CS5300
Database Systems

Query Processing and Query Optimization

A.R. Hurson
323 CS Building
hurson@mst.edu

Database Systems

Note, this unit will be covered in four lectures. In case you finish it earlier, then you have the following options:

1) Take the early test and start CS5300.module6
2) Study the supplement module (supplement CS5300.module5)
3) Act as a helper to help other students in studying CS5300.module5

Note, options 2 and 3 have extra credits as noted in course outline.

Database Systems

Enforcement of background

Glossary of prerequisite topics

Familiar with the topics?

No

Review CS5300.module5.background

Yes

Take Test

Pass?

No

Remedial action

Yes

Glossary of topics

At the end: take exam, record the score, impose remedial action if not successful

Current Module

Familiar with the topics?

No

Take the Module

Yes

Take Test

Pass?

No

Yes

Options

Study next module?

Lead a group of students in this module (extra credits)?

Study more advanced related topics (extra credits)?

Extra Curricular activities

Database Systems

- You are expected to be familiar with:
  - Relational database model,
  - SQL
- If not, you need to study CS5300.module5.background

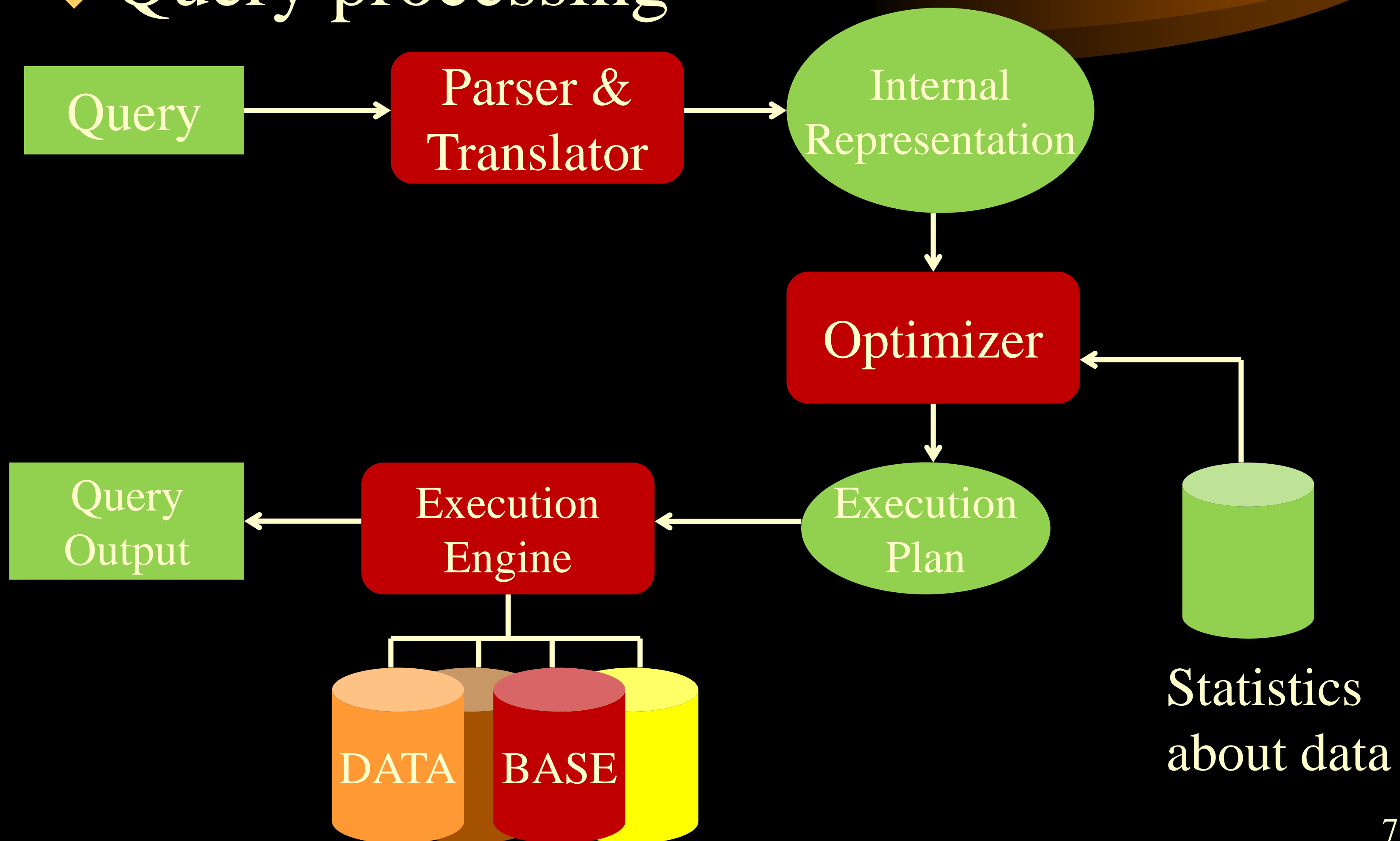
- In this part We will learn:
  - General steps in query processing and determining a suitable query plan,
  - Several heuristic rules to optimize a query,
  - Several systematic optimization rules, and
  - How to perform basic database operations, in general, and more specifically based on the underlying data organization and computer platform.

Database Systems

- Query processing
  - A query processing involves three steps:
    - Parsing and Translation
    - Optimization
    - Evaluation

Database Systems

- Query processing



Database Systems

- Query processing – An Example

```
Select   balance
From     account
Where    balance < 2,500
```

Database Systems

- Query processing – An Example

  $\sigma_{balance < 2500} (\Pi_{balance} (account))$

  or

  $\Pi_{balance}(\sigma_{balance < 2500} (account))$

- Note there might be different ways to define and execute a query. It is the role of optimizer to select an efficient way to execute a query. Therefore, the optimizer needs to determine different ways (plans) that one can execute a query, determine the execution cost of each plan, and then choose the most cost effective plan for execution.

Database Systems

- Query processing – An Example
  - Factors such as number of accesses to the disks and CPU time must be taken into consideration to estimate cost of a plan.
  - In large databases, however, disk accesses (the number of data block transfers) are usually the most dominating cost factor. Hence, it can be used as a cost metric.

Database Systems

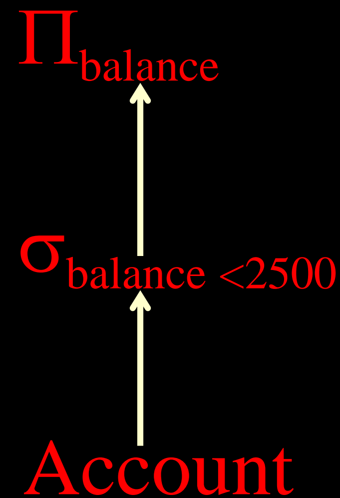
- Query processing — An Example
  - To simplify the cost estimation, we can assume that all block transfers cost the same (i.e., variances in rotational latency and seek time are ignored).
  - For more accurate measure, one also need to distinguish the difference between sequential I/O and random I/O as well.

Database Systems

- Query processing — An Example
  - One also needs to distinguish between the number of data blocks being read and written.
  - Techniques such as pipelining and parallelism, if possible, depending on the underlying platform, can be applied to execute basic operations.
  - Different algorithms can be developed to execute basic operations.

Database Systems

- Query processing – An Example

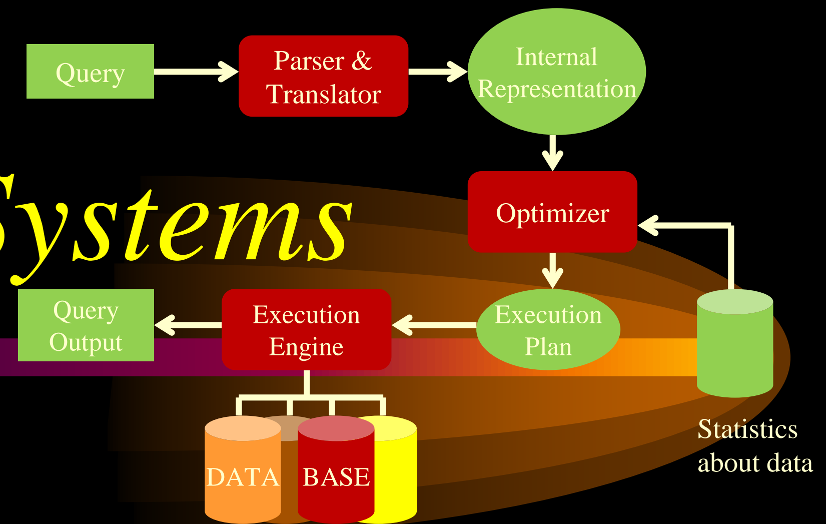


Database Systems

- Query Optimization
  - In general, optimization is required in such a system if the system is expected to achieve acceptable objectives (e.g., performance).
  - It is one of the strength of relational algebra that optimization can be done automatically, since relational expression are at a sufficiently high semantic level.

Database Systems

- Query Optimization
  - The overall goal of an optimization is to choose an efficient strategy for evaluation of a given relational expression (i.e., a query).
  - An optimizer might actually do better than a human programmer since:



Database Systems

Query Optimization

- An optimizer will have a wealth of information available to it that human programmers typically do not have.
- If the data base statistics changes drastically, then an optimizer may choose a different strategy.
- Optimizer can potentially considers several strategies for a given request.
- Optimizer is written by an expert.

