

Лекция 7: Выполнение запросов

1. Выполнение запросов и оптимизация



Повышение производительности запросов

Способы повышения производительности запросов:

- Использование индексов.
- Настройка физических параметров СУБД (способ разделения пространства хранения данных, стратегии работы с транзакциями и т. д.).

Зачем это нужно?

Язык SQL декларативен:

- В запросах указывается, какими должны быть данные, которые необходимо получить.
- **Не говорится** о том, как система должна выполнить запрос.

Выполнение запросов

- 1) Разбор запроса (parser) → строится дерево
- 2) Преобразование запроса (rewriter)
- 3) Планировщик + Оптимизатор (planner) → план выполнения запроса
- 4) Выполнение плана → executor

План выполнения запроса

- Чтобы выполнить SQL-запрос необходимо построить программу — **план выполнения запроса**.
- Таких программ может быть несколько.

План выполнения запроса

СУБД должна:

- уметь построить возможные программы, результаты которых соответствуют заданному SQL-запросу
- выбрать программу, выполнение которой наиболее эффективно

Выбор плана выполнения запроса

Критерий: оценочная стоимость выполнения запроса по данному плану.

Компоненты оцениваемой стоимости:

- число обменов с устройствами внешней памяти, которые потребуются при выполнении плана запроса;
- среднее время обмена;

Обозначения (реляц. алгебра)

При построении плана выполнения запроса нам понадобится ряд обозначений из реляционной алгебры:

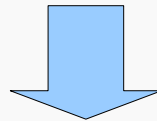
- R, S — отношения (таблицы)
- φ — предикат (условие), $\varphi_1 \wedge \varphi_2$ — составное условие

Операция выборки

- $\sigma_{\varphi}(R)$ — **операция выборки** — в результате операции формируется отношение на основе R , которое содержит только те строки (кортежи), которые удовлетворяют заданному предикату.

Операция выборки

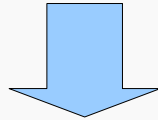
```
SELECT * FROM STUDENTS WHERE  
    STUDENTS.GROUP = '3100' AND  
    STUDENTS.ID >= 150000;
```


$$\sigma_{(\text{STUDENTS.GROUP}='3100') \wedge (\text{STUDENTS.ID} \geq 150000)}(\text{STUDENTS})$$

Проекция

- $\pi_{attr}(R)$ — проекция — в результате операции формируется новое отношение, содержащее только те атрибуты из R , которые были указаны в проекции:

SELECT name, group FROM STUDENTS;



$\pi_{\text{name, group}}(\text{STUDENTS})$

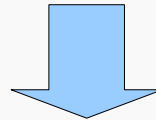
Соединение

$R \bowtie_{\theta} S$ — **соединение** (тета-соединение)

$$R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$$

Соединение

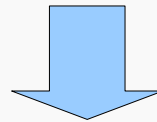
```
SELECT * FROM STUDENTS  
JOIN EXAMS ON STUDENTS.ID = EXAMS.STUD_ID;
```



STUDENTS ⋈_{STUDENTS.ID=EXAMS.STUD_ID} EXAMS

Пример

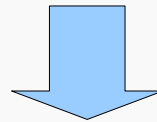
```
SELECT * FROM STUDENTS  
JOIN EXAMS ON STUDENTS.ID = EXAMS.STUD_ID  
WHERE  
  
    STUDENTS.GROUP = '3100' AND  
    STUDENTS.ID >= 150000;
```



$\sigma_{\text{STUDENTS.GROUP} = '3100' \wedge \text{STUDENTS.ID} \geq 150000} (\text{STUDENTS} \bowtie_{\text{STUDENTS.ID}=\text{EXAMS.STUD_ID}} \text{EXAMS})$

Сокращенная запись

```
SELECT * FROM STUDENTS  
JOIN EXAMS ON STUDENTS.ID = EXAMS.STUD_ID  
WHERE  
    STUDENTS.GROUP = '3100' AND  
    STUDENTS.ID >= 150000;
```



$\sigma_{\text{STUDENTS.GROUP} \wedge \text{STUDENTS.ID}} (\text{STUDENTS} \bowtie_{\text{STUDENTS.ID}=\text{EXAMS.STUD_ID}} \text{EXAMS})$

Построение плана (1)

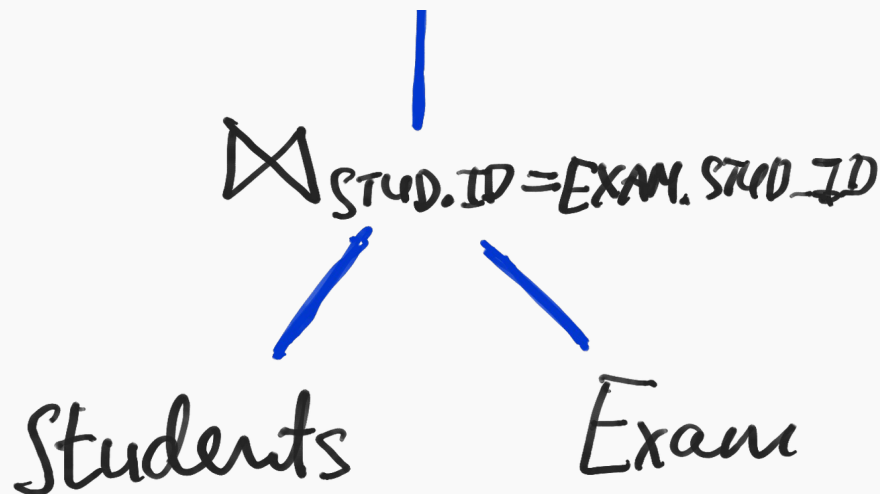
SELECT * **FROM STUDENTS**

JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID

WHERE

STUDENTS.GROUP = '3100' AND

STUDENTS.ID >= 150000;



Построение плана (2)

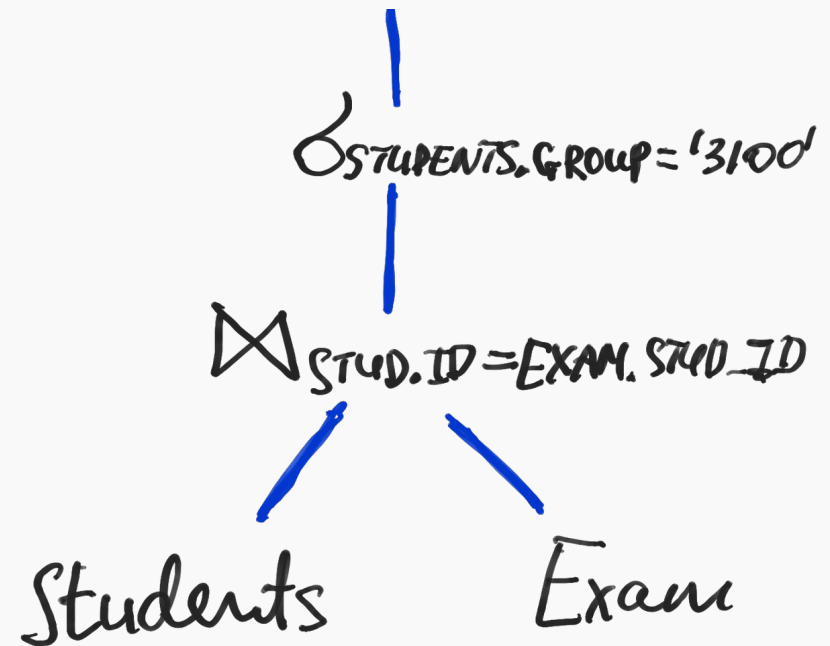
SELECT * FROM STUDENTS

JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID

WHERE

STUDENTS.GROUP = '3100' AND

STUDENTS.ID >= 150000;



