Методы сбора, хранения, обработки и анализа данных

Лекция 6

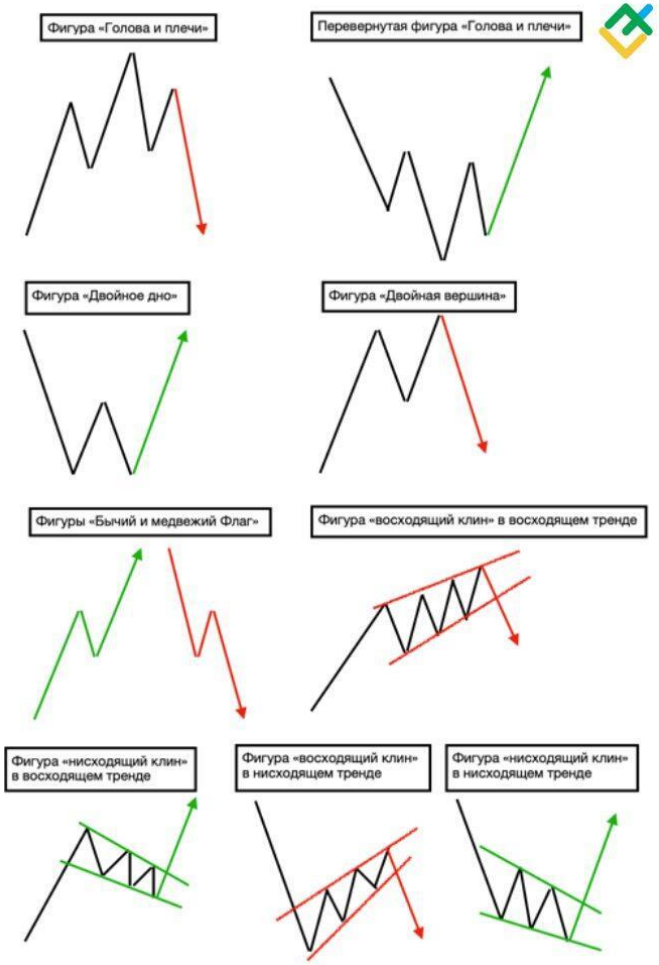
Распознавание закономерностей

Анализ закономерностей

- Закономерности при трейдинге
- Поведение покупателей онлайн
- Анализ безопасности
- Соблюдение требований – комплаенс

Закономерности при трейдинге

- Наличие закономерностей
- Зависимость от даты/времени



Поведение покупателей онлайн

- Навигация, клики, оценки, рекомендации
- Анализ трафика (заказ / отказ от заказа)
- Количество посещенных страниц во время типичной сессии
- Время, которое пользователь тратит на просмотр каждой страницы
- События на страницах (заказ, изменение количества, отмена, избранное)

Анализ безопасности

- Попытки повторной аутентификации
- Увеличение времени сессий с течением времени
- Увеличение количества сессий
- Выполнение значительно большего количества запросов

Комплаенс

- Комплаенс – система внутреннего контроля в организации, которая помогает работать в соответствии с определенными правилами
  - Требования законодательства
  - Регламенты
  - Антифрод
  - Антикоррупционный
  - Антимонопольный
  - Финансовый

match_recognize ()

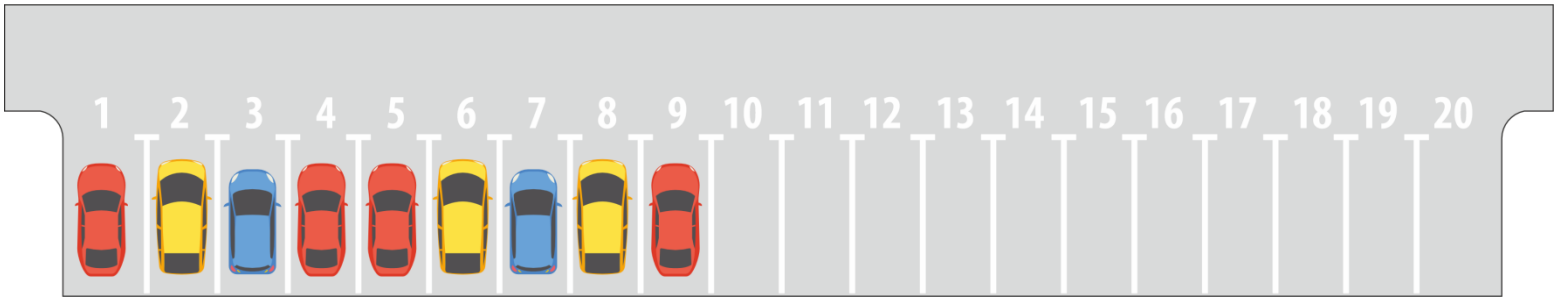
- Позволяет задать параметры закономерности – секция **define**
- Позволяет анализировать заданные закономерности – секция **pattern**
- Поддерживает секционирование – секция **partition**
- Поддерживает сортировку – секция **order by**
- Вводит меры – секция **measures**
- Последовательности могут накладываться друг на друга – секция **after match**
- Фильтрация полученных результатов – секция **rows per match**

