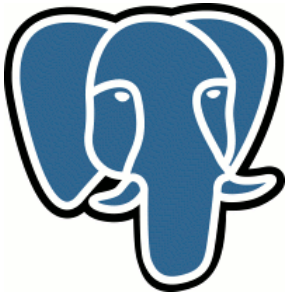
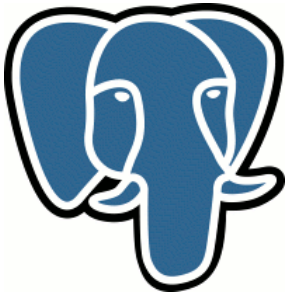


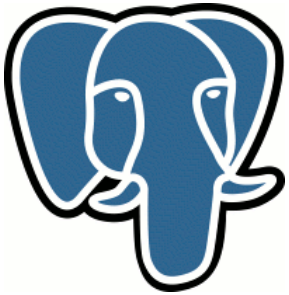
Индексный поиск по регулярным выражениям

Александр Коротков

Интаро Софт



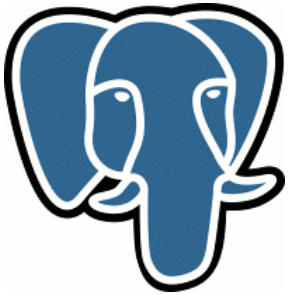
Введение



Регулярные выражения

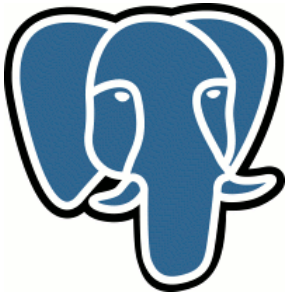
- Мощный инструмент обработки текстовой информации
- Основаны на математической лингвистике
- Выражают тот же класс языков, что и лингвистические автоматы

Автоматы… нет, не слышал



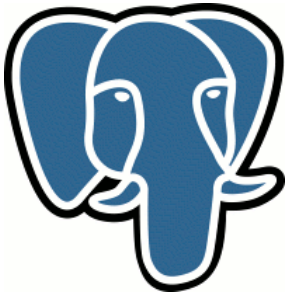
Итак… автоматы

- Регулярное выражение может быть преобразовано в лингвистический автомат
- Такое преобразование используется в «движках» регулярных выражений



Итак... автоматы

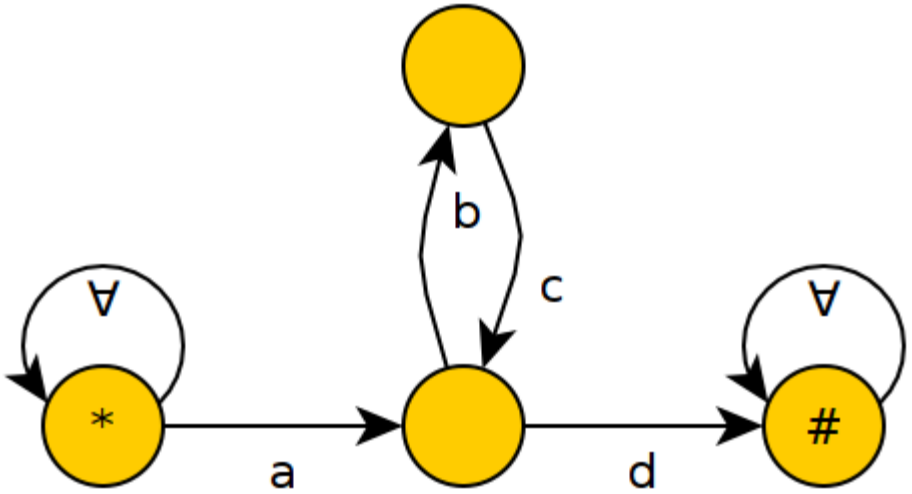
- Автомат – это ориентированный граф, вершины которого – «состояния», а дуги – «переходы», помеченные символами.
- Выделяются начальное состояние и конечное состояние
- Автомат «читает» строку.

