Поиск по образцу на последовательностях строк в БД

Евгений Зверев

разработчик YDB-платформы

HL 2023 HighLoad++

Евгений Зверев

- 20+ лет в IT
- Более 10 лет работы разработчиком и тимлидом
- Более 10 лет работы руководителем подразделений
- Опыт руководства командами продуктовой и сервисной разработки
- Опыт руководства DevOps-командами

План:

1. Обзор MATCH_RECOGNIZE
2. Таблицы vs Потоки
3. Масштабируемый пайплайн обработки
4. NFA
5. Дальнейшие планы

- Пример для антифрода
- Поиск паттерна изменения цены акций
- «Алгоритм работы»

MATCH_RECOGNIZE, SQL:2016

«RegEx на таблицах БД»

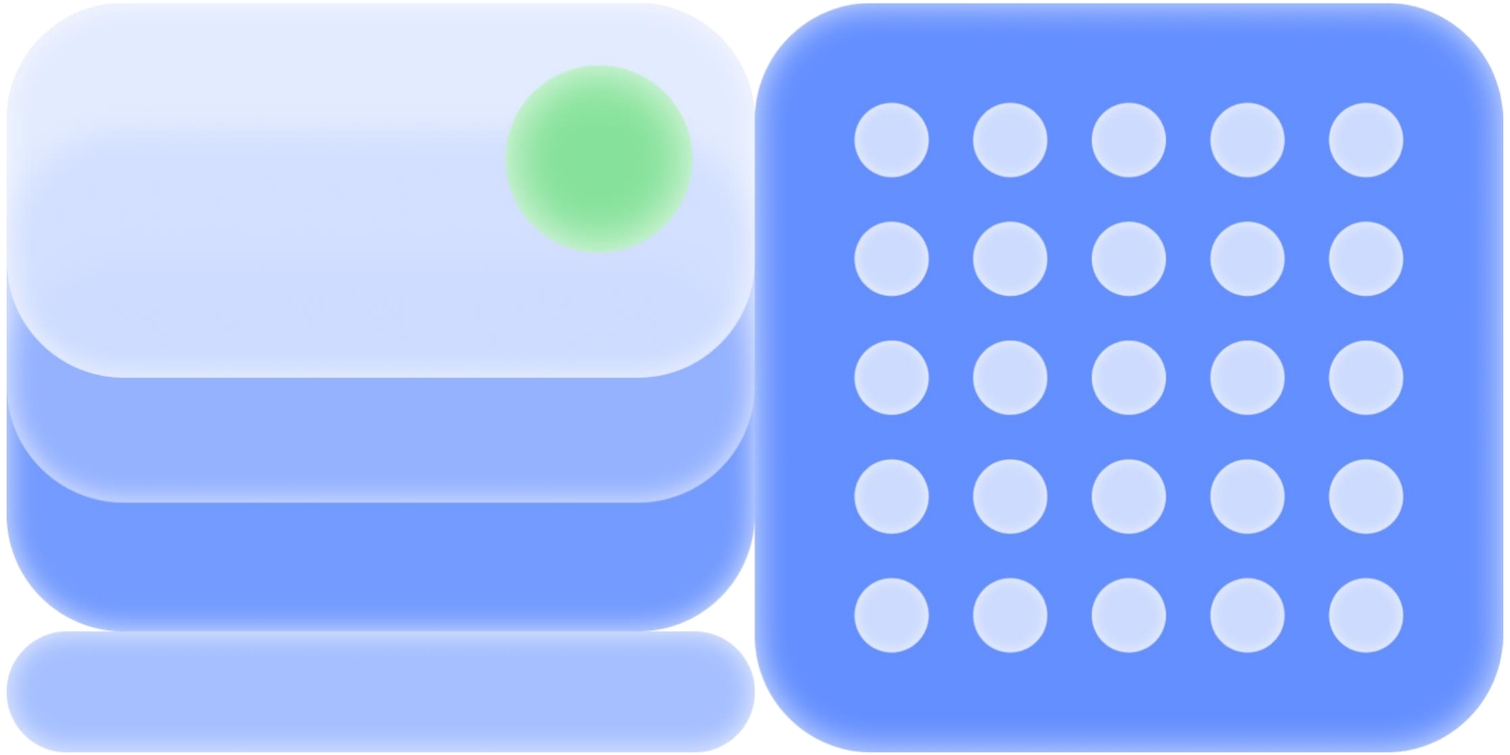
REGEX

```
(AB*){3,}C
```

PATTERN

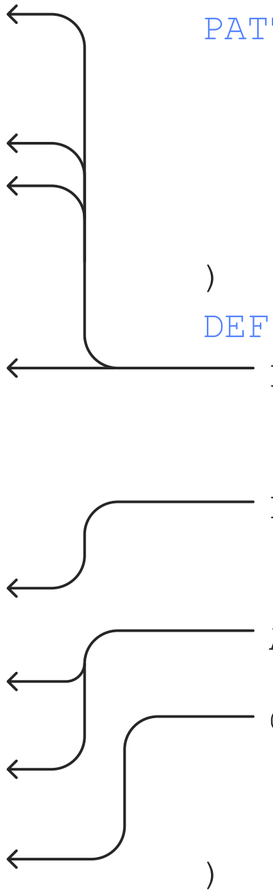
```
(LOGIN_FAIL ANY*){3,}
LOGIN_SUCCESS
```

вместо символов – предикаты над данными



Детект взлома аккаунта (антифрод)

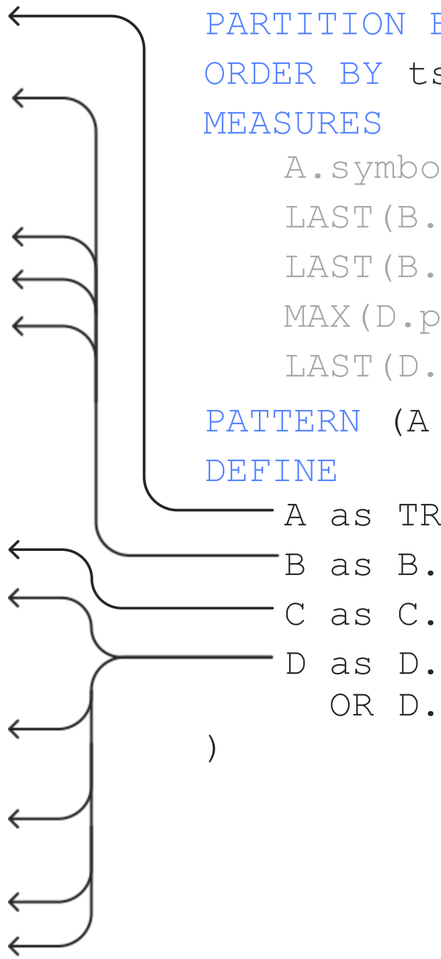
tstmp	ev_type	ev_status	user
10:00	неважно	неважно	Alice
10:01	неважно	неважно	Bob
10:02	LOGIN	FAIL	JOHN
10:03	неважно	неважно	неважно
10:03	неважно	неважно	неважно
10:04	LOGIN	FAIL	JOHN
10:05	LOGIN	FAIL	JOHN
10:05	неважно	неважно	неважно
10:05	неважно	неважно	неважно
10:05	неважно	неважно	неважно
10:06	LOGIN	FAIL	JOHN
10:06	неважно	неважно	неважно
10:06	неважно	неважно	неважно
10:06	неважно	неважно	неважно
10:06	неважно	неважно	неважно
10:07	LOGIN	SUCCESS	JOHN
10:08	неважно	неважно	неважно
10:09	BALANCE	$1M	JOHN
10:10	неважно	неважно	неважно
10:11	DETAILS	<..>	JOHN
10:12	неважно	неважно	JOHN
10:13	CHANGE_PIN	SUCCESS	JOHN



