datalab

data is everywhere, value is hidden

Relational Databases

Lecturer: Азат Якупов (Azat Yakupov)
https://datalaboratory.one

```
SELECT *
  FROM document
ORDER BY id
  OFFSET 1000000
   LIMIT 5
```

```
WITH _doc AS (
        SELECT id
          FROM document
      ORDER BY id
        OFFSET 999999
         LIMIT 1)
  SELECT *
    FROM document
   WHERE id > (SELECT id
                 FROM _doc)
ORDER BY id
   LIMIT 5
```

Student

Name	ID*
Ivan	1
Anna	2
Peter	3
Kate	4
…	…

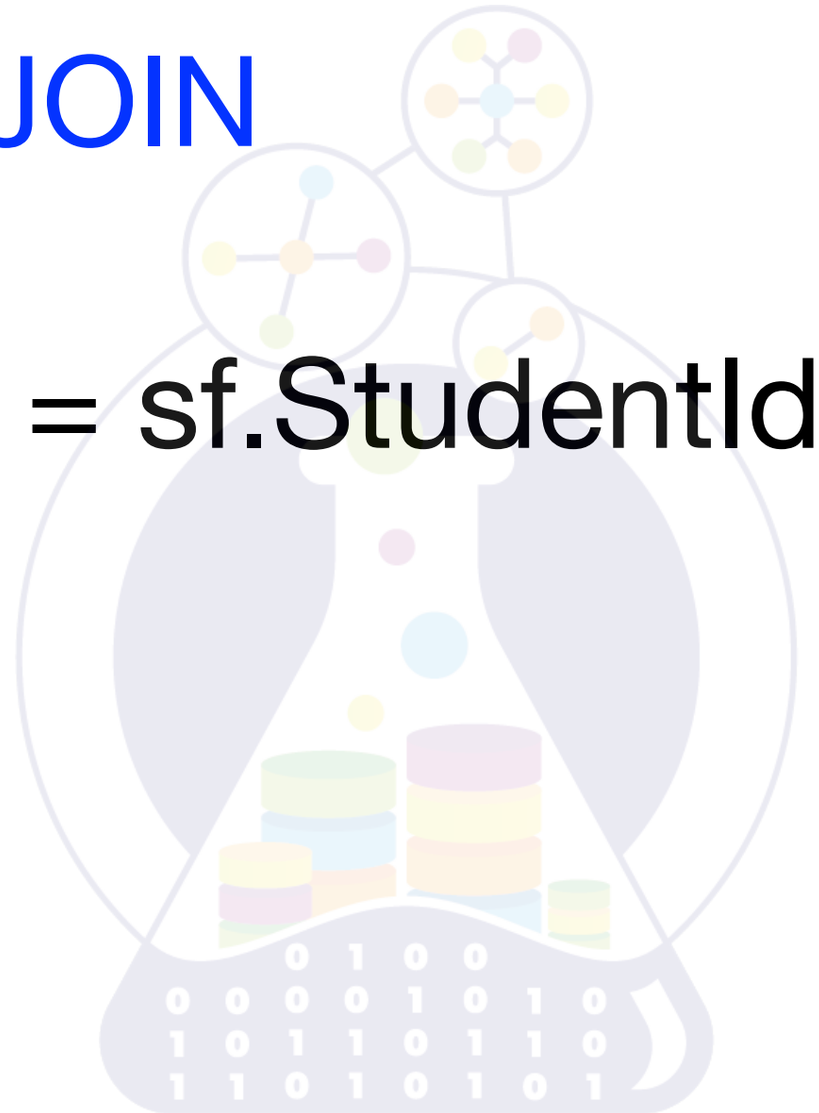
StudentFun

StudentID*	FunID*
1	1
1	3
2	1
2	2
4	3
…	…

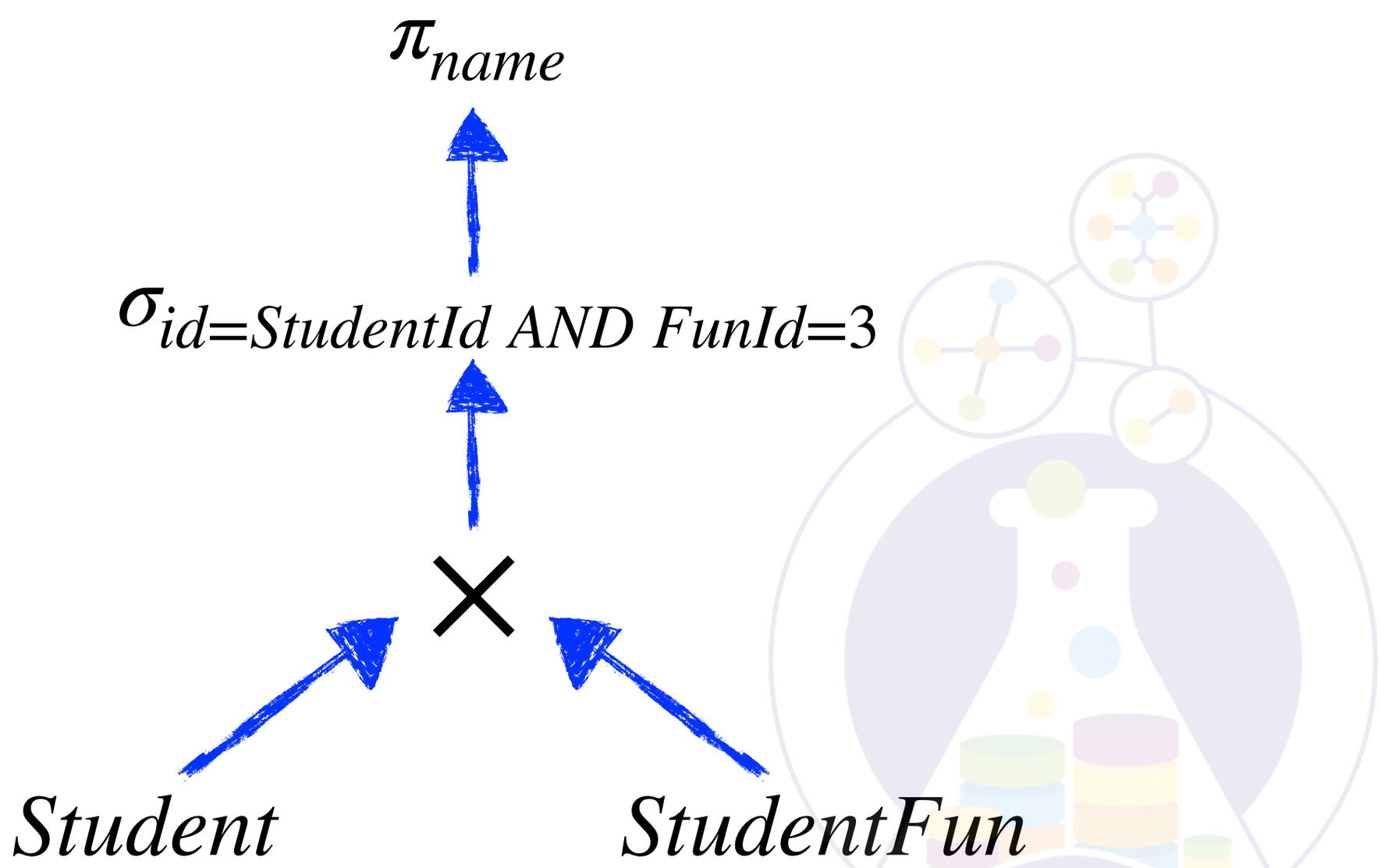
Fun

Name	ID*
Party	1
Disco	2
Library	3

```
SELECT s.name
  FROM Student s INNER JOIN
       StudentFun sf
            ON s.id = sf.StudentId
WHERE sf.FunId = 3
```