Анализ закономерностей – пример 1

```
-- parking example
create table parking_example (
    place number(2),
    car_colour varchar(6));

insert into parking_example values (1, 'red');
insert into parking_example values (2, 'yellow');
insert into parking_example values (3, 'blue');
insert into parking_example values (4, 'red');
insert into parking_example values (5, 'red');
insert into parking_example values (6, 'yellow');
insert into parking_example values (7, 'blue');
insert into parking_example values (8, 'yellow');
insert into parking_example values (9, 'red');
commit;
```

	PLACE	CAR_COLOUR
1	1	red
2	2	yellow
3	3	blue
4	4	red
5	5	red
6	6	yellow
7	7	blue
8	8	yellow
9	9	red

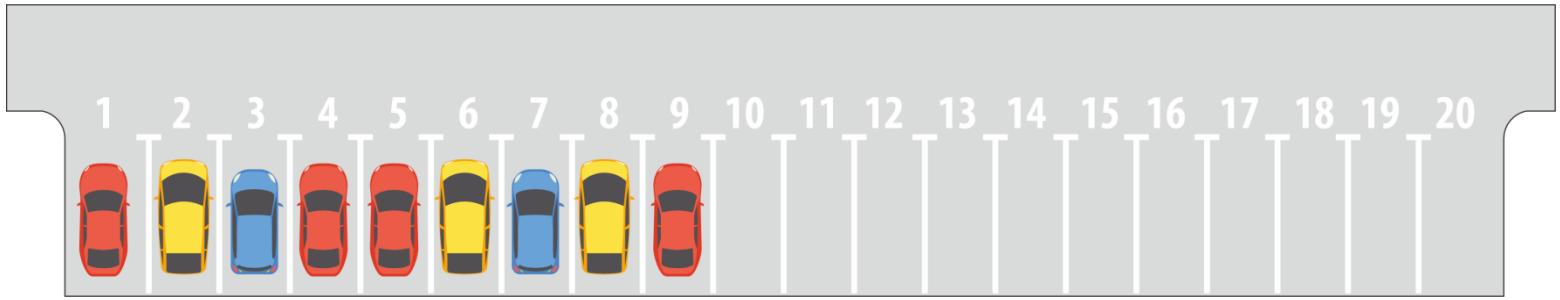


Анализ закономерностей – пример 1

- Найти позиции на которых последовательно стоят красная, желтая и синяя машины (1, 2, 3) и (5, 6, 7)

```
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               YELLOW.place AS yellow_place,
               BLUE.place AS blue_place
     ONE ROW PER MATCH
     PATTERN (RED YELLOW BLUE) -- конкатенация шаблона
     DEFINE
        RED AS RED.car_colour = 'red',
        YELLOW AS YELLOW.car_colour = 'yellow',
        BLUE AS BLUE.car_colour = 'blue'
     );
```

	RED_PLACE	YELLOW_PLACE	BLUE_PLACE
1	1	2	3
2	5	6	7

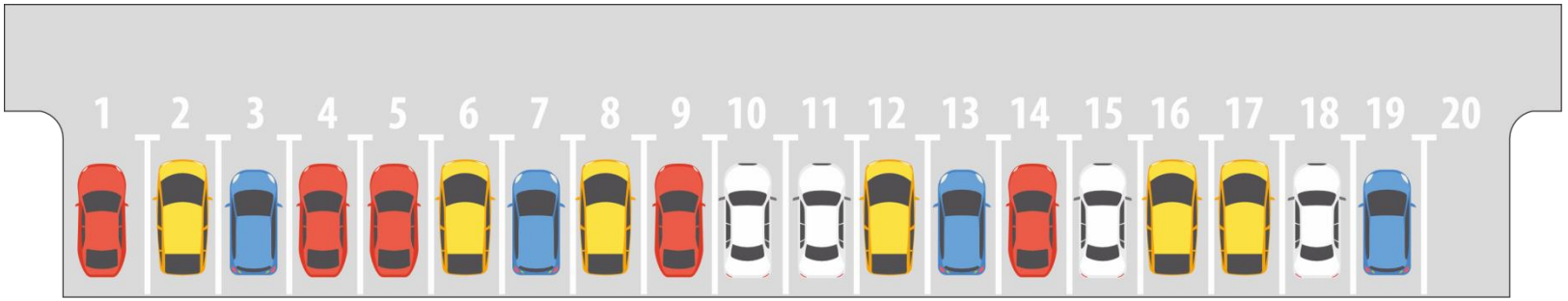


Анализ закономерностей – пример 1

```
-- add some more cars
insert into parking_example values (10, 'white');
insert into parking_example values (11, 'white');
insert into parking_example values (12, 'yellow');
insert into parking_example values (13, 'blue');
insert into parking_example values (14, 'red');
insert into parking_example values (15, 'white');
insert into parking_example values (16, 'yellow');
insert into parking_example values (17, 'yellow');
insert into parking_example values (18, 'white');
insert into parking_example values (19, 'blue');
commit;
```

