2. Textbook Query Optimization

- Algebra Revisited
- Canonical Query Translation
- Logical Query Optimization
- Physical Query Optimization
Algebra Revisited

The algebra needs some more thought:

- correctness is critical for query optimization
- can only be guaranteed by a formal model
- the algebra description in the introduction was too cursory

What we ultimately want to do with an algebraic model:

- decide if two algebraic expressions are equivalent (produce the same result)

This is too difficult in practice (not computable in general), so we at least want to:

- guarantee that two algebraic expressions are equivalent (for some classes of expressions)

This still requires a strong formal model. We accept false negatives, but not false positives.
Tuples

Tuple:
- a (unordered) mapping from attribute names to values of a domain
- sample: [name: "Sokrates", age: 69]

Schema:
- a set of attributes with domain, written $A(t)$
- sample: $\{(\text{name}, \text{string}), (\text{age}, \text{number})\}$

Note:
- simplified notation on the slides, but has to be kept in mind
- domain usually omitted when not relevant
- attribute names omitted when schema known
 Tuple Concatenation

- notation: $t_1 \circ t_2$
- sample: $[\text{name: "Sokrates", age: 69}] \circ [\text{country: "Greece"}]$  
  $= [\text{name: "Sokrates", age: 69, country: "Greece"}]$
- note: $t_1 \circ t_2 = t_2 \circ t_1$, tuples are unordered

Requirements/Effects:
- $A(t_1) \cap A(t_2) = \emptyset$
- $A(t_1 \circ t_2) = A(t_1) \cup A(t_2)$
Tuple Projection

Consider $t = \{\text{name: "Sokrates", age: 69, country: "Greece"}\}$

Single Attribute:
- notation $t.a$
- sample: $t.name = "Sokrates"

Multiple Attributes:
- notation $t|_A$
- sample: $t|\{name,age\} = \{\text{name: "Sokrates", age: 69}\}$

Requirements/Effects:
- $a \in A(t), A \subseteq A(t)$
- $A(t|_A) = A$
- notice: $t.a$ produces a value, $t|_A$ produces a tuple
Relations

Relation:
- a set of tuples with the same schema

Schema:
- schema of the contained tuples, written $A(R)$
- sample: { (name,string), (age, number) }
Sets vs. Bags

- relations are sets of tuples
- real data is usually a multi set (bag)

Example:

<table>
<thead>
<tr>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

- we concentrate on sets first for simplicity
- many (but not all) set equivalences valid for bags

The optimizer must consider three different semantics:

- logical algebra operates on bags
- physical algebra operates on streams (order matters)
- explicit duplicate elimination \( \Rightarrow \) sets
Set Operations

Set operations are part of the algebra:

- union \((L \cup R)\), intersection \((L \cap R)\), difference \((L \setminus R)\)
- normal set semantic
- but: schema constraints
- for bags defined via frequencies \((\text{union} \rightarrow +, \text{intersection} \rightarrow \min, \text{difference} \rightarrow -)\)

Requirements/Effects:

- \(A(L) = A(R)\)
- \(A(L \cup R) = A(L) = A(R), A(L \cap R) = A(L) = A(R),\)
  \(A(L \setminus R) = A(L) = A(R)\)
Free Variables

Consider the predicate $age = 62$

- can only be evaluated when $age$ has a meaning
- $age$ behaves a free variable
- must be bound before the predicate can be evaluated
- notation: $F(e)$ are the free variables of $e$

Note:

- free variables are essential for predicates
- free variables are also important for algebra expressions
- dependent join etc.
Selection

Selection:

- notation: $\sigma_p(R)$
- sample: $\sigma_{a \geq 2}([a : 1], [a : 2], [a : 3]) = ([a : 2], [a : 3])$
- predicates can be arbitrarily complex
- optimizer especially interested in predicates of the form $attrib = attrib$ or $attrib = const$

Requirements/Effects:

- $\mathcal{F}(p) \subseteq \mathcal{A}(R)$
- $\mathcal{A}(\sigma_p(R)) = \mathcal{A}(R)$
Projection

Projection:

- notation: $\Pi_A(R)$
- sample: $\Pi_\{a\}(\{[a : 1, b : 1], [a : 2, b : 1]\}) = \{[a : 1], [a : 2]\}$
- eliminates duplicates for set semantic, keeps them for bag semantic
- note: usually written as $\Pi_{a,b}$ instead of the correct $\Pi_\{a,b\}$