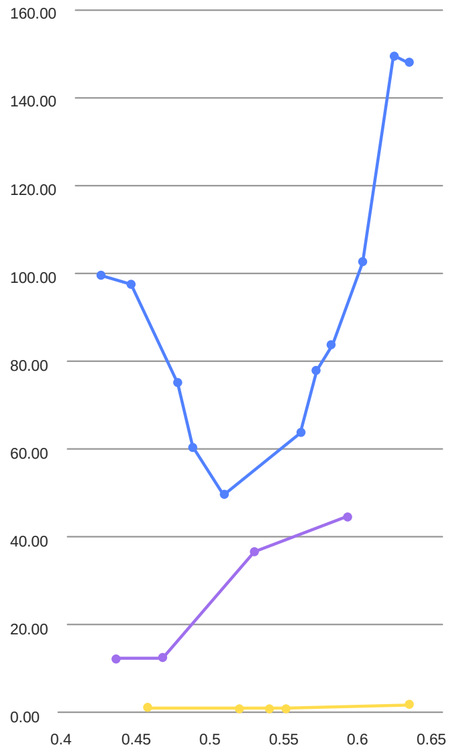
```
SELECT * FROM Events MATCH_RECOGNIZE(
PARTITION BY user
...
PATTERN (
    LoginFail{3,}
    LoginSuccess
    Any*
    ChangePin
)
DEFINE (
    LoginFail as LoginFail.ev_type = “LOGIN”
      and LoginFail.ev_status = “FAIL”,

    LoginSuccess as LoginSuccess.ev_type = “LOGIN”
      and LoginSuccess.ev_status = “SUCCESS”,

    Any as True, -- можно не писать

    ChangePin as ChangePin.ev_type = “CHANGE_PIN”
      and ChangePin.tstamp - FIRST(LoginFail.tstamp)
      < `15 min`
)
```

Поиск паттерна изменения цены

symbol	tstamp	price
XYZ	10:00	100.00
XMP	10:15	1.67
XYZ	10:30	98.00
LQS	10:45	1.23
XMP	11:00	12.92
XYZ	11:15	75.50
XYZ	11:30	61.00
XYZ	12:00	50.00
LQS	12:15	1.00
XMP	12:30	37.00
LQS	12:45	1.01
LQS	13:00	1.15
XYZ	13:15	64.00
XYZ	13:40	78.25
XYZ	13:45	84.00
XMP	14:00	45.00
XYZ	14:15	103.00
MOM	14:30	22.48
XYZ	14:45	121.00
LQS	15:00	1.97
XYZ	15:00	150.00
XYZ	15:15	148.50



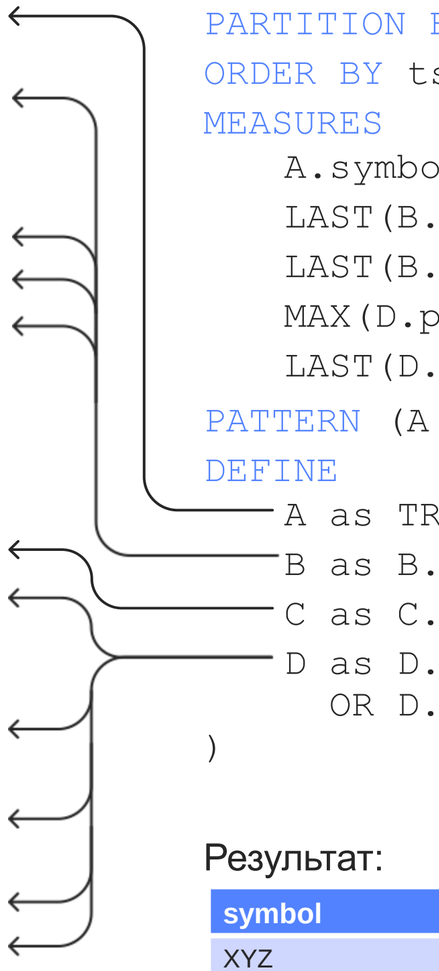
```
SELECT * FROM stock_ticker MATCH_RECOGNIZE(
PARTITION BY symbol
ORDER BY tstamp
MEASURES
    A.symbol as symbol,
    LAST(B.price) as min_price,
    LAST(B.tstamp) as min_price_tstamp,
    MAX(D.price) as max_price,
    LAST(D.price) as last_price
PATTERN (A B+ C D+)
DEFINE
    A as TRUE
    B as B.price < PREV(B.price),
    C as C.price > LAST(B.price),
    D as D.PRICE > PREV(D.PRICE)
      OR D.PRICE > AVG(B.price)
)
```



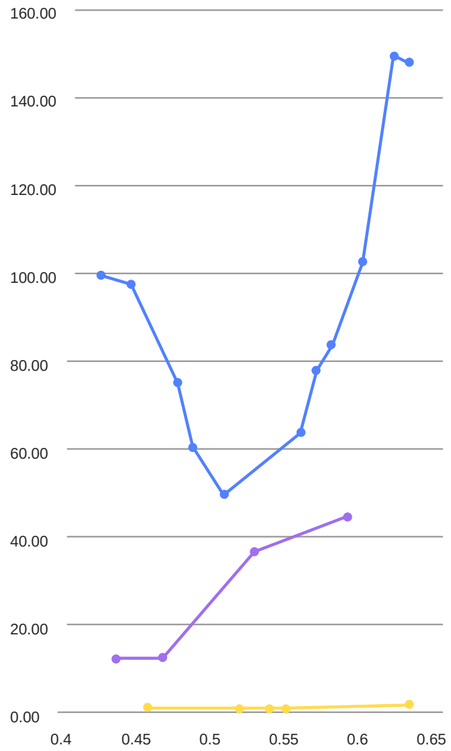
Модифицированный пример из CAN/CSA-ISO/IEC 9075-2:18

Поиск паттерна изменения цены

symbol	tstamp	price
XYZ	10:00	100.00
XMP	10:15	1.67
XYZ	10:30	98.00
LQS	10:45	1.23
XMP	11:00	12.92
XYZ	11:15	75.50
XYZ	11:30	61.00
XYZ	12:00	50.00
LQS	12:15	1.00
XMP	12:30	37.00
LQS	12:45	1.01
LQS	13:00	1.15
XYZ	13:15	64.00
XYZ	13:40	78.25
XYZ	13:45	84.00
XMP	14:00	45.00
XYZ	14:15	103.00
MOM	14:30	22.48
XYZ	14:45	121.00
LQS	15:00	1.97
XYZ	15:00	150.00
XYZ	15:15	148.50



```
SELECT * FROM stock_ticker MATCH_RECOGNIZE(
PARTITION BY symbol
ORDER BY tstamp
MEASURES
    A.symbol as symbol,
    LAST(B.price) as min_price,
    LAST(B.tstamp) as min_price_tstamp,
    MAX(D.price) as max_price,
    LAST(D.price) as last_price
PATTERN (A B+ C D+)
DEFINE
    A as TRUE
    B as B.price < PREV(B.price),
    C as C.price > LAST(B.price),
    D as D.PRICE > PREV(D.PRICE)
      OR D.PRICE > AVG(B.price)
)
```



Результат:

symbol	min_price	min_price_tstamp	max_price	last_price
XYZ	50.00	12:00	150.00	148.50

