]
  if C only uses attributes of R1

- Push selection inside an argument of a Cartesian product
  \[ \sigma_C ( R_1 \times R_2 ) = \sigma_C ( R_1 ) \times R_2 \]
  if C only uses attributes of R1

- If this is impossible for both R_1 and R_2, i.e., C uses attributes of R_1 and of R_2, then substitute selection applied to Cartesian product with join
  \[ \sigma_C ( R_1 \times R_2 ) = R_1 \bowtie_{c} R_2 \]
The strategy “pushing selections” fails as soon as dynamically generated selection conditions are considered:

\[ p(X,Y) \leftarrow e(4,1), \ p(1,Y), \ not \ d(X,Y) \]
The bindings for Y are dynamically generated. So, there is no initial selection condition for Y=2 to be pushed:

\[ p(X,Y) \leftarrow e(4,1), p(1,2), \text{not } d(4,2). \]

Next subquery \(?-d(4,2)\) asking for corresp. facts in relation d.
Query Evaluation by Fixpoint Iteration

\[
\{ (Y) : p(2,Y) \} ?
\]

- \( n(X) \leftarrow q(X), t(X) \)
- \( q(Y) \leftarrow r(Y) \)
- \( q(Y) \leftarrow t(Y) \)
- \( m(X) \leftarrow r(X) \)
- \( m(X) \leftarrow s(X,Y), m(Y) \)
- \( p(X,Y) \leftarrow s(X,Y), q(Y) \)

\[
\begin{align*}
&n(X) \\
t(5). \quad t(6). \quad t(7). \\
&t \rightarrow \\
r(1). \quad r(3). \quad r(4). \\
r \rightarrow \\
m \rightarrow \\
s(2,1). \quad s(3,4). \quad s(2,2). \quad s(5,2). \quad s(2,3). \\
s \rightarrow
\end{align*}
\]
Motivating Example (2)

n(X) ← q(X), t(X).

q(Y) ← r(Y).
q(Y) ← t(Y).

r(1).
r(3).
r(4).

m(X) ← r(X).
m(X) ← s(X,Y), m(Y).

s(2,1). s(2,2). s(5,2). s(2,3).

p(X,Y) ← s(X,Y), q(Y).

{(Y) : p(2,Y)}.

answer(Y) ← p(2,Y).

answer(Y) ← p(2,Y).

answer(Y) ← p(2,Y).
Motivating Example (3)

\[
\begin{align*}
n(X) & \leftarrow q(X), t(X). \\
q(Y) & \leftarrow r(Y), t(Y). \\
q(Y) & \leftarrow t(Y). \\
n(5), n(6), n(7). & \\
q(5), q(6), q(7), q(1), q(3), q(4). & \\
r(1), r(3), r(4). & \\
p(X,Y) & \leftarrow s(X,Y), q(Y). \\
p(2,1), p(2,3), p(7,6). & \\
n(5), n(6), n(7). & \\
p(X,Y) & \leftarrow s(X,Y), q(Y). \\
p(2,1), p(2,3), p(7,6). & \\
m(1), m(2), m(3), m(4), m(5). & \\
m(5). & \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
m(X) & \leftarrow s(X,Y), m(Y). \\
m(X) & \leftarrow r(X), s(X,Y), m(Y). \\
n(X) & \leftarrow q(X), t(X). \\
Motivating Example (4)

```
m(X) ← s(Y,X), r(X).
q(Y) ← r(Y).
q(Y) ← t(Y).
n(X) ← q(X), t(X).
p(X,Y) ← s(X,Y), q(Y).
```

```
s(2,1).  s(7,6).
s(2,2).  s(5,2).
s(2,3).
```

```
t(5).  t(6).  t(7).
r(1).  r(3).  r(4).
```

```
answer(Y) ← p(2,Y).
answer(Y) ← p(2,Y).
p(2,1).  p(2,3).  p(7,6).
```

```
Irrelevant for this query
```

```
Irrelevant, too
```
Motivating Example (5)

Only very few facts are really necessary for generating the answer set!
Naive Evaluation of Deductive Queries

• A direct implementation of our query semantics as a query evaluation method suffers from (at least) two **problems**, leading to intolerable **efficiency flaws** if applied to databases of realistic size:
  • Many **irrelevant relations** are materialized!
  • Many facts from relevant relations are unnecessary for constructing the required set of answer facts (**irrelevant facts**).