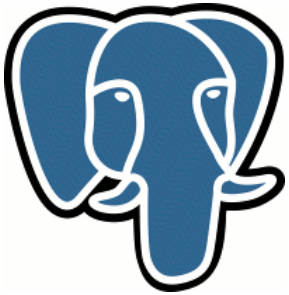


Пример

/a(bc)*d/

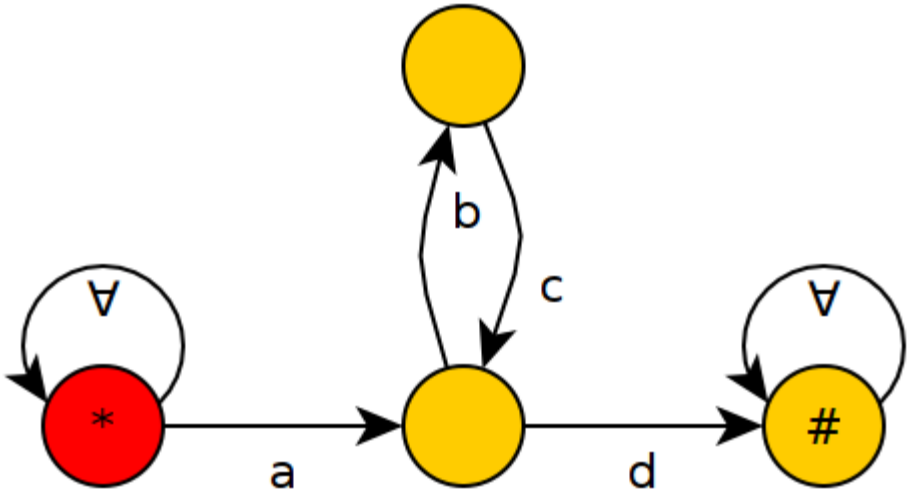
становится





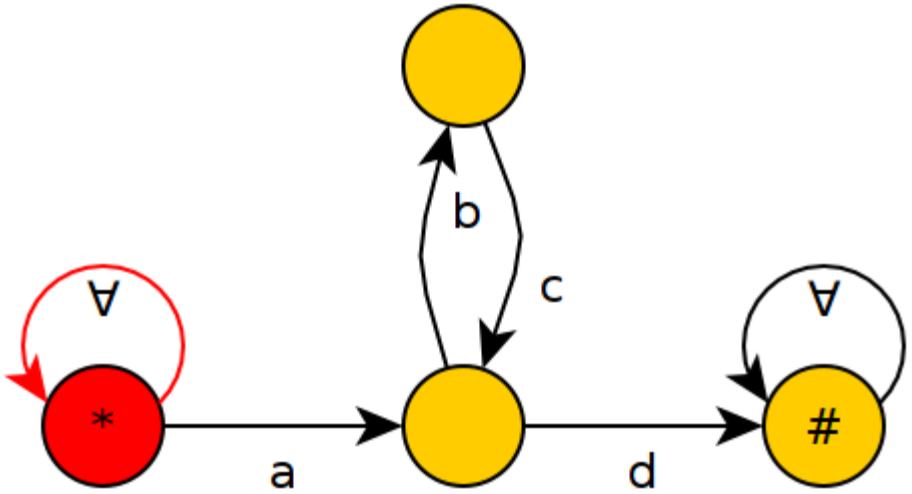
Пример

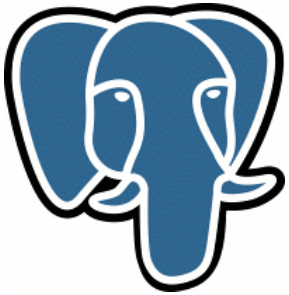
xyzabcbcdxyz



Пример

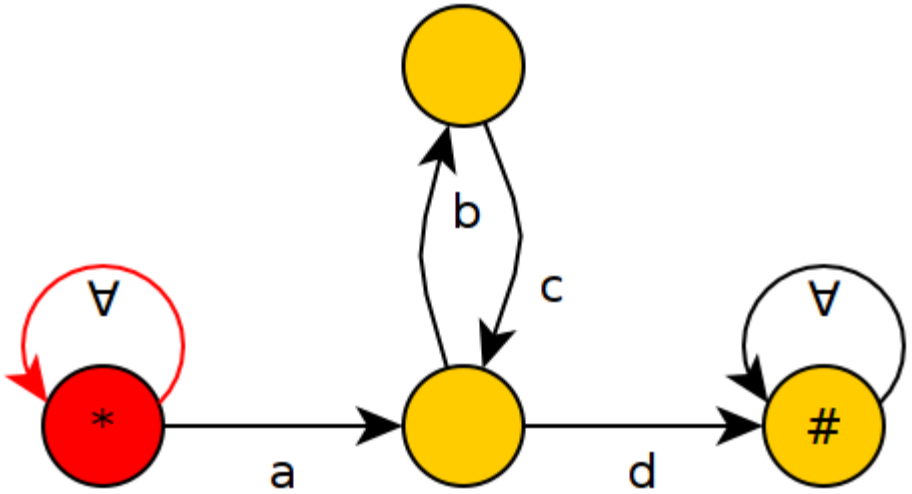
xyzabcbcdxyz

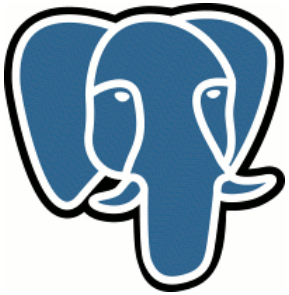




Пример

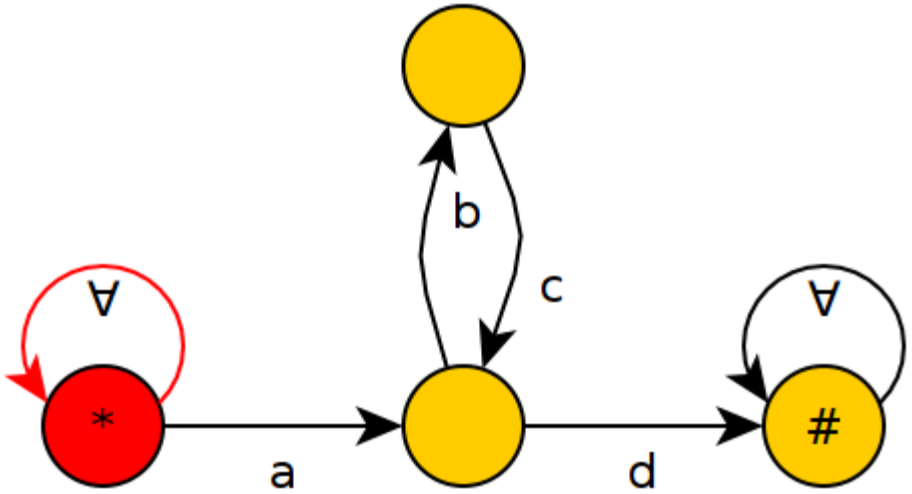
xyzabcbcdxyz

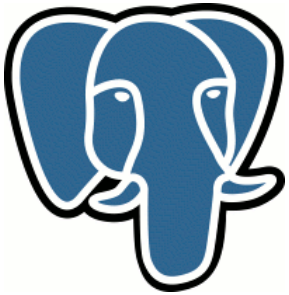




Пример

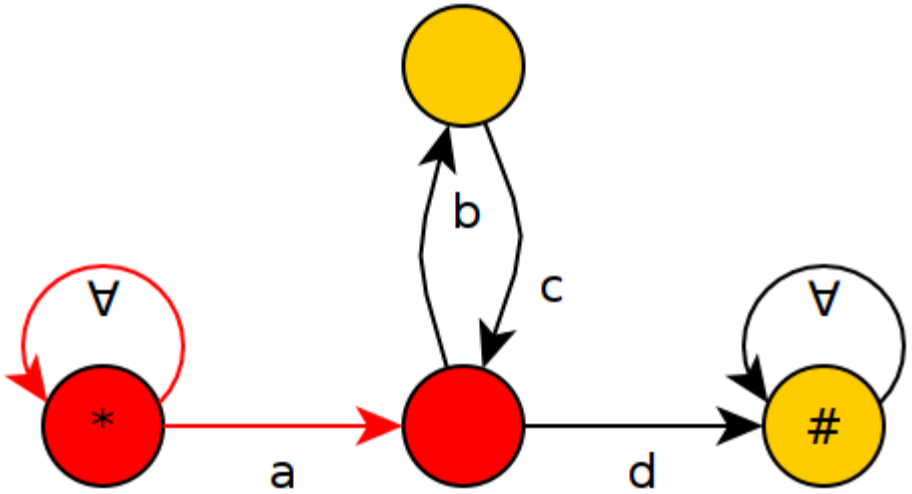
xyzabcbcdxyz

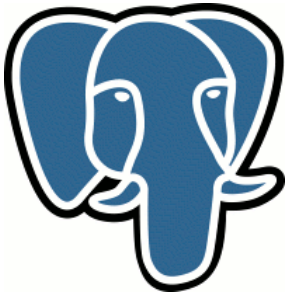




Пример

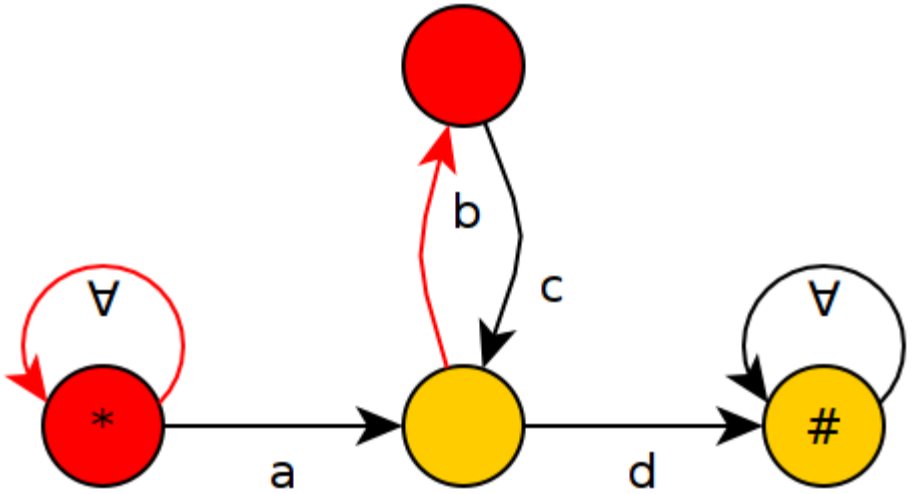
xyzabcbcdxyz

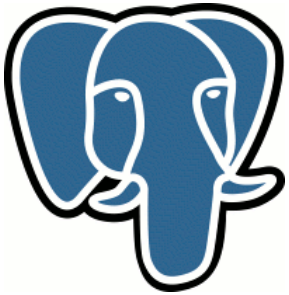




Пример

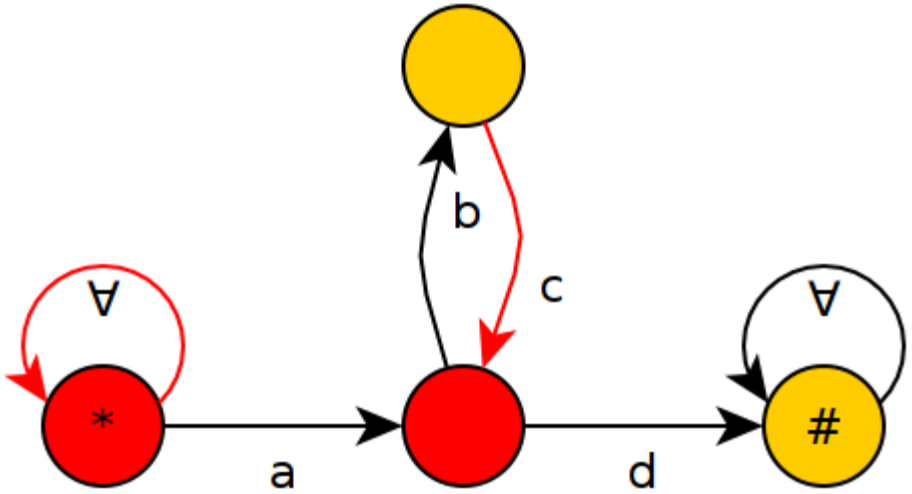
xyzabcbcdxyz

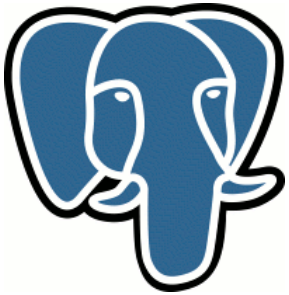




Пример

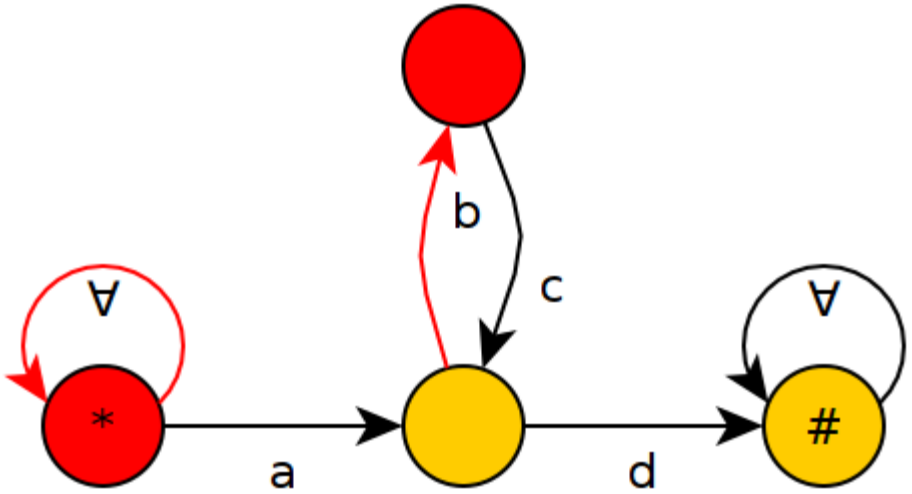
xyzabcbcdxyz

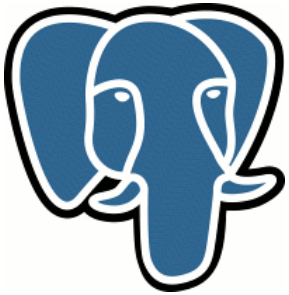




Пример

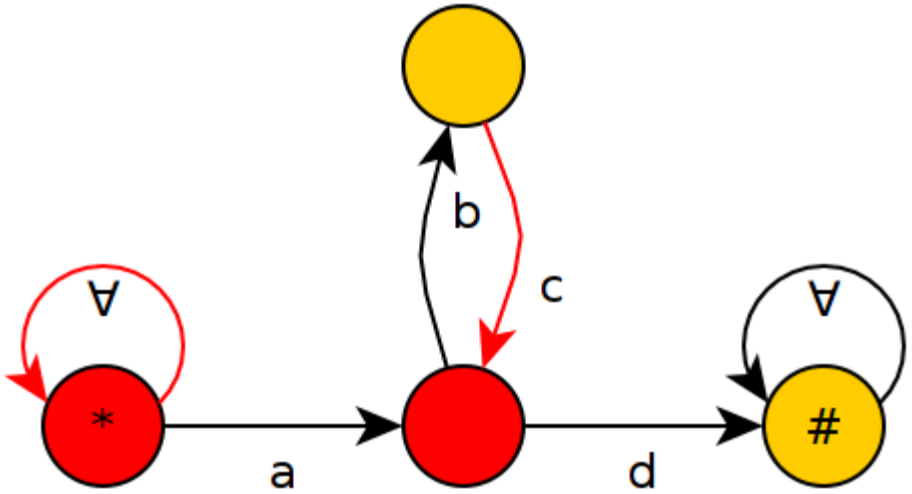
xyzabcbcdxyz

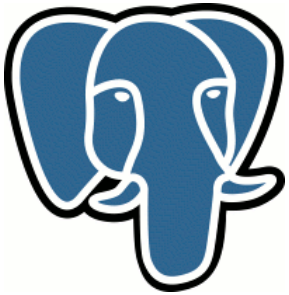




Пример

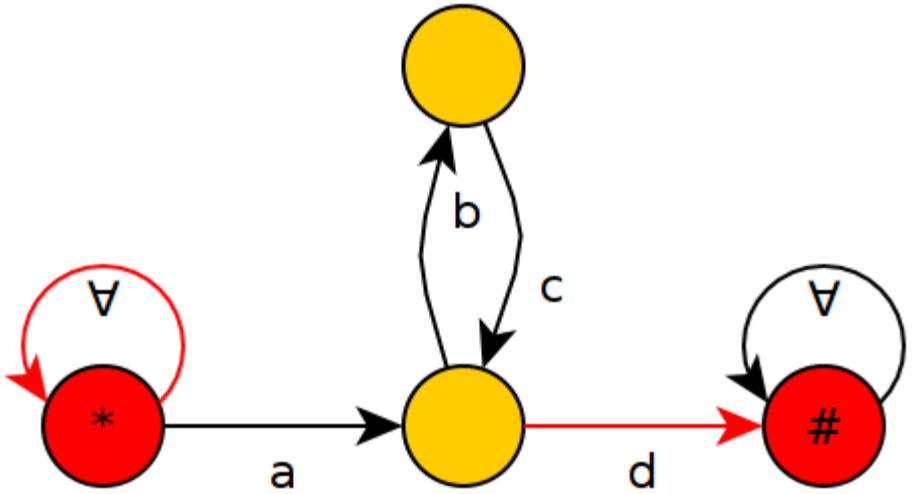
xyzabcbcdxyz

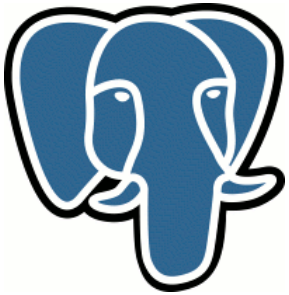




Пример

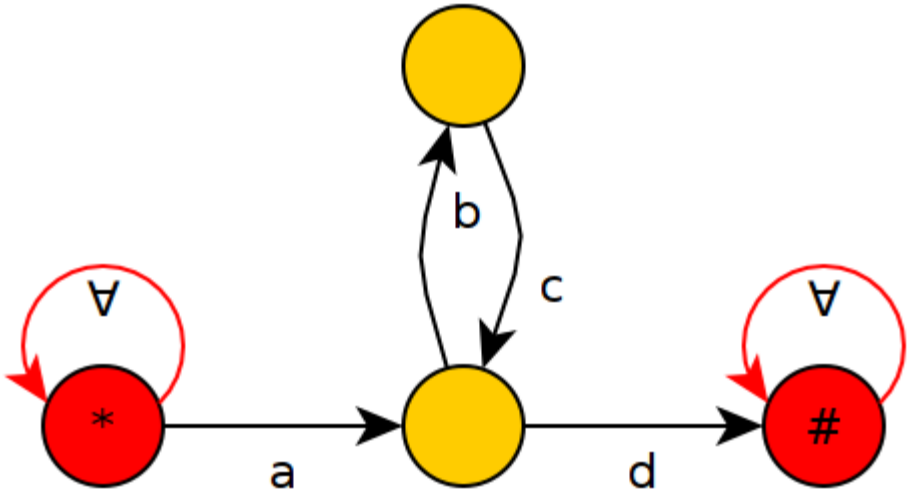
xyzabcbcdxyz

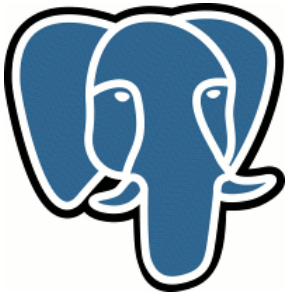




Пример

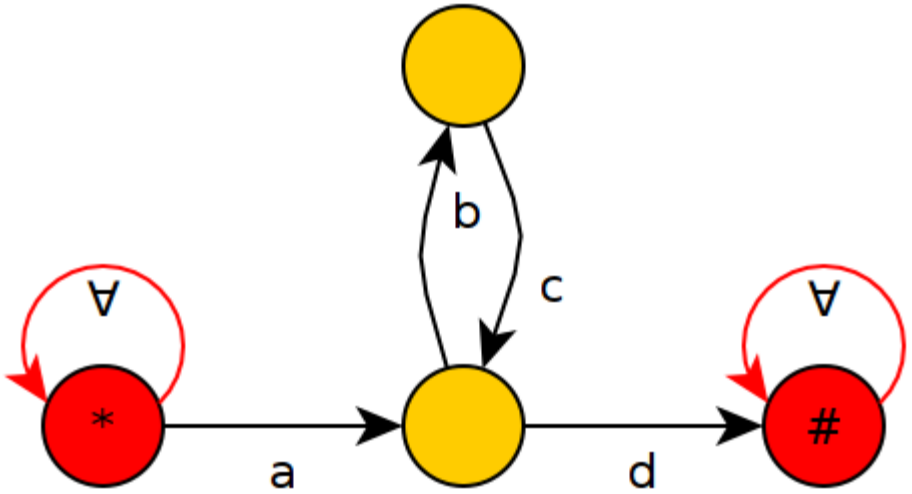
xyzabcbcdxyz

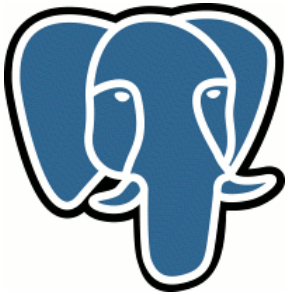




Пример

xyzabcbcdxyz

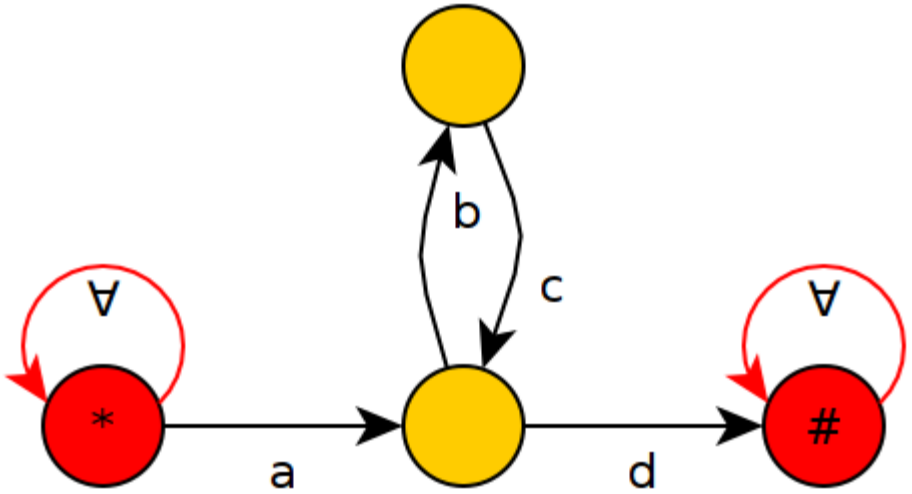




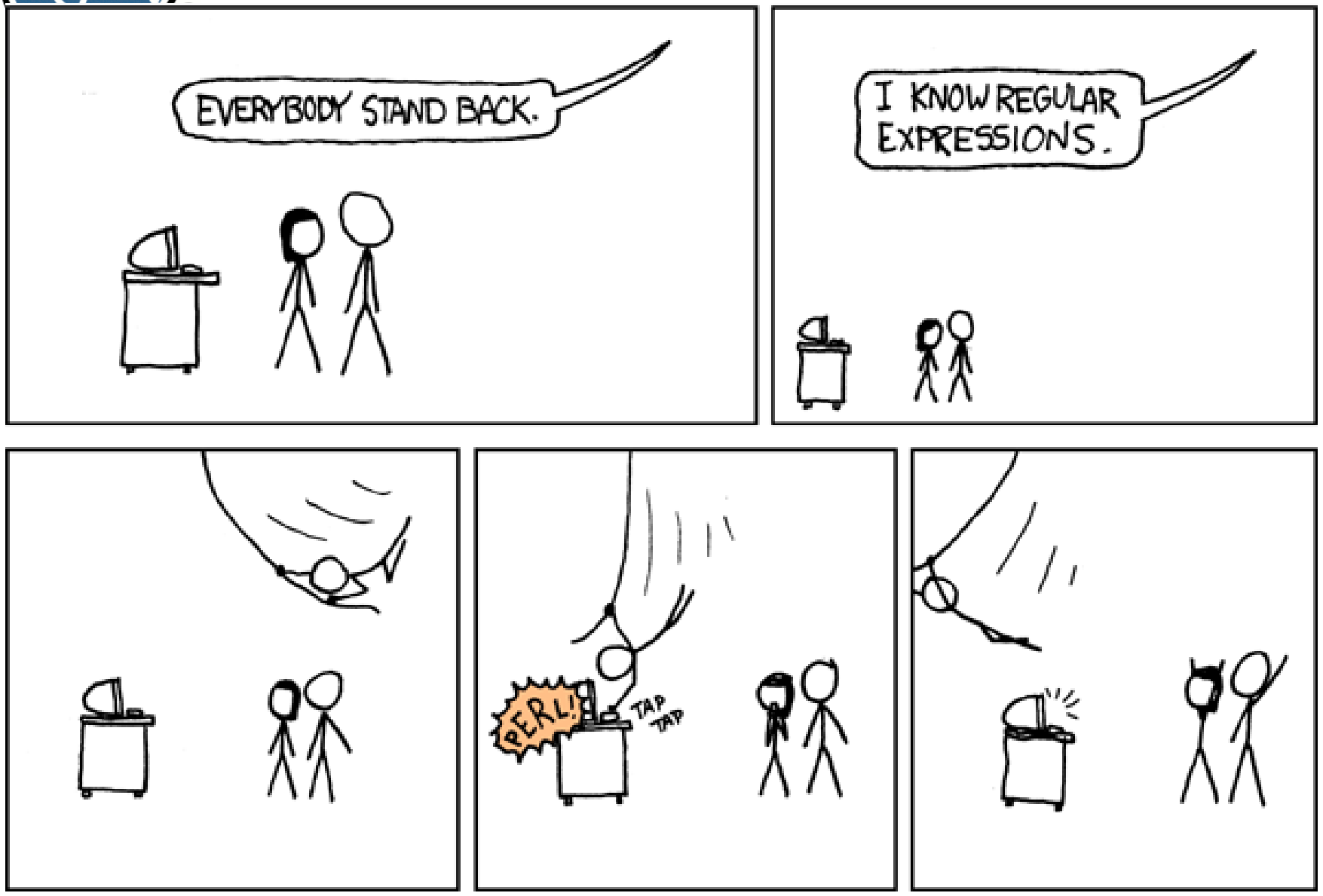
Пример

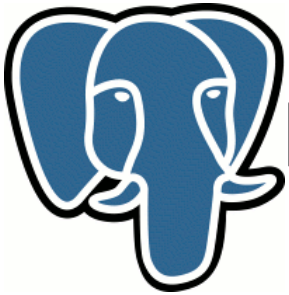
xyzabcbcdxyz

Ура! Подходит!



