Индексы в СУБД

	MySQL	PostgreSQL	MS SQL	Oracle
B-Tree index	Есть	Есть	Есть	Есть
Поддерживаемые пространственные индексы(Spatial indexes)	R-Tree с квадратичным разбиением	R-tree с линейным разбиением	Grid-based spatial index	R-Tree с квадратичным разбиением
Hash index	Только в таблицах типа Memory	Есть	Нет	Нет
Bitmap index	Нет	Есть	Нет	Есть
Reverse index	Нет	Нет	Нет	Есть
Inverted index	Есть	Есть	Есть	Есть
Partial index	Нет	Есть	Есть	Нет
Function based index	Нет	Есть	Есть	Есть

Виды индексов:

- B-деревья
- Инвертированный индекс
- Hash-индексы
- Индексы на основе битовых карт
- Пространственные индексы

Поиск документов по содержащимся в них словам

- WHERE Field1 like ‘%алгоритм%’
- Полное сканирование таблицы

Инвертированные индексы

- Документ (текстовое поле) – это последовательность слов

- D1: w1 w2 w3 w1 w4 w2
- D2: w1 w7 w8 w9 w5
- D3: w1 w7 w3 w2 w8

Поиск документов по содержащимся в них словам

- Задача:
- W1: d1 d2 d3
- W2: d1 d3
- W3: d1
- W5: d2

Полнотекстовый индекс FULLTEXT

- В полнотекстовый индекс включается один или несколько символьных столбцов в таблице.
- Эти столбцы могут иметь тип данных:
  - **char**, **varchar**, **nchar**, **nvarchar**, **text**, **ntext**, **image**, **xml** и **varbinary(max)**.
- Каждому столбцу может соответствовать определенный язык (из 50-ти возможных).
  - Английский 1033,
  - Русский 1049.

Процесс полнотекстового индексирования

- Фильтрацию, разбиение по словам,
- Удаление стоп-слов и нормализация токенов
- Преобразует конвертированные данные в инвертированный список слов
- Создание полнотекстового индекса.

Обработка полнотекстовых запросов

- разбиение по словам
- расширение тезауруса
- морфологический поиск
- обработка стоп-слов
- поиск в индексе
- ранжирование

- В полнотекстовых запросах не учитывается регистр букв.
- Все полнотекстовые запросы используют предикаты (CONTAINS и FREETEXT) и функции (CONTAINSTABLE и FREETEXTTABLE)

CONTAINS

Предикат, используемый в предложении WHERE для и проверки точного или нечеткого совпадения с отдельными словами, расстояния между словами или взвешенных совпадений.

CONTAINS ({ column_name | * } , 'condition')

- слова или фразы;
- префикса слова или фразы;
- слова около другого слова;

FREETEXT

Этот предикат используется в предложении WHERE для поиска значений, которые соответствуют условию поиска по смыслу, а не написанию.

FREETEXT ({ column_name | * } , 'string')

- Разбивает строку на отдельные слова
- Формирует словоформы.
- Определяет список расширений или замен на основании совпадений в тезаурусе.

Виды запросов

- Простое выражение.
- Префиксные выражения.
- Производное выражение.
- Выражения с учетом расположения.
- Синонимы.
- Взвешенное выражение.

Простое выражение

- Одно или несколько конкретных слов или фраз.
- { AND | & } | { AND NOT | &! } | { OR | | }

```
SELECT Comments
FROM ProductReview
WHERE CONTAINS(Comments, 'ужасно');
… CONTAINS(Comments, 'ужасно OR плохо');
```

Префиксные выражения

- Слова, начинающиеся заданным текстом, или фразы с такими словами.
- … CONTAINS(Comments, ' "ужасн*" ');

Производное выражение

- Словоформы конкретного слова.
- Синонимические формы конкретного слова.

- SELECT * FROM t3 WHERE freetext(s,'рама')

Выражения с учетом расположения

- Слова или фразы, находящиеся рядом с другими словами или фразами.

- CONTAINS(*,'мама ~ мыла')

Взвешенное выражение

- Слова или фразы со взвешенными значениями ()

```
SELECT * from CONTAINSTABLE (
t3
, *
, 'ISABOUT (мама WEIGHT(0.9)
        , auto WEIGHT(0.1)) ' )
ORDER BY RANK;
```

Результат: ранжированная таблица (ключ,

Для FULL TEXT индекса

- Выбрать столбцы таблицы или индексированного представления
- Построить для таблицы индекс по **одному** полю, которое не позволяет дубликатов и нулевых значений
- Построить каталог
- А потом уже строить индекс…

Создание каталога

- CREATE FULLTEXT CATALOG catalog_name

- Полнотекстовый каталог — это логическое понятие, обозначающее группу полнотекстовых индексов.

