

Лекция 6: Индексы. Выполнение запросов

Логические операторы и NULL

A AND B

A	B	Результат
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
TRUE	NULL	NULL
FALSE	FALSE	FALSE
FALSE	NULL	FALSE
NULL	NULL	NULL

A OR B

A	B	Результат
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	NULL	TRUE
FALSE	FALSE	FALSE
FALSE	NULL	NULL
NULL	NULL	NULL

NOT A

A	Результат
TRUE	FALSE
FALSE	TRUE
NULL	NULL

1. Индексы



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.).

Индекс SQL — список всех значений в группе из одного или нескольких столбцов, упорядоченный в некотором приемлемом для данного типа данных смысле (например, в порядке возрастания для чисел или в алфавитном порядке для символьных строк).

Каждое значение имеет указатель на строку в таблице, где это значение встречается.

STUDENT

StudID	GroupID	Name
14	51	Ivan Petrov
23	53	Vladimir Ivanov
18	54	Eugene Serov
13	55	Gennady Surikov
25	56	Nikolay Petrov
25	58	Svetlana Ivanova
25	59	AntonyFedorov
18	63	Petr Menchikov
27	67	Gennady Klovok
23	69	Serge Vidov
18	72	Roman Klever
13	73	Pavel Borisov

Пример

```
SELECT * FROM STUDENT  
WHERE StudID = 18;
```

Как найти нужные строки?

StudID	GroupID	Name
14	51	Ivan Petrov
23	53	Vladimir Ivanov
18	54	Eugene Serov
13	55	Gennady Surikov
25	56	Nikolay Petrov
25	58	Svetlana Ivanova
25	59	AntonyFedorov
18	63	Petr Menchikov
27	67	Gennady Klovov
23	69	Serge Vidov
18	72	Roman Klever
13	73	Pavel Borisov

Пример

```
SELECT * FROM STUDENT  
WHERE StudID = 18;
```

**Перебираем все по
очереди!**

StudID	GroupID	Name
14	51	Ivan Petrov
23	53	Vladimir Ivanov
18	54	Eugene Serov
13	55	Gennady Surikov
25	56	Nikolay Petrov
25	58	Svetlana Ivanova
25	59	AntonyFedorov
18	63	Petr Menchikov
27	67	Gennady Klovok
23	69	Serge Vidov
18	72	Roman Klever
13	73	Pavel Borisov

Пример

```
SELECT * FROM STUDENT  
WHERE StudID = 18;
```

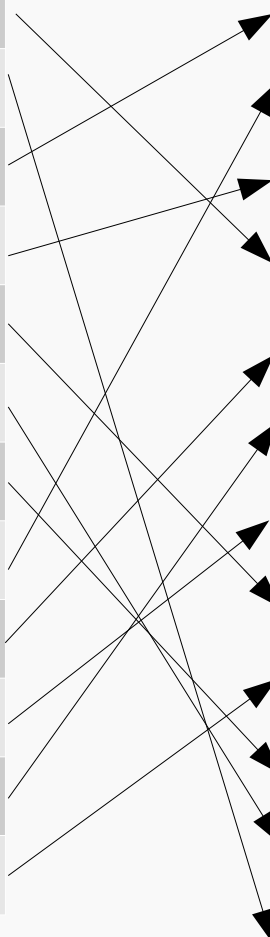
Может быть долго!

Перебираем все по очереди!

StudID	GroupID	Name
14	51	Ivan Petrov
23	53	Vladimir Ivanov
18	54	Eugene Serov
13	55	Gennady Surikov
25	56	Nikolay Petrov
25	58	Svetlana Ivanova
25	59	AntonyFedorov
18	63	Petr Menchikov
27	67	Gennady Klovok
23	69	Serge Vidov
18	72	Roman Klever
13	73	Pavel Borisov

Индексы

index	StudID	GroupID	Name
13	14	51	Ivan Petrov
13	23	53	Vladimir Ivanov
14	18	54	Eugene Serov
18	13	55	Gennady Surikov
18	25	56	Nikolay Petrov
18	25	58	Svetlana Ivanova
23	25	59	AntonyFedorov
23	18	63	Petr Menchikov
25	27	67	Gennady Klovov
25	23	69	Serge Vidov
25	18	72	Roman Klever
27	13	73	Pavel Borisov



The diagram illustrates the mapping from the 'index' column to the 'StudID' column. Arrows indicate the following connections:

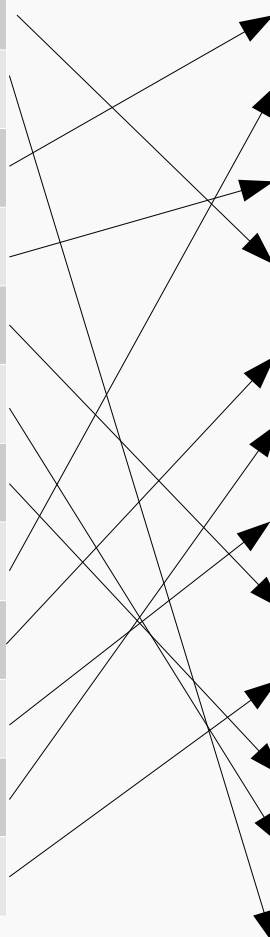
- Index 13 maps to StudID 14 and StudID 23.
- Index 14 maps to StudID 18.
- Index 18 maps to StudID 13 and StudID 25.
- Index 23 maps to StudID 25 and StudID 18.
- Index 25 maps to StudID 27, StudID 23, and StudID 18.
- Index 27 maps to StudID 13.

Индексы

SELECT * FROM STUDENT

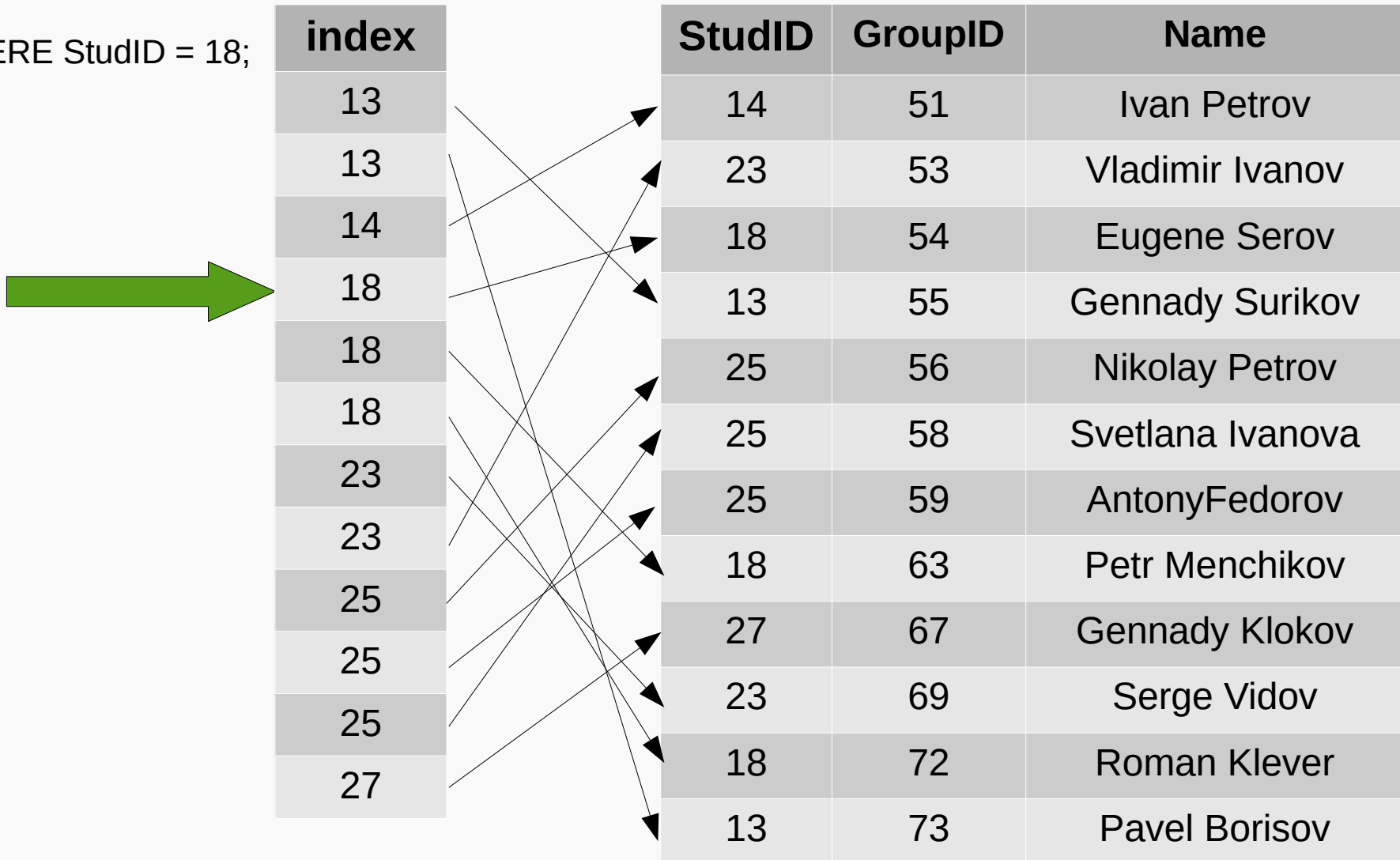
WHERE StudID = 18;