Database Systems

- Running Example

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_Location

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Database Systems

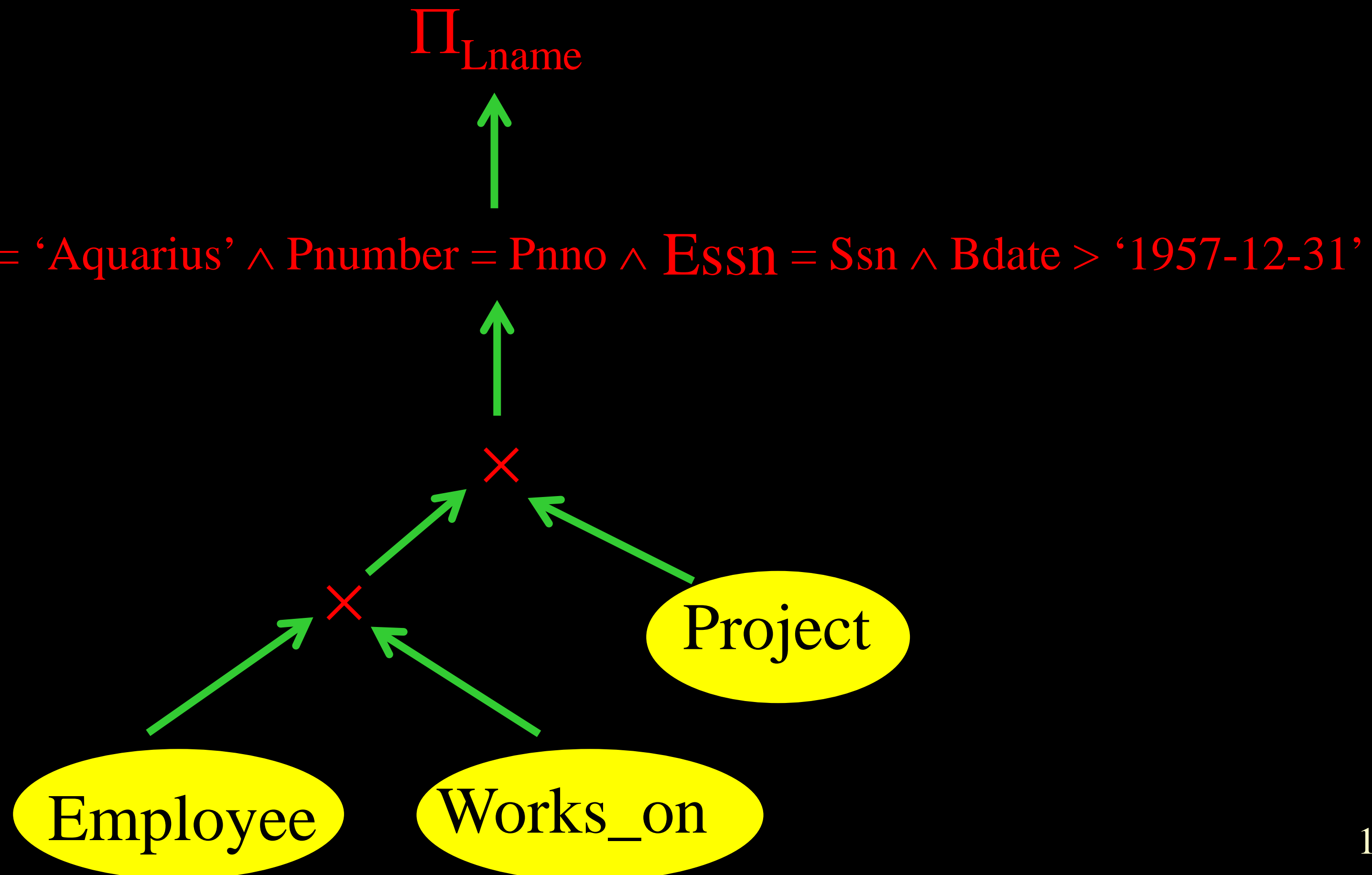
◆ Query Optimization — Running Example

- Find the last name of employees born after 1957 and working on a project named “Aquarius”.

```
SELECT    Lname
FROM      EMPLOYEE, WORKS_ON, PROJECT
WHERE     Pname = 'Aquarius' AND Pnumber = Pno  AND Essn
          = Ssn AND Bdate > '1957-12-31';
```

Database Systems

- Query Optimization — Running Example



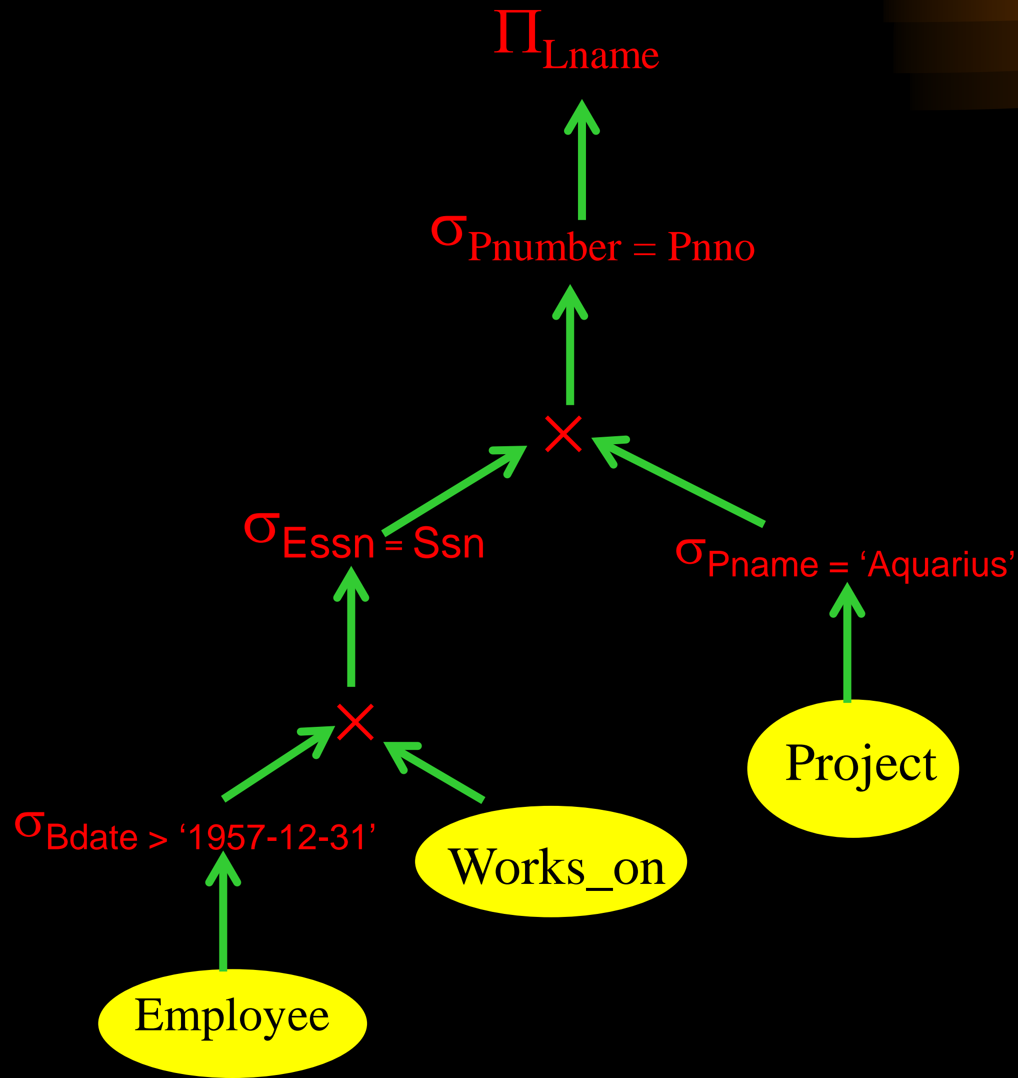
Database Systems

Query Optimization — An Example

- Execution of the previous query tree generates a very large relation because of performing Cartesian products on input relations.
- It makes sense to perform some Select operations on base relations before performing the Cartesian products.

Database Systems

◆ Query Optimization — Running Example

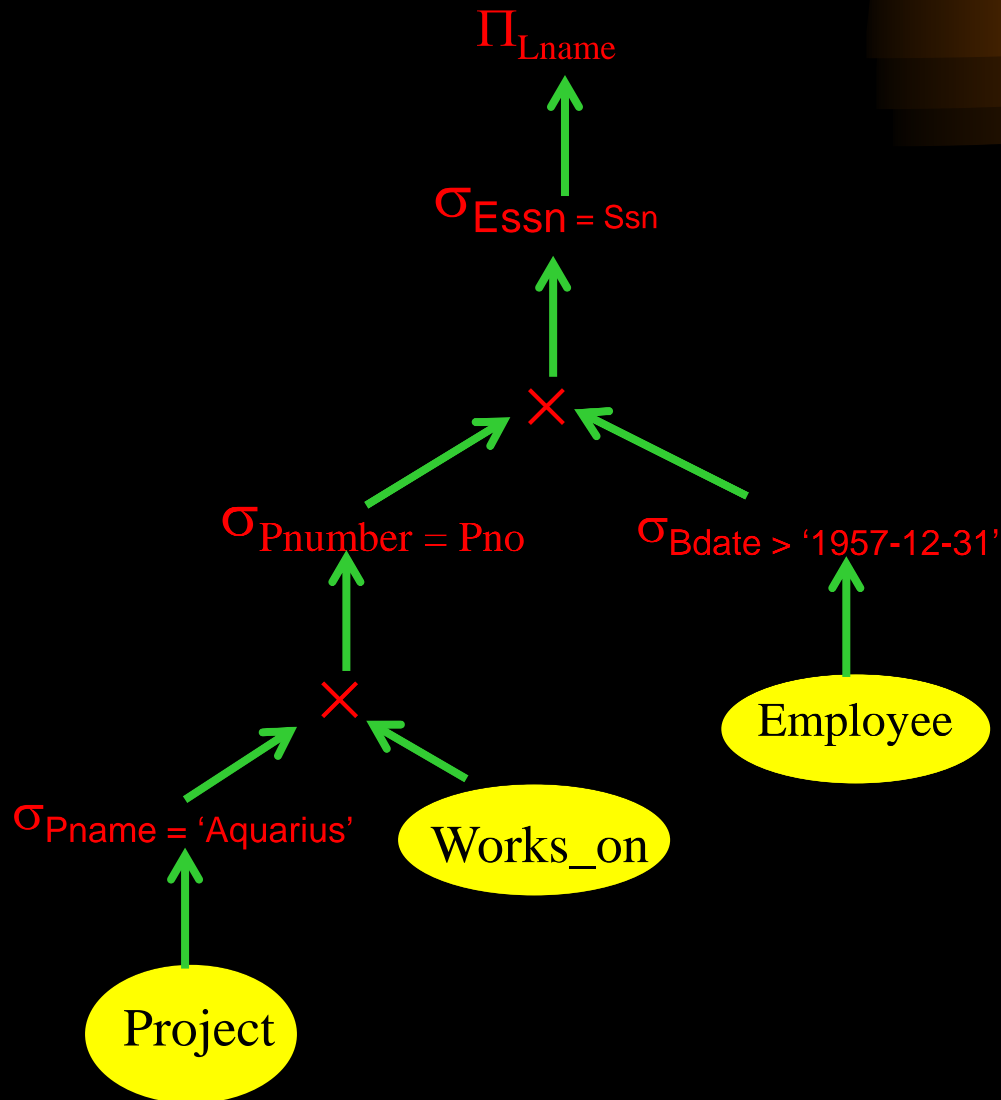


Database Systems

- Query Optimization — Running Example
  - By closer observation, one should realize that just one tuple from the Project will be involved with the query. So it makes sense to switch the order of operations on input relations.

