

# Обработка и исполнение запросов в СУБД (Лекция 1)

Классические системы: общая архитектура, модель volcano, подходы к реализации различных операторов

v3

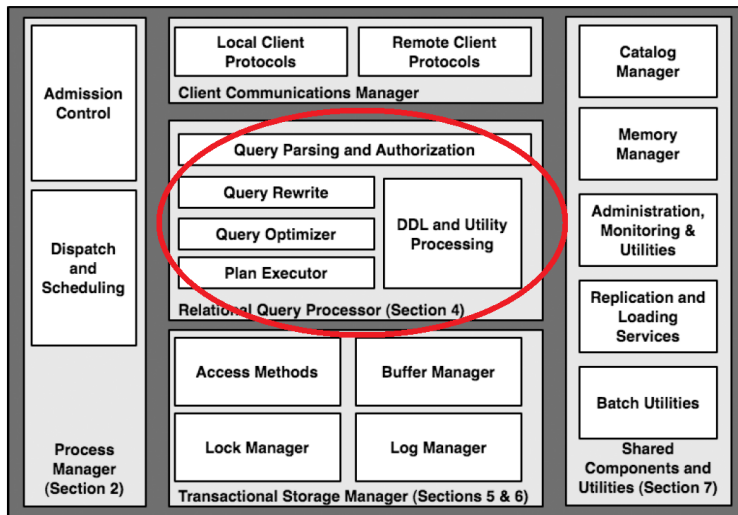
Георгий Чернышев

Академический Университет

*chernishev@gmail.com*

4 сентября 2017 г.

# Основные компоненты классической реляционной СУБД



1

<sup>1</sup>Изображение взято из [Hellerstein et al., 2007]

# “SELECT... → ответ. Как?”

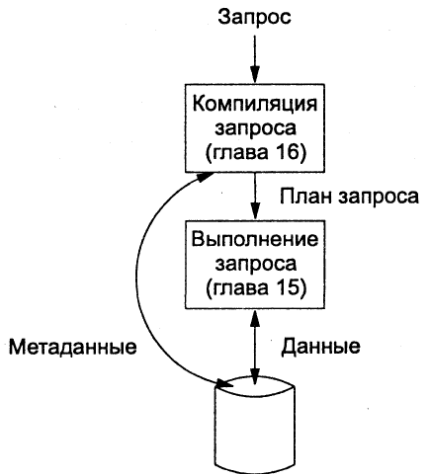


Рис. 15.1. Главные функции процессора запросов

2

<sup>2</sup>Изображение взято из [Garcia-Molina et al., 2004]

# Фаза компиляции запроса

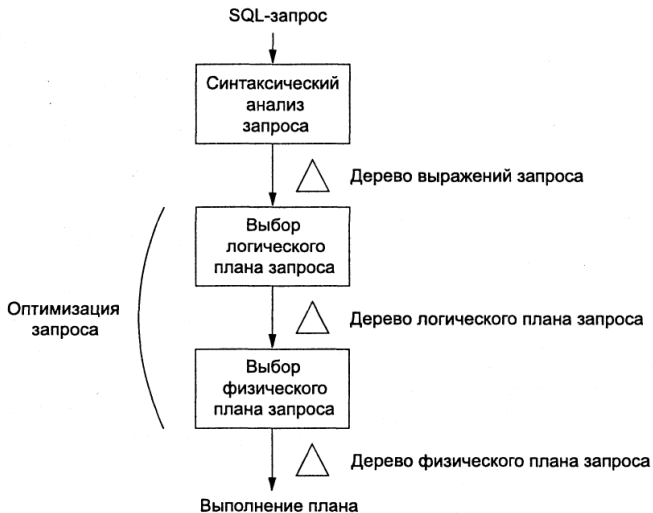


Рис. 15.2. Диаграмма процесса компиляции запроса

3

<sup>3</sup> Изображение взято из [Garcia-Molina et al., 2004]

# Шаги компиляции запроса

## Высокоуровнево:

- Синтаксический анализ (parsing) — построение дерева разбора описывающего структуру запроса;
- Перезапись запроса (query rewrite) — по дереву разбора строим начальный логический план, затем улучшаем его до эффективного логического плана;
- Генерация физического плана (physical plan generation) — по логическому плану, на основе метаданных выбираются конкретные реализации операторов.

