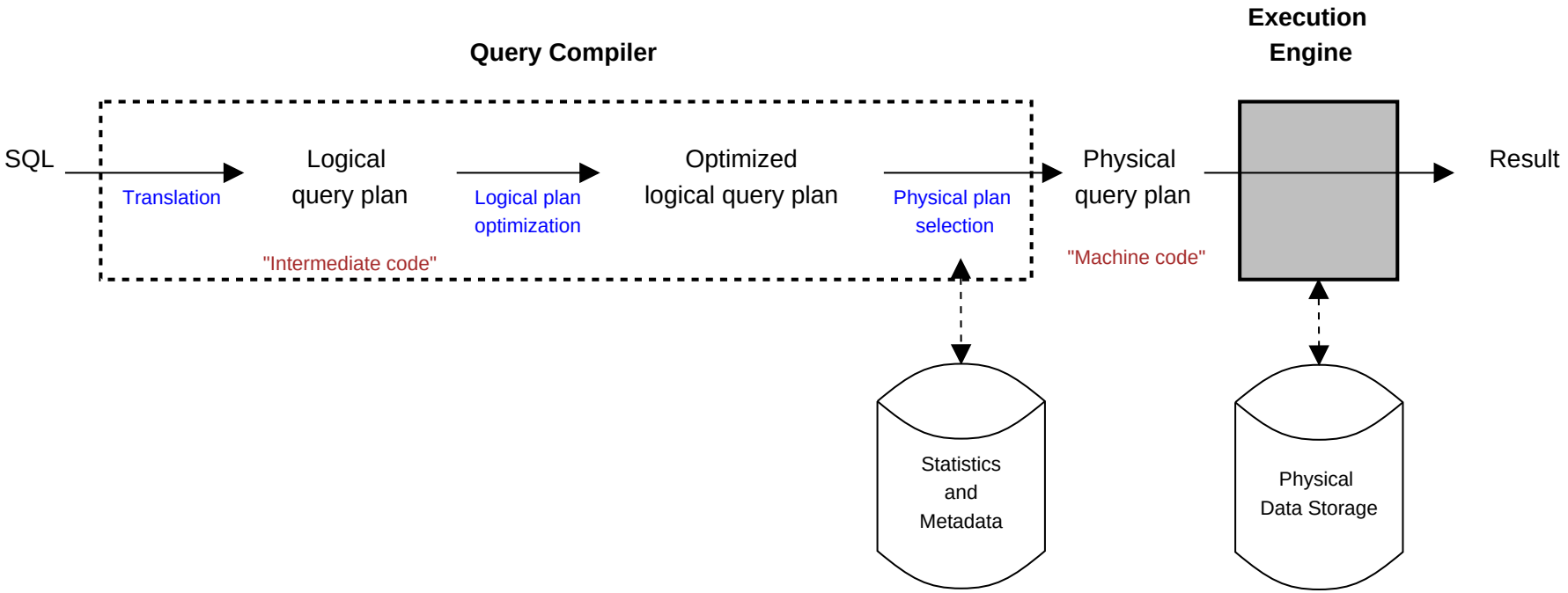
Optimization of logical query plans

Eliminating redundant joins

Optimization of logical query plans



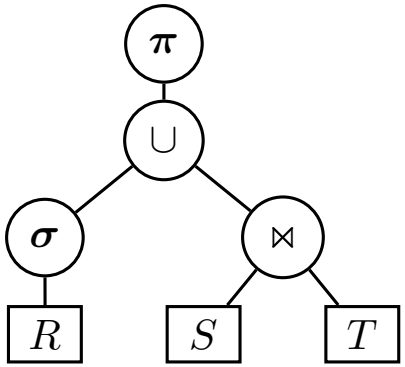
Optimization of logical query plans

Logical plans = execution trees

A logical query plan like

$$\pi_{A,B}(\sigma_{A=5}(R) \cup (S \bowtie T))$$

is essentially an execution tree



A physical query plan is a logical query plan in which each node is assigned the algorithm that should be used to evaluate the corresponding relational algebra operator. (There are multiple ways to evaluate each algebra operator)

Optimization of logical query plans

The need to optimize

- Every internal node (i.e., each occurrence of an operator) in the plan must be executed.
- Hence, the fewer nodes we have, the faster the execution

Definition

A relational algebra expression e is optimal if there is no other expression e' that is both (1) equivalent to e and (2) "shorter" than e (i.e., has fewer occurrence of operators than e).

