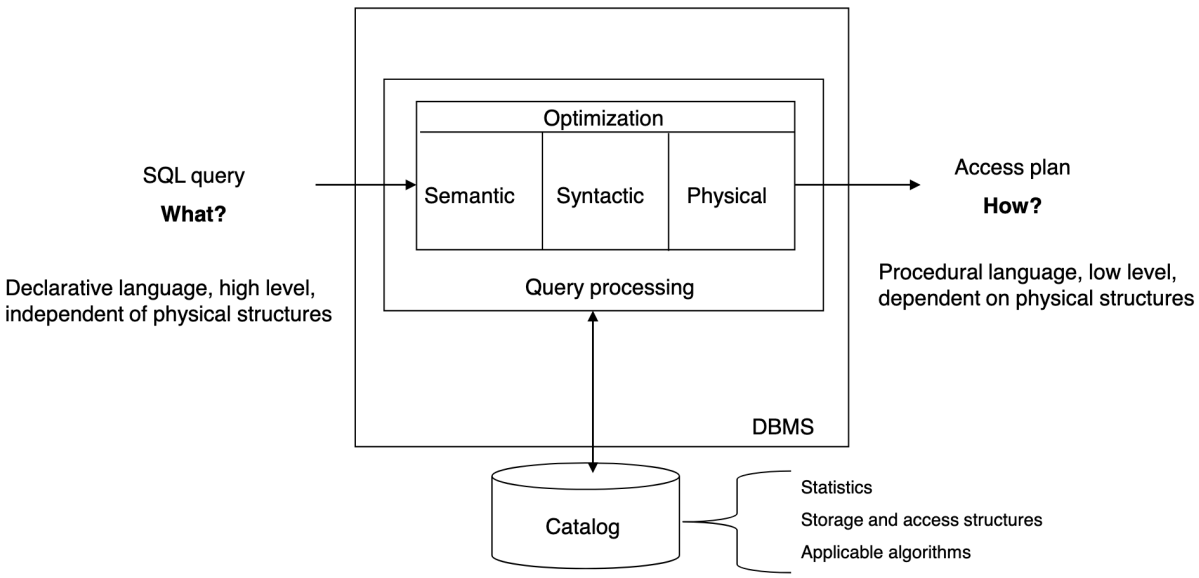
FIB - Disseny de Bases de Dades

Additional Material: Semantic and Syntactic Optimization (Slides)

Preliminary considerations

- Optimization is the last step in query processing
- The input for optimization is an SQL query over tables (or views), syntactically correct and authorized
- The output of optimization is the algorithm (access plan) that must be followed by the DBMS in order to get the result
- The goal is to minimize the use of resources
- In general, a DBMS does not find the optimal access plan, but it obtains an approximation (in a reasonable time)

Architecture



Semantic optimization

Consists of **transforming** the SQL sentence into an **equivalent** one with a lower cost, by considering:

- Integrity constraints
- Logics

Examples of semantic optimization

```
CREATE TABLE students
(
  id CHAR(8) PRIMARY KEY,
  mark FLOAT CHECK (mark>3)
);
```

```
SELECT *
FROM students
WHERE mark<2;

SELECT *
FROM students
WHERE mark<6 AND mark>8;

SELECT *
FROM students
WHERE mark<6 AND mark<7;
```

Example of semantic optimization (ORACLE)

```
SELECT *
FROM employees e, departments d
WHERE e.dpt=d.code AND d.code>5;
```

↓

```
SELECT *
FROM employees e, departments d
WHERE e.dpt=d.code AND d.code>5AND e.dpt>5;
```

Example of semantic optimization (DB2)

```
SELECT *
FROM students
WHERE mark=5 OR mark=6;
```

↓

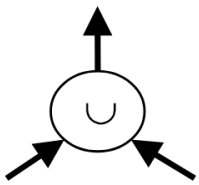
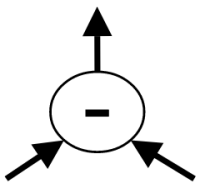
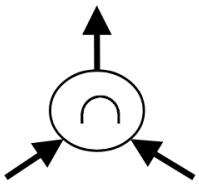
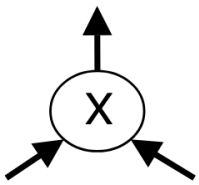
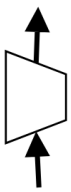
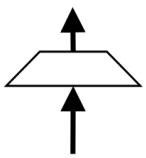
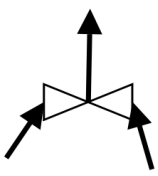
```
SELECT *
FROM students
WHERE mark IN [5, 6];
```

