Query Optimization Phases

Knowledge objectives

1. Enumerate the three phases of query optimization
2. Define semantic query optimization
3. Define syntactic query optimization
4. Exemplify the benefits of syntactic query optimization
5. Explain two syntactic optimization heuristics
6. Define physical query optimization
7. Enumerate four sources of complexity in physical query optimization

Understanding objectives

1. Perform simple semantic optimizations over a query
2. Optimize a syntactic tree considering the heuristics about the order of operations
3. Transform a syntactic tree into a process tree

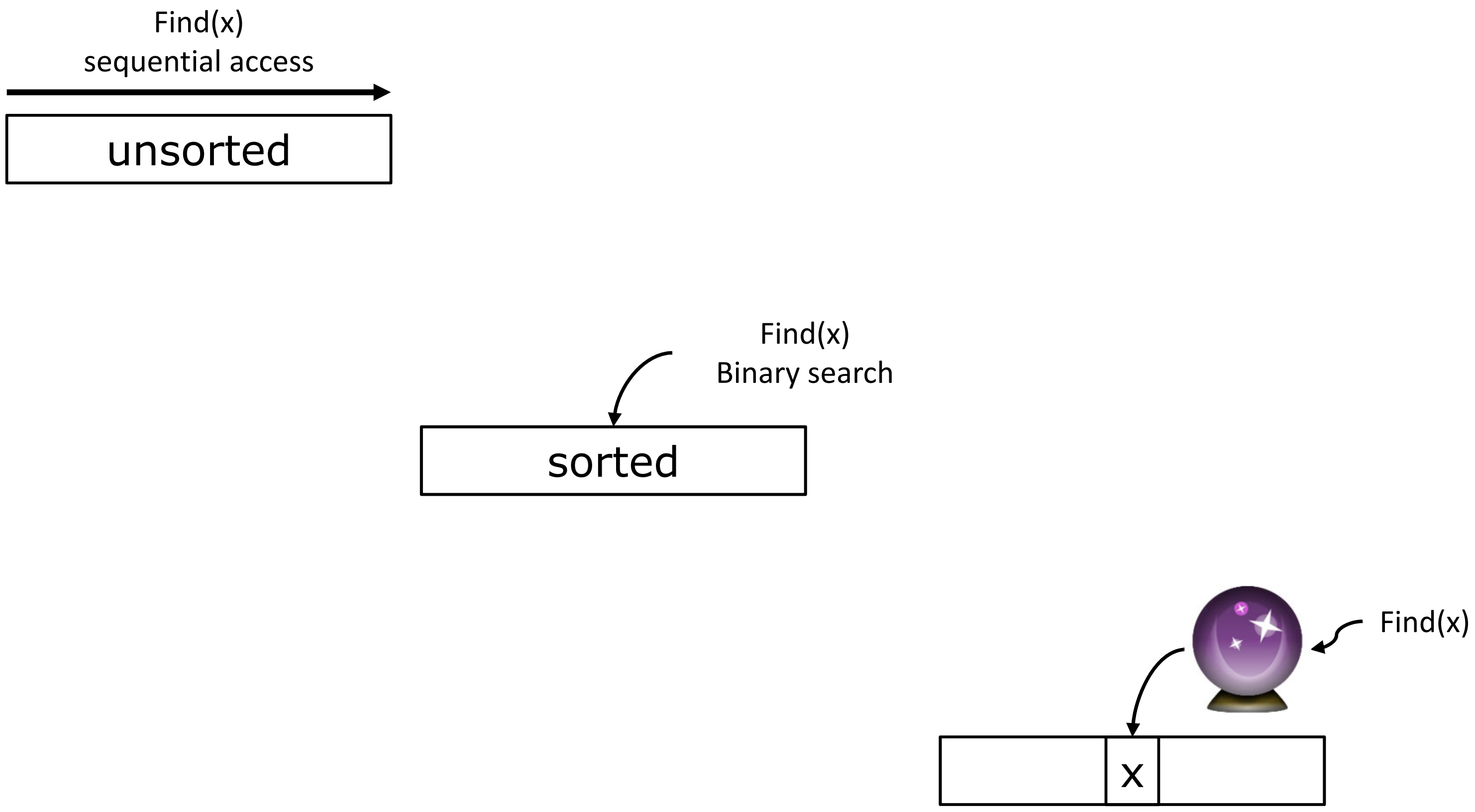
Optimization phases

Semantic

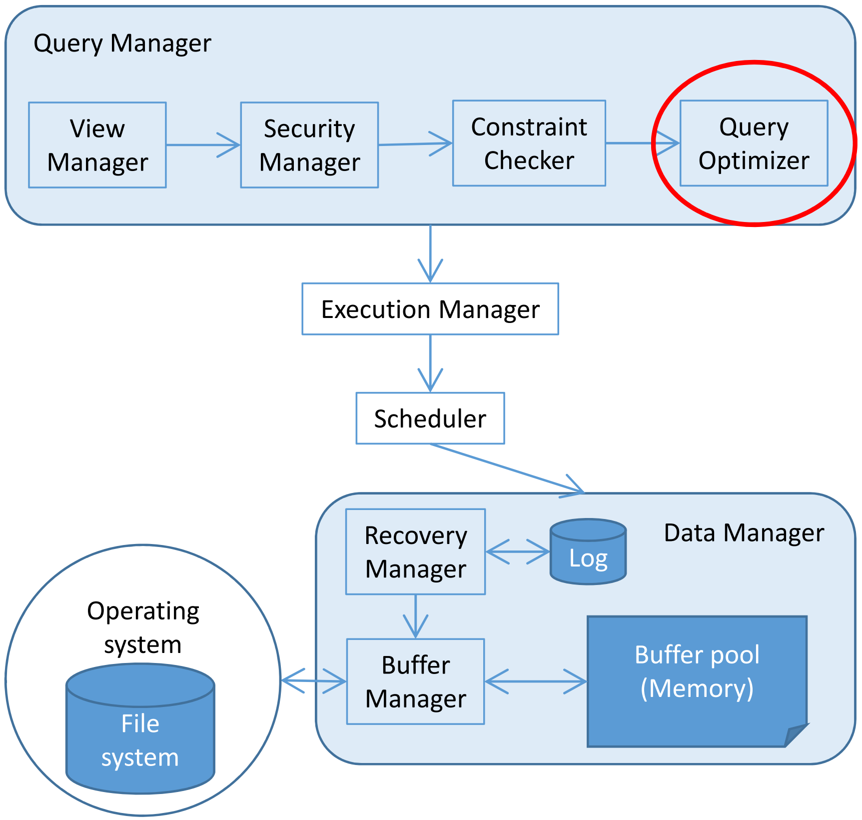
Syntactic

Physical

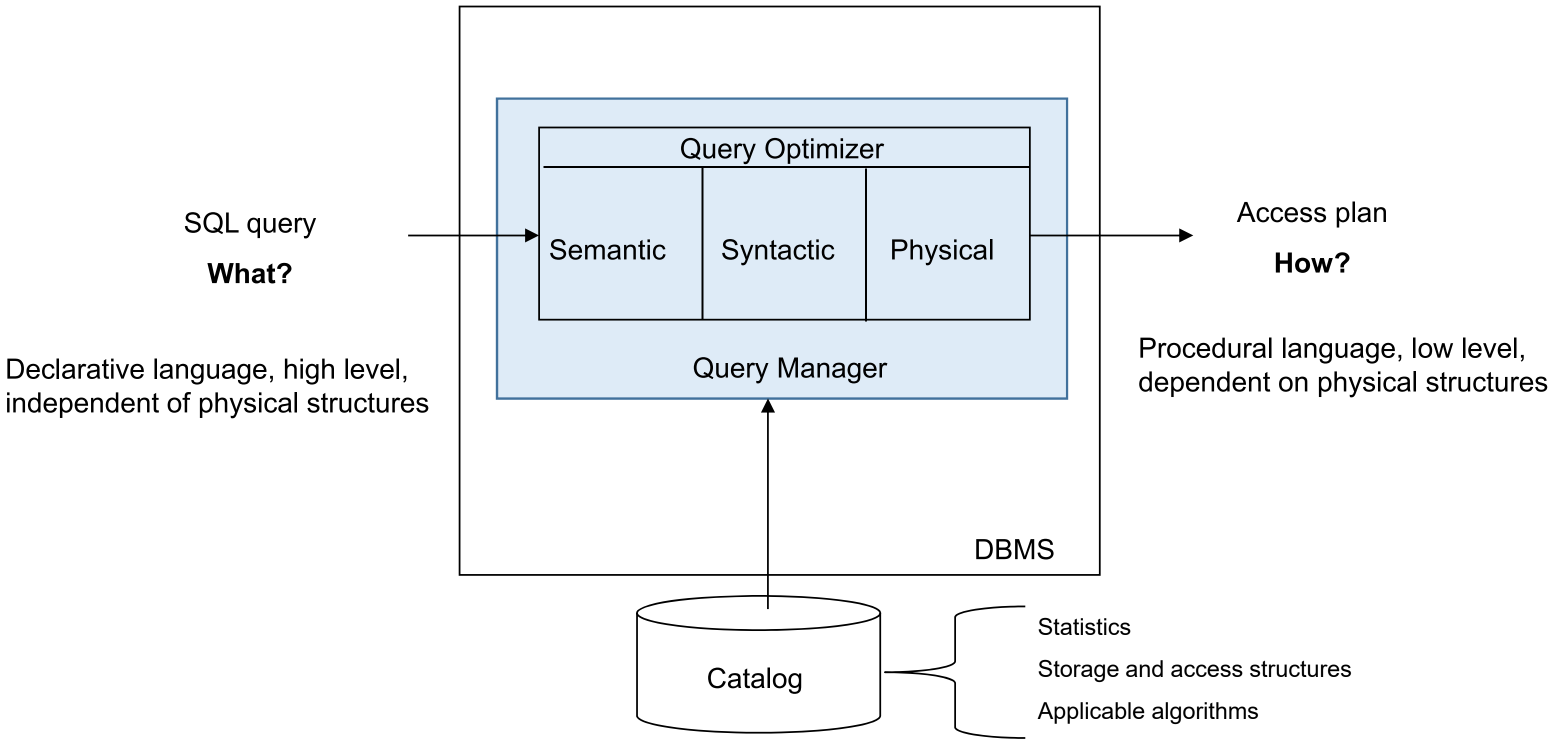
Cost efficiency



DBMS Functional Architecture (overall view)



DBMS Functional Architecture (zoom in)