• Both problems are caused by the "**erratic bottom up-computation** of facts during fix-point iteration, which does not "know" which relations and facts will be **relevant** for the query under evaluation:

There is an urgent need for making answer set computation **goal-directed**!
Motivating Example: Subquery Approach (1)

Key idea towards goal-directedness:
Exploit the information about the current query for "directing" answer generation!

\[
\begin{align*}
q(Y) & \leftarrow r(Y), t(Y) \\
n(X) & \leftarrow q(X), t(X) \\
p(X, Y) & \leftarrow s(X, Y), q(Y) \\
m(X) & \leftarrow r(X), m(Y) \\
m(X) & \leftarrow s(X, Y), m(Y) \\
\end{align*}
\]

\[
\{(Y) : p(2, Y)\}
\]

\[
answer(Y) \leftarrow p(2, Y).
\]

\[
p(2,1), p(2,3).
\]

\[
s(2,1), s(2,3).
\]
Motivating Example: Subquery Approach (2)

Another "key idea":

Apply the technique of introducing query-specific internal rules instead of user rules as well, not just instead of queries!

\[
\begin{align*}
\text{n}(X) & \leftarrow q(X), t(X). \\
\text{r}(X) & \leftarrow r(X) \\
\text{s}(X,Y) & \leftarrow s(X,Y), m(Y). \\
\end{align*}
\]

\[
\begin{align*}
\text{answer}(Y) & \leftarrow \text{p_answer}(2,Y). \\
\text{p}(X,Y) & \leftarrow s(X,Y), q(Y). \\
\text{p_answer}(2,Y) & \leftarrow s(2,Y), q(Y). \\
\end{align*}
\]
Subqueries with Dynamic Restrictions

Next observation:
We just need those q-facts, that „come from“ the relevant part of the s-relation!
Subqueries with Dynamic Restrictions

Express this idea in terms of internal rules, too!

\[ \text{q} \leftarrow \text{r}(\text{Y}) \]
\[ \text{q} \leftarrow \text{t}(\text{Y}) \]

\[ \text{answer}(\text{Y}) \leftarrow \text{p}\_\text{answer}(\text{Y}) \]

\[ \text{p}(\text{X}, \text{Y}) \leftarrow \text{s}(\text{X}, \text{Y}), \text{q}(\text{Y}) \]

\[ \text{p}\_\text{answer}(\text{Y}) \leftarrow \text{s}(2, \text{Y}), \text{q}\_\text{answer}(\text{Y}) \]

\[ \text{q}\_\text{answer}(\text{Y}) \leftarrow \text{s}(2, \text{Y}), \text{r}(\text{Y}) \]
\[ \text{q}\_\text{answer}(\text{Y}) \leftarrow \text{s}(2, \text{Y}), \text{t}(\text{Y}) \]

Express this idea in terms of internal rules, too!
Subquery Approach: Intermediate Summary

The specific constant value present in the query has now been pushed as closely to the source of answer generation as possible!

No irrelevant facts anymore! (at least in the relevant part of the graph)
Summary of the Idea

- The example shows that it may be beneficial to take constants possibly occurring in the query into consideration. These constant restrictions establish on the required intermediate results needed for computing the answer set.
- Pushing such variable bindings top-down into the rules needed for answering the query is a step forward.
- In addition, it can be helpful to dynamically compute further such restrictions which can be identified during query evaluation only.
- By the way: This is what relational (SQL-)DBMS do anyway as part of their logical optimization phase, where „pushing selection conditions“ is a well-known technique.

Pushing restrictions originating from the respective query seems to be a way of making answer set computation goal-directed!
Subquery Approach: Rule Transformation

- The **key idea** of this approach is to transform not only the query to be answered, but those rules that are needed to answer the query into an internal format which can be evaluated more efficiently.
- The transformation is inherently a top-down process (from the query down to the base relations). It involves pushing constants in place of variables, and pushing literals into lower level rules.

```
{ (Y) : p(2,Y) } ?
p(X,Y) ← s(X,Y), q(Y).
q(Y) ← r(Y).
q(Y) ← t(Y).
```

```
answer(Y) ← p_answer(Y).
p_answer(Y) ← q_answer(Y).
q_answer(Y) ← s(2,Y), r(Y).
q_answer(Y) ← s(2,Y), t(Y).
```

- Doing so, enables us to retain fixpoint computation as the means of actually computing answers (i.e., relevant derivable facts).
- Fixpoint computation as such is a purely bottom-up process (from the base relations up to the query), which is now controlled by the prior top-down transformation.
• The „subquery approach“ is, however, not yet „perfect“. First, there are a number of smaller „defects“ – a really big problem will be discussed on the next slides!