Syntactic optimization

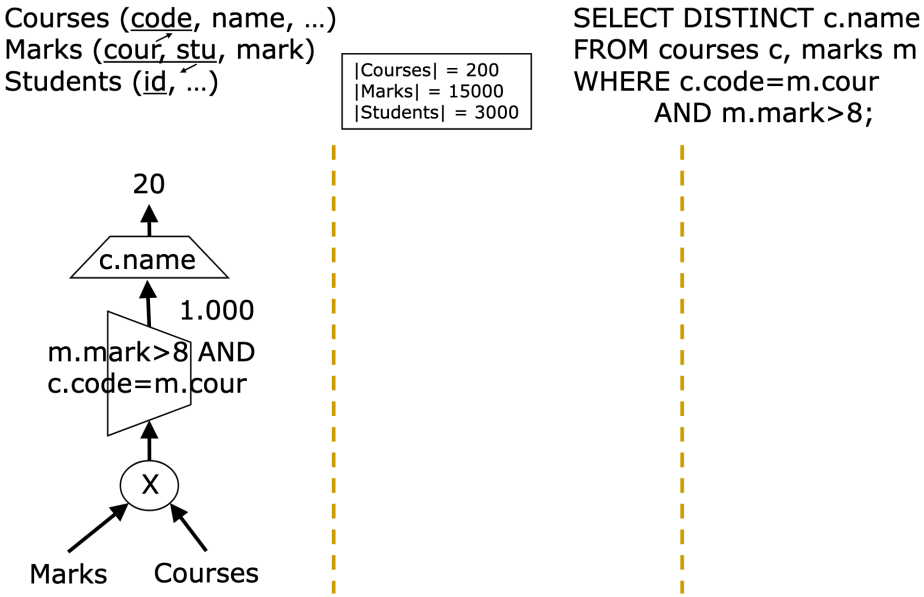
Consists of **translating** the sentence from SQL into a sequence of **algebraic** operations in the form of **syntactic tree**, with minimum cost, by means of **heuristics** (there is more than one solution)

- Nodes
  - Internal: Operations
  - Leaves: Tables
  - Root: Result
- Edges
  - Denote direct usage

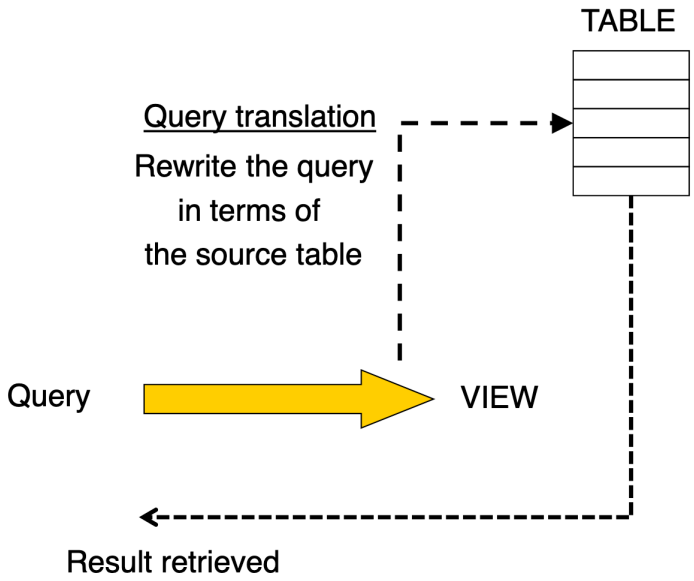
Internal nodes of the syntactic tree

What	Notation
Union	
Difference	
Intersection	
Cross product	
Selection	
Projection	
Join	