Создание FULL TEXT индекса

```
CREATE FULLTEXT INDEX ON table_name
   [ ( { column_name
            [ TYPE COLUMN type_column_name ]
            [ LANGUAGE language_term ]
            [ STATISTICAL_SEMANTICS ]
       } [ ,...n]
     ) ]
    KEY INDEX index_name
```

Полнотекстовый индекс FULLTEXT

- Загрузка данных в таблицу, уже имеющую индекс FULLTEXT, будет более медленной.

Индексы на основе битовых карт

- create bitmap index ind_4 on table_1(field1)
- Нужны тогда, когда у столбца может быть ограниченное число значений.
- В индекс входят:
  - значение столбца
  - битовая последовательность по количеству строк таблицы, в кот. 1 означает, что в данной строке атрибут принимает заданное значение

Индексы на основе битовых карт

Строка	Цвет глаз	Цвет волос	Рост
1	карие	блондинка	средний
2	зеленые	блондинка	средний
3	голубые	шатенка	высокий
4	серые	рыжая	средний
5	карие	брюнетка	средний
6	карие	блондинка	средний
7	серые	брюнетка	высокий
8	зеленые	шатенка	средний
9	голубые	рыжая	ниже среднего
10	голубые	брюнетка	ниже среднего

create bitmap index ind_4 on table_1(рост):

- Высокий 0010001000
- Средний 1101110100
- Ниже среднего 0000000011

create bitmap index ind_4 on table_1(рост):

- Блондинка 1100010000
- Шатенка 0010000100
- Брюнетка 0000101001
- Рыжая 0001000010

Блондинка среднего роста:

- Блондинка 1100010000
- Средний 1101110100
- Побитовое умножение 1100010000

Появилась девушка с голубыми волосами:

- Блондинка 1100010000
- Шатенка 0010000100
- Брюнетка 0000101001
- Рыжая 0001000010
- Мальвина 00000000001