	PLACE	CAR_COLOUR
1	1	red
2	2	yellow
3	3	blue
4	4	red
5	5	red
6	6	yellow
7	7	blue
8	8	yellow
9	9	red
10	10	white
11	11	white
12	12	yellow
13	13	blue
14	14	red
15	15	white
16	16	yellow
17	17	yellow
18	18	white
19	19	blue

	RED_PLACE	YELLOW_PLACE	BLUE_PLACE
1	1	2	3
2	5	6	7

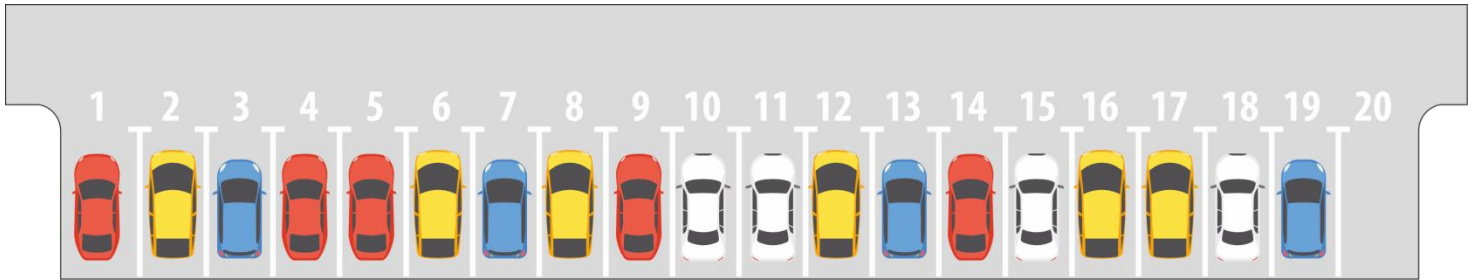


Анализ закономерностей – пример 1

- Найти позиции на которых последовательно стоят красная, желтая и синяя машины, а между ними могут находиться и белые машины

```
-- white pattern
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               YELLOW.place AS yellow_place,
               BLUE.place AS blue_place
     ONE ROW PER MATCH
     PATTERN (RED WHITE* YELLOW WHITE* BLUE)
     DEFINE
        RED AS RED.car_colour = 'red',
        YELLOW AS YELLOW.car_colour = 'yellow',
        BLUE AS BLUE.car_colour = 'blue',
        WHITE AS WHITE.car_colour = 'white'
     );
```

	RED_PLACE	YELLOW_PLACE	BLUE_PLACE
1	1	2	3
2	5	6	7
3	9	12	13

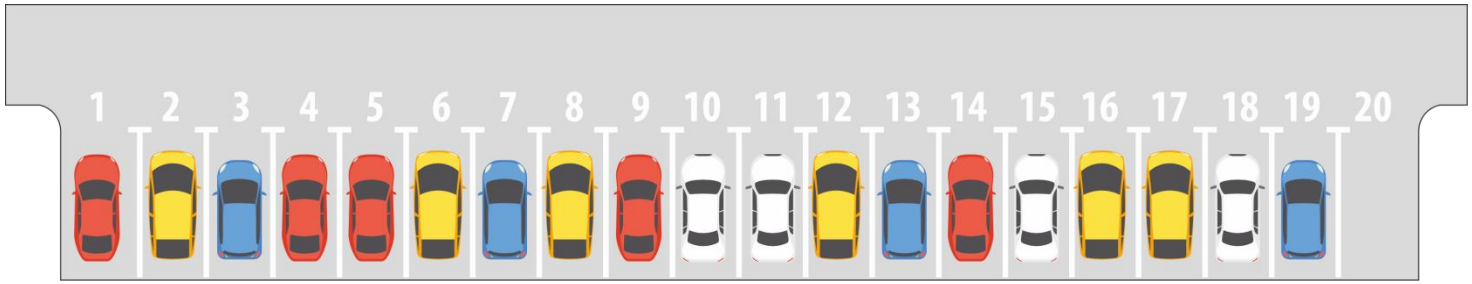


Анализ закономерностей – пример 1

- Найти позиции на которых последовательно стоят несколько красных, желтых и синих машин, а между ними могут находиться и белые машины

```
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               YELLOW.place AS yellow_place,
               BLUE.place AS blue_place
     ONE ROW PER MATCH
     PATTERN (RED+ WHITE* YELLOW+ WHITE* BLUE+)
     DEFINE
        RED AS RED.car_colour = 'red',
        YELLOW AS YELLOW.car_colour = 'yellow',
        BLUE AS BLUE.car_colour = 'blue',
        WHITE AS WHITE.car_colour = 'white'
     );
```