Example of syntactic optimization

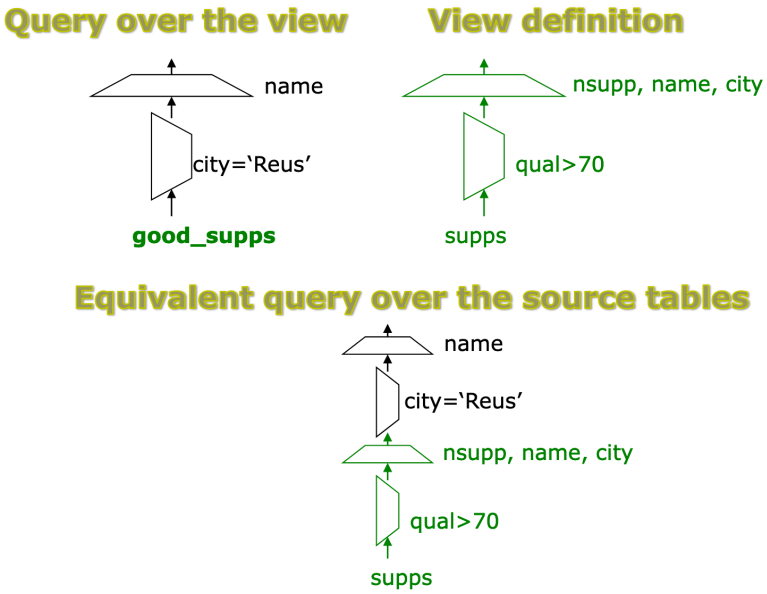


View expansion



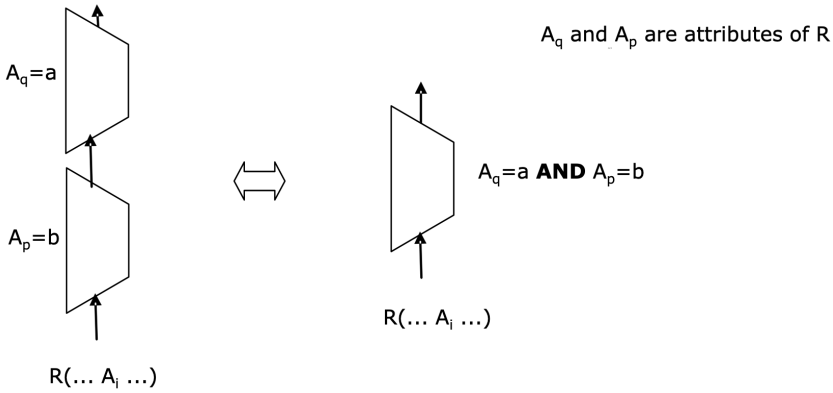
- Build the syntactic tree of the query
- Build the syntactic tree(s) of the view(s)
- Substitute the view definition(s) in the syntactic tree
  - They will always be at the leaves

