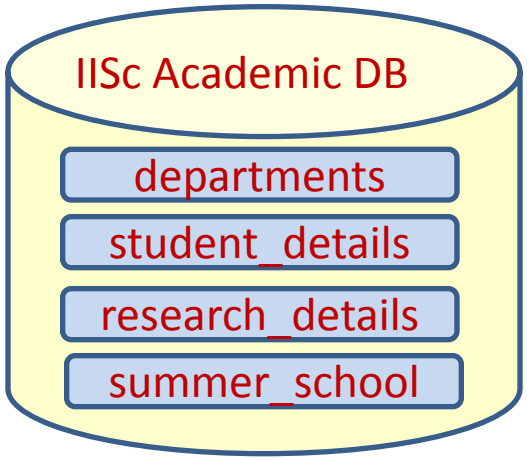
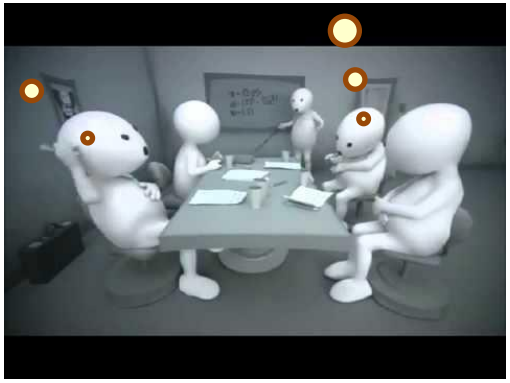



```
Select   *
From     summer_school  as SS,
         departments  as D,
         research_details  as  R,
         student_details  as  SP
Where    SS.speaker_id = D.student_id and .....
         SS.talk like '%Magic Behind%'
```

Who is speaker for this talk ?

Looks like a student here …



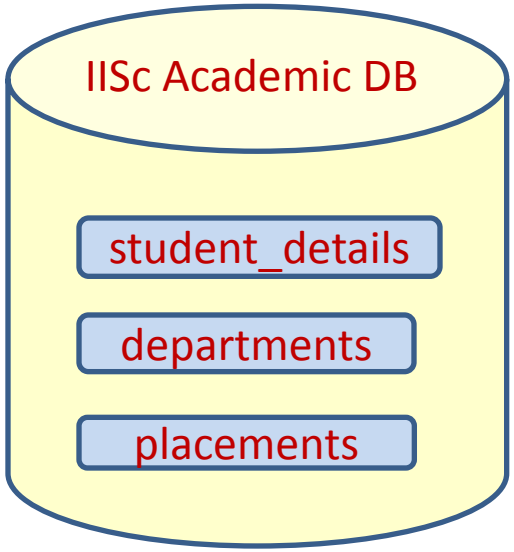
The Magic! behind declarative query processing

NAME	Program	Year	LAB	ADVISOR	RESEARCH TOPIC	GATE RANK	Permanent Address
Anshuman Dutt	Ph.D. (CSA)	2010	Database Systems Lab	Prof. Jayant Haritsa	Robust Query Processing	13	Muzaffarnagar U.P.

Querying in RDBMS

Select max(GATE rank)
From student_details
Where dept = 'CSA' and year = 2012

Aspiring Student



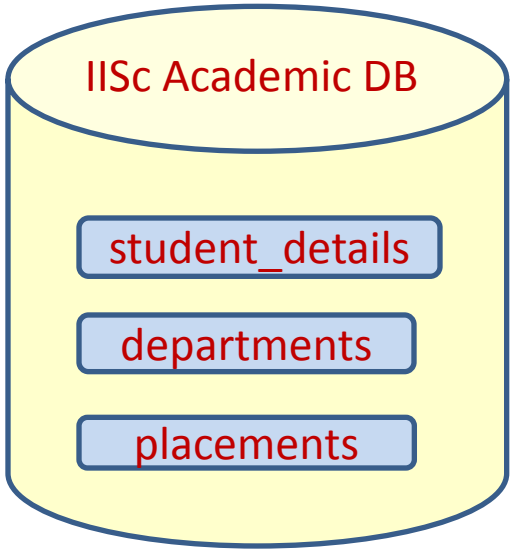
Max(GATE rank)
58

Querying in RDBMS

Declarative not Procedural

```
Select avg(salary)
From student_details S,
        placements P
Where S.Student ID  = P.Student ID
and  dept = 'CSA'
```