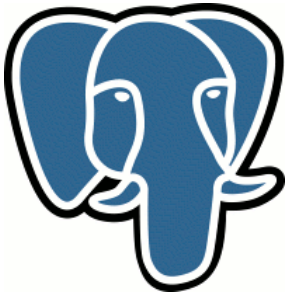
Ну теперь все мы знаем...





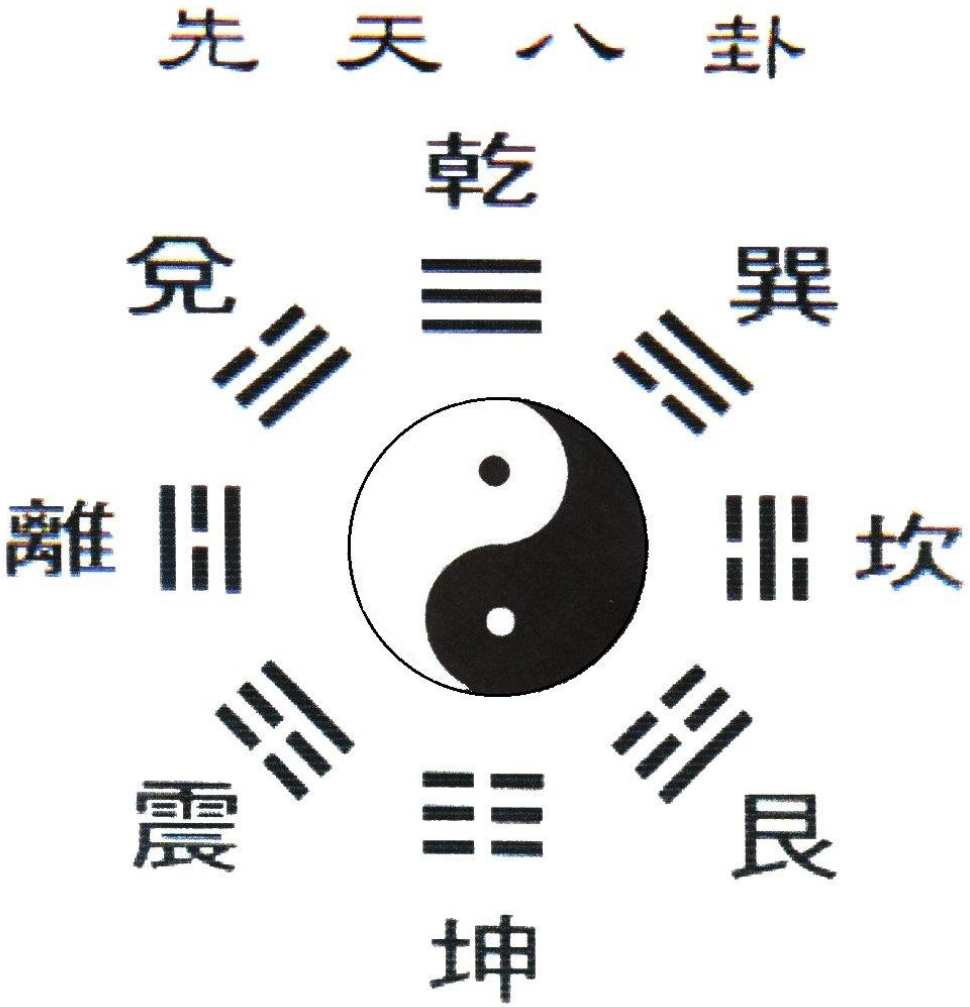
Поиск по регекспам в PostgreSQL

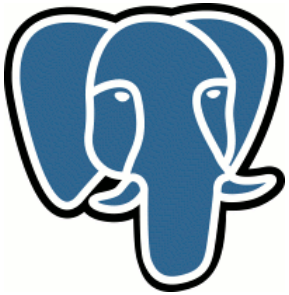
- PostgreSQL поддерживает поиск по регулярным выражениям
- До версии 9.3 это был только последовательный поиск



Обратные индексы на n-грамах

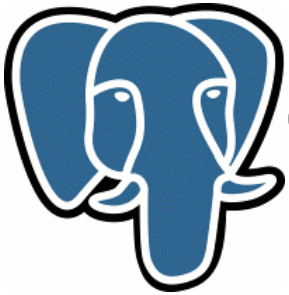
N-грамы?





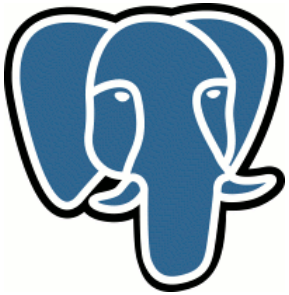
N-грамы

- N-грамы – это подстрока длины n
- Широко используются в различных нетривиальных задачах поиска по строкам



Обратные индексы на n-грамах

- Обратный индекс на n-граммах содержит соответствие между n-граммой и всеми строками, где она встречается.
- Модуль PostgreSQL pg_trgm содержит реализацию такого индекса для n = 3.

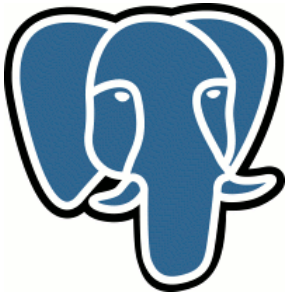


pg_trgm

1. "regular expressions",
2. "expressive speach",
3. "regular speaker"

=>

' e': {1,2}	'egu': {1,3}	'pre': {1,2}
' r': {1,3}	'er ': {3}	'reg': {1,3}
' s': {2,3}	'ess': {1,2}	'res': {1,2}
' ex': {1,2}	'exp': {1,2}	'sio': {1}
' re': {1,3}	'gul': {1,3}	'siv': {2}
' sp': {2,3}	'ion': {1}	'spe': {2,3}
'ach': {2}	'ive': {2}	'ssi': {1,2}
'ake': {3}	'ker': {3}	'ula': {1,3}
'ar ': {1,3}	'lar': {1,3}	've ': {2}
'ch ': {2}	'ns ': {1}	'xpr': {1,2}
'eac': {2}	'ons': {1}	
'eak': {3}	'pea': {2,3}	

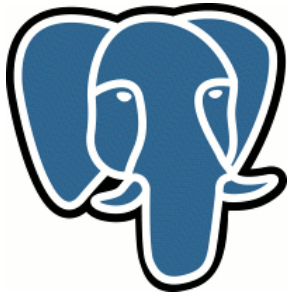


Частоты N-грам

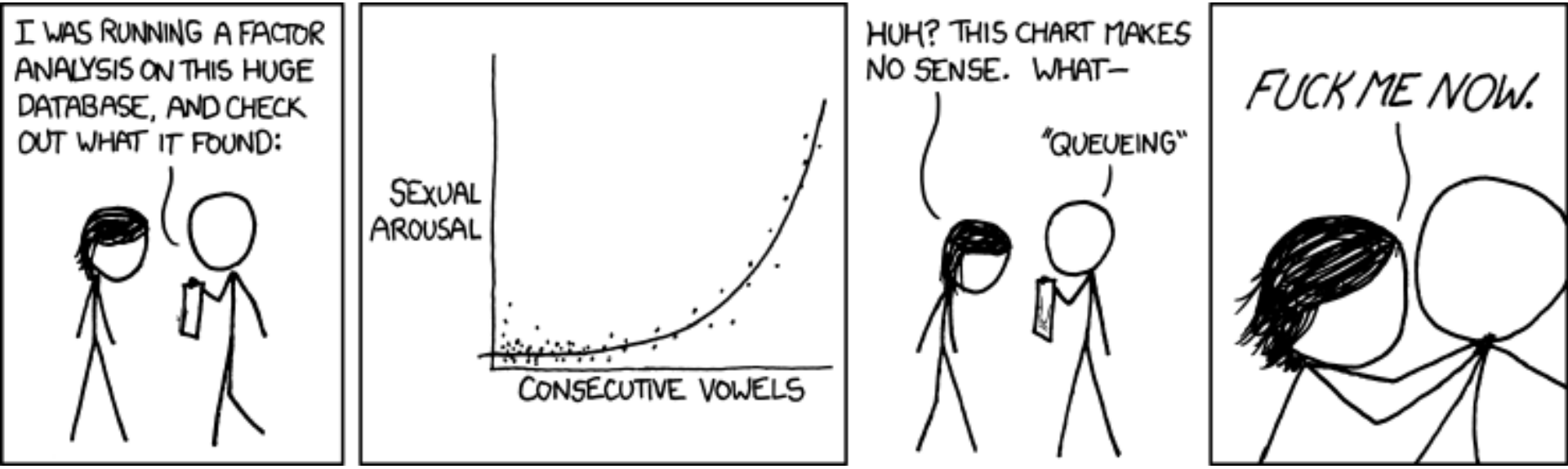
DBLP: 2.5M заголовков статей

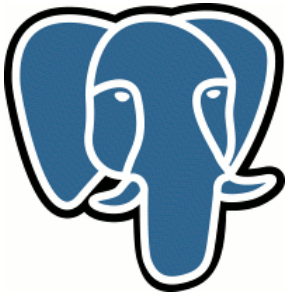
- “the” - 360K
- “zzz” – 1

  глава "Zzzzzzzzzzzzzzzzzzzzzzzzz" книги "Formal Specification and Development in Z and B" от David Everett



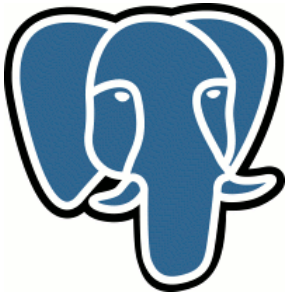
Не все N-грамы одинаково полезны





V-грамы и мультиграмы

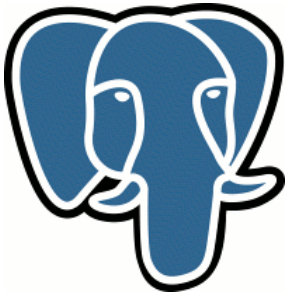
- У каждой n-грамы – своё индивидуальное n
- Выше эффективность!



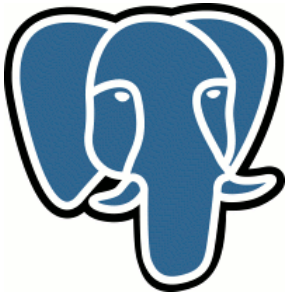
V-грамы и мультиграмы

Проблемы:

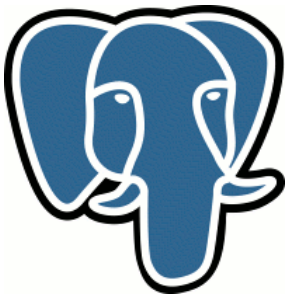
- Трудно поддерживать актуальность
- …



Патентные тролли!



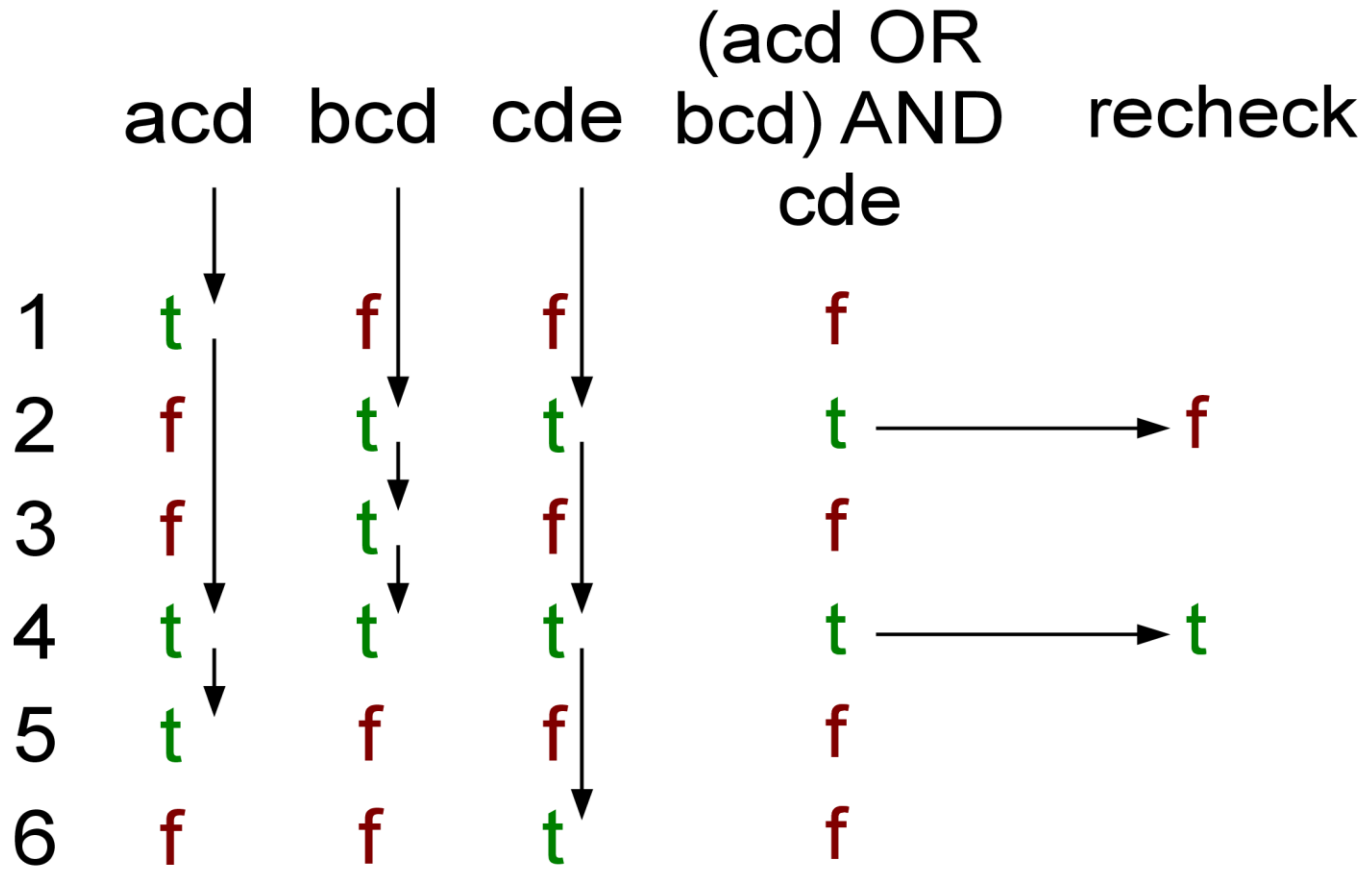
Как использовать индекс для поиска по регэкспам?