Построение плана (2)

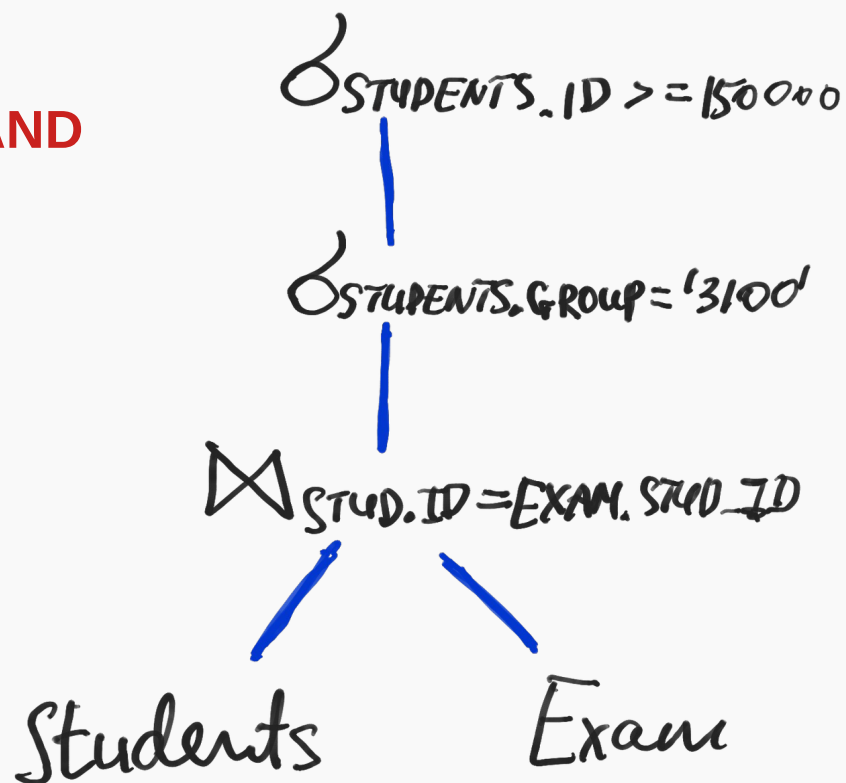
SELECT * FROM STUDENTS

JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID

WHERE

STUDENTS.GROUP = '3100' **AND**

STUDENTS.ID >= 150000;

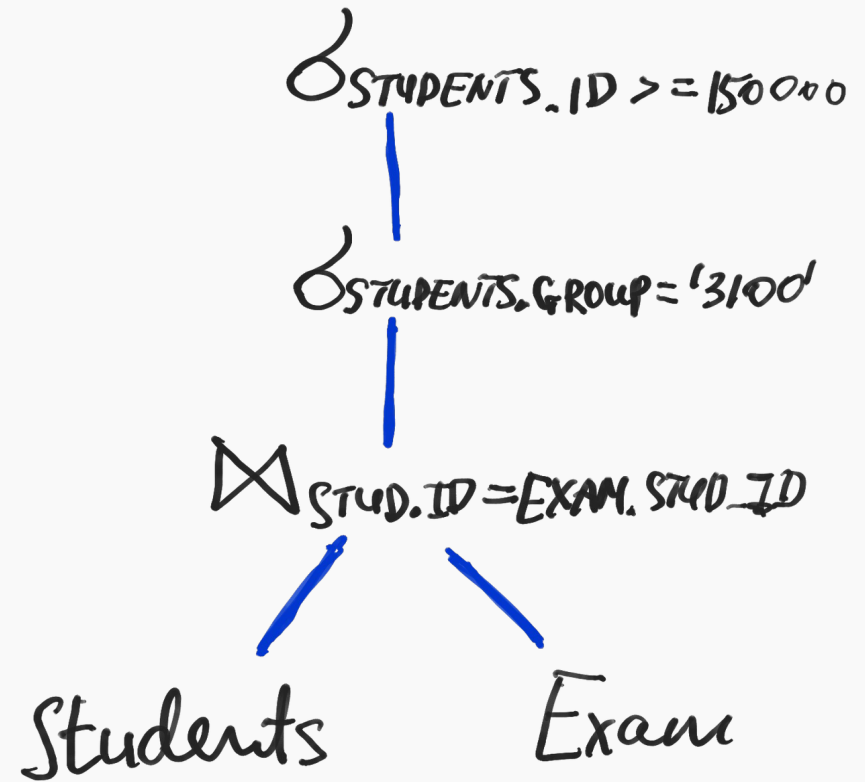
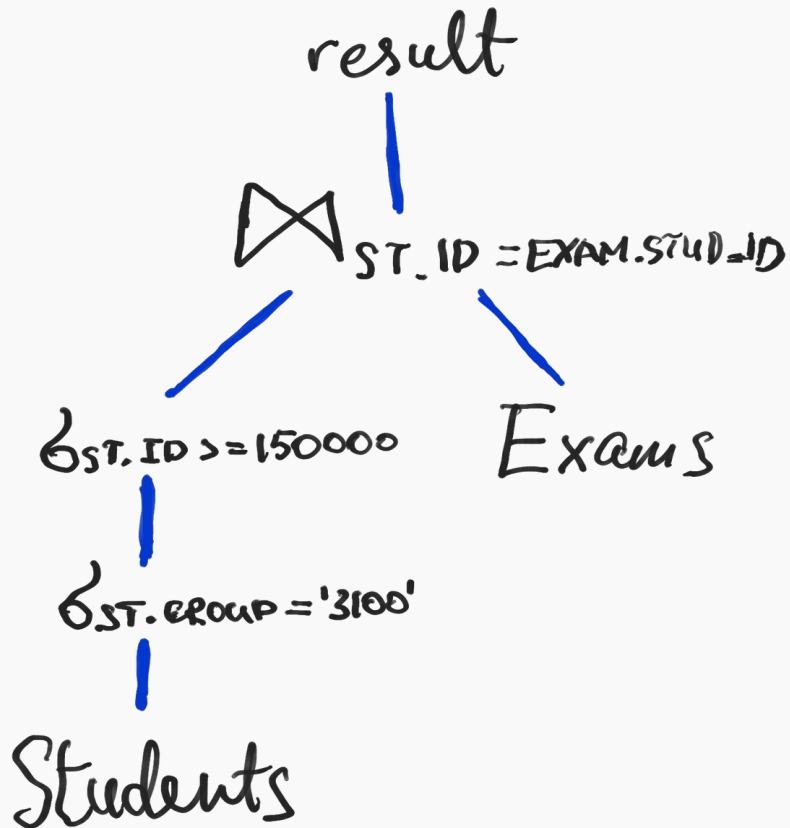


Можно ли выполнить запрос другим способом?

```
SELECT * FROM STUDENTS  
JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID  
WHERE  
    STUDENTS.GROUP = '3100' AND  
    STUDENTS.ID >= 150000;
```

Какие планы выполнения запроса возможны?