Индексы на основе битовых карт

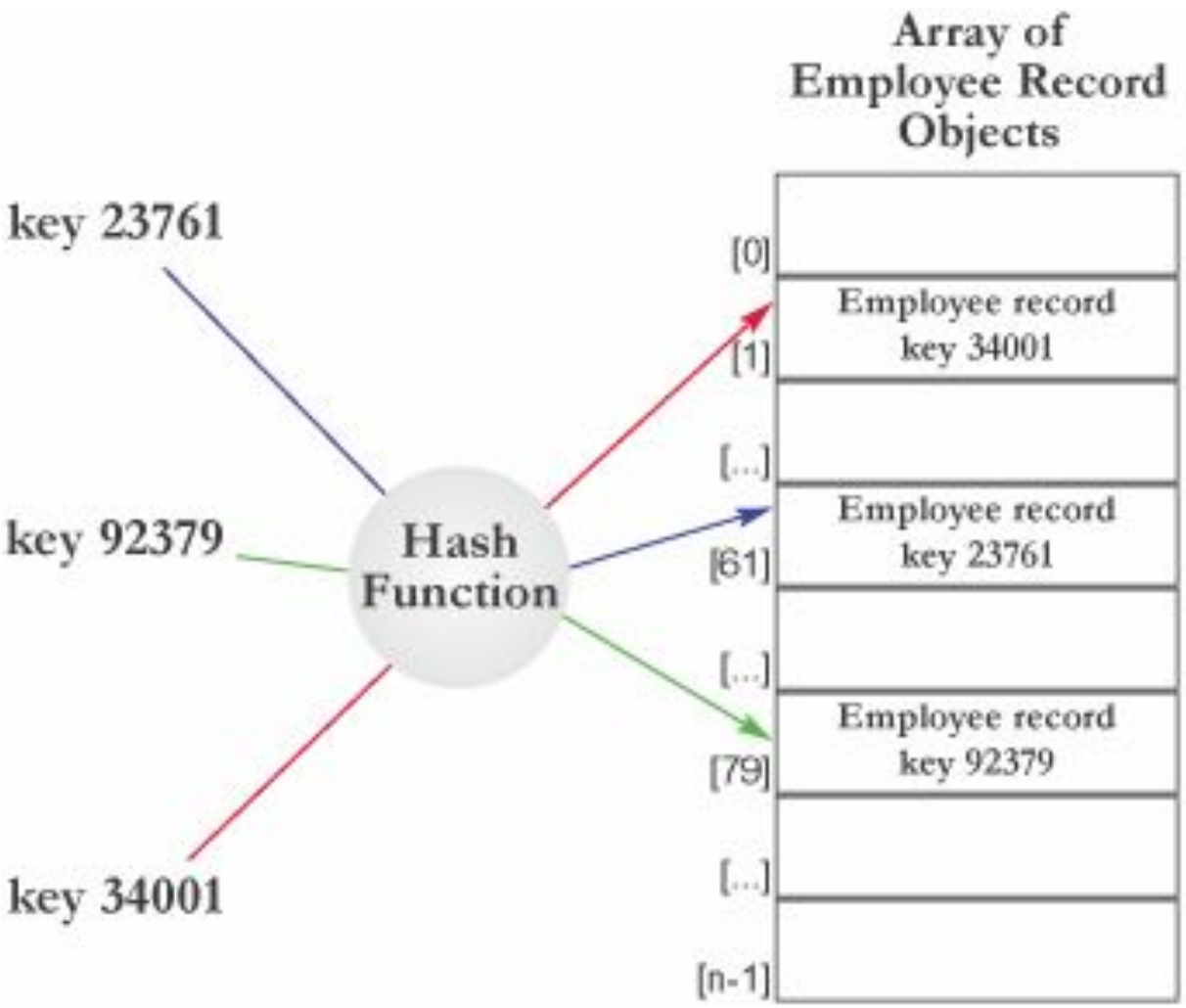
- Индексы на основе битовых карт обычно создаются быстрее и могут занимать удивительно мало места.
- Размер индекса на основе битовых карт существенно зависит от распределения данных.
- Индексы на основе битовых карт обычно выбираются стоимостным оптимизатором, если для выполнения запроса можно использовать несколько таких индексов.
- Изменения столбцов, входящих в индексы на основе битовых карт, а также вставки и удаления данных могут вызывать существенные конфликты блокировок.
- Изменения столбцов, входящих в индексы на основе битовых карт, а также вставки и удаления данных могут весьма существенно "ухудшать" индексы.

Hash-индекс

- Выбираем количество участков, в которых будем размещать записи.
- Подбираем функцию перемешивания, которая от ключевого столбца будет выдавать номер участка.
- В памяти храним таблицу адресов участков

Создание hash-индекса

```
CREATE INDEX имя_индекса
    USING HASH
   ON имя_таблицы (имя_столбца)
```


Array of
Employee Record
Objects
key 23761
key 92379
key 34001
Hash
Function
[0]
[1]
[...]
[61]
[...]
[79]
[...]
[n-1]
Employee record
key 34001
Employee record
key 23761
Employee record
key 92379