The optimization problem:

Input: a relational algebra expression e

Output: an optimal relational algebra expression e' equivalent to e.

Optimization of logical query plans

The optimization problem is undecidable:

It is known that the following problem is undecidable:

Input: relational algebra expressions e_1 and e_2 over a single relation $R(A, B)$
Output: $e_1 \equiv e_2$?

Proof: Suppose that we can compute e', the optimal expression for $(e_1 - e_2) \cup (e_2 - e_1)$. Note that $e_1 \equiv e_2$ if, and only if, e' is either $\sigma_{\texttt{false}}(R)$ or $R - R$.

Conclusion: the optimization problem is undecidable

Question: **can we optimize plans that are of a particular form?**

Optimization of select-project-join expressions

In practice, most SQL queries are of the following form:

```
SELECT ...
FROM R1, R2, ..., Rm
WHERE A1 = B1 AND A2 = B2 AND ...  AND An = Bn
```

The corresponding logical query plans are of the form:

$$\boldsymbol{\pi}_{\ldots}\, \boldsymbol{\sigma}_{A_1=B_1 \wedge A_2=B_2 \wedge \cdots \wedge A_n=B_n}\, (R_1 \times R_2 \times \cdots \times R_m).$$

We call such relational algebra expressions select-project-join expressions (SPJ expressions for short)

Optimization of select-project-join expressions

Removing redundant joins:

A careless SQL programmer writes the following query:

```
SELECT movieTitle FROM StarsIn S1
WHERE starName IN (SELECT name
                   FROM MovieStar, StarsIn S2
                   WHERE birthdate = 1960
                    AND S2.movieTitle = S1.movieTitle)
```

This query is equivalent to the following one, which has one join less to execute!

```
SELECT movieTitle FROM StarsIn
WHERE starName IN (SELECT name
                   FROM MovieStar
                   WHERE birthdate = 1960)
```

Optimization of select-project-join expressions

Here are the corresponding logical query plans:

$$\boldsymbol{\pi}_{S_1.\texttt{movieTitle}}\big(\boldsymbol{\rho}_{S_2}(\texttt{StarsIn}) \underset{S_2.\texttt{movieTitle}=S_1.\texttt{movieTitle}}{\bowtie} \boldsymbol{\rho}_{S_1}(\texttt{StarsIn}) \underset{S_1.\texttt{starName}=\texttt{name}}{\bowtie} \boldsymbol{\sigma}_{\texttt{birthdate}=1960}(\texttt{MovieStar})\big)$$

versus

$$\boldsymbol{\pi}_{S_1.\texttt{movieTitle}}\big(\boldsymbol{\rho}_{S_1}(\texttt{StarsIn}) \underset{S_1.\texttt{starName}=\texttt{name}}{\bowtie} \boldsymbol{\sigma}_{\texttt{birthdate}=1960}(\texttt{MovieStar})\big).$$

Redundant joins may also be introduced because of view expansion

Can we optimize select-project-join expressions? (And hence remove redundant joins?)

Optimization of select-project-join expressions

Definition

A conjunctive query is an expression of the form

$$Q\underbrace{(x_1, \ldots, x_n)}_{\text{head}} \leftarrow \underbrace{R(t_1, \ldots, t_m), \ldots, S(t'_1, \ldots, t'_k)}_{\text{body}}$$

Here $t_1, \ldots, t'_k$ denote variables and constants, and $x_1, \ldots, x_n$ must be variables that occur in $t_1, \ldots, t'_k$. We call an expression like $R(t_1, \ldots, t_m)$ an atom. If an atom does not contain any variables, and hence consists solely of contstants, then it is called a fact.

Optimization of select-project-join expressions

Semantics of conjunctive queries

Consider the following toy database D:

R

1	2
2	3
2	5
6	7
7	5
5	5

S

2
7

as well as the following conjunctive query over the relations $R(A, B)$ and $S(C)$:

$$Q(x, y) \leftarrow R(x, y),\ R(y, 5),\ S(y).$$

Intuitively, Q wants to retrieve all pairs of values (x, y) such that (1) this pair occurs in relation R; (2) y occurs together with the constant 5 in a tuple in R; and (3) y occurs as a value in S. The formal definition is as follows.