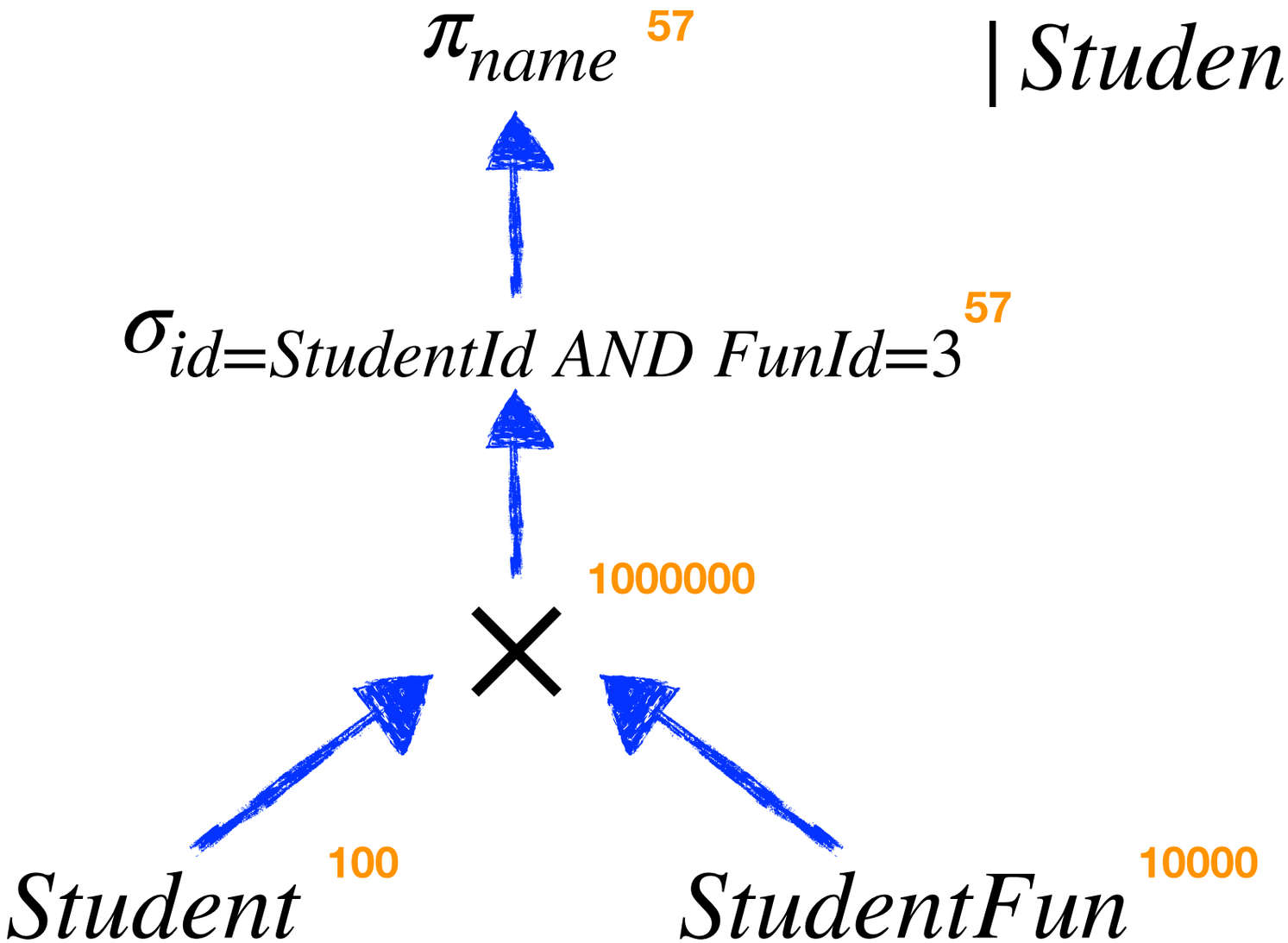
$$R = \pi_{name}\, \sigma_{id=StudentId\ AND\ FunId=3}(Student \times StudentFun)$$



$$|Student| = 100$$

$$|StudentFun| = 10000$$

$$|StudentFun|_{FunId=3} = 57$$



Student	
ID	BIGINT
NAME	CHAR(10)