Пример



$R \bowtie_{\theta} S \equiv S \bowtie_{\theta} R$ (коммутативность)

$R \bowtie_{\theta} (S \bowtie_{\varphi} T) \equiv (R \bowtie_{\theta} S) \bowtie_{\varphi} T$ (ассоциативность)

$\sigma_{\theta \wedge \varphi} (R) \equiv \sigma_{\theta} (\sigma_{\varphi} (R))$

$\sigma_{\varphi}(R \bowtie_{\theta} S) \equiv (\sigma_{\varphi}(R) \bowtie_{\theta} S)$, если φ относится к атрибутам R

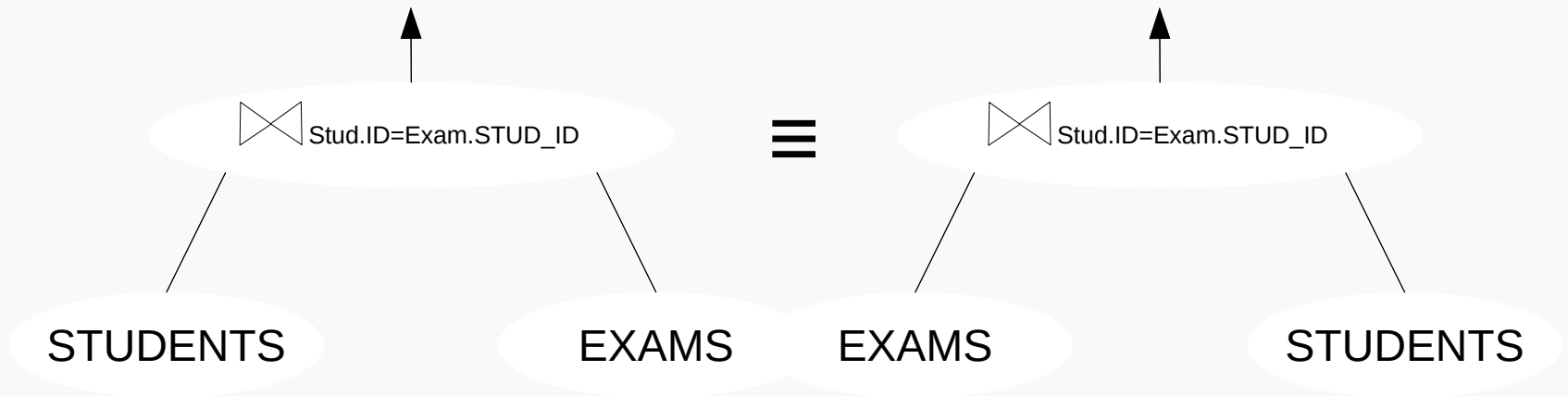
$\pi_A (R \bowtie_{\theta} S) \equiv \pi_A(\pi_{(A \cup B) \cap \text{attrs}(R)}(R) \bowtie_{\theta} S)$, B — атрибуты из условия θ

Законы (1)

$R \bowtie_{\theta} S \equiv S \bowtie_{\theta} R$ (коммутативность)

SELECT * FROM STUDENTS

JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID



LEFT JOIN и коммутативность

R LEFT JOIN S ON R.id = S.id
S LEFT JOIN R ON R.id = S.id $\equiv ?$

R

id	vr
1	a
2	b
3	c

S

id	vs
3	d
4	e
5	f

LEFT JOIN и коммутативность

R LEFT JOIN S ON R.id = S.id

R

id	vr
1	a
2	b
3	c

S

id	vs
3	d
4	e
5	f

Result

R.id	R.vr	S.id	S.vs
1	a	NULL	NULL
2	b	NULL	NULL
3	c	3	d

LEFT JOIN и коммутативность

S LEFT JOIN R ON R.id = S.id

R

id	vr
1	a
2	b
3	c

S

id	vs
3	d
4	e
5	f

Result

R.id	R.vr	S.id	S.vs
3	c	3	d
NULL	NULL	4	e
NULL	NULL	5	f

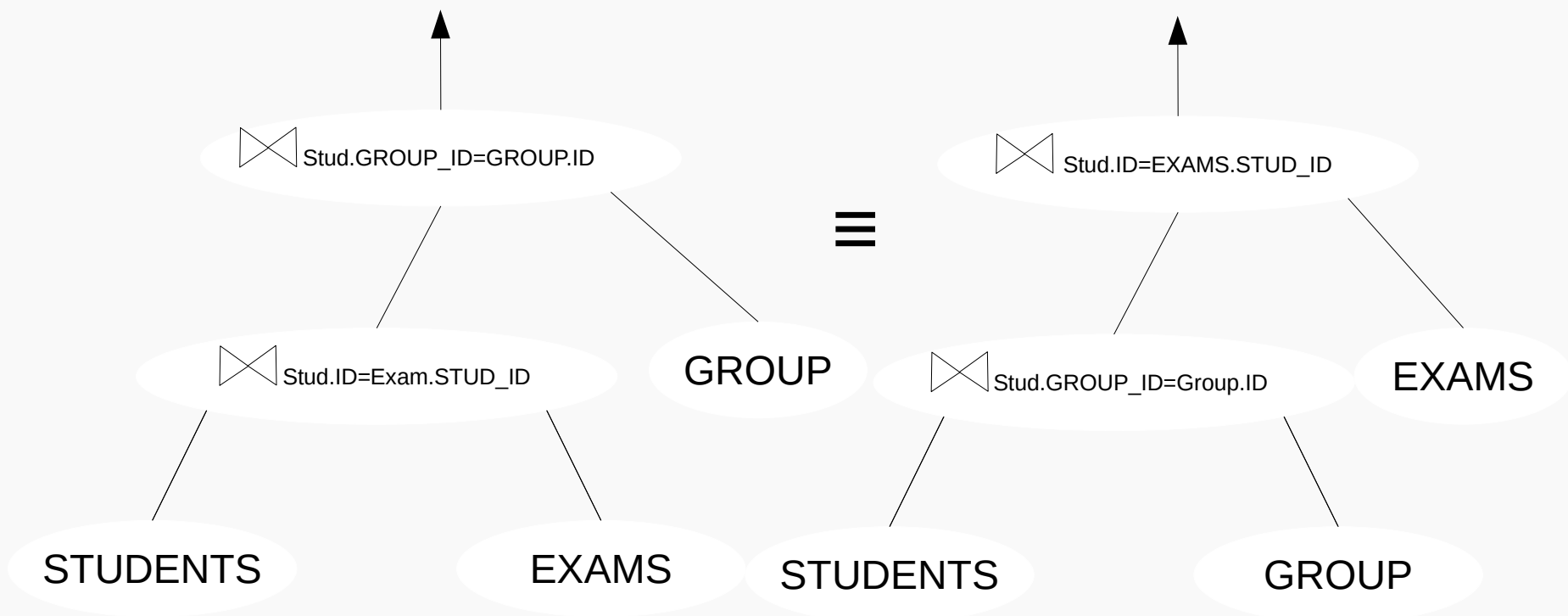
Законы (2)

$R \bowtie_{\theta} (S \bowtie_{\varphi} T) \equiv (R \bowtie_{\theta} S) \bowtie_{\varphi} T$ (ассоциативность)

```
SELECT * FROM STUDENTS  
JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID  
JOIN GROUP ON STUDENTS.GROUP_ID = GROUP.ID
```