Database Systems

◆Query Optimization — Running Example

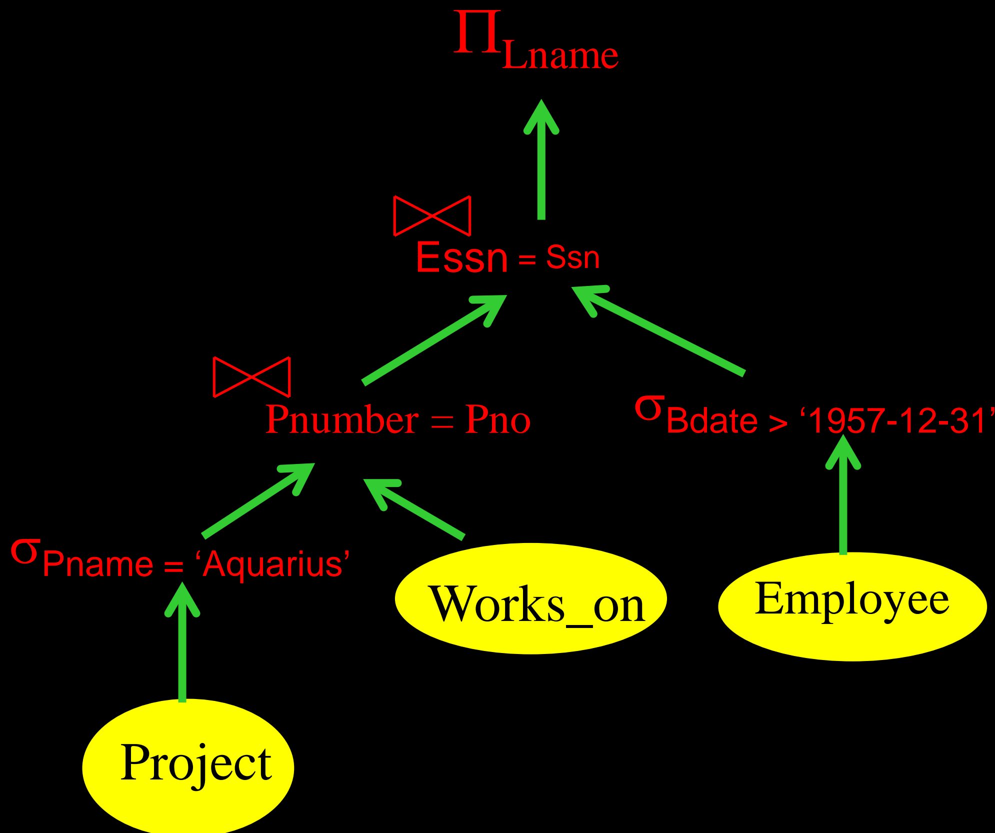


Database Systems

- Query Optimization — Running Example
  - It also makes sense to replace any Cartesian product followed by a Select operation with a Join operation.

Database Systems

- Query Optimization — Running Example

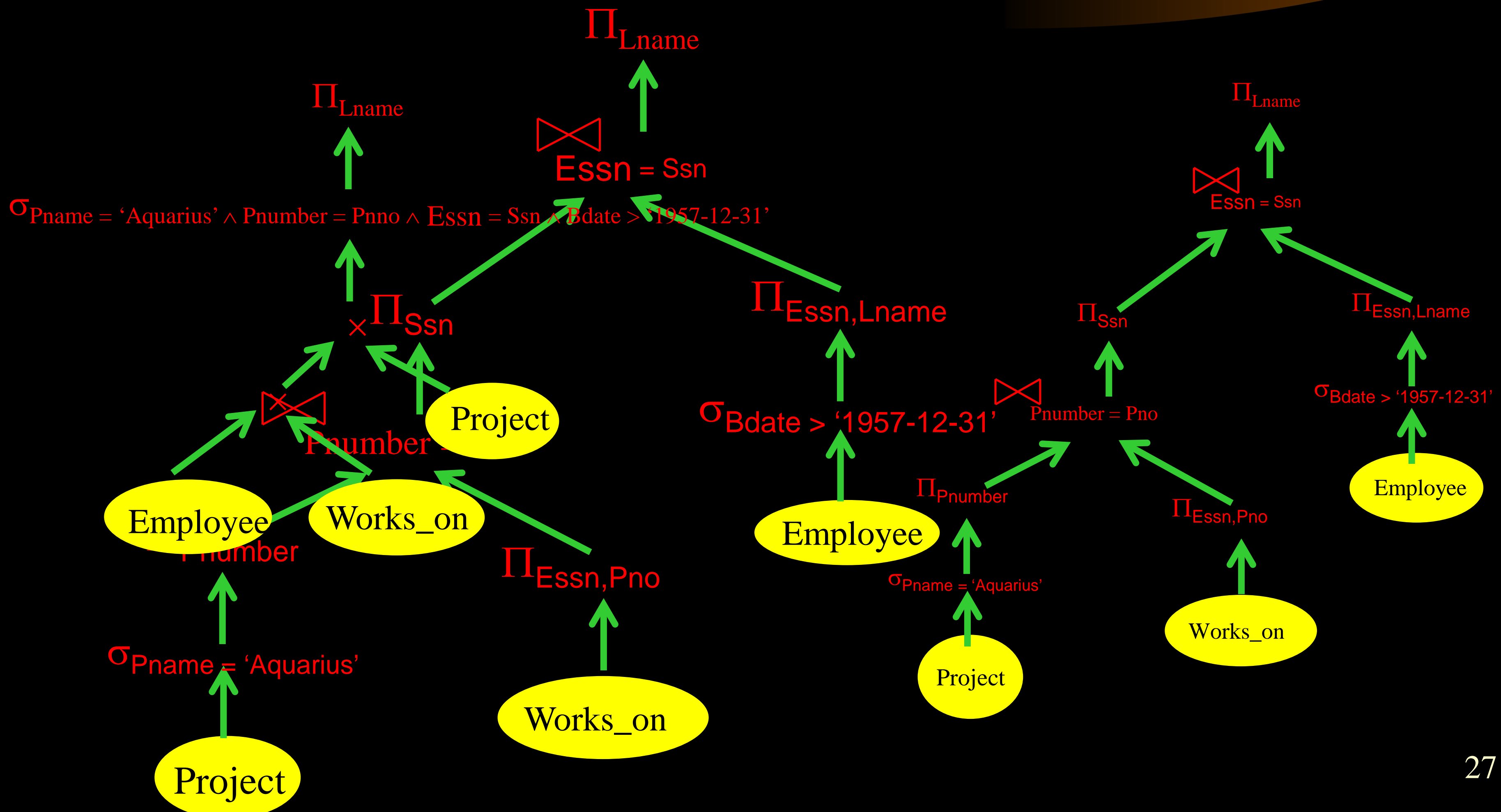


Database Systems

- Query Optimization — Running Example
  - It also makes sense to reduce the size of intermediate results by keeping just attributes that are needed for correct execution of this query.

Database Systems

◆ Query Optimization — Running Example



Database Systems

- Query Optimization — A Simple Example

S

S#	Sname	Status	City
S_1	Smith	20	London
S_2	Jones	10	Paris
S_3	Blake	30	Paris
S_4	Clark	20	London
S_5	Adams	30	Athens

SP

S#	P#	QTY
S_1	P_1	300
S_1	P_2	200
S_1	P_3	400
S_1	P_4	200
S_1	P_5	100
S_1	P_6	100
•		
•		
•		

Database Systems

- Query Optimization — A Simple Example
  - Get names of suppliers who supply part P_2:

```
SELECT    DISTINCT  Sname
FROM      S, SP
WHERE     S.S# = SP.S#
AND       SP.P# = 'P2';
```

  - Suppose that the cardinality of *S* and *SP* are 100 and 10,000, respectively. Furthermore, assume 50 tuples in *SP* are for part P_2.

Database Systems

- Query Optimization — A Simple Example

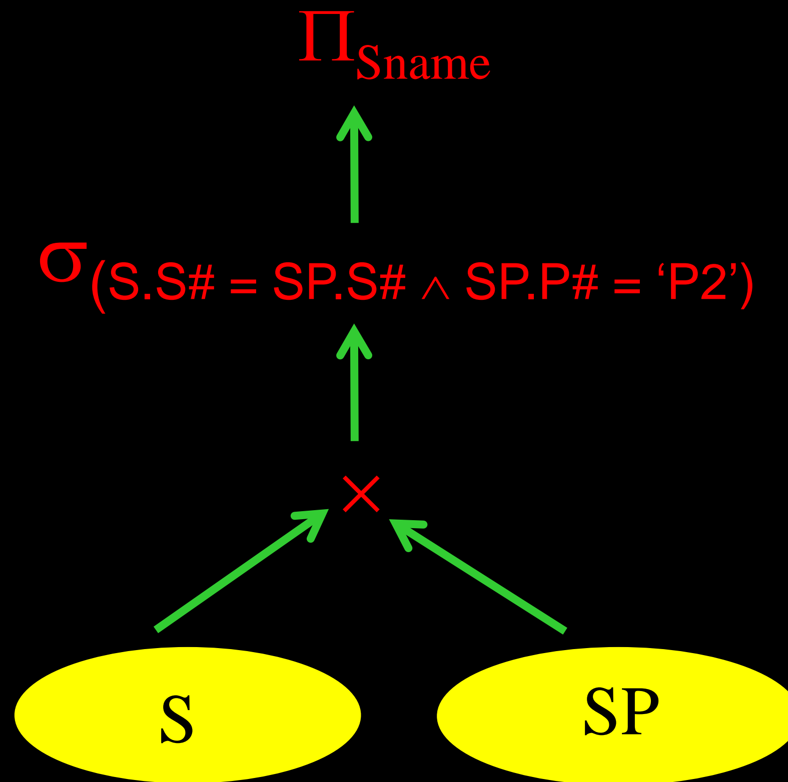
S#	Sname	Status	S.City	S#	P#	QTY
S_1	Smith	20	London	S_1	P_1	300
S_1	Smith	20	London	S_1	P_2	200
S_1	Smith	20	London	S_1	P_3	400
S_1	Smith	20	London	S_1	P_4	200
S_1	Smith	20	London	S_1	P_5	100
S_1	Smith	20	London	S_1	P_6	100
S_2	Jones	10	Paris	S_2	P_1	300
S_2	Jones	10	Paris	S_2	P_2	400
•						
•						

Database Systems

- Query Optimization — A Simple Example
  - Without an optimizer, the system will:
    - Generates Cartesian product of *S* and *SP*. This will generate a relation of size 1,000,000 tuples — Too large to be kept in the main memory.
    - Restricts results of previous step as specified by WHERE clause. This means reading 1,000,000 tuples of which 50 will be selected.
    - Projects the result of previous step over *Sname* to produce the final result.