= 8 *bytes*

= 80 *bytes*

StudentFun	
StudentID	BIGINT
FunID	BIGINT

= 8 *bytes*

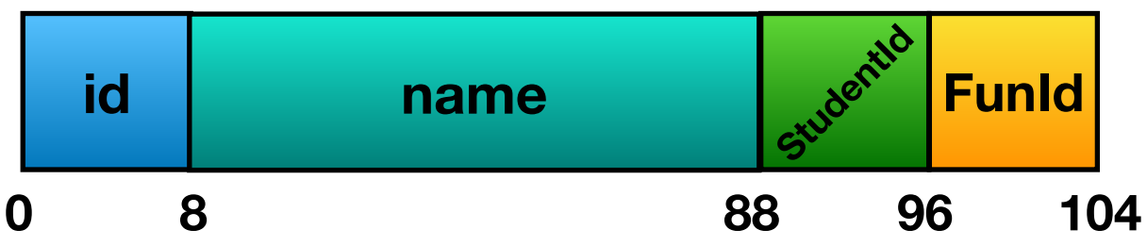
= 8 *bytes*

$8 + 80 = 88$ *bytes*

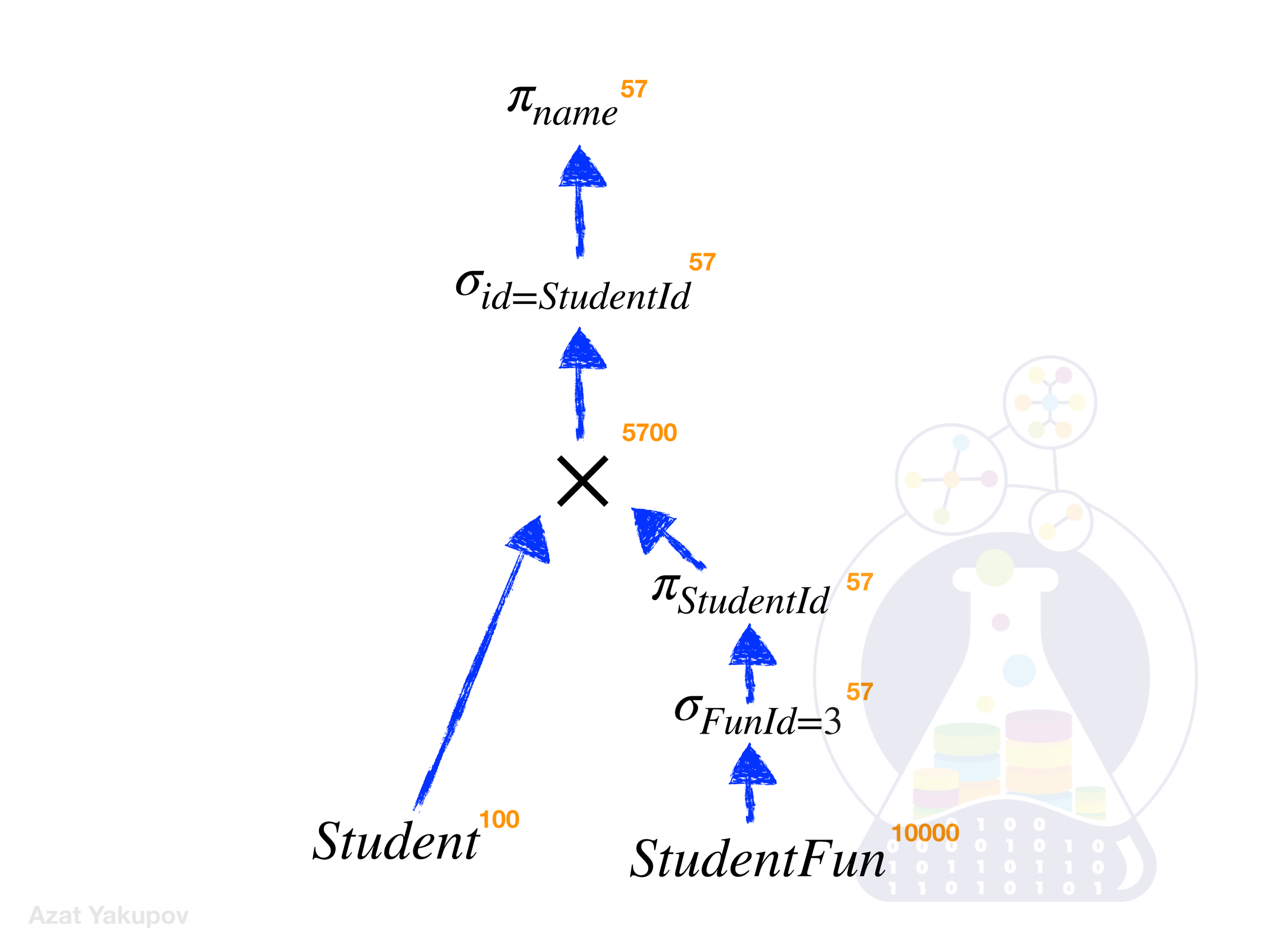
$8 + 8 = 16$ *bytes*

104 *bytes*

1000000



Total memory = 99 Mb



Student	
ID	BIGINT
NAME	CHAR(10)

$= 8$ *bytes*

$= 80$ *bytes*

StudentFun	
StudentID	BIGINT

$= 8$ *bytes*

$8 + 80 = 88$ *bytes*

8 *bytes*

$\times$ 5700

96 *bytes*

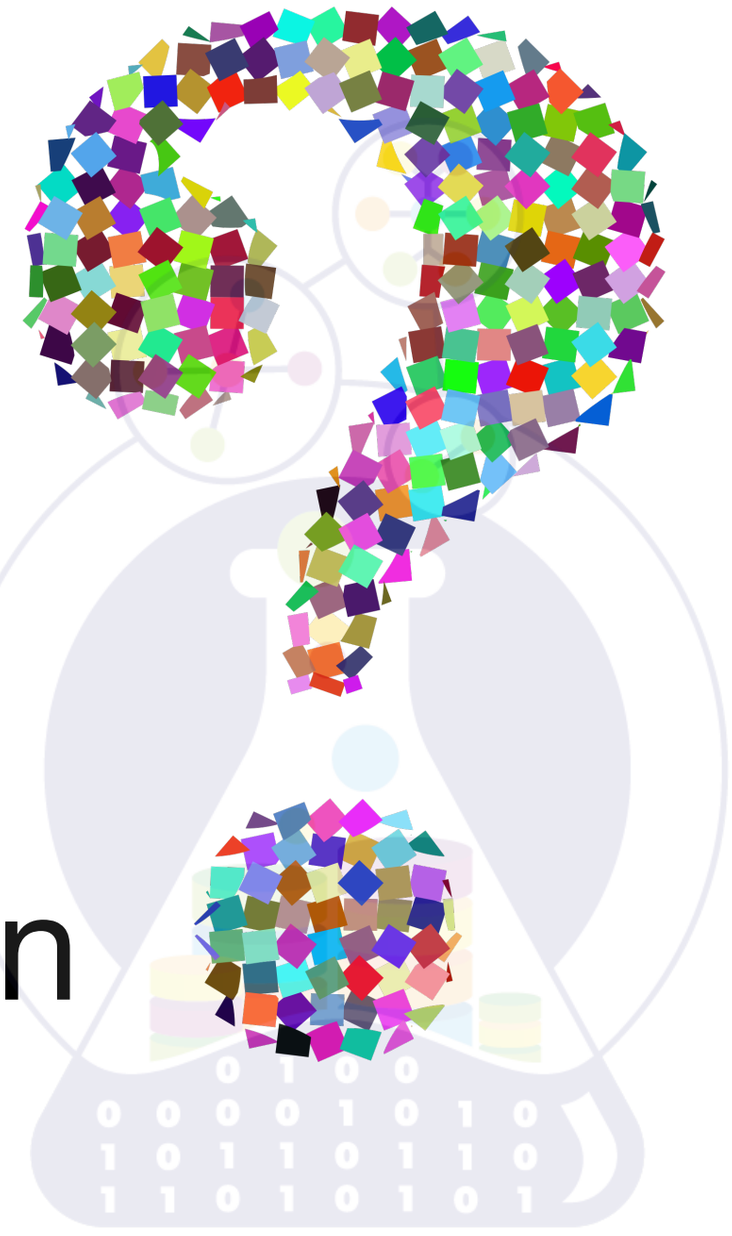
Total memory $= 0.52$ *Mb*

```
SELECT s.name
    FROM Student s INNER JOIN
            (SELECT StudentId
                FROM StudentFun sf
            WHERE sf.FunId = 3) sf
                ON s.id = sf.StudentId
```