	RED_PLACE	YELLOW_PLACE	BLUE_PLACE
1	1	2	3
2	5	6	7
3	9	12	13
4	14	17	19



match_recognize () – PATTERN

- PATTERN:
  - Конкатенация
  - Квантификаторы
  - Альтернатива
  - Группировка

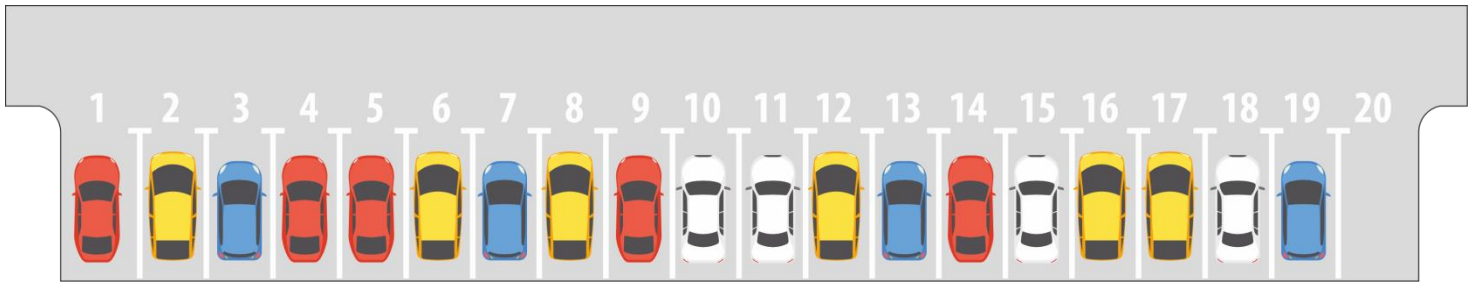
match_recognize () – PATTERN

- Можно задавать следующие квантификаторы:
  - * — 0 или более
  - + — 1 или более
  - ? — 0 или 1
  - {n} — n (n > 0)
  - {n,} — n или более (n >= 0)
  - {n,m} — от n до m (включительно)
  - {,m} — от 0 до m (включительно)

match_recognize () – PATTERN

```
-- 1-2 белых машины между красной и желтой
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               WHITE.place AS white_place,
               YELLOW.place AS yellow_place
     ONE ROW PER MATCH
     PATTERN (RED WHITE{1,2} YELLOW)
     DEFINE
        RED AS RED.car_colour = 'red',
        YELLOW AS YELLOW.car_colour = 'yellow',
        WHITE AS WHITE.car_colour = 'white'
     );
```

	RED_PLACE	WHITE_PLACE	YELLOW_PLACE
1	9	11	12
2	14	15	16

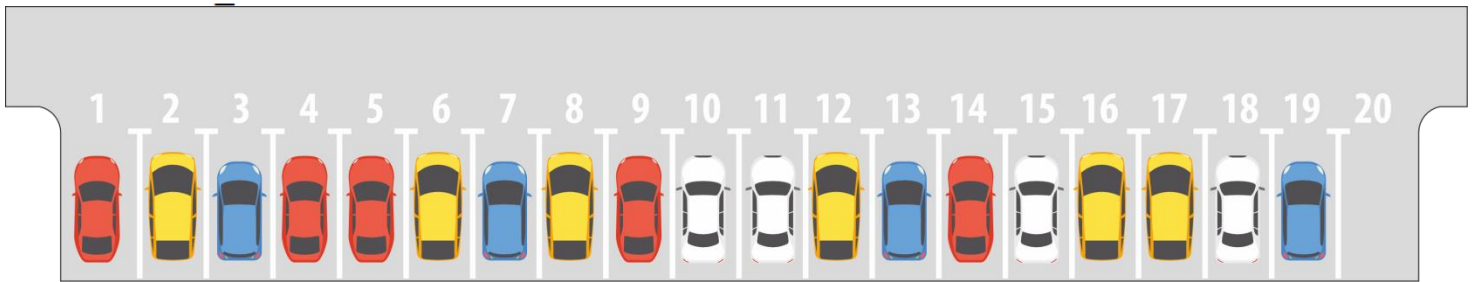


match_recognize () – PATTERN

- Можно задавать альтернативные значения и группировать условия скобками:

```
-- Первой стоит либо красная либо белая машина,
-- а за ней -  желтая и голубая
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               WHITE.place AS white_place,
               YELLOW.place AS yellow_place,
               BLUE.place AS blue_place
     ONE ROW PER MATCH
     PATTERN (( RED|WHITE ) YELLOW  BLUE)
     DEFINE
        RED AS RED.car_colour = 'red',
        YELLOW AS YELLOW.car_colour = 'yellow',
        BLUE AS BLUE.car_colour = 'blue',
        WHITE AS WHITE.car_colour = 'white'
     );
```