Considerations

- Input: An SQL query over tables (or views), syntactically correct and authorized
  - Optimization is the last step in query processing
- Output: An algorithm (access plan) that must be followed by the execution manager in order to get the result
- Goal: Minimize the use of resources
  - In general, a DBMS does **not** find the optimal access plan
    - It obtains an approximation (in a reasonable time)

Semantic optimization

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 (I)

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

SELECT * FROM students WHERE mark<2;	→	SELECT * FROM students WHERE false;
SELECT * FROM students WHERE mark<6 AND mark>8;	→	SELECT * FROM students WHERE false;
SELECT * FROM students WHERE mark<6 AND mark<7;	→	SELECT * FROM students WHERE mark<6;

Examples of semantic optimization (II)

```
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>5
        AND e.dpt>5;
```

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Examples of semantic optimization (III)

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

⇩

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

Syntactic optimization

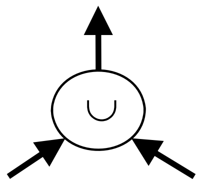
Syntactic optimization

Consists of translating the sentence from SQL into a sequence of algebraic operations in the form of syntactic tree over tables (i.e., performs view expansion) with minimum cost, by means of heuristics (there is more than one solution)

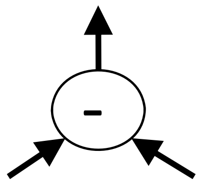
- Nodes
  - Internal: Operations
  - Leaves: Tables
  - Root: Result
- Edges
  - Denote direct usage

Graphical representation of the syntactic tree

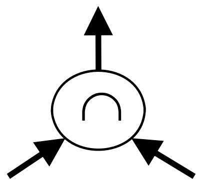
Union



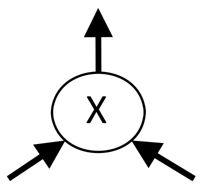
Difference



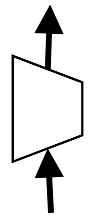
Intersection



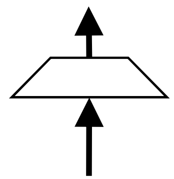
Cross product



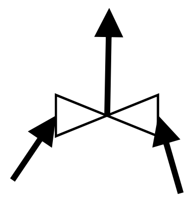
Selection



Projection



Join

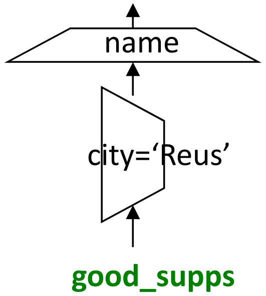


Expand views

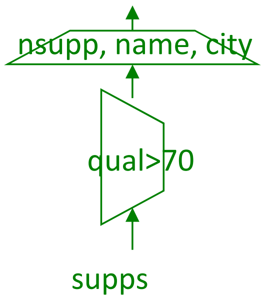
1. Build the syntactic tree of the query
2. While there are views in the tree
    1. Build the syntactic tree(s) of the view(s)
    2. Replace the view definition(s) in the syntactic tree
        - They will always be at the leaves

Example of view expansion

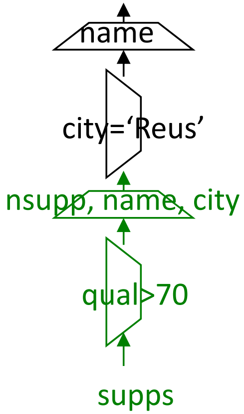
Query over a view



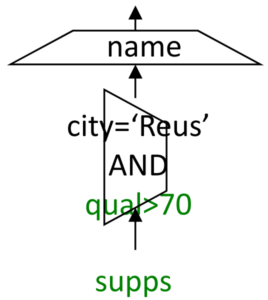
View definition



Equivalent query over the source table



Simplified query



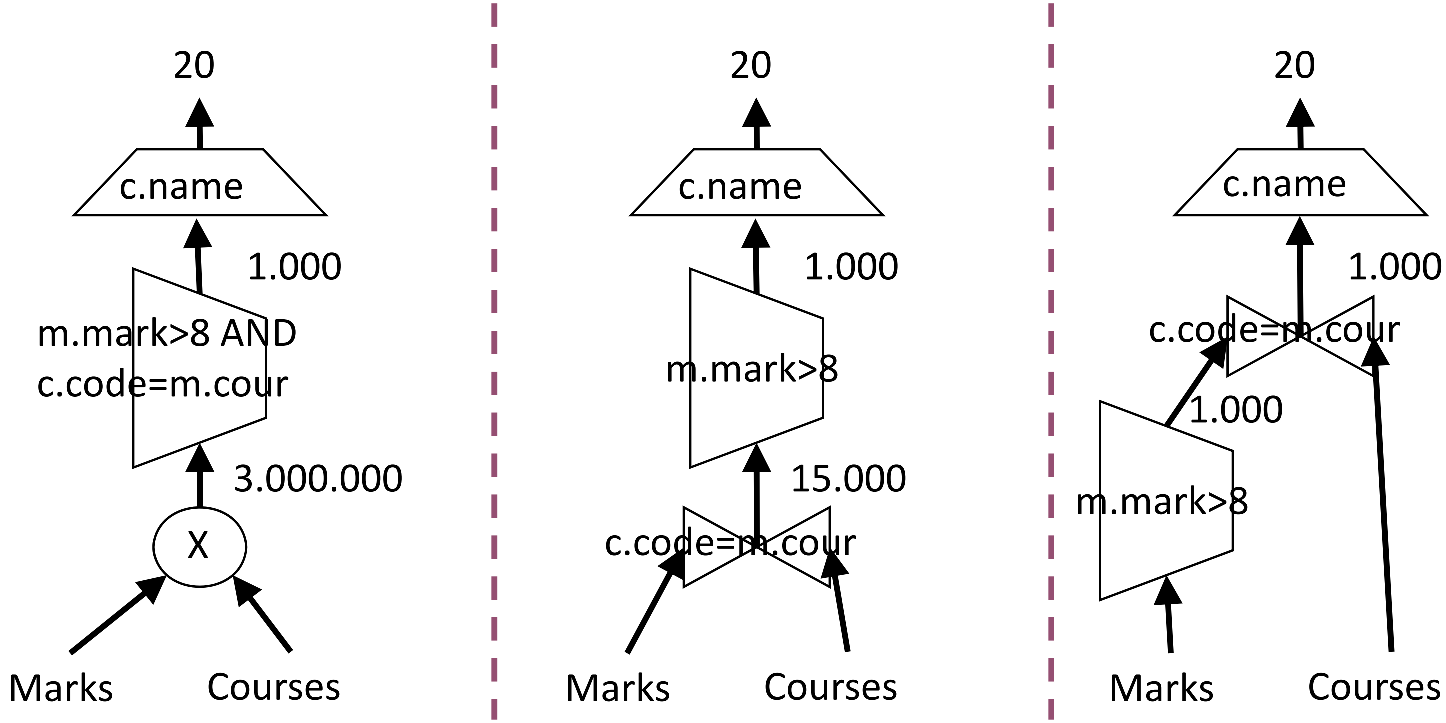
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Example of syntactic optimization

Courses (<u>code</u>, name, …)
Marks (<u>cour, stu</u>, mark)
Students (<u>id</u>, …)

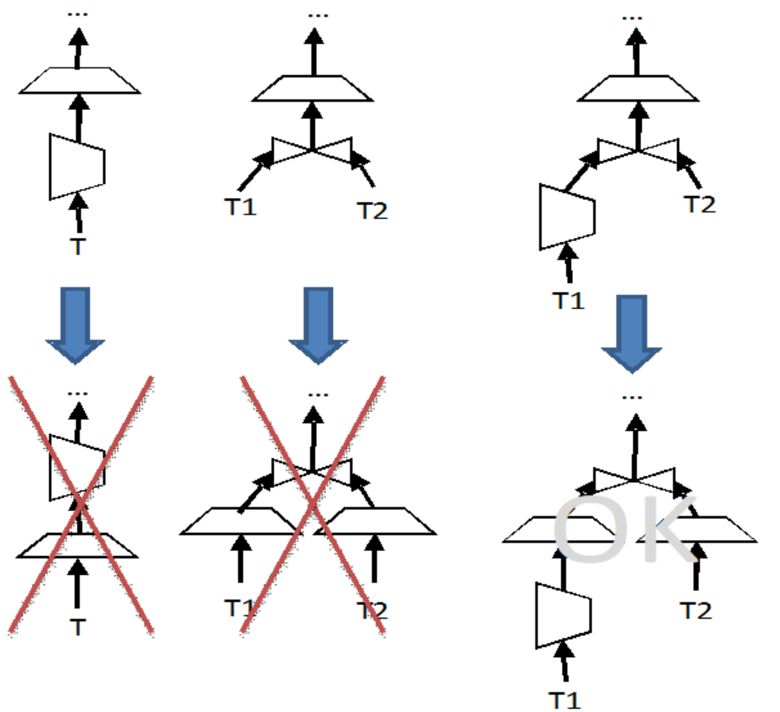
|Courses| = 200
|Marks| = 15000
|Students| = 3000

```
SELECT DISTINCT c.name
FROM courses c, marks m
WHERE c.code=m.cour
        AND m.mark>8;
```



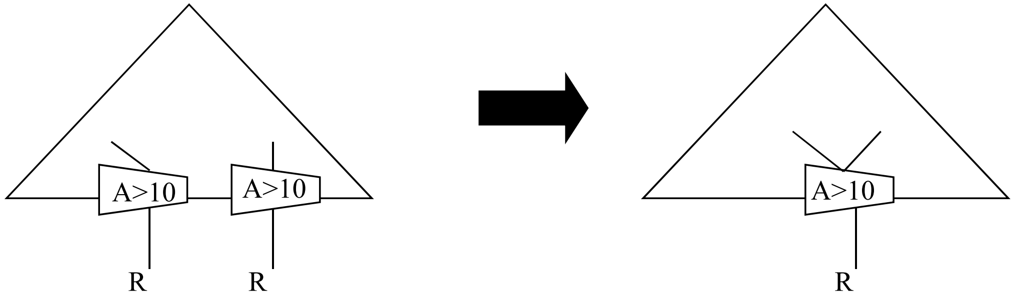
Transforming the syntactic tree

- 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 posible
     - Do not leave them directly on top of a table
       - Except when one branch leaves the projection just on top of the table and the other does not
  5. Group consecutive projections
     - Simplify them if possible



Simplification of the syntactic tree

- Fusion of common subtrees



- Removal of tautologies:

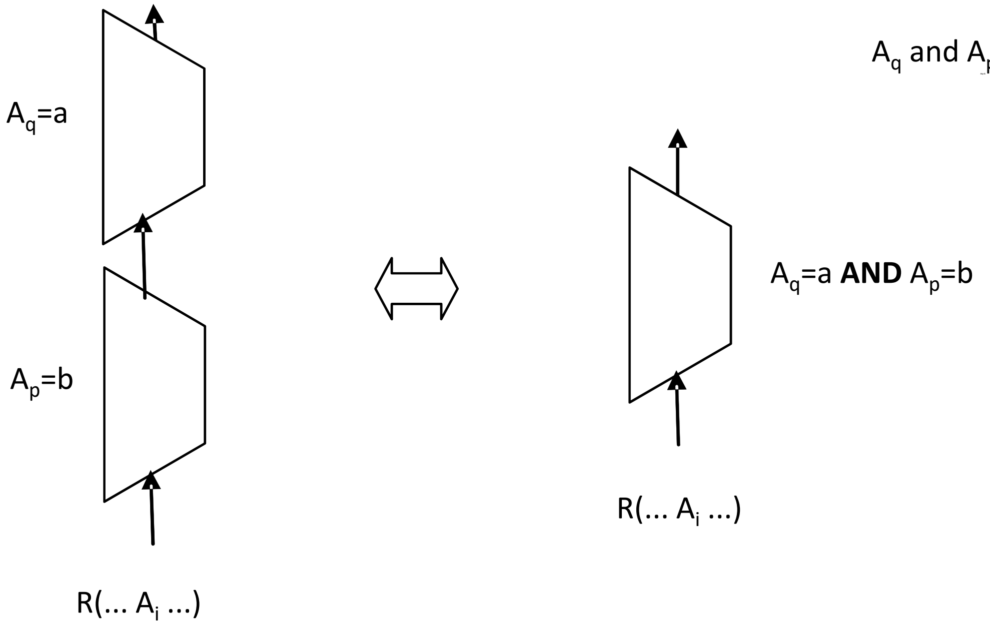
$$R \cap \varnothing = R - R = \varnothing - R = \varnothing$$

$$R \cap R = R \cup R = R \cup \varnothing = R - \varnothing = R$$

Equivalence rules

Equivalence rules (I)

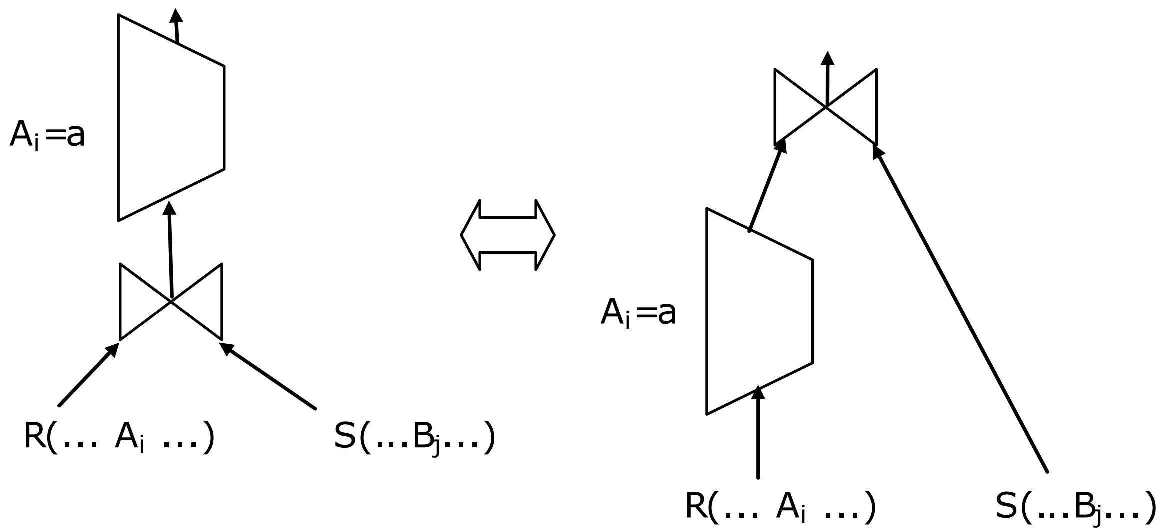
- Splitting/grouping selections



A_q and A_p are attributes of R

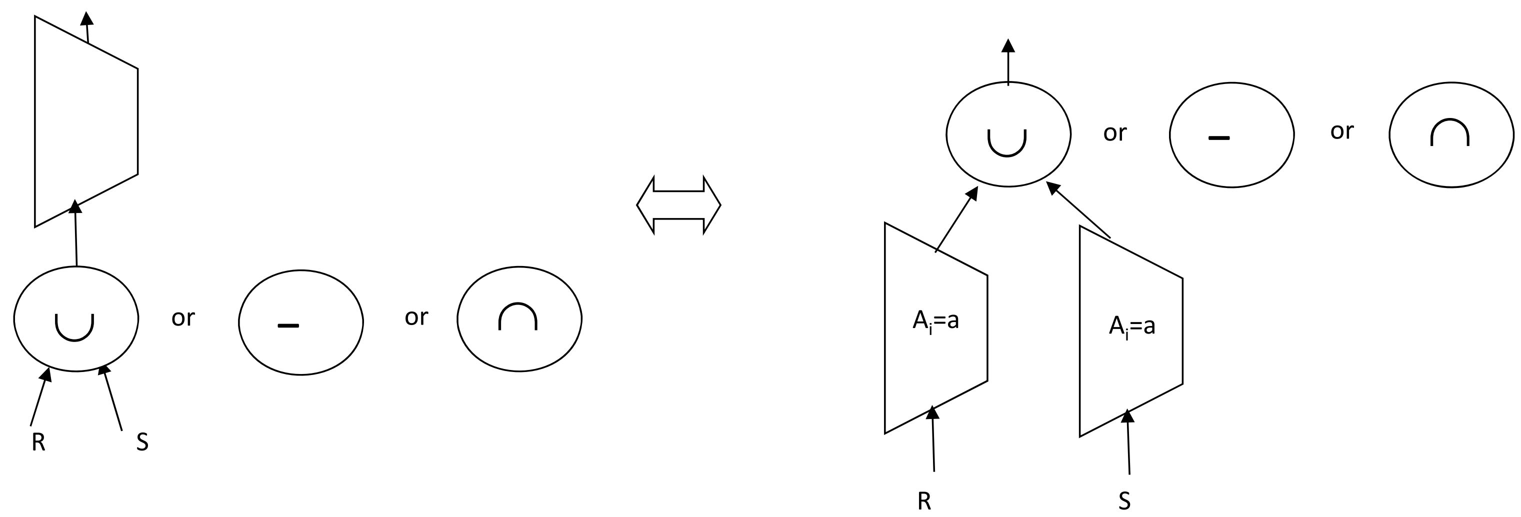
Equivalence rules (II)

- Commuting the precedence of selection and join



Equivalence rules (III)

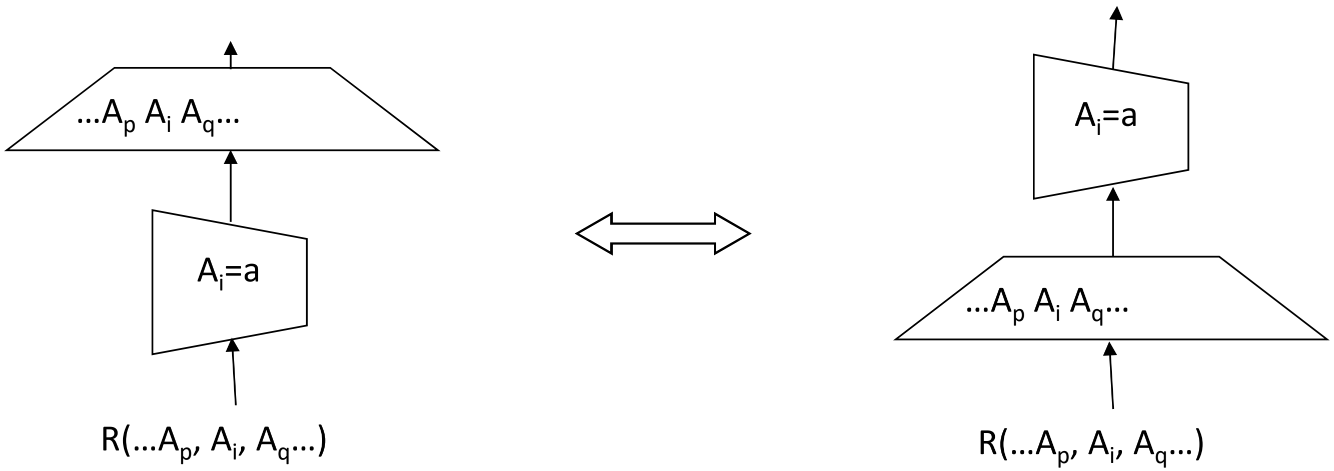
- Commuting the precedence of selection and set operations



Equivalence rules (IV)

- Commuting the precedence of selection and projection

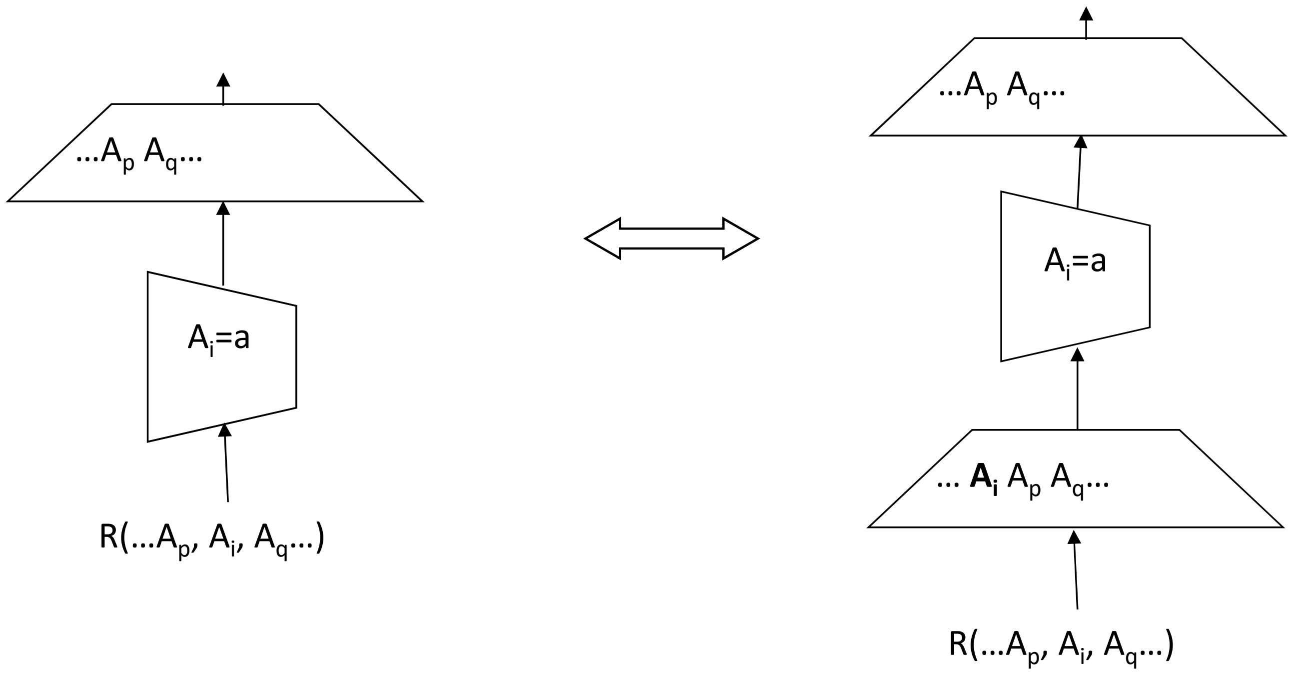
  When $A_i \in \{\ldots A_p, A_i, A_q \ldots\}$