# Фаза компиляции запроса

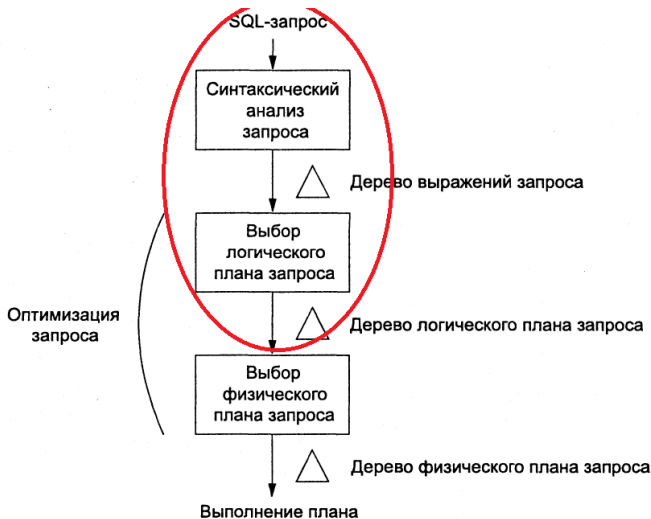


Рис. 15.2. Диаграмма процесса компиляции запроса

4

<sup>4</sup> Изображение взято из [Garcia-Molina et al., 2004]

# От запроса к логическому плану



Рис. 16.1. От запроса к логическому плану запроса

5

<sup>5</sup> Изображение взято из [Garcia-Molina et al., 2004]

# Замечания I

- Синтаксический анализ делается достаточно просто, есть масса стандартных средств: YACC, BISON, FLEX. На выходе получается некое внутреннее представление, дерево запроса;

```
%%      /* Bison grammar rules */
input   : /* empty production to allow an empty input */
        | input line
        ;

line    : term '\n'    { printf("Result is %f\n", $1); }

        ;

term    : term '*' factor { $$ = $1 * $3; }
        | term '/' factor { $$ = $1 / $3; }
        | factor         { $$ = $1; }
        ;

factor  : NUMBER       { $$ = $1; }
        ;
```

6



- Препроцессор — подстановка дерева выражений для представлений, разрешение сущностей, семантический контроль:
  - Контроль употребления имен отношений;
  - Контроль использования имен атрибутов и их разрешение;
  - Контроль типов.
- Фаза перезаписи запроса: упрощение без потери семантики;
- Запрос “собирается” из набора реляционных операторов, по определенным правилам;
- Реляционная алгебра + теоретико-множественные операции позволяют комбинировать эти операторы;

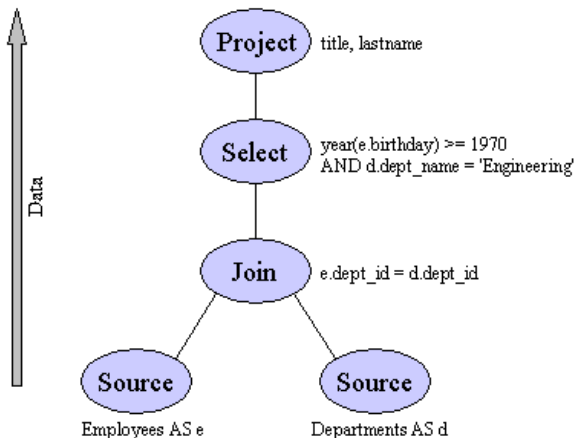
Итог: логический план запроса.

<sup>6</sup>

Изображение взято из

<https://image.slidesharecdn.com/yacclex-140518211555-phpapp02/95/yacc-lex-11-638.jpg>

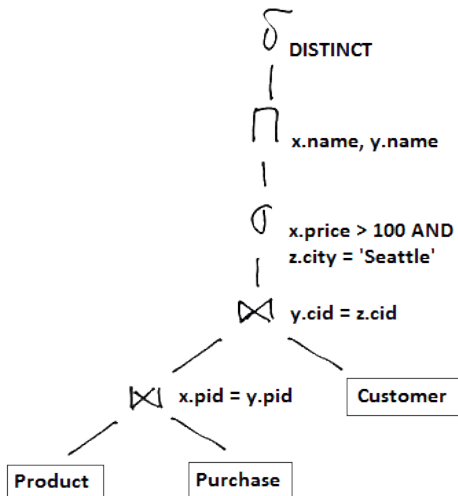
# Примеры плана запросов (1)



7

<sup>7</sup> Изображение взято с [https://docs.jboss.org/teiid/6.0/reference/en-US/html/federated\\_planning.html](https://docs.jboss.org/teiid/6.0/reference/en-US/html/federated_planning.html)

## Примеры плана запросов (2)



8

8

Изображение взято с [http://0agr.ru/wiki/index.php/Logical\\_Query\\_Plan\\_Optimization](http://0agr.ru/wiki/index.php/Logical_Query_Plan_Optimization)

# Проблемы:

- Планы могут быть очень разные по качеству!
- Планов может быть очень много!
- Планы редко когда можно переиспользовать.
- ...