Optimization of select-project-join expressions

Semantics of conjunctive queries

Consider the following toy database D:

R

1	2
2	3
2	5
6	7
7	5
5	5

S

2
7

as well as the following conjunctive query over the relations $R(A, B)$ and $S(C)$:

$$Q(x, y) \leftarrow R(x, y),\ R(y, 5),\ S(y).$$

A substitution f of Q into D is a function that maps variables in Q to constants in D. For example:

$$f: \begin{array}{ccc} x & \mapsto & 1 \\ y & & 2 \end{array}$$

Optimization of select-project-join expressions

Semantics of conjunctive queries

Consider the following toy database D:

R

1	2
2	3
2	5
6	7
7	5
5	5

S

2
7

as well as the following conjunctive query over the relations $R(A, B)$ and $S(C)$:

$$Q(x, y) \leftarrow R(x, y),\ R(y, 5),\ S(y).$$

A matching is a substitution that maps the body of Q into facts in D. For example:

$$f: \begin{array}{ccc} x & \mapsto & 1 \\ y & & 2 \end{array}$$

Optimization of select-project-join expressions

Semantics of conjunctive queries

Consider the following toy database D:

R

1	2
2	3
2	5
6	7
7	5
5	5

S

2
7

as well as the following conjunctive query over the relations $R(A, B)$ and $S(C)$:

$$Q(x, y) \leftarrow R(x, y),\ R(y, 5),\ S(y).$$

The result of a conjunctive query is obtained by applying all possible matchings to the head of the query. In our example:

$$Q(D) = \{(1, 2), (6, 7)\}.$$

Optimization of select-project-join expressions

Translation of SPJ expressions into conjunctive queries

SPJ: $\boldsymbol{\pi}_{\texttt{title}}(\texttt{Movie} \underset{\texttt{title=movieTitle}}{\bowtie} \texttt{StarsIn} \underset{\texttt{starName=name}}{\bowtie} \boldsymbol{\sigma}_{\texttt{birthdate}=1960}(\texttt{MovieStar}))$

CQ: $Q_1(t) \leftarrow \texttt{Movie}(t, y, \ell, i, s, p),\ \texttt{StarsIn}(t, y_2, n),\ \texttt{MovieStar}(n, a, g, 1960)$

Translation of conjunctive queries into SPJ expressions

CQ: $Q(x, y) \leftarrow R(x, y),\ R(y, 5),\ S(y)$

SPJ: $\boldsymbol{\pi}_{R_1.A, R_1.B}\, \boldsymbol{\sigma}_{R_1.B=R_2.A}\, \boldsymbol{\sigma}_{R_2.B=5}\, \boldsymbol{\sigma}_{R_1.B=C}(\boldsymbol{\rho}_{R_1}(R) \times \boldsymbol{\rho}_{R_2}(R) \times S)$

Conclusion:

Select-project-join expressions and conjunctive queries are two separate syntaxes for the same class of queries.

Optimization of select-project-join expressions

In-class exercise

Consider the following SQL expression over the relations $R(A, B)$ and $S(B, C)$:

```
SELECT R1.A, S1.B
FROM R R1, R R2, R R3, S S1, S S2
WHERE
    R1.A = R2.A AND R2.B =4 AND R2.A = R3.A
AND R3.B = S1.B AND S1.C = S2.C AND S2.B=4
```

Exercise: Translate this SQL expression into a logical query plan. If this plan is an SPJ-expression, then translate this plan into a conjunctive query.

Optimization of select-project-join expressions

Another in-class exercise

Consider the following conjunctive query over the relations

- MovieStar(name: `string`, address: `string`, gender: `char`, birthdate: `date`)
- StarsIn(movieTitle: `string`, movieYear: `string`, starName: `string`)

$$Q(t) \leftarrow \texttt{MovieStar}(n, a, g, 1940), \texttt{StarsIn}(t, y, n)$$

Translate this conjunctive query into an SPJ expression. What is the corresponding SQL query?

Optimization of select-project-join expressions

Containment of conjunctive queries

Q_1 is contained in Q_2 if $Q_1(D) \subseteq Q_2(D)$, for every database D.