Example of view expansion

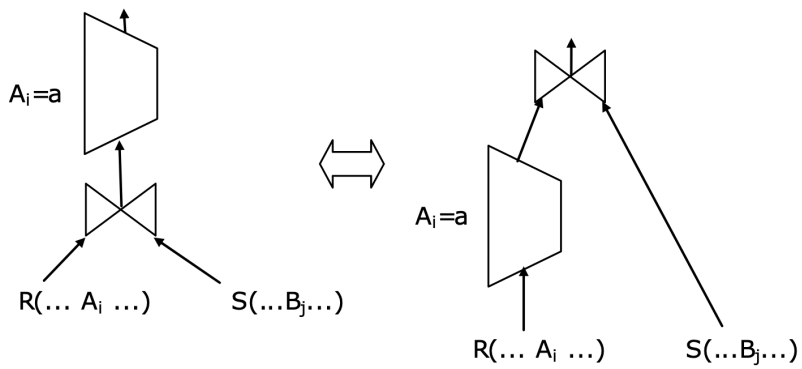


Equivalence rules

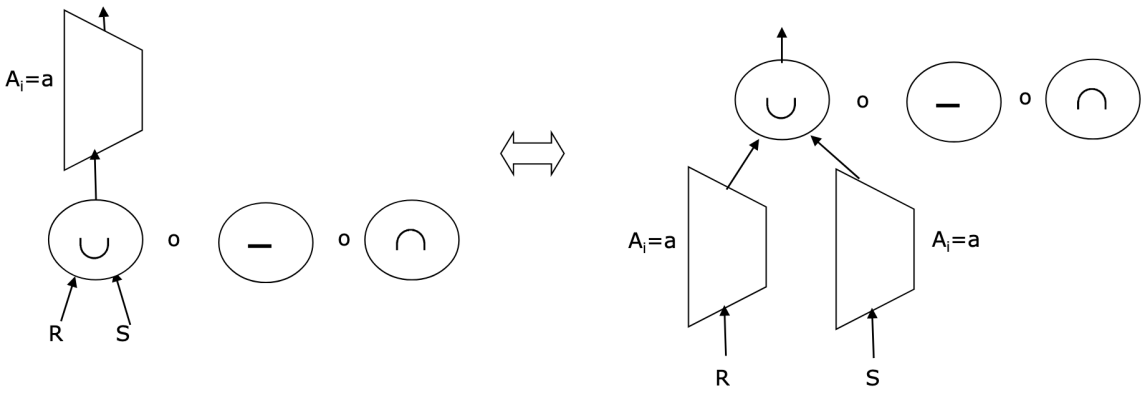
Splitting/grouping selections



Commutating the precedence of selection and join

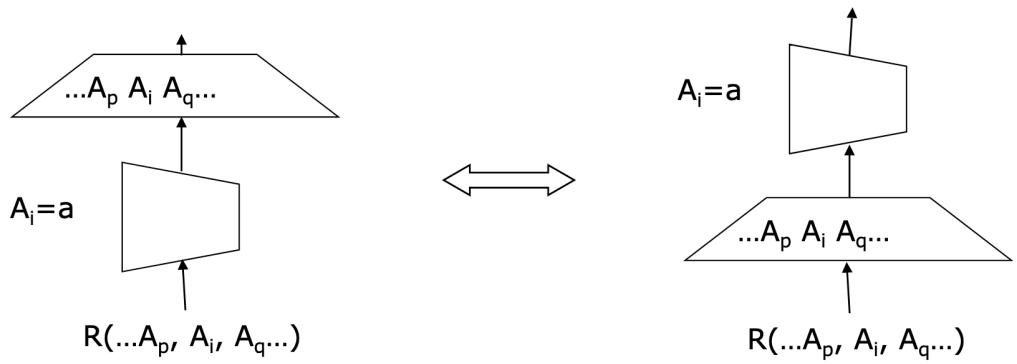


Commutating the precedence of selection and union



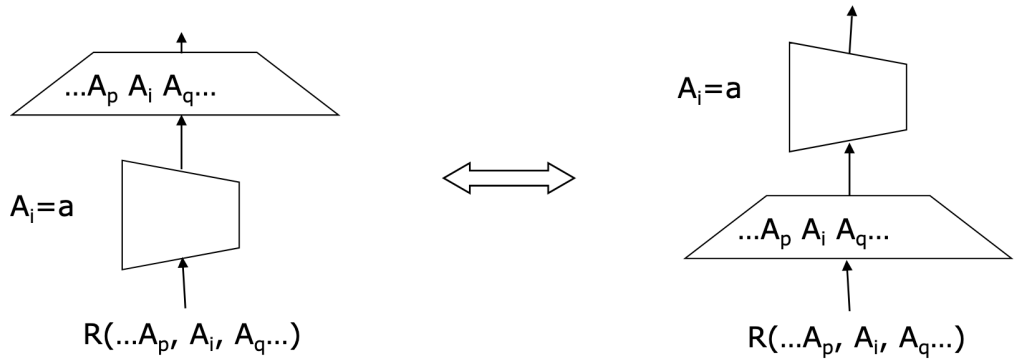
Commutating the precedence of selection and projection