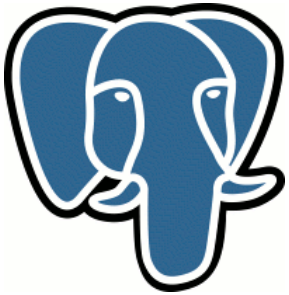


Общая идея

/[ab]cde/ => (acd OR bcd) AND cde

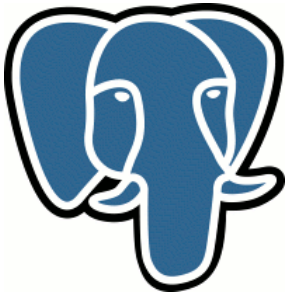
acd: {1,4,5}, bcd: {2,3,4}, cde:{2,4,6}



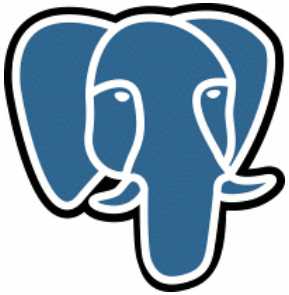


/[ab]cde/ => (acd OR bcd) AND cde

Как это сделать в общем случае?



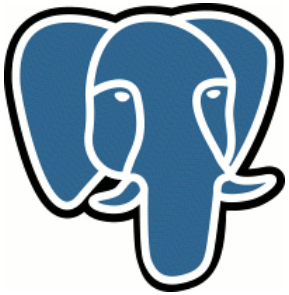
Существующие подходы к извлечению n-грам из регулярных выражений



Научная работа

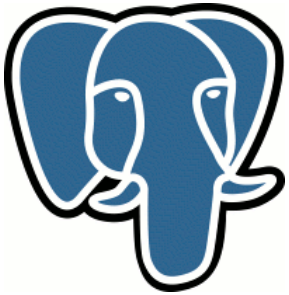
Junghoo Ch and Sridhar Rajagopalan, **A fast regular expression indexing engine**, Proceedings 18th International Conference on Data Engineering, 2002

Всё ещё широко цитируется, как актуальное решение в данной области.



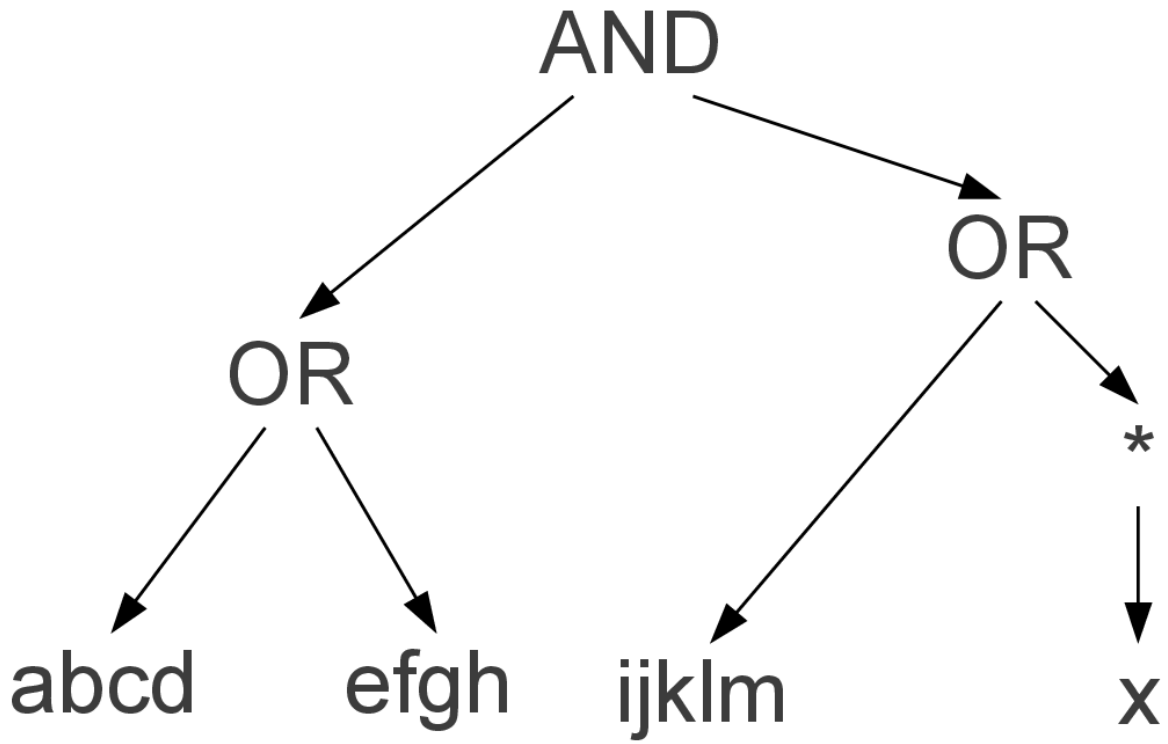
Метод FREE

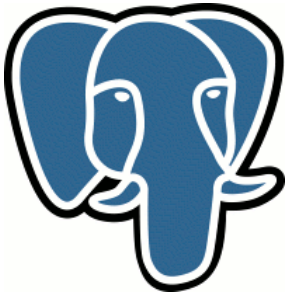
- Извлечь дерево непрерывных участков строки из регэкспа.
- Преобразование этих непрерывных кусков в мультиграмы (n-грамы с изменяемым n)
- Использовать инвертированный индекс на мультиграмах для выполнения запроса



