Query Evaluation

- **Problem**
  - An SQL query is declarative – does not specify a query execution plan.
  - A relational algebra expression is procedural and there is an associated query execution plan.

- **Solution**
  - Convert SQL query to an equivalent relational algebra and evaluate it using the associated query execution plan.
  - *But which equivalent expression is best?*
Naive Conversion

\[
\text{SELECT DISTINCT TargetList FROM R1, R2, ..., RN WHERE Condition}
\]

is equivalent to: \( \pi_{\text{TargetList}}(\sigma_{\text{Condition}}(R_1 \times R_2 \times ... \times R_N)) \)

but this may imply a very inefficient query execution plan.

Example: \( \pi_{\text{Name}}(\sigma_{\text{Id} = \text{ProfId} \land \text{CrsCode} = \text{CS532}}(\text{Professor} \times \text{Teaching})) \)

- Result can be < 100 bytes
- But if each relation is 50K then we end up computing an intermediate result Professor \( \times \) Teaching of size 500M before shrinking it down to just a few bytes.

Problem statement:
Find an equivalent relational algebra expression that can be evaluated “efficiently”.

Query Processing Architecture
### Query Optimizer

- **Uses heuristic algorithms to evaluate relational algebra expressions.** This involves:
  - Estimating the cost of a relational algebra expression
  - Transforming one relational algebra expression to an equivalent one
  - Choosing access paths for evaluating the sub-expressions

- Query optimizers do not “optimize” – just try to find “reasonably good” evaluation strategies

---

### Equivalence Preserving Transformations

- To transform a relational expression into another equivalent expression, we need **transformation rules** that preserve equivalence

- Each transformation rule
  - Is *provably correct* (i.e., does preserve equivalence)
  - Has a heuristic associated with it
Selection and Projection Rules

- **Break complex selection into simpler ones:**
  \[ \sigma_{\text{Cond}_1 \land \text{Cond}_2}(R) \equiv \sigma_{\text{Cond}_1}(\sigma_{\text{Cond}_2}(R)) \]

- **Break projection into stages:**
  \[ \pi_{\text{attr}}(R) \equiv \pi_{\text{attr}}(\pi_{\text{attr}'}(R)), \quad \text{if attr} \subseteq \text{attr}' \]

- **Commute projection and selection:**
  \[ \pi_{\text{attr}}(\sigma_{\text{Cond}}(R)) \equiv \sigma_{\text{Cond}}(\pi_{\text{attr}}(R)), \quad \text{if attr} \supseteq \text{all attributes in Cond} \]

Commutativity and Associativity of Join
(and Cartesian Product as Special Case)

- **Join commutativity:** \( R \bowtie S \equiv S \bowtie R \)
  - used to reduce cost of nested loop evaluation strategies (smaller relation should be in outer loop)

- **Join associativity:** \( R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T \)
  - used to reduce the size of intermediate relations in computation of multi-relational join – first compute the join that yields smaller intermediate result

- **N-way join** has \( T(N) \times N! \) different evaluation plans
  - \( T(N) \) is the number of parenthesized expressions
  - \( N! \) is the number of permutations

- **Query optimizer cannot** look at all plans (might take longer to find an optimal plan than to compute query brute-force). Hence it does not necessarily produce optimal plan
Pushing Selections and Projections

- $\sigma_{\text{Cond}} (R \times S) \equiv R \bowtie_{\text{Cond}} S$
  - $\text{Cond}$ relates attributes of both $R$ and $S$
  - Reduces size of intermediate relation since rows can be discarded sooner

- $\sigma_{\text{Cond}} (R \times S) \equiv \sigma_{\text{Cond}} (R) \times S$
  - $\text{Cond}$ involves only the attributes of $R$
  - Reduces size of intermediate relation since rows of $R$ are discarded sooner

- $\pi_{\text{attr}}(R \times S) \equiv \pi_{\text{attr}}(\pi_{\text{attr}}(R) \times S)$,
  - if $\text{attributes}(R) \supseteq \text{attr'} \supseteq \text{attr} \cap \text{attributes}(R)$
  - reduces the size of an operand of product

Equivalence Example

- $\sigma_{C_1 \land C_2 \land C_3} (R \times S) \equiv \sigma_{C_1} (\sigma_{C_2} (\sigma_{C_3} (R \times S)))$
- $\equiv \sigma_{C_1} (\sigma_{C_2} (R) \times \sigma_{C_3} (S))$
- $\equiv \sigma_{C_2} (R) \bowtie_{C_1} \sigma_{C_3} (S)$

- assuming that
  - $C_2$ involves only attributes of $R$,
  - $C_3$ involves only attributes of $S$, and
  - $C_1$ relates attributes of $R$ and $S$