Вычисление оптимального плана  $NP$ -трудная задача. Поэтому ищут просто хороший.

# Как искать? (1)

Стоимостная модель + эвристический метод:

- Генетические алгоритмы;
- Метод симуляции отжига;
- Метод восхождения к вершине;
- Метод муравьиной оптимизации;
- ...

На самом деле чаще всего имеется какая-то database-specific процедура перебора планов.

# Как искать? (2): иллюстрация

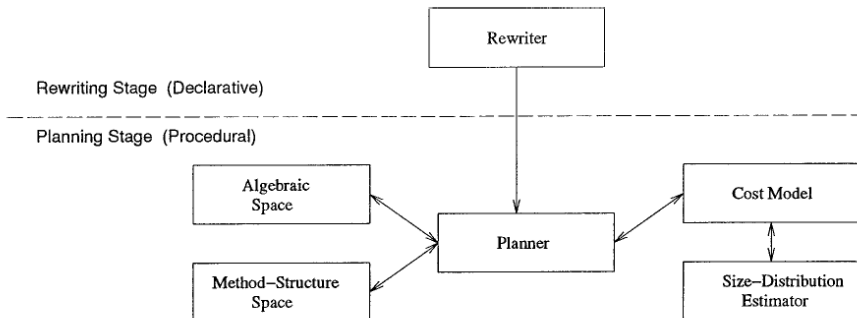


Figure 1. Query optimizer architecture.

9

<sup>9</sup> Изображение взято из [Ioannidis, 1996]

# Итераторная модель Volcano

Пусть у нас есть план, надо его запустить на выполнение.

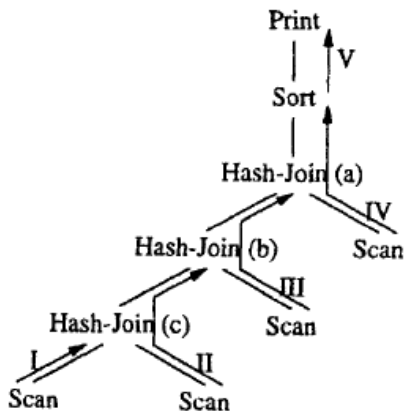


Figure 1. Left-deep query plan with plan phases

10

<sup>10</sup> Изображение взято из [Graefe, 1996]

# Итератор

```
1 class AbstractNode{
2     private:
3         AbstractNode* LChild;
4         AbstractNode* RChild;
5         void* ..._// internal state
6     public:
7         int Open();
8         int Close();
9         void* GetNext();
10 };
```

- Идея: строим дерево из таких итераторов;
- Его можно запускать, вызывая у верхнего оператора GetNext();
- Обычно, между операторами ходит не одна, а много записей (блочная модель Volcano).



# Примеры простых итераторов

Table I. Simplified iterator methods

Iterator	<i>Open</i>	<i>Next</i>	<i>Close</i>	Local state
Print	<i>open</i> input	call <i>next</i> on input; format the item on screen	<i>close</i> input	
Scan	open file	read next item	close file	open file descriptor
Select	<i>open</i> input	call <i>next</i> on input until an item qualifies	<i>close</i> input	
Hash join (without overflow resolution)	allocate hash directory; <i>open</i> left 'build' input; build hash table calling <i>next</i> on build input; <i>close</i> build input; <i>open</i> right 'probe' input	call <i>next</i> on probe input until a match is found	<i>close</i> probe input; deallocate hash directory	hash directory
Merge-join (without duplicates)	<i>open</i> both inputs	get <i>next</i> item from input with smaller key until a match is found	<i>close</i> both inputs	
Sort	<i>open</i> input; build all initial run files calling <i>next</i> on input; <i>close</i> input; merge run files until only one merge step is left	determine next output item; read new item from the correct run file	destroy remaining run files	merge heap; open file descriptors for run files

11

<sup>11</sup>Изображение взято из [Graefe, 1996]

Нужен минимальный набор:

- ( $\sigma$ ) Выборка — то, что содержится во WHERE:  $T1.X > 255$ ;
- ( $\bowtie$ ) Соединение — бывает:
  - во WHERE:  $T1.X = T2.Y$  и,
  - в FROM: `FROM Orders INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID`
- ( $\Pi$ ) Проекция — остается то, что содержится в SELECT: `SELECT Orders.OrderID, Customers.CustomerName`

Такой класс запросов называется SPJ запросы (Select, Project, Join).