Метод FREE: пример

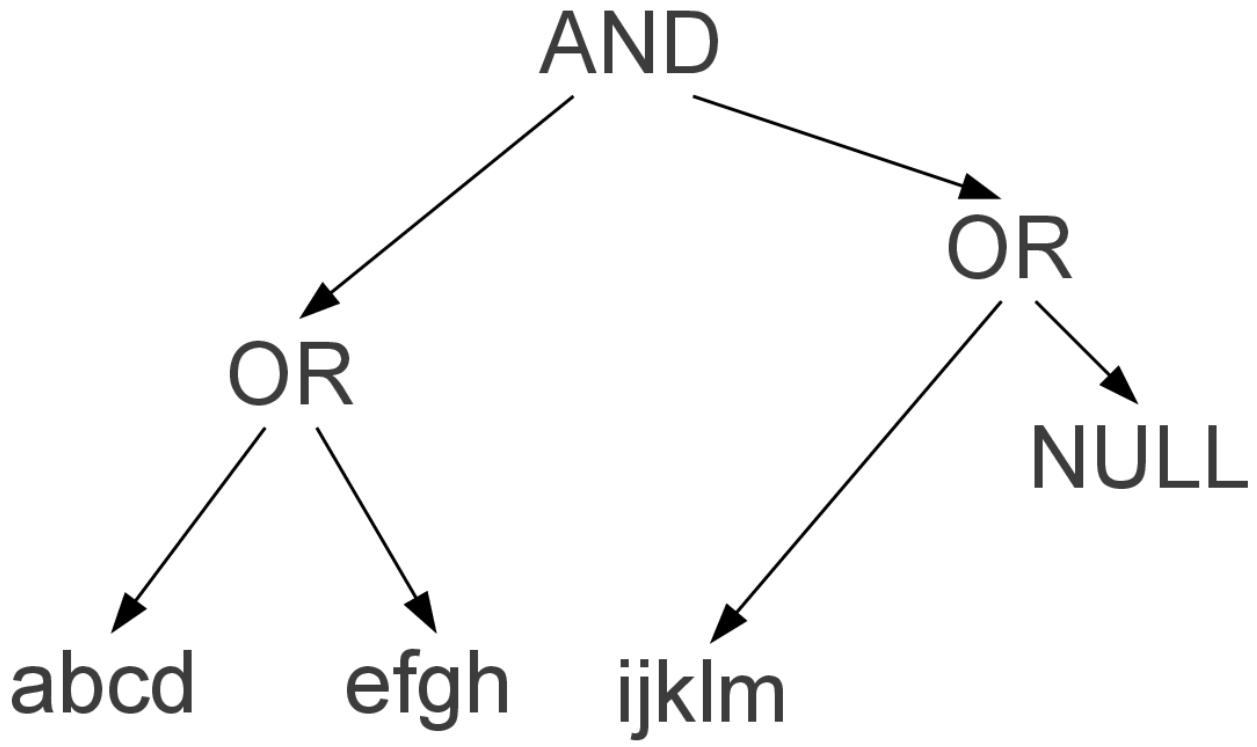
Дерево для /(abcd|efgh)(ijklm|x*)/





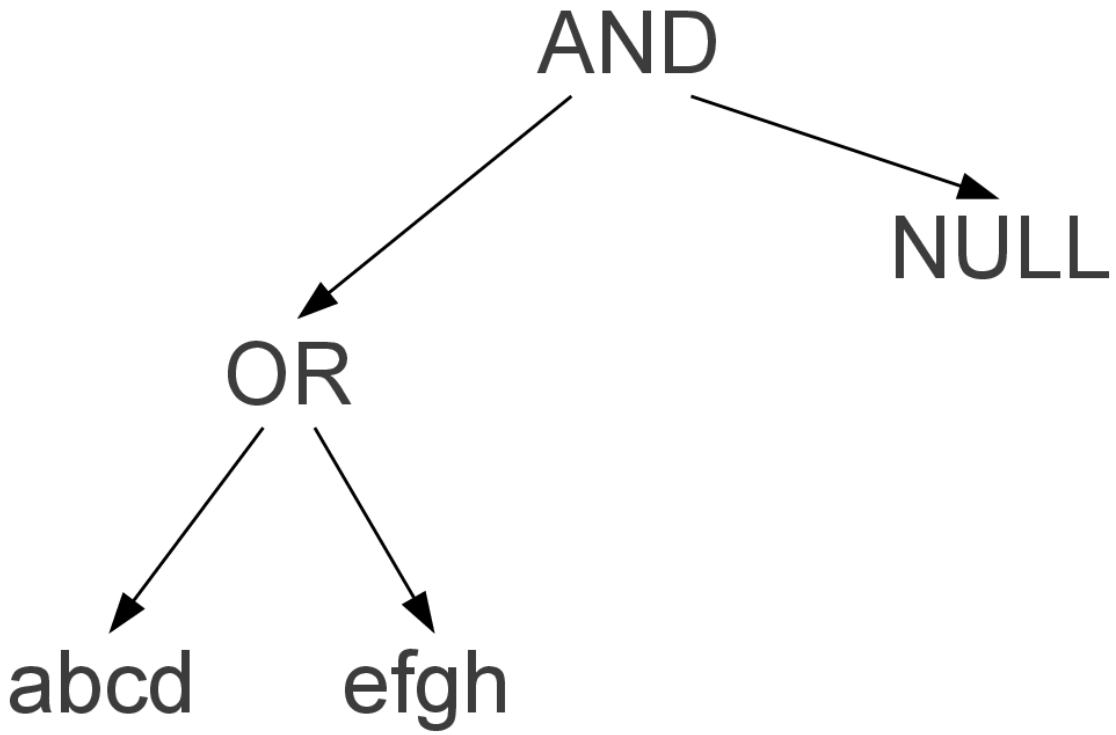
Метод FREE: пример

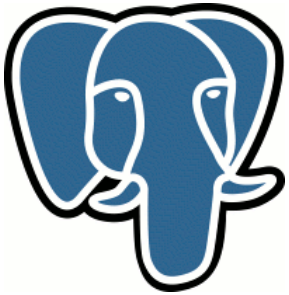
Заменить узлы “*” на NULL



Метод FREE: пример

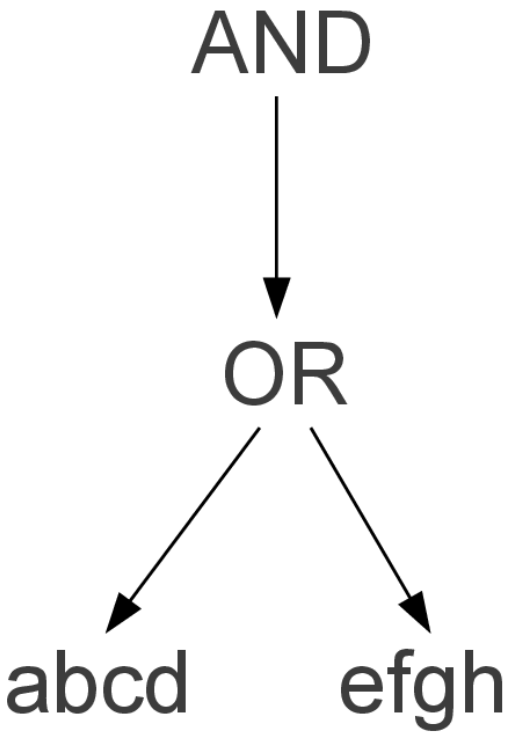
NULL “съедает” родительский узел OR

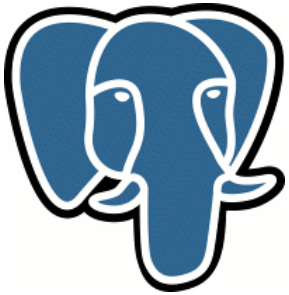




Метод FREE: пример

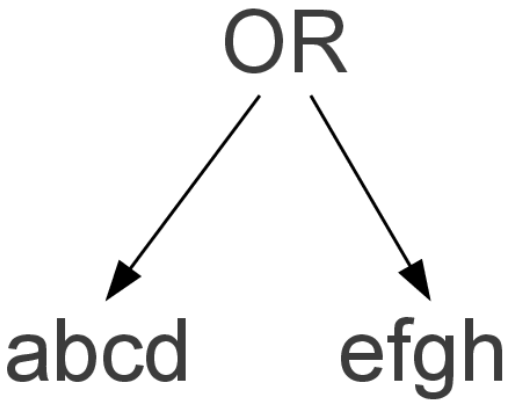
Узел AND “съедает” дочерний NULL

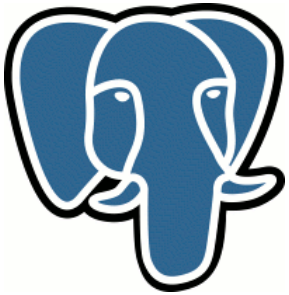




Метод FREE: пример

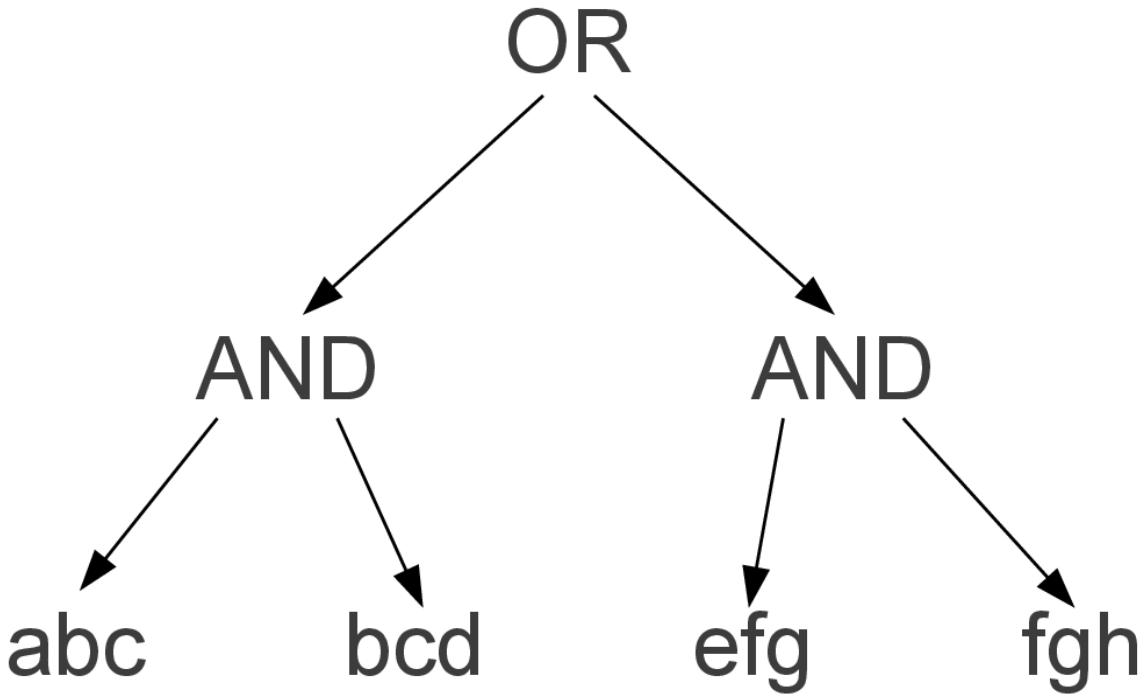
Уберём вырожденный AND

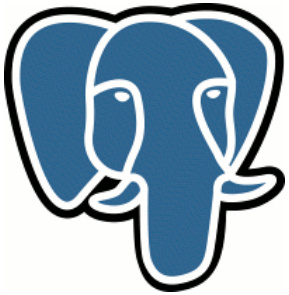




Метод FREE: пример

Разобьем непрерывные куски строк на n-грамы



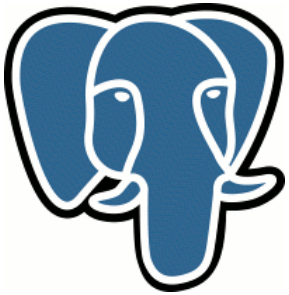


Google code search

Google™
/*Code Search*/ LABS

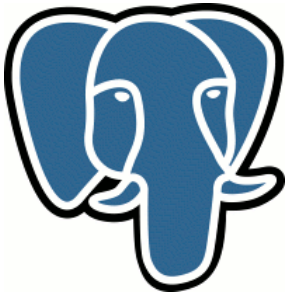
Search Code

Search public source code.



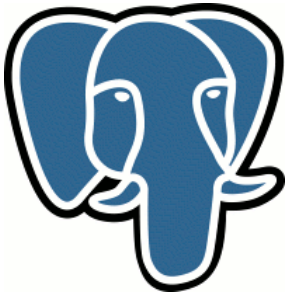
Google code search

- Запустился в 2006.
- Скорее всего использовал что-то получше, чем предыдущий метод.
- Но мы не знаем что... :(



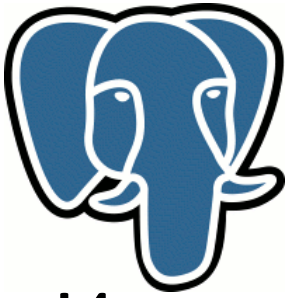
Мы не знали что, пока..

- Google code search закрылся в 2011 :(
- http://swtch.com/~rsc/regexp/regexp4.html
- Более 5 лет интриги!



Метод google code search

- 5 характеристик: emptyable, exact, prefix, suffix, match.
- Характеристики рекурсивно объединяются
- Инвертированный индекс на триграммах



Метод google code search

Исходное выражение: /a(bc)+d/

a: {exact: a}

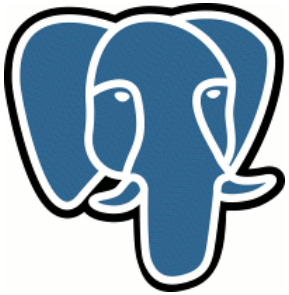
bc: {exact: bc}

d: {exact: d}

(bc)+: {prefix:bc, suffix: bc}

a(bc)+: {prefix:abc, suffix:bc}

a(bc)+d: {prefix:abc, suffix:bcd}

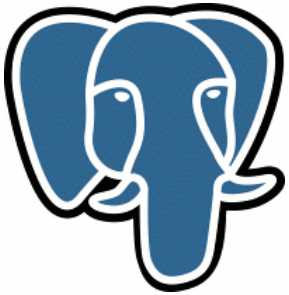


Метод google code search

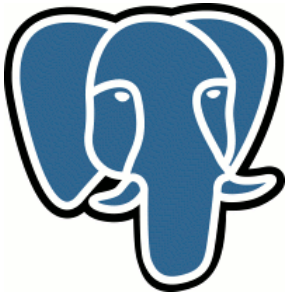
/a(bc)+d/

{prefix:abc, suffix:bcd}

abc AND bcd



Предлагаемый метод

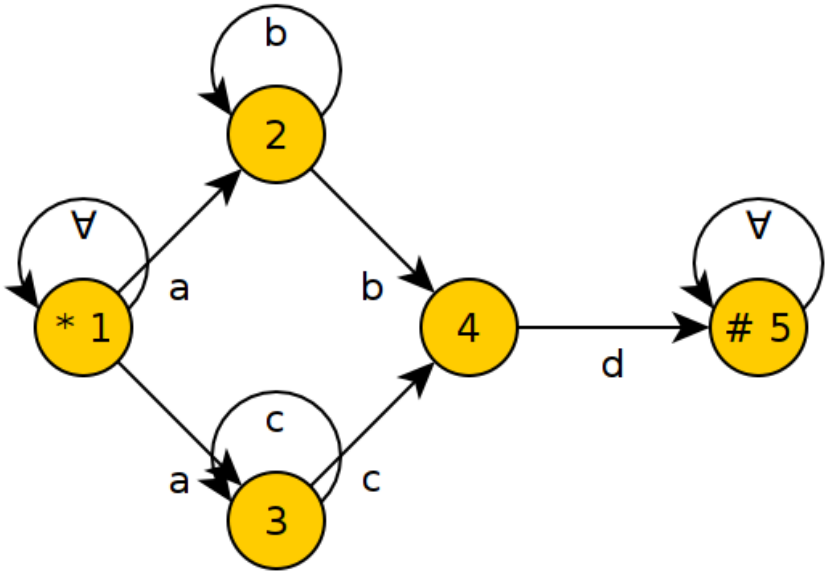


Предлагаемый метод

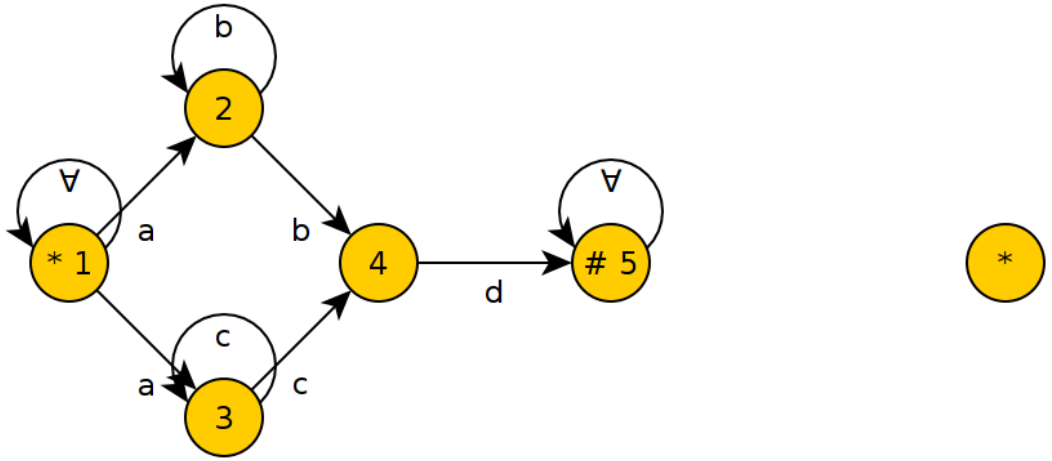
- Анализировать не само регулярное выражение, а соответствующий ему лингвистический автомат.
- Преобразовывать лингвистический автомат в контактную схему на триграммах.
- Использовать обратный индекс модуля pg_trgm вместе с полученной контактной схемой.

Пример преобразования

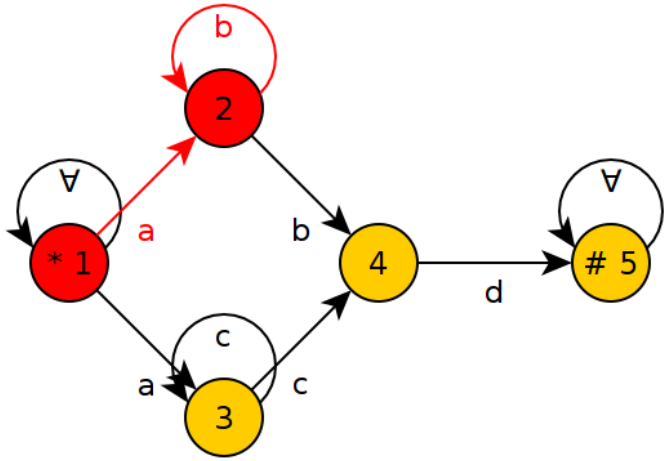
/a(b+|c+)d/



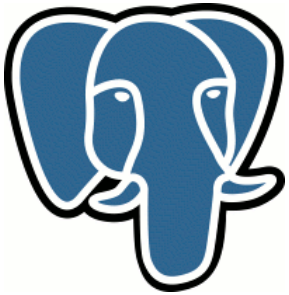
Пример преобразования



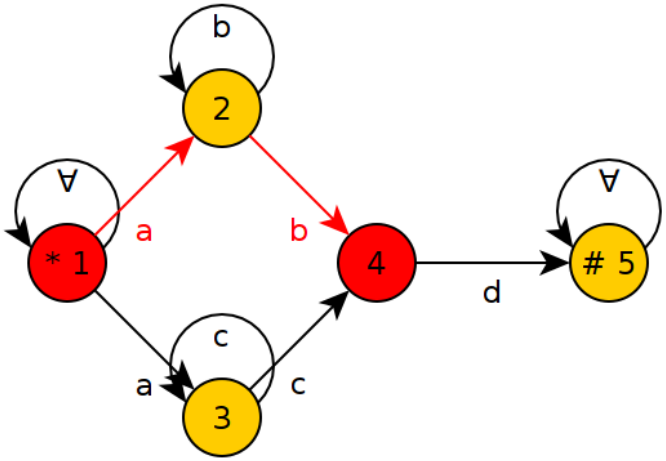
Пример преобразования



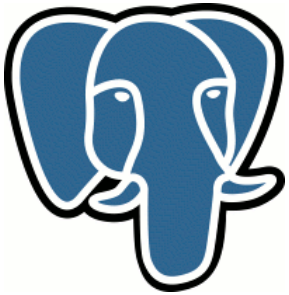
(ab, 2)



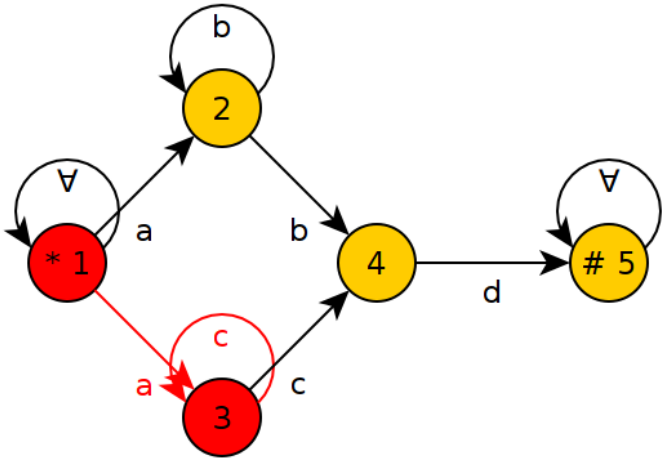
Пример преобразования



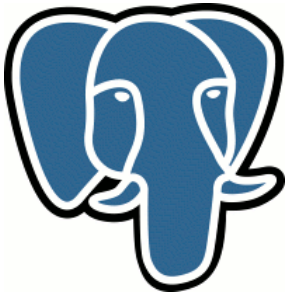
(ab, 2)
(ab, 4)