Aspiring Student



avg(salary)
20,00,000

The Magic / Planning

All plans give the correct answer, only difference is time taken

I know what I want (admission/query result) but the problem is how to get it

Learn and use MAGIC

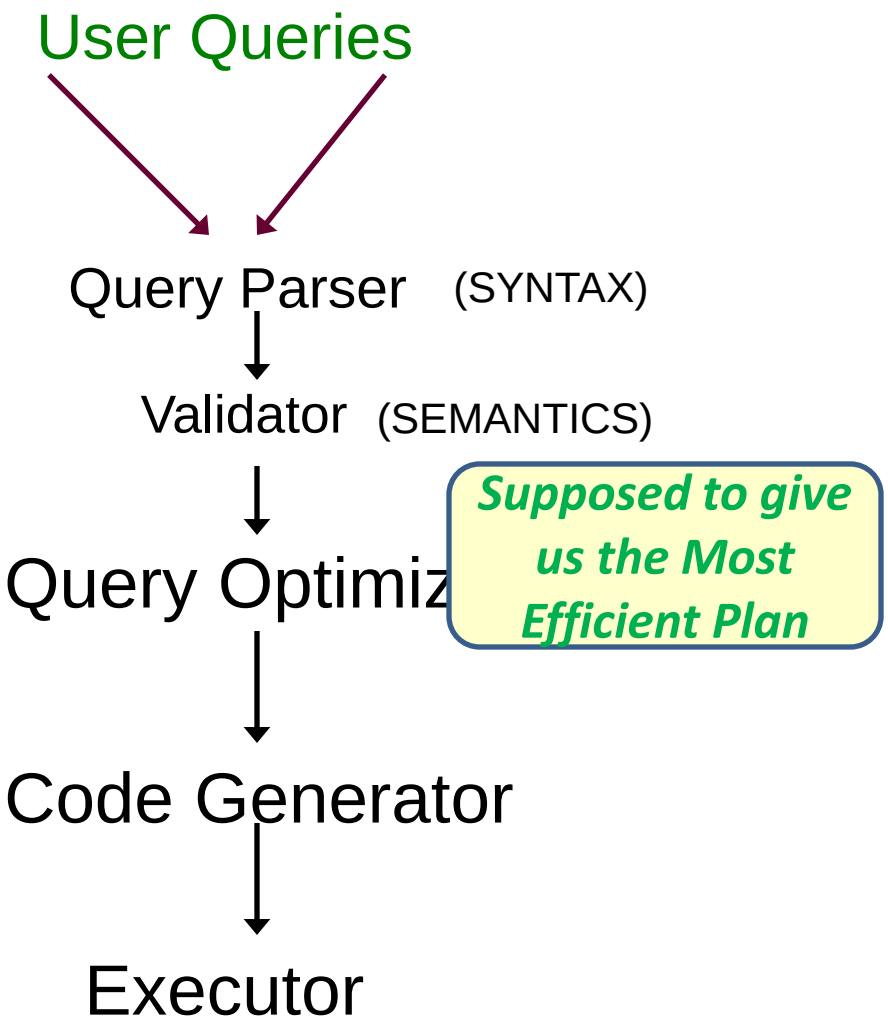
OR

Ask for professional advice

Mr. Query Optimizer

Okay, then give me the Most Efficient Plan !

Query Processing Steps



Specify WHAT we want as result

Parser - checks for spelling errors as well as the SQL keywords
Validator - checks whether the relations, attributes used in the query are VALID

Optimizer – finds HOW to execute the query efficiently to get the results

… self explanatory ☺

Query Complexity

- Various cases:
  - Single-relation query

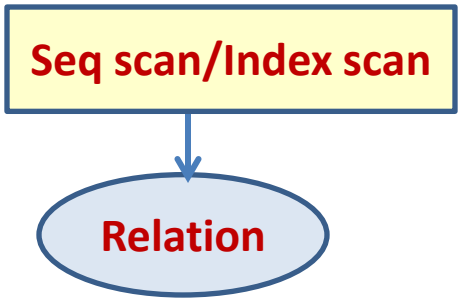
```
Select max(GATE rank)
From students_details
Where dept = 'CSA'
```

  - Multiple-relation query

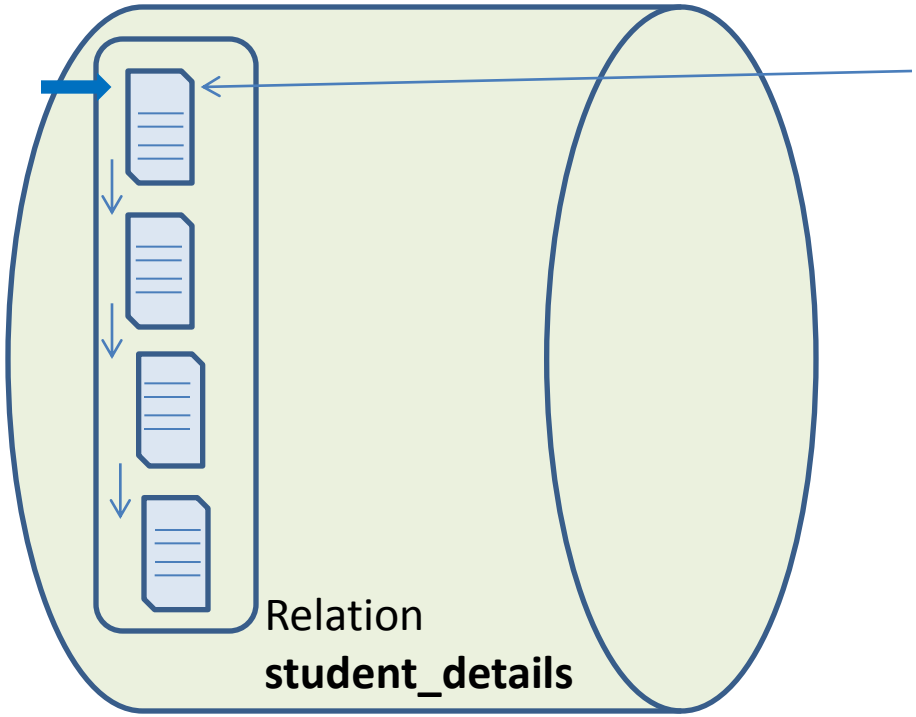
```
Select avg(salary)
From student_details S, placements P
Where S.Student ID  = P.Student ID and  dept = 'CSA'
```

Single relation : What is there to plan about ?

- For single relation queries
  - We need to read the relation from the database/stored on disk
  - Can be read in two ways
    - Scan all the tuples of the relation directly and for each tuple – do the processing (predicate evaluation)
    - Scan the relation INDIRECTLY (index knows pages having CSA tuples) and do processing of selected tuples