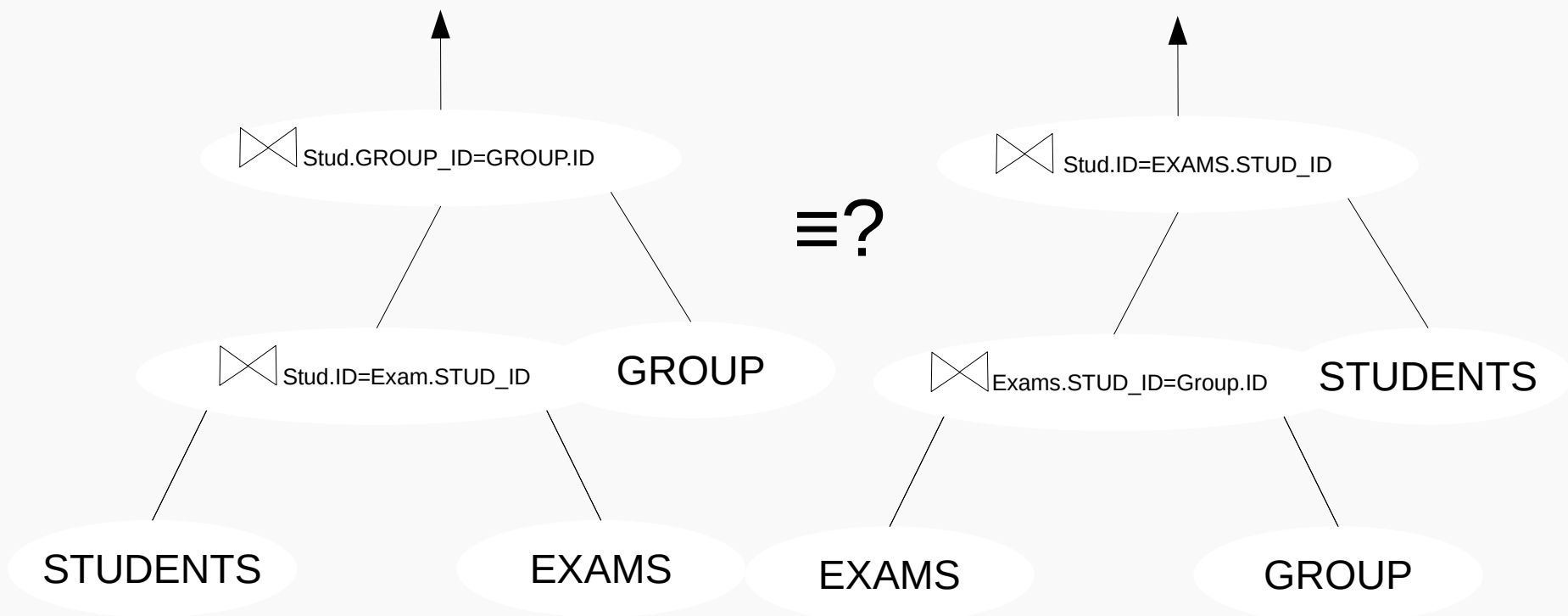
Законы (2)

$R \bowtie_{\theta} (S \bowtie_{\varphi} T) \equiv (R \bowtie_{\theta} S) \bowtie_{\varphi} T$ (ассоциативность)



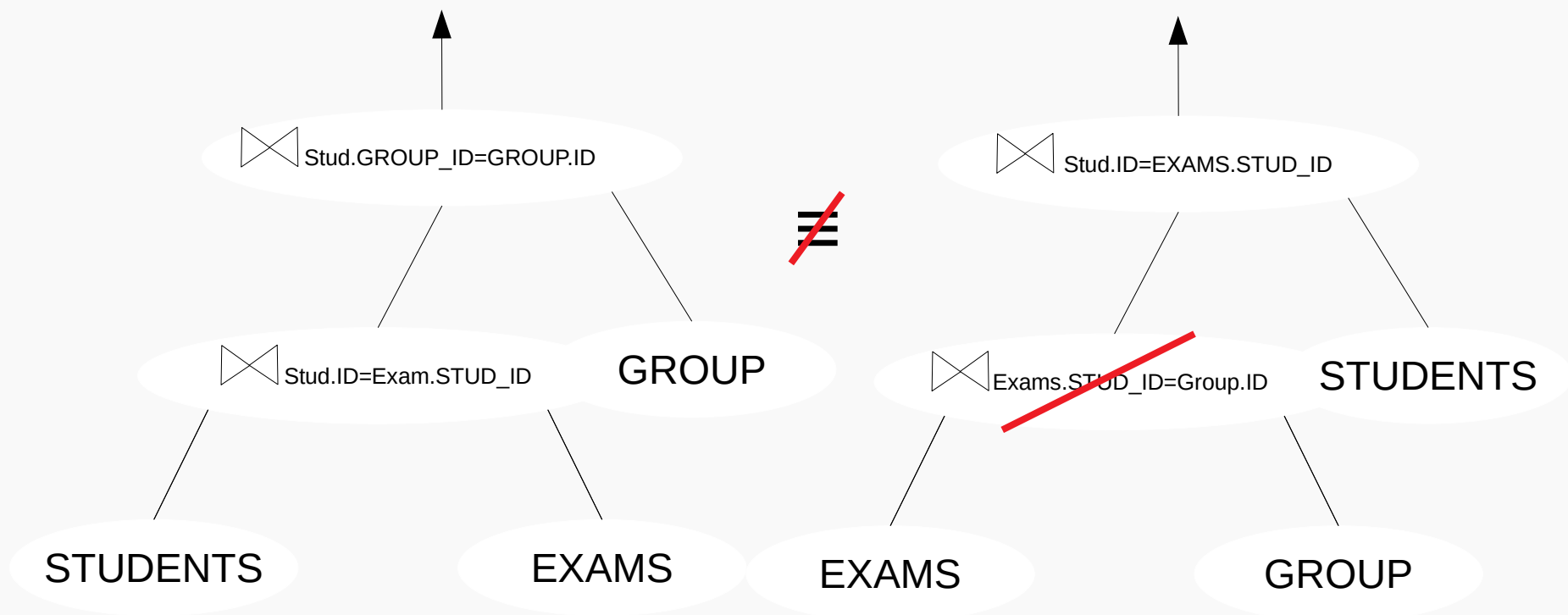
Законы (2)

Эквивалентны ли данные планы?



Законы (2)

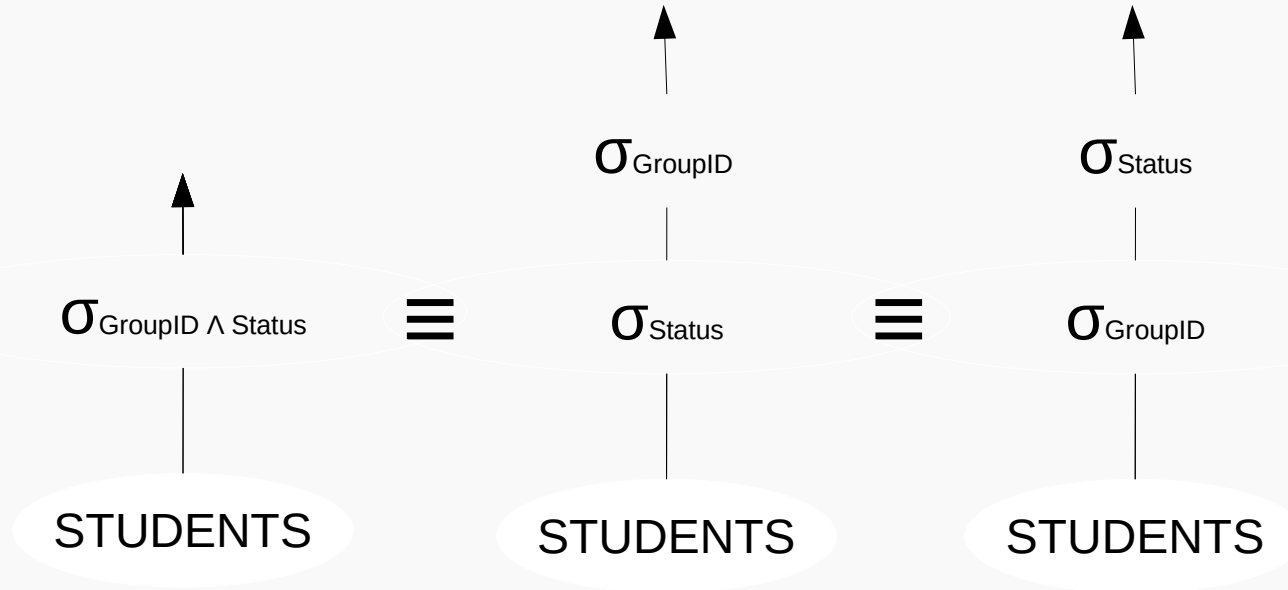
Нет, EXAM и STUDENTS не связаны.