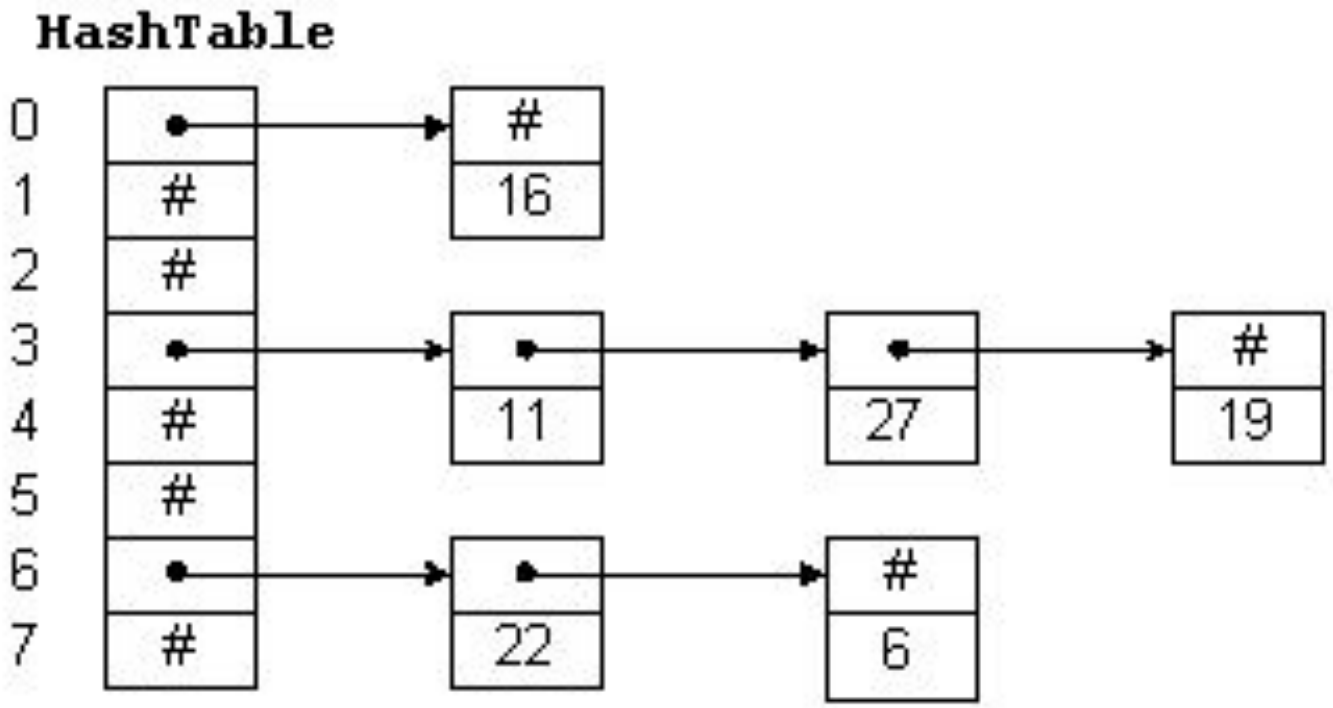
Hash-индекс

- Для размещения таблицы отводится заданное количество участков
- Есть функция hash(key)=n, где n – номер участка
- В памяти хранится таблица адресов участков
- Доступ к данным за одно обращение к диску

Недостатки hash-индексов

- Таблица адресов участков может быть слишком велика
- Если в один участок попало слишком много записей, придется выделять дополнительный блок.
- Проблема – неравномерность размещения записей, возникновение коллизий

Коллизии



Функции Hash

- Деление
- Мультипликативный метод

Функции Hash деление

- Размер таблицы hashTableSize - простое число.
- Хеширующее значение hashValue, изменяющееся от 0 до (hashTableSize - 1), равно остатку от деления ключа на размер хеш-таблицы.
- Увеличиваем число участков в два раза

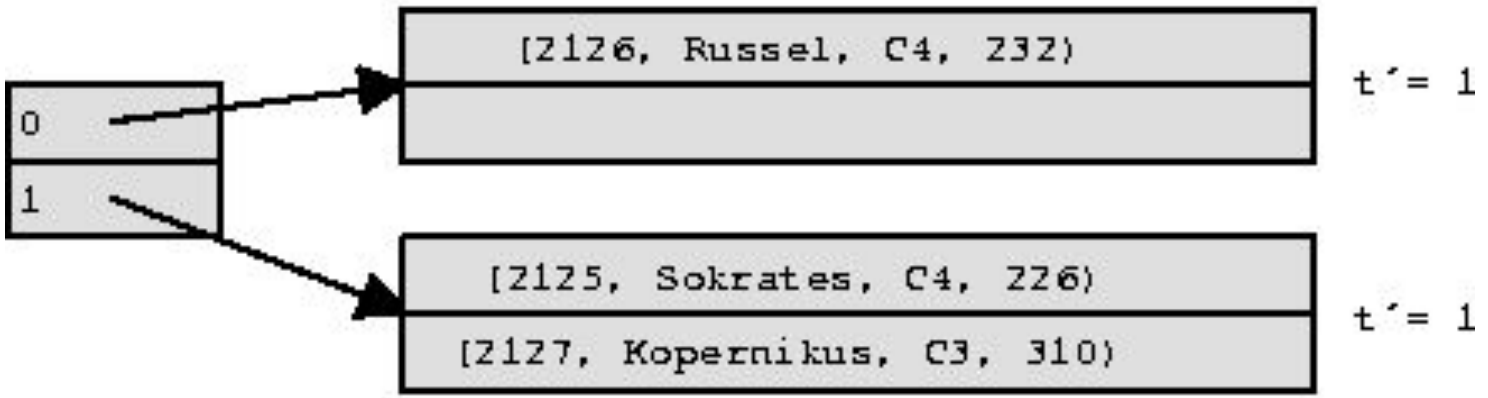
Функции Hash мультипликативный метод

- Размер таблицы hashTableSize есть степень 2n.
- Значение key умножается на константу, затем от результата берется n бит.
- В качестве такой константы Кнут рекомендует золотое сечение (sqrt(5) - 1)/2 = 0.6180339887499.

