2.1 The Magic Sets Approach
Recursive rules represent the key example where the ‘pushing selections’-strategy for constants or even literals fails:

queries for all nodes reachable from 1

not equivalent anymore
Magic Sets (1)

Magic Sets generalizes the pushing selections approach to recursive rules:

?- p(1,Y)

\[ p(X, Y) \leftarrow r(X, Z), p(Z, Y) \]
Magic Sets generalizes the pushing selections approach to recursive rules:
Magic Sets generalizes the pushing selections approach to recursive rules:
Relation names are adorned with binding patterns:

?- p^{bf}(1,Y)

\[ p^{bf}(X,Y) \leftrightarrow e^{bf}(X,Z), \quad p^{bf}(Z,Y) \]
Magic Sets (5)

Subqueries for dynamically generating new selection constants are defined by (magic) query rules:

?- p^{bf}(1,Y) \rightarrow m_{p^{bf}}(1)

p^{bf}(X,Y) \iff m_{p^{bf}}(X), e^{bf}(X,Z), p^{bf}(Z,Y)

m_{e^{bf}}(X) \iff m_{p^{bf}}(X)

m_{p^{bf}}(Z) \iff m_{p^{bf}}(X), e^{bf}(X,Z)
Subquery rules together with answer rules represent a complete and sound search strategy:

path is the transitive closure of edge

```
res(Y) ← path(X,Y), X=1
path(X,Y) ← edge(X,Y)
path(X,Y) ← edge(X,Z), path(Z,Y)
```

queries for all nodes reachable from 1

```
res(Y) ← path(X,Y)
pathbf(X,Y) ← m_pathbf(X), edge(X,Y)
pathbf(X,Y) ← m_pathbf(X), edge(X,Z), pathbf(Z,Y)

m_pathbf(Z) ← m_pathbf(X), edge(X,Z)
```

query `?-path(1,Y)`
Magic Sets: Overview

- The different ideas discussed and motivated in the previous slides have been summarized in a method for deductive query evaluation which is well-known in the database community by now and even begins to influence commercial SQL systems:

  - This method has been introduced in 1986 by a group of scientists in the USA (F. Bancilhon, D. Maier, J. Ullman, Y. Sagiv), who published an article entitled "Magic Sets and other strange ways to implement logic programs."

  - The real relationship between this method and "classical" optimization techniques in SQL (but also with techniques of Logic Programming like resolution and tabulation) have been understood only recently.

- We will first treat the simple case only, i.e., we will consider only positive Datalog without built-ins – negation will be admitted later.
MS is a Transformation-Based Approach

Rule Compiler

\( R \) → \( R_{\text{magic}} \)

Internal Rules

per query pattern

Query

Fixpoint Iteration

Answers and intermediate results

\( F \)
The most important part of the Magic Sets method is the rule compilation: In this phase, the ideas of a dynamic, logical optimization via internal temporary relations are applied systematically to all rules.

From a given set $R$ of rules, an internal rule set $R_{\text{magic}}$ is generated via an (automatically performed) transformation. This internal rule set is “visible” to the DBMS only and is used for making query evaluation efficient.

$R_{\text{magic}}$ itself consists of two subsets, . . .
- a set $R_{\text{answer}}$ of answer generating rules and . . .
- a set of rules $R_{\text{query}}$ dynamically generating subqueries (resp. their input bindings).

Answer rules are variants of the original application rules, generating relevant parts of the full relations only containing all answers to the top query and all its subqueries.

Query rules define auxiliary relations representing subqueries – the “magic sets”!
Rule Compilation: Example

"Transitive Closure" example:

Input on the 1st position

Internal rules for query pattern p(X,Y):

\[
\text{answer}_p(X,Y) \leftarrow \text{query}_p^{bf}(X), \ e(X,Y).
\]

\[
\text{answer}_p(X,Y) \leftarrow \text{query}_p^{bf}(X), \ e(X,Z), \ \text{answer}_p(Z,Y).
\]

\[
\text{query}_p^{bf}(Z) \leftarrow \text{query}_p^{bf}(X), \ e(X,Z).
\]

Output:

\[
p(X,Y) \leftarrow e(X,Y).
\]

\[
p(X,Y) \leftarrow e(X,Z), \ p(Z,Y).
\]

(p: path; e: edge)
The query rule specifies a top-down and left-to-right propagation of input bindings on the 1st parameter position "through" the recursive literal:

query_pbf(Z) ← query_pbf(X), e(X,Z).