Законы (3)

$$\sigma_{\theta \wedge \varphi} (R) \equiv \sigma_{\theta} (\sigma_{\varphi} (R))$$

SELECT * FROM STUDENTS
WHERE GroupID = 32 AND Status = 'Обучение';



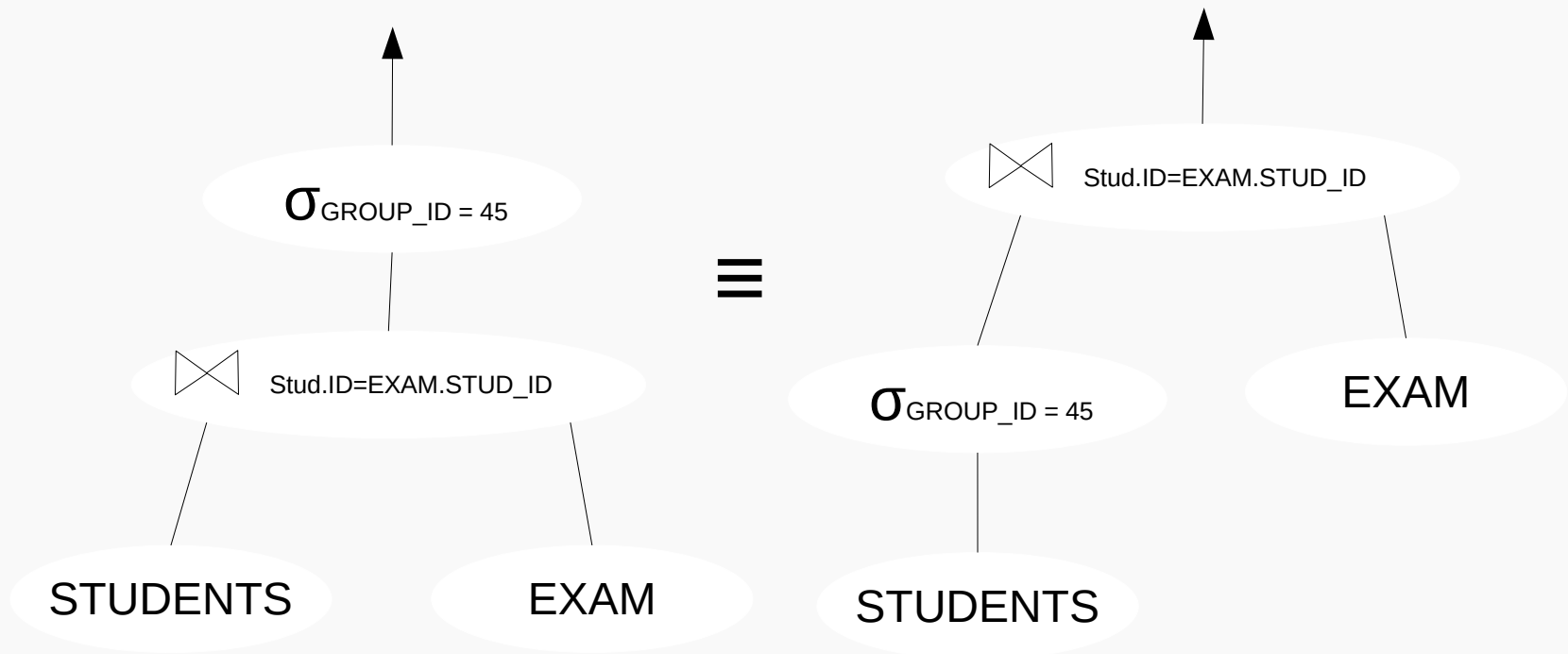
Законы (4)

$\sigma_{\varphi}(R \bowtie_{\theta} S) \equiv (\sigma_{\varphi}(R) \bowtie_{\theta} S)$, если φ относится к атрибутам R

```
SELECT * FROM STUDENTS  
JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID  
WHERE GROUP_ID = 45;
```

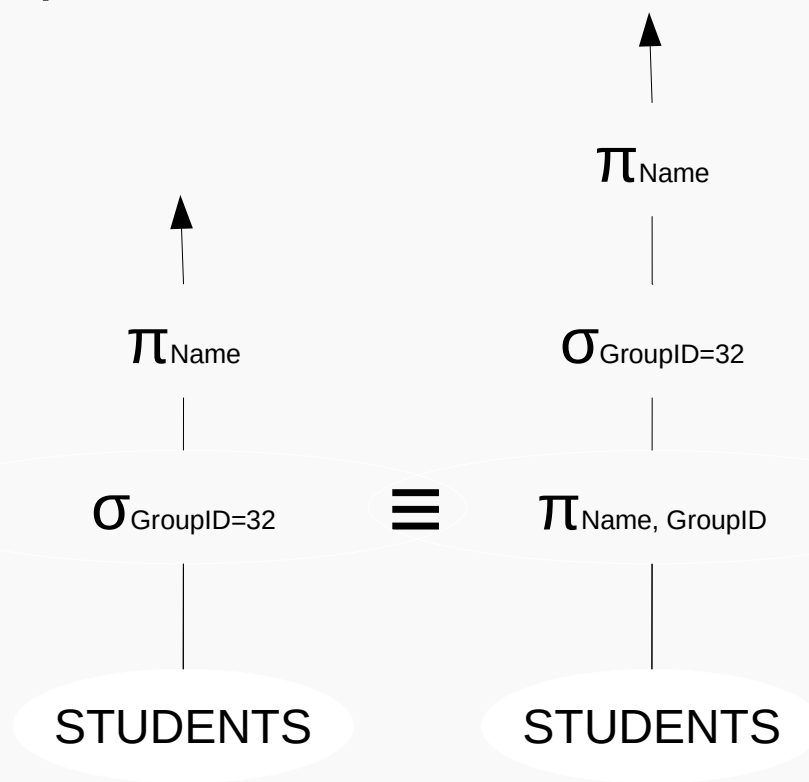
Законы (4)

```
SELECT * FROM STUDENTS  
JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID  
WHERE GROUP_ID = 45;
```



Выборка и проекция

SELECT Name FROM STUDENTS
WHERE GroupID = 32;



Законы (5)

$\pi_A (R \bowtie_{\theta} S) \equiv \pi_A (\pi_{(A \cup B) \cap \text{attrs}(R)}(R) \bowtie_{\theta} S)$, B — атрибуты из условия θ