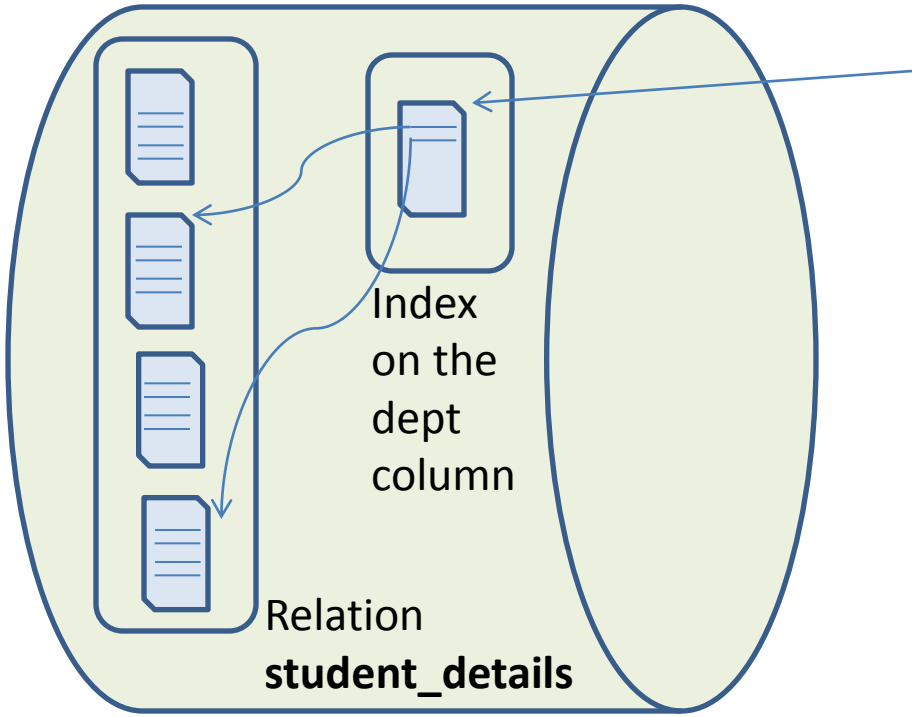


Seq Scan Operator



Select max(GATE rank)
From student_details
Where dept = 'CSA'

Index Scan Operator



Select max(GATE rank)
From student_details
Where dept = 'CSA'

Remember:* *Sequentially accessing disk pages is cheaper than random access

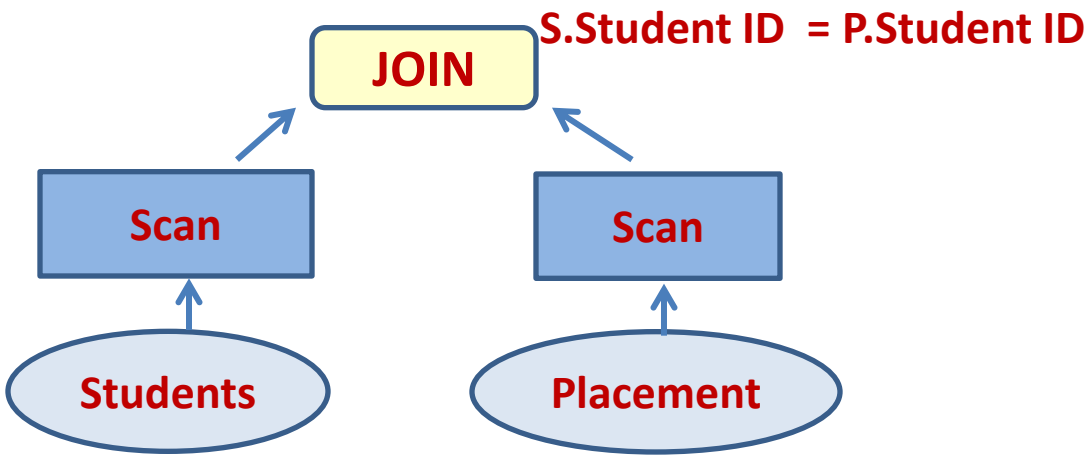
Index scan - preferable if significantly less number of tuples need to be accessed

Two-Relation Queries

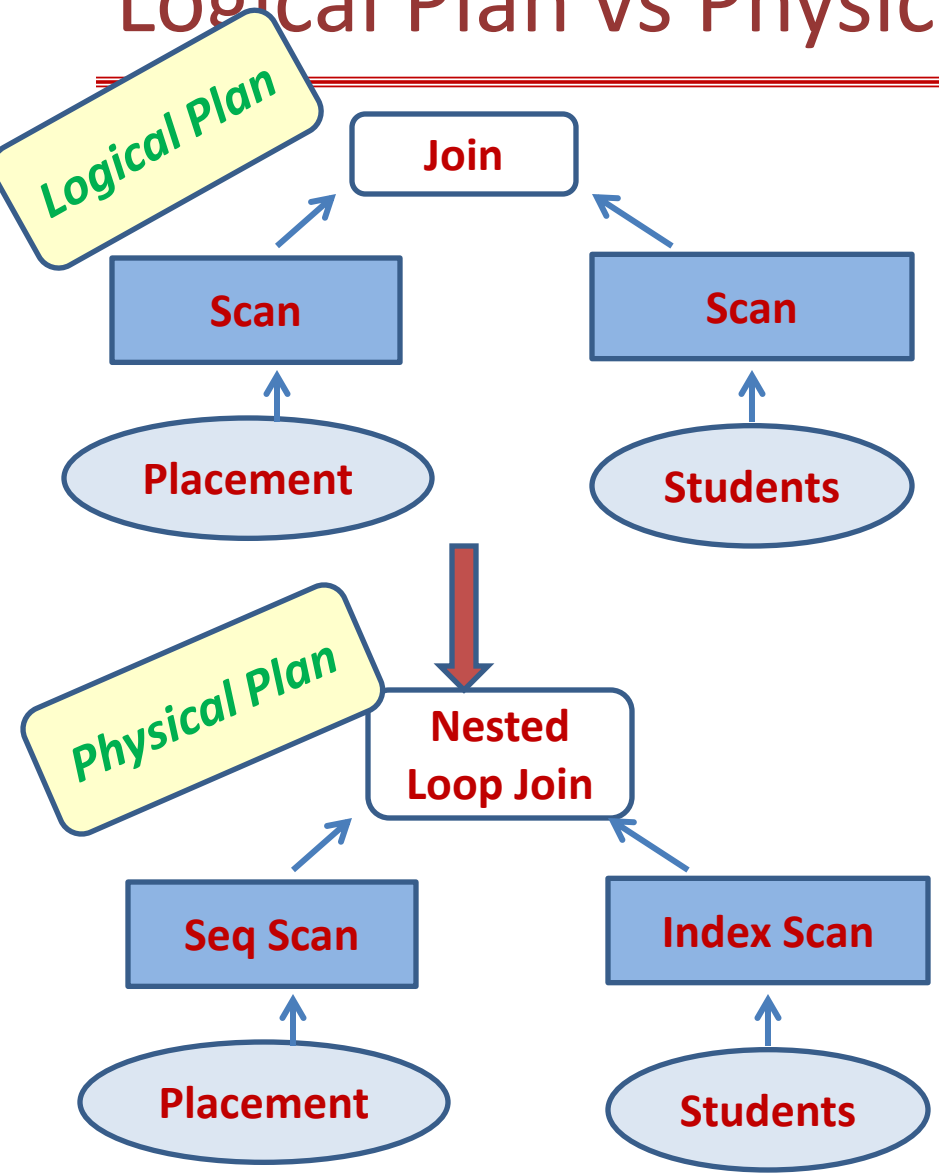
```
Select max(salary)
From Students S, Placements P
Where S.Student ID  = P.Student ID and  dept = 'CSA'
```