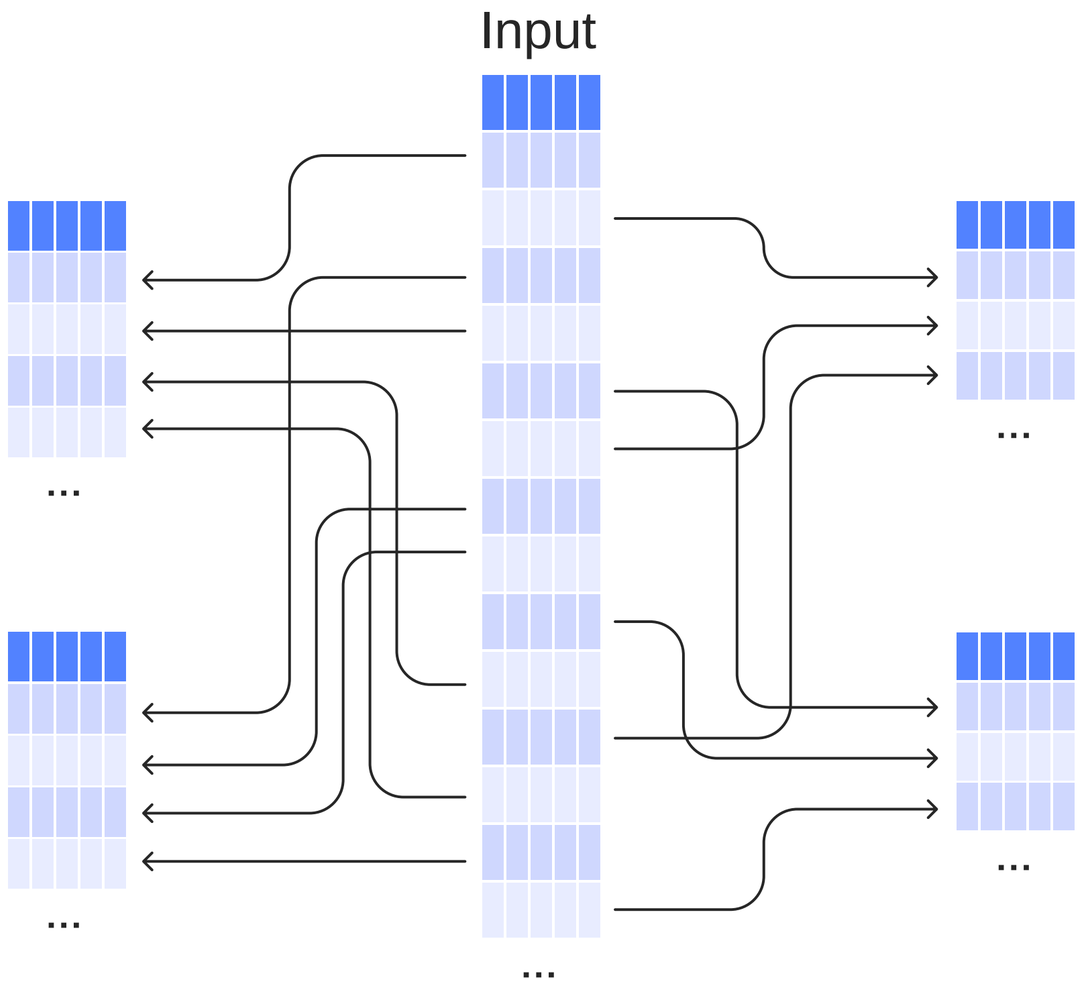
Модифицированный пример из CAN/CSA-ISO/IEC 9075-2:18

«Алгоритм работы» MATCH_RECOGNIZE


```
SELECT * FROM Input

MATCH_RECOGNIZE(
    PARTITION BY pexpr as P,…
    ORDER BY sort_expr, …
    MEASURES
       A.symbol as a_symbol, …
    ONE ROW PER MATCH
    AFTER MATCH SKIP PAST LAST ROW
    PATTERN (A…)
    DEFINE
       A as TRUE, …
)
```

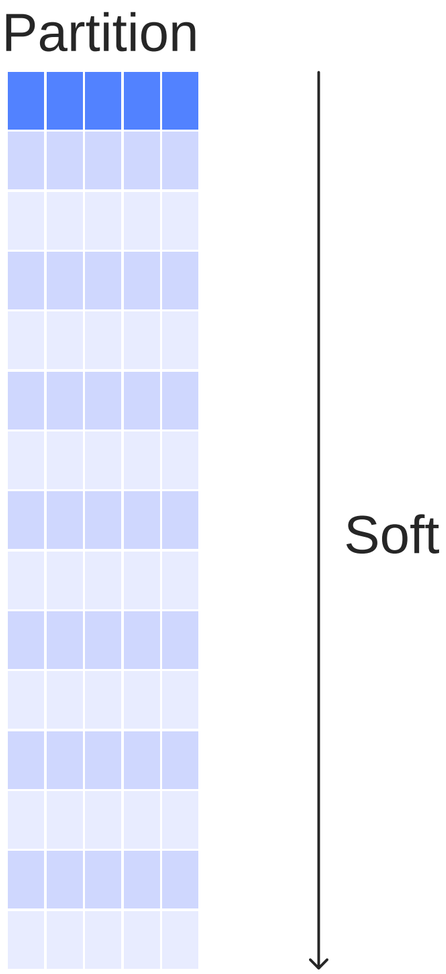
«Алгоритм работы» MATCH_RECOGNIZE

```
SELECT * FROM Input

MATCH_RECOGNIZE(

    PARTITION BY pexpr as P,…

    ORDER BY sort_expr, …

    MEASURES

       A.symbol as a_symbol, …

    ONE ROW PER MATCH

    AFTER MATCH SKIP PAST LAST ROW

    PATTERN (A…)

    DEFINE

       A as TRUE, …

)
```



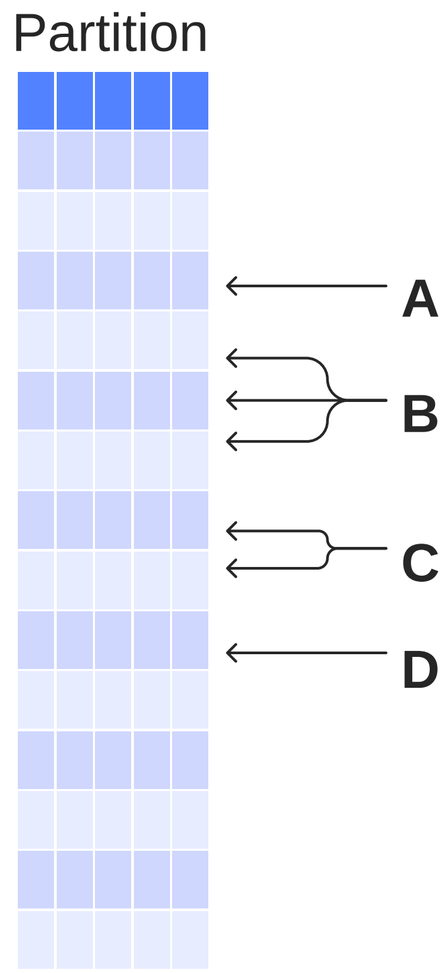
«Алгоритм работы» MATCH_RECOGNIZE


```
SELECT * FROM Input
MATCH_RECOGNIZE(
    PARTITION BY pexpr as P,…
    ORDER BY sort_expr, …
    MEASURES
       A.symbol as a_symbol, …
    ONE ROW PER MATCH
    AFTER MATCH SKIP PAST LAST ROW
    PATTERN (A…)
    DEFINE
       A as TRUE, …
)
```



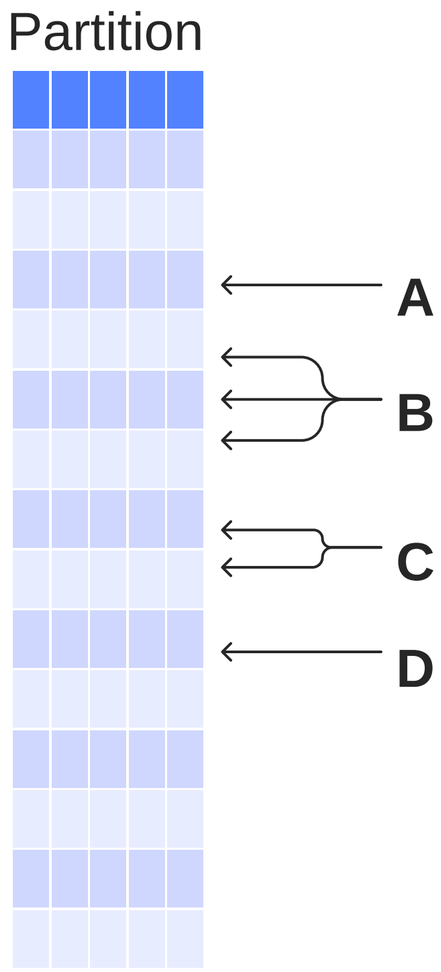
«Алгоритм работы» MATCH_RECOGNIZE


```
SELECT * FROM Input
MATCH_RECOGNIZE(
    PARTITION BY pexpr as P,…
    ORDER BY sort_expr, …
    MEASURES
       A.symbol as a_symbol, …
    ONE ROW PER MATCH
    AFTER MATCH SKIP PAST LAST ROW
    PATTERN (A B+ C* D)
    DEFINE
       A as TRUE, …
)
```



