Введение в реляционные базы данных

Лекция 8: Индексы

Артем Толканев

November 13, 2024

Индексы

Табличный **индекс** - копия определенного подмножества атрибутов отношения, организованная специально для эффективного поиска кортежей таблицы.

СУБД гарантирует синхронизацию данных между таблицами и индексами

Чтобы получить данные об определенном объекте с заданным идентификатором, СУБД производит поиск по объекту бд, называемым **индексом**, чтобы определить в каком файле и какой странице памяти находится соответствующая запись об этом объекте. Без **индексов** пришлось бы считывать все страницы отношений, для поиска только определенных записей.

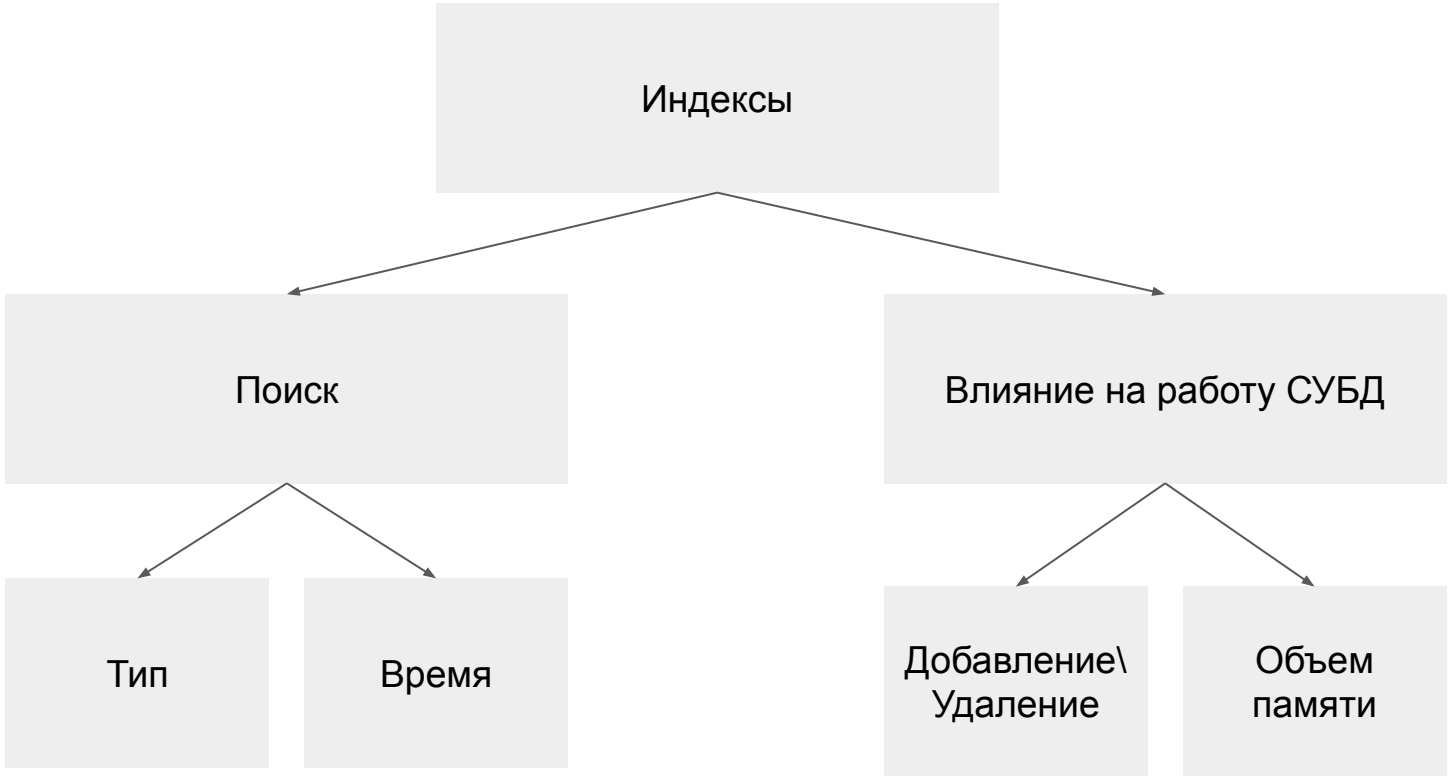
Хранить записи в определенном порядке - не решение проблемы. Потому что поиск может осуществляться по другому атрибуту отношения.