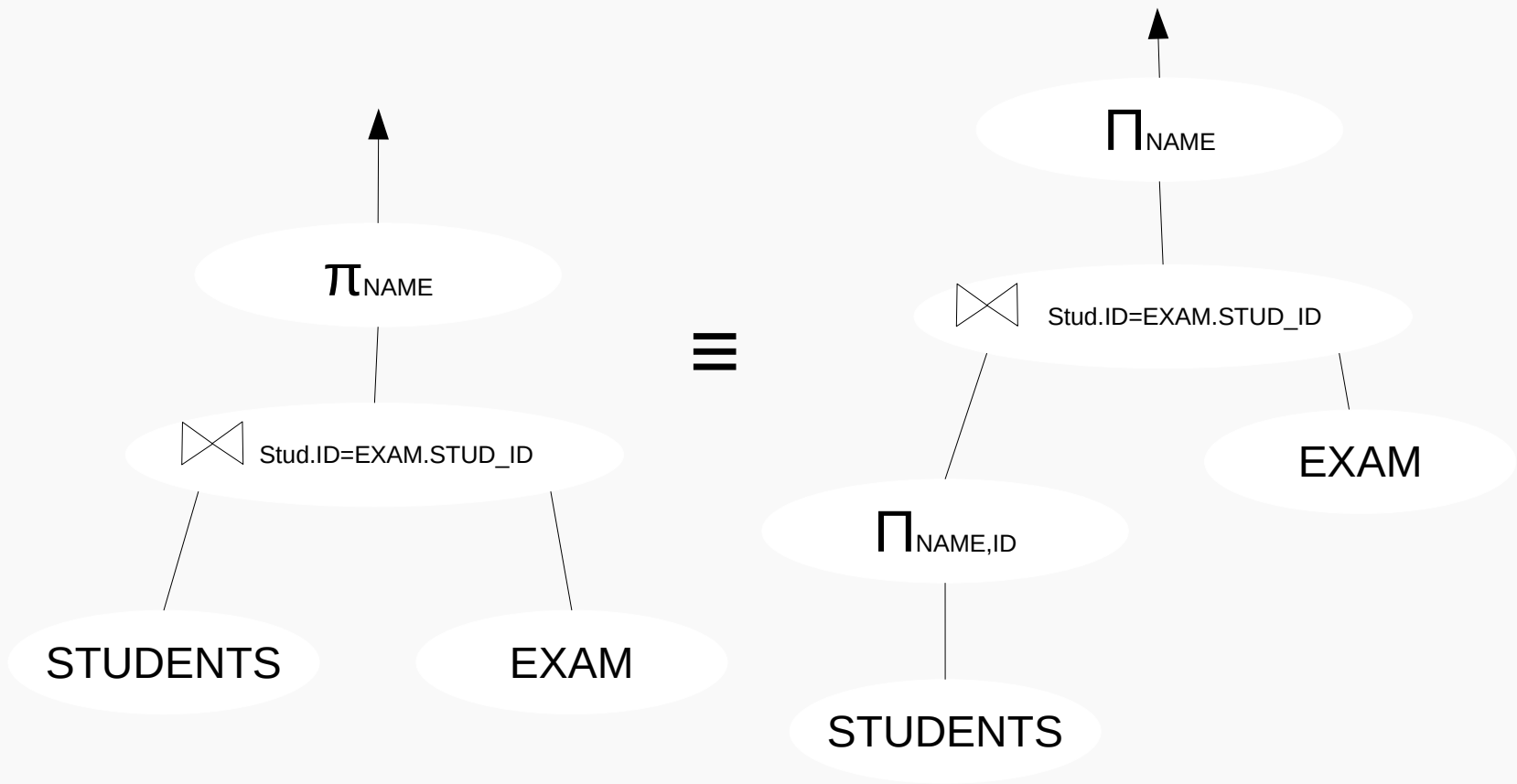
```
SELECT NAME FROM STUDENTS
```

```
JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID;
```

Законы (5)

SELECT NAME FROM STUDENTS

JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID;



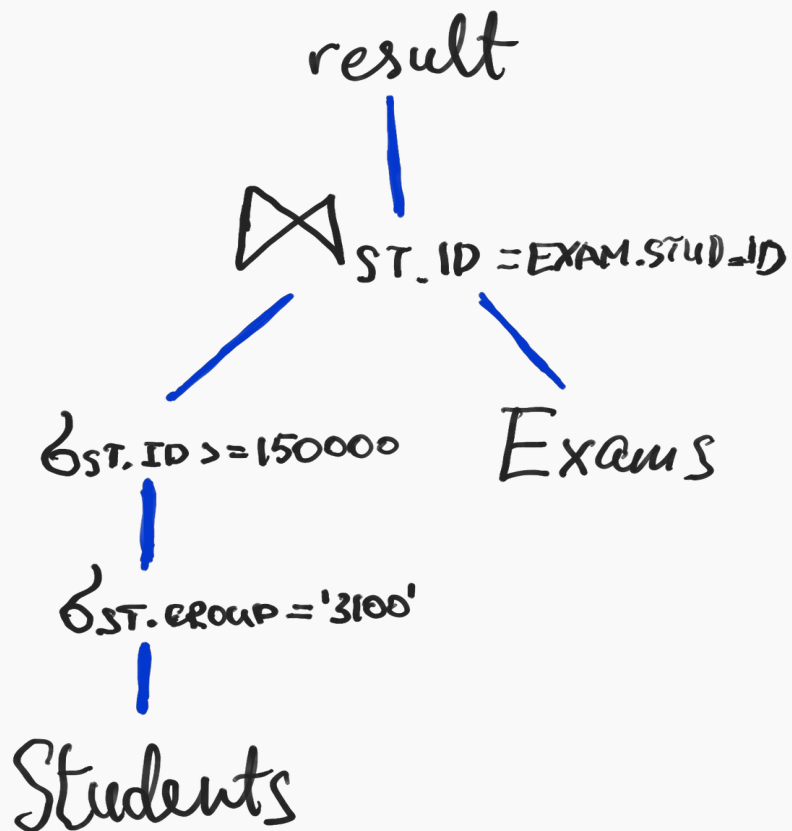
Материализация данных

- Сохранение результатов промежуточных операций.
- Увеличивает время выполнения запроса:
 - запись промежуточных результатов;
 - чтение сохраненных результатов для выполнения последующих операций;

Пример

```
SELECT * FROM STUDENTS  
JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID  
WHERE  
    STUDENTS.GROUP = '3100' AND  
    STUDENTS.ID >= 150000;
```

Пример



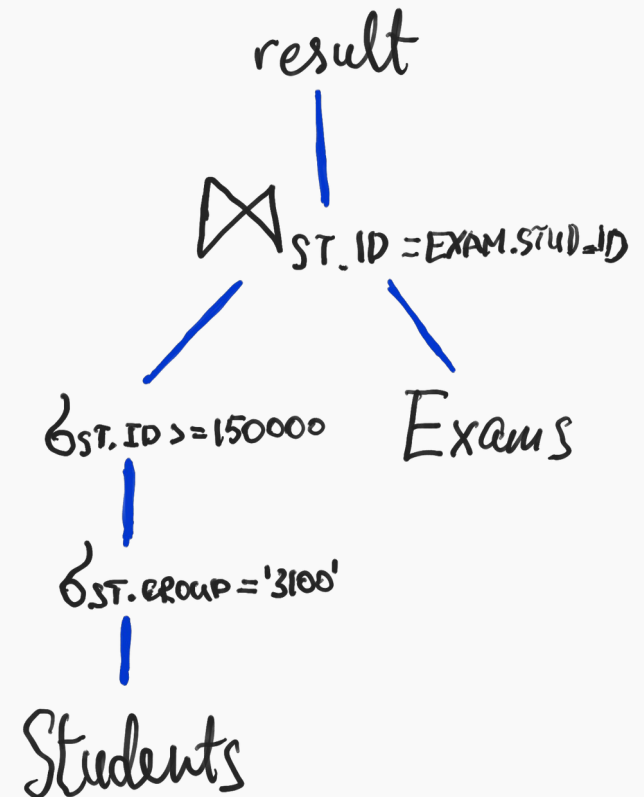
Конвейерная обработка данных

- **Конв. обр. данных** — передача результатов одной операции на обработку другой без создания временных отношений (для хранения промежуточных результатов).
- Для планов, в которых большинство операций происходят в конвейере:
при расчете стоимости плана выполнения запроса **отпадает необходимость** в учете стоимости записи и последующего чтения промежуточных таблиц.