$$|Student| = 100$$

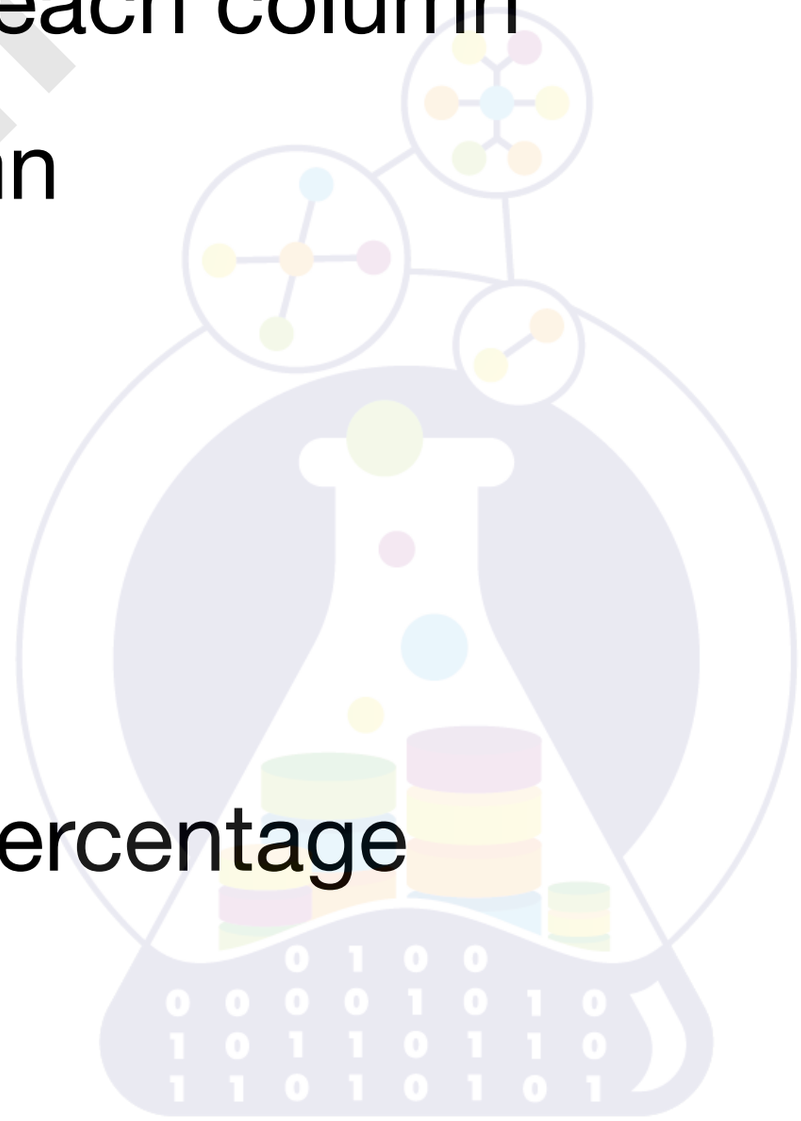
$$|StudentFun| = 10000$$

$$|StudentFun|_{FunId=3} = 57$$

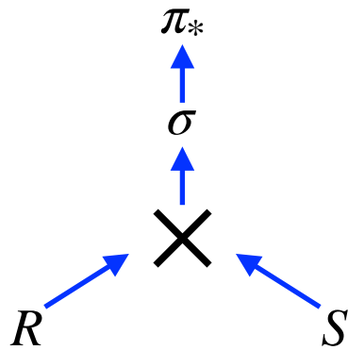


What do values mean

- Amount of rows of table
- % of *NULL* values for each column
- Amount of distinct values for each column
- Average length of each column
- Frequency column values
- Correlation
- Data Histogram
- Functional dependencies in percentage

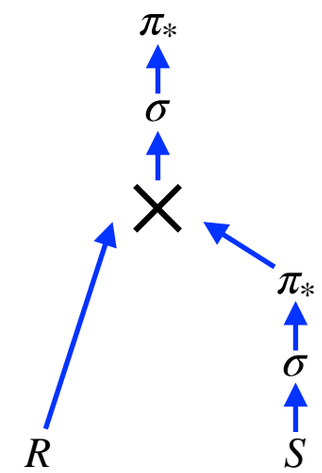


I



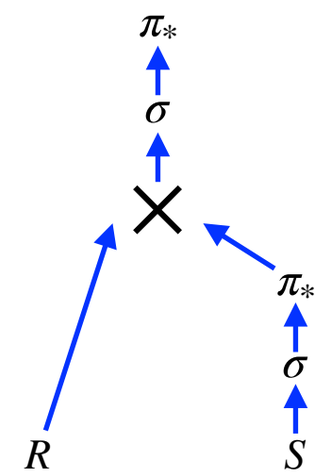
Parse Tree

II



Query Tree

III



Plan Tree

Initial SQL query

Syntax analysis, Translation

Compiled SQL query

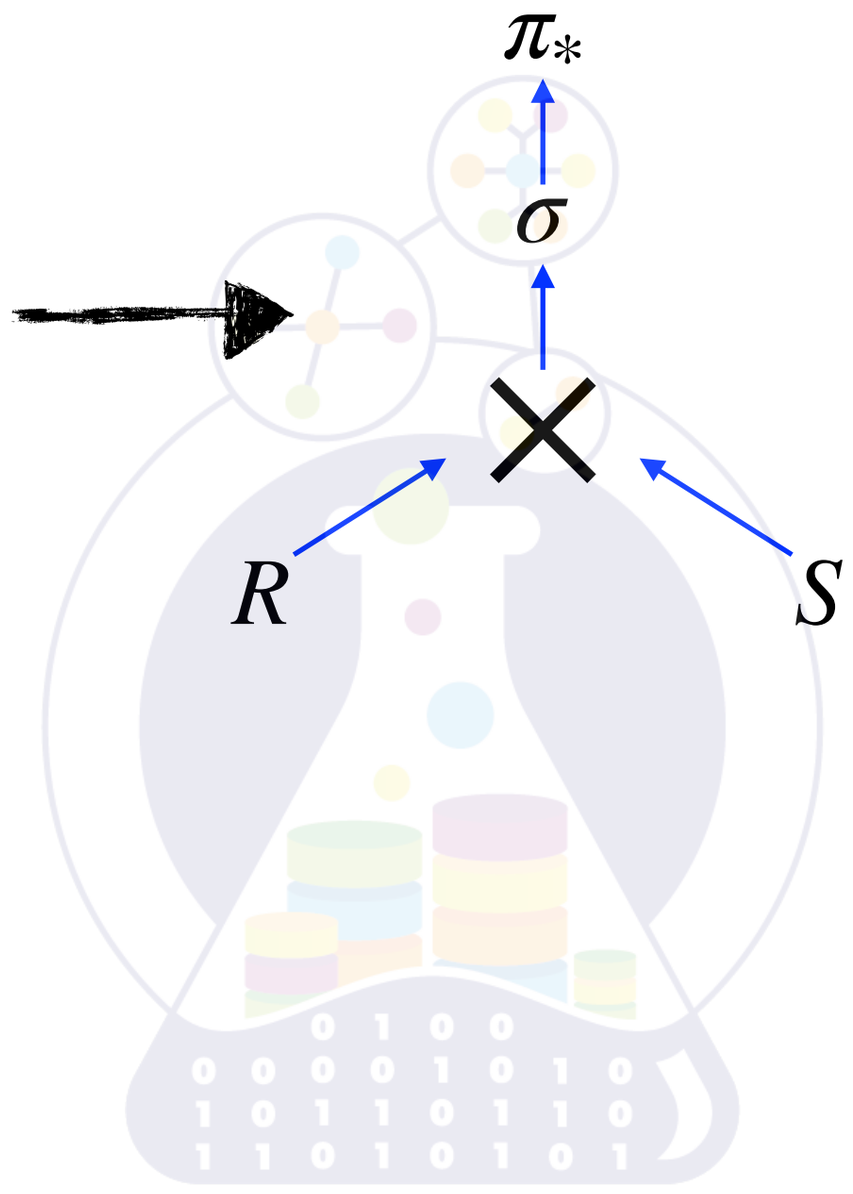
Transformation, Getting “cost”