- Decide the scan operator for each of the relations

- Match the two sets of tuples for equality of student ID to get the answer (this operation is termed as Join)



Logical Plan vs Physical Plan



- Different ways of Scanning
  - Sequential Scan
  - Index Scan
- Different ways of joining
  - Nested Loop join
  - Sort Merge Join
  - Hash based Join

Nested Loop Join

student_details

Student_ID	Dept
1	CSA
2	EC
3	CSA
…	…
999	CSA
1000	SERC

placements

Student_ID	Salary
1000	4500000
365	5000000
2	500000
…	…
1	9000000
653	370000

Nested Loop Join

Can also utilize index on placements(student_id) (if present)

student_details

Student_ID	Dept
1	CSA
2	EC
3	CSA
…	…
999	CSA
1000	SERC

placements

Student_ID	Salary
1000	4500000
365	5000000
2	500000
…	…
1	9000000
653	370000

Sort-Merge Join

student_details

Student_ID	Dept
1	CSA
2	EC
3	CSA
…	…
999	CSA
1000	SERC

placements

Student_ID	Salary
1000	4500000
365	5000000
2	500000
…	…
1	9000000
653	370000

Sort-Merge Join

Sort the relation(s) on join column

student_details

Student_ID	Dept
1	CSA
2	EC
3	CSA
…	…
999	CSA
1000	SERC

placements

Student_ID	Salary
1	9000000
2	500000
3	9500000
…	…
999	10000000
1000	8500000

Sort-Merge Join

student_details

Student_ID	Dept
1	CSA
2	EC
3	CSA
…	…
999	CSA
1000	SERC

placements

Student_ID	Salary
1	9000000
2	500000
3	9500000
…	…
999	10000000
1000	8500000

Sort-Merge Join

student_details

Student_ID	Dept
1	CSA
2	EC
3	CSA
…	…
999	CSA
1000	SERC

placements

Student_ID	Salary
1	9000000
2	500000
3	9500000
…	…
999	10000000
1000	8500000

Hash Join

Build a Hash Table

HashKey	(Student_id, row_id)
0	(1000,1), (365, 2) ….
1	(1,999), ….
2	(2,3) …
3	(653, 1000) …
4	…

student_details

Student_ID	Dept
1	CSA
2	EC
3	CSA
…	…
999	CSA
1000	SERC

placements

Student_ID	Salary
1000	4500000
365	5000000
2	500000
…	…
1	9000000
653	370000

Hash Join

HashKey	(Student_id, row_id)
0	(1000,1), (365, 2) ….
1	(1,999), ….
2	(2,3) …
3	(653, 1000) …
4	…

Probe using key

student_details

Student_ID	Dept
1	CSA
2	EC
3	CSA
…	…
999	CSA
1000	SERC

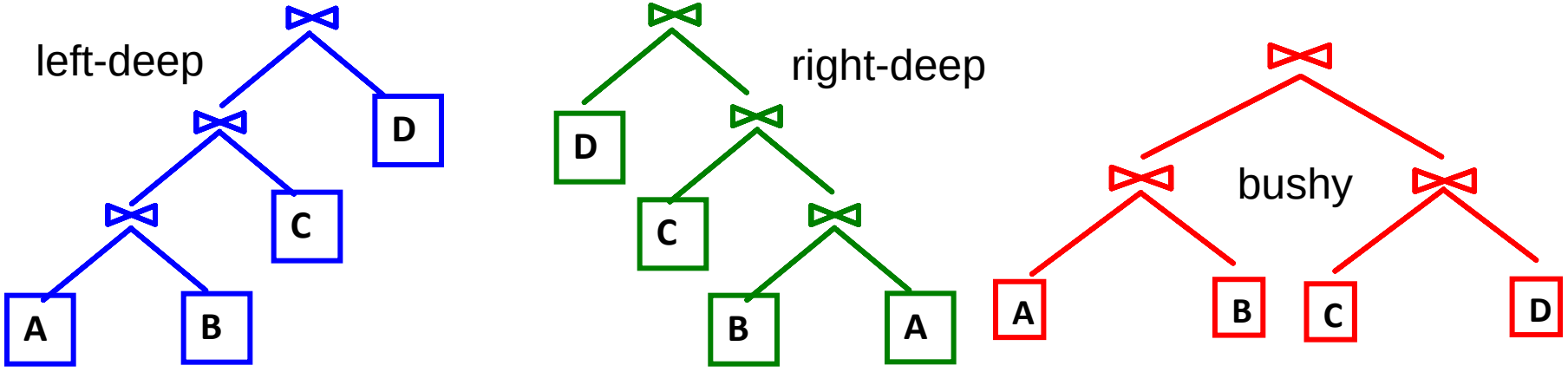
placements

Student_ID	Salary
1000	4500000
365	5000000
2	500000
…	…
1	9000000
653	370000

Multi-relation Query

- Join of Relations R_1, R_2, R_3, ..., R_n
- Decide join order (joined two at a time)

- Decide $a(r_i)$ (access method for R_j)
- Decide j_i (join method)



Queries Over Multiple Relations

- As the number of joins increases,
  - number of alternative plans grows rapidly
  - need to restrict the search space and process it efficiently.