Database Systems

- Query Optimization — A Simple Example

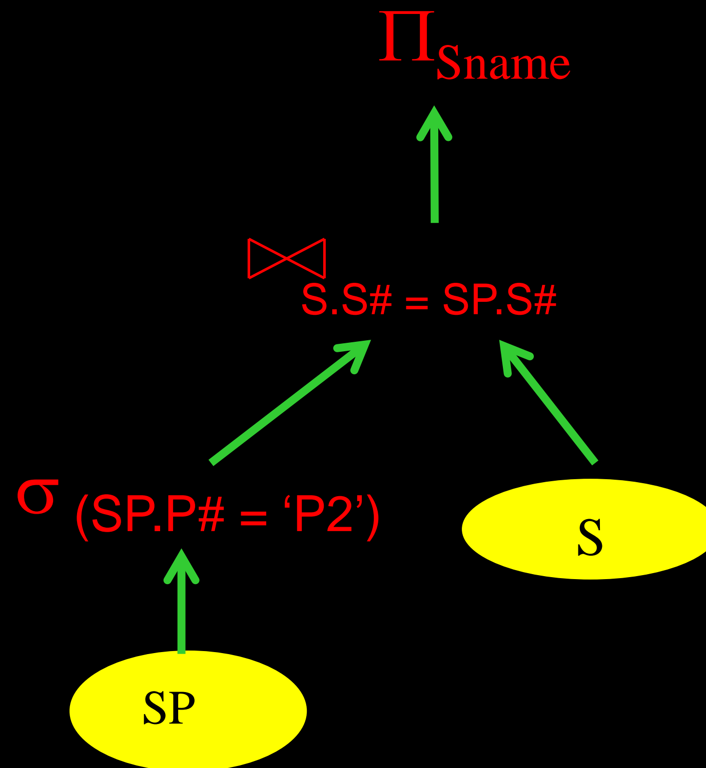


Database Systems

- Query Optimization — A Simple Example
  - An Optimizer on the other hand:
    - Restricts *SP* to just the tuples for part P_2. This will involve reading 10,000 tuples, but produces a relation with 50 tuples.
    - Joins the result of the previous step with *S* relation over *S#*. This involves the retrieval of only 100 tuples and the generation of a relation with at most 50 tuples.
    - Projects the result of the last operation over *Sname*.

Database Systems

- Query Optimization — A Simple Example



Database Systems

- Query Optimization — A Simple Example
  - If the number of tuples I/O's is used as the performance measure, then it is clear that the second approach is far faster that the first approach. In the first case we read/write about 3,000,000 tuples and in the second case we read about 10,000 tuples.
  - So a simple policy — doing restriction and then join instead of doing product and then a restriction sounds a good heuristic.

Database Systems

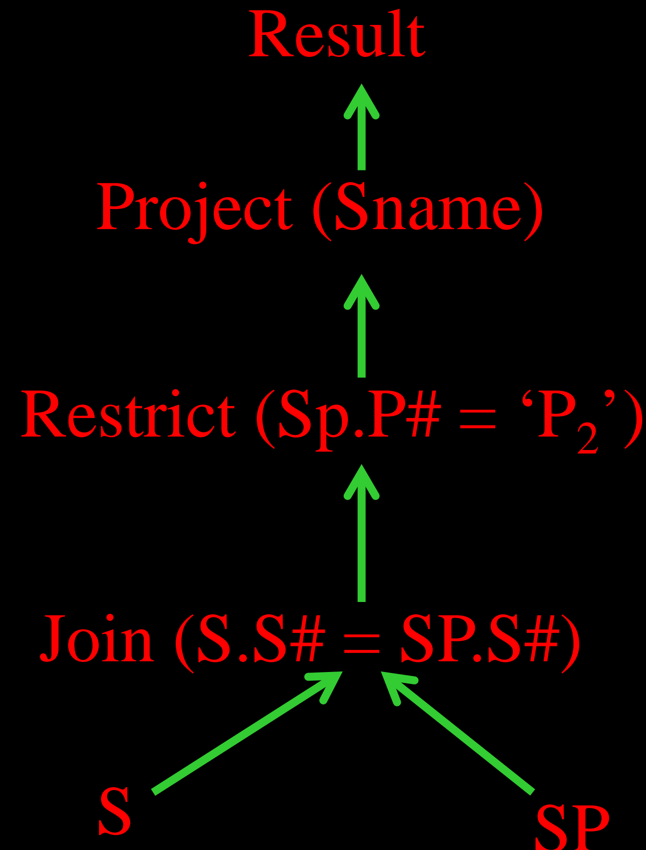
- Optimization Process
  - Cast the query into some internal representation — Convert the query to some internal representation that is more suitable for machine manipulation, relational algebra.

$$\Pi_{(Sname)}(\sigma_{P\# = \text{“P2”}}(S \bowtie_{S.S\# = SP.S\#} SP))$$

  - Now we can build a query tree very easily.

Database Systems

- Optimization Process



Database Systems

- Optimization Process
  - Convert the result of the previous step into a canonical form — during this phase, optimizer performs a number of optimization that are "guaranteed to be good" regardless of the actual data value and the access paths. For Example:

Database Systems

- Optimization Process

(A Join B) WHERE restriction-on-B
can be transformed into
(A Join (B WHERE restriction-on-B))

(A Join B) WHERE restriction-on-A AND restriction-on-B
can be transformed into
(A WHERE restriction-on-A) Join (B WHERE restriction-on-B))

Database Systems

- Optimization Process
  - General rule: It is a good idea to perform the restriction before the join, because:
    - It reduces the size of the input to the join operation,
    - It reduces the size of the output from the join.

Database Systems

- Optimization Process

WHERE p OR (q AND r)
can be converted into
WHERE (p OR q) AND (p OR r)

Database Systems

- Optimization Process
  - General rule: Transform restriction condition into an equivalent condition in conjunctive normal form, because:
    - A condition that is in conjunctive normal form evaluates to "true" only if every conjunct evaluates to "true". Consequently, it evaluates to "false" if any conjunct evaluates to "false". This is specially useful in the domain of parallel systems where conjuncts can be evaluated in parallel.

- Optimization Process

  (A WHERE restriction-1) WHERE restriction-2

  can be converted into

  A WHERE restriction-1 AND restriction-2

- Optimization Process
  - General rule: A sequence of restrictions can be combined into a single restriction.

- Optimization Process

  (A [projection-1]) [projection-2]

  can be converted into

  A [projection-2]

- Optimization Process
  - General rule: A sequence of projections can be transferred into a single projection.

- Optimization Process
  - General rule: A restriction and projection can be converted into a projection and restriction.

Database Systems

- Optimization Process
  - Finally, consider the following query:
  - Get the supplier numbers who supply at least one part;

    (SP Join P) [S#]
  - However, we know that P# is the foreign key in SP, therefore the above query is semantically equivalent to:

    SP [S#]

Database Systems

Optimization Process

- An equivalence rule says that expressions in different forms are equivalent. In another words, an expression in one form can be replaced by its equivalent expression.
- Since the computational cost of equivalent relations may vary, the optimizer can use equivalence rules to transform expression while satisfying performance metrics.

Database Systems

- Optimization Process
  - Rule 1: Conjunctive selection operations (cascade of selections) can be deconstructed into a sequence of individual selections:

$$\sigma_{\theta 1 \wedge \theta 2}(E) = \sigma_{\theta 1}(\sigma_{\theta 2}(E))$$

Database Systems

- Optimization Process
  - Rule 2: Selection operation is commutative:

$$\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$$

Database Systems

- Optimization Process
  - Rule 3: A sequence of projections is the same as the last projection operation (cascade of projections):

$$\Pi_{L1}(\Pi_{L2}(\ldots(\Pi_{Ln}(E))\ldots)) = \Pi_{L1}(E)$$

Database Systems

- Optimization Process
  - Rule 4: A combination of selection and Cartesian product operations is equivalent to theta join operation:

$$\sigma_{\theta}(E_1 \times E_2) = E_1 \bowtie_{\theta} E_2$$

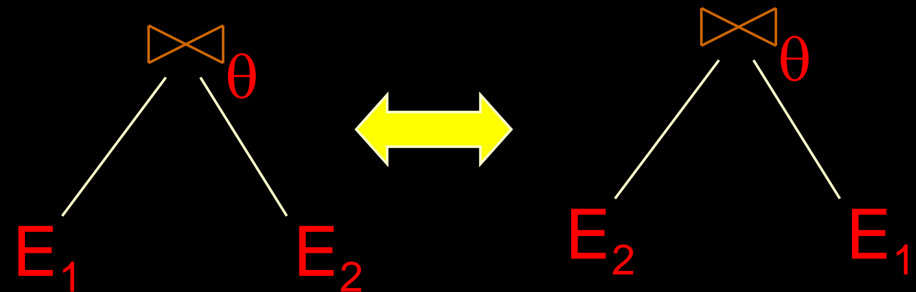
  - This can be extended to:

$$\sigma_{\theta 1}(E_1 \bowtie_{\theta 2} E_2) = E_1 \bowtie_{\theta 1 \wedge \theta 2} E_2$$