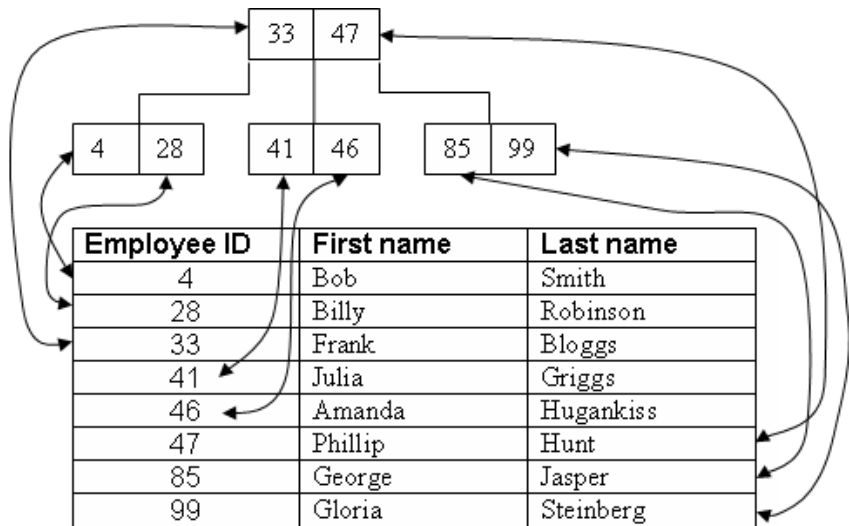
Это самый простой класс, есть еще операторы: агрегация, DISTINCT, TOP N, множественные операции (UNION, MINUS, ...), подзапросы, ...

Реализация выборки = методы доступа (access methods). Основные:

- Полный просмотр — медленный, но не требует ничего дополнительно, данные-то всяко есть;
- Просмотр по кластеризованному индексу — быстрый, но можно только по одному атрибуту;
- Просмотр с использованием индекса — быстрый, можно по всем атрибутам, но надо строить.

Сложные: например, по нескольким индексам, пересечение и вычитка.

# Напоминание про индекс на B-tree



12

<sup>12</sup>Изображение взято из [https://upload.wikimedia.org/wikipedia/en/0/03/Btree\\_index.PNG](https://upload.wikimedia.org/wikipedia/en/0/03/Btree_index.PNG)

# Реализация реляционной операции соединение

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
2	303	30	3
3	302	24	3

Таблица: rooms

Основные методы: Nested Loop, Sort-Merge, Hash-Join.

В следующих слайдах обращайте внимание на отсортированность атрибутов!

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1



# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2



# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2, 4-3

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2, 4-3

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2, 4-3

# Nested Loop

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2, 4-3, 5-3

# Sort-Merge

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
4	vasya	60	php	2
5	sasha	70	java	2
3	slava	30	java	3

Таблица: wages

id	name	screen	floor
1	401	14	4
2	302	24	3
3	303	30	3

Таблица: rooms

Выдача: 1-1

# Sort-Merge

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
4	vasya	60	php	2
5	sasha	70	java	2
3	slava	30	java	3

Таблица: wages

id	name	screen	floor
1	401	14	4
2	302	24	3
3	303	30	3

Таблица: rooms

Выдача: 1-1, 2-1

# Sort-Merge

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
4	vasya	60	php	2
5	sasha	70	java	2
3	slava	30	java	3

Таблица: wages

id	name	screen	floor
1	401	14	4
2	302	24	3
3	303	30	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2

# Sort-Merge

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
4	vasya	60	php	2
5	sasha	70	java	2
3	slava	30	java	3

Таблица: wages

id	name	screen	floor
1	401	14	4
2	302	24	3
3	303	30	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2, 4-3



# Sort-Merge

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
4	vasya	60	php	2
5	sasha	70	java	2
3	slava	30	java	3

Таблица: wages

id	name	screen	floor
1	401	14	4
2	302	24	3
3	303	30	3

Таблица: rooms

Выдача: 1-1, 2-1, 3-2, 4-3, 5-3

## Фаза хеширования

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Хеш: 1

Выдача:

## Фаза хеширования

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Хеш: 1, 3

Выдача:

## Фаза хеширования

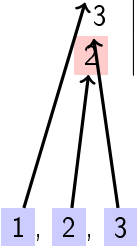
id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Хеш: 1, 2, 3  
Выдача:



## Фаза пробинга

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Хеш:

1, 2, 3

Выдача: 1-1

## Фаза пробинга

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Хеш:

1, 2, 3

Выдача: 1-1, 2-1

## Фаза пробинга

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Хеш: 1, 2, 3  
Выдача: 1-1, 2-1, 3-2

## Фаза пробинга

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Хеш: 1, 2, 3  
Выдача: 1-1, 2-1, 3-2, 4-3



## Фаза пробинга

id	name	wage	skill	roomid
1	ivan	80	c++	1
2	petr	50	c++	1
3	slava	30	java	3
4	vasya	60	php	2
5	sasha	70	java	2

Таблица: wages

id	name	screen	floor
1	401	14	4
3	303	30	3
2	302	24	3

Таблица: rooms

Хеш:

1, 2, 3

Выдача: 1-1, 2-1, 3-2, 4-3, 5-3

- Обычно стараются делать сразу, как только возможно;
- Часто прячут внутри других операторов;

id	name	wage	skill	exp
1	ivan	80	c++	5
2	petr	50	c++	3
3	slava	30	java	1
4	vasya	60	php	1
5	sasha	70	java	3
6	dasha	65	c++	3
7	katya	50	java	1
8	glasha	30	bash	2
9	oleg	20	php	3
10	boris	50	java	3

Таблица: wages

```
SELECT skill, avg(wage), exp FROM wages GROUP BY skill, exp
```

# Агрегация: как?

- Влоб: завести набор групп, сверяться каждый раз со всеми;
- Отсортировать по GROUP BY, а потом аккуратно пройти один раз.

Упражнения:

- Что выдаст запрос?
- Расписать как будет происходить вычисление.

# Замечание о порядках сортировки

- Они очень сильно помогают;
- Каждый раз при запросе пересортировывать — долго;
- Сделать пресортировку один раз, по этому атрибуту. Однако, на диске таблицу держать в отсортированном виде можно только по одному атрибуту.

# Замечание про операторы

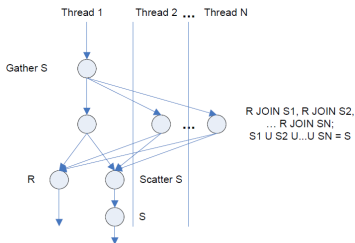
Операторы бывают:

- Блокирующие, например hash-join или сканирование с сортировкой;
- Не блокирующие.

А еще они могут ломать сортировку или нет:

- Sort-merge join сохраняет сортировку по обоим атрибутам;
- Hash join испортит по одному атрибуту.

Операторы можно параллелить:



13

Более подробно смотри [Taniar et al., 2008].



Гектор Гарсиа-Молина, Джеффри Д. Ульман, Дженнифер Уидом. Системы баз данных. Полный курс. ISBN 5-8459-0384-X; 2004 г.



Joseph M. Hellerstein, Michael Stonebraker, and James Hamilton. 2007. Architecture of a Database System. Found. Trends databases 1, 2 (February 2007), 141–259.



Yannis E. Ioannidis. 1996. Query optimization. ACM Comput. Surv. 28, 1 (March 1996), 121–123. DOI=<http://dx.doi.org/10.1145/234313.234367>



Kirill Smirnov and George Chernishev. Benchmarking Inter and Intra Operator Parallelism on Contemporary Desktop Hardware. In Proc. of SYRCoDIS 2011, p. 62–67, 2011.



Goetz Graefe. 1996. Iterators, schedulers, and distributed-memory parallelism. Softw. Pract. Exper. 26, 4 (April 1996), 427–452.  
DOI=[http://dx.doi.org/10.1002/\(SICI\)1097-024X\(199604\)26:4<427::AID-SPE20>3.3.CO;2-8](http://dx.doi.org/10.1002/(SICI)1097-024X(199604)26:4<427::AID-SPE20>3.3.CO;2-8)



David Taniar, Clement H. C. Leung, Wenny Rahayu, and Sushant Goel. 2008. High Performance Parallel Database Processing and Grid Databases. Wiley Publishing.



Raghu Ramakrishnan and Johannes Gehrke. 2000. Database Management Systems (2nd ed.). Osborne/McGraw-Hill, Berkeley, CA, USA.