«Алгоритм работы» MATCH_RECOGNIZE

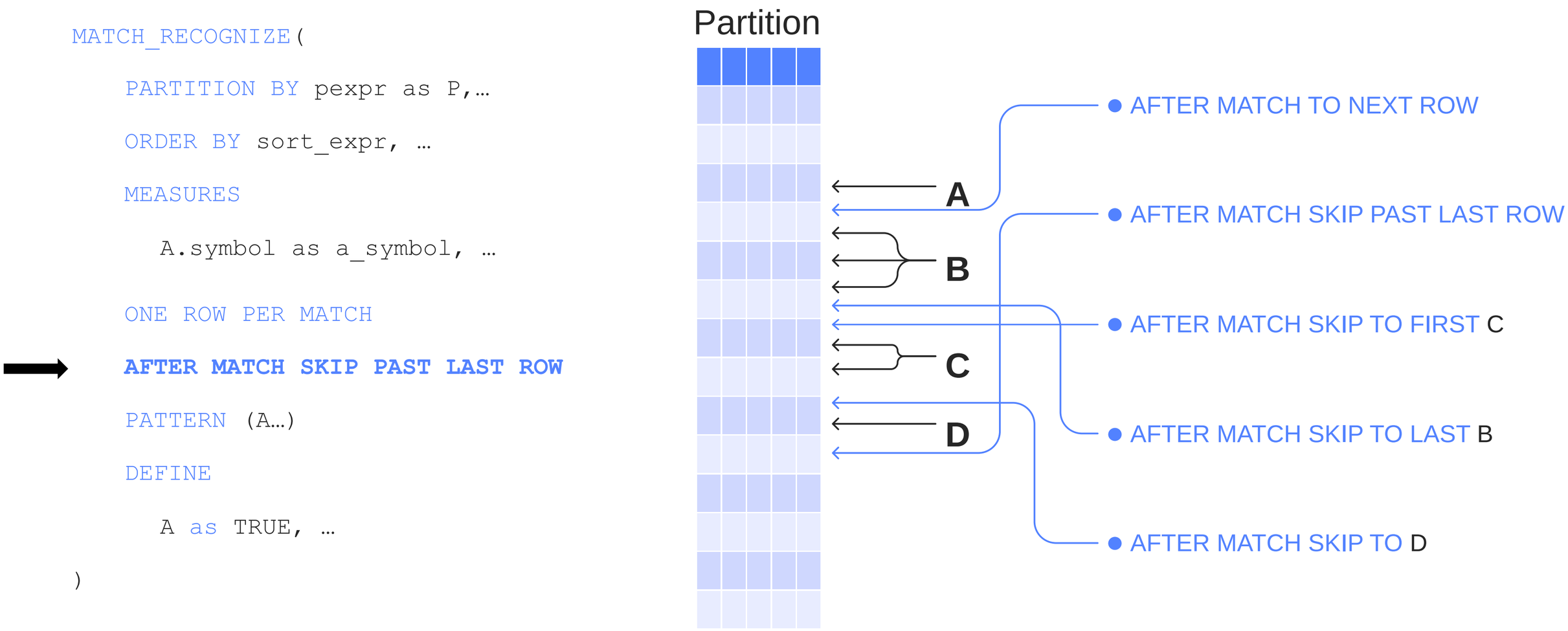
```
SELECT * FROM Input

MATCH_RECOGNIZE(

    PARTITION BY pexpr as P,…

    ORDER BY sort_expr, …

    MEASURES

       A.symbol as a_symbol, …

    ONE ROW PER MATCH

    AFTER MATCH SKIP PAST LAST ROW

    PATTERN (A…)

    DEFINE

       A as TRUE, …

)
```



- ONE ROW PER MATCH
- ALL ROWS PER MATCH

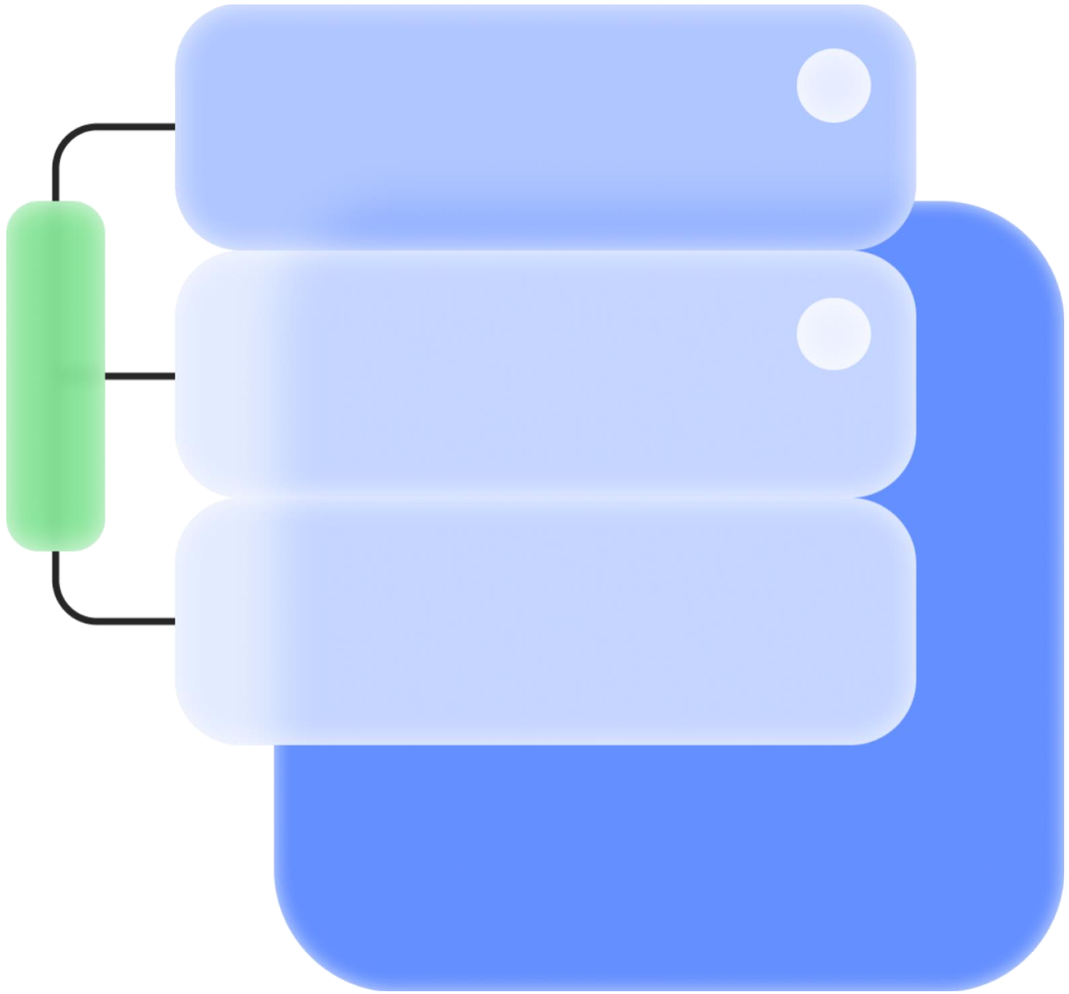
«Алгоритм работы» MATCH_RECOGNIZE


```
SELECT * FROM Input
MATCH_RECOGNIZE(
    PARTITION BY pexpr as P,…
    ORDER BY sort_expr, …
    MEASURES
       A.symbol as a_symbol, …
    ONE ROW PER MATCH
    AFTER MATCH SKIP PAST LAST ROW
    PATTERN (A…)
    DEFINE
       A as TRUE, …
)
```



План:

1. Обзор MATCH_RECOGNIZE
2. Таблицы vs Потоки
3. Масштабируемый пайплайн обработки
4. NFA
5. Дальнейшие планы



Сравнение аналитических и потоковых запросов

Аналитические (на таблицах)

- Сортировка имеющимися средствами (индексы)
- Неограниченные итерации по данным
- Более широкий класс алгоритмов (back_tracking, N(D)FA)
- Trade-off между скоростью обработки и размером стейта
- Стандартные механизмы оптимизации: join, push down предикатов и т.п.