Некоторые бд хранят записи, отсортированные по кластерному индексы - например MS SQL

Некоторые бд хранят записи, независимо от какого-то индекса - пример Postgres

Концептуально, можно начать с рассмотрение индексов:

- Упорядоченные
- На основе хеш-таблиц


Индексы
Поиск
Влияние на работу СУБД
Тип
Время
Добавление\
Удаление
Объем
памяти

Индексная запись:

Значение ключа поиска	+	Указатель на запись в отношение (страница + смещение внутри страницы до записи)

Индексная запись:

Значение ключа поиска + Указатель на запись в отношение (страница + смещение внутри страницы до записи)

Сами значения записи

Индексная запись:

Значение ключа поиска + Указатель на запись в отношение (страница + смещение внутри страницы до записи)

Сами значения записи

Если значения в блоке памяти отсортированы по ключу поиска данного индекса, то указатель на страницу

Индексная запись:

Значение ключа поиска

+

Если в отношении есть кластерный ключ, то указатель на кластерный ключ необходимой записи

Указатель на запись в отношение (страница + смещение внутри страницы до записи)

Сами значения записи

Если значения в блоке памяти отсортированы по ключу поиска данного индекса, то указатель на страницу

Индексная запись:

Если в отношении есть кластерный ключ, то указатель на кластерный ключ необходимой записи

Значение ключа поиска

+

Указатель на запись в отношение (страница + смещение внутри страницы до записи)

Значение ключа поиска (несколько колонок поиска)

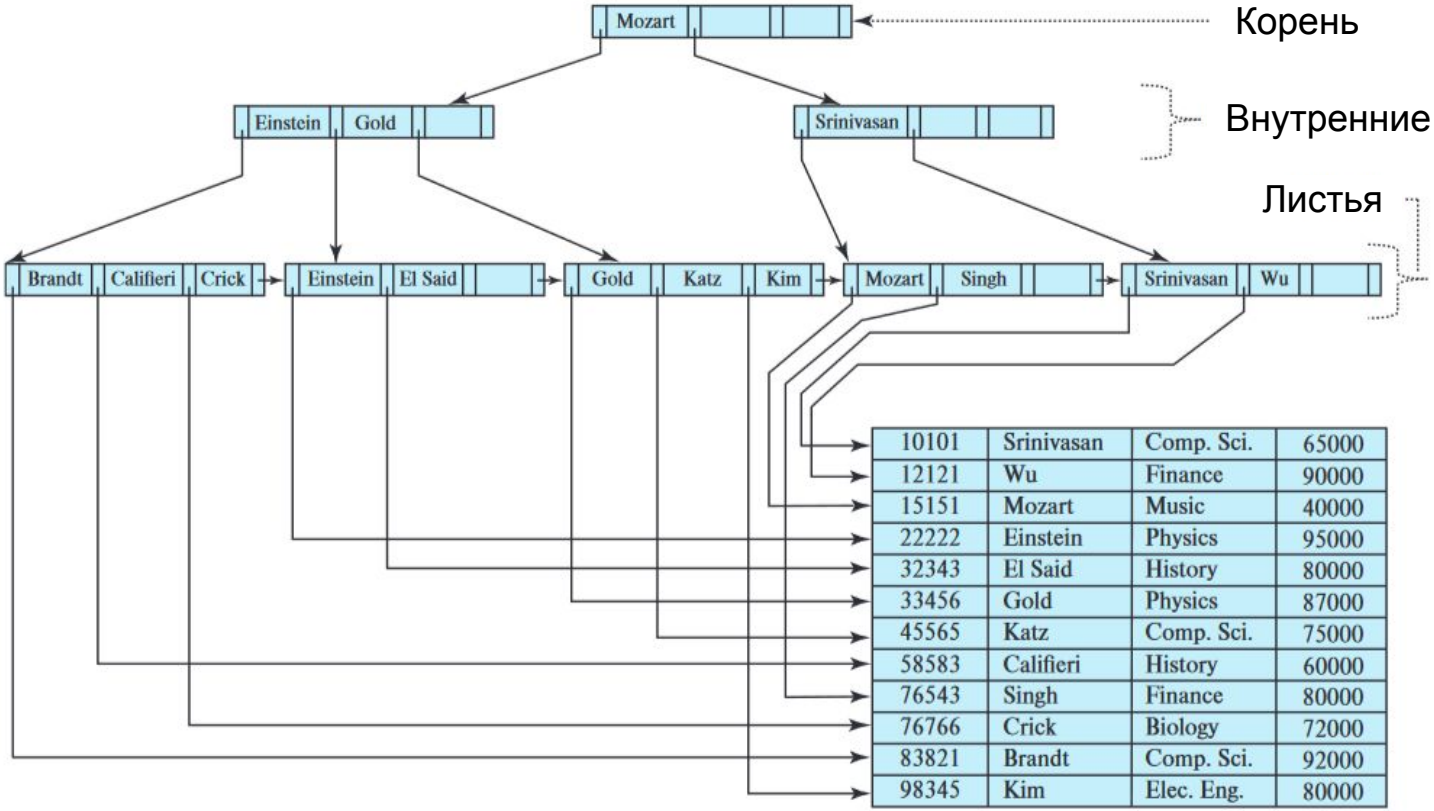
Сами значения записи

составное индекс

Если значения в блоке памяти отсортированы по ключу поиска данного индекса, то указатель на страницу