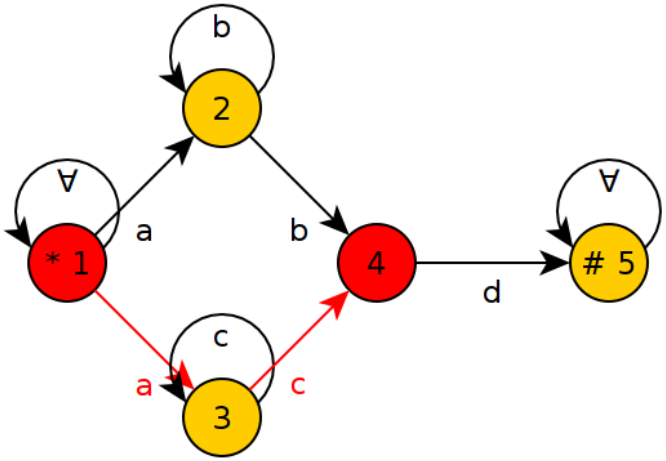
Пример преобразования



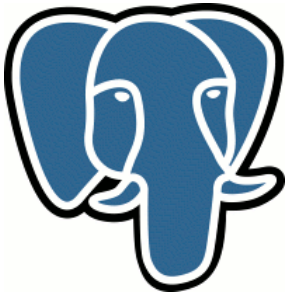
(ab, 2)
(ab, 4)
(ac, 3)



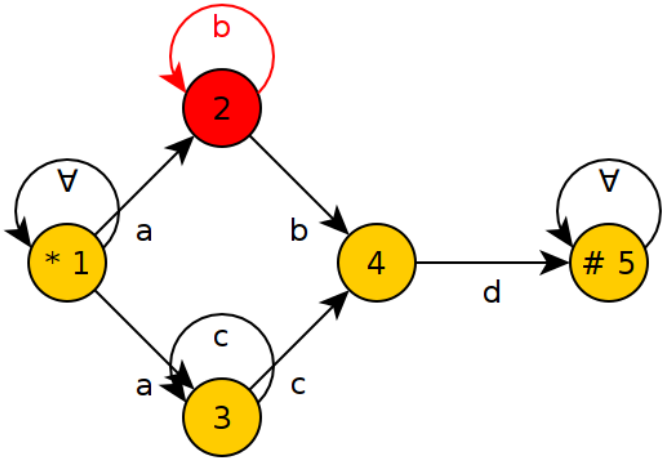
Пример преобразования

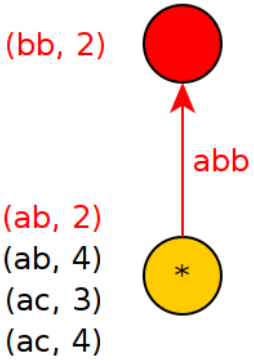


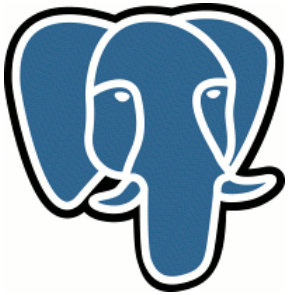
(ab, 2)
(ab, 4)
(ac, 3)
(ac, 4)