index	StudID	GroupID	Name
13	14	51	Ivan Petrov
13	23	53	Vladimir Ivanov
14	18	54	Eugene Serov
18	13	55	Gennady Surikov
18	25	56	Nikolay Petrov
18	25	58	Svetlana Ivanova
23	25	59	AntonyFedorov
23	18	63	Petr Menchikov
25	27	67	Gennady Klovok
25	23	69	Serge Vidov
25	18	72	Roman Klever
27	13	73	Pavel Borisov



Индексы

SELECT * FROM STUDENT
WHERE StudID = 18;



Индексы

SELECT * FROM STUDENT

WHERE StudID = 18;



index	StudID	GroupID	Name
13	14	51	Ivan Petrov
13	23	53	Vladimir Ivanov
14	18	54	Eugene Serov
18	13	55	Gennady Surikov
18	25	56	Nikolay Petrov
18	25	58	Svetlana Ivanova
23	25	59	AntonyFedorov
23	18	63	Petr Menchikov
25	27	67	Gennady Klovok
25	23	69	Serge Vidov
25	18	72	Roman Klever
27	13	73	Pavel Borisov

Индексы

SELECT * FROM STUDENT

WHERE StudID = 18;



index	StudID	GroupID	Name
13	14	51	Ivan Petrov
13	23	53	Vladimir Ivanov
14	18	54	Eugene Serov
18	13	55	Gennady Surikov
18	25	56	Nikolay Petrov
18	25	58	Svetlana Ivanova
23	25	59	AntonyFedorov
23	18	63	Petr Menchikov
25	27	67	Gennady Klovok
25	23	69	Serge Vidov
25	18	72	Roman Klever
27	13	73	Pavel Borisov

Индексы

SELECT * FROM STUDENT

WHERE StudID = 18;



index	StudID	GroupID	Name
13	14	51	Ivan Petrov
13	23	53	Vladimir Ivanov
14	18	54	Eugene Serov
18	13	55	Gennady Surikov
18	25	56	Nikolay Petrov
18	25	58	Svetlana Ivanova
23	25	59	AntonyFedorov
23	18	63	Petr Menchikov
25	27	67	Gennady Klovok
25	23	69	Serge Vidov
25	18	72	Roman Klever
27	13	73	Pavel Borisov

Индексы

SELECT * FROM STUDENT

WHERE StudID = 18;



index	StudID	GroupID	Name
13	14	51	Ivan Petrov
13	23	53	Vladimir Ivanov
14	18	54	Eugene Serov
18	13	55	Gennady Surikov
18	25	56	Nikolay Petrov
18	25	58	Svetlana Ivanova
23	25	59	AntonyFedorov
23	18	63	Petr Menchikov
25	27	67	Gennady Klovok
25	23	69	Serge Vidov
25	18	72	Roman Klever
27	13	73	Pavel Borisov

Индексы

SELECT * FROM STUDENT

WHERE StudID = 18;



index	StudID	GroupID	Name
13	14	51	Ivan Petrov
13	23	53	Vladimir Ivanov
14	18	54	Eugene Serov
18	13	55	Gennady Surikov
18	25	56	Nikolay Petrov
18	25	58	Svetlana Ivanova
23	25	59	AntonyFedorov
23	18	63	Petr Menchikov
25	27	67	Gennady Klovok
25	23	69	Serge Vidov
25	18	72	Roman Klever
27	13	73	Pavel Borisov

Индексы

SELECT * FROM STUDENT

WHERE StudID = 18;

index	StudID	GroupID	Name
13	14	51	Ivan Petrov
13	23	53	Vladimir Ivanov
14	18	54	Eugene Serov
18	13	55	Gennady Surikov
18	25	56	Nikolay Petrov
18	25	58	Svetlana Ivanova
23	25	59	AntonyFedorov
23	18	63	Petr Menchikov
25	27	67	Gennady Klovok
25	23	69	Serge Vidov
25	18	72	Roman Klever
27	13	73	Pavel Borisov



Индексы работают **неявно**:

- При выполнении запроса СУБД определяет, какие индексы нужно использовать.
- В рамках того или иного запроса СУБД может не использовать индекс.

Правила работы с индексами

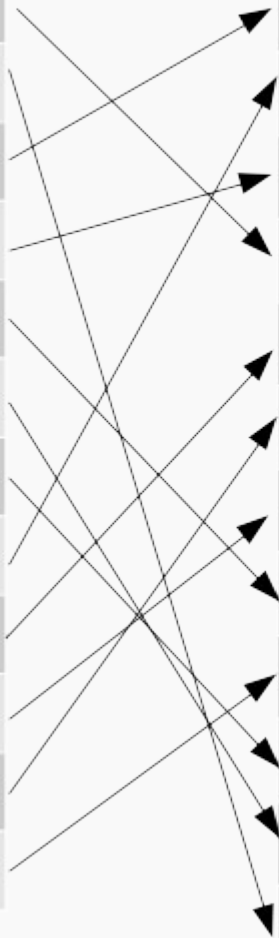
Индексы создаются по таблицам для ускорения операций, включающих:

- предложения WHERE и JOIN
- определение значений MIN() или MAX() по индексированному столбцу;
- сортировку и группировку столбцов таблицы

Недостатки индексов (1)

- Индекс занимает место в памяти:

index	StudID	GroupID	Name
13	14	51	Ivan Petrov
13	23	53	Vladimir Ivanov
14	18	54	Eugene Serov
18	13	55	Gennady Surikov
18	25	56	Nikolay Petrov
18	25	58	Svetlana Ivanova
23	25	59	AntonyFedorov
23	18	63	Petr Menchikov
25	27	67	Gennady Klovok
25	23	69	Serge Vidov
25	18	72	Roman Klever
27	13	73	Pavel Borisov



Недостатки индексов (2)

- При изменении/удалении содержимого индексируемого столбца/при добавлении новой строки индекс необходимо обновлять. Эти действия замедляют операции.