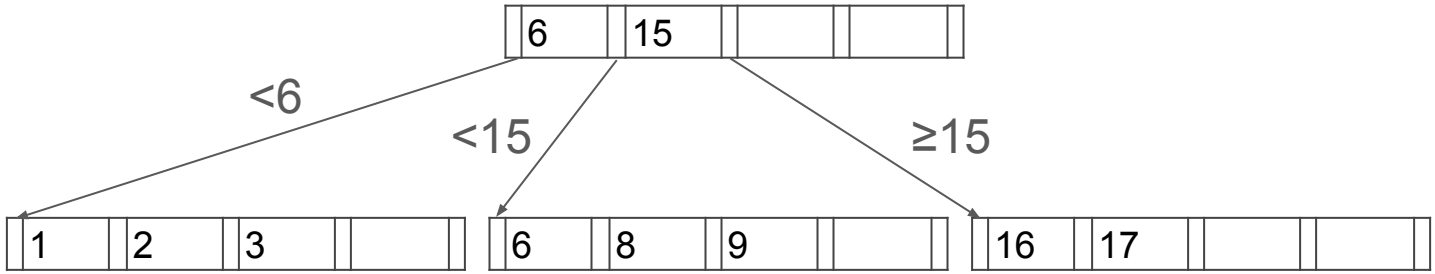
B+ дерево

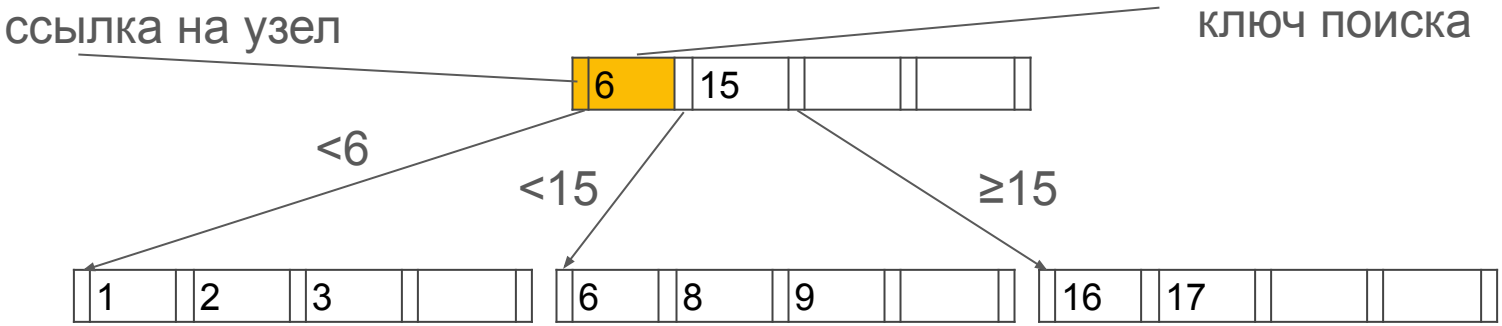
Будем предполагать для начала, что нет дублей по выбранному ключу поиска