Потоковые

- Невозможность сортировки бесконечного потока
- Ограничение на доступ к данным из прошлого
- Вариации на основе N(D)FA
- Проблема растущего стейта

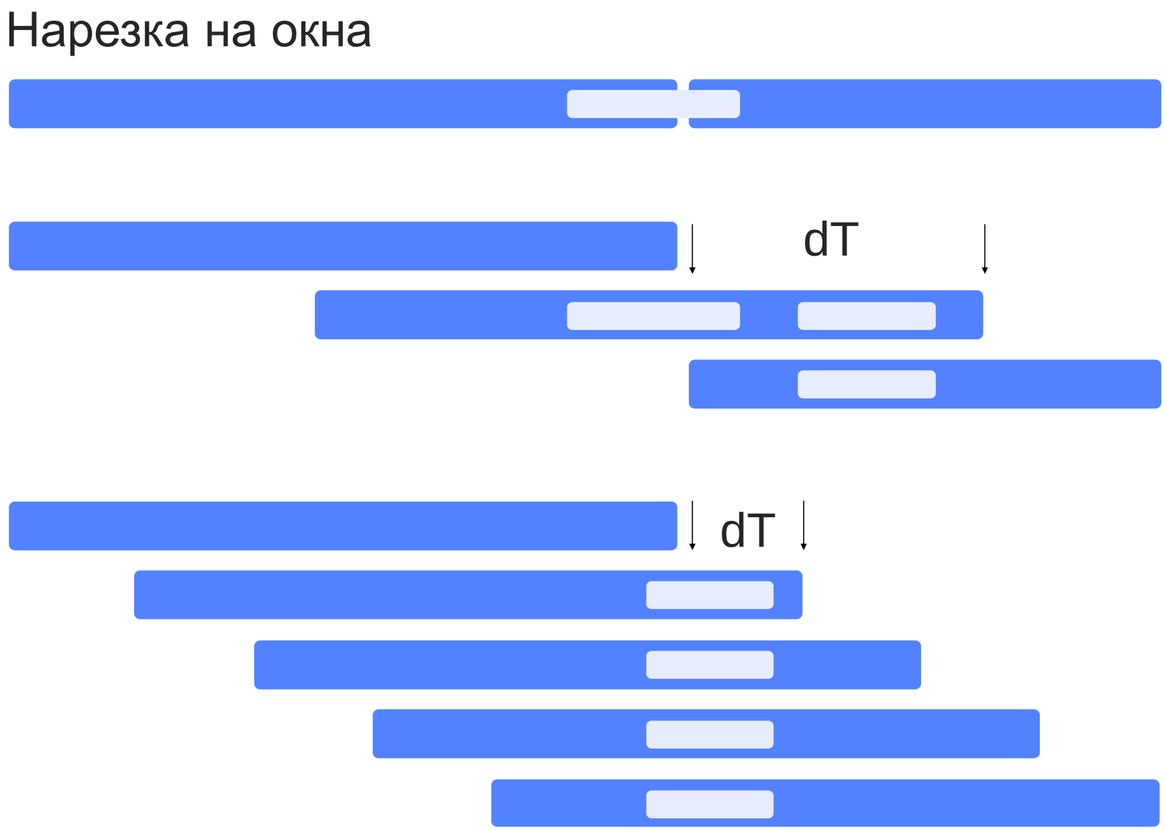
Таблицы через стриминг и стриминг через таблицы

Таблицы через стриминг

- Тривиально (хоть и не оптимально)

Стриминг через таблицы

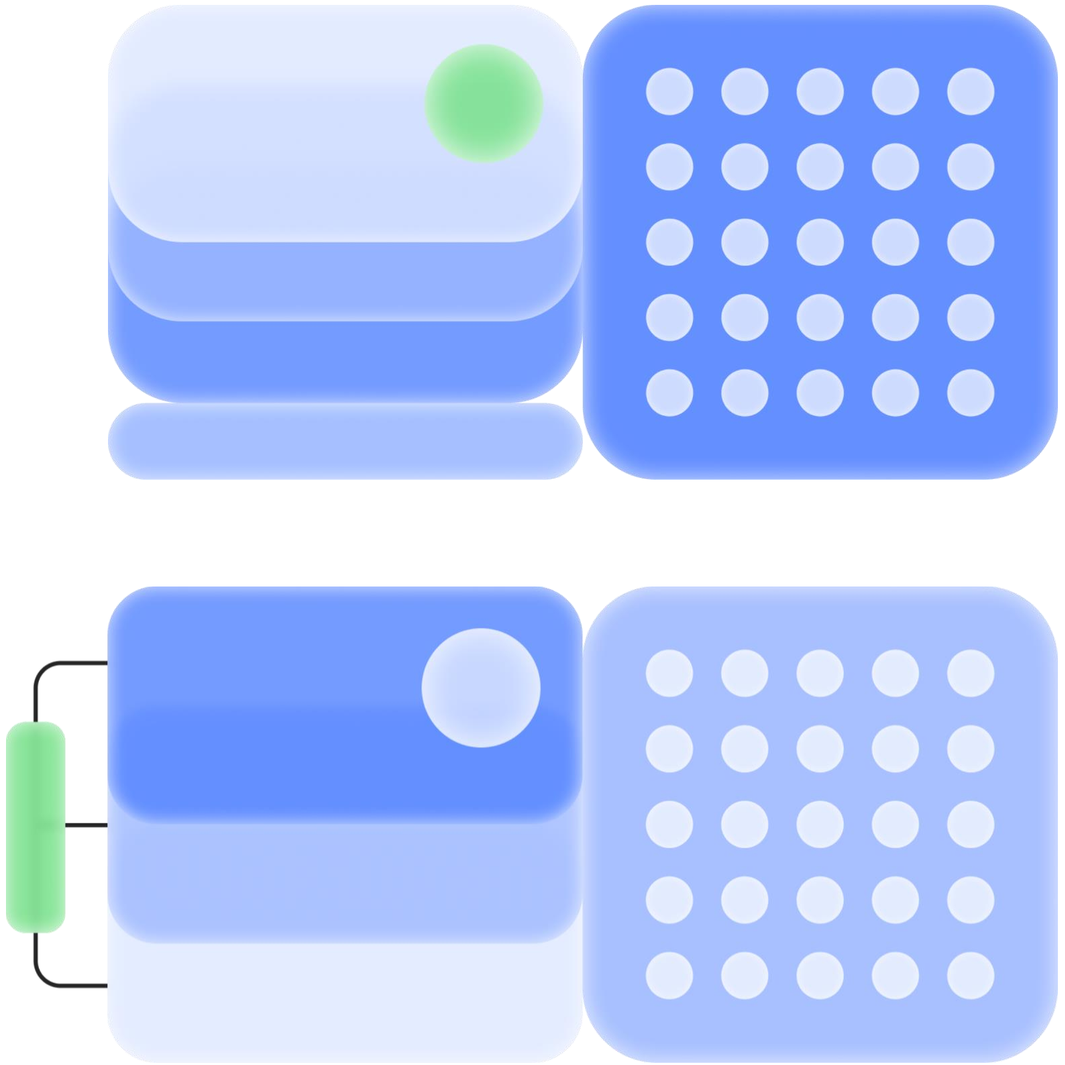
- Кратное умножение ресурсов
- Задержка получения результатов (dT)
- Дублирование результатов