- Once first k relations are joined, method to join the composite to $(k+1)^{st}$ relation
  - is independent of the order of joining first k

$\Rightarrow$ Dynamic Programming

Example Search Space Reduction

- Consider $r_1 \bowtie r_2 \bowtie r_3$. There are 12 different join orders:

$r_1 \bowtie (r_2 \bowtie r_3)$	$r_2 \bowtie (r_1 \bowtie r_3)$	$r_3 \bowtie (r_1 \bowtie r_2)$
$r_1 \bowtie (r_3 \bowtie r_2)$	$r_2 \bowtie (r_3 \bowtie r_1)$	$r_3 \bowtie (r_2 \bowtie r_1)$
$(r_2 \bowtie r_3) \bowtie r_1$	$(r_1 \bowtie r_3) \bowtie r_2$	$(r_1 \bowtie r_2) \bowtie r_3$
$(r_3 \bowtie r_2) \bowtie r_1$	$(r_3 \bowtie r_1) \bowtie r_2$	$(r_2 \bowtie r_1) \bowtie r_3$

Example Search Space Reduction

- To find best join order for $(r_1 \bowtie r_2 \bowtie r_3) \bowtie r_4 \bowtie r_5$,
  - 12 join orders for computing $r_1 \bowtie r_2 \bowtie r_3$,
  - 12 join orders for computing this result with r_4 and r_5.
  - there appear to be 144 join orders to examine.
  - find the best join order for the subset of relations $\{r_1, r_2, r_3\}$
  - use only that order for further joins with r_4 and r_5.
  - instead of 144 choices
  - only 12 + 12 = 24 choices.

Comment on Search Complexity

- Consider finding the best join-tree for $r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$.
- There are $(2(n-1))! / (n-1)!$ different join-trees:
  - $n = 5$, number is 1680; $n = 7$, number is 665280
  - $n = 10$, the number is greater than 176 billion!
- No need to generate all join-trees.
- Using DP, the best join order for any subset of $\{r_1, r_2, \ldots r_n\}$ is computed only once and stored for future use.
- This reduces time complexity to around $O(3^n)$
  - $n = 10$, number is ~59000.

Comparing plans (cost estimation)

- How do we know which plan is better/more efficient
  - by estimating the amount of work to be done

- For each plan considered:
  - estimate cost of each operation in plan tree and take the summation
  - Cost = w_1 * Page Fetches from disk (I/O work)
    + w_2 * # of tuples processed by CPU (processing work)
  - Need to estimate size of result (cardinality) for each operation in tree.
  - # tuples (NTuples) and # pages (NPages) for each relation.
  - # pages (NPages) for each index.

Cardinality Estimation

- Consider a query:

```
SELECT  attribute list
FROM  relation list
WHERE  term1 AND ... AND termk
```

- Maximum # tuples in result is the product of the cardinalities of relations in the FROM clause.

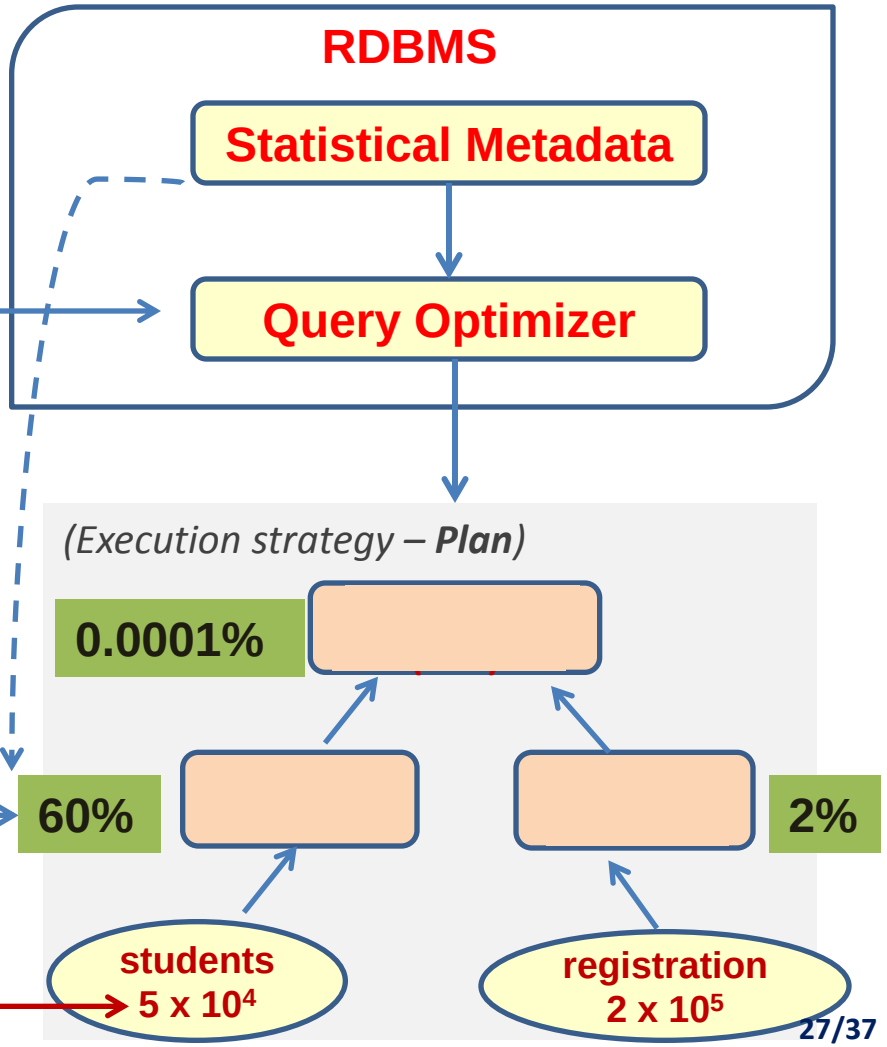
- Reduction factor (RF) associated with each term reflects the impact of the term in reducing cardinality from underlying relations.