Конвейерная обработка

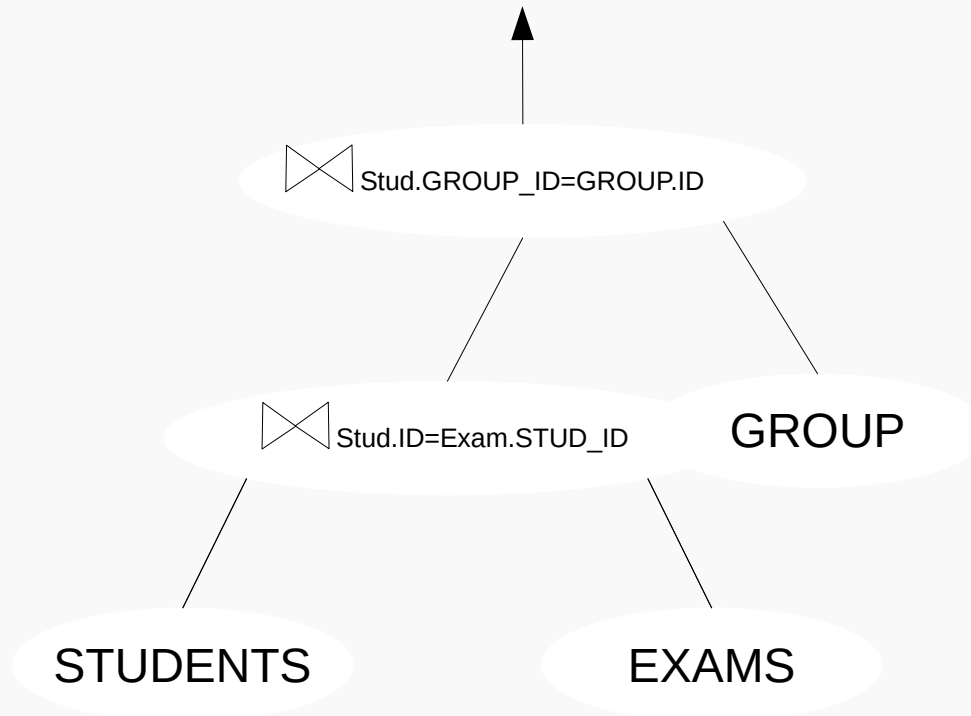
```
SELECT * FROM STUDENTS  
JOIN EXAMS ON STUDENTS.ID = EXAM.STUD_ID  
WHERE  
    STUDENTS.GROUP = '3100' AND  
    STUDENTS.ID >= 150000;
```

$\sigma_{SID \geq 150000}(\sigma_{GROUP = '3100'}(STUDENTS))$



Левосторонние деревья

- **Левостороннее дерево** — результат соединения - в левой части дерева, представляющего план.
- Внешнее отношение — слева.



Левосторонние деревья

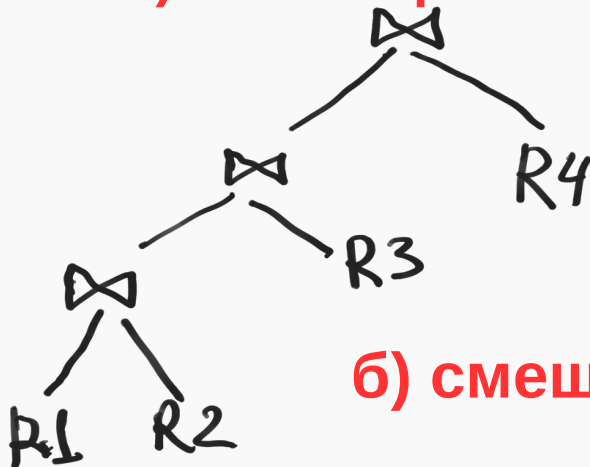
Обычно оптимизаторы запросов в СУБД рассматривают только **левосторонние деревья**:

- 1) необходимо сократить число планов для анализа;
- 2) такие планы позволяют избежать материализации, используя **конвейерную обработку данных**;

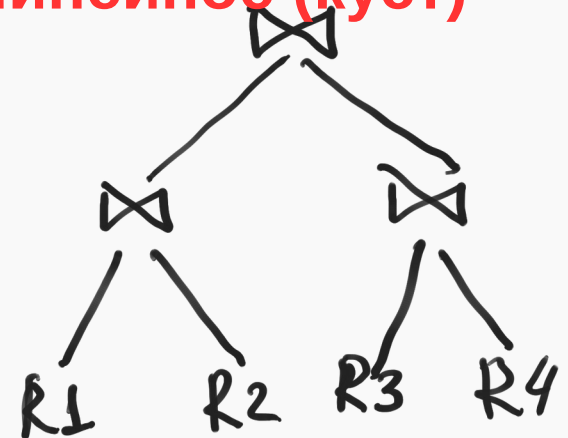
Типы деревьев при построении планов

$R1 \bowtie R2 \bowtie R3 \bowtie R4$

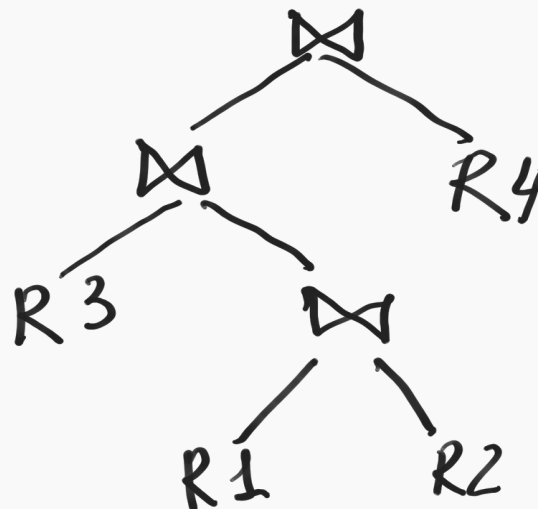
а) левостороннее



в) нелинейное (куст)



б) смешанное линейное



Советы при построении плана

- Использовать конвейерную обработку (левосторонние планы, избегать блокирующих операций).
- Делать выборку как можно раньше.
- Делать проекции раньше.
- Грамотно планировать соединения.

Цель: уменьшение размеров промежуточных данных => уменьшение **числа операций** чтения записи во внешнюю память

Индексы при построении планов

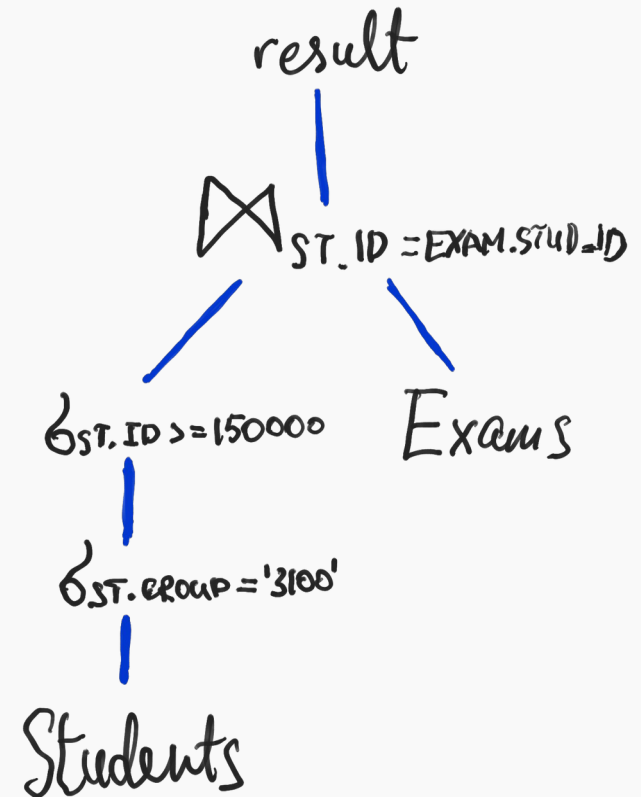
Могут быть полезны:

- выборка данных (фильтрация);
- соединения;

Негативные эффекты:

- может потребоваться материализация промежуточных результатов.