Наш выбор: в первую очередь потоковая обработка

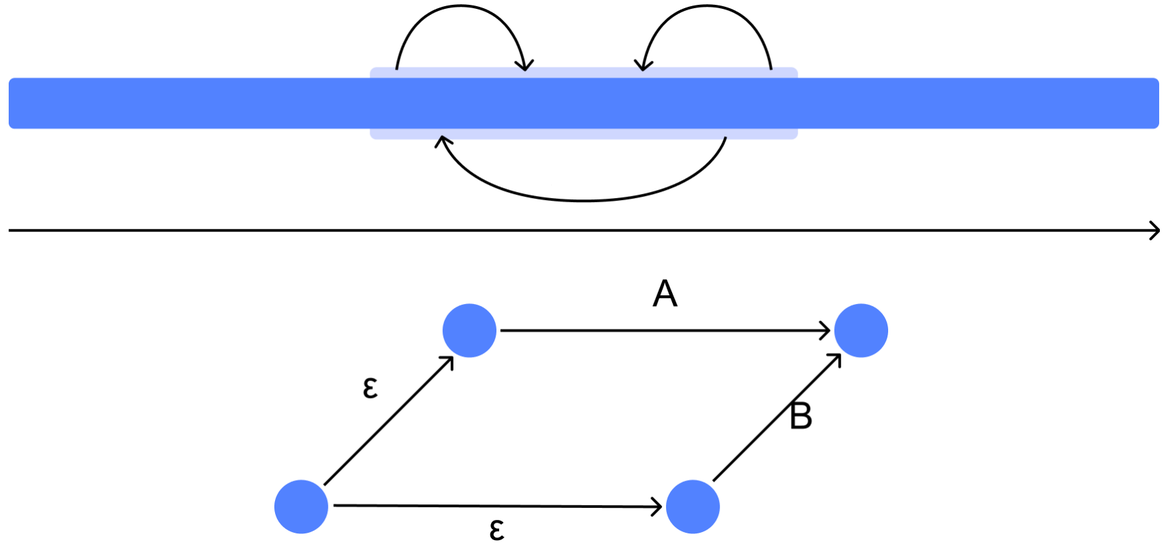
План:

1. Обзор MATCH_RECOGNIZE
2. Таблицы vs Потоки
3. Масштабируемый пайплайн обработки
4. NFA
5. Дальнейшие планы



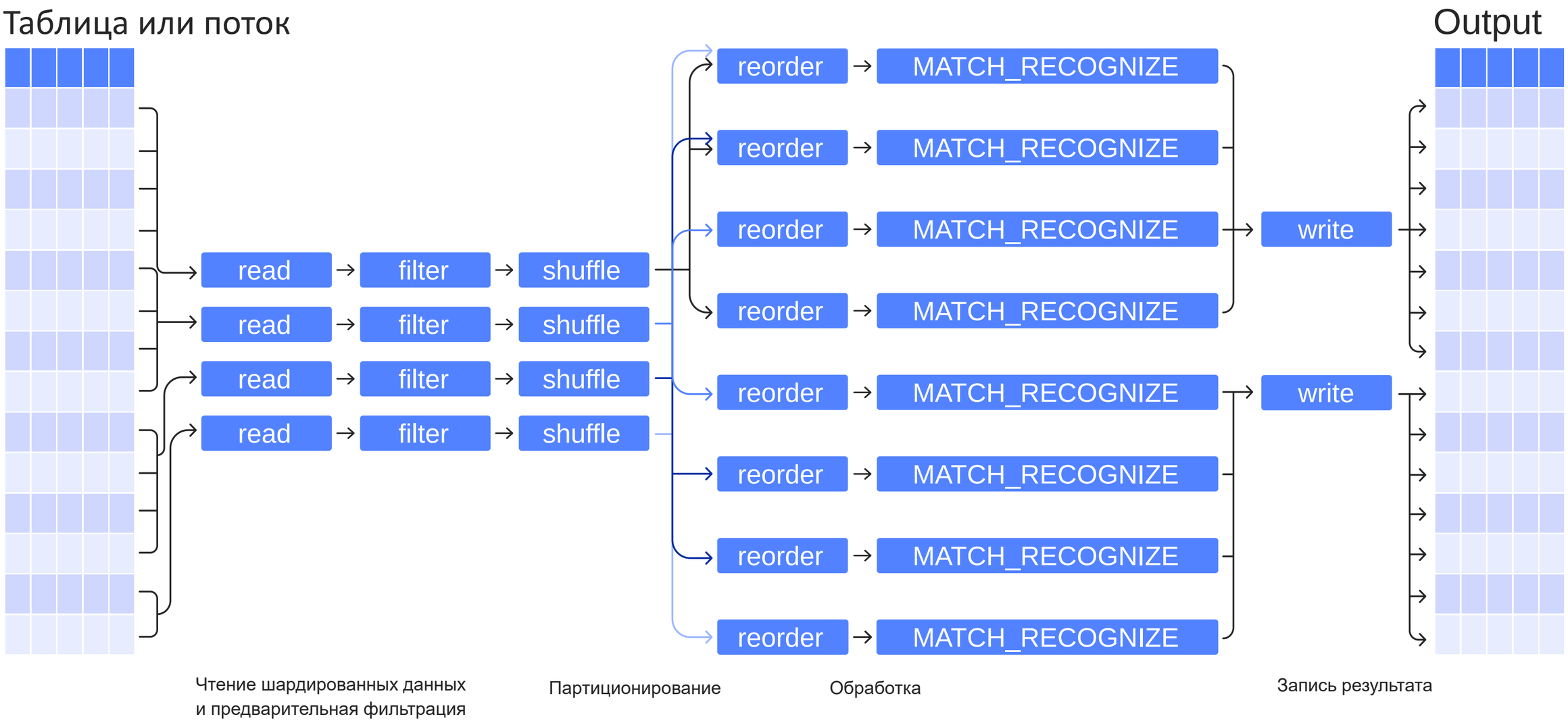
Реализация потоковой обработки

- Сортировку заменили
  на переупорядочение на окне по времени
  с эвристиками управления шириной окна
- Используем алгоритм на основе NFA
- Временные окна в шаблонах выражаем
  через условия над входными данными
- На данный момент – лишь небольшое
  подмножество возможностей из стандарта



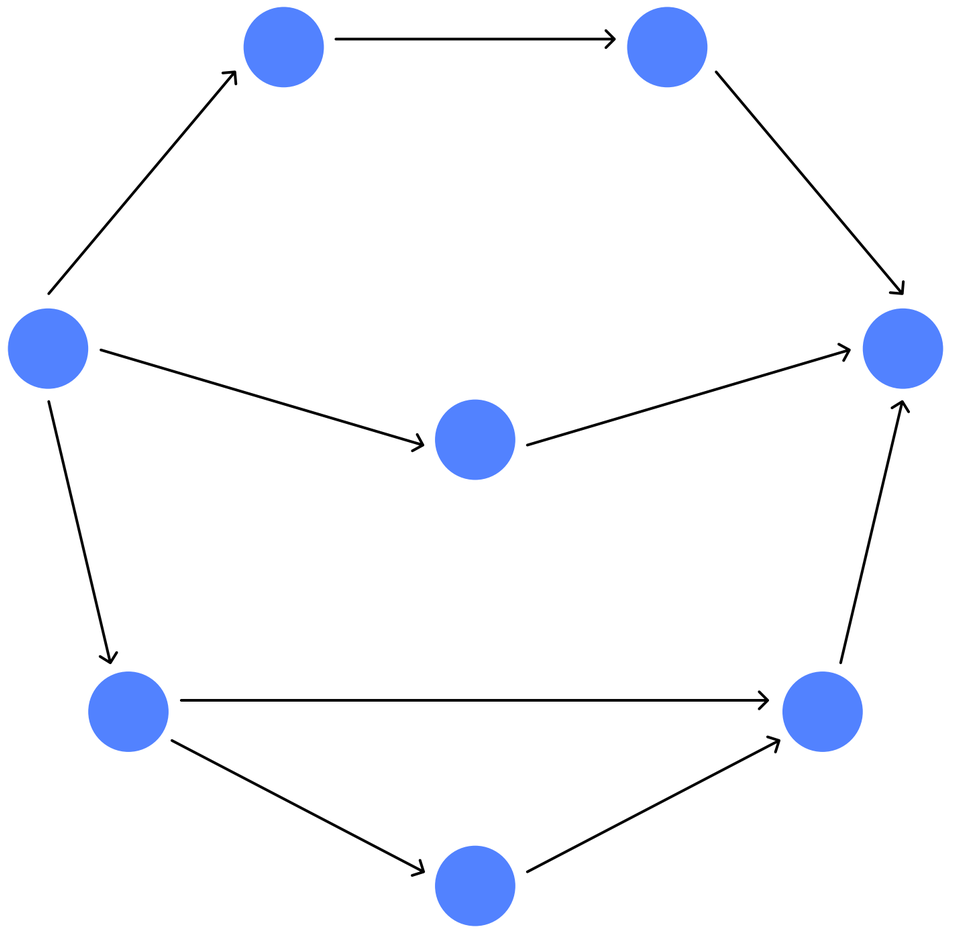

```
and ChangePin.tstamp -
FIRST(LoginFail.tstamp) < `15 min`
```