Недостатки индексов (2)

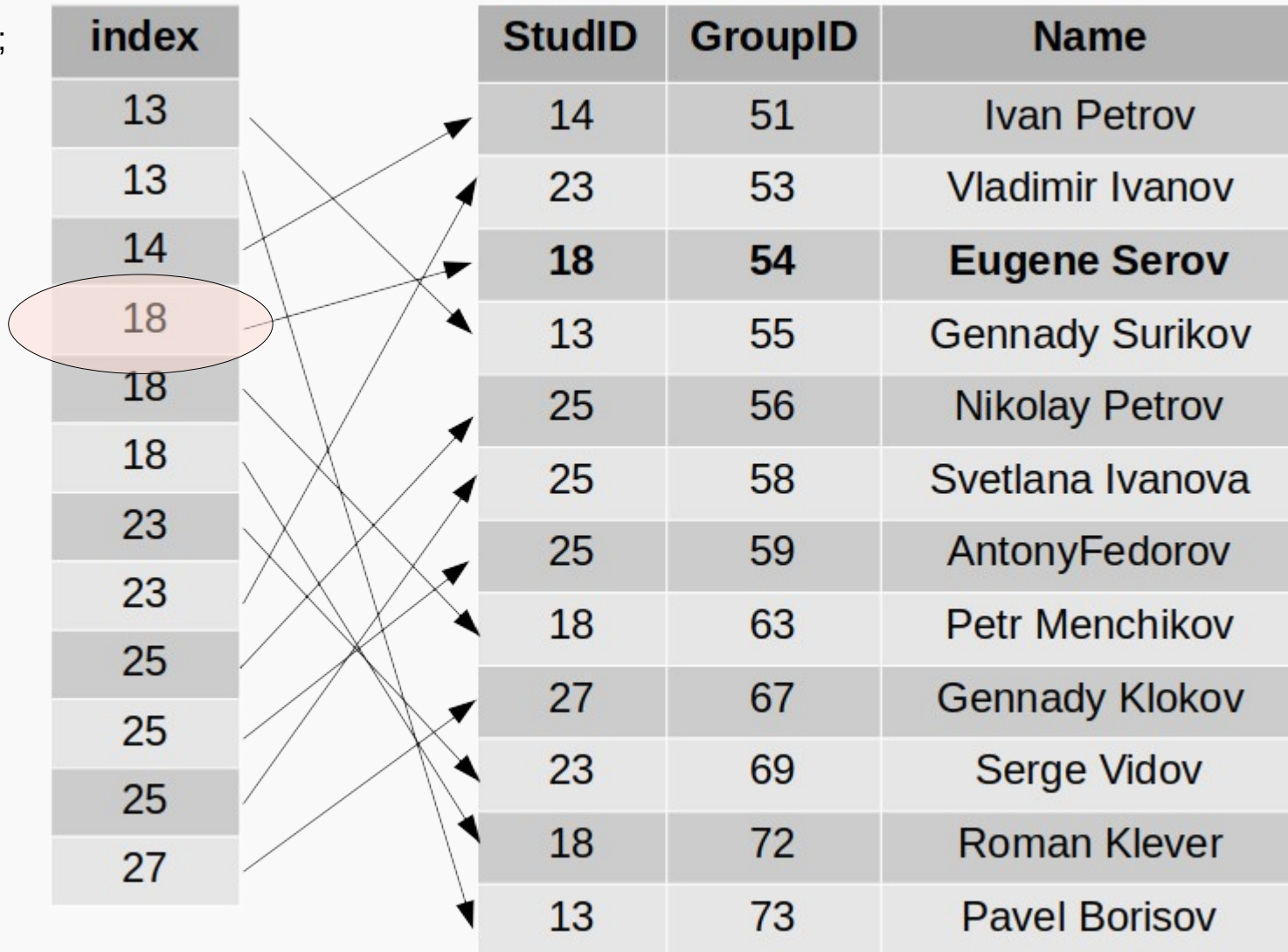
```
DELETE FROM STUDENT  
WHERE GroupID = 54;
```

StudID	GroupID	Name
14	51	Ivan Petrov
23	53	Vladimir Ivanov
18	54	Eugene Serov
13	55	Gennady Surikov
25	56	Nikolay Petrov
25	58	Svetlana Ivanova
25	59	AntonyFedorov
18	63	Petr Menchikov
27	67	Gennady Klovok
23	69	Serge Vidov
18	72	Roman Klever
13	73	Pavel Borisov

Недостатки индексов (2)

DELETE FROM STUDENT
WHERE GroupID = 54;

index	StudID	GroupID	Name
13	14	51	Ivan Petrov
13	23	53	Vladimir Ivanov
14	18	54	Eugene Serov
18	13	55	Gennady Surikov
18	25	56	Nikolay Petrov
18	25	58	Svetlana Ivanova
23	25	59	AntonyFedorov
23	18	63	Petr Menchikov
25	27	67	Gennady Klovov
25	23	69	Serge Vidov
25	18	72	Roman Klever
27	13	73	Pavel Borisov



Недостатки индексов (3)

- Индексы неэффективны, если в таблице мало строк:

RAM

StudID	GroupID	Name
14	51	Ivan Petrov
23	53	Vladimir Ivanov
18	54	Eugene Serov
13	55	Gennady Surikov
25	56	Nikolay Petrov
25	58	Svetlana Ivanova
25	59	AntonyFedorov
18	63	Petr Menchikov
27	67	Gennady Klovok
23	69	Serge Vidov
18	72	Roman Klever
13	73	Pavel Borisov

Недостатки индексов (4)

- Индексы могут быть неэффективными, если по условию выбираются большие объемы данных.

- Какие операции будут применяться к таблицам: запрос данных или обновление таблицы?
- Какие столбцы и как часто будут использоваться в предикатах?
- Как часто столбцы таблиц будут использоваться в соединениях (join)?

Создание индексов (1)

```
CREATE INDEX index_name  
ON table_name (column_name);
```

```
CREATE INDEX index_name  
ON table_name (column1_name, column2_name);
```

Создание индексов (2)

```
CREATE INDEX index_name ON table_name USING  
btree(column1, column2);
```

B-tree

- Дерево — связный ациклический граф.
- Связность — означает наличие путей между любой парой вершин;
- Ацикличность — отсутствие циклов и то, что между парами вершин имеется только по одному пути.