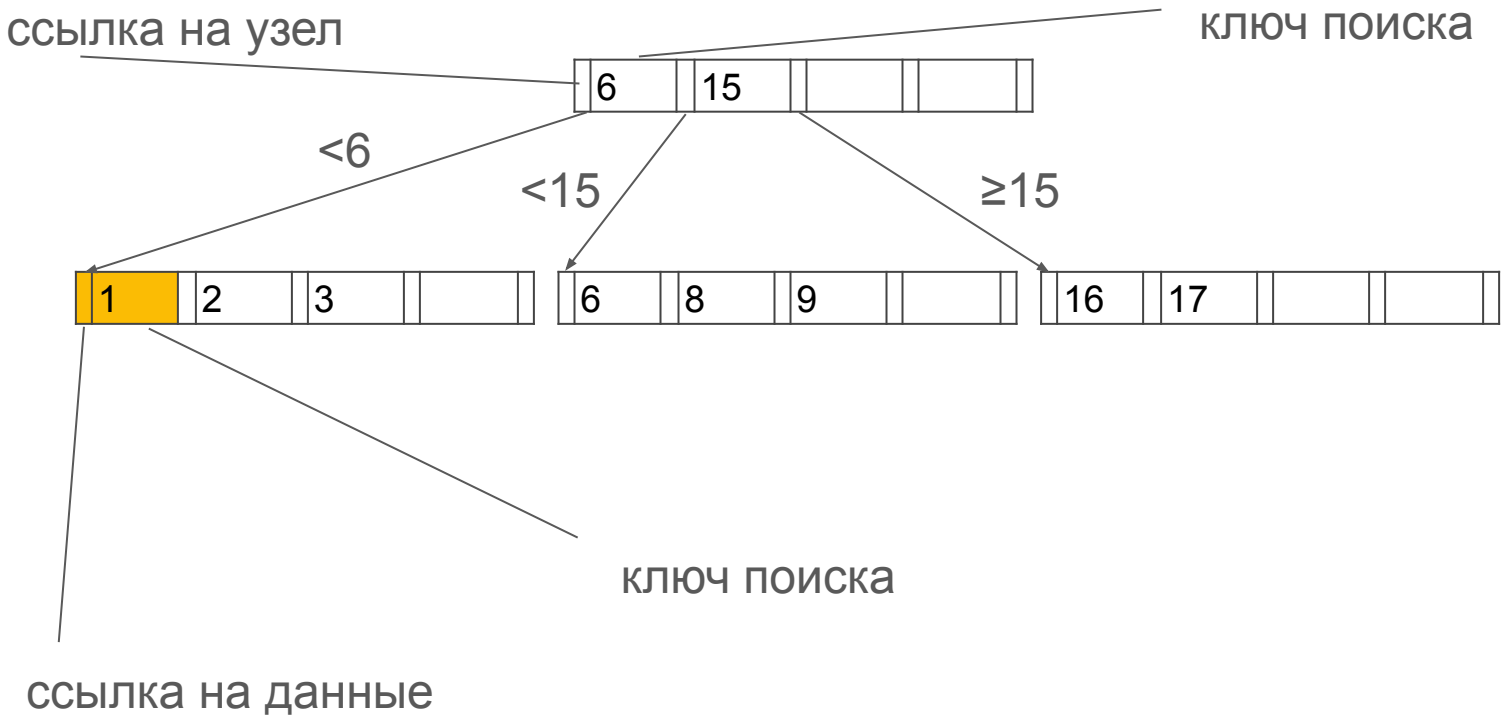

Mozart
Einstein
Gold
Srinivasan
Brandt
Califieri
Crick
Einstein
El Said
Gold
Katz
Kim
Mozart
Singh
Srinivasan
Wu
Корень
Внутренние вершины
Листья
10101 Srinivasan Comp. Sci. 65000
12121 Wu Finance 90000
15151 Mozart Music 40000
22222 Einstein Physics 95000
32343 El Said History 80000
33456 Gold Physics 87000
45565 Katz Comp. Sci. 75000
58583 Califieri History 60000
76543 Singh Finance 80000
76766 Crick Biology 72000
83821 Brandt Comp. Sci. 92000
98345 Kim Elec. Eng. 80000

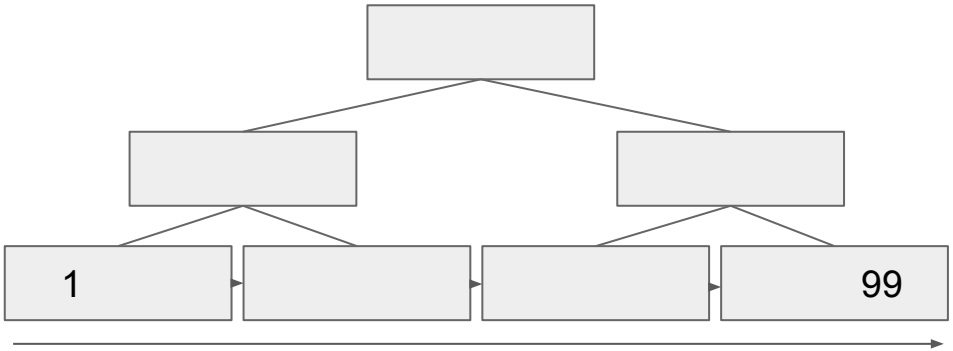
Разветвленность - количество дочерних вершин.

Если разветвленность = m, то количество заполненных дочерних вершин должна быть не меньше половины