Влияние индексов (пример) — конвейерная обработка



Индекс:

Hash index на STUDENT.GROUP

(оба индекса ставить не имеет смысла из-за конв. обработки)

Расчет стоимости плана выполнения запроса

Можно выделить 3 основных составляющих:

- Чтение входных или промежуточных таблиц.
- Запись промежуточных данных (материализация — результат сохраняется во временных отношениях после выполнения одной операции для обработки следующей операцией).
- Сортировка результата (DISTINCT).

2. Выполнение соединений

Соединения (JOIN)

$$R \bowtie_{R.id=S.id} S = \sigma_{R.id=S.id}(R \times S)$$

R

id	vr
1	a
3	c

S

id	vs
3	d
5	f

Result:

R.id	R.vr	S.id	S.vs
1	a	3	d
1	a	5	f
3	c	3	d
3	c	5	f

Выполнение соединений (JOIN)

- Реализации:
 - Nested Loop Join
 - Hash Join
 - Sort-merge Join

Соединение с использованием вложенного цикла

$R \bowtie_{\text{condition}} S$

R — внешнее отношение

S — внутреннее отношение

Для каждой записи r в R :

 Для каждой записи s в S :

 Если $\text{condition} == \text{true}$, $r+s$ - в результат.

Соединение с использованием вложенного цикла

```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups

GroupID	GrName
1	P3100
3	P3102
4	P3103
2	P3101


Students

StudID	GroupID	Name
45	4	Ivan
15	2	Alex
54	2	Gleb
54	1	Svetlana

Пример


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan

Пример

```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan

Пример


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan

Пример


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina

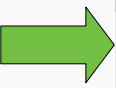
Result:

GrName	Name
P3100	Ivan

Пример

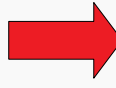
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina

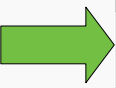
Result:

GrName	Name
P3100	Ivan

Пример

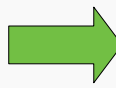
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina

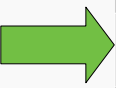
Result:

GrName	Name
P3100	Ivan
P3101	Petr

Пример


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina

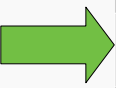
Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Пример

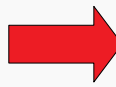
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Пример

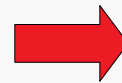
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Пример


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Пример

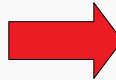
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Пример


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Пример

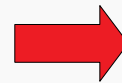
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Пример


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Пример

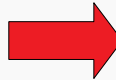
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina

Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Пример

```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students

StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina

Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb
P3103	Irina

Выполнение соединения

```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Страница

Students

StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения

```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Страница в
буфере СУБД

Students

StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Страница на
внешнем диске

Выполнение соединения

Предположим: одновременно можно поместить 3 страницы (для наглядности)

Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Страница в
буфере СУБД

Students

StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Страница на
внешнем диске

Выполнение соединения


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students



StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students



StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения

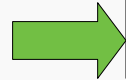
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students



StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students



StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения

```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students



StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения

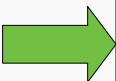
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students




StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Проблема

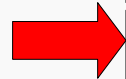
Проблема: для того, чтобы продолжить выполнение, нужно загрузить страницу.

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students




StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения

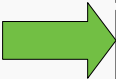
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students



StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students



StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students




StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students



StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Выполнение соединения

Так для всех GroupID из Groups (на рисунке состояние последнего шага):

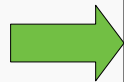
Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107



Students

StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina



Проблема

Для каждой итерации внешнего цикла нужно заново загружать страницы для внутреннего.

Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Students

StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Block nested loop Join

$R \bowtie_{\text{condition}} S$

R — внешнее отношение

S — внутреннее отношение

Для каждого блока N страниц в R :

Для каждой страницы из S :

Проверка совпадающих страниц:

если $\text{condition} == \text{true}$, $r+s$ - в результат.

Block nested loop Join

```
SELECT GrName, Name FROM STUDENTS s
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Блок

Страница

Students

StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Block nested loop Join

```
SELECT GrName, Name FROM STUDENTS s
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Блок

Страница

Students

StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Block nested loop Join

```
SELECT GrName, Name FROM STUDENTS s
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Блок

Страница

Students

StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Block nested loop Join

```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103
5	P3105
6	P3104
8	P3109
7	P3107

Блок

Страница

Students

StudID	GroupID	Name
23	1	Ivan
131	2	Petr
454	2	Gleb
623	1	Svetlana
211	2	Petr
554	2	Kira
151	3	Vasily
654	2	Anton
223	4	Irina

Соединение с использованием вложенного цикла

$R \bowtie_{\text{condition}} S$

R — внешнее отношение

S — внутреннее отношение

Эффективнее, если внешнее отношение (R) —
меньше, чем внутреннее (S):

- уменьшается число чтений внутреннего отношения

Sort-Merge Join

Соединение слиянием — обе таблицы должны быть **отсортированы** (по соединяемым столбцам).

Groups

GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103


Students

StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina

Sort-Merge Join

```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan

Sort-Merge Join

```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan

Sort-Merge Join


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina


Result:

GrName	Name
P3100	Ivan
P3101	Petr

Sort-Merge Join

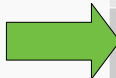
```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina

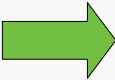
Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Sort-Merge Join


```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups



GroupID	GrName
1	P3100
2	P3101
3	P3102
4	P3103

Students



StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina

Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb

Sort-Merge Join

```
SELECT GrName, Name FROM STUDENTS s  
JOIN GROUPS g ON s.GroupID = g.GroupID;
```

Groups

GroupID	Name
1	P3100
2	P3101
3	P3102
4	P3103

Students

StudID	GroupID	Name
23	1	Ivan
11	2	Petr
54	2	Gleb
23	4	Irina

Result:

GrName	Name
P3100	Ivan
P3101	Petr
P3101	Gleb
P3103	Irina

Index nested loop Join

$$R \bowtie_{R.attr=S.attr} S$$

R — внешнее отношение

S — внутреннее отношение

Есть индекс на $S.attr$

Для каждой записи r в R :

ищем соответствующую запись s в S — используя индекс

3. Просмотр плана выполнения запроса

EXPLAIN — позволяет посмотреть план выполнения запроса, отобранный PostgreSQL.

EXPLAIN *query*;

query — не выполняется.

EXPLAIN SELECT * FROM STUDENTS;

QUERY PLAN

*Seq Scan on Students (cost=0.00..309.00 rows=900
width=170)*

Пример

EXPLAIN SELECT * FROM STUDENTS;

QUERY PLAN

*Seq Scan on Students (**cost=0.00..309.00 rows=900 width=170**)*

Предположительная
стоимость операции

Предположительный
размер строки

Предположительное
число строк


```
EXPLAIN SELECT * FROM STUDENTS  
WHERE StudID =942;
```

QUERY PLAN

Index Scan using Students_StudId on Students

(cost=0.32..8.34 rows=1 width=150)

Index Cond: (StudId = 942)

EXPLAIN ANALYZE

Для выполнения запроса используется EXPLAIN ANALYZE:

```
EXPLAIN ANALYZE SELECT * FROM STUDENTS  
WHERE StudID =942;
```

При подготовке презентации использовались материалы из:

- Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов, Издательство: BHV, 2009 г.
- Документация PostgreSQL.

<https://www.postgresql.org/about/licence/>

PostgreSQL is released under the PostgreSQL License, a liberal Open Source license, similar to the BSD or MIT licenses.

PostgreSQL Database Management System
(formerly known as Postgres, then as Postgres95)

Portions Copyright © 1996-2020, The PostgreSQL Global Development Group

Portions Copyright © 1994, The Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.