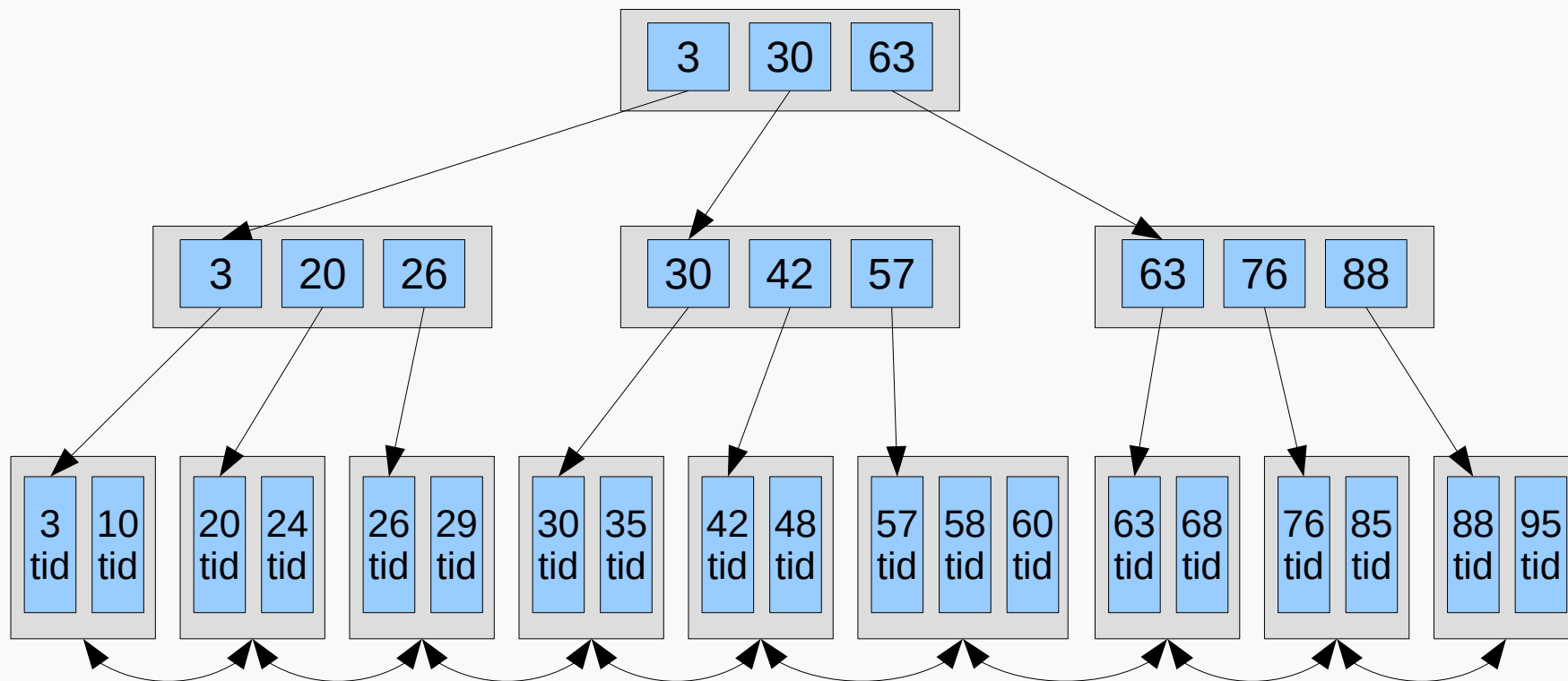
B-tree

- В-дерево — структура данных, дерево поиска. С точки зрения логического представления — сбалансированное, сильно ветвистое дерево.
- Сбалансированность — длина любых двух путей от корня до листьев различается не более, чем на единицу.
- Ветвистость дерева — свойство каждого узла дерева ссылаться на большое число узлов-потомков.

B-tree index

- B-tree (balanced tree index) — индекс сгруппированный по листьям сбалансированного дерева

B-tree индекс



tid — tuple identifier, идентификатор записи (номер блока, индекс внутри блока).

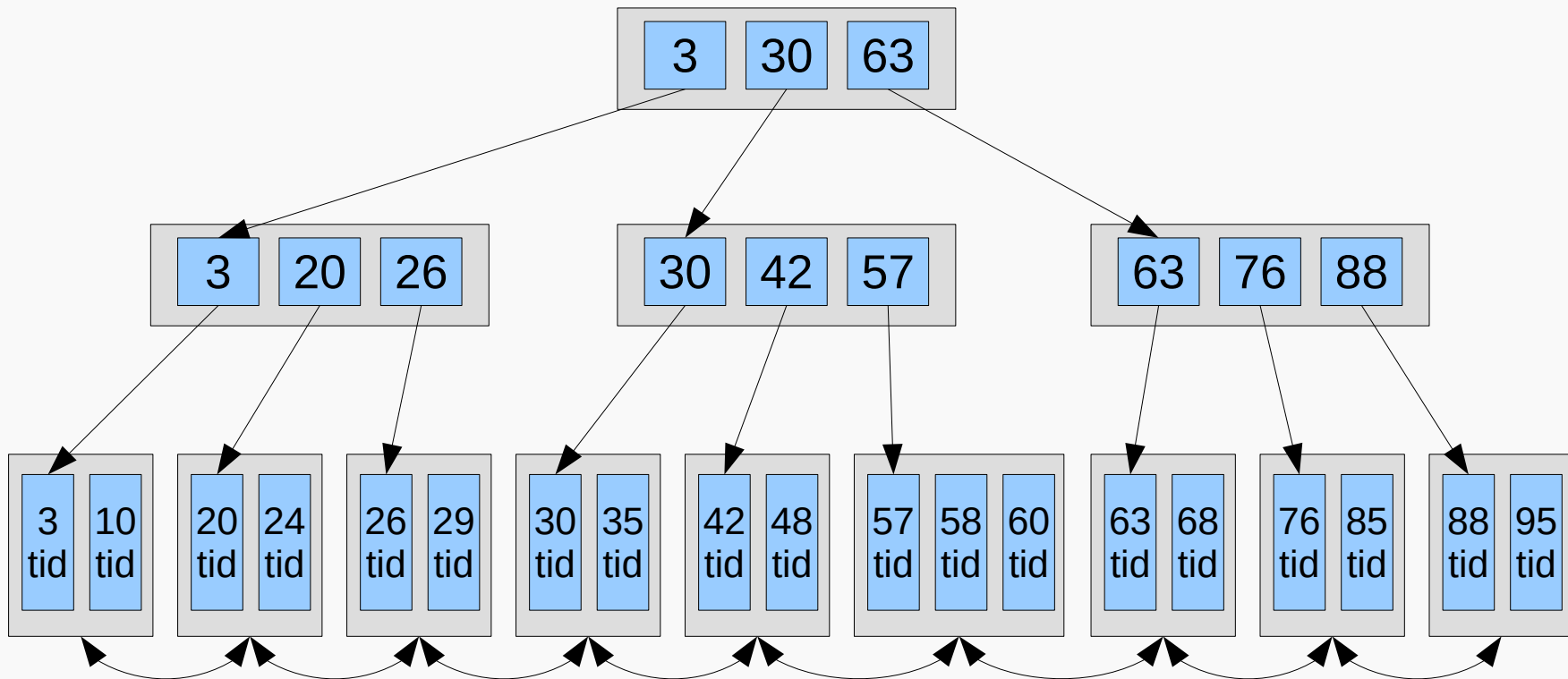
Пример

STUDENT

StudID	Name	GroupID
3	Ivan Petrov	1
20	Vasily Ivanov	1
29	Vlad Egorov	1
35	Serge Kirov	2
24	Kirill Ivanov	2
95	Ivan Vladov	2
76	Ivan Maslov	3
...