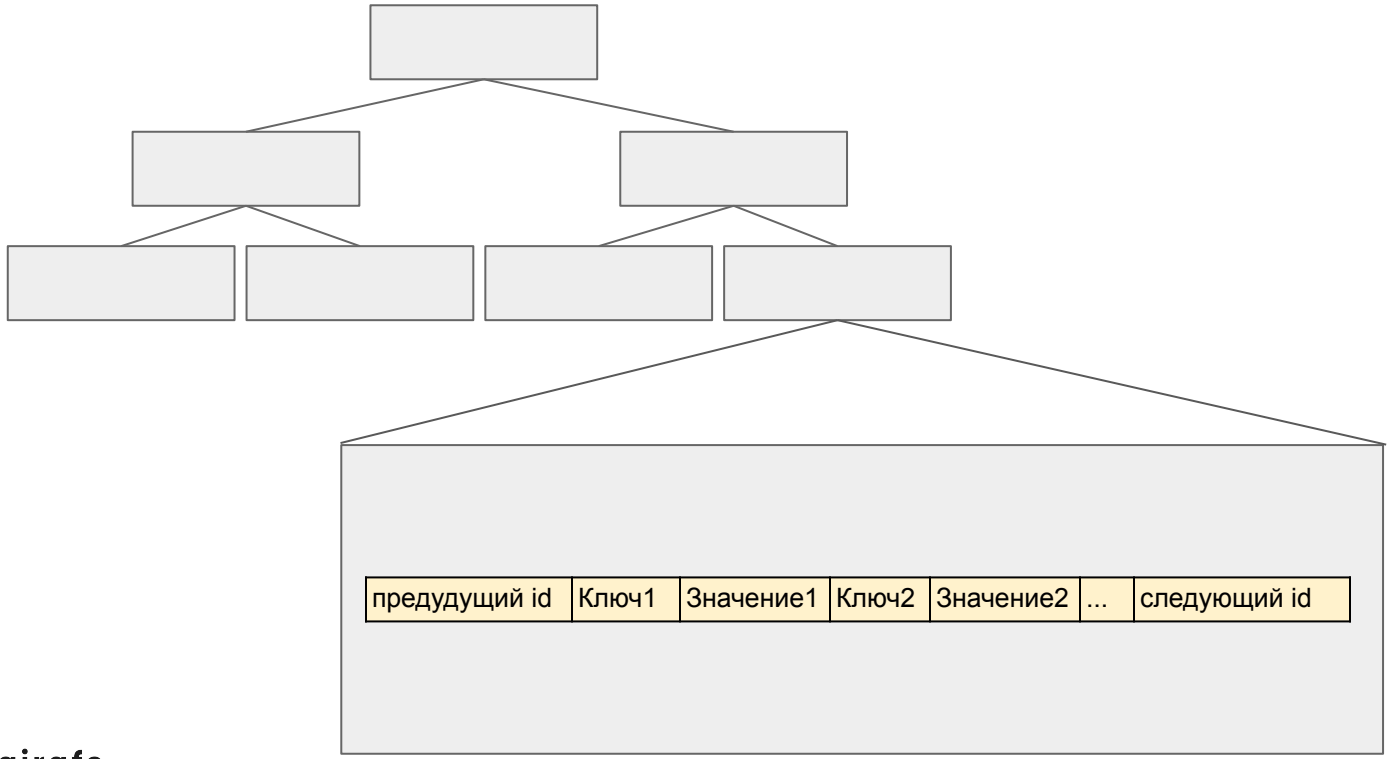

6
15
<6
<15
≥15
1
2
3
6
8
9
16
17

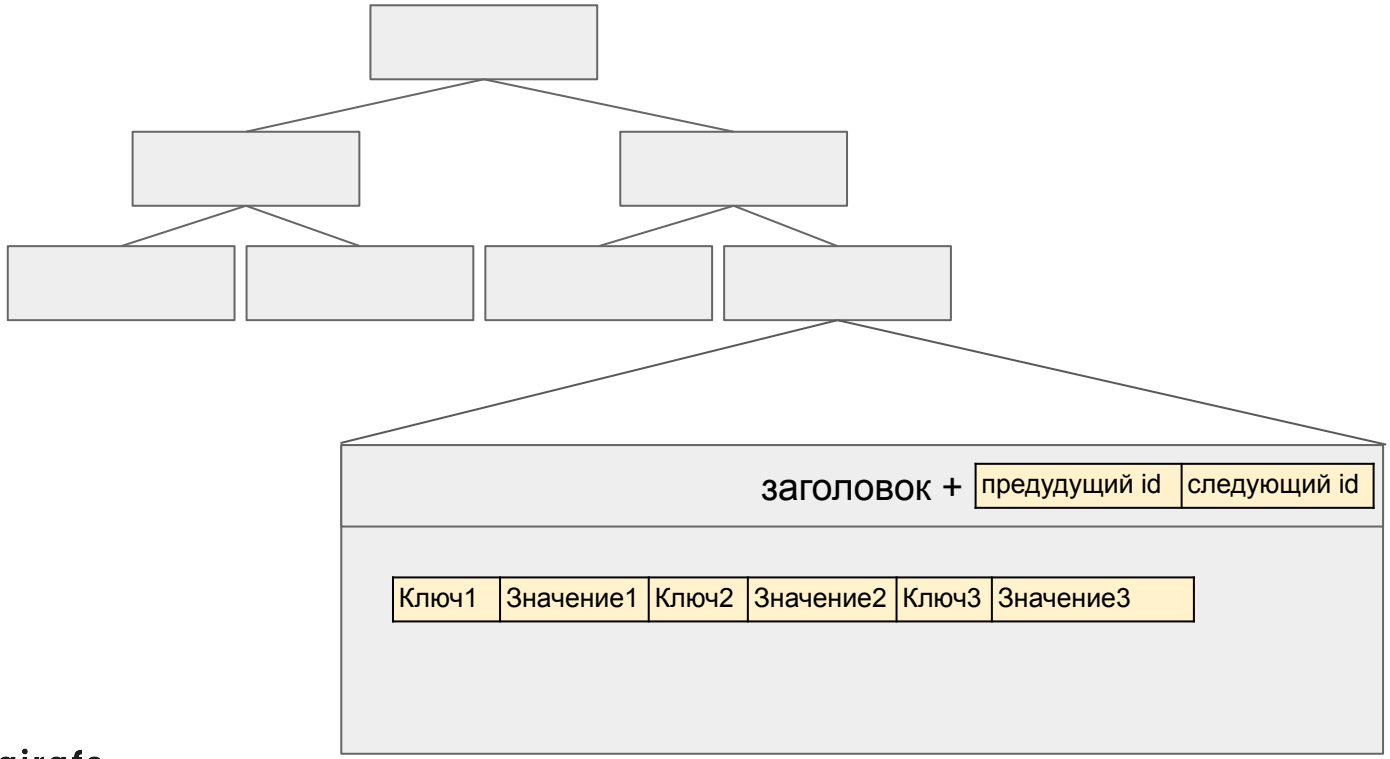

ссылка на узел
ключ поиска
6
15
<6
<15
≥15
1
2
3
6
8
9
16
17

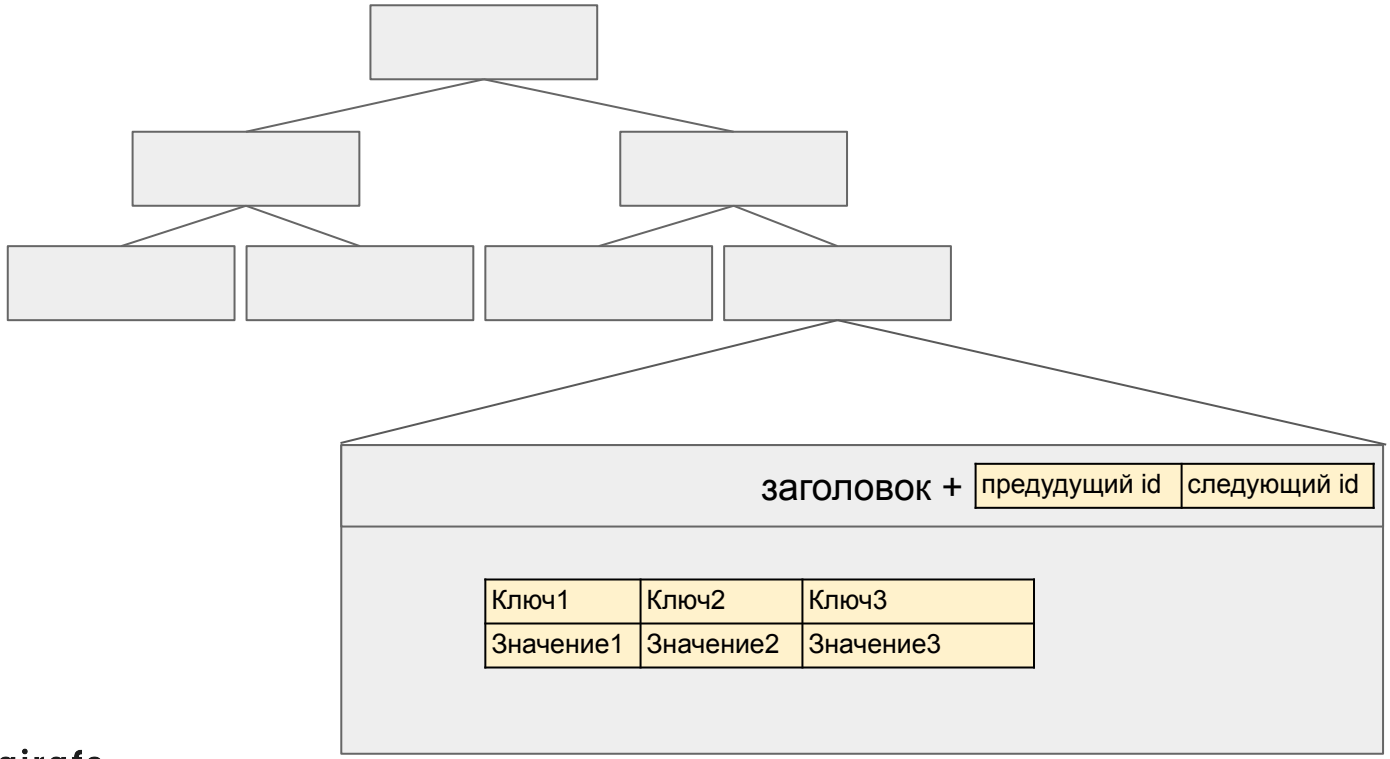

ссылка на узел
ключ поиска
6
15
<6
<15
≥15