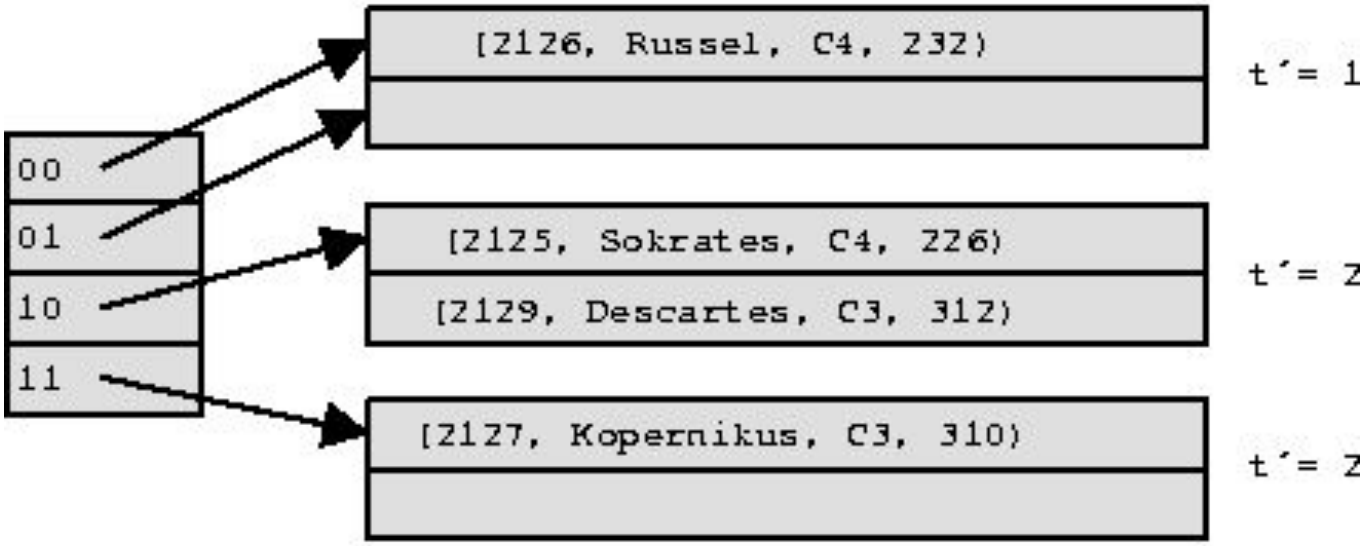
Функции Hash для строк переменной длины

- Аддитивный метод – преобразовываем слова в числа, складываем и берем остаток деления по модулю 256.
- Метод ИЛИ

x	h(x)	
	d	p
2125	1	01100100001
2126	0	11100100001
2127	1	11100100001



x	h(x)	
	d	p
2125	10	1100100001
2126	01	1100100001
2127	11	1100100001
2129	10	0010100001



t = 2

Пространственные типы данных

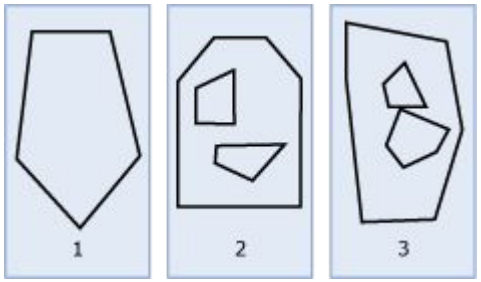
- **geometry** используется для планарных или евклидовых данных
- **geography**, который используется для хранения эллиптических данных, таких как координаты GPS широты и долготы

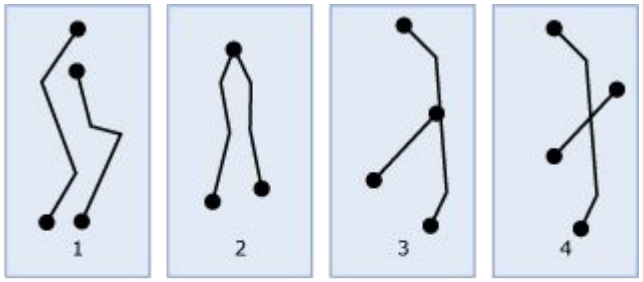
Пространственные типы данных

- **geometry** используется для планарных или евклидовых данных
- **geography**, который используется для хранения эллиптических данных, таких как координаты GPS широты и долготы
- объекты **geography** должны помещаться в одном полушарии, расстояние обычно вычисляется в метрах