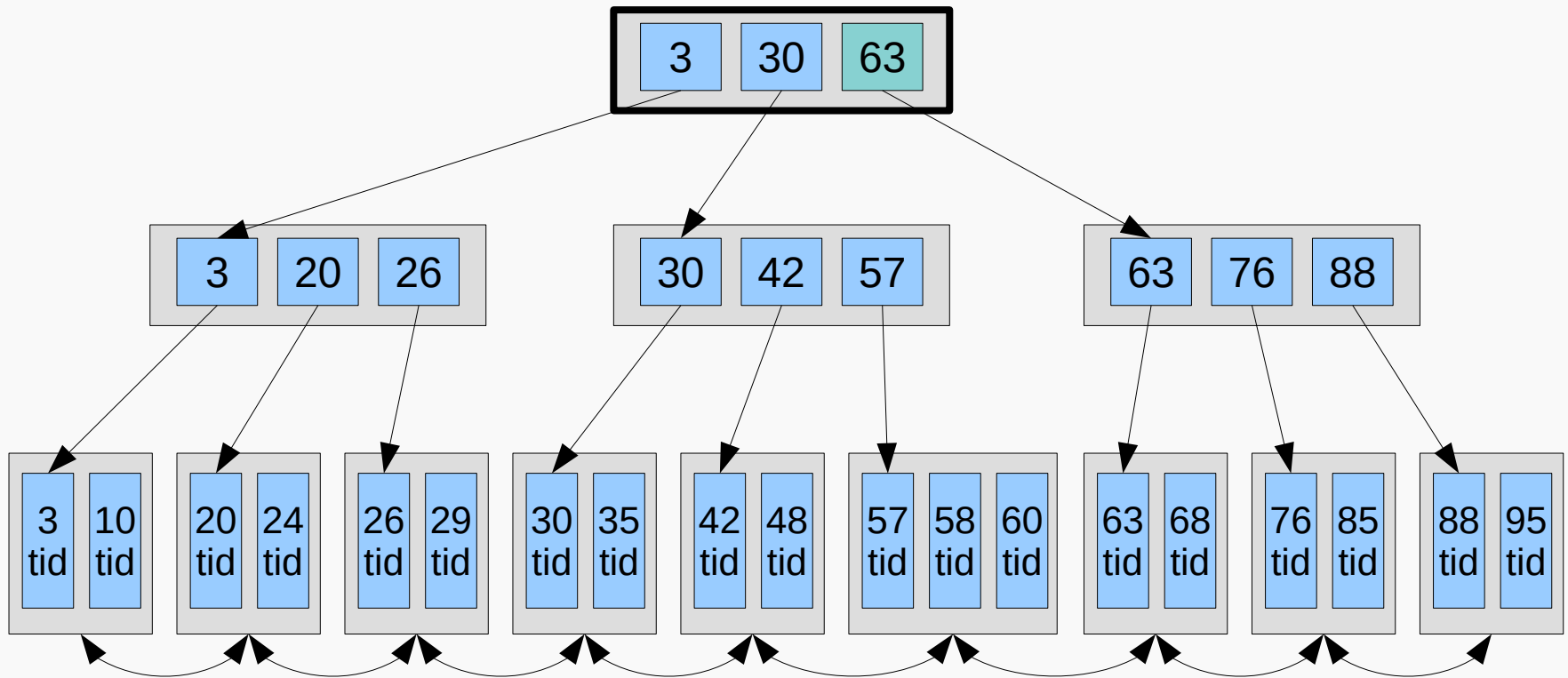
B-tree index

2 типа вершин: внутренние (ссылки на дочерние стр. и листовые страницы (ключ + tid)).



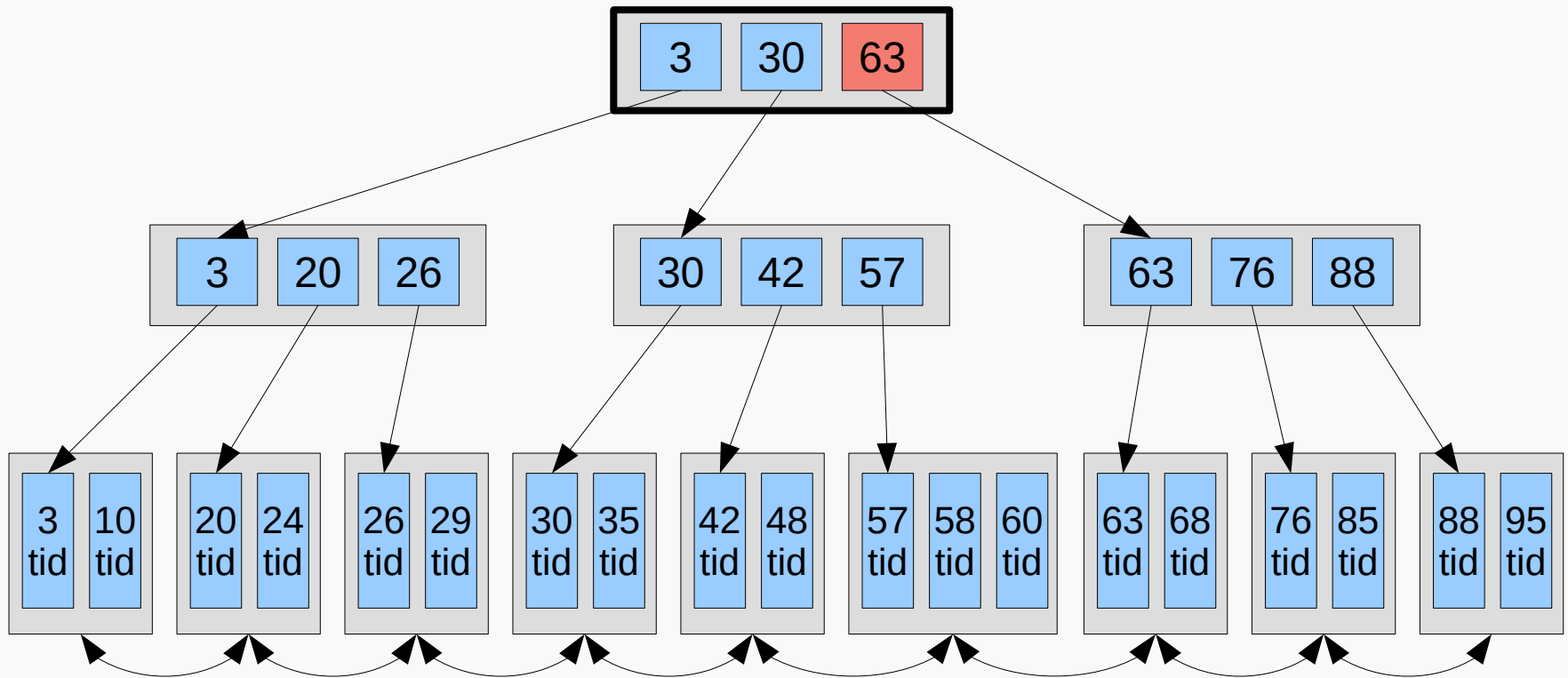
B-tree index

SELECT * FROM STUDENT
WHERE StudID = 85;