Requirements/Effects:

- $A \subseteq A(R)$
- $A(\Pi_A(R)) = A$
Rename

Rename:

- notation: $\rho_{a \rightarrow b}(R)$
- sample:
  $\rho_{a \rightarrow c}([\{a: 1, b: 1\}, \{a: 2, b: 1\}]) = \{[c: 1, b: 1], [c: 2, b: 2]\}$?
- often a pure logical operator, no code generation
- important for the data flow

Requirements/Effects:

- $a \in \mathcal{A}(R), b \notin \mathcal{A}(R)$
- $\mathcal{A}(\rho_{a \rightarrow b}(R)) = \mathcal{A}(R) \setminus \{a\} \cup \{b\}$
Join

Consider \( L = \{[a : 1], [a : 2]\} \), \( R = \{[b : 1], [b : 3]\} \)

Cross Product:
- notation: \( L \times R \)
- sample: \( L \times R = \{[a : 1, b : 1], [a : 1, b : 3], [a : 2, b : 1], [a : 2, b : 3]\} \)

Join:
- notation: \( L \Join_p R \)
- sample: \( L \Join_a=b R = \{[a : 1, b : 1]\} \)
- defined as \( \sigma_p (L \times R) \)

Requirements/Effects:
- \( \mathcal{A}(L) \cap \mathcal{A}(R) = \emptyset \), \( \mathcal{F}(p) \in (\mathcal{A}(L) \cup \mathcal{A}(R)) \)
- \( \mathcal{A}(L \times R) = \mathcal{A}(L) \cup \mathcal{R} \)
Equivalences for selection and projection:

1. \( \sigma_{p_1 \land p_2}(e) \equiv \sigma_{p_1}(\sigma_{p_2}(e)) \) (1)
2. \( \sigma_{p_1}(\sigma_{p_2}(e)) \equiv \sigma_{p_2}(\sigma_{p_1}(e)) \) (2)
3. \( \Pi_{A_1}(\Pi_{A_2}(e)) \equiv \Pi_{A_1}(e) \) (3)
   if \( A_1 \subseteq A_2 \)
4. \( \sigma_p(\Pi_A(e)) \equiv \Pi_A(\sigma_p(e)) \) (4)
   if \( \mathcal{F}(p) \subseteq A \)
5. \( \sigma_p(e_1 \cup e_2) \equiv \sigma_p(e_1) \cup \sigma_p(e_2) \) (5)
6. \( \sigma_p(e_1 \cap e_2) \equiv \sigma_p(e_1) \cap \sigma_p(e_2) \) (6)
7. \( \sigma_p(e_1 \setminus e_2) \equiv \sigma_p(e_1) \setminus \sigma_p(e_2) \) (7)
8. \( \Pi_A(e_1 \cup e_2) \equiv \Pi_A(e_1) \cup \Pi_A(e_2) \) (8)
Equivalences

Equivalences for joins:

\[ e_1 \times e_2 \equiv e_2 \times e_1 \]  \hspace{1cm} (9)

\[ e_1 \bowtie_p e_2 \equiv e_2 \bowtie_p e_1 \]  \hspace{1cm} (10)

\[ (e_1 \times e_2) \times e_3 \equiv e_1 \times (e_2 \times e_3) \]  \hspace{1cm} (11)

\[ (e_1 \bowtie_{p_1} e_2) \bowtie_{p_2} e_3 \equiv e_1 \bowtie_{p_1} (e_2 \bowtie_{p_2} e_3) \]  \hspace{1cm} (12)

\[ \sigma_p(e_1 \times e_2) \equiv e_1 \bowtie_p e_2 \]  \hspace{1cm} (13)

\[ \sigma_p(e_1 \times e_2) \equiv \sigma_p(e_1) \times e_2 \]  \hspace{1cm} (14)

if \( \mathcal{F}(p) \subseteq A(e_1) \)

\[ \sigma_{p_1}(e_1 \bowtie_{p_2} e_2) \equiv \sigma_{p_1}(e_1) \bowtie_{p_2} e_2 \]  \hspace{1cm} (15)

if \( \mathcal{F}(p_1) \subseteq A(e_1) \)

\[ \Pi_A(e_1 \times e_2) \equiv \Pi_{A_1}(e_1) \times \Pi_{A_2}(e_2) \]  \hspace{1cm} (16)

if \( A = A_1 \cup A_2, A_1 \subseteq A(e_1), A_2 \subseteq A(e_2) \)