• Pushing query constants into internal rules „at run time“, i.e., each time a new query arises, is not ideal, as this means to create new internal rules every time!
• It would be much better, if such a transformation could be done ahead of querying, i.e., immediately after the proper rules have been designed, once and for all:

  { (Y) : p(z,Y) } ?

  answer(Y) ← p_answer(Y).

  q(Y) ← r(Y).
  q(Y) ← t(Y).

  q_answer(Y) ← s(z,Y), r(Y).
  q_answer(Y) ← s(z,Y), t(Y).

• Another problem is that all the other rules not contributing to answering the current query are still „around“ and will produce irrelevant relations – unless „switched off“ in a suitable manner.

  But remember: The next query might use different rules again!
How does the transformation approach work in case of recursion?
Applying the same rule transformation causes „disaster“! Not a single m-fact is computed, not even m(5)!

The reason for this seems to be, that \texttt{m\_answer(2)} is no longer computed, an intermediate result which is essential for computing \texttt{m\_answer(5)}!

Thus, pushing the restriction \(X=5\) is far too strong!
Crucial point for solving this problem:

We have to „open up“ the recursive subquery for other values rather than just the single constant appearing in the top query!

How do we know which further constants are required for answering the relevant subqueries?
Another key observation:

The non-recursive literal in the recursive rule (containing the constant “pushed” down from the query) causes relevant Y-values to be generated!
Subquery Approach and Recursion (5)

 Brilliant idea:

The process of dynamic generation of further relevant constants can be completely expressed via internal rules, too!
The additional m_query-relation serves as a control mechanism for the generation of m_answers.
How does all of this work during fixpoint computation?

1st round of iteration:

Just one further (sub-)query fact is computed – no answers yet!

```
m_answer(X) ← m_query(X), r(X).
m_answer(X) ← m_query(X), s(X,Y), m_answer(Y).
```

```
m_query(5).
m_query(2).
```

```
m_query(Y) ← m_query(X), s(X,Y).
```
How does all of this work during fixpoint computation?

2nd round of iteration:
Two more (sub-)query fact arise – still no answers yet!
How does all of this work during fixpoint computation?

3rd round of iteration:
No more (sub-)query facts
– but two answer facts at last!

\[
\text{m_answer}(X) \leftarrow \text{m_query}(X), \text{r}(X).
\]

\[
\text{m_answer}(X) \leftarrow \text{m_query}(X), \text{s}(X,Y), \text{m_answer}(Y).
\]
How does all of this work during fixpoint computation?

4th round of iteration:
The recursive answer rule „fires“, too!

\[ \text{answer} \leftarrow \text{m\_answer}(5). \]

\[ \text{m\_answer}(1). \]
\[ \text{m\_answer}(3). \]
\[ \text{m\_answer}(2). \]

\[ \text{m\_answer}(X) \leftarrow \text{m\_query}(X), \text{r}(X). \]
\[ \text{m\_answer}(X) \leftarrow \text{m\_query}(X), \text{s}(X,Y), \text{m\_answer}(Y). \]
Subquery Approach and Recursion (11)

5th round of iteration:
The recursive answer rule „fires“ once more!

m_answer(X) ← m_query(X), r(X).
m_answer(X) ← m_query(X), s(X,Y), m_answer(Y).

How does all of this work during fixpoint computation?

r(1).
r(3).
r(4).
s(2,1).
s(7,6).
s(2,2).
s(2,3).
s(2,3).
m_query(5).
m_query(2).
m_query(1).
m_query(3).

m_answer(1).
m_answer(3).
m_answer(2).
m_answer(5).

answer ← m_answer(5).
How does all of this work during fixpoint computation?

6th round of iteration:
No more answer facts – but the top query is finally answered as the parameterless literal (representing the expected answer 'yes' is derived, too!

\[
\text{answer} \leftarrow \text{m_answer}(5).
\]

\[
\text{m_answer}(1).
\]
\[
\text{m_answer}(3).
\]
\[
\text{m_answer}(2).
\]
\[
\text{m_answer}(5).
\]

\[
\text{m_answer}(X) \leftarrow \text{m_query}(X), \text{r}(X).
\]
\[
\text{m_answer}(X) \leftarrow \text{m_query}(X), \text{s}(X,Y), \text{m_answer}(Y).
\]

Done !!!
Subquery Approach and Recursion (13)

Final resume for this example:

The transformed rules compute all those m-facts which are relevant for computing the answer to the current query, . . . . . . and nothing more!

answer.

needs
m_answer(5).
needs
m_answer(2).
needs
m_answer(1).
or
m_answer(3).
Subquery Approach: Rules

That’s not really ok – violates the standardization requirement!