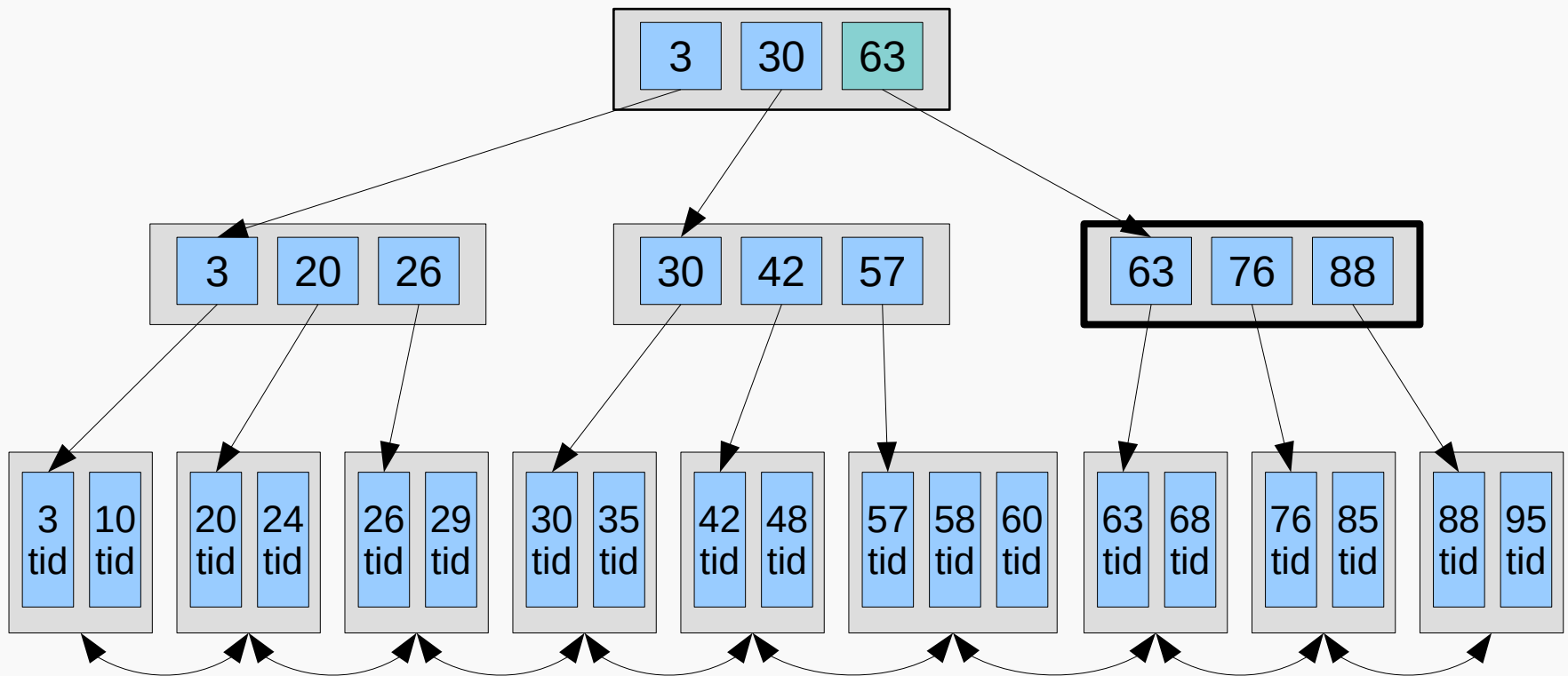
B-tree index

SELECT * FROM STUDENT
WHERE StudID = 85;



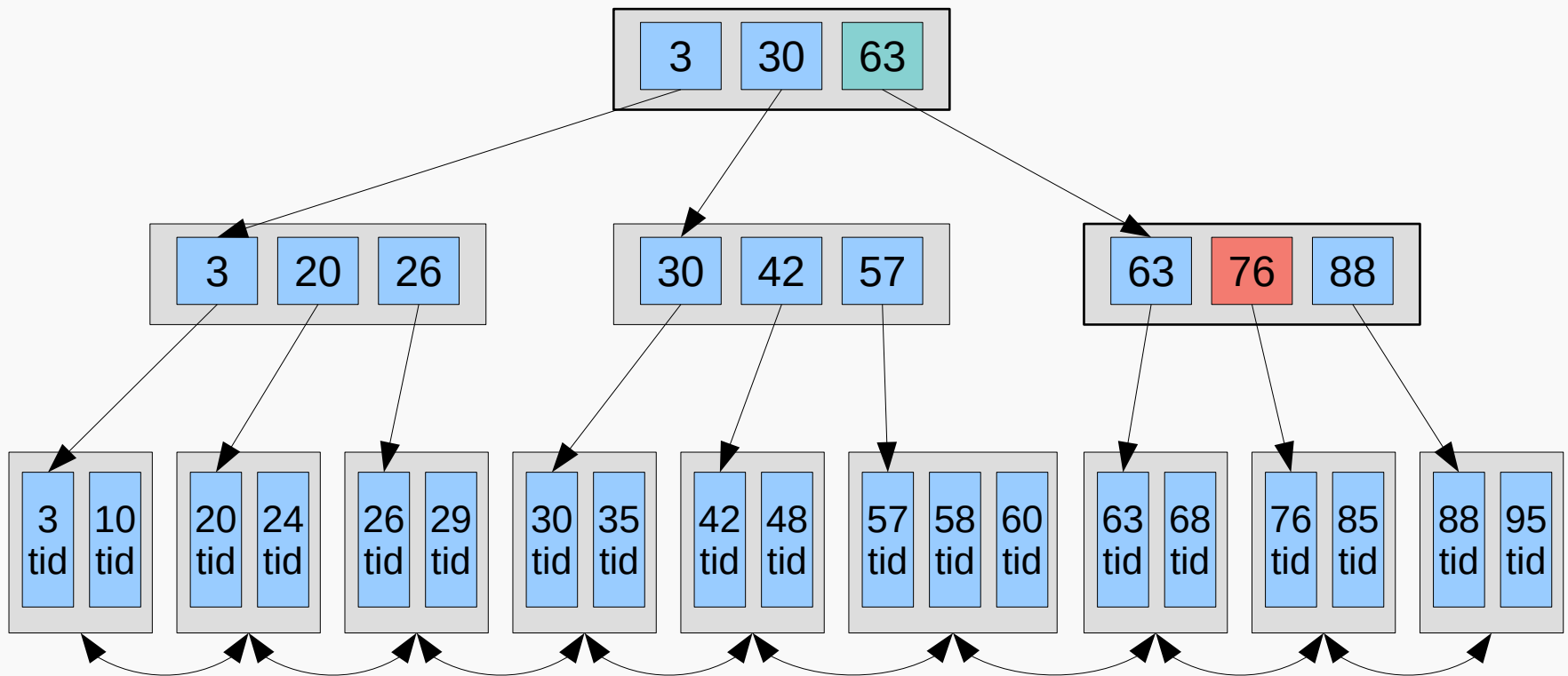
B-tree index

SELECT * FROM STUDENT
WHERE StudID = 85;