Equivalence rules (V)

- Commuting the precedence of selection and projection

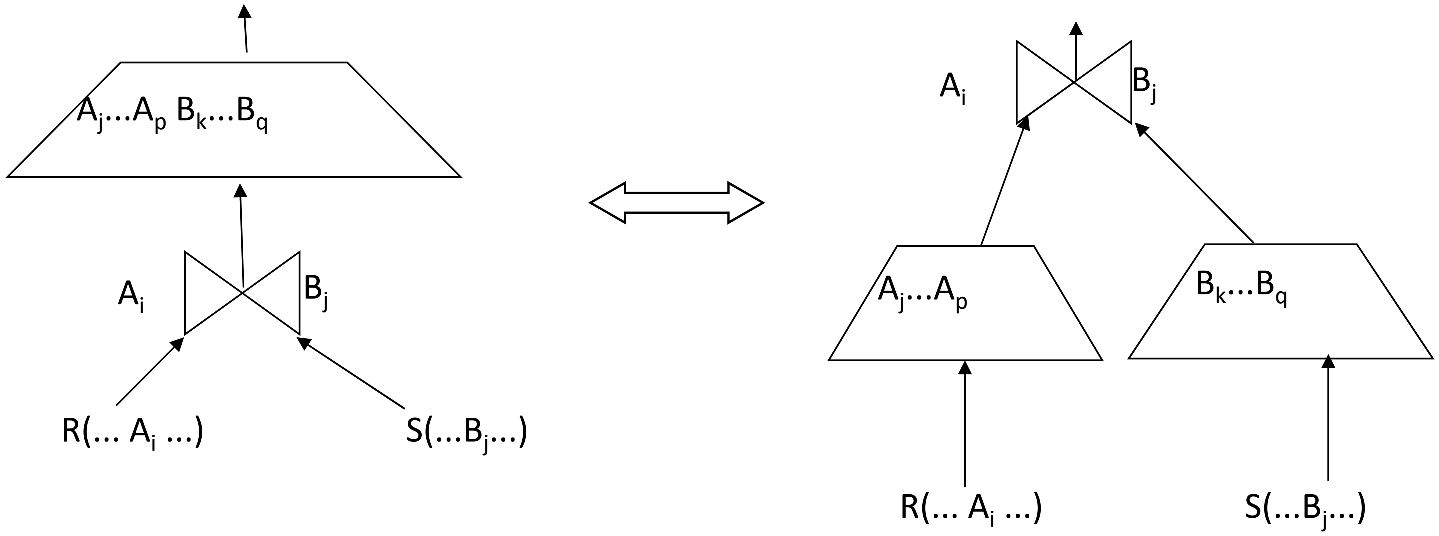
  When $A_i \notin \{\ldots A_p, A_q \ldots\}$



Equivalence rules (VI)

- Commuting the precedence of projection and join

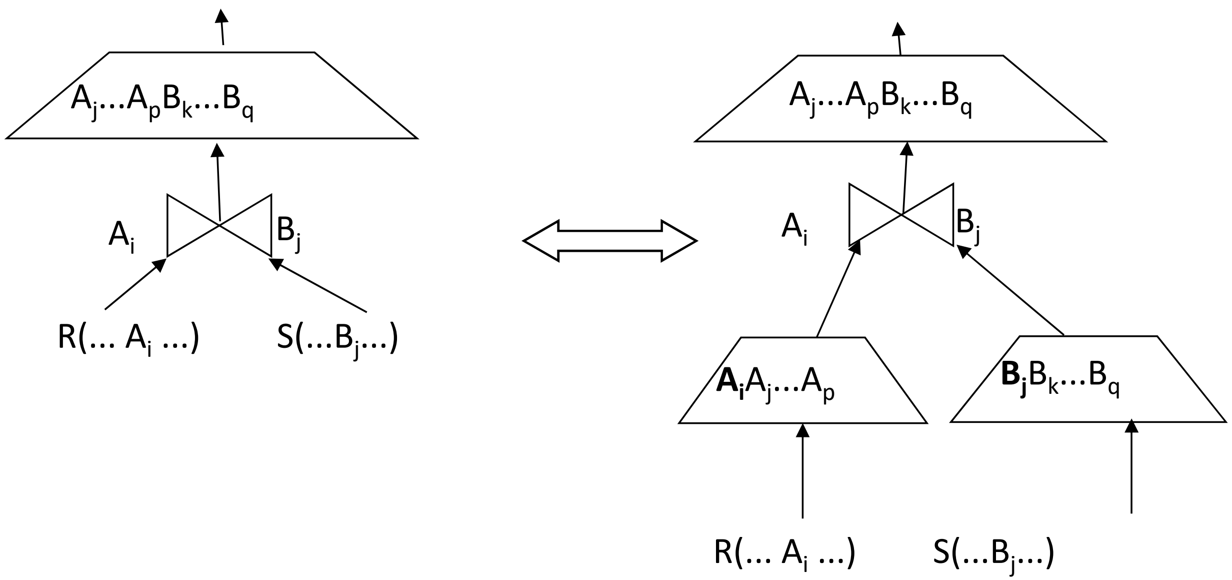
  When A_i , B_j $\in\{A_j \ldots A_p, B_k \ldots B_q\}$



Equivalence rules (VII)

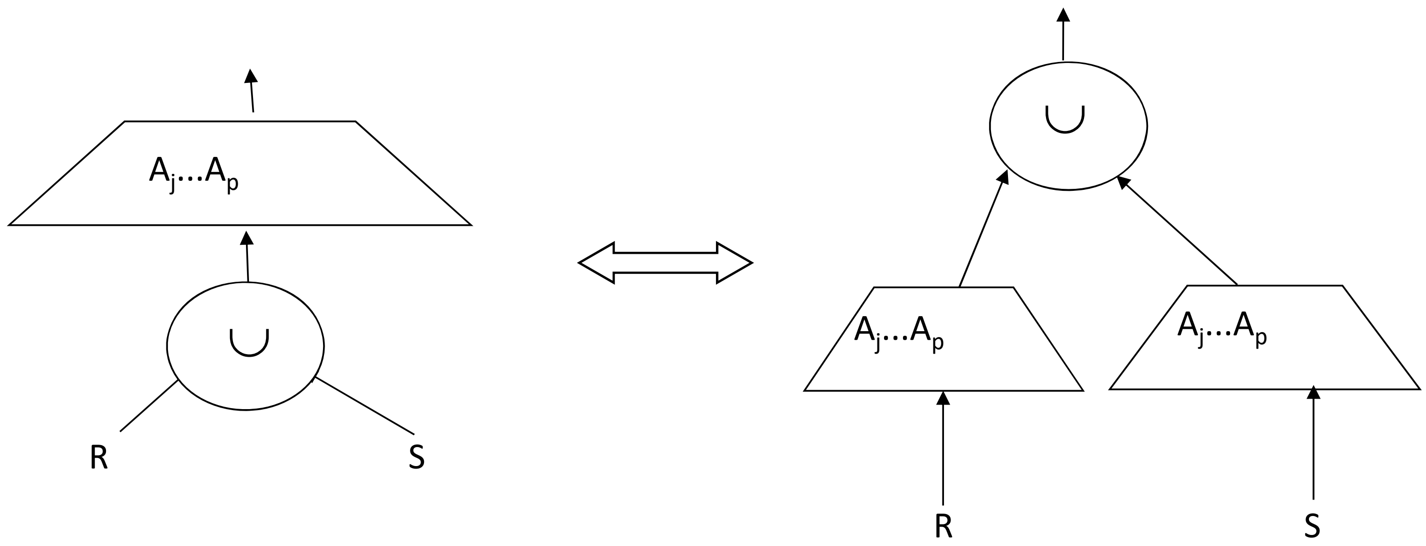
- Commuting the precedence of projection and join

  When A_i or B_j (or both) $\notin \{A_j \ldots A_p, B_k \ldots B_q\}$



Equivalence rules (VIII)