Big Picture

(SQL version)

```
select   count(S.student_id)
from     registration R, students S
where    R.student_id = S. student_id
  and    R.cname IN ('DBMS','OS', …)
  and    S.program IN ('Ph.D.', 'M.S')
```

Number of research students in CSA systems courses



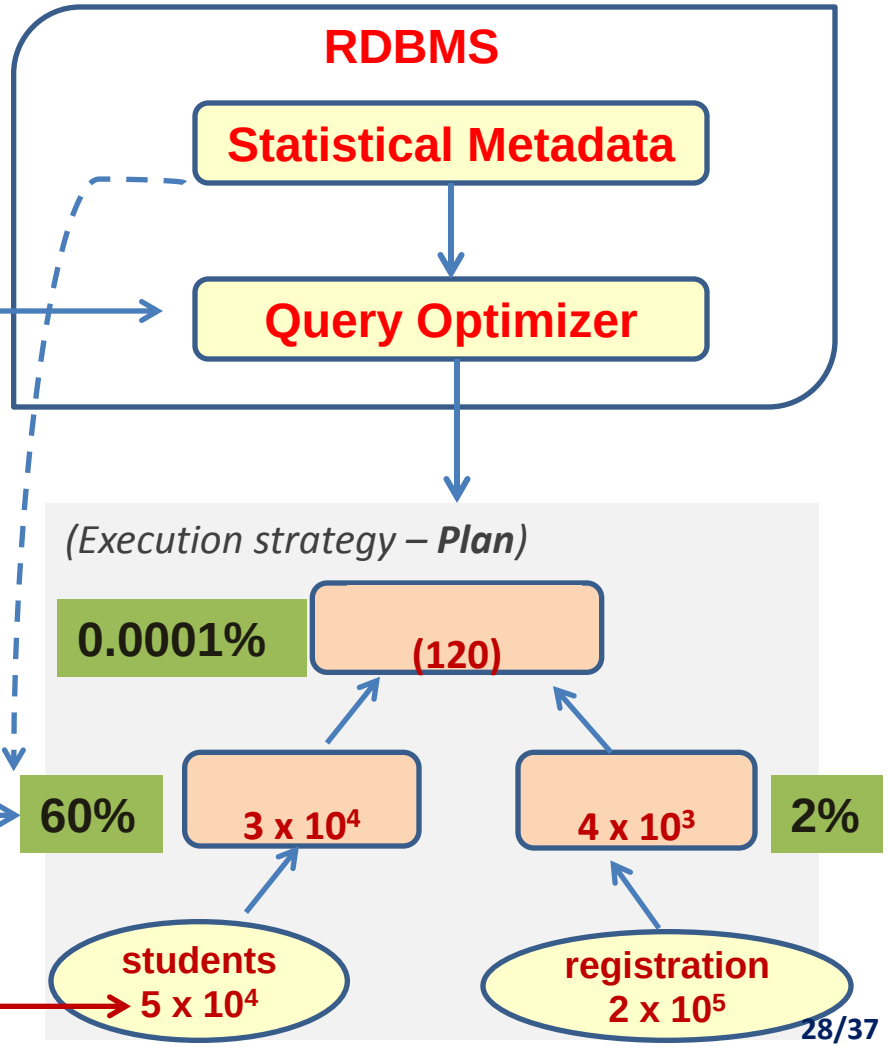
Reduction Factor:
# of rows satisfying the predicate

Big Picture

(SQL version)

```
select   count(S.student_id)
from     registration R, students S
where    R.student_id = S. student_id
  and    R.cname IN ('DBMS','OS', …)
  and    S.program IN ('Ph.D.', 'M.S')
```

Number of research students in CSA systems courses



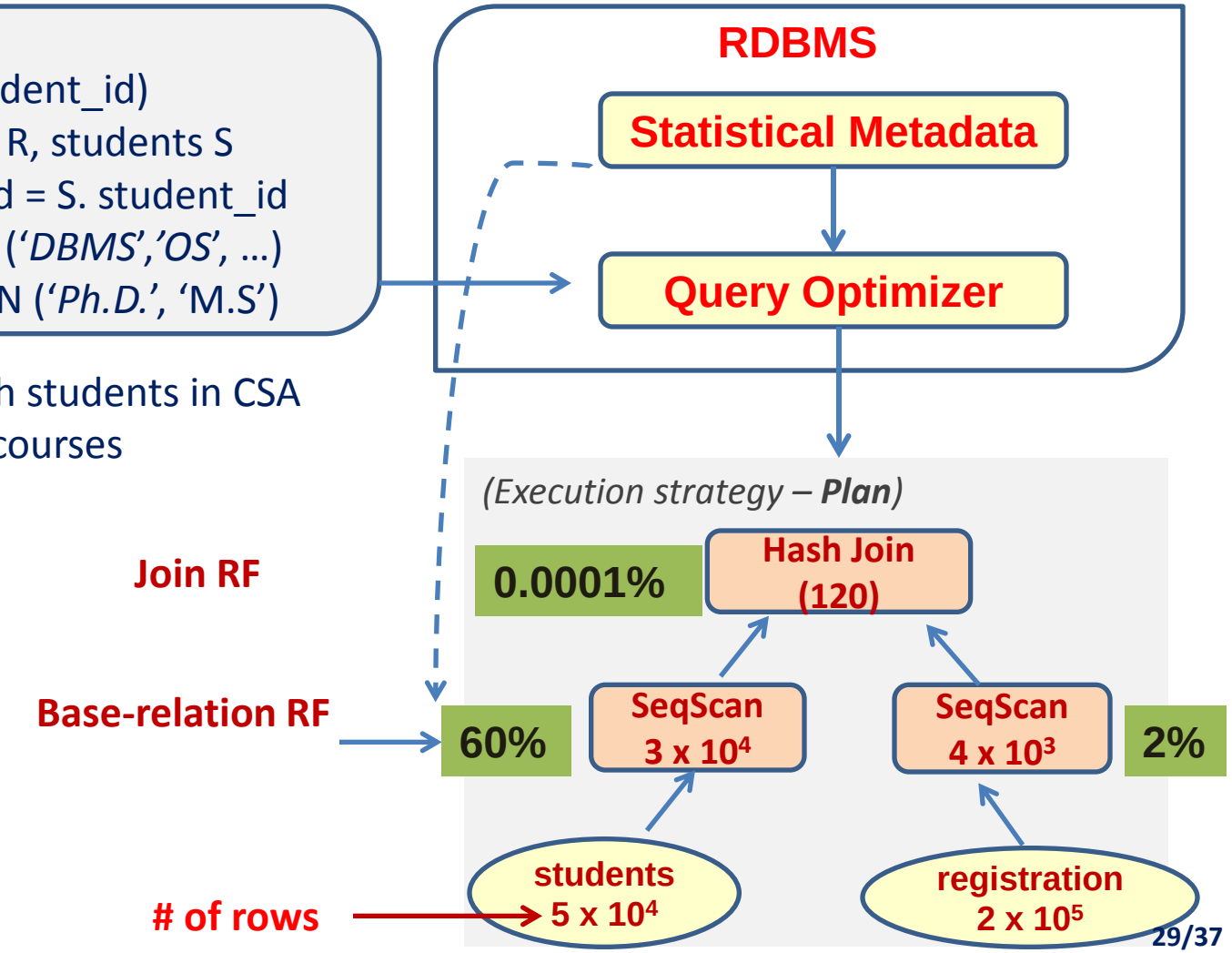
Reduction Factor:
# of rows satisfying the predicate