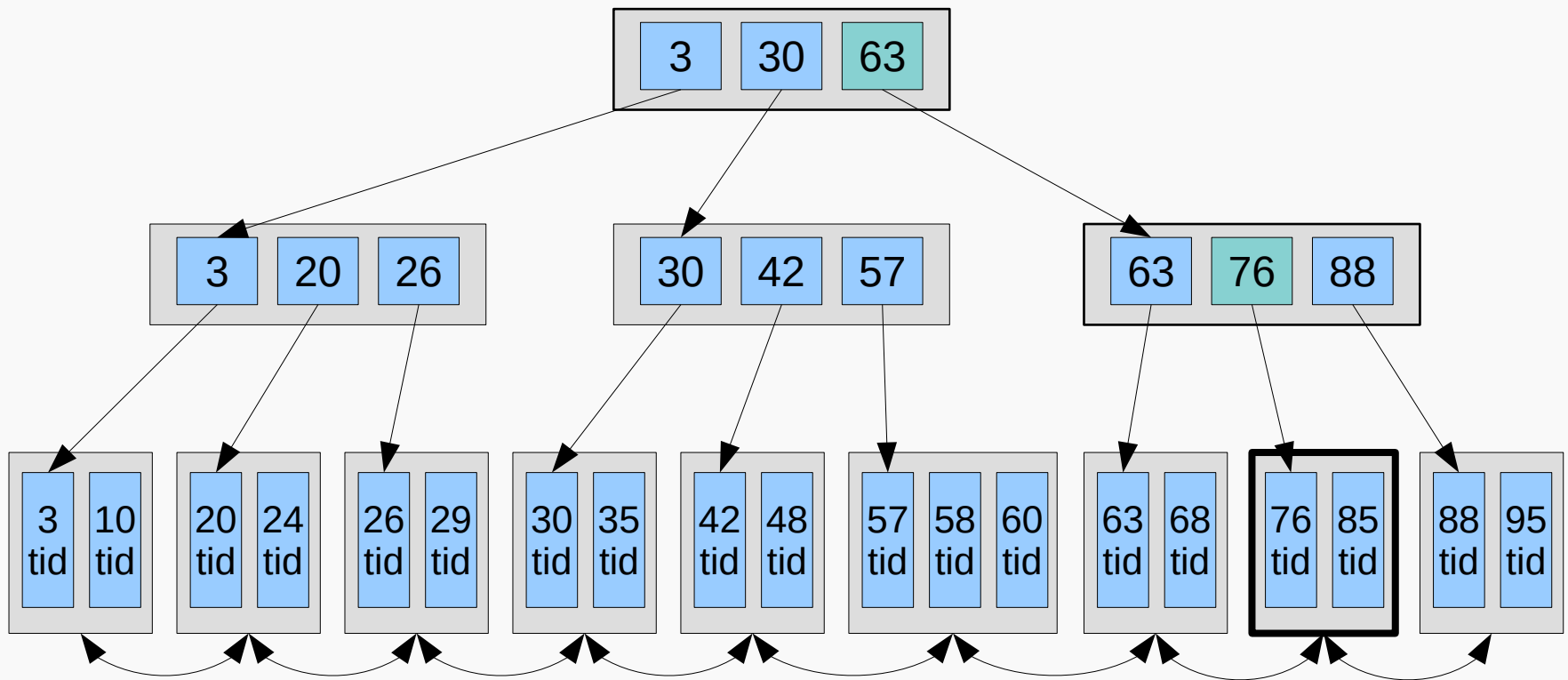
B-tree index

SELECT * FROM STUDENT
WHERE StudID = 85;



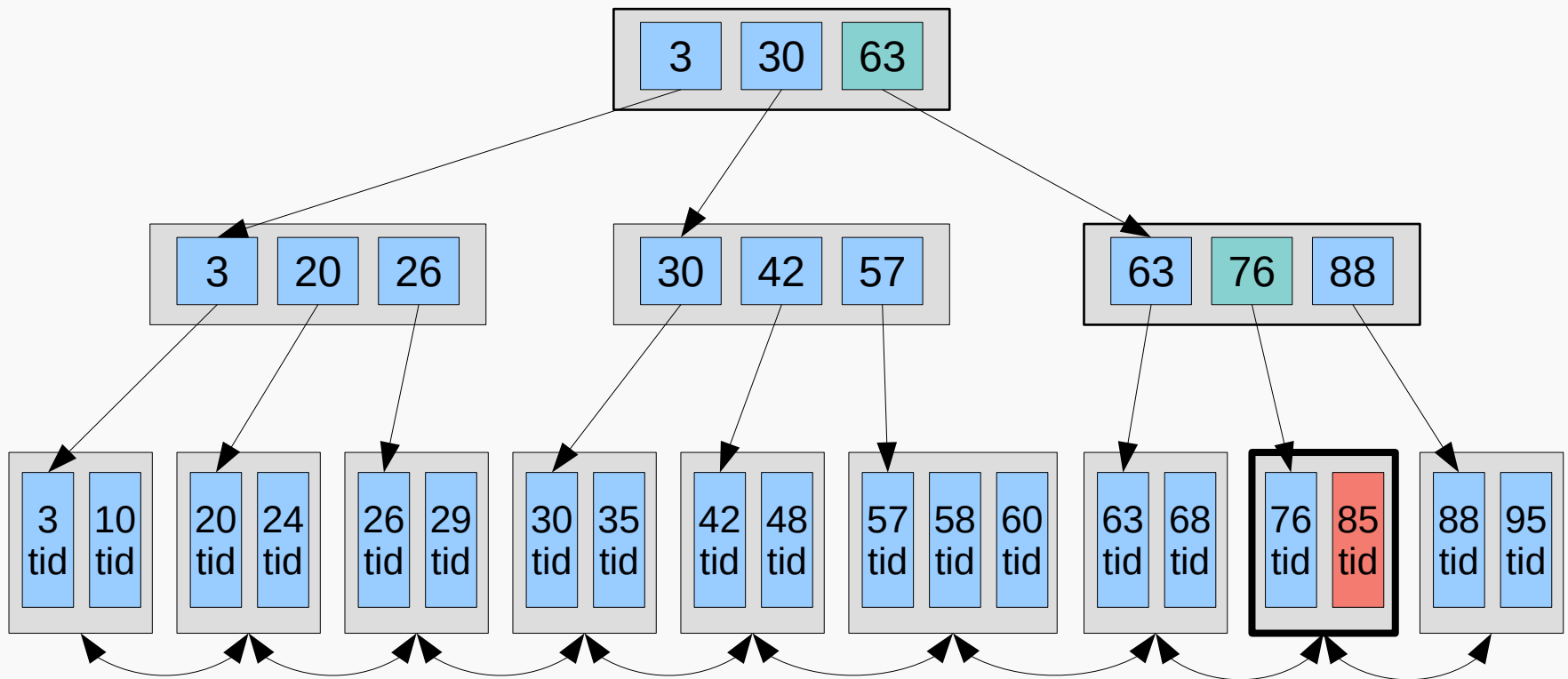
B-tree index

SELECT * FROM STUDENT
WHERE StudID = 85;



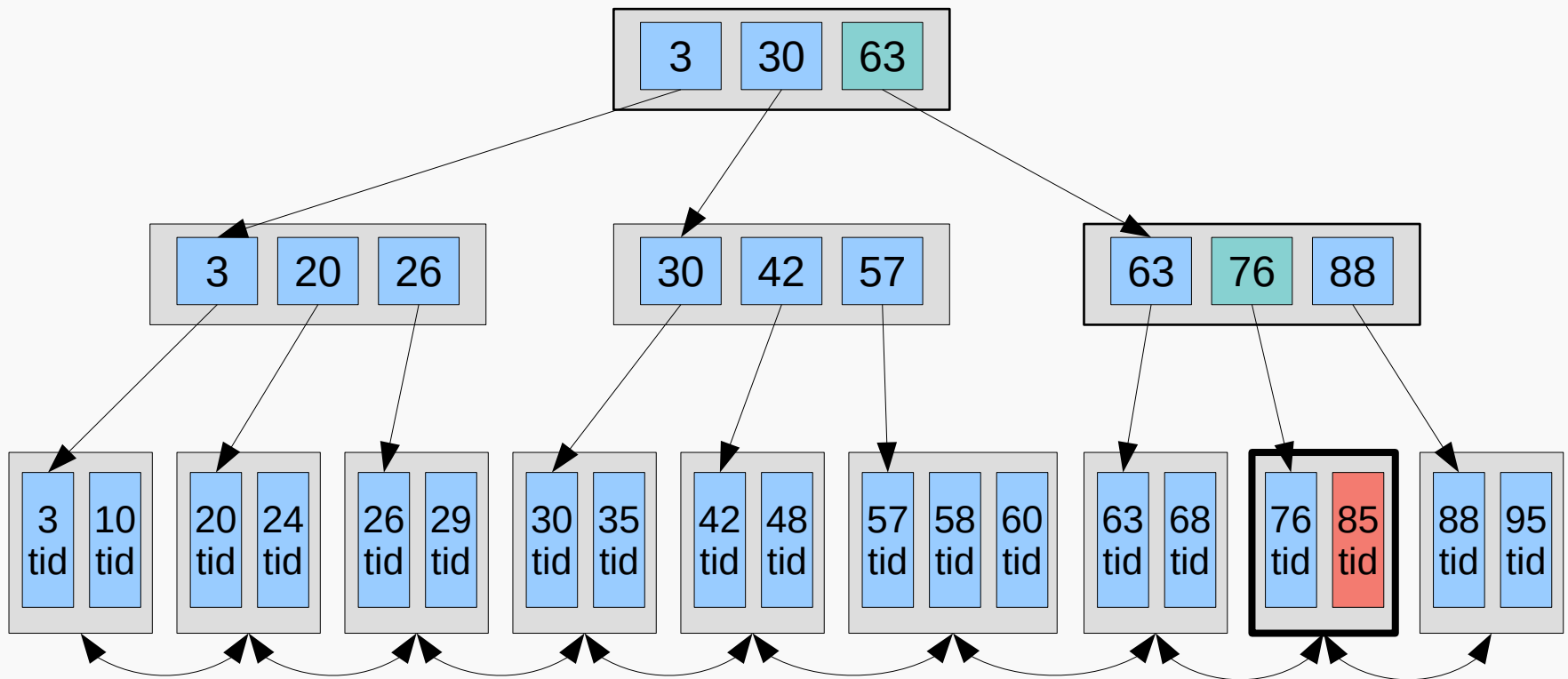
B-tree index

SELECT * FROM STUDENT
WHERE StudID = 85;