Пространственные типы данных

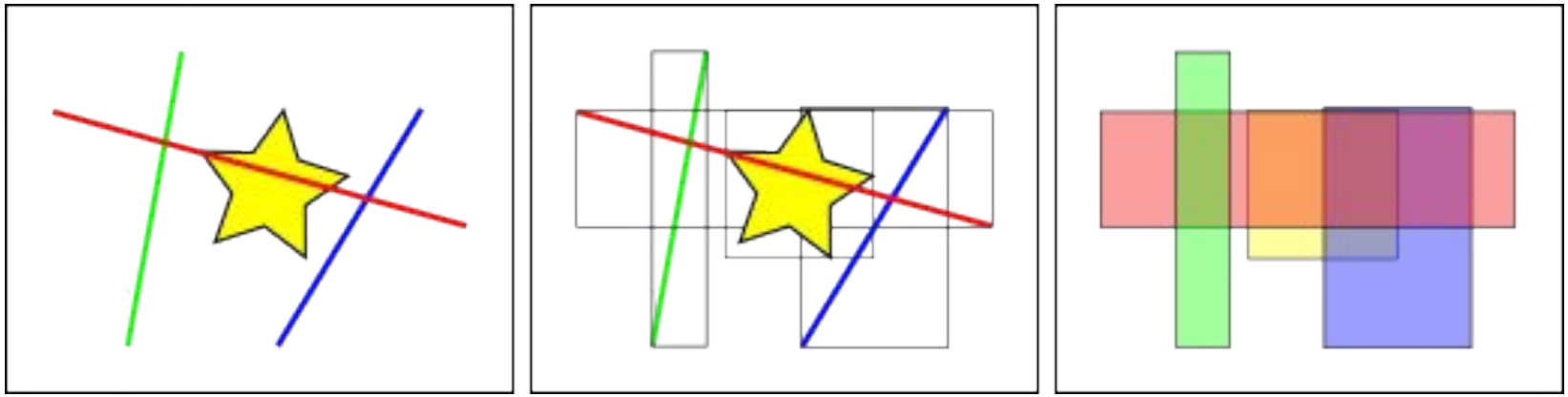
- Point
- MultiPoint
- LineString
- MultiLineString
- Polygon





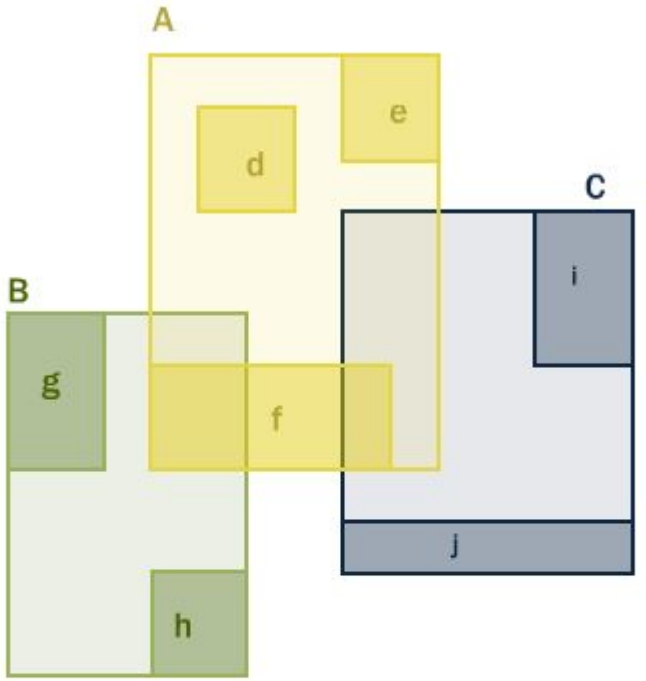
R-дерево

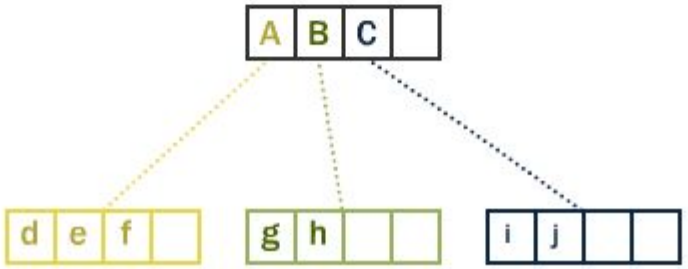
- Избавляемся от формы – окружаем фигуру min ограничивающим прямоугольником
  (oid, Rectangle), oid – ссылка на запись