	RED_...	WHITE_PLACE	YELLOW_PLACE	BLUE_PLACE
1	1	(null)	2	3
2	5	(null)	6	7
3	(null)	11	12	13

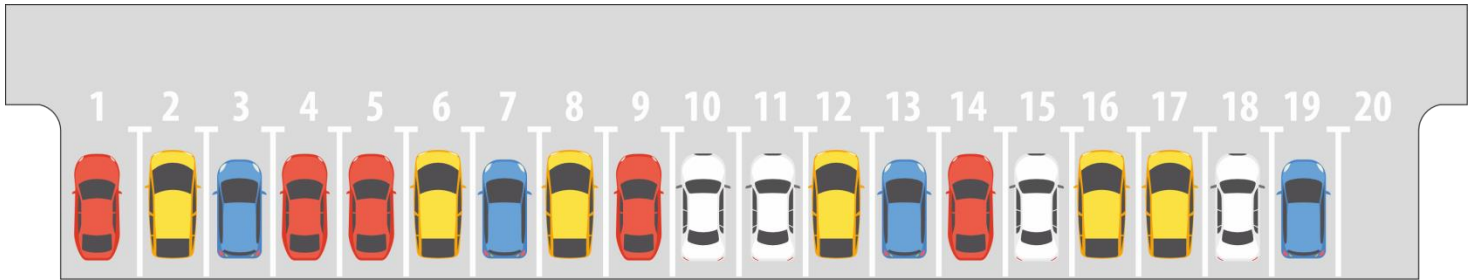


match_recognize () – PATTERN

- Перестановка условий:

```
-- Белая и желтая машины в любом порядке
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  WHITE.place AS white_place,
               YELLOW.place AS yellow_place
     ONE ROW PER MATCH
     PATTERN (PERMUTE(WHITE,YELLOW))
     DEFINE
        YELLOW AS YELLOW.car_colour = 'yellow',
        WHITE AS WHITE.car_colour = 'white'
     );
```

	WHITE_PLACE	YELLOW_PLACE
1	11	12
2	15	16
3	18	17

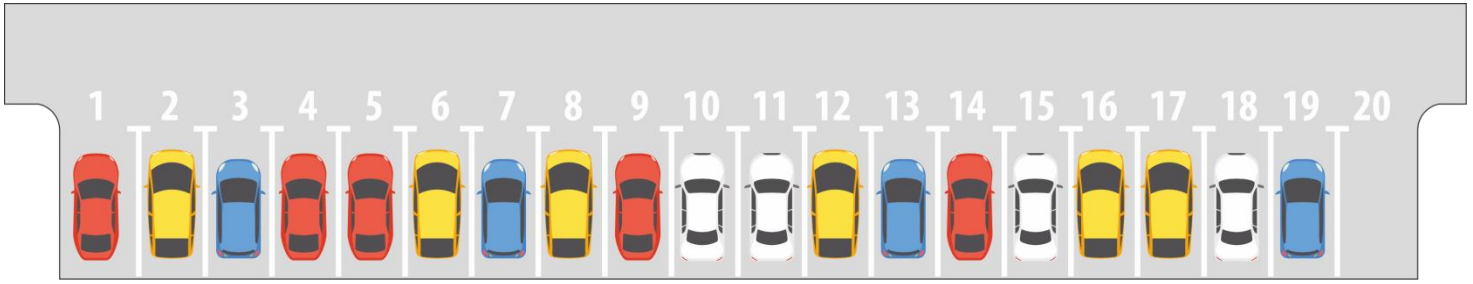


match_recognize () – PARTITION

- Поддерживается секционирование данных:

```
-- пара машин одинакового цвета
SELECT * from parking_example
MATCH_RECOGNIZE (
     PARTITION BY car_colour
     ORDER BY place
     MEASURES  PREV(T.place) as start_pair_place,
               T.place AS end_pair_place
     ONE ROW PER MATCH
     PATTERN (T)
     DEFINE  T AS place = PREV(place) + 1
     );
```

	CAR_COLOUR	START_PAIR_PLACE	END_PAIR_PLACE
1	red	4	5
2	white	10	11
3	yellow	16	17