1
2
3
6
8
9
16
17
ключ поиска
ссылка на данные



*типа отсортированы

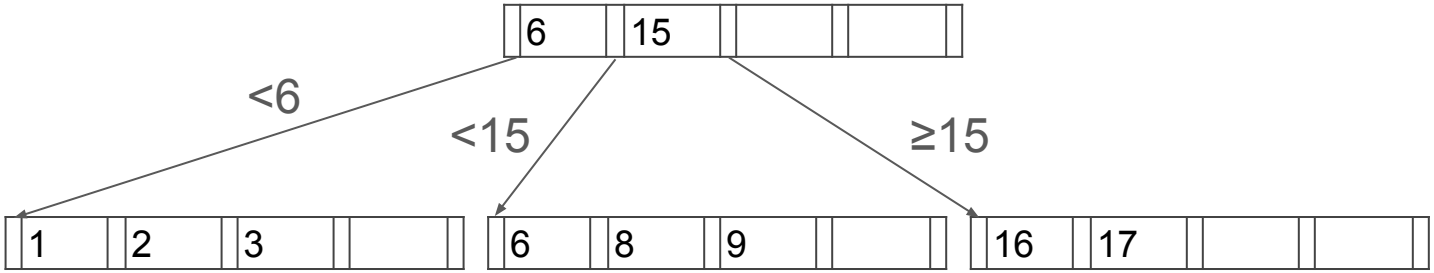

предудущий id
Ключ1
Значение1
Ключ2
Значение2
...
следующий id


заголовок +
предудущий id
следующий id
Ключ1
Значение1
Ключ2
Значение2
Ключ3
Значение3