Иерархия R-дерева

- Окружаем фигуры ограничивающими прямоугольниками
- *(cp, Rectangle)*
- При переполнении делим пополам

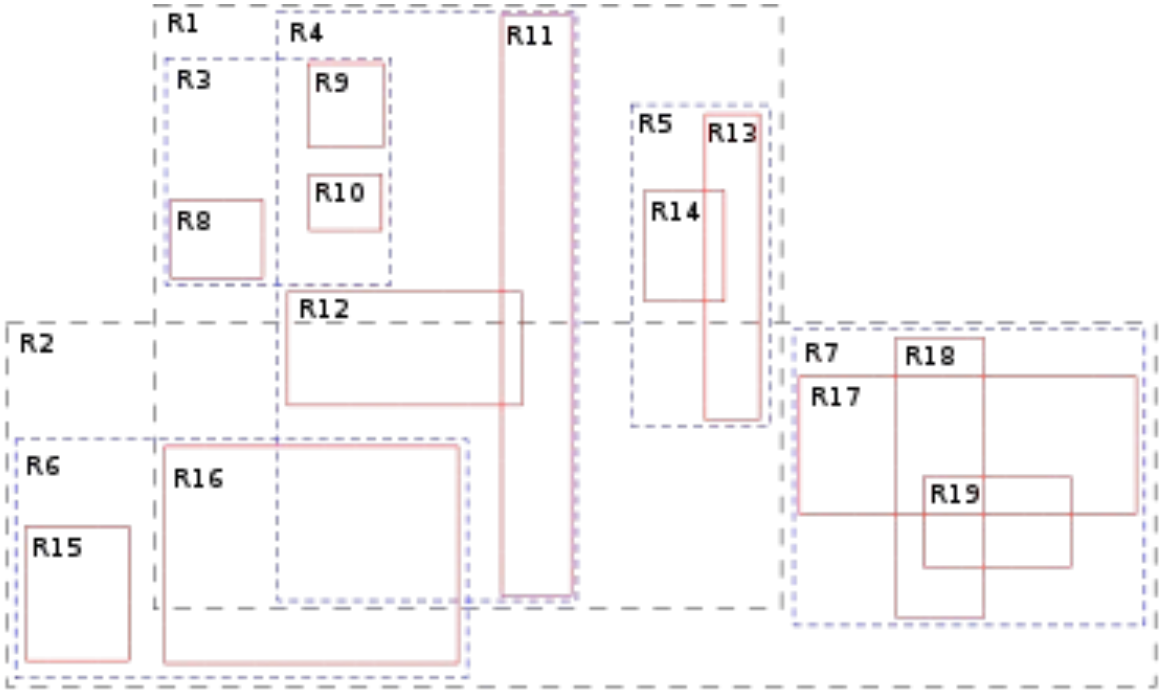


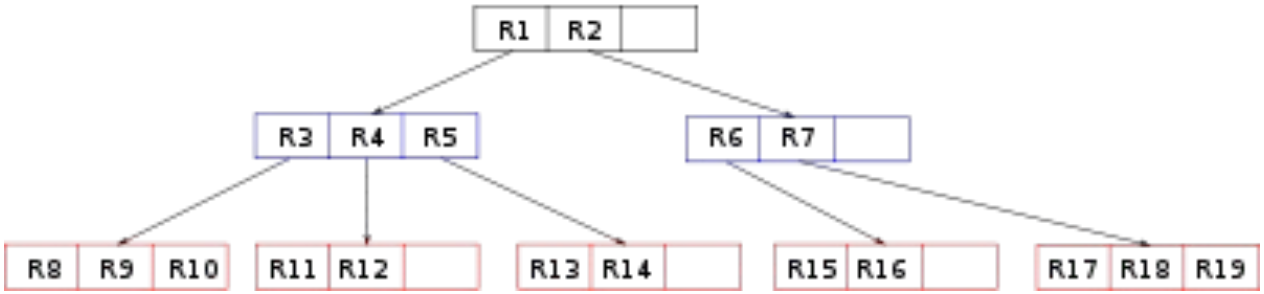


Критерии разделения узла

- Минимальная площадь
- Минимальное перекрытие
- Минимальные границы

R-дерево

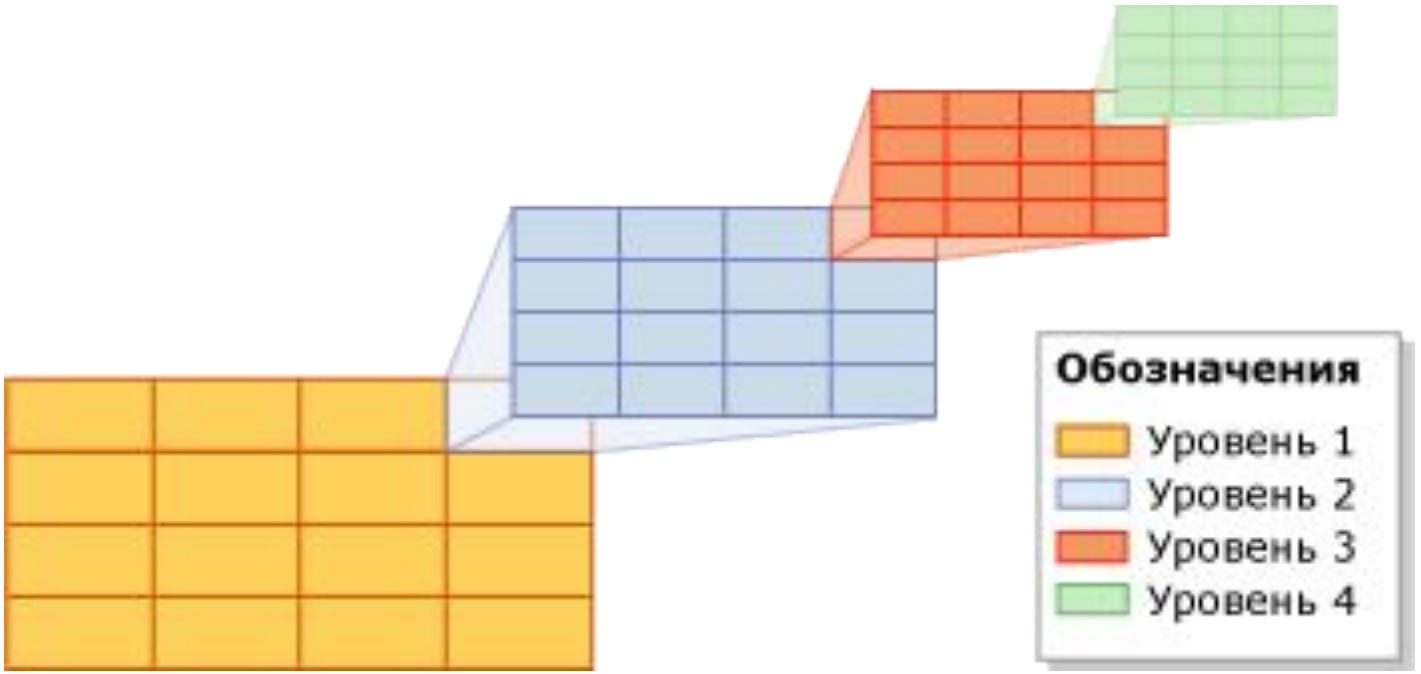

R1
R4
R11
R3
R9
R5
R13
R10
R14
R8
R12
R2
R7
R18
R17
R6
R16
R19
R15


R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19

R-дерево - недостатки

- Не удается избежать перекрытий – необходим просмотр нескольких веток

Spatial grid



Spatial grid

- CREATE SPATIAL INDEX
- 4 уровня вложенности

Ключевое слово	Конфигурация сетки	Число ячеек
LOW	4X4	16
MEDIUM	8X8	64
HIGH	16X16	256

Spatial grid

- Декомпозиция индексированного пространства в сеточную иерархию
- Считывает данные из пространственного столбца по строкам.
- *Тесселяция* - помещаем объект в сеточную иерархию, устанавливая связь между объектом и набором сеточных ячеек, с которыми он соприкасается.

Правила тесселяции

- Правило покрытия
- Правило ячеек на объект
- Правило самой глубокой ячейки

SQL Server

Oracle 10g