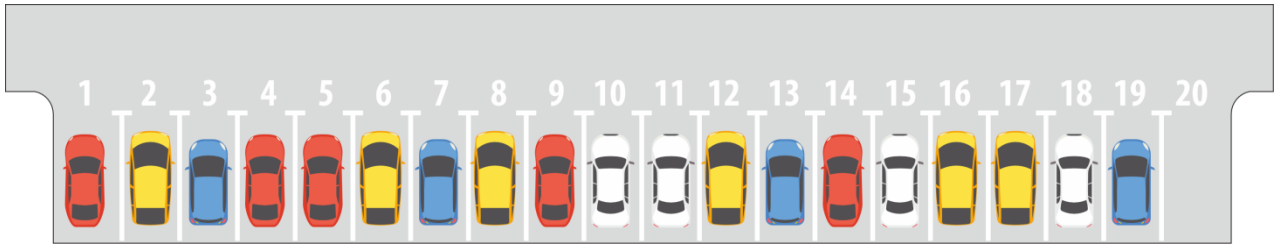
match_recognize () – PER MATCH

- Вывод одной строки для каждого найденного образца ONE ROW PER MATCH
- Вывод всех задействованных строк для каждого найденного образца ALL ROWS PER MATCH

```
-- ALL ROWS PER MATCH
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               YELLOW.place AS yellow_place,
               BLUE.place AS blue_place
     ALL ROWS PER MATCH
     PATTERN (RED YELLOW BLUE)
     DEFINE
        RED AS RED.car_colour = 'red',
        YELLOW AS YELLOW.car_colour = 'yellow',
        BLUE AS BLUE.car_colour = 'blue');
```

	PLACE	RED_PLACE	YELLOW_PLACE	BLUE_PLACE	CAR_COLOUR
1	1	1	(null)	(null)	red
2	2	1	2	(null)	yellow
3	3	1	2	3	blue
4	5	5	(null)	(null)	red
5	6	5	6	(null)	yellow
6	7	5	6	7	blue

	RED_PLACE	YELLOW_PLACE	BLUE_PLACE
1	1	2	3
2	5	6	7



match_recognize () – AFTER MATCH

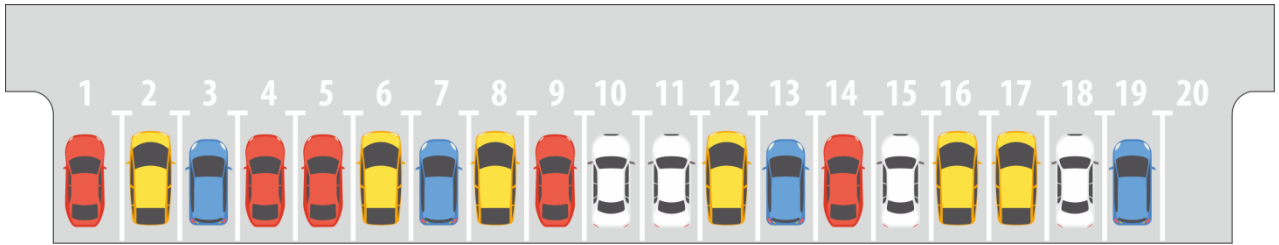
- По умолчанию – поиск начинается со следующей строки, перекрытия не допускаются PAST LAST ROW
- SKIP TO NEXT ROW – возможны перекрытия

```
-- AFTER MATCH SKIP PAST LAST ROW
-- AFTER MATCH SKIP TO NEXT ROW
-- AFTER MATCH SKIP TO [ FIRST | LAST ] pattern_variable
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               YELLOW.place AS yellow_place,
               BLUE.place AS blue_place
     ONE ROW PER MATCH
     AFTER MATCH SKIP PAST LAST ROW
     PATTERN (PERMUTE(RED, YELLOW, BLUE))
     DEFINE
        RED AS RED.car_colour = 'red',
        YELLOW AS YELLOW.car_colour = 'yellow',
        BLUE AS BLUE.car_colour = 'blue');
```

	RED_PLACE	YELLOW_PLACE	BLUE_PLACE
1	1	2	3
2	5	6	7
3	14	12	13

```
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               YELLOW.place AS yellow_place,
               BLUE.place AS blue_place
     ONE ROW PER MATCH
     AFTER MATCH SKIP TO NEXT ROW
     PATTERN (PERMUTE(RED, YELLOW, BLUE))
     DEFINE
        RED AS RED.car_colour = 'red',
        YELLOW AS YELLOW.car_colour = 'yellow',
        BLUE AS BLUE.car_colour = 'blue');
```