If 'p' is called with input X . . .

. . . and X is linked via 'e' with Z, . . .

. . . then 'p' will be called recursively with input Z.
The **answer rules** define a sub-relation of \( p \) containing "queried" \( p \)-facts only:

\[
\begin{align*}
\text{answer}_p(X,Y) & \leftarrow \text{query}_p(X), \ k(X,Y). \\
\text{answer}_p(X,Y) & \leftarrow \text{query}_p(X), \ k(X,Z), \ \text{answer}_p(Z,Y).
\end{align*}
\]

"Switch literals":
Prevent production of answers for which there are no (sub-)queries

Sub-relation of \( p \):
"relevant" \( p \)-facts only
In the original "Magic Sets" papers, the internal relations were called differently:
- Prefix `answer` was omitted
- Prefix `query` was called `magic`.

Of course, we believe that our version is more intuitive!
Query Answering by Fixpoint Iterations

\[
\begin{align*}
\text{answer}_p(X,Y) &\leftarrow \text{query}_p^{bf}(X), \ e(X,Y). \\
\text{answer}_p(X,Y) &\leftarrow \text{query}_p^{bf}(X), \ e(X,Z), \ \text{answer}_p(Z,Y).
\end{align*}
\]

\[
\{ \ (Y) : \ p(1, Y) \ \} \ ?
\]

\[
\begin{array}{c|c|c}
\text{Iteration round} & \text{query}_p^{bf} & \text{answer}_p \\
\hline
0 & - & - \\
1 & 1 & 12 \\
 & 2 & 13 \\
2 & 3 & 14 \\
 & 4 & 24 \\
3 & - & 41 \\
 & & 42 \\
4 & & 43 \\
5 & & 21 \\
6 & & 22 \\
\end{array}
\]

\[
\text{F:}
\]

\[
\begin{align*}
e(1,2). \\
e(1,3). \\
e(2,4). \\
e(4,1).
\end{align*}
\]
The initial fact in the query-relation, containing the input-constant of the query is called a 'seed fact', because all derived facts "grow" out of it.

\[
\{ (Y) : p(1, Y) \} \quad ?
\]

<table>
<thead>
<tr>
<th>Iteration round</th>
<th>query_p^{bf}</th>
<th>answer_p</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Last iteration: Test of fixpoint property

Generation of subqueries

( interleaved with )

Generation of (intermediate) results
The Magic Sets transformation ("Rule compilation") depends on the binding pattern of the respective query. Up till now, just one (example) pattern has been discussed:

- If the input variable binding goes to a different variable of the query-predicate 'p', then there may be a different form of propagation of subqueries that is recommendable, and thus a different form of internal rules required, e.g.:

  \[
  \begin{align*}
  p(X,Y) & \leftarrow e(X,Y). \\
  p(X,Y) & \leftarrow e(X,Z), p(Z,Y).
  \end{align*}
  \]

If the input variable binding goes to a different variable of the query-predicate 'p', then there may be a different form of propagation of subqueries that is recommendable, and thus a different form of internal rules required, e.g.:
MS Transformation for Other Query Patterns

Variable bindings are identically transferred to the recursive p-literal:

This query-rule is completely unproductive, as it just „confirms“ the Y-binding present anyway. Thus, we don‘t generate such query-rules at all!

There is just a seed fact for this kind of query-relations.
MS Transformation for Other Query Patterns

answer_p(X,Y) ← query_p^{fb}(Y), e(X,Y).
answer_p(X,Y) ← query_p^{fb}(Y), answer_p(Z,Y), e(X,Z).