Параллельная обработка



План:

1. Обзор MATCH_RECOGNIZE
2. Таблицы vs Потоки
3. Масштабируемый пайплайн обработки
4. NFA
5. Дальнейшие планы

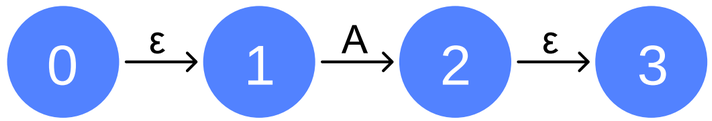


● Статический граф переходов ● Рантайм-стейт

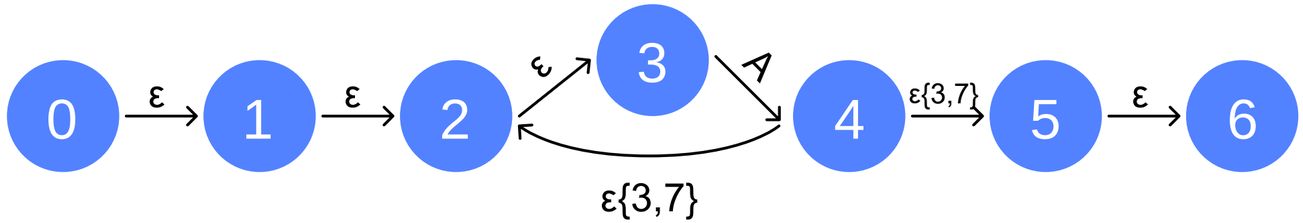
NFA (недетерминированный конечный автомат)

Граф переходов

PATTERN (A)

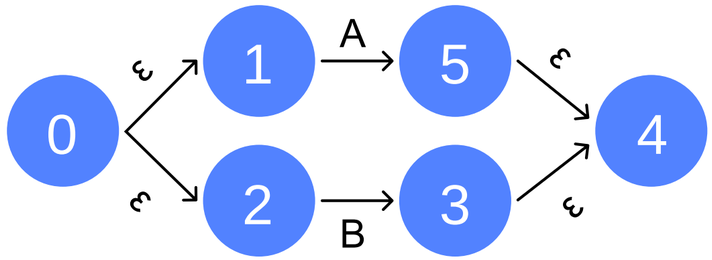


PATTERN (A{3,7})



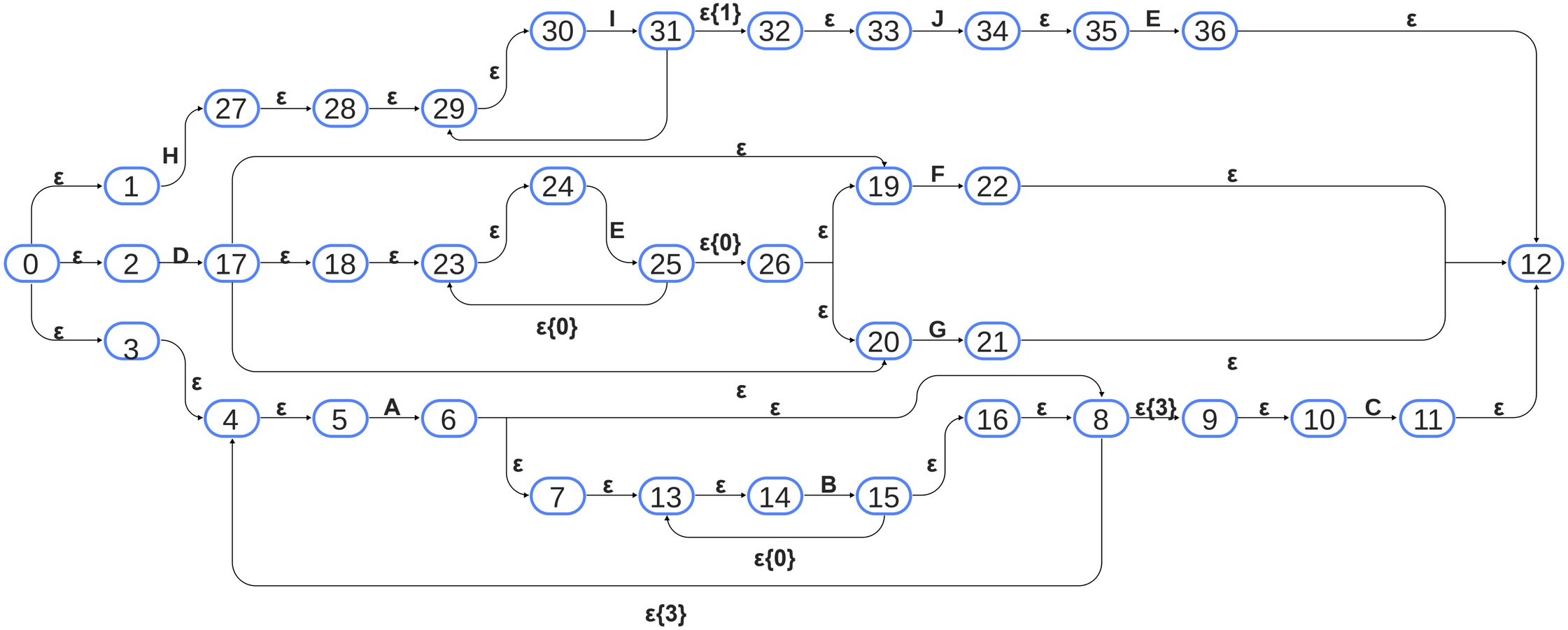
PATTERN (A B)

PATTERN (A|B)



NFA Граф переходов

PATTERN(
(A B*){3,} C | D E* (F | G) | H I+ K E)

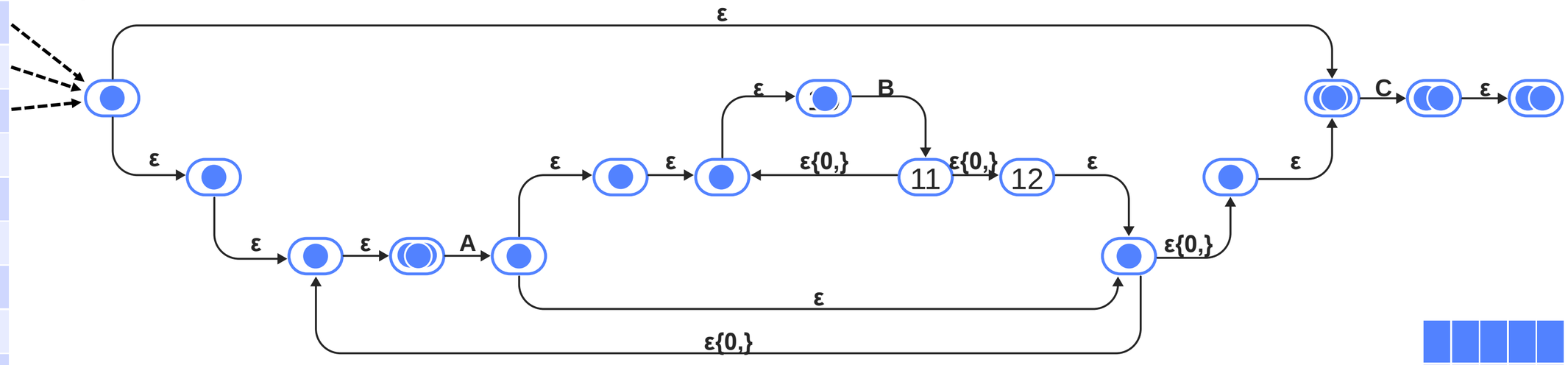


NFA Логика работы

Partition

A
C

PATTERN (
(A B*)*C
)




```
FOR_EACH(row){
    - ε – transitions
    - check predicates
    - ε – transitions
}
```