Database Systems

- Optimization Process
  - Rule 5: Theta join operation is commutative:

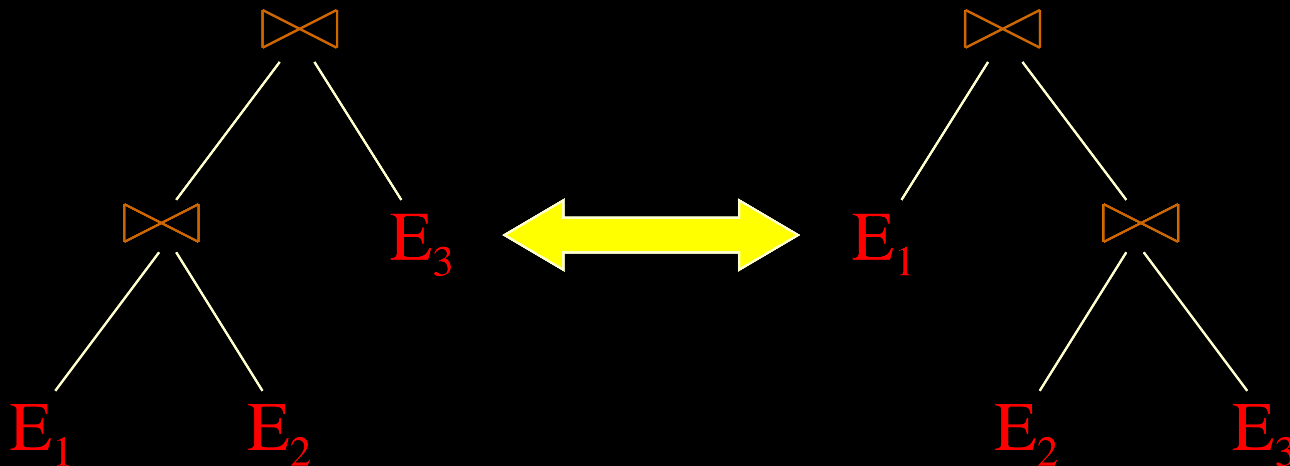
$$E_1 \bowtie_{\theta} E_2 = E_2 \bowtie_{\theta} E_1$$



Database Systems

- Optimization Process
  - Rule 6: Natural join is associative:

$$(E_1 \bowtie E_2) \bowtie E_3 = E_1 \bowtie (E_2 \bowtie E_3)$$



Database Systems

- Optimization Process
  - Rule 7: Theta join is associative in the following manner:

$$(E_1 \bowtie_{\theta 1} E_2) \bowtie_{\theta 2 \wedge \theta 3} E_3 = E_1 \bowtie_{\theta 1 \wedge \theta 3} (E_2 \bowtie_{\theta 2} E_3)$$

Where θ_2 involves attributes from only E_2 and E_3.

Database Systems

- Definition
  - Selectivity is defined as the ratio of the number of tuples that satisfy the equality condition to the cardinality of the relation.

$$selectivity = \frac{\#of\ tuples\ satisfying\ the\ search}{|r(R)|}$$

  - Selectivity is used to estimate size of intermediate relation and hence number of accesses.

Database Systems

- In practice selectivities of all conditions is not available so we use estimated selectivity as part of statistical data to aid query optimization.

Database Systems

- Selectivity on key attribute and search on equality then:

$$s = \frac{1}{|r(R)|}$$

Database Systems

- Selectivity on an attribute with *i* distinct values is:

$$s = {}^{|r(R)|}/_{i} \Big/ _{|r(R)|}$$

- Hence the number of tuples that satisfy an equality search is:

$$\frac{1}{i} * |r(R)|$$

Database Systems

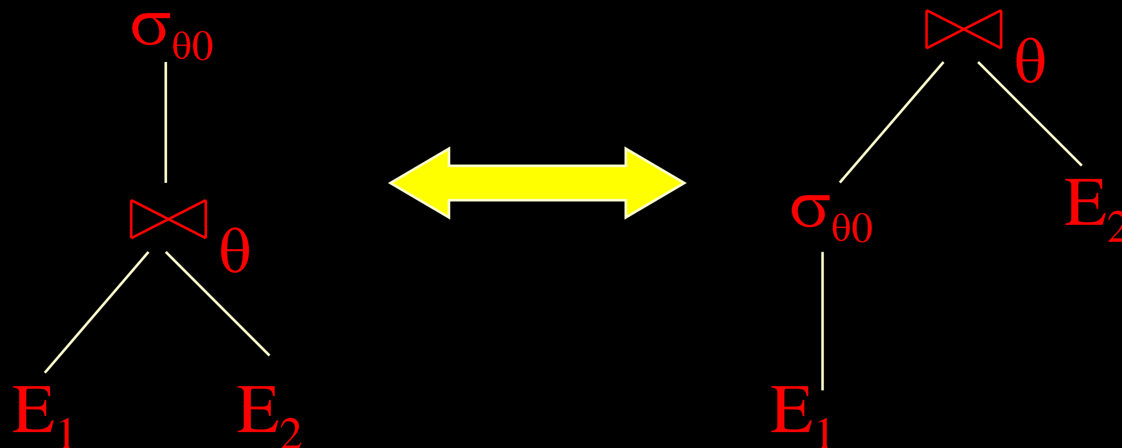
- Optimization Process
  - Rule 8: Selection operation distribute over the theta join under the following conditions:
    - When all attributes in selection condition θ_0 involve only the attributes of one relation (E_1 in this case):

$$\sigma_{\theta 0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_{\theta} E_2$$

Database Systems

- Optimization Process
  - Rule 8:

$$\sigma_{\theta 0}(E_1 \bowtie_{\theta} E_2) = (\sigma_{\theta 0}(E_1)) \bowtie_{\theta} E_2$$



Database Systems

- Optimization Process
  - Rule 9: The projection operation distributes over theta-join under the following condition:
    - Join condition θ only involves attributes in $L_1 \cup L_2$:

$$\Pi_{L1 \cup L2}(E_1 \bowtie_\theta E_2) = (\Pi_{L1}(E_1)) \bowtie_\theta (\Pi_{L2}(E_2))$$

Database Systems

- Optimization Process
  - Rule 10: Set union and set intersection operations are commutative:

$$(E_1 \cup E_2) = (E_2 \cup E_1)$$
$$(E_1 \cap E_2) = (E_2 \cap E_1)$$

  - Note, set difference is not commutative.

Database Systems

- Optimization Process
  - Rule 11: Set union and set intersection operations are associative:

$$(E_1 \cup E_2) \cup E_3 = E_1 \cup (E_2 \cup E_3)$$

$$(E_1 \cap E_2) \cap E_3 = E_1 \cap (E_2 \cap E_3)$$

Database Systems

- Optimization Process
  - Rule 12: Selection operation distributes over the set union, set intersection, and set difference operations:

$$\sigma_p (E_1 - E_2) = \sigma_p (E_1) - \sigma_p (E_2)$$

$$\sigma_p (E_1 - E_2) = \sigma_p (E_1) - (E_2)$$

Database Systems

- Optimization Process
  - Rule 12

$$\sigma_p (E_1 \cup E_2) = \sigma_p (E_1) \cup \sigma_p (E_2)$$

$$\sigma_p (E_1 \cup E_2) \neq \sigma_p (E_1) \cup (E_2)$$

Database Systems

- Optimization Process
  - Rule 12

$$\sigma_p (E_1 \cap E_2) = \sigma_p (E_1) \cap \sigma_p (E_2)$$

$$\sigma_p (E_1 \cap E_2) = \sigma_p (E_1) \cap (E_2)$$

Database Systems

- Optimization Process
  - Rule 13: Projection operation distributes over the set union, set intersection?, and set difference operations?:

$$\Pi_L(E_1 - E_2) = (\Pi_L(E_1)) - (\Pi_L(E_2))?$$

$$\Pi_L(E_1 \cup E_2) = \Pi_L(E_1) \cup \Pi_L(E_2)$$

$$\Pi_L(E_1 \cap E_2) = \Pi_L(E_1) \cap \Pi_L(E_2)?$$

Database Systems

- Optimization Process
  - Choose candidate low-level procedure — After transferring the query into more desirable form, the optimizer must then decide how to evaluate the transformed query. At this stage issues such as:
    - existence of indexes or other access paths ─ To reduce I/O cost, and
    - physical clustering of records ─ To reduce I/O cost, …

    comes into play.

Database Systems

- Optimization Process
  - So, in short:
    - after scanning and parsing,
    - the query will be translated into an equivalent representation, this internal representation is in the form of a query tree or query graph,
    - an execution strategy will be chosen. The execution strategy is a plan for accessing the data, executing the query, and storing the intermediate results.

Database Systems

- Optimization Process
  - Generate query plans — The final stage of optimization involve the construction of a set of candidate query plans and the choice of "the best of these plans".
  - Choosing the cheapest plan, naturally, requires a method for assigning a cost to any given plan — This cost formula should estimate the number of disk accesses, CPU utilization and execution time, space utilization,….

Database Systems

- Optimization Process
  - There are two main techniques for query optimization:
    - Heuristic rules
    - Systematic estimation approach
  - In this course, as noted before, we will talk about the heuristic rules.

Database Systems

- Optimization Process – heuristic rules
  - Perform selection operations as early as possible.
  - Perform projections early.
  - It is usually better to perform selections earlier than projections.

Database Systems

- Optimization Process – heuristic rules
  - Based on heuristic rules the optimizer uses equivalence relationships to reorder operations in a query for execution.

- Definition:
  - Materialized evaluation: Generation of intermediate result (relation).
  - Pipeline evaluation: Combining several operations.

Database Systems

- Assume we want to perform:

$$\Pi_{a1,\ a2}(r \bowtie s)$$

We can perform the join operation, materialize the resultant, and then apply projection.