Planned SQL query

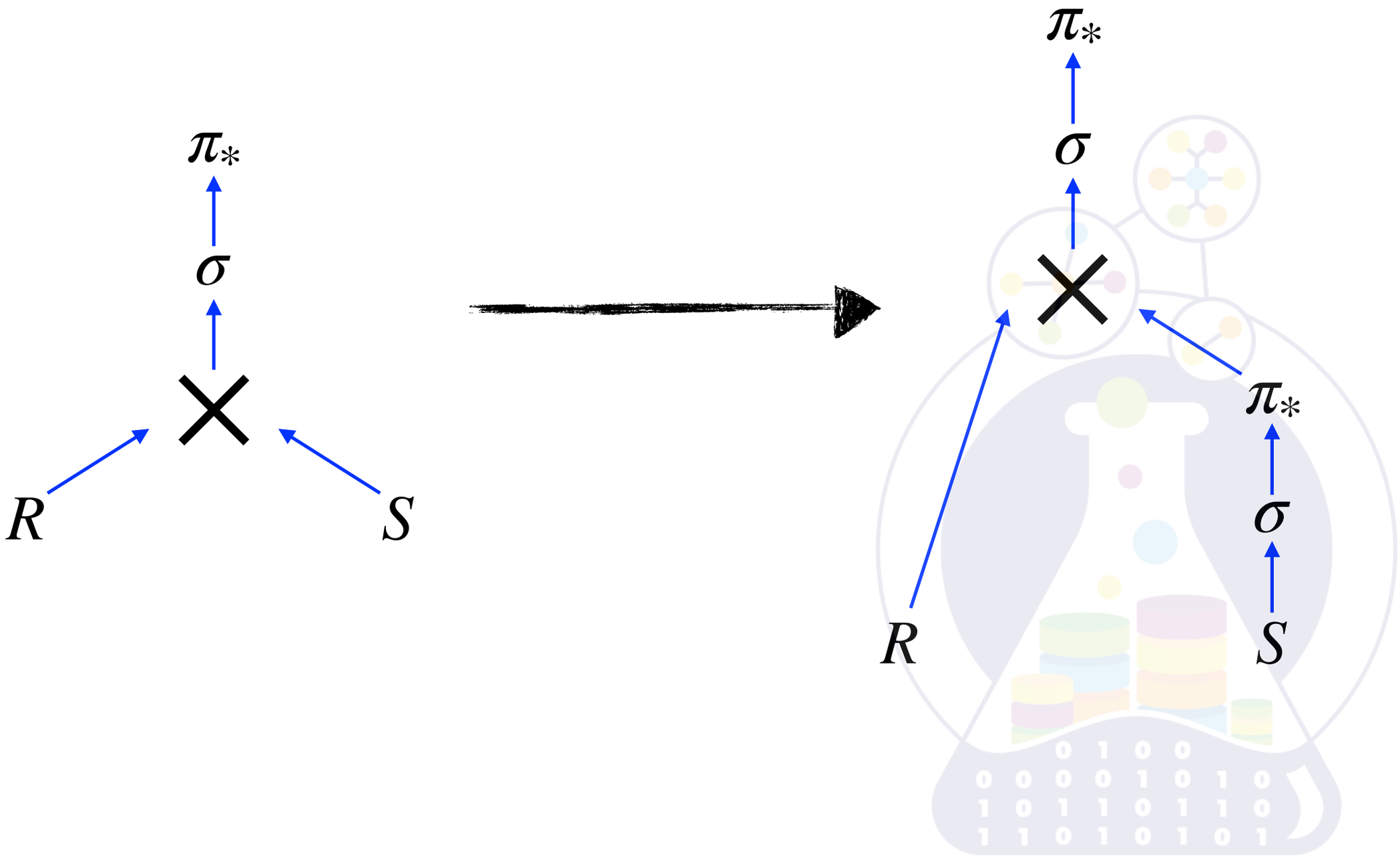
Execution

Parse Tree

```
SELECT *
  FROM R INNER JOIN S
    ON R.ID = S.R_ID
```



Query Tree



Reduction Law

$$(A\ WHERE\ p1)\ WHERE\ p2 \quad \sim \quad A\ WHERE\ p1\ AND\ p2$$

```
SELECT *
   FROM (SELECT *
            FROM Student
           WHERE ID=1) t0
WHERE t0.Name = ‘Kate’
```

```
SELECT *
   FROM Student
WHERE ID=1 AND
      Name = ‘Kate’
```

Projection Law

$$(A\ \{acl1\})\ \{acl2\} \quad \sim \quad A\ \{acl2\}$$

```
SELECT name
    FROM (SELECT id, name
                FROM Student
    WHERE ID=1) t0
```

```
SELECT name
    FROM Student
WHERE ID=1
```

Reduction Projection Law

$$(A\ \{acl1\})\ \ WHERE\ p \quad \sim \quad (A\ WHERE\ p)\ \{acl1\}$$

```
SELECT name
    FROM (SELECT id, name
            FROM Student) t0
WHERE t0.id=1
```

```
SELECT name
    FROM Student
WHERE id=1
```


Azat Yakupov

Distribution Law

$$f(A \circ B) \equiv f(A) \circ f(B)$$

$$(A\ UNION\ B)\{acl\} \equiv A\{acl\} UNION\ B\{acl\}$$

$$(A\ INTERSECT\ B)\{acl\} \equiv A\{acl\} INTERSECT\ B\{acl\}$$

$$(A\ JOIN\ B)\{acl\} \equiv A\{acl1\} JOIN\ B\{acl2\}$$

Distribution Law

$$f(A \circ B) \equiv f(A) \circ f(B)$$

```
SELECT s.name,
       sf.StudentId
  FROM Student s INNER JOIN
       StudentFun sf
    ON s.id=sf.StudentId
```

```
SELECT *
  FROM (SELECT id
          FROM Student s) s
INNER JOIN
(SELECT StudentId
   FROM StudentFun sf) sf
ON s.id=sf.StudentId
```

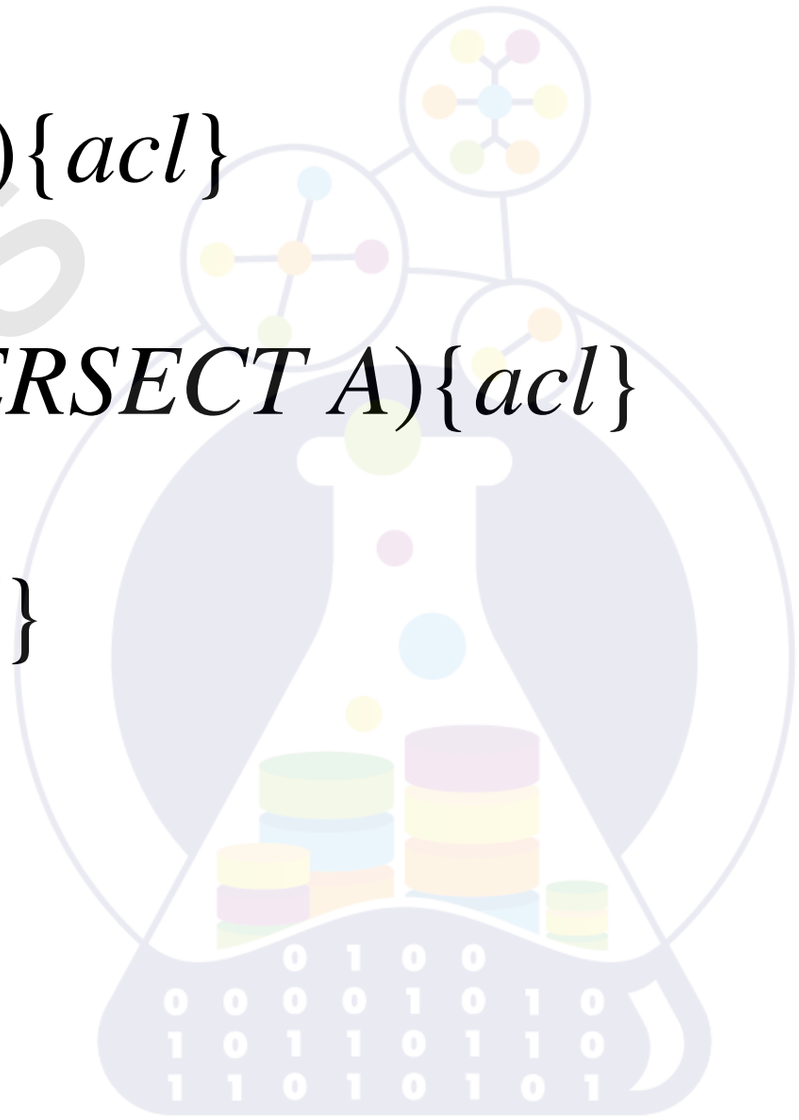
Commutative Law

$$A \circ B \equiv B \circ A$$

$$(A\ UNION\ B)\{acl\} \equiv (B\ UNION\ A)\{acl\}$$

$$(A\ INTERSECT\ B)\{acl\} \equiv (B\ INTERSECT\ A)\{acl\}$$

$$(A\ JOIN\ B)\{acl\} \equiv (B\ JOIN\ A)\{acl\}$$



Commutative Law

$$A \circ B \equiv B \circ A$$

```
SELECT s.name,
       sf.StudentId
  FROM Student s INNER JOIN
       StudentFun sf
    ON s.id=sf.StudentId
```

```
SELECT s.name,
       sf.StudentId
  FROM StudentFun sf INNER JOIN
       Student s
    ON s.id=sf.StudentId
```