Big Picture

(SQL version)

```
select   count(S.student_id)
from     registration R, students S
where    R.student_id = S. student_id
  and    R.cname IN ('DBMS','OS', …)
  and    S.program IN ('Ph.D.', 'M.S')
```

Number of research students in CSA systems courses



Reduction Factor:
# of rows satisfying the predicate

Issue 1: Plan Space Construction

- Operator Algorithms:
  - Access: Sequential, Index
  - Join: Nested Loop, Sort-merge, Hash based

- Example (recommendation of first paper)
  - Only left-deep plans are considered
    - Left-deep plans allow output of each operator to be pipelined into the next operator without storing it in a temporary relation.
  - Cartesian products are avoided

Issue 2: Search/Enumeration Algorithm

- Dynamic Programming (DP)
  - practical for $n \leq 10$
- Other choices are
  - Top down/transformation based algorithm
  - Randomized algorithm
  - Sampling based algorithm

Issue 3: Cost Modeling

- Weighted combination of CPU and I/O costs
  - disk resident database
  - Memory resident database

- Simple counts of relations and indexes maintained in system catalogs
  - Wide variety of metadata structures have been proposed
  - No easy way for intermediate cardinality estimation

Some more Issues

- Can we exploit parallelization during query optimization ?
- Multi-query optimization
  - Simultaneous optimization of multiple queries
- Adaptive/Robust Query Processing
  - estimated RF values are often far from actual values

Design Framework

- Issues
  - For a given query, what plans are considered (i.e. plan space)?
  - How to efficiently search through plan space for “cheapest” plan?
  - How is the cost of a plan estimated (i.e. cost model, statistics)?
  - Parallelization, Multi-query optimization, Robust Query Processing …

Few selected papers in Query-Optimization

- 1979: Access Path Selection in RDBMS
- 1984: Implementation techniques for main memory DB
- 1988: Multiple query optimization
- 1989: Optimization of large join queries (randomization algos)
- 1990: Measuring Complexity of Join enumeration
- 1991: Analysis of Strategy spaces and its implications(Left Deep vs Bushy)

Few selected papers in Query-Optimization

- 1991: On propagation of errors in join result sizes
- 1994: Dynamic query evaluation plans
- 1997: Parametric query Optimization
- 1998: Efficient mid-query re-optimization
- 2002: Least Expected Cost Optimization
- 2004: Robust Query processing through Progressive Optimization

Few selected papers in Query-Optimization

2002: Plan Selection based on Query Clustering

- 2005: Proactive re-optimization

2005: Analyzing Plan Diagrams of Database Query Optimizers

- 2007: Adaptive Join reordering at run-time
- 2007: Optimal Top-down join enumeration

2007: On the Production of Anorexic Plan Diagrams

- 2008: Dynamic Programming Strikes back

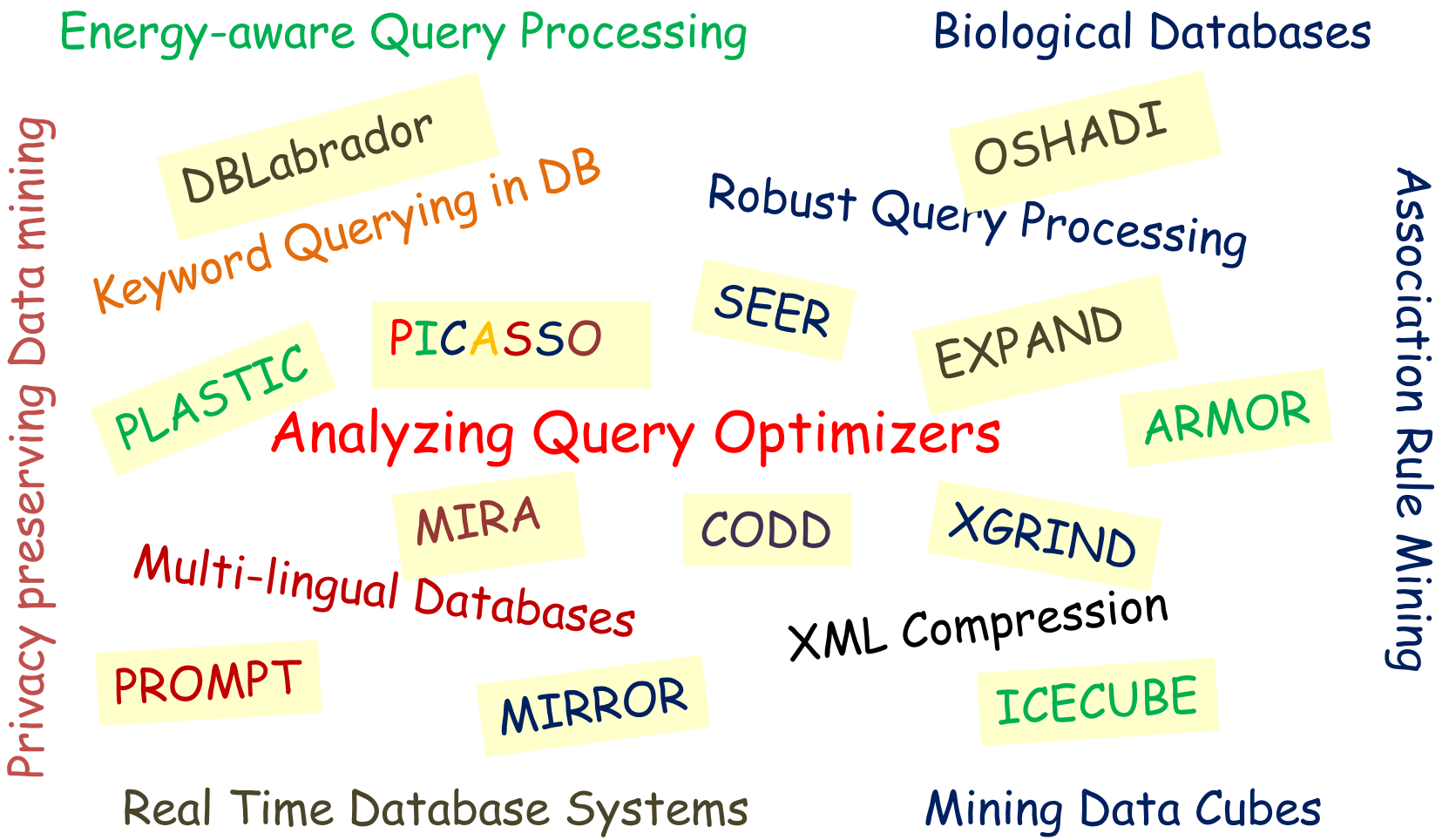
2008: Identifying Robust plans through Plan-diagram reduction