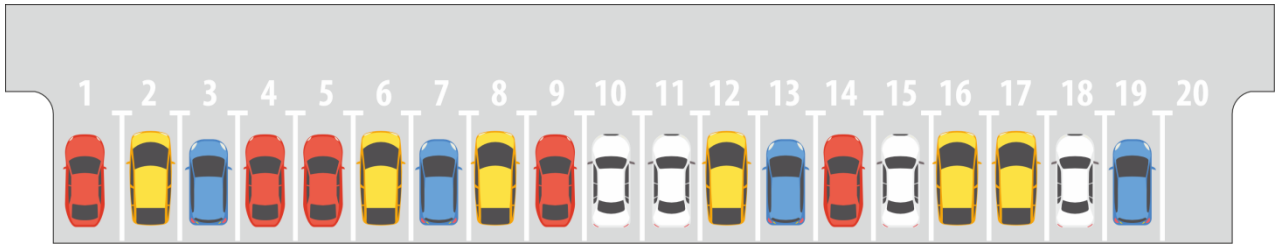
	RED_PLACE	YELLOW_PLACE	BLUE_PLACE
1	1	2	3
2	4	2	3
3	5	6	7
4	9	8	7
5	14	12	13



match_recognize () – classifier()

```
-- classifier()
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               BLUE.place AS blue_place,
               match_number() AS nm,
               classifier() AS car_type
     ONE ROW PER MATCH
     PATTERN (RED | BLUE)
     DEFINE
        RED AS RED.car_colour = 'red',
        BLUE AS BLUE.car_colour = 'blue');
```

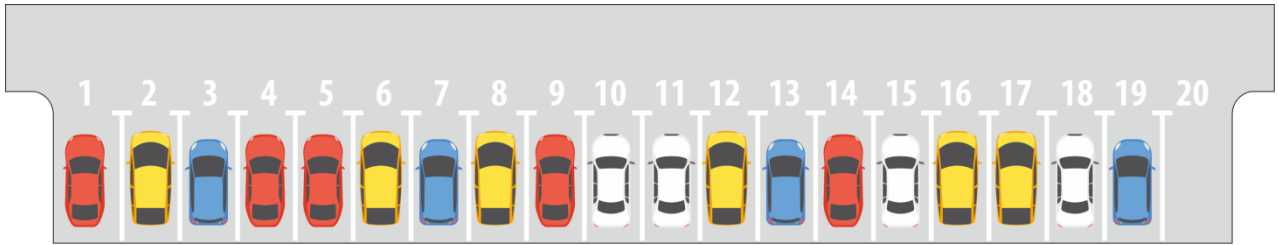
	RED_PLACE	BLUE_PLACE	NM	CAR_TYPE
1	1	(null)	1	RED
2	(null)	3	2	BLUE
3	4	(null)	3	RED
4	5	(null)	4	RED
5	(null)	7	5	BLUE
6	9	(null)	6	RED
7	(null)	13	7	BLUE
8	14	(null)	8	RED
9	(null)	19	9	BLUE



match_recognize () – match_number()

```
-- match_number()
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               YELLOW.place AS yellow_place,
               BLUE.place AS blue_place,
               match_number() AS nm
     ONE ROW PER MATCH
     PATTERN (RED YELLOW BLUE)
     DEFINE
        RED AS RED.car_colour = 'red',
        YELLOW AS YELLOW.car_colour = 'yellow',
        BLUE AS BLUE.car_colour = 'blue');
```

	RED_PLACE	YELLOW_PLACE	BLUE_PLACE	NM
1	1	2	3	1
2	5	6	7	2

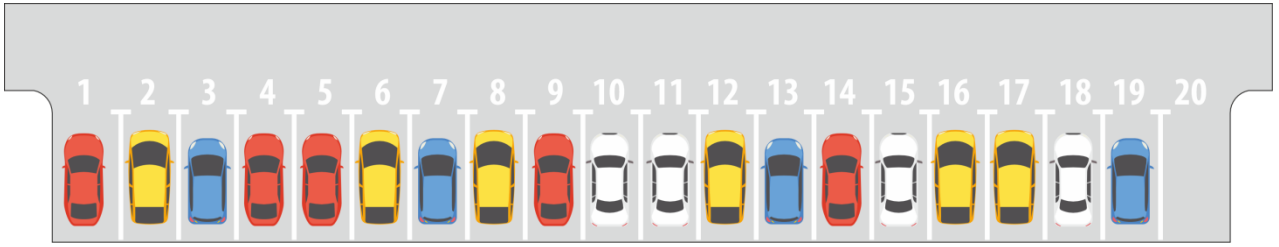


match_recognize () – навигация

- Логическая навигация: функции FIRST и LAST
- Физическая навигация: функции PREV и NEXT

```
-- PREV и NEXT
-- пара машин одинакового цвета
SELECT * from parking_example
MATCH_RECOGNIZE (
     PARTITION BY car_colour
     ORDER BY place
     MEASURES  PREV(T.place) as start_pair_place,
               T.place AS end_pair_place,
               NEXT(T.place) as next_point
     ONE ROW PER MATCH
     PATTERN (T)
     DEFINE  T AS place = PREV(place) + 1
     );
```