Associative Law

$$A \circ (B \circ C) \equiv (A \circ B) \circ C$$

$$(A\ UNION\ (B\ UNION\ C))\{acl\} \equiv (A\ UNION\ B)\ UNION\ C))\{acl\}$$

$$(A\ INTERSECT\ (B\ INTERSECT\ C))\{acl\} \equiv$$

$$((A\ INTERSECT\ B)\ INTERSECT\ C))\{acl\}$$

$$(A\ JOIN\ (B\ JOIN\ C))\{acl\} \equiv ((A\ JOIN\ B)\ JOIN\ C)\{acl\}$$

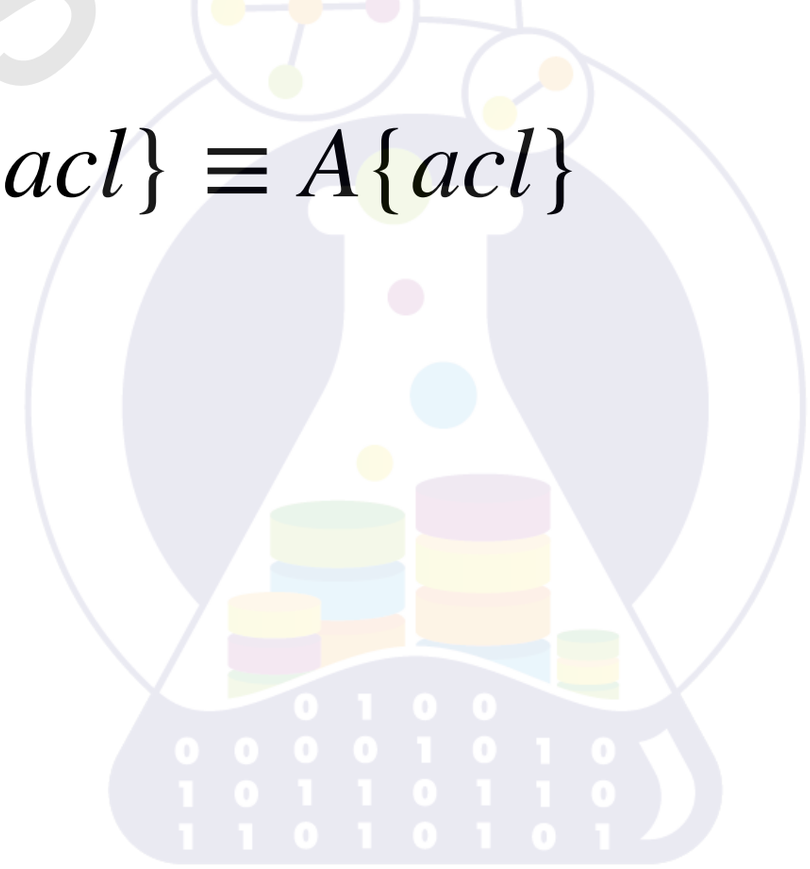
Azat Yakupov

Idempotent Law

$$A \circ A \equiv A$$

$$(A\ UNION\ (A\ INTERSECT\ B))\{acl\} \equiv A\{acl\}$$

$$A\ INTERSECT\ (A\ UNION\ B)\{acl\} \equiv A\{acl\}$$



Expression Transformation

- $A \ \delta \ (B \circ C) \equiv (A \ \delta \ B) \circ (A \ \delta \ C)$

- $A > B \ AND \ B > 3 \quad \sim$

$$A > B \ AND \ B > 3 \ AND \ A > 3$$

- $A > B \ OR \ (C = D \ AND \ E < F) \quad \sim$

$$(A > B \ OR \ C = D) \ AND \ (A > B \ OR \ E < F)$$

```
SELECT *
  FROM emp
 WHERE salary > 40000
```