$$(A_i \in \{..., A_p, A_i, A_q, ...\})$$



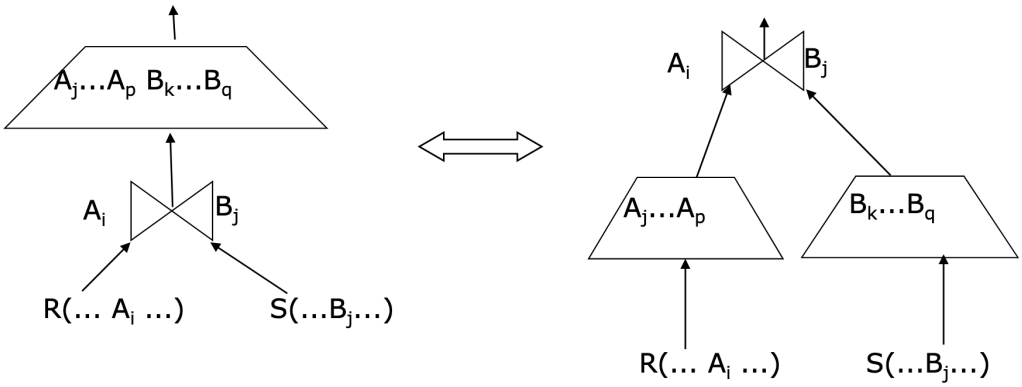
Commutating the precedence of selectionand projection

$$(A_i \notin \{..., A_p, A_q, ...\})$$



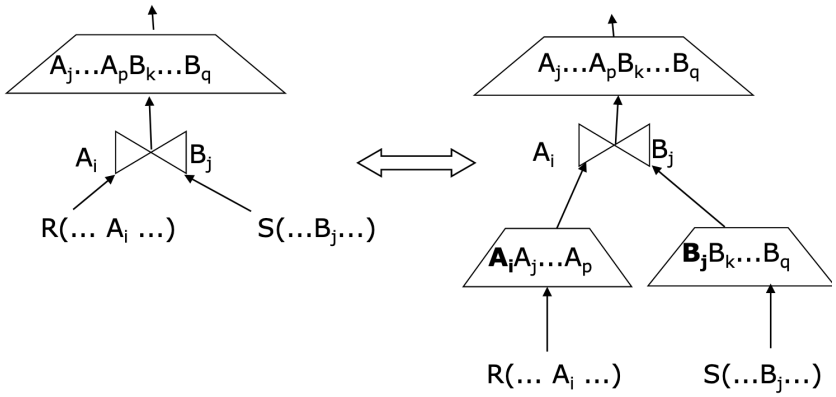
Commutating the precedence of projection and join

$$(A_i and B_j \in \{A_j, ..., A_p, B_k, ..., B_q\})$$

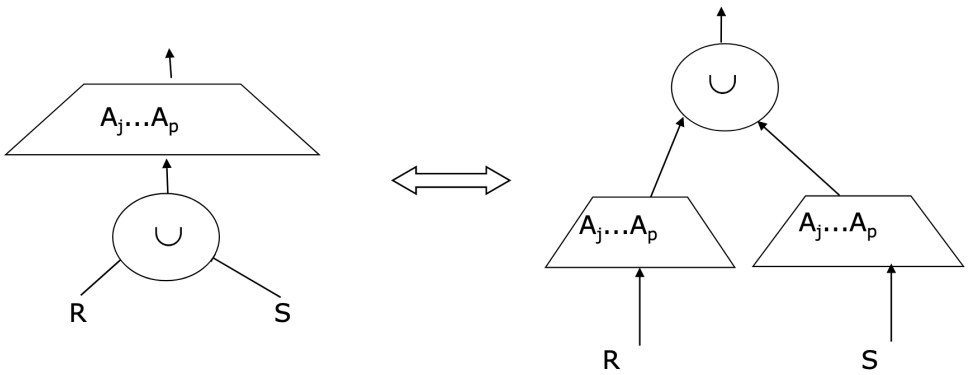


Commutating th eprecedenceof projection and join

$$(A_i or B_j or Both \notin \{A_j, ..., A_p, B_k, ..., Bq\})$$



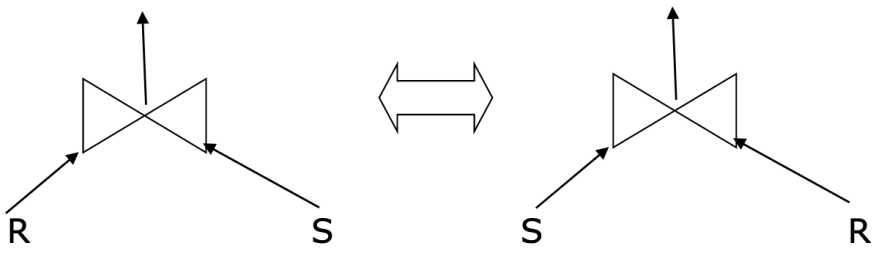
Commutating the precedence of projectionand union