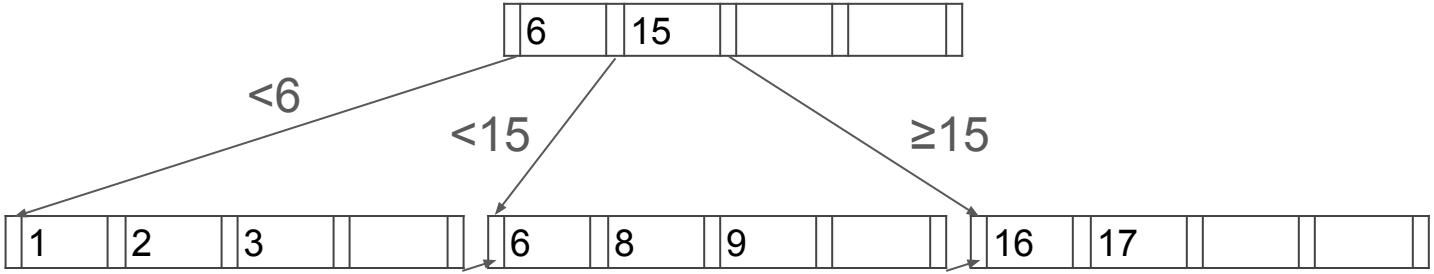


B+ дерево

girafe ai



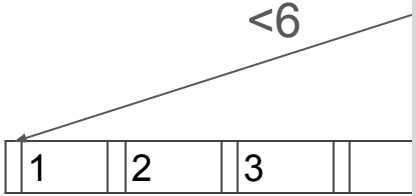

```
FUNCTION find(value)
    SET N = root_node;

    WHILE N IS NOT leaf_node:
        SET i = min{N.keys| value <= N.keys}
        IF i IS NULL:
            SET N = N(last(N.pointers))
        ELSEIF value = i:
            SET N = N(NEXT pointer WHERE keys = i)
        ELSE
            SET N = N(pointer WHERE keys = i)
        END
    END;

    FOR k IN N.keys:
        IF k = value:
            RETURN pointer
    RETURN NULL;
```




```
FUNCTION find(value)
SET N = root_node;

WHILE N IS NOT leaf_node:
      SET i = min{N.keys| value <= N.keys}
      IF i IS NULL:
            SET N = N(last(N.pointers))
      ELSEIF value = i:
            SET N = N(NEXT pointer WHERE keys = i)
      ELSE
            SET N = N(pointer WHERE keys = i)
      END
END;

FOR k IN N.keys:
      IF k = value:
            RETURN pointer
RETURN NULL;
```

find_range(value, upper_value)

```
WHILE (...)
      leaf_node ...
      FOR n IN leaf_node
            IF n.key <= upper_value:
                  RESULT_SET.append(n
.pointer);
      ...
      IF n = LAST(leaf_node):
            THEN next(leaf_node)
```

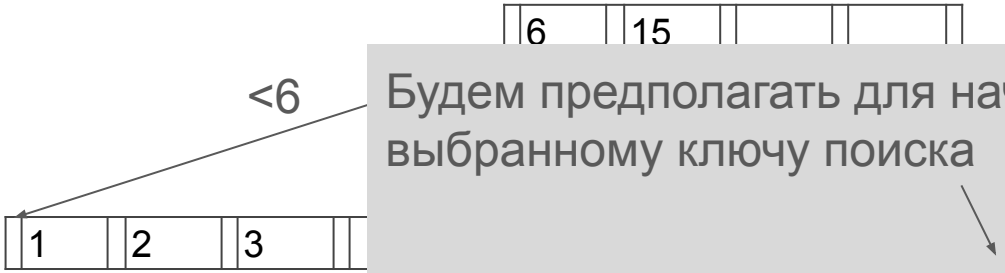


Будем предполагать для начала, что нет дублей по выбранному ключу поиска

find_range(value, upper_value)

```
WHILE (...)
      leaf_node ...
      FOR n IN leaf_node
            IF n.key <= upper_value:
                  RESULT_SET.append(n
.pointer);
      ...
      IF n = LAST(leaf_node):
            THEN next(leaf_node)
```

```
      SET i = min{N.keys| value <= N.keys}
      IF i IS NULL:
            SET N = N(last(N.pointers))
      ELSEIF value = i:
            SET N = N(NEXT pointer WHERE keys = i)
      ELSE
            SET N = N(pointer WHERE keys = i)
      END
END;

FOR k IN N.keys:
      IF k = value:
            RETURN pointer
RETURN NULL;
```



Будем предполагать для начала, что нет дублей по выбранному ключу поиска

key* = (key, что-то еще)