≠

```
SELECT *
  FROM emp
 WHERE salary > 40000
   AND job = 'manager'
```

syntactically nonequivalent

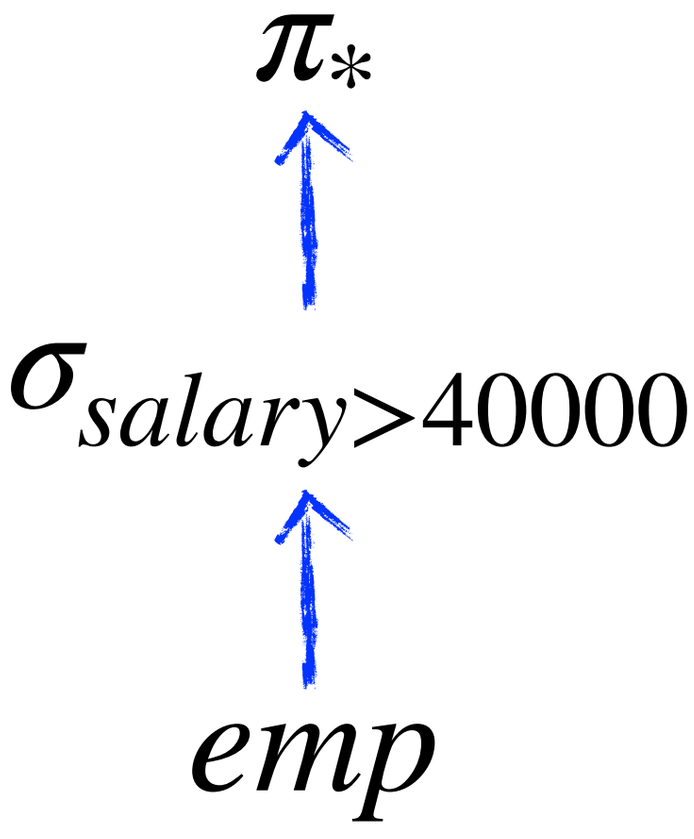
emp	
Name	**ID***
Ivan	1
Anna	2

=

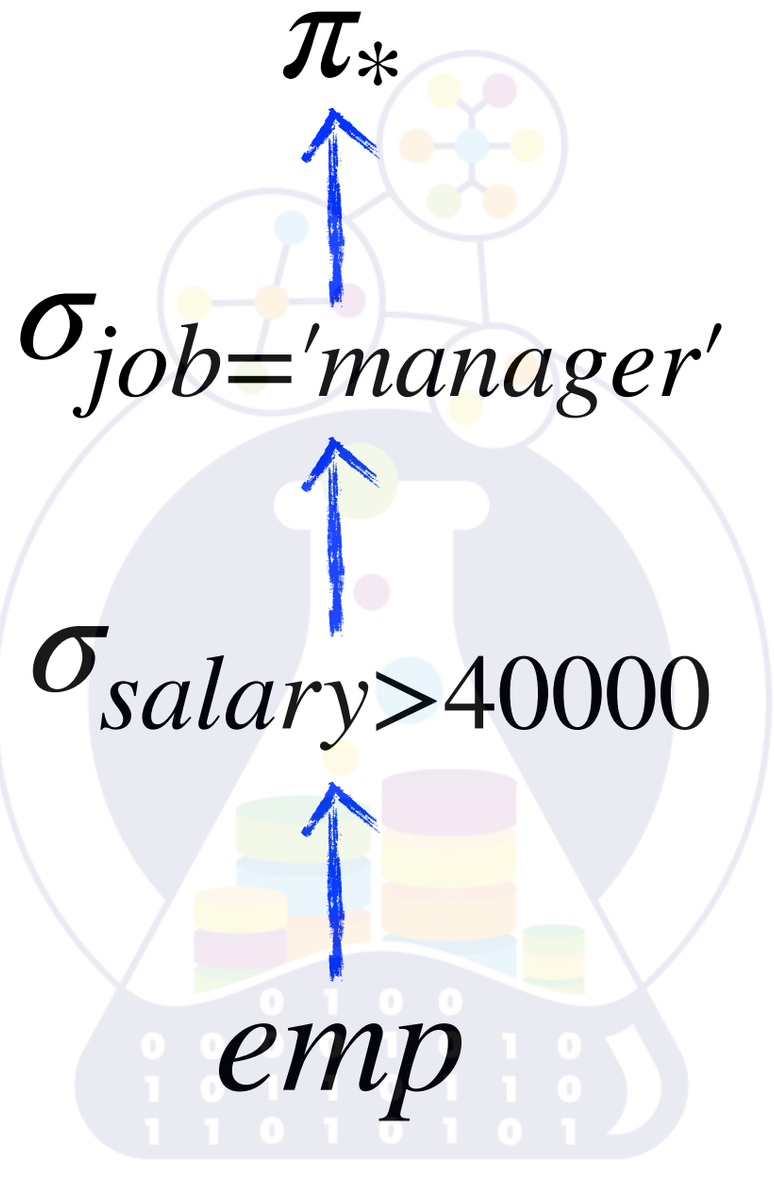
semantic equivalent

emp	
Name	**ID***
Ivan	1
Anna	2

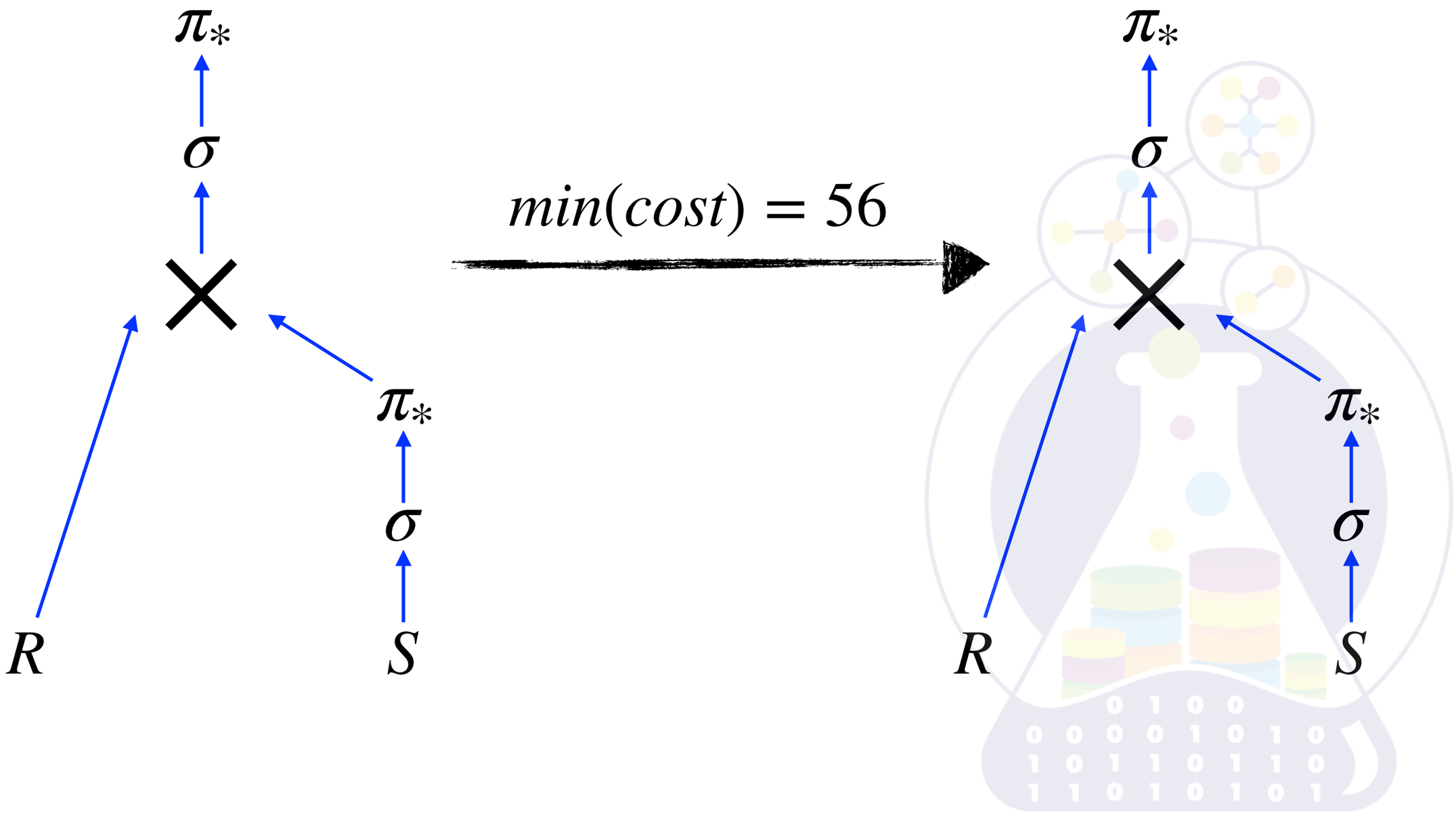
```
SELECT *
  FROM emp
 WHERE salary > 40000
```