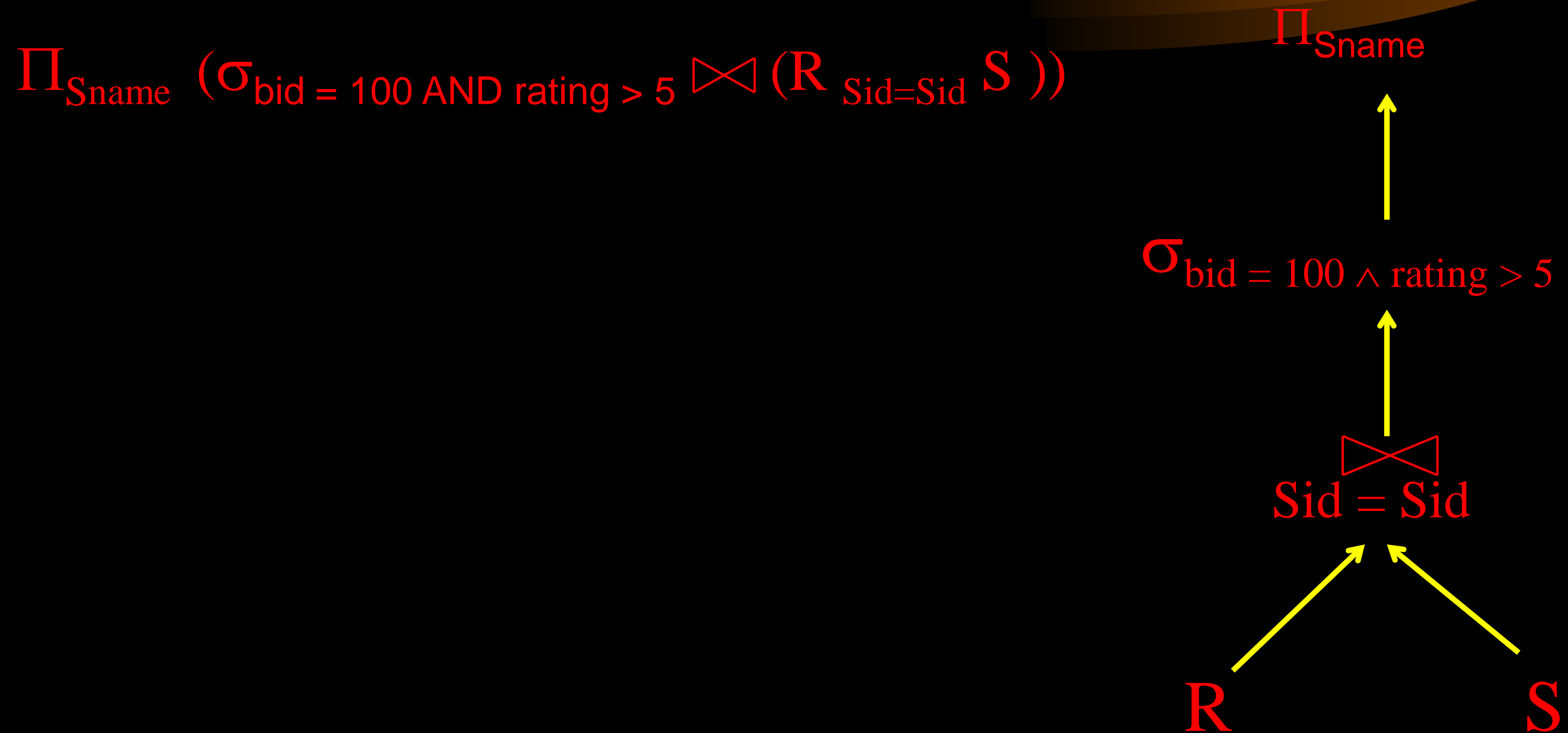
Alternatively, we can do the following: When the join operation generates a tuple, it will be passes directly to the project operation for processing.

Database Systems

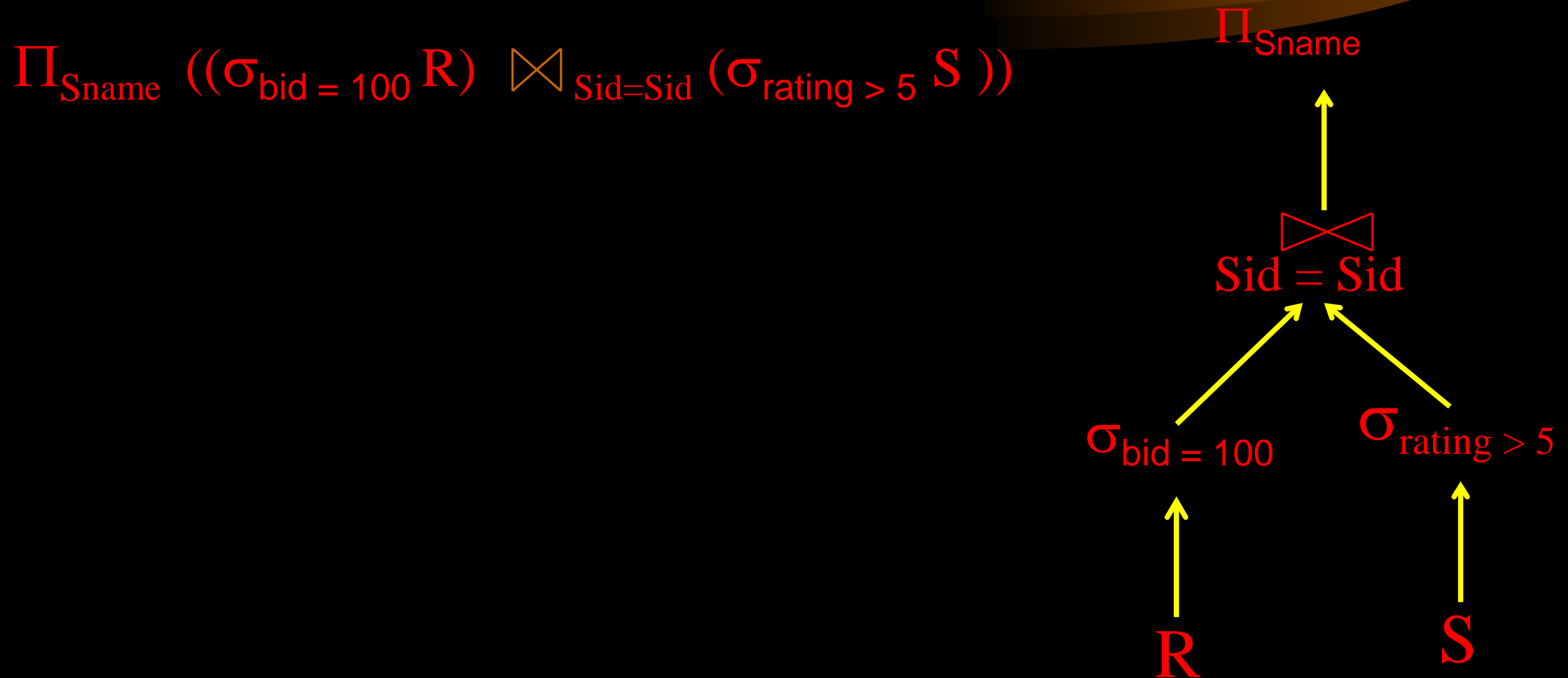
- Assume the following relations:
  - S (S_{id}: integer, S_{name}: string, rating: integer, age: real)
  - R (S_{id}: integer, bid: integer, day: dates, r_{name}: string)
- Further assume the following query:

```
SELECT    S.S_name
FROM      R, S
WHERE     R.S_id = S.S_id
          AND R.bid = 100  AND S.rating > 5
```

Database Systems



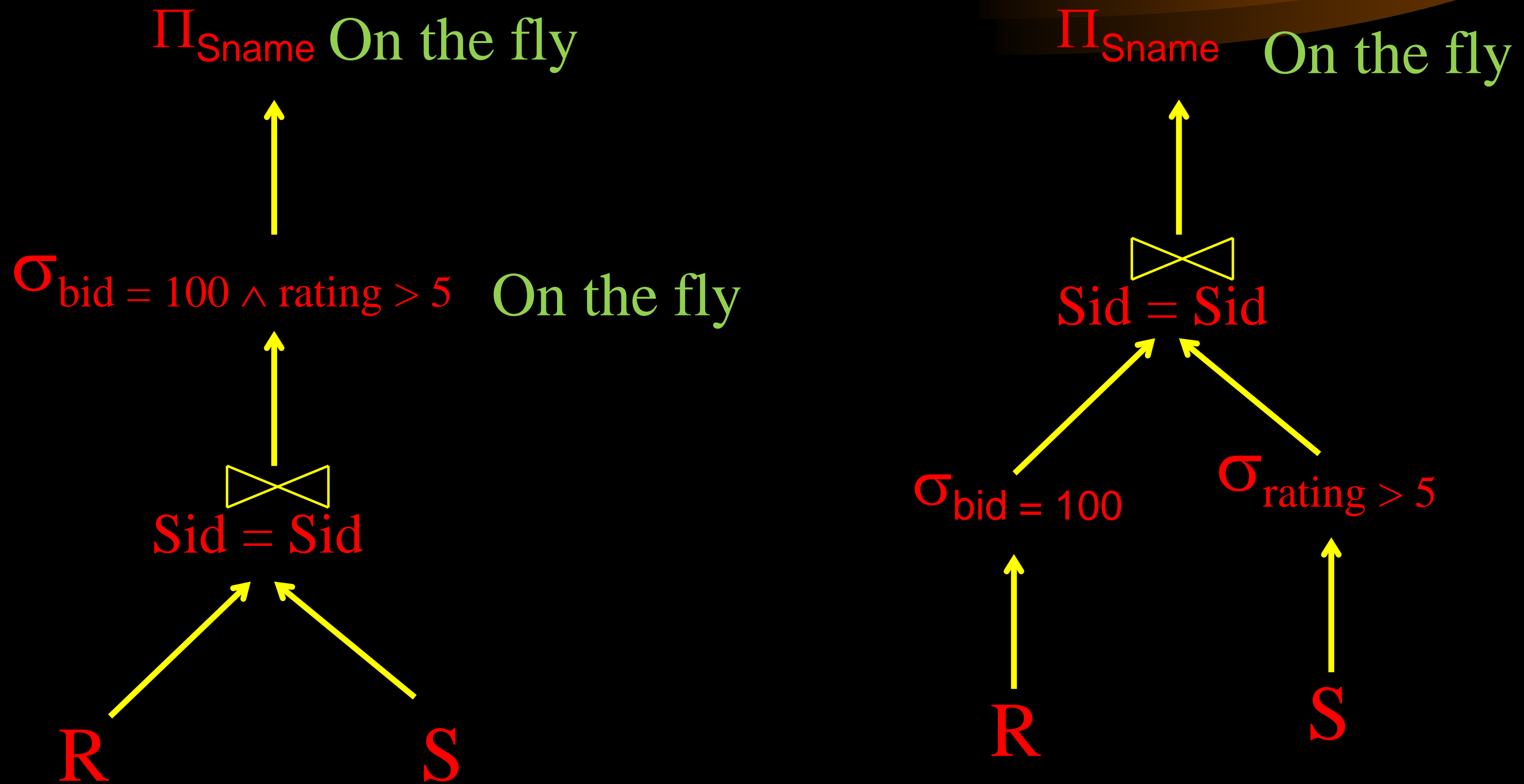
Database Systems



Database Systems

- Assume the underlying platform can perform the basic relational operations in "pipeline" fashion – i.e., result of one operation is fed to another operation.
- In this case, articulate the way the previous query is going to be executed?