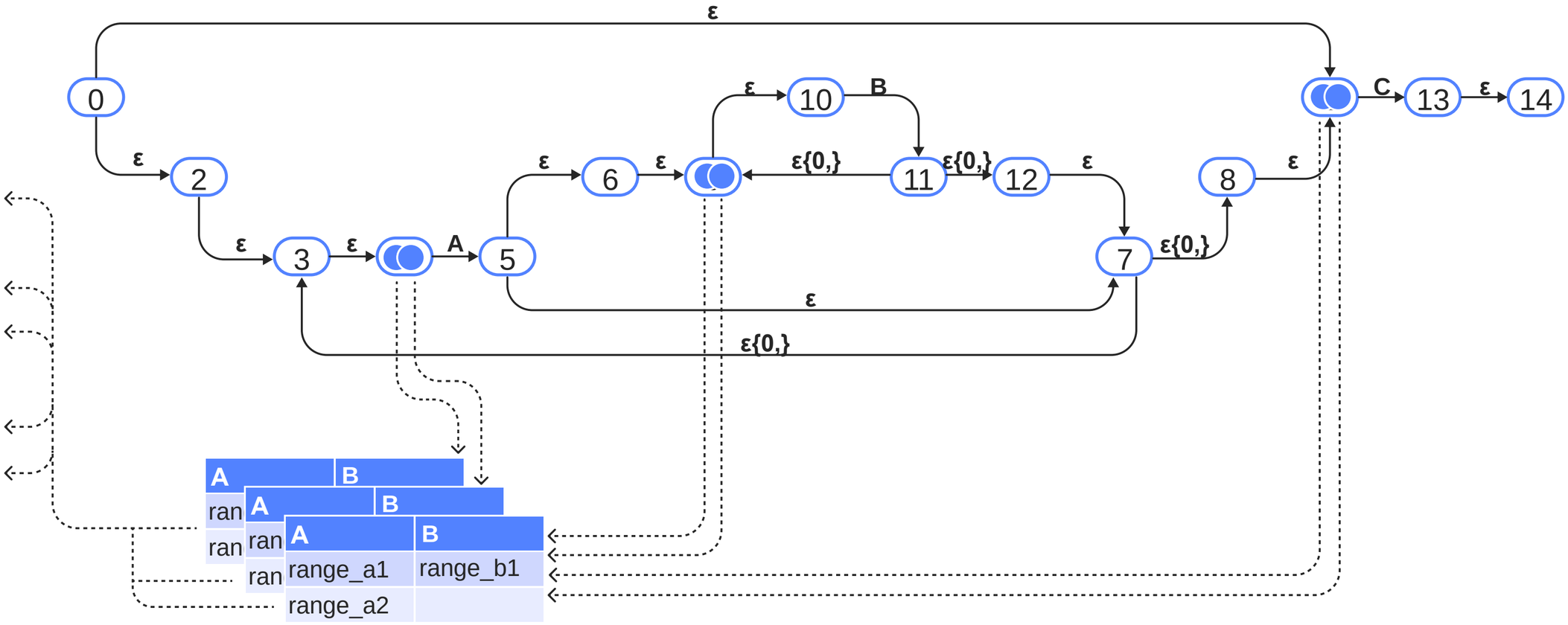
Строка может удовлетворять нескольким предикатам

	A	
A		C

NFA (state)

Partition
(sparse list)

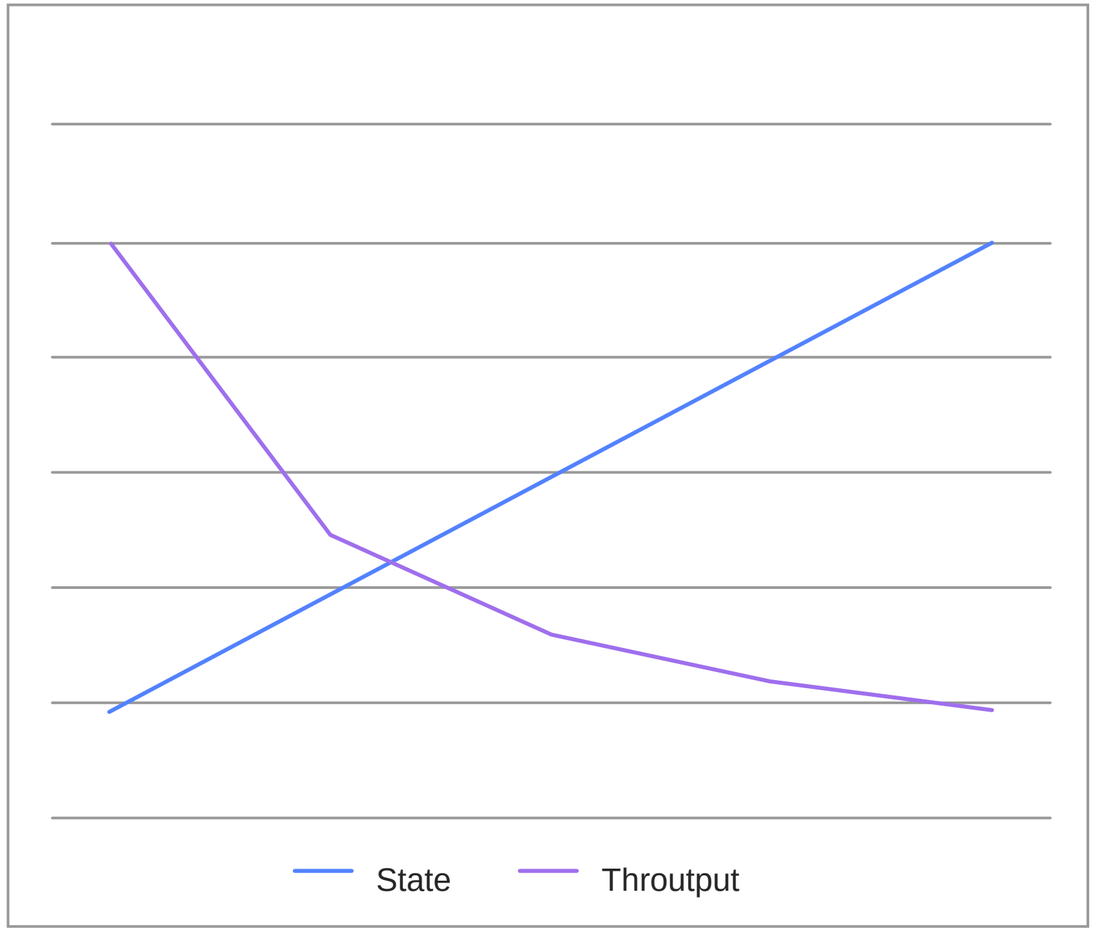
PATTERN (
(A B*)*C
)



Проблемы большого количества частичных матчей

Как уменьшить влияние

- Взаимоисключающие предикаты в паттерне
- Партиционирование
- Уменьшение временного окна



Влияние частично сматчившихся цепочек на производительность

План:

1. Обзор MATCH_RECOGNIZE
2. Таблицы vs Потоки
3. Масштабируемый пайплайн обработки
4. NFA
5. Дальнейшие планы

- Оптимизация обработки частичных матчей
- Расширение функциональности
- Отказ от требования упорядоченности входного потока
- Lazy NFA

Буду рад продолжить общение

Евгений Зверев
Разработчик YDB-платформы
zverevgeny@yandex-team.ru

Оценить доклад