- Commuting the precedence of projection and union

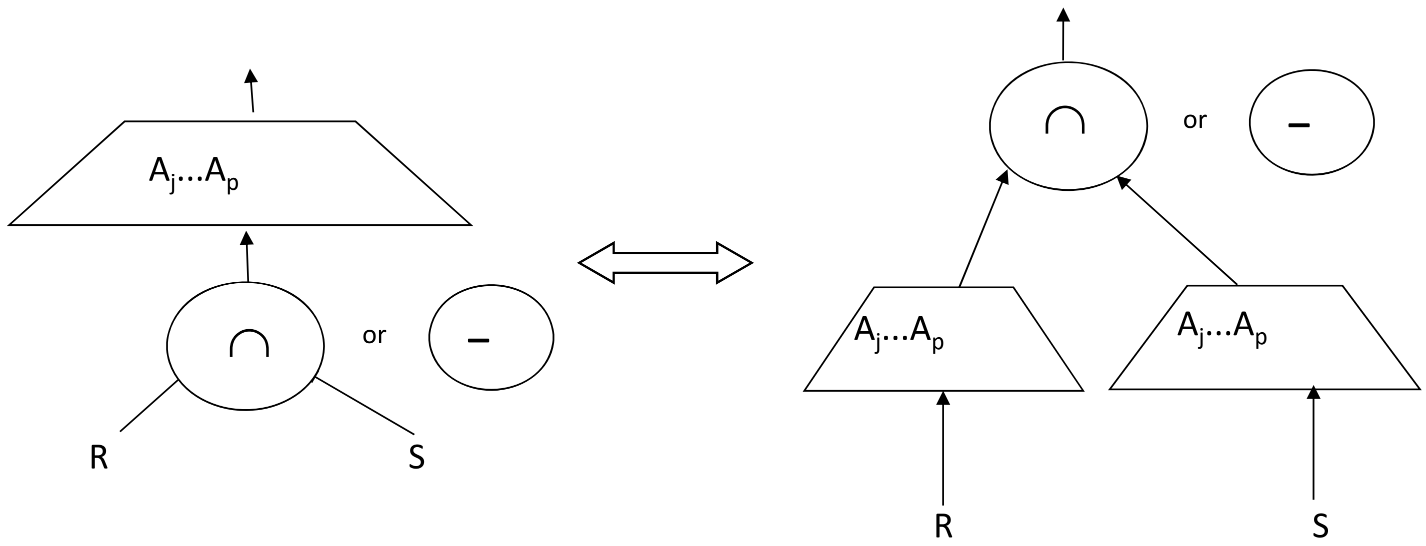


Important:

- Projection and intersection precedence cannot be freely commuted
- Projection and difference precedence cannot be freely commuted

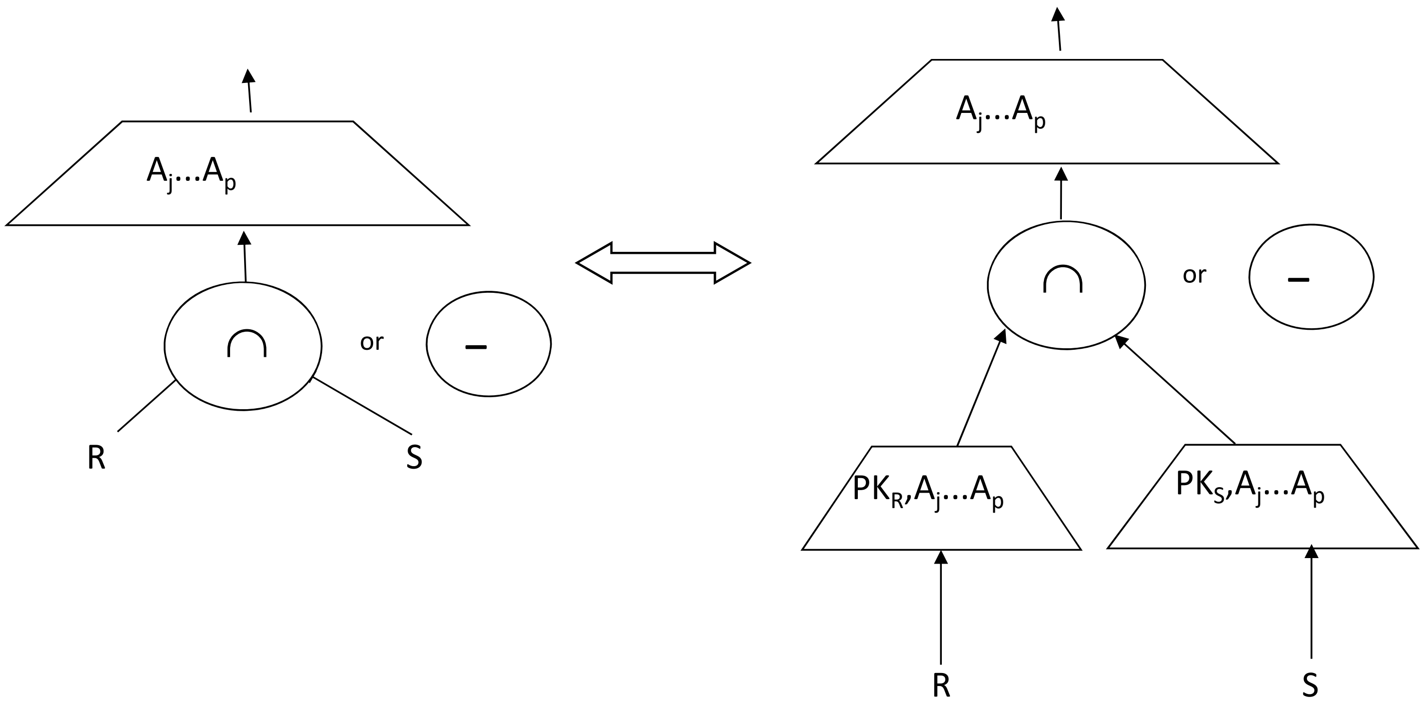
Equivalence rules (IX)

- Commuting the precedence of projection and intersection/difference
  When PK_R, $PK_S \in \{A_j, \ldots, A_p\}$



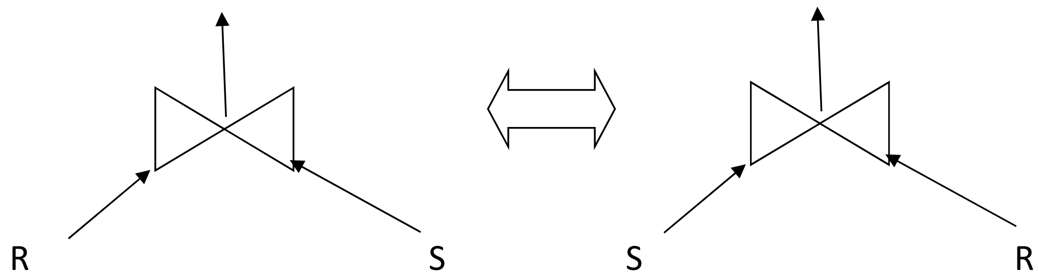
Equivalence rules (X)

- Commuting the precedence of projection & intersection/difference
  When PK_R, $PK_S \notin \{A_j,\ldots, A_p\}$



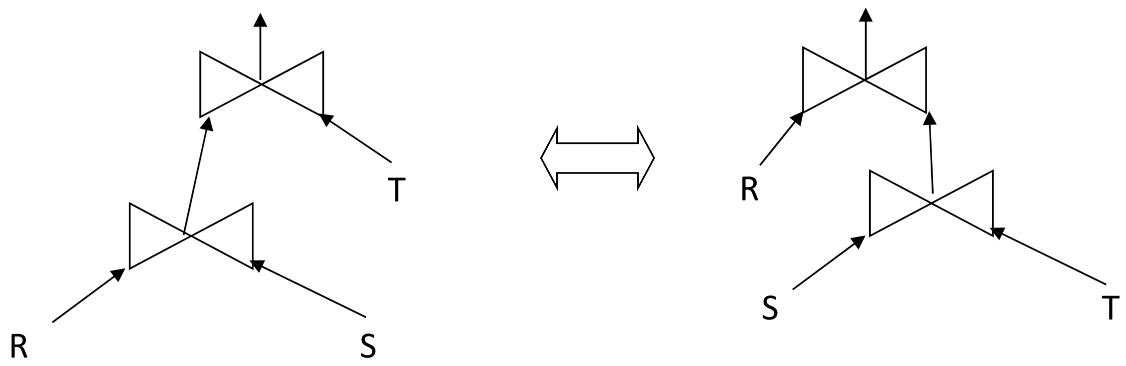
Equivalence rules (XI)

- Commuting join branches



Equivalence rules (XII)

- Associating joins



Example of syntactic optimization

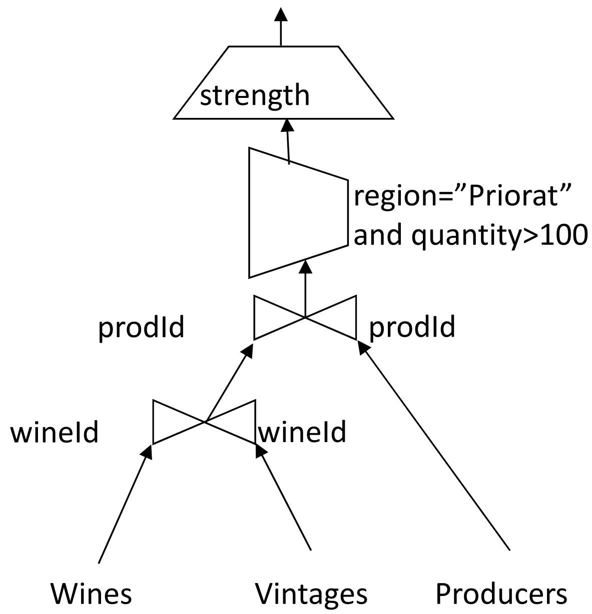
Example of syntactic optimization (I)

Wines(wineId, wineName, strength)

Vintages(wineId, prodId, quantity)

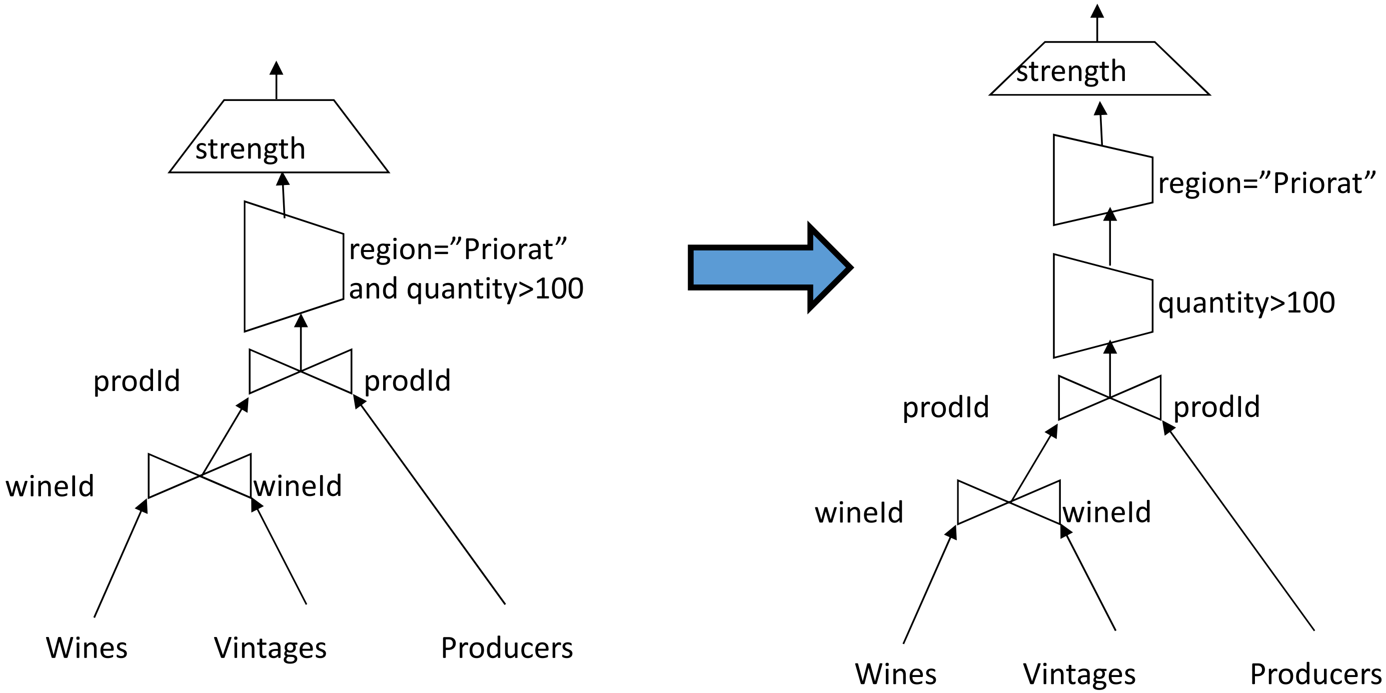
Producers(prodId, prodName, region)

```
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;
```