Database Systems



Database Systems

- Cost of Plan
  - The cost associated with each plan needs to be estimated. This will be accomplished by estimating the cost of each operation.
  - Factors such as: size of relation (s), underlying architecture, buffer size, size of the memory, "reduction factor" for each operation, … need to be taken into consideration.

Database Systems

- Optimization Process — a commercial system
  - In this system, we are dealing with SQL statements — in the form of "SELECT-FROM-WHERE":
    - An order of execution of query blocks are decided (nested query).
    - An attempt is made to minimize the total cost by choosing the cheapest implementation for each individual block.
    - For a given query block, two cases are considered;

Database Systems

- Optimization Process — a commercial system
  - For a block that involves just a restriction and/or projection of a single relation:
    - statistical information from the system catalog together with proper formulas to estimate size of the intermediate results and cost of low-level operations are used to choose an execution strategy. The statistics maintained in the system catalog are;
      - Number of tuples in each relation,
      - Number of pages occupied by each relation,
      - Percentage of pages occupied by each relation with respect to all pages in the relevant portion of the data base,
      - Number of distinct data values for each index,
      - Number of pages occupied by each index.

Database Systems

- Optimization Process — a commercial system
  - For a block that involves two or more relations to be joined together with local restrictions and/or projections, the optimizer:
    - treats each individual relation as in the case above, and
    - decides on the sequence for performing joins.

Database Systems

Optimization Process — a commercial system

- Given a set of relations to be joined, the optimizer always chooses a pair to be joined first, then a third to be joined to the result of joining the first two, and so on:

  A Join B Join C Join D

  is converted into

  ((D Join C) Join A) Join B

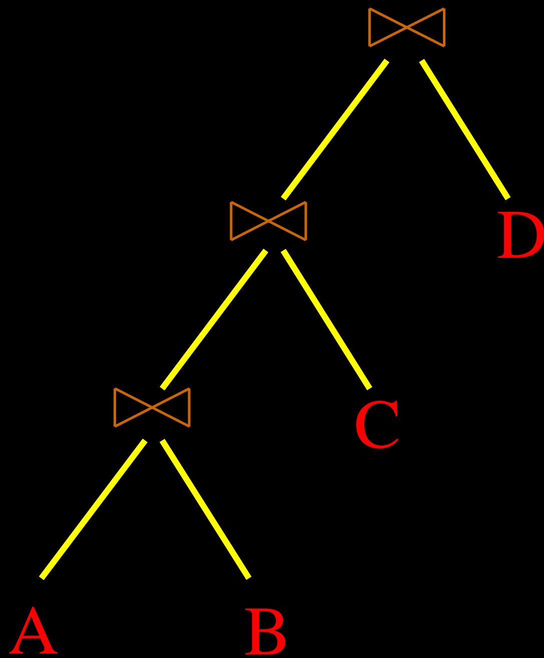
- Either the nested loop method or sort/merge method is used to perform the join.

Database Systems

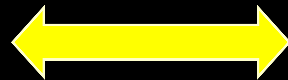
- Definition:
  - Linear tree: in a linear tree, at least one child of a join is a base relation.
  - Assume we have:

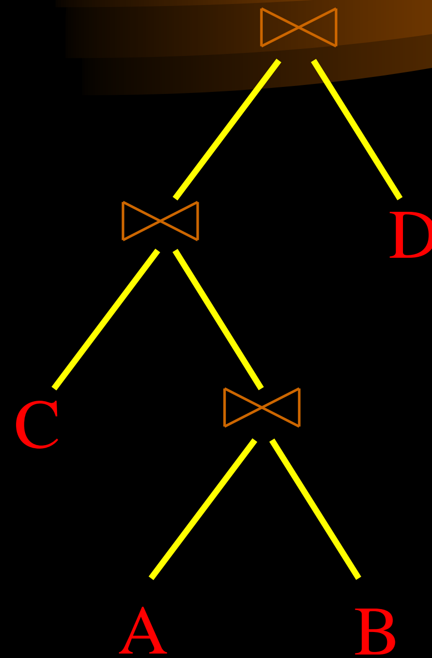
$A \bowtie B \bowtie C \bowtie D \Leftrightarrow (((A \bowtie B) \bowtie C) \bowtie D)$

Database Systems



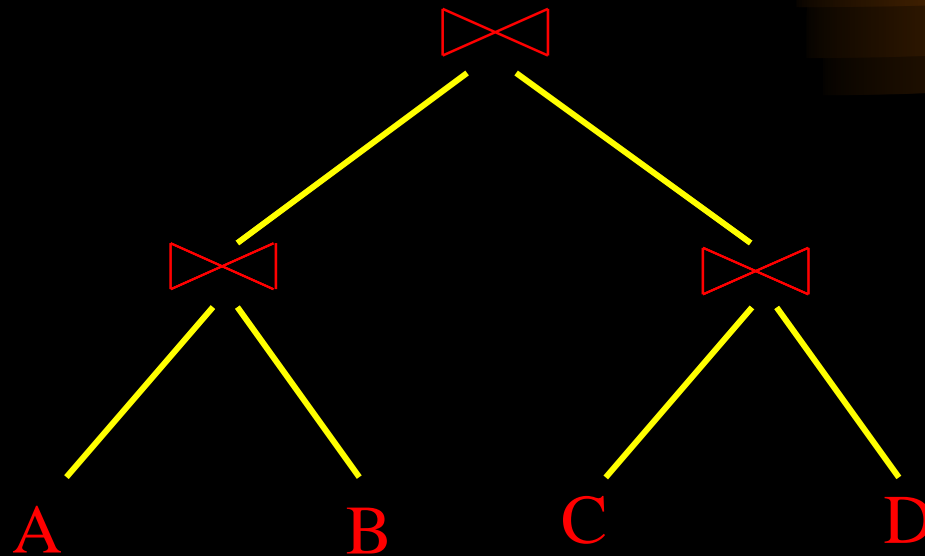
Left deep linear join tree





Bushy linear join tree

Database Systems



Non linear join tree

- Questions:
  - System R is using “left deep plan”, why?
  - Under what condition (s) “non linear Join tree” can be implemented?

- Optimization Process — Search methods for Selection
  - General Philosophy: Make effort to reduce the search space.

Database Systems

- Optimization Process — Search methods for Selection
  - Linear search: Retrieve every records in the file and test whether or not its attribute values satisfy the selection condition (In this case, data is not organized and no meta data is available).
  - Binary search: Use binary search method if the selection condition involves an equality comparison on a key attribute on which the file is ordered.

Database Systems

- Optimization Process — Search methods for Selection
  - Using a primary index or hash key to retrieve a single record: Use the primary index or hash key to retrieve the record if the selection condition involves an equality comparison on a key attribute with a primary index or hash key (note in this case at most one record is retrieved).

$$\sigma_{SSN = 123456789}(\text{EMPLOYEE})$$

Database Systems

- Optimization Process — Search methods for Selection
  - Using a primary index or hash key to retrieve multiple records: If the comparison condition is >, <, ≤, ≥ on a key field with a primary index, use the index to find the record satisfying the corresponding equality condition and then retrieve all the subsequent records in the file (note in this case, data is also sorted).

$$\sigma_{DNUMBER > 5}(DEPARTMENT)$$

Database Systems

- Query Optimization — Search methods for Selection
  - Using a clustering index to retrieve multiple records: If the selection condition involves an equality comparison on a non-key attribute with clustering index, use the clustering index to retrieve all the records satisfying the selection condition (clustered data).

$$\sigma_{DNO = 5}(EMPLOYEE)$$

Database Systems

- Query Optimization — Search methods for Selection
  - Conjunctive selection: conjunctive selection is of the following form;

$$\sigma_{\theta 1 \wedge \theta 2 \wedge \ldots\ldots \wedge \theta n}(r)$$

  - Disjunctive selection: disjunctive selection is of the following form;

$$\sigma_{\theta 1 \vee \theta 2 \vee \ldots\ldots \vee \theta n}(r)$$

Database Systems

- Query Optimization — Search methods for Selection
  - Conjunctive selection: If an attribute involved in any single simple condition in the conjunctive condition has an access path that allows the use of any aforementioned techniques, use that condition to retrieve the records and then apply the rest of the conditions.

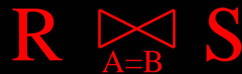
Database Systems

Query Optimization — Search methods for Selection

- Disjunctive selection by union of record pointers: If access path exists for all the attributes involved in disjunctive selection then each index is scanned for pointers to tuples that satisfy individual condition.
- The union of all the retrieved pointers yields the set of pointers to tuples satisfying the disjunctive condition.
- Note, even if one of the conditions does not have an access path, we will have to perform a linear scan of the relation.

Database Systems

- Query Optimization — JOIN Operation
  - Nested loop: For each record $t \in R$ (outer loop), retrieve every record of $s \in S$ (inner loop) and then check the join condition t[A] = s[B].