The transformation involved in this example is already quite sophisticated – if not magical (unless you understand very well what it is good for)!

Dynamic generation of restrictions

Application of restrictions for controlling answer generation
Back to the Non-Recursive Example

How does the „new“ technique (including query-relations) work for our initial example without recursion – if at all?
Subquery Approach: Dynamic Pushing (1)

\[
\begin{align*}
\{ \{Y : p(2,Y)\}\} & \quad ? \\
n(X) & \leftarrow q(X), t(X).
\end{align*}
\]

\[
\begin{align*}
answer(Y) & \leftarrow p\_answer(2,Y).
q(Y) & \leftarrow r(Y). \\
q(Y) & \leftarrow t(Y).
\end{align*}
\]

\[
\begin{align*}
p(X,Y) & \leftarrow s(X,Y), q(Y). \\
p\_answer(X,Y) & \leftarrow p\_query(X), s(X,Y), q\_answer(Y).
\end{align*}
\]

\[
\begin{align*}
m(X) & \leftarrow s(Y,X), r(X). \\
q\_query(Y) & \leftarrow p\_query(X), s(X,Y).
\end{align*}
\]
Subquery Approach: Dynamic Pushing (2)

\[
\text{answer}(Y) \leftarrow \text{p_answer}(2,Y).
\]

\[
\text{p_answer}(X,Y) \leftarrow \text{p_query}(X), s(X,Y), \text{q_answer}(Y).
\]

\[
\text{q_query}(Y) \leftarrow \text{p_query}(X), s(X,Y).
\]

\[
\text{q_answer}(Y) \leftarrow \text{q_query}(Y), r(Y).
\]

\[
\text{q_answer}(Y) \leftarrow \text{q_query}(Y), t(Y).
\]

Same behaviour as before:
Relevant facts only!

\[
\begin{align*}
\text{t} & \quad \text{r} & \quad \text{s} \\
\text{t(5).} & \quad \text{r(1).} & \quad \text{s(2,1).} \\
\text{t(6).} & \quad \text{r(3).} & \quad \text{s(3,4).} \\
\text{t(7).} & \quad \text{r(4).} & \quad \text{s(2,2).} \\
\end{align*}
\]
Subquery Approach: Dynamic Pushing (3)

\[
\begin{align*}
\text{n(X) } & \leftarrow \text{q(X), t(X).} \\
\text{n_answer(X) } & \leftarrow \text{n_query(X), q_answer(X), t(X).} \\
\text{q_query(X) } & \leftarrow \text{n_query(X).} \\
\text{q_answer(Y) } & \leftarrow \text{q_query(Y), r(Y).} \\
\text{q_answer(Y) } & \leftarrow \text{q_query(Y), t(Y).} \\
\text{p_query(2).} \\
\text{p_answer(X,Y) } & \leftarrow \text{p_query(X), s(X,Y), q_answer(Y).} \\
\text{q_query(Y) } & \leftarrow \text{p_query(X), s(X,Y).} \\
\text{p(X,Y) } & \leftarrow \text{s(X,Y), q(Y).} \\
\text{p(2,1).} \\
\text{p(2,3).} \\
\text{answer(Y) } & \leftarrow \text{p_answer(X,Y).} \\
\text{answer(Y) } & \leftarrow \text{q_query(Y), r(Y).} \\
\text{answer(Y) } & \leftarrow \text{q_query(Y), t(Y).} \\
\text{q_answer(Y) } & \leftarrow \text{q_query(Y), r(Y).} \\
\text{q_answer(Y) } & \leftarrow \text{q_query(Y), t(Y).} \\
\text{answer(Y) } & \leftarrow \text{p_answer(X,Y).} \\
\text{m_answer(X) } & \leftarrow \text{m_query(X), r(X).} \\
\text{m_answer(X) } & \leftarrow \text{m_query(X), s(X,Y), m_answer(Y).} \\
\text{m_query(Y) } & \leftarrow \text{m_query(X), s(X,Y).} \\
\end{align*}
\]
Subquery Approach: Dynamic Pushing (4)

Dependency graph of the original rules

Query-facts representing the current top query do „switch on“ relevant rules only!

Dependency graph of the internal rules