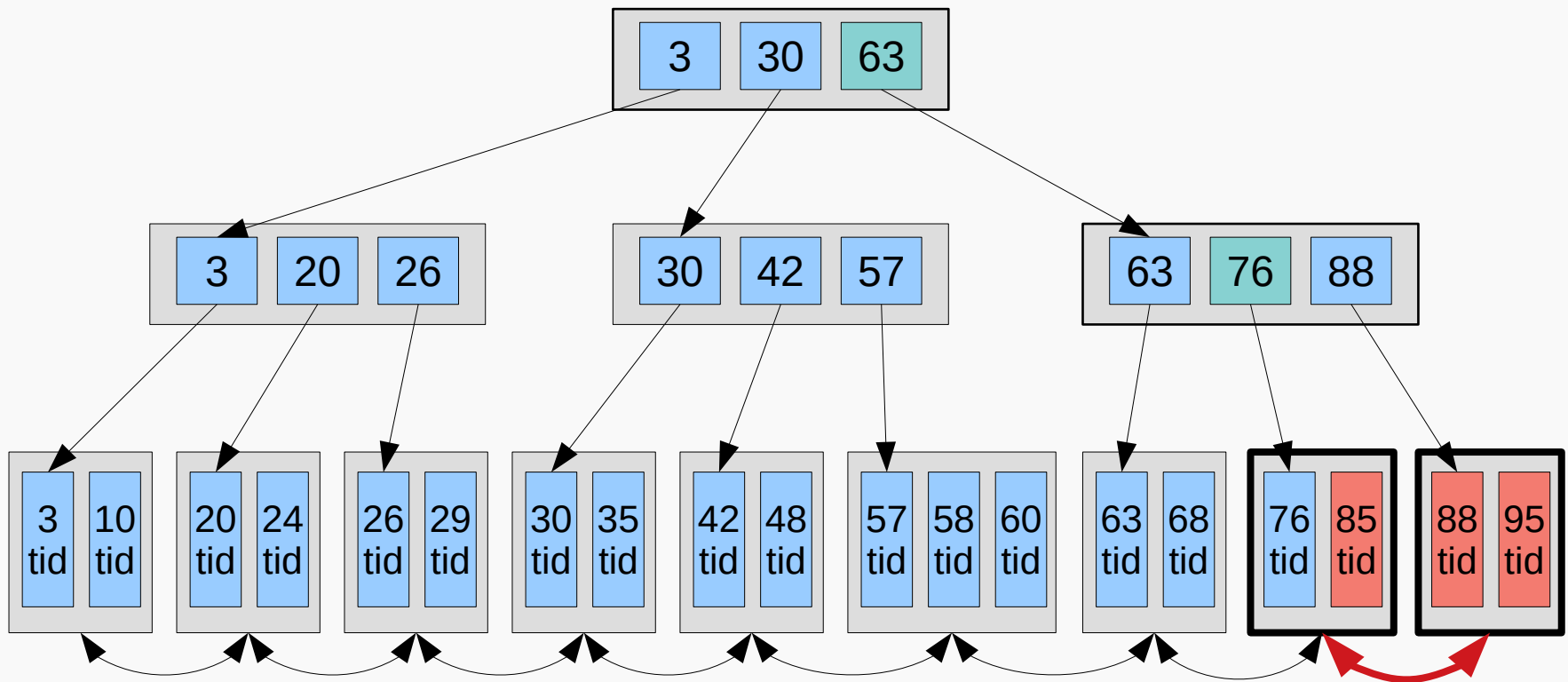
B-tree index

SELECT * FROM STUDENT
WHERE StudID \geq 85;



B-tree index

SELECT * FROM STUDENT
WHERE StudID \geq 85;



B-tree index

- Значения (ключи) внутри каждого узла отсортированы.
- Дерево сбалансировано ключи равномерно распределены по узлам, что позволяет минимизировать количество переходов.
- Полезен при использовании совместно с $=$, $>$, \geq , $<$, \leq , BETWEEN и подобными операторами.

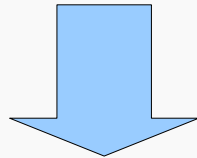
Hash-index

Для построения такого индекса используется хэш-функция

Хэш-функция — функция для преобразования входных данных в результирующие данные фиксированного формата.

Hash-index

StudID	GroupID	Name	Surname
23	2	Ivan	Petrov
34	2	Vasily	Ivanov
34543	4	Georgy	Simonov
234	4	Vladimir	Egorov
2355	5	Egor	Sidorov



Hash_Function(23, 2)

Hash_Function(34, 2)

Hash_Function(34543, 4)

Hash_Function(234, 4)

Hash_Function(2355, 5)

Hash_Res

234543534

574543545

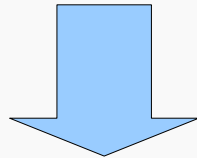
235456546

345734543

345743534

Hash-index

StudID	GroupID	Name	Surname
23	2	Ivan	Petrov
34	2	Vasily	Ivanov
34543	4	Georgy	Simonov
234	4	Vladimir	Egorov
2355	5	Egor	Sidorov



Hash_Function(23, 2)

Hash_Function(34, 2)

Hash_Function(34543, 4)

Hash_Function(234, 4)

Hash_Function(2355, 5)

Hash_Res

234543534

574543545

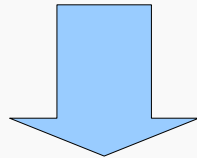
235456546

345734543

345743534

Hash-index

StudID	GroupID	Name	Surname
23	2	Ivan	Petrov
34	2	Vasily	Ivanov
34543	4	Georgy	Simonov
234	4	Vladimir	Egorov
2355	5	Egor	Sidorov



	Hash_Res
Hash_Function(23, 2)	2345
Hash_Function(34, 2)	3423
Hash_Function(34543, 4)	3457
Hash_Function(234, 4)	
Hash_Function(2355, 5)	

Hash-index

	Hash_Res	Memory
Hash_Function(23, 2)	2345	tid1+hv, tid3+hv
Hash_Function(34, 2)	3423	tid2+hv
Hash_Function(34543, 4)	3457	tid4+hv, tid5+hv
Hash_Function(234, 4)		
Hash_Function(2355, 5)		

tid	StudID	GroupID	Name	Surname
tid1	23	2	Ivan	Petrov
tid2	34	2	Vasily	Ivanov
tid3	34543	4	Georgy	Simonov
tid4	234	4	Vladimir	Egorov
tid5	2355	5	Egor	Sidorov

Hash-index

- Очень эффективны, когда используется прямое сравнение (на attr — hash index):

... WHERE attr = 1

- Но индекс не будет применен, если:

... WHERE attr = 1 OR A=10

Индексы в PostgreSQL

- GiST — обобщенное дерево поиска.
- GIN — обобщенный инвертированный индекс (для ускорения полнотекстового поиска).

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.