$R \bowtie_{A=B} S$

Database Systems

- Query Optimization — JOIN Operation (nested loop)
  - Suppose we want to perform

$$r \bowtie_{r.A\ \Theta\ s.B} s$$

  - A and B are attributes or set of attributes (i.e., join attributes) of relations r and s. Further assume $n_r = | r |$ and $n_s = | s |$ are the cardinality of the relations. Finally assume b_r and b_s are the number of blocks of each relation.

Database Systems

- Query Optimization — JOIN Operation (nested loop)
  - The following algorithm performs the nested loop join operation:

For each t_r ε r do begin

  For each t_s ε s do begin

    If r.A Θ s.B true then add $t_r \| t_s$ to the result

  end

end

Database Systems

- Query Optimization — JOIN Operation (nested loop)
  - Cost of nested loop algorithm is $n_r * n_s$.
  - In best case scenario, both relations fit into the physical space and hence we need $b_s + b_r$ block accesses.

Database Systems

- Query Optimization — JOIN Operation (nested loop)
  - If one of the relations fits in the physical space then $b_s + b_r$ block accesses will be the cost.

- Query Optimization — JOIN Operation (block nested loop)
  - If the buffer is too small to hold either relation, entirely, we can still obtain a major saving in the number of block accesses.

Database Systems

- Query Optimization — JOIN Operation (block nested loop)

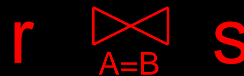
```
For each block B_r of  r do begin
    For each block B_s of  s do begin
        For each t_r ε  B_r do begin
            For each t_s ε  B_s do begin
                If r.A  Θ s.B true then add t_r || t_s to the result
            end
        end
    end
end
```

Database Systems

- Query Optimization — JOIN Operation (block nested loop)
  - Cost of block nested loop in term of number of block accesses is $b_r * b_s + b_r$.
  - How can we improve block nested loop?

Database Systems

- Query Optimization — JOIN Operation
  - Use of access structure to retrieve the matching record(s): If an index or hash key exists for one of the join attributes, say *B* of *s*, retrieve each record $t_r \in r$, one at a time, and then use the access structure to retrieve all the matching records $t_s \in S$ that satisfy $t_r[A] = t_s[B]$.

r  s

Database Systems

◆ Query Optimization — JOIN Operation

- Sort-merge: If the records of *r* and *s* are physically sorted by the value of the join attributes, then this technique can be applied by scanning *r* and *s* linearly.

Database Systems

Query Optimization — JOIN Operation (Merge)

- 1 pointer, initially pointing to the first tuple, is assigned to each relation. As the algorithm proceeds, the pointers move through the relations.
- Since the relations are sorted, each tuple is accessed once and hence the number of block accesses is:

$$b_s + b_r$$

Assuming that the set of all tuples with the same value for the join attributes fit in the main memory.

Database Systems

◆ Query Optimization — JOIN Operation

- hash-join: The records of both files *r* and *s* are hashed to the same hash file using the same hashing function. A single pass through each file hashes the records to the hash file buckets. Each bucket is then examined for records from *r* and *s* with matching join attribute values to produce a possible result for the join operation.

Database Systems

- Query Optimization — Complex JOIN Operation
  - Nested loop join can be used regardless of the join condition. The other join techniques, though more efficient than nested loop, can handle simple join conditions.
  - Join with complex join conditions (i. e., conjunctive and disjunctive conditions) can be implemented using techniques discussed for conjunctive and disjunctive selections.

Database Systems

Query Optimization — Complex JOIN Operation

- Consider the following join operation

$$r \bowtie_{\theta 1 \wedge \theta 2 \wedge \ldots \wedge \theta n} s$$

- One or more of the join techniques may be applicable for joins on individual conditions.
- We can perform the overall join by first computing one of theses simpler joins, say $r \bowtie_{\theta 1} s$. The result of complete join consists of those tuples in the intermediate result that satisfy the remaining conditions.

Database Systems

◆ Query Optimization — Complex JOIN Operation

- Now consider the following join operation

$$r \bowtie_{\theta 1 \vee \theta 2 \vee \dots \vee \theta n} s$$

- The join can be performed as the union of the tuples in individual joins $r \bowtie_{\theta i} s$.

Database Systems

Query Optimization — Project Operation

- A project operation $\Pi_{<attribute\text{-}list>}(R)$ is straightforward to implement if <attribute list> includes a key of relation R.
- If <attribute list> does not include a key, then we may end up with duplicates. Duplicates can be eliminated by sorting the result and then eliminating the duplicate or by using hashing technique.

Database Systems

Query Optimization — Set Operations

- Cartesian product is very expensive operation to perform. Hence, it is important to avoid it as much as possible.
- The other set operations can be implemented by sorting the relations and then a single scan through each relation is sufficient to generate the result.
- Hashing technique is another way to implement Union, intersection, and difference operations.

- Questions
  - Devise algorithms to perform variation of outer join operations.
  - Devise algorithms to perform aggregate operations.

Database Systems

- Query Optimization — An Example
  - Assume the following relations:

Department (Dname, Dnumber, Mgr-ssn, …)

Project (Pname, Pnumber, Plocation, Dnum)

Employee (Fname, Lname, Ssn, Bdate, address, Dno, …)

Database Systems

- Query Optimization — An Example

```
SELECT    Pnumber, Dnum, Lname, Bdate,
          Address
FROM      Project, Department, Employee
WHERE     Dnum = Dnumber
AND       MGRSSN = SSN
AND       Plocation = 'California';
```

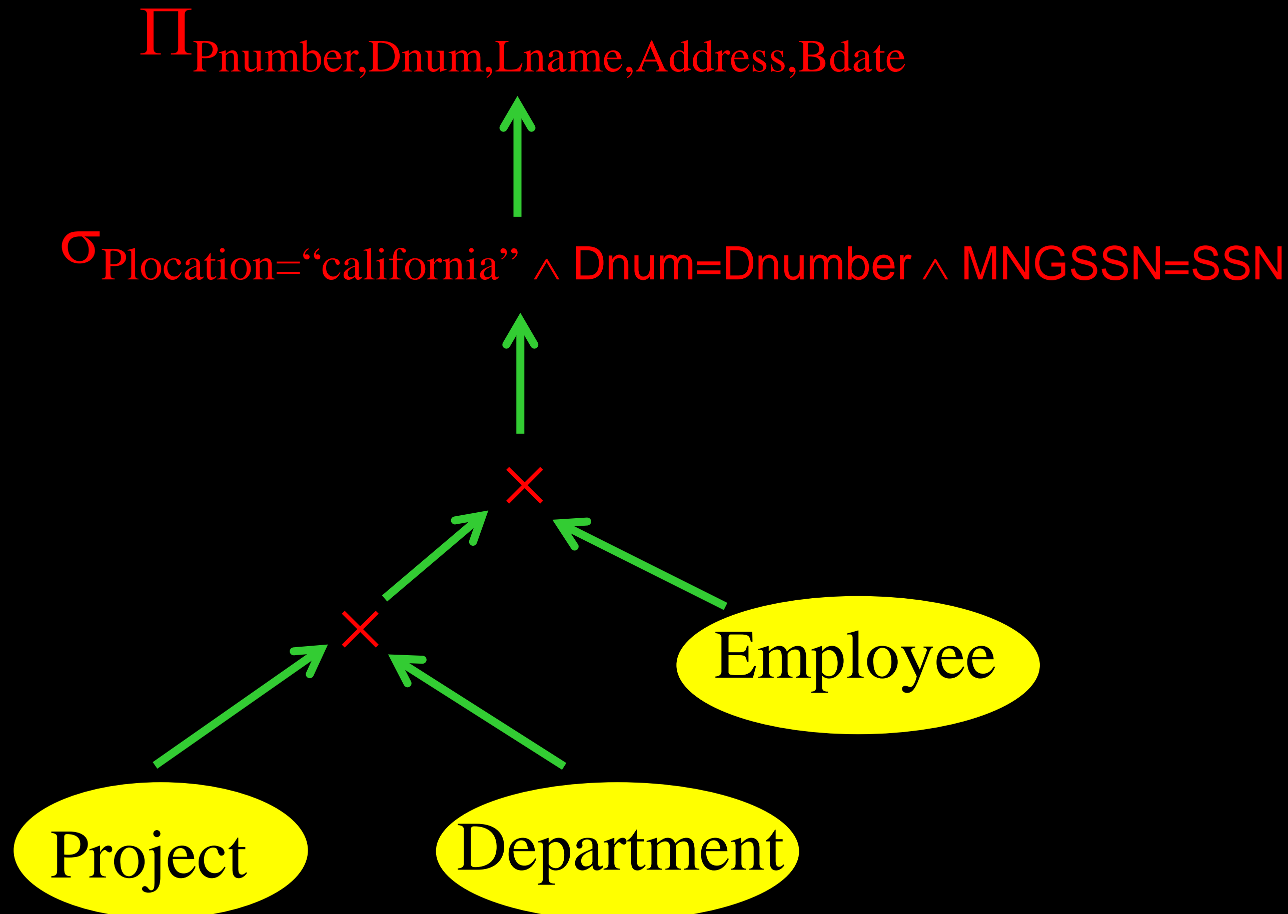
Database Systems

- Query Optimization — An Example
  - The above query can be translated into:

$$\Pi_{\text{Pnumber,Dnum,Lname,Address,Bdate}}(\sigma_{\text{Plocation="california"} \wedge \text{Dnum=Dnumber} \wedge \text{MNGSSN=SSN}}(\text{Project} \times (\text{Department} \times \text{Employee})))$$

Database Systems

- Query Optimization — An Example



Database Systems

- Query Optimization — An Example
  - The previous scenario will result in an inefficient query processing. Assume Project, Department, and Employee relations had tuples sizes of 100, 50, and 150 bytes, and contained 100, 20, and 5,000 tuples, respectively. Then the Cartesian products would generate a relation of 10 million tuples each of 300 bytes.

Database Systems

- Query Optimization — An Example
  - However, the above query based on the schemas of the relations can be translated into:

$$\Pi_{Pnumber,Dnum,Lname,Address,Bdate}(((\sigma_{Plocation=\text{“california”}}(Project)) \bowtie_{Dnum=Dnumber} (Department)) \bowtie_{MNGSSN=SSN} (Employee))$$

Database Systems

◆ Query Optimization — An Example