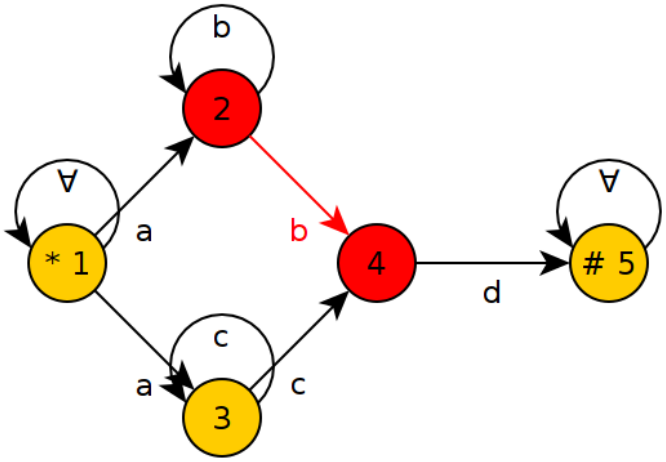
Пример преобразования

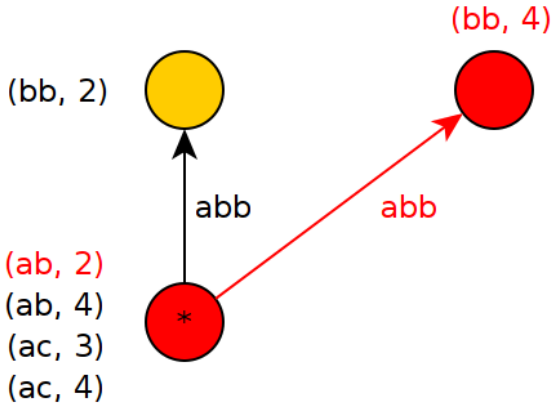


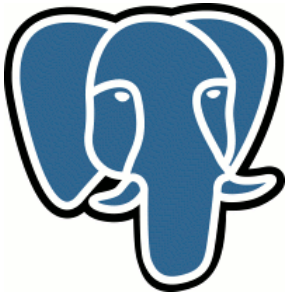




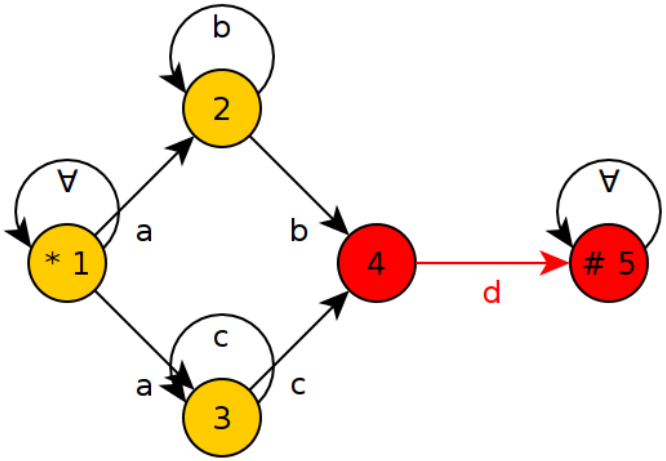
Пример преобразования

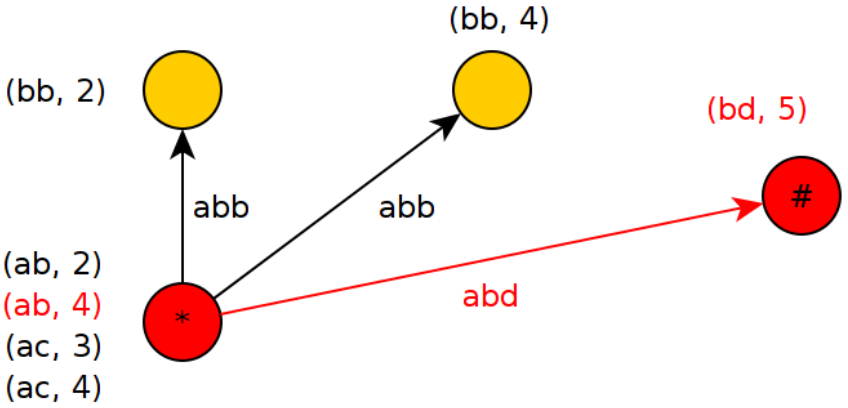


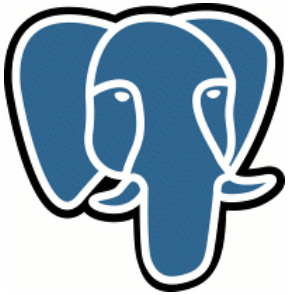




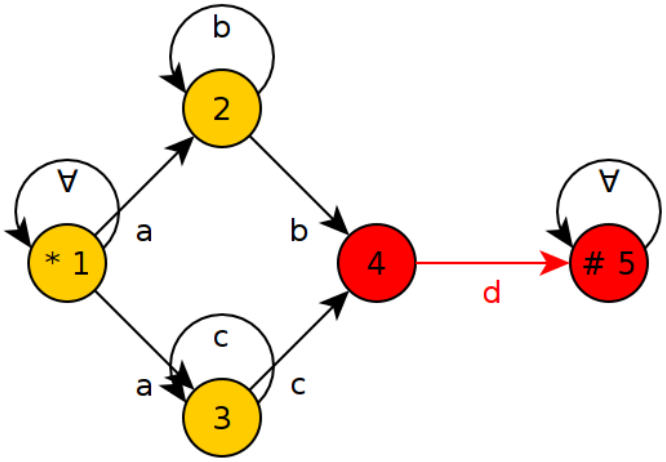
Пример преобразования

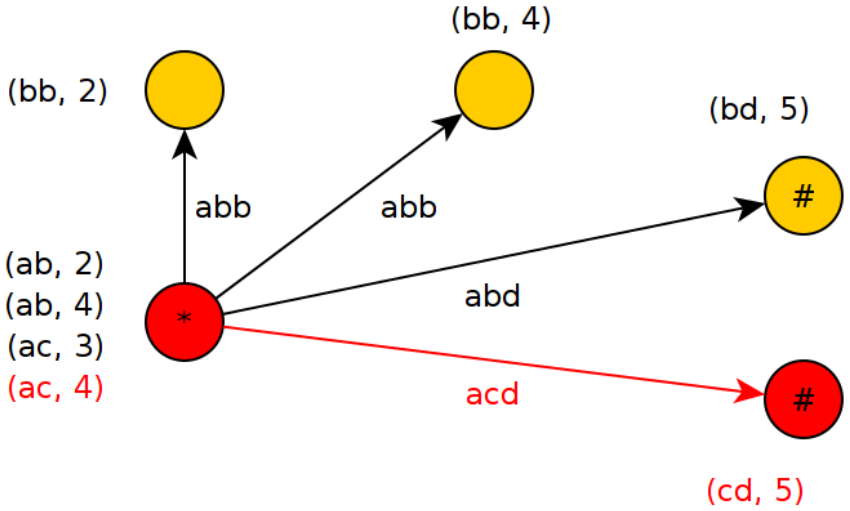


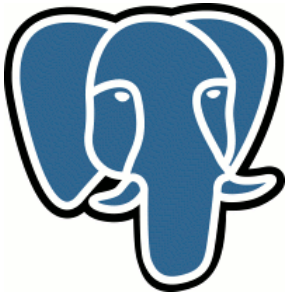




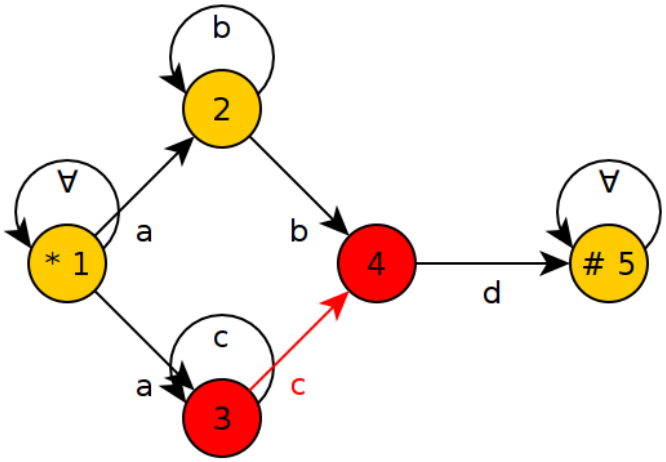
Пример преобразования

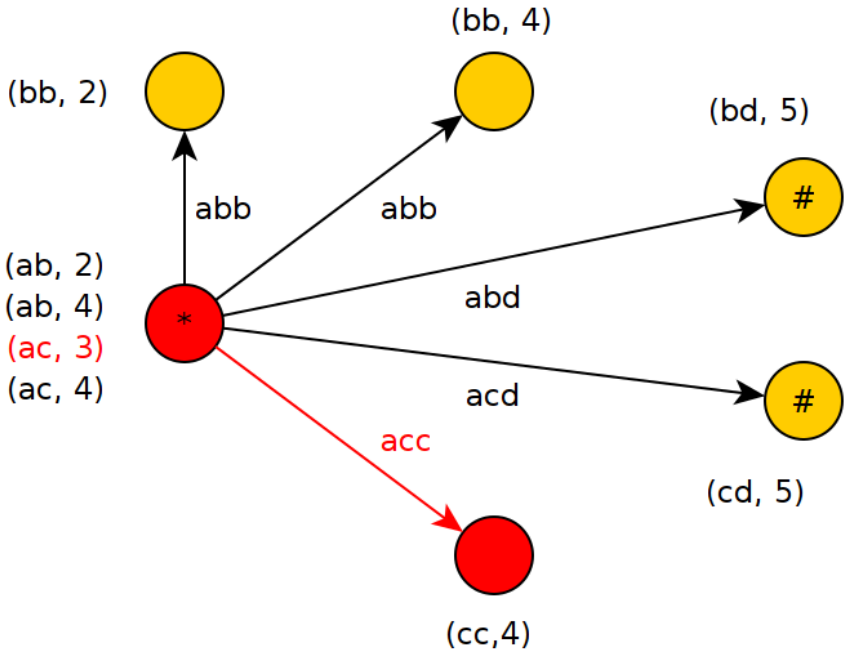


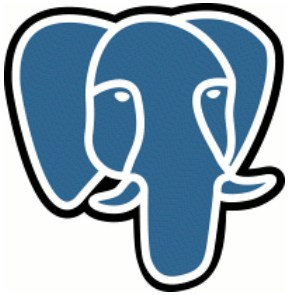




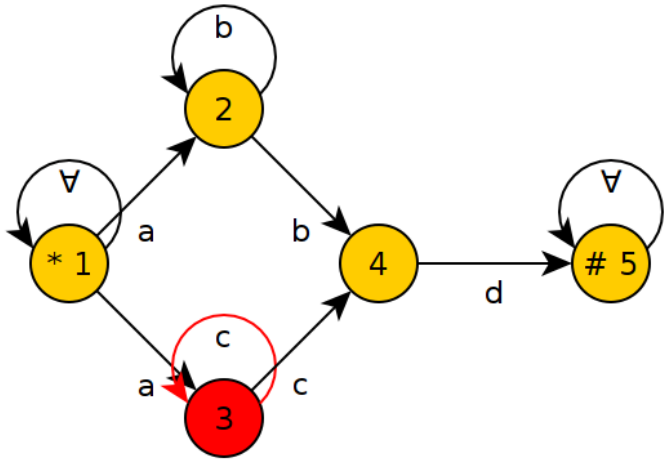
Пример преобразования

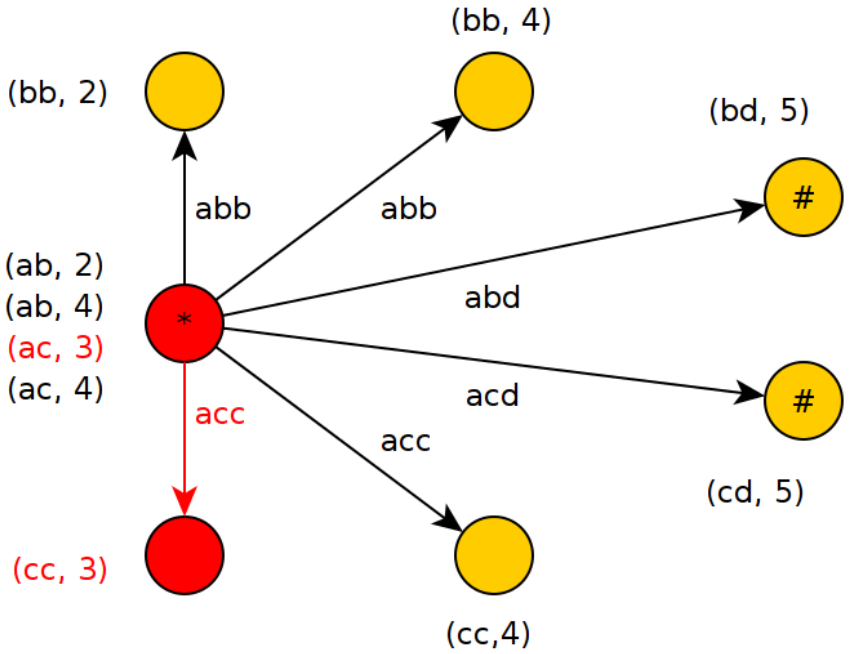


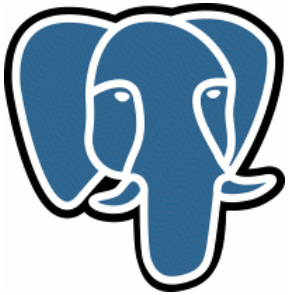




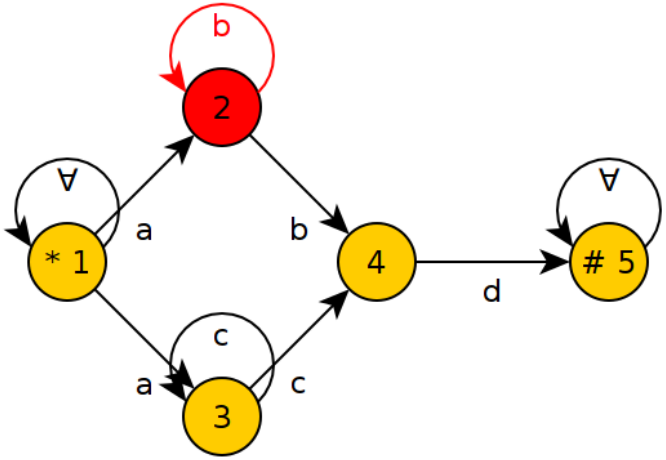
Пример преобразования

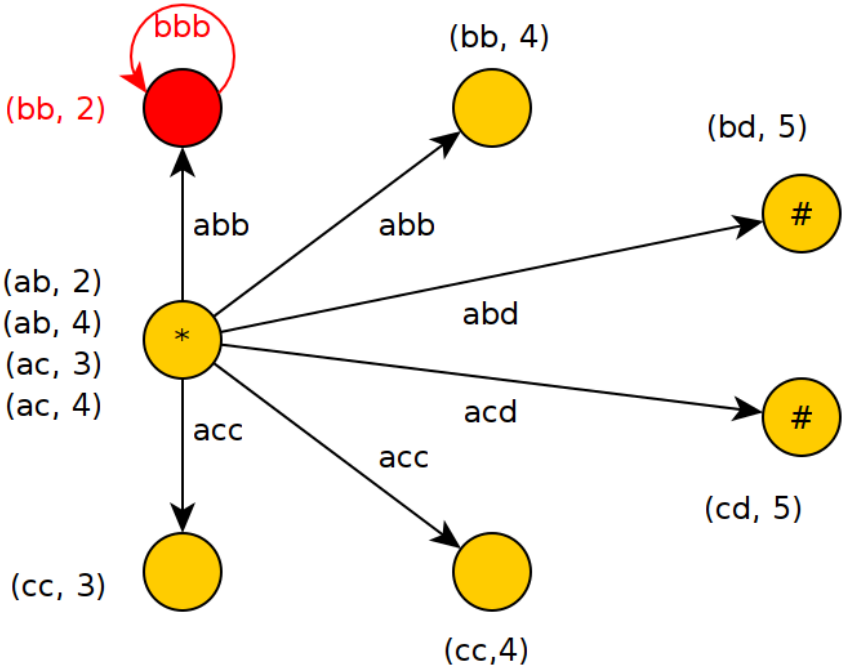


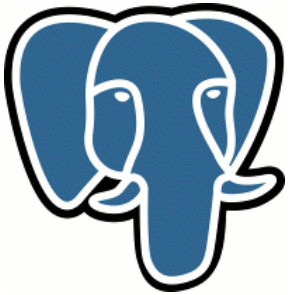




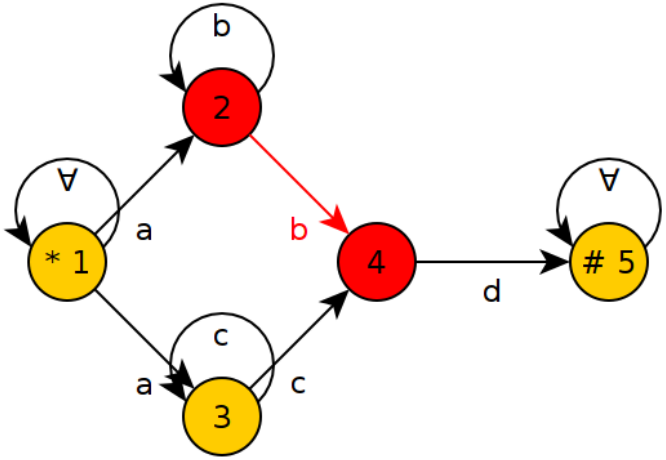
Пример преобразования

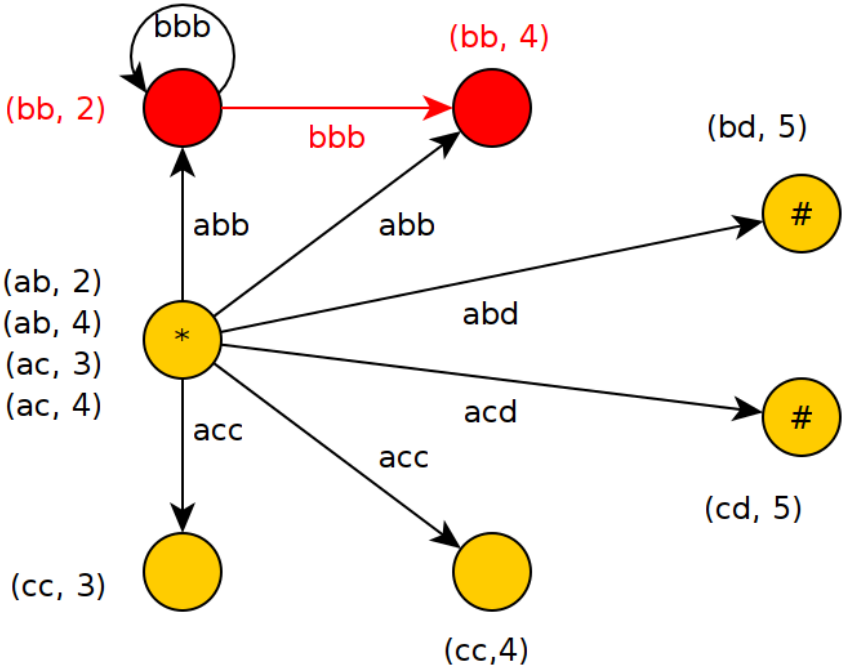


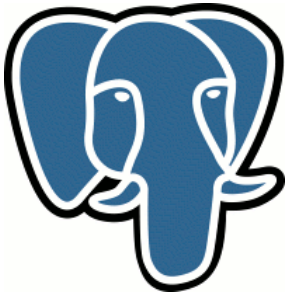




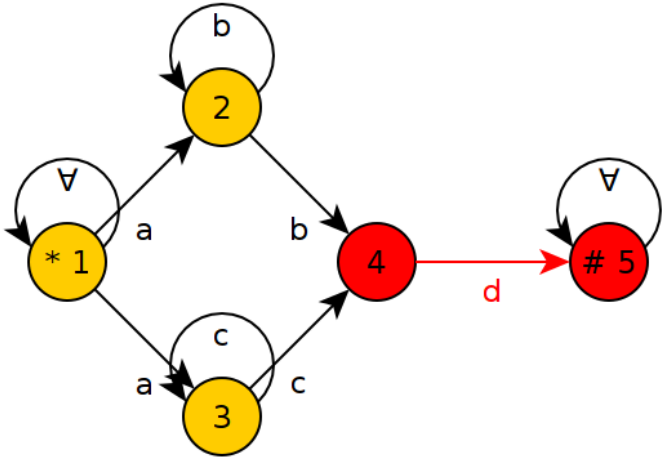
Пример преобразования

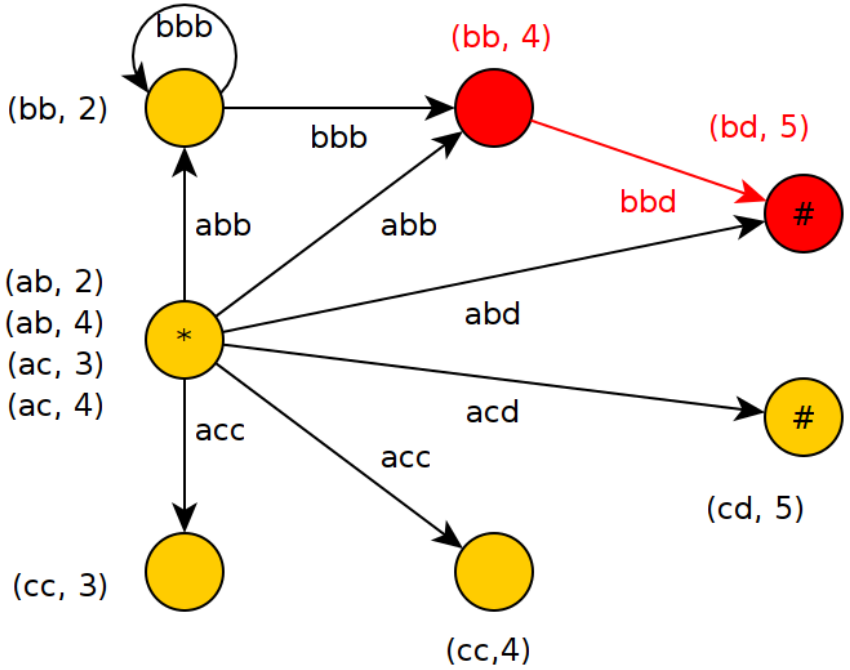


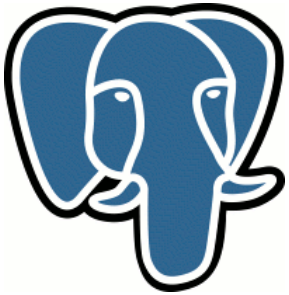




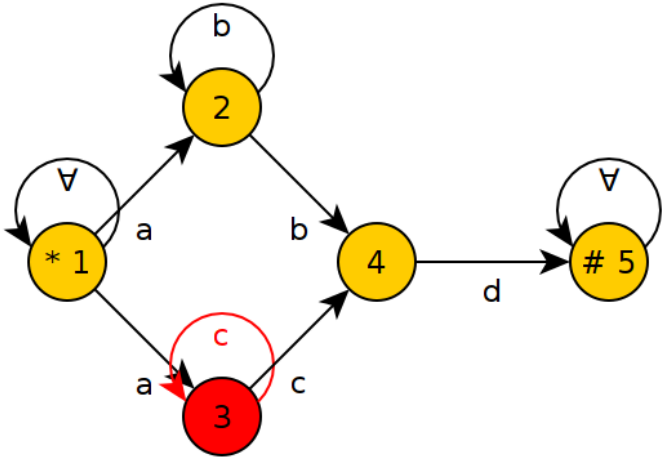
Пример преобразования

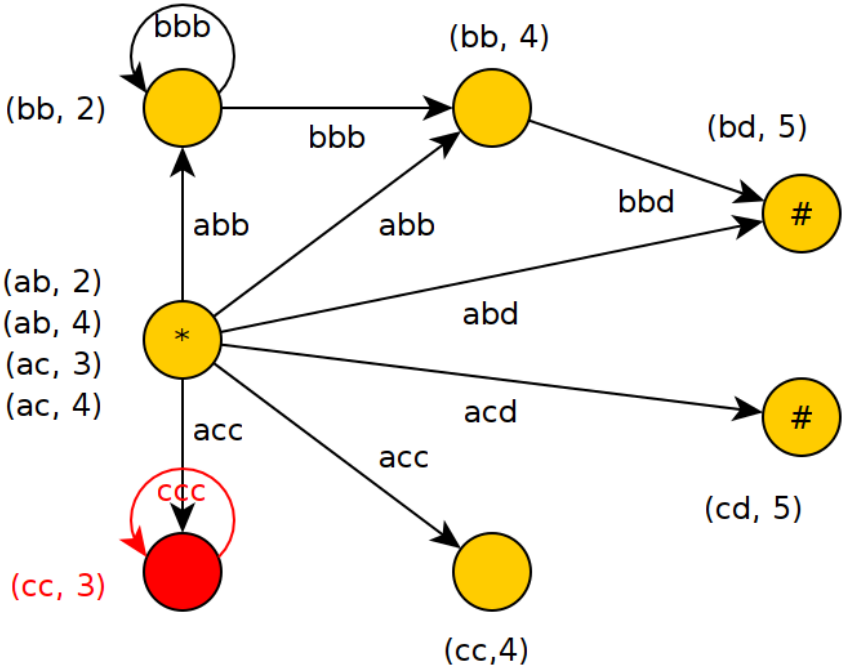


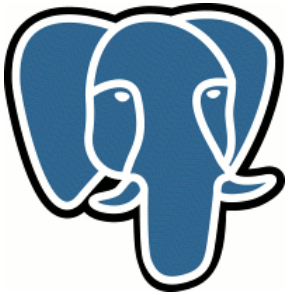




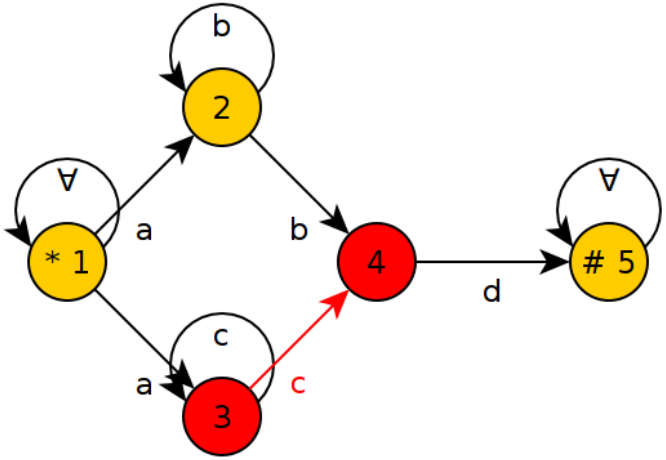
Пример преобразования

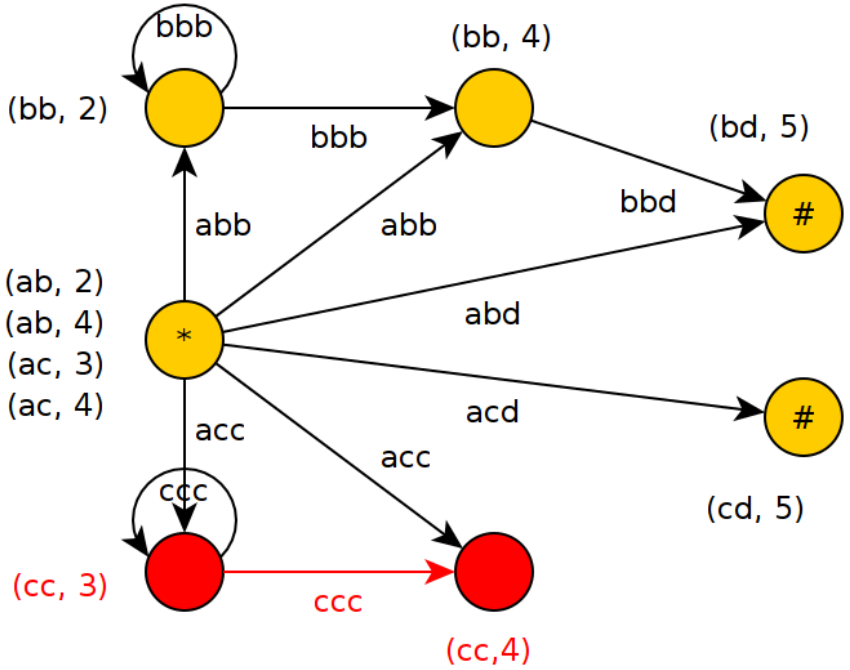


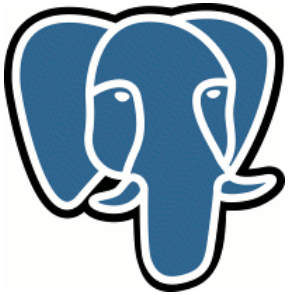




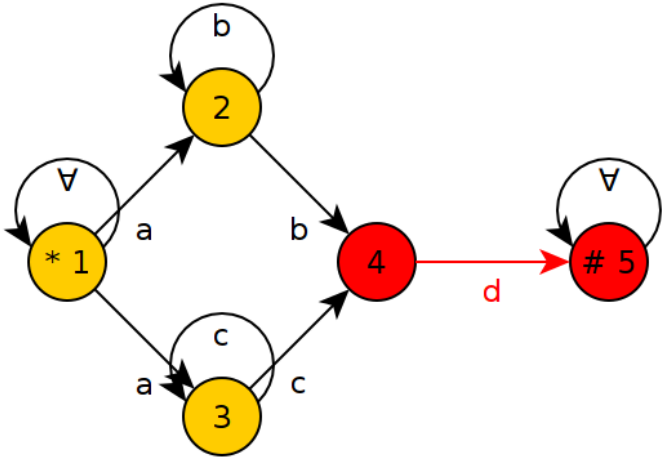
Пример преобразования

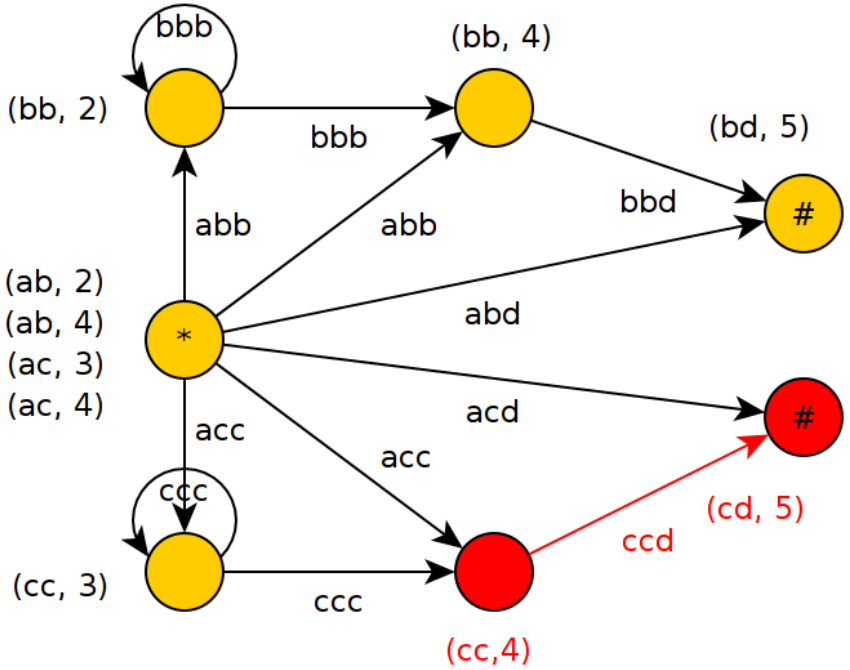


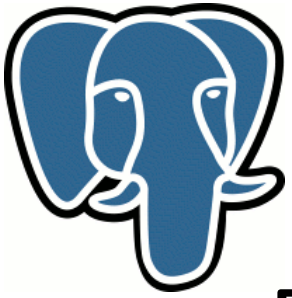




Пример преобразования

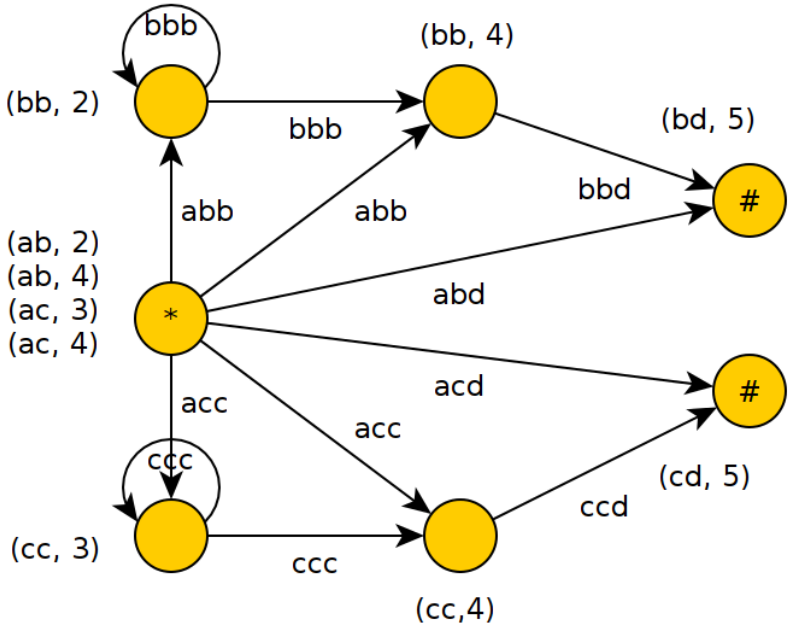


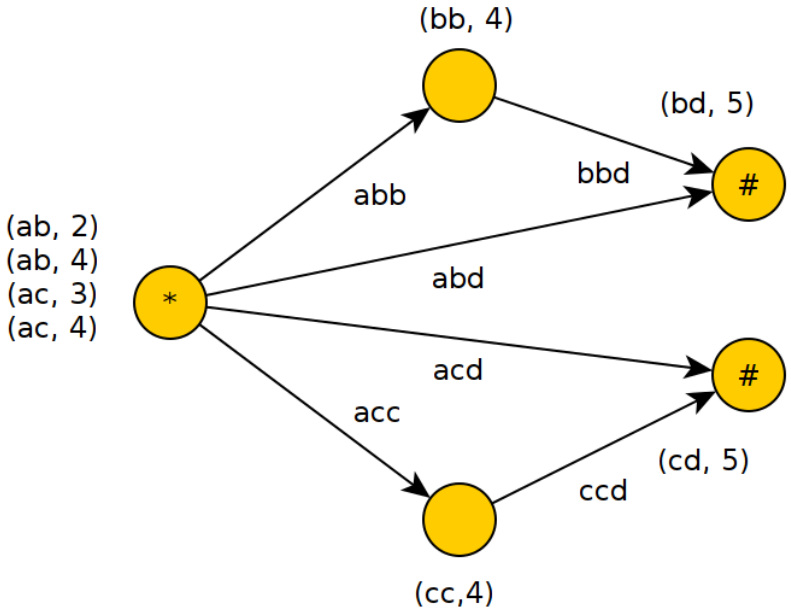