Example:

$$A(x, y) \leftarrow R(x, w),\ G(w, z),\ R(z, y)$$
$$B(x, y) \leftarrow R(x, w),\ G(w, w),\ R(w, y)$$

Then B is contained in A. Proof:

1. Let D be an arbitrary database, and let $t \in B(D)$.
2. Then there exists a matching f of B into D such that $t = (f(x), f(y))$. We need to show that $(f(x), f(y)) \in A(D)$.
3. Let h be the following substitution:

$$x \to f(x) \quad y \to f(y) \quad w \to f(w) \quad z \to f(w)$$

4. Then h is a matching of A into D, and hence

$$t = (f(x), f(y)) = (h(x), h(y)) \in A(D)$$

Optimization of select-project-join expressions

Containment of conjunctive queries is decidable

$$A(x, y) \leftarrow R(x, w),\ G(w, z),\ R(z, y)$$
$$B(x, y) \leftarrow R(x, w),\ G(w, w),\ R(w, y)$$

Golden method to check whether $B \subseteq A$:

1. First calculate the canonical database D for B:

R	
x	w
w	y

G	
w	w

2. Then check whether $(x, y) \in A(D)$. If so, $B \subseteq A$, otherwise $B \not\subseteq A$.

Optimization of select-project-join expressions

Containment of conjunctive queries is decidable

$$A(x, y) \leftarrow R(x, w),\ G(w, z),\ R(z, y)$$
$$B(x, y) \leftarrow R(x, w),\ G(w, w),\ R(w, y)$$

Fact: $B \subseteq A \Leftrightarrow (x, y) \in A(D)$ with D the canonical database for B.

First possibility: $(x, y) \notin A(D)$

In this case we have just constructed a counter-example because $(x, y) \in B(D)$.

R

x	w
w	y

G

w	w

Optimization of select-project-join expressions

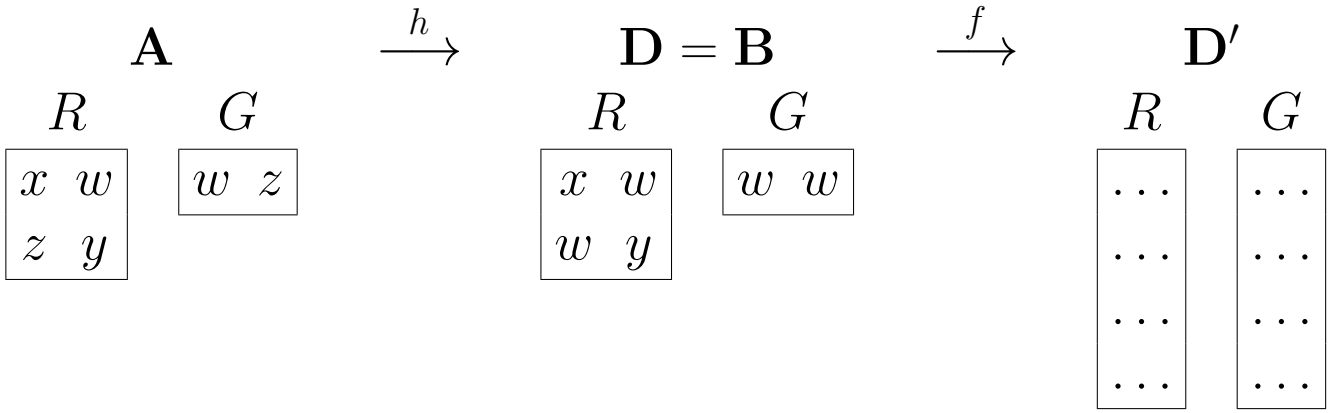
Containment of conjunctive queries is decidable

$$A(x, y) \leftarrow R(x, w),\ G(w, z),\ R(z, y)$$
$$B(x, y) \leftarrow R(x, w),\ G(w, w),\ R(w, y)$$

Fact: $B \subseteq A \Leftrightarrow (x, y) \in A(D)$ with D the canonical database for B.

Second possibility: $(x, y) \in A(D)$

- There hence exists a matching h of A into D such that $h(x) = x$ and $h(y) = y$
- Let D' be an arbitrary other database, and pick $t \in B(D')$. There hence exists a matching f such that $t = (f(x), f(y))$.
- Then $f \circ h$ is a matching of A on D':



Optimization of select-project-join expressions

Containment of conjunctive queries is decidable

$$A(x, y) \leftarrow R(x, w),\ G(w, z),\ R(z, y)$$
$$B(x, y) \leftarrow R(x, w),\ G(w, w),\ R(w, y)$$

Fact: $B \subseteq A \Leftrightarrow (x, y) \in A(D)$ with D the canonical database for B.