\{ (X) : p(X, 1) \} ?

<table>
<thead>
<tr>
<th>Iteration round</th>
<th>query_p^{fb}</th>
<th>answer_p</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2 1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1 1</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

For a different query pattern the transformed rules may look completely different, and the corresponding fixpoint iteration „works“ very differently, too.
MS Transformation for Other Query Patterns

Pure test query:

Input on 1st and 2nd position

\[ p(X,Y) \]

\[ e(X,Z) \]

\[ p(Z,Y) \]

1. \[ p(X,Y) \]
2. \[ e(X,Z) \]
3. \[ p(Z,Y) \]

**R\text{\_query}**

\[ \text{query\_p}^{bb}(Z,Y) \leftarrow \text{query\_p}^{bb}(X,Y), \ e(X,Z). \]

**R\text{\_answer}**

\[ \text{answer\_p}(X,Y) \leftarrow \text{query\_p}^{bb}(X,Y), \ e(X,Y). \]

\[ \text{answer\_p}(X,Y) \leftarrow \text{query\_p}^{bb}(X,Y), \ e(X,Z), \ \text{answer\_p}(Z,Y). \]

If no input bindings are present (query \{(X,Y): p(X,Y)\}), then there is no query-rule. Thus, parameterless "switch literals" 'query\_p' are used in each answer-rule.
MS for the "Transitive Closure": Summary

The internal rules are pre-compiled for all possible query patterns directly after Schema design and may thus be activated any time by query-specific seed facts:

\[
\text{answer}_p(X,Y) \leftarrow \text{query}_p, \ e(X,Y).
\]

\[
\text{answer}_p(X,Y) \leftarrow \text{query}_p, \ e(X,Z), \ \text{answer}_p(Z,Y).
\]

\[
\text{answer}_p(X,Y) \leftarrow \text{query}_p^{bf}(X), \ e(X,Y).
\]

\[
\text{answer}_p(X,Y) \leftarrow \text{query}_p^{bf}(X), \ e(X,Z), \ \text{answer}_p(Z,Y).
\]

\[
\text{answer}_p(X,Y) \leftarrow \text{query}_p^{bb}(X,Y), \ e(X,Y).
\]

\[
\text{answer}_p(X,Y) \leftarrow \text{query}_p^{bb}(X,Y), \ e(X,Z), \ \text{answer}_p(Z,Y).
\]

\[
\text{query}_p^{bf}(Z) \leftarrow \text{query}_p^{bf}(X), \ e(X,Z).
\]

\[
\text{query}_p^{bb}(Z,Y) \leftarrow \text{query}_p^{bb}(X,Y), \ e(X,Z).
\]
MS for the "Transitive Closure": Summary

\begin{align*}
\text{answer}_p(X,Y) & \leftarrow \text{query}_p(X,Y). \\
\text{answer}_p(X,Y) & \leftarrow \text{query}_p(X,Z), \text{answer}_p(Z,Y). \\
\text{answer}_p(X,Y) & \leftarrow \text{query}_pbf(X), \text{e}(X,Y). \\
\text{answer}_p(X,Y) & \leftarrow \text{query}_pbf(X), \text{e}(X,Z), \text{answer}_p(Z,Y). \\
\text{answer}_p(X,Y) & \leftarrow \text{query}_pbb(Y), \text{e}(X,Y). \\
\text{answer}_p(X,Y) & \leftarrow \text{query}_pbb(Y), \text{answer}_p(Z,Y), \text{e}(X,Z). \\
\text{answer}_p(X,Y) & \leftarrow \text{query}_pbb(X,Y), \text{e}(X,Y). \\
\text{answer}_p(X,Y) & \leftarrow \text{query}_pbb(X,Y), \text{e}(X,Z), \text{answer}_p(Z,Y). \\
\end{align*}

\begin{align*}
\text{query}_pbf(Z) & \leftarrow \text{query}_pbf(X), \text{e}(X,Z). \\
\text{query}_pbb(Z,Y) & \leftarrow \text{query}_pbb(X,Y), \text{e}(X,Z). \\
\end{align*}
Supplementary Relations (1)