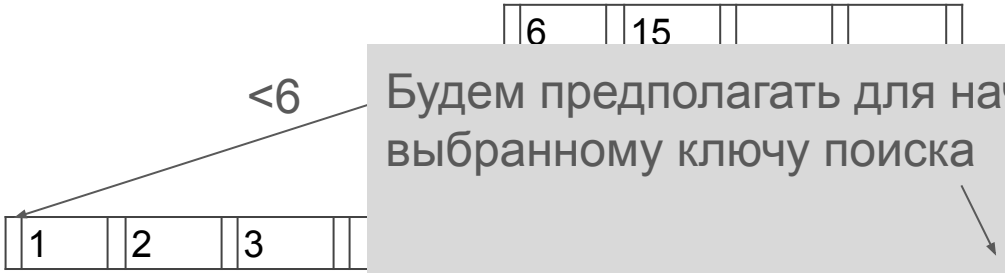
pk или oid
или …

find_range(value, upper_value)

```
WHILE (...)
      leaf_node ...
      FOR n IN leaf_node
            IF n.key <= upper_value:
                  RESULT_SET.append(n
.pointer);
      ...
      IF n = LAST(leaf_node):
            THEN next(leaf_node)
```

```
      IF i IS NULL:
            SET N = N(last(N.pointers))
      ELSEIF value = i:
            SET N = N(NEXT pointer WHERE keys = i)
      ELSE
            SET N = N(pointer WHERE keys = i)
      END
END;

FOR k IN N.keys:
      IF k = value:
            RETURN pointer
RETURN NULL;
```


girafe ai



Будем предполагать для начала, что нет дублей по выбранному ключу поиска

key* = (key, что-то еще)

pk или oid
или …

find(key*) = find_range((key, min), (key,max))

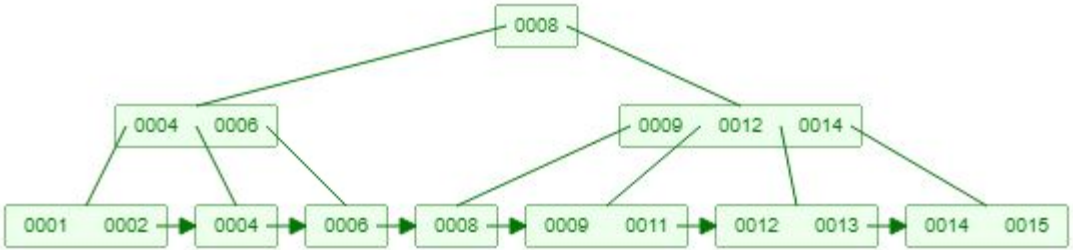
find_range(valu

```
WHILE (...)
      leaf_node ...
      FOR n IN leaf_node
            IF n.key <= upper_value:
                  RESULT_SET.append(n
.pointer);
      ...
      IF n = LAST(leaf_node):
            THEN next(leaf_node)
```

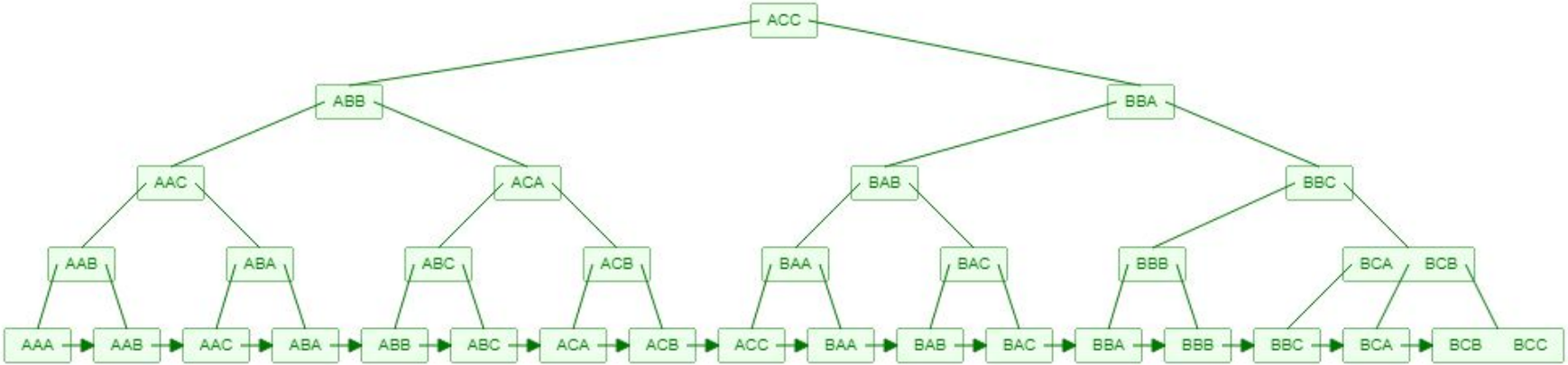
```
      END
END;

FOR k IN N.keys:
      IF k = value:
            RETURN pointer
RETURN NULL;
```


girafe ai

https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html



(key1,key2,key3)



girafe ai