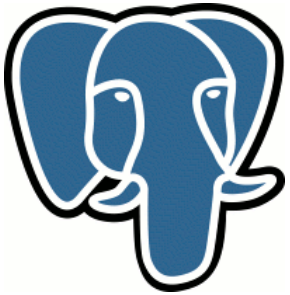


Пример преобразования

Результат можно упростить







Сравнение на примерах

Regex: /(abc|cba)def/

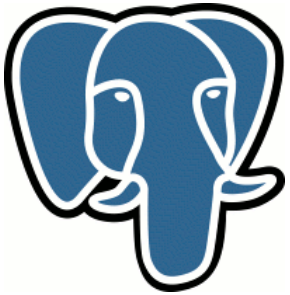
FREE: (abc OR cba) AND def

GSC:

def AND ((abc AND bcd AND cde) OR (ade AND bad AND cba))

Предложенный метод:

(abc AND bcd AND cde AND def) OR (ade AND bad AND cba AND def)



Сравнение на примерах

Regex: /abc+de/

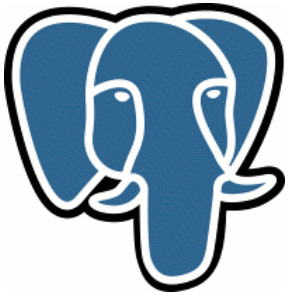
FREE: nothing

GSC: abc AND cde

Предложенный метод :

(abc AND cde AND bcd) OR

(abc AND cde AND bcc AND ccd)



Сравнение на примерах

Regex: /(abc*)+de/

FREE: nothing

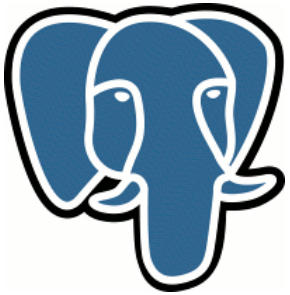
GSC: nothing

Предложенный метод :

(abd AND bde) OR

(abc AND bcd AND cde) OR

(abc AND bcc AND ccd AND cde)



Сравнение на примерах

Regex: /ab(cd)*ef/

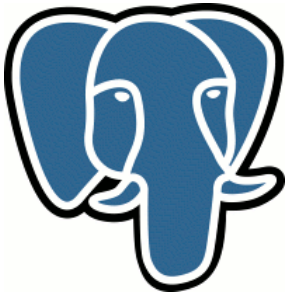
FREE: nothing

GSC: nothing

Предложенный метод :

(abe AND bef) OR

(abc AND bde AND cde AND def)



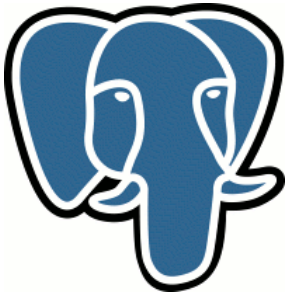
Проблемы

- Может получиться большой граф

  - Есть graceful degradation.

- Использование триграм вместо v-грам (мультиграм)

  - Смягчается с помощью GIN fast scan в PostgreSQL 9.4

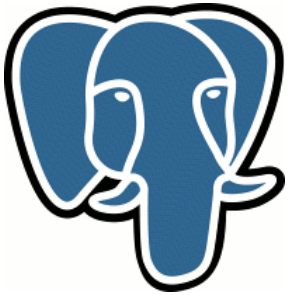


Тесты производительности

2.5 M заголовков статей DBLP средней длины 47

Regex	Seq scan, мс	9.3, мс	9.4, мс
/database.*(sql|query)/	5045	187	67
/postgres(ql)?/	4573	75	4,3
/plan+er/	3810	655	37
/(nucl|anino).*acid/	4327	560	16
/[aei](bc)+a/	2195	639	1,3

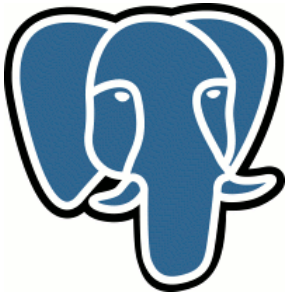
GIN fast scan рулит!



Текущее состояние

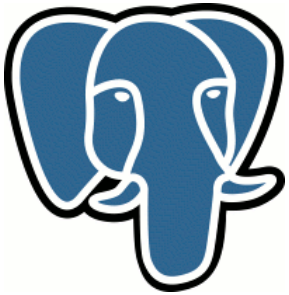
Нужно:

- PostgreSQL 9.3+
- pg_trgm contrib



Было опущено

- Детерминированные vs недетерминированные автоматы
- Группировка символов в цвета
- Начало и конец строки



Спасибо за внимание!