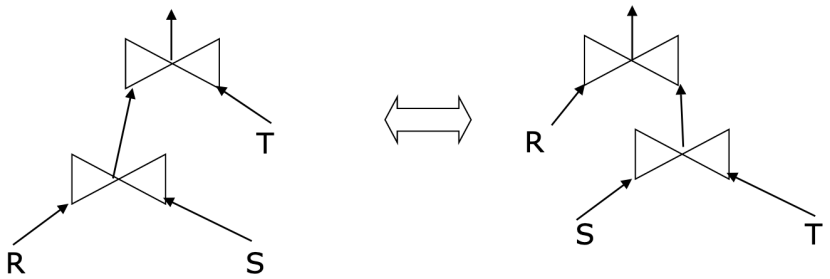
Important:

- Projection and difference precedence cannot be commutated
- Projection and intersection precedence cannot be commutated

Commutating join branches



Associating join tables



Transforming the syntactic tree

- Objective:
  - Reduce the size of intermediate nodes
- Steps:
  1. Split the selection predicates into simple clauses
  2. Lower selections as much as possible
  3. Group consecutive selections (simplify them if possible)
  4. Lower projections as much as possible (do not leave them just on a table)
  5. Group consecutive projections (simplify them if possible)

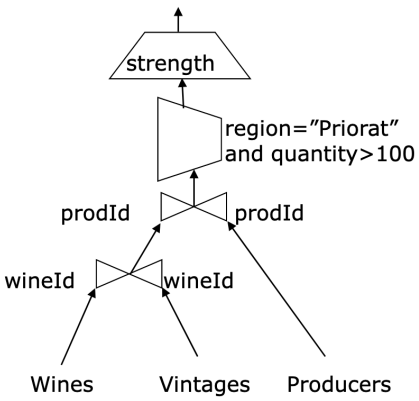
Example of syntactic optimization

Wines(wineId, wineName, strength)

Vintages(wineId, prodId, quantity)

Producers(prodId, prodName, region)

```
SELECT DISTINCT w.strength
FROM wines w, producers p, vintages v
WHERE v.wineId=w.wineId
        AND p.prodId=v.prodId
        AND p.region="Priorat"
        AND v.quantity>100;
```

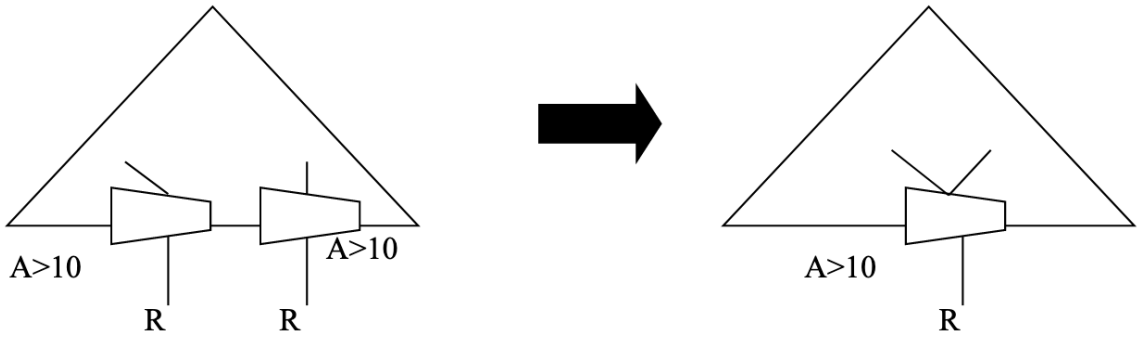


Try to do it.

Simplification of the syntactic tree

- Removal of disconnected components

- Fusion of common branches



- Removal of tautologies
  - $R \cap \varnothing = \varnothing$
  - $R - R = \varnothing$
  - $\varnothing - R = \varnothing$
  - $R \cap R = R$
  - $R \cup R = R$
  - $R \cup \varnothing = R$
  - $R - \varnothing = R$