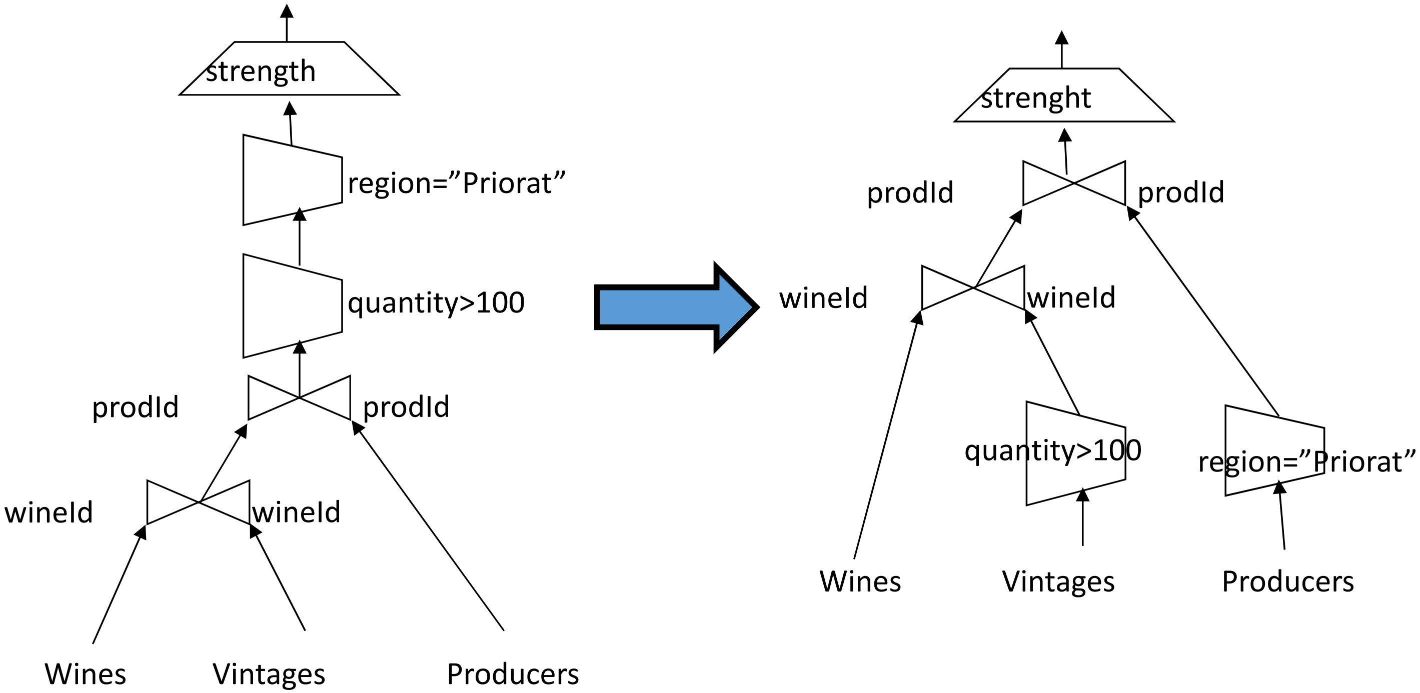
Example of syntactic optimization (II)

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, except when one branch leaves the projection on the table and the other does not)
5. Group consecutive projections (simplify them if possible)



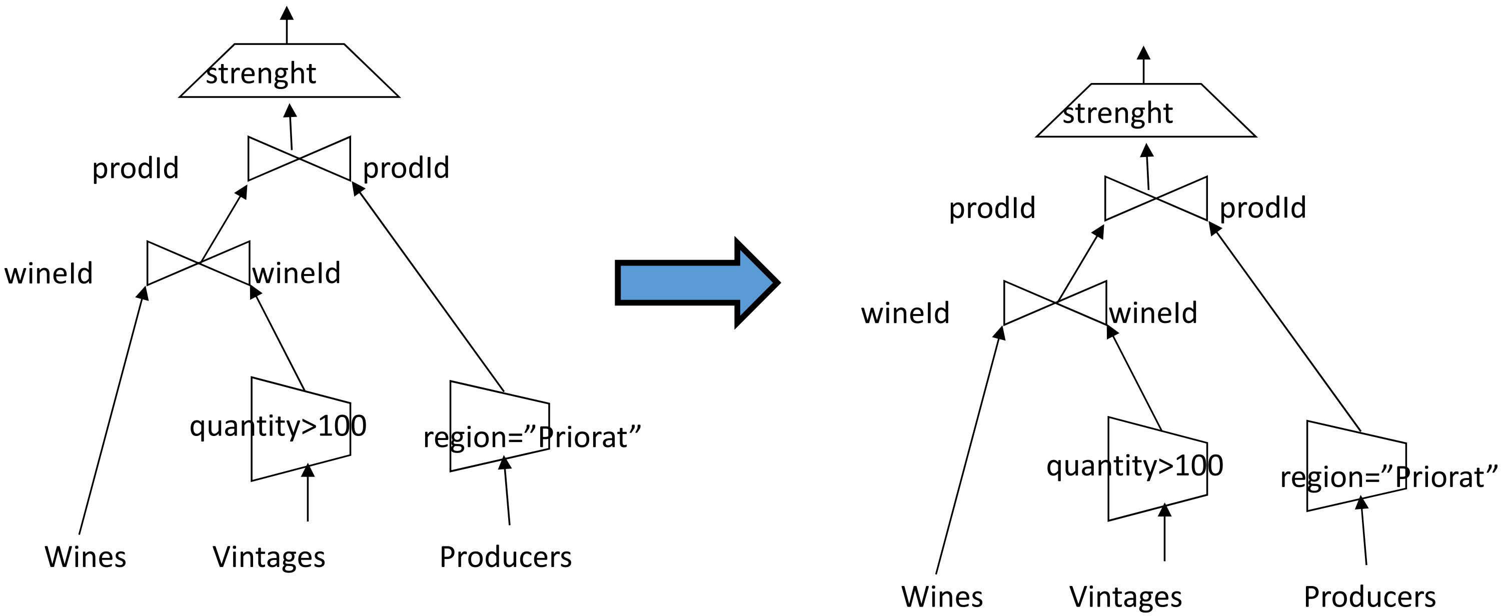
Example of syntactic optimization (III)

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, except when one branch leaves the projection on the table and the other does not)
5. Group consecutive projections (simplify them if possible)



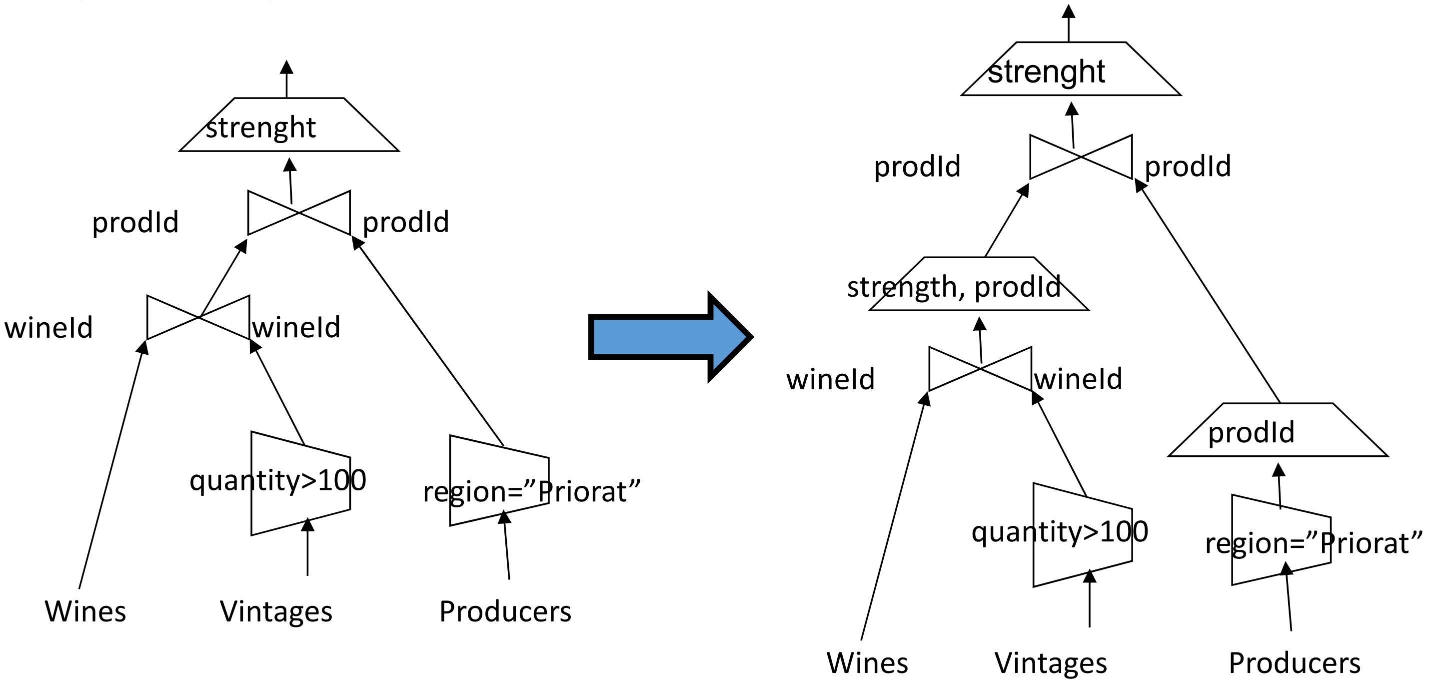
Example of syntactic optimization (IV)