In general, query- and answer-rules for the same binding pattern may contain several common subexpressions, which would be evaluated several times if materializing the answer relations:

In the diagram, the common subexpressions are highlighted in blue.
Supplementary Relations (2)

It is recommendable to compute these relations (containing intermediate results) only once! This can be achieved by introducing rules defining so called "supplementary" relations, referenced in those parts of the magic rules where the common subexpressions occur:

$$\text{supp}_{pbf}(X,Z) \leftarrow \text{query}_{pbf}(X), \ e(X,Z).$$
$$\text{supp}_{pbb}(X,Y,Z) \leftarrow \text{query}_{pbb}(X,Y), \ e(X,Z).$$

$$\text{query}_{pbf}(Z) \leftarrow \text{supp}_{pbf}(X,Z).$$
$$\text{query}_{pbb}(Z,Y) \leftarrow \text{supp}_{pbb}(X,Y,Z).$$

$$\text{answer}_{p}(X,Y) \leftarrow \text{query}_{p}, \ e(X,Y).$$
$$\text{answer}_{p}(X,Y) \leftarrow \text{query}_{p}, \ e(X,Z), \ \text{answer}_{p}(Z,Y).$$
$$\text{answer}_{p}(X,Y) \leftarrow \text{supp}_{pbf}(X,Y).$$
$$\text{answer}_{p}(X,Y) \leftarrow \text{supp}_{pbf}(X,Z), \ \text{answer}_{p}(Z,Y).$$
$$\text{answer}_{p}(X,Y) \leftarrow \text{query}_{pfb}(Y), \ e(X,Y).$$
$$\text{answer}_{p}(X,Y) \leftarrow \text{query}_{pfb}(Y), \ \text{answer}_{p}(Z,Y), \ e(X,Z).$$
$$\text{answer}_{p}(X,Y) \leftarrow \text{supp}_{pbb}(X,Y,Y).$$
$$\text{answer}_{p}(X,Y) \leftarrow \text{supp}_{pbb}(X,Y,Z), \ \text{answer}_{p}(Z,Y).$$
Adornments (1)

- The query relations associated with a relation p have one parameter for each of the input parameters of p in the binding pattern under consideration, e.g.:

  \[ p(X,Y): \]

  \[ \text{Input on 1}\text{st position} \]

  \[ \text{query}_p^{bf}(X), \]

- If p has several parameters, some of which are no input constants, there would be an ambiguity with respect to the position of the parameters of query_p (representing input bindings throughout) in comparison to those of p: query_p(X) may represent a binding on the 1\text{st} or the 2\text{nd} position in the parameter list of p!

- Therefore, the assignment of parameters of query_p to those of p is explicitly represented by an upper index: The parameter of query_p^{bf} corresponds to an input on the 1\text{st} position of p, whereas query_p^{fb} corresponds to the 2\text{nd} parameter position of p.
Adornments (2)

- The indexes of the query-relations thus generated (used for the corresp. supplementary-relations, too), have been called adornments by the authors of the magic set papers (as they are like little „crowns“ sitting on these literals).

- The answer-relations do not use adornments, as we need all parameters in an answer even in case where the parameter value is already fixed in the resp. query.

- **Reason:** Answer relations do not only contain answers to the top query (issued by the user), but contain answers to subqueries (arising dynamically during evaluation), too. Such answers to subqueries can be viewed as relevant intermediate results from the „perspective“ of the top query.
Passing Sub-Query Bindings with MS

\[ p(X, Y) \leftarrow q(X, Z), r(Y, Z), s(X, W), t(W, Y). \]

\[
\begin{align*}
\{ (X) : p(X, 1) \} & \rightarrow \{ (Y) : p(2, Y) \} \\
\{ (X) : p(X, 1) \} & \rightarrow \{ (X,Y) : p(X, Y) \} \\
\{ (X) : p(2, 1) \} & \rightarrow \{ (X,Y) : p(X, Y) \}
\end{align*}
\]