- 2008: Parallelizing Query Optimization
- 2009: Query Opt- time to rethink the contract

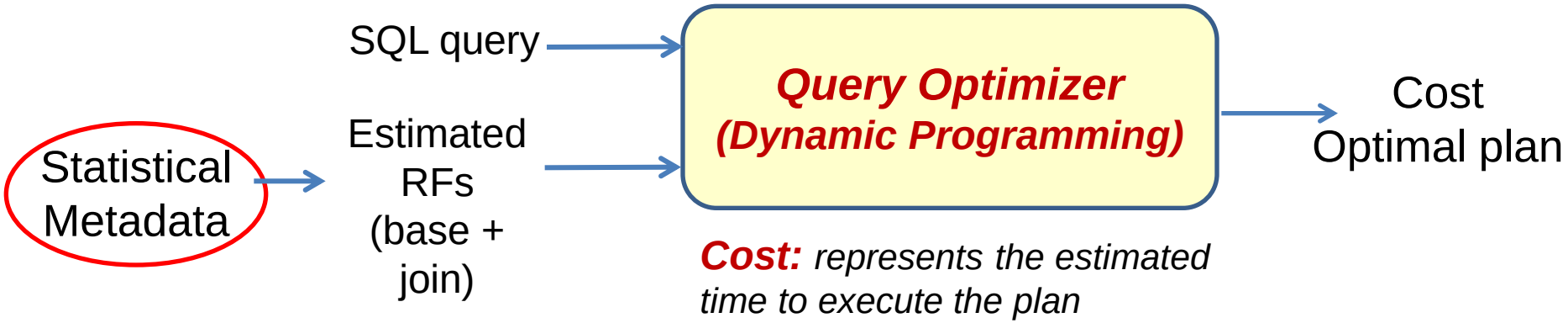
2010: On the stability of Plan costs and cost of plan stability
2010: The Picasso Database Query Optimizer Analyzer
2012: Peak Power Plays in Databases

DSL Contributions

DSL – Research Projects



Robust Query Processing



- Plan-choice may turn out to be ***very-inefficient***
  - Due to Estimation Errors:
    - outdated statistics
    - attribute-value independence(AVI) assumption
    - insufficient data distribution details
    - multiplication of errors

Long standing problem (> 30 years)

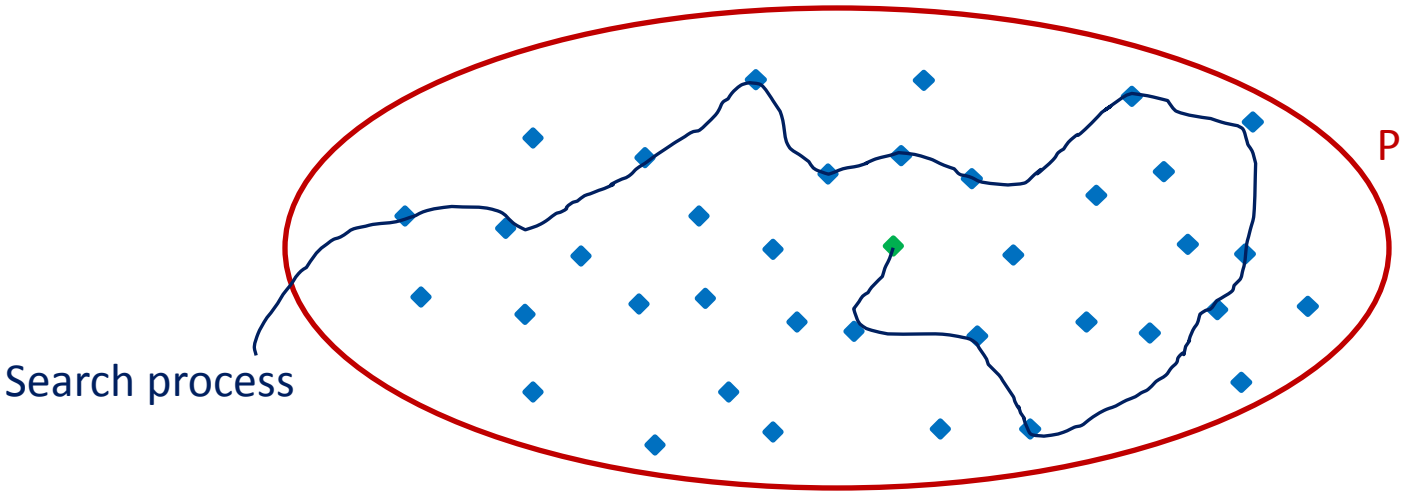
Plan Bouquet: Query Processing without Estimation (submitted to VLDB)

Research in Databases

- Fine combination of Theory and Systems
  - Simulation
  - Operating System
  - Real time system
  - Computer Networks
  - Data mining techniques
  - Clustering techniques

  - **Linear Algebra (Matrix theory)**
  - Probability, Probabilistic Graphical models
  - **Randomized algorithms, Online algorithms**
  - Statistical Inference
  - Algebraic structures

PODS, ICDT …



Thanks for your attention
Queries ??