Second possibility: $(x, y) \in A(D)$

- There hence exists a matching h of A into D such that $h(x) = x$ and $h(y) = y$
- Let D' be an arbitrary other database, and pick $t \in B(D')$. There hence exists a matching f such that $t = (f(x), f(y))$.
- Then $f \circ h$ is a matching of A on D':
- And hence

$$t = (f(x), f(y)) = (f(h(x)), f(h(y)) \in A(D')$$

Optimization of select-project-join expressions

Conclusion:

- Containment of conjunctive queries is decidable
- Consequently the equivalence of conjunctive queries is also decidable

Optimization of select-project-join expressions

Optimizing conjunctive queries

Input: A conjunctive query Q

Output: A conjunctive query Q' equivalent to Q that is optimal (i.e., has the least number of atoms in its body).

For each conjunctive query we can obtain an equivalent, optimal query, by removing atoms from its body

- Let Q be a CQ and let P be an arbitrary optimal and equivalent query.
- Then $Q \subseteq P$ and hence $(x, y) \in P(D_Q)$ with D_Q the canonical database for Q. Let f be the matching that ensures this fact.
- Let Q' be obtained by removing from Q all atoms that are not in the range of f
- Then $Q \subseteq Q'$
- Moreover, also $Q' \subseteq P$ (because $(x, y) \in P(D_{Q'})$ still holds) and $P \subseteq Q$. Hence $Q' \equiv Q$.
- Note that Q' contains at most the same number of atoms as P. Hence Q' is optimal.

Optimization of select-project-join expressions

Optimization of conjunctive queries

Input: A conjunctive query Q

Output: A conjunctive query Q' equivalent to Q that is optimal (i.e., has the least number of atoms in its body).

Optimization algorithm

- A conjunctive query is given. Consider for example:

  $$Q(x) \leftarrow R(x, x),\ R(x, y)$$

- We check, atom by atom, what atoms in its body are redundant.

  In our example we first try to delete $R(x, x)$:

  $$Q_1(x) \leftarrow R(x, y)$$

  Note that $Q \subseteq Q_1$ but $Q_1 \not\subseteq Q$. We hence cannot remove this atom.

Optimization of select-project-join expressions

Optimization of conjunctive queries

Input: A conjunctive query Q

Output: A conjunctive query Q' equivalent to Q that is optimal (i.e., has the least number of atoms in its body).

Optimization algorithm

- A conjunctive query is given. Consider for example:

  $$Q(x) \leftarrow R(x, x),\ R(x, y)$$

- We check, atom by atom, what atoms in its body are redundant.

  We next try to remove $R(x, y)$:

  $$Q_2(x) \leftarrow R(x, x)$$

  Note that $Q \subseteq Q_2$ and $Q_2 \subseteq Q$.

  Q_2 is certainly shorter than Q and hence closer to the optimal query. Since there remain no other atoms to test, our result is Q_2.

Optimization of select-project-join expressions

Optimization of select-project-join expressions

1. Translate the select-project-join expression e into an conjunctive query Q.
2. Optimize Q.
3. Translate Q back into a select-project-join expression.

Optimization of logical query plans

Optimization of complete, arbitrary query plans

1. Detect and optimize the select-project-join sub-expressions in the plan
2. Then use heuristics to further optimize the modified plan.

Heuristic: rewriting through algebraic laws

Consider relations $R(A, B)$ and $S(C, D)$ and the following expression that we want to optimize

$$\boldsymbol{\pi}_A \boldsymbol{\sigma}_{A=5 \wedge B<D}(R \times S)$$

Pushing selections:

$$\boldsymbol{\pi}_A \boldsymbol{\sigma}_{B<D}(\boldsymbol{\sigma}_{A=5}(R) \times S)$$

Recognizing joins:

$$\boldsymbol{\pi}_A(\boldsymbol{\sigma}_{A=5}(R) \underset{B<D}{\bowtie} S)$$

Introduce projections where possible:

$$\boldsymbol{\pi}_A(\boldsymbol{\sigma}_{A=5}(R) \underset{B<D}{\bowtie} \pi_D(S))$$

Optimization of logical query plans

An in-class integrated exercise

Recall the relational schema.

- MovieStar(name: `string`, address: `string`, gender: `char`, birthdate: `date`)
- StarsIn(movieTitle: `string`, movieYear: `string`, starName: `string`)

A careless SQL programmer writes the following query:

```
SELECT movieTitle FROM StarsIn S1
WHERE starName IN (SELECT name
                   FROM MovieStar, StarsIn S2
                   WHERE birthdate = 1960
                   AND S2.movieTitle = S1.movieTitle)
```

Translate this query into a logical query plan, remove redundant joins from it, and then use heuristics to further optimize the obtained logical plan.