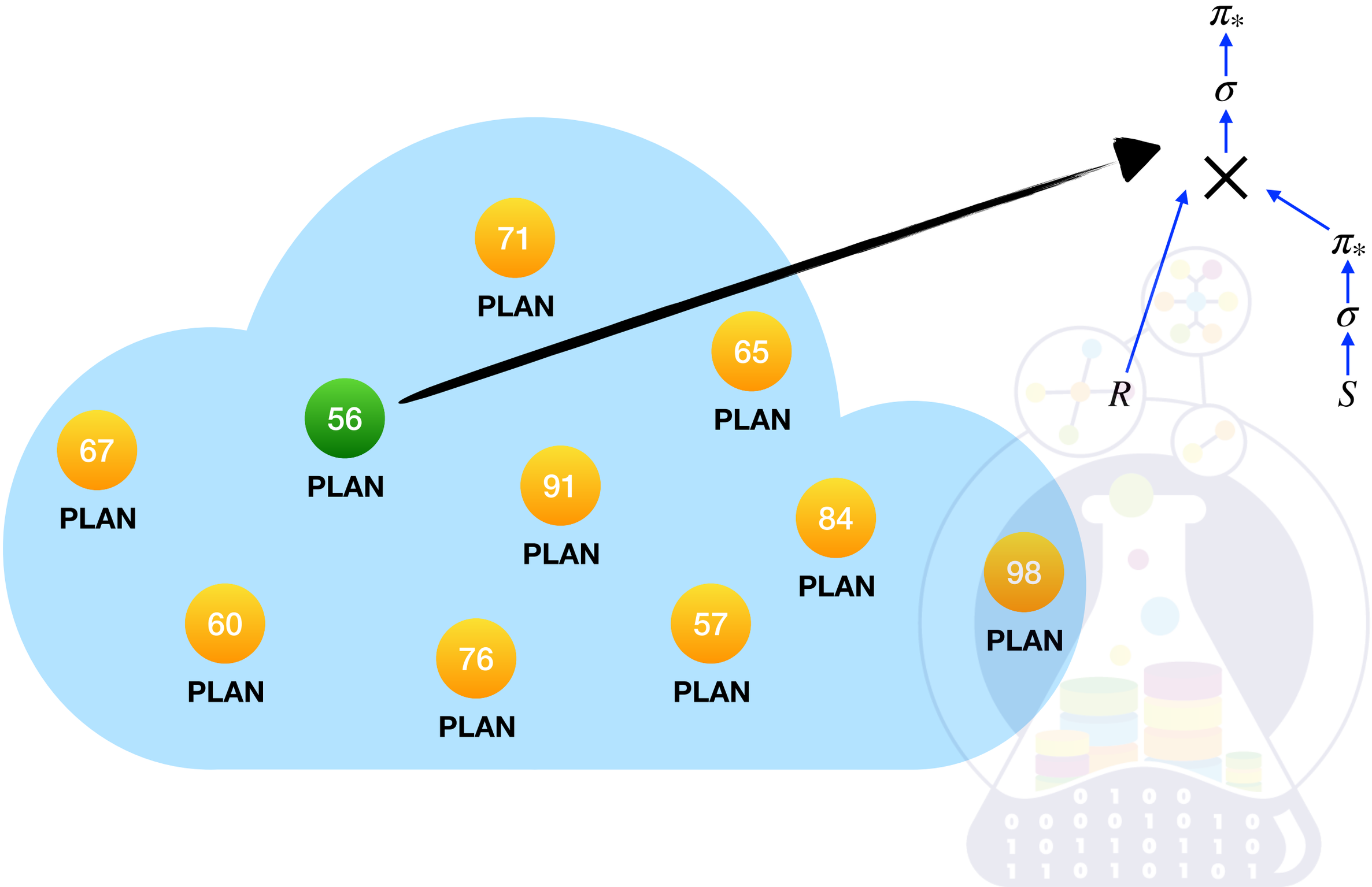


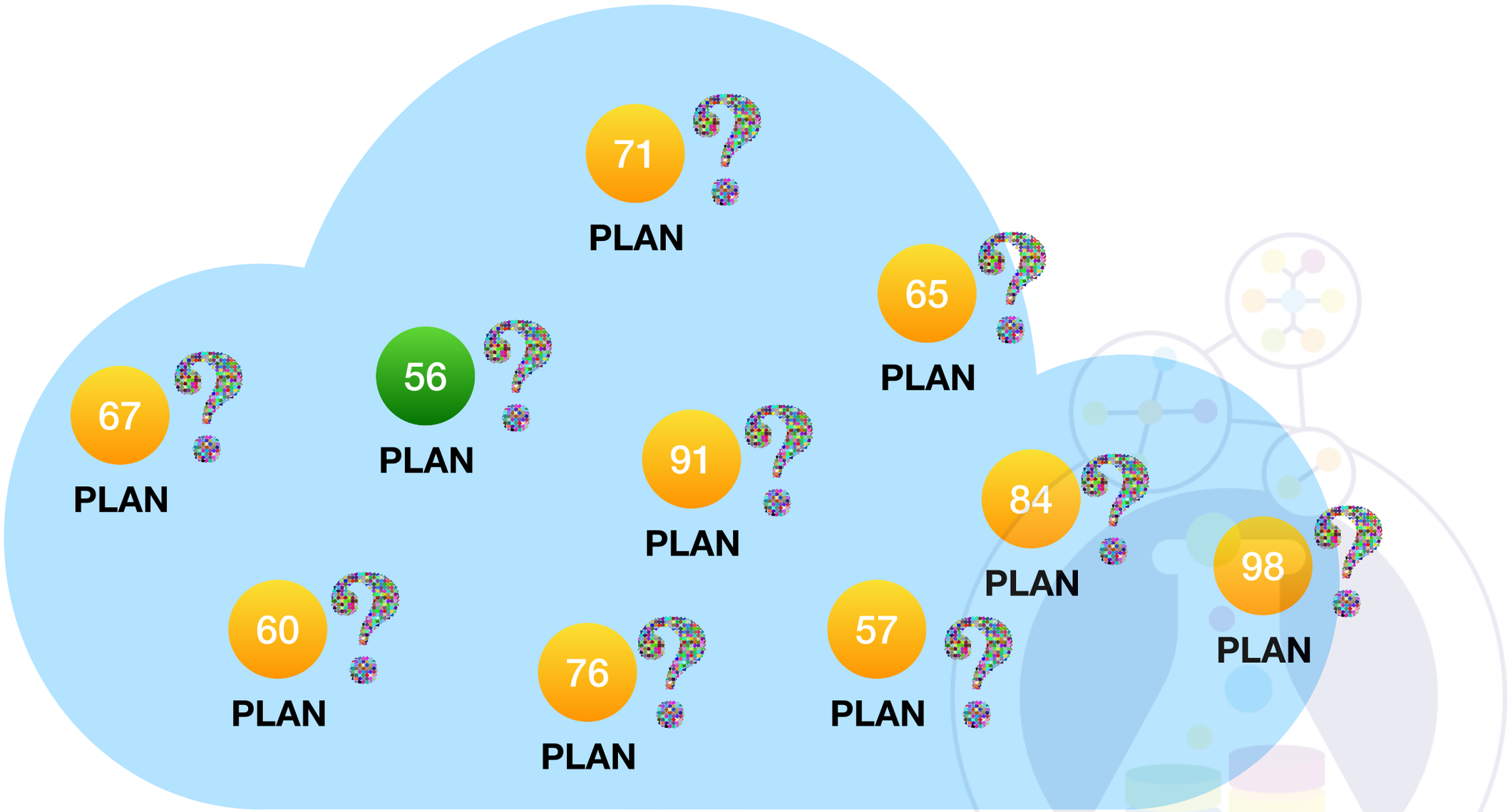
```
SELECT *
  FROM emp
 WHERE salary > 40000
       AND job = ‘manager’
```



Query Tree







```
EXPLAIN (ANALYSE , VERBOSE , BUFFERS )
    SELECT *
      FROM test
ORDER BY id
OFFSET  5000
LIMIT 5;
```

QUERY PLAN
Limit (cost=**173.78..173.96** rows=5 width=29) (actual time=1.192..1.200 rows=5 loops=1)
Output: id, name
Buffers: shared hit=37 read=15
-> **Index Scan** using test_id_idx on public.test (cost=0.29..347.29 rows=10000 width=29) (actual time=0.034..0.994 rows=5005 loops=1)
Output: id, name
Buffers: shared hit=37 read=15
Planning Time: 0.350 ms
Execution Time: 1.285 ms

```
EXPLAIN (ANALYSE , VERBOSE , BUFFERS )
WITH _start AS (
    SELECT id
        FROM public.test
ORDER BY id OFFSET 4999
LIMIT 1)
SELECT *
FROM public.test
WHERE id > (SELECT id FROM _start)
ORDER BY id
LIMIT 5;
```

QUERY PLAN
Limit (cost=**174.09..174.28** rows=5 width=29) (actual time=0.931..0.932 rows=5 loops=1)
Output: test.id, test.name
CTE _start
-> Limit (cost=173.75..173.78 rows=1 width=8) (actual time=0.920..0.920 rows=1 loops=1)
Output: test_1.id
-> **Index Only Scan** using test_id_idx on public.test test_1 (cost=0.29..347.29
Output: test_1.id
Heap Fetches: 5000
InitPlan 2 (returns $1)
-> CTE Scan on _start (cost=0.00..0.02 rows=1 width=8) (actual time=0.922..0.922 rows=1
Output: _start.id
-> **Index Scan** using test_id_idx on public.test (cost=0.29..126.61 rows=3333 width=29) …
Output: test.id, test.name
Index Cond: (test.id > $1)
Planning Time: 0.097 ms
Execution Time: 0.953 ms

COMMIT;

Azat Yakupov