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, except when one branch leaves the projection on the table and the other does not)
5. Group consecutive projections (simplify them if possible)



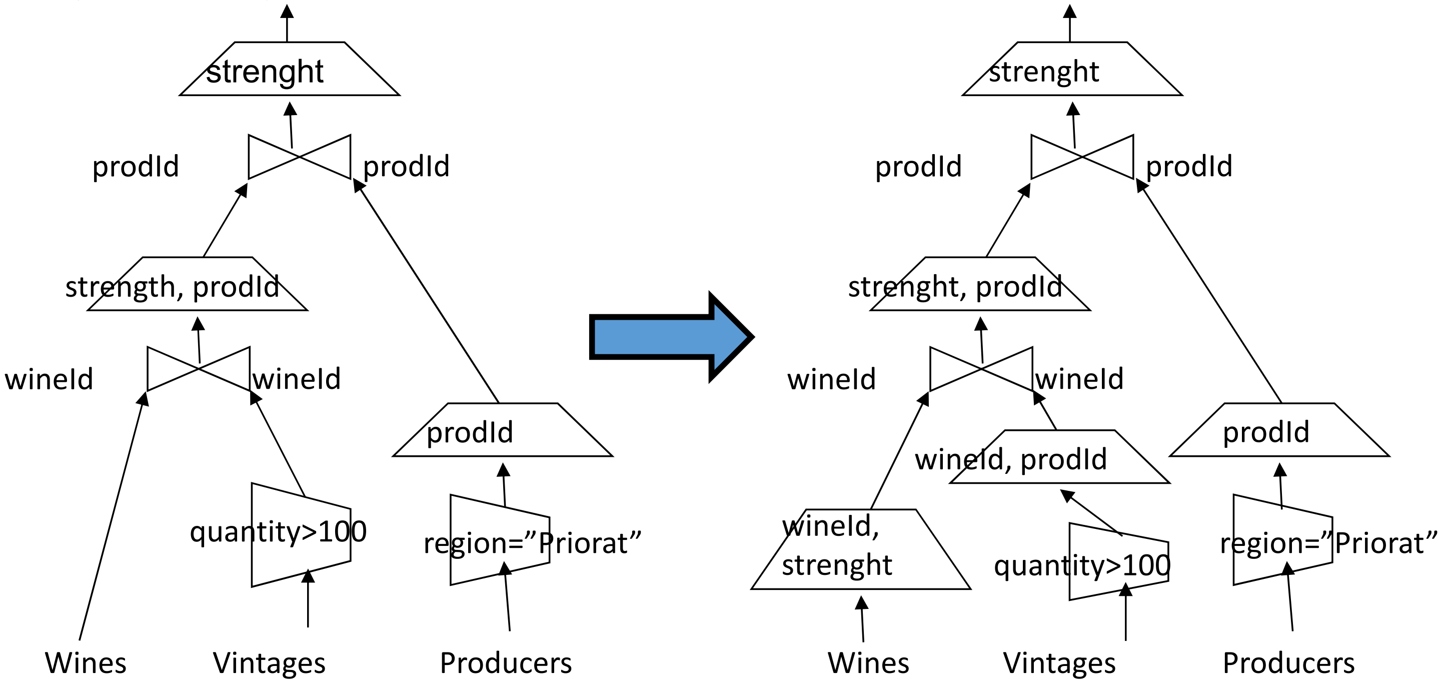
Example of syntactic optimization (V)

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, except when one branch leaves the projection on the table and the other does not)
5. Group consecutive projections (simplify them if possible)



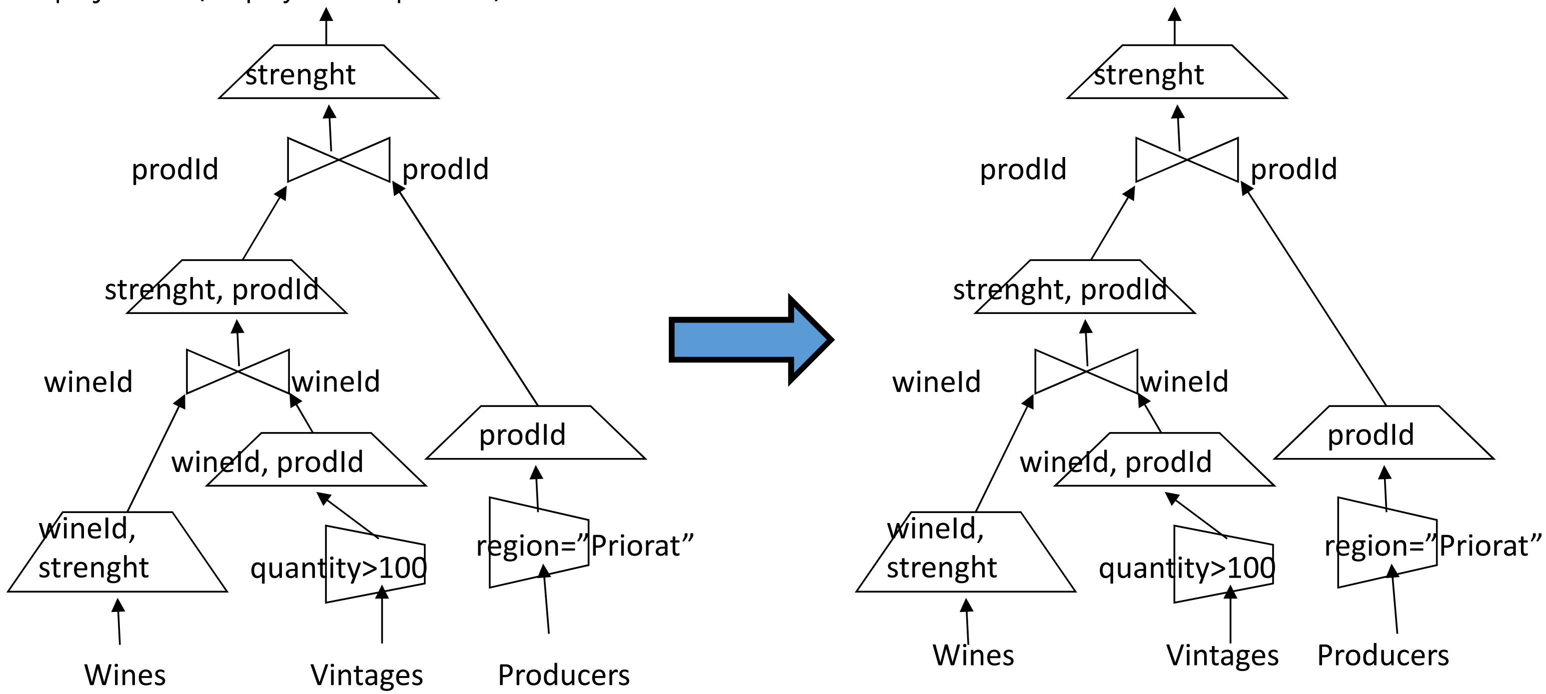
Example of syntactic optimization (VI)

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, except when one branch leaves the projection on the table and the other does not)**
5. Group consecutive projections (simplify them if possible)



Example of syntactic optimization (VII)

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, except when one branch leaves the projection on the table and the other does not)
5. Group consecutive projections (simplify them if possible)



UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Physical optimization

Physical optimization

Consists of generating the execution plan of a query (from the best syntactic tree) considering:

- Physical structures
- Access paths
- Algorithms

Process tree

This is the tree associated to the syntactic tree that models the execution strategy

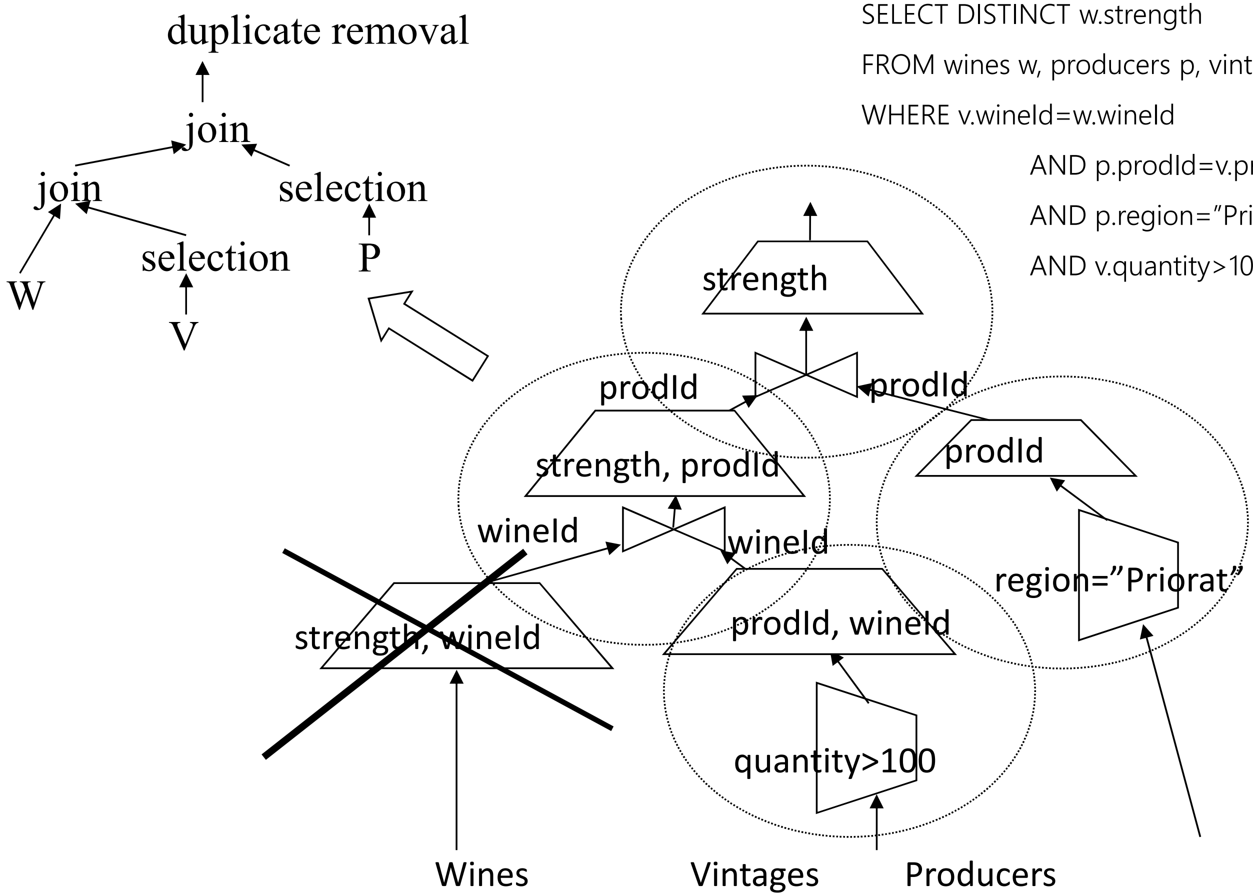
- Nodes
  - Leaves: Tables (or Indexes)
  - Internal: Intermediate tables generated by a physical operation
  - Root: Result
- Edges
  - Denote direct usage

Physical operations

- Related to relational algebra
  - Physical selection: Selection [+ projection]
  - Physical join: Join [+ projection]
  - Set operations:
    - Union [+ projection]
    - Difference [+ projection]
- Other operations:
  - Duplicate removal
  - Sorting
  - Grouping and calculating aggregates

Example of process tree

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;

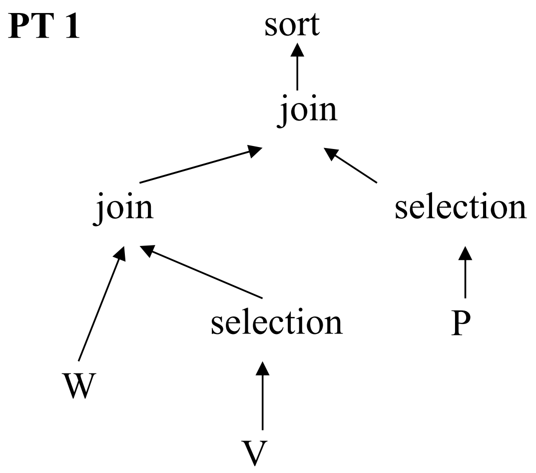


Cost-based optimization steps

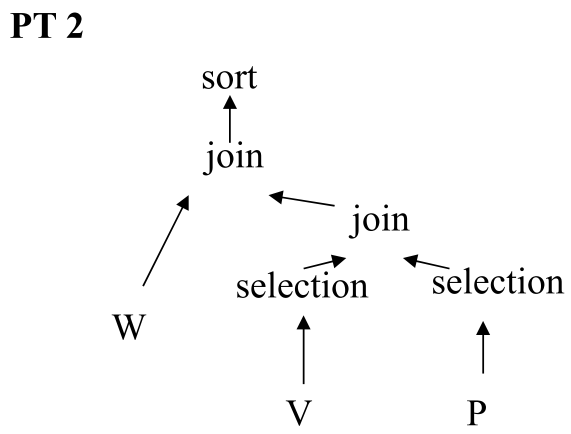
1. Generate alternatives in the search space
   a. Join order
   b. Potential algorithms
   c. Available structures (access path)
   d. ~~Materialization or not of intermediate results~~
      - We will assume that they are always materialized
2. Evaluate those alternatives
   1. Intermediate results cardinality and size estimation
   2. Cost estimation
3. Choose the best option
4. Generate the corresponding access plan

Join order

- We can generate different process trees by using the associative property of joins (Rule XII)



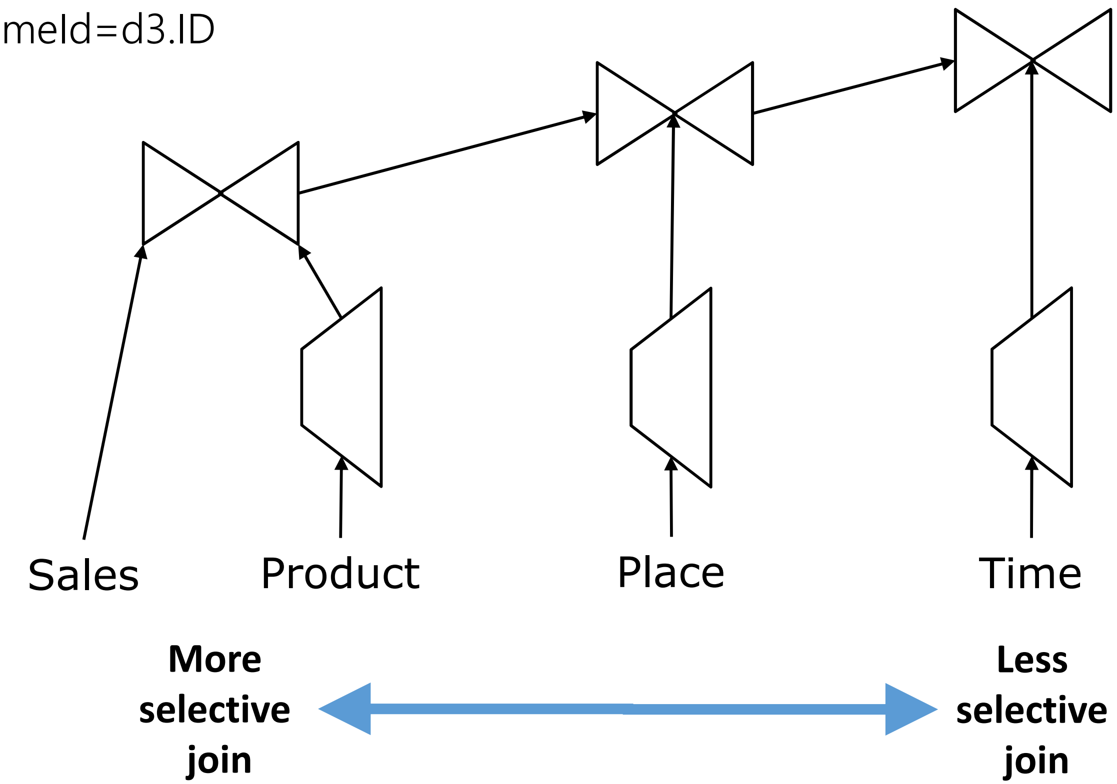
(W*V)*P



W*(V*P)

Join order in pipelining

SELECT d1.articleName, d2.region, d3.month, SUM(f.articles)
FROM Sales f, Product d1, Place d2, Time d3
WHERE f.productId=d1.ID AND f.placeId=d2.ID AND f.timeId=d3.ID
    AND d1.articleName IN ('Ballpoint','Rubber')
    AND d2.region='Catalunya'
    AND d3.month IN ('January02','February02')
GROUP BY ...



Cost estimation

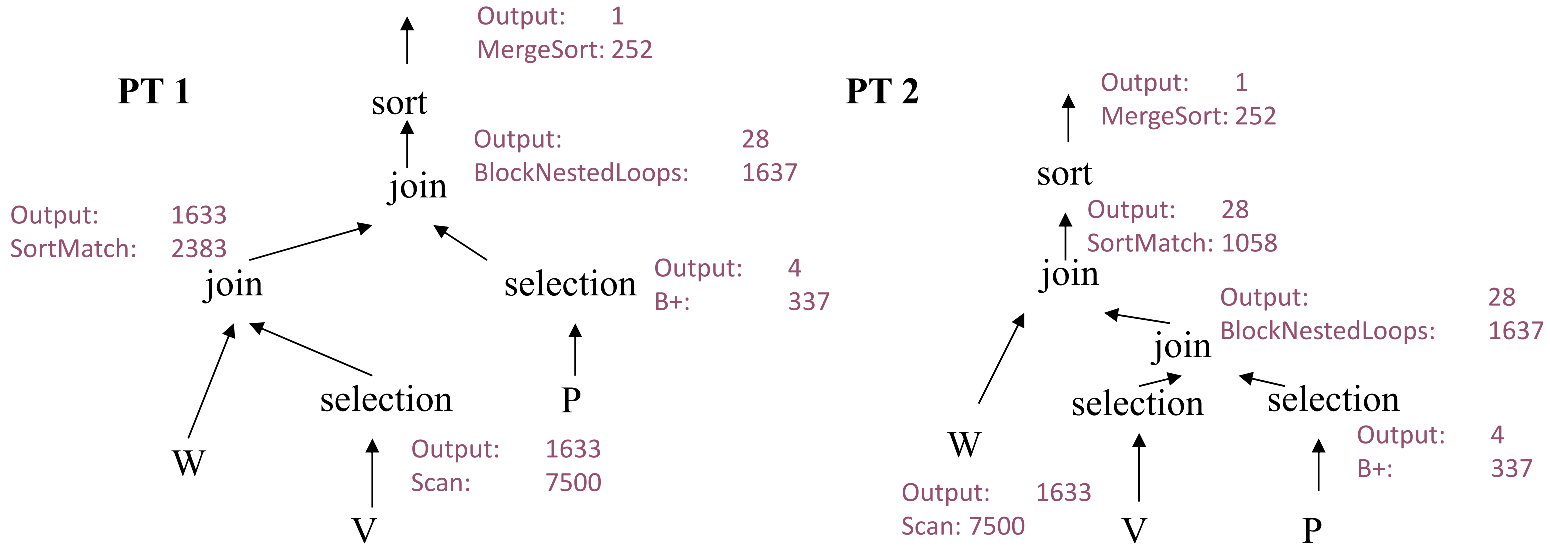
- The cost of the process tree is the sum of costs of each physical operation

- The cost of each operation is the sum of
  - Cost of solving it
  - Cost of writing its result

- Cost factors:
  - CPU
  - Memory access time
  - <u>Disk access time</u>

Example of cost estimation

```
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;
```



$Cost_{PT1}$: 15,408

$Cost_{PT2}$: 12,478

Closing

Summary

- Query optimization phases
  - Semantic
  - Syntactic
    - Heuristics for the order of operations
    - Relational algebra equivalence rules
  - Physical
    - Cost-based optimization steps
      - Generation of alternatives
      - Cost estimation

Bibliography

- Y. Ioannidis. *Query Optimization*. ACM Computing Surveys, vol. 28, num. 1, March 1996
- R. Ramakrishnan and J. Gehrke. *Database Management Systems.* McGraw-Hill, 3rd Edition, 2003
- J. Lewis. *Cost-Based Oracle Fundamentals*. Apress, 2006
- S. Lightstone, T. Teorey and T. Nadeau. *Physical Database Design*. Morgan Kaufmann, 2007