	CAR_COLOUR	START_PAIR_PLACE	END_PAIR_PLACE	NEXT_POINT
1	red	4	5	9
2	white	10	11	15
3	yellow	16	17	(null)

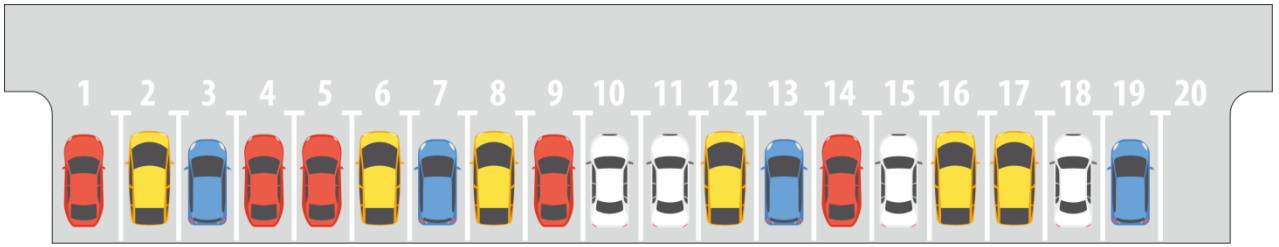


match_recognize () – навигация

- Логическая навигация: функции FIRST и LAST
- Физическая навигация: функции PREV и NEXT

```
-- FIRST и LAST
-- две красные подряд
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               FIRST(RED.place) AS first_red_place,
               LAST(RED.place) AS last_red_place
     ONE ROW PER MATCH
     PATTERN (RED RED)
     DEFINE
        RED AS RED.car_colour = 'red');
```

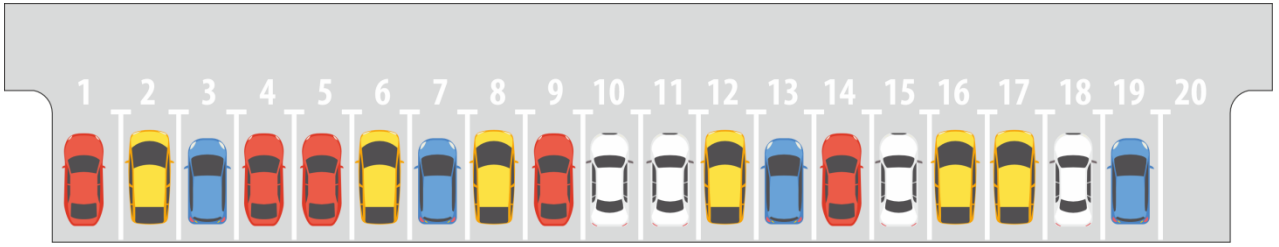
	RED_PLACE	FIRST_RED_PLACE	LAST_RED_PLACE
1	5	4	5



match_recognize () – исключения

```
--красная, а за ней любая, только не красная
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  RED.place AS red_place,
               OTHER_COLOUR.place AS other_colour_car_place,
               OTHER_COLOUR.car_colour AS other_car_colour
     ONE ROW PER MATCH
     PATTERN (RED OTHER_COLOUR)
     DEFINE
        RED AS RED.car_colour = 'red',
        OTHER_COLOUR AS car_colour != 'red' );
```

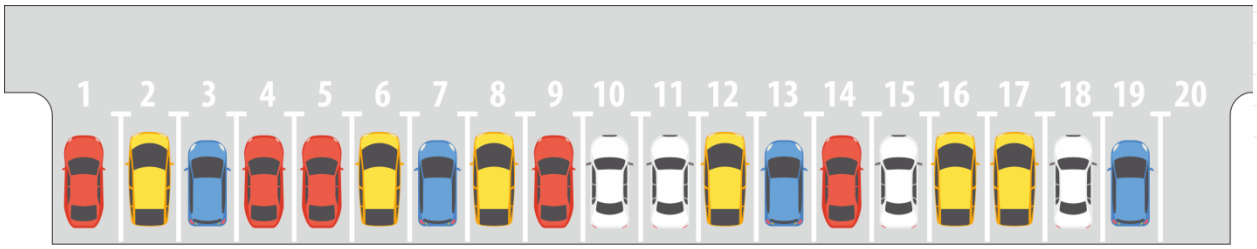
	RED_PLACE	OTHER_COLOUR_CAR_PLACE	OTHER_CAR_COLOUR
1	1	2	yellow
2	5	6	yellow
3	9	10	white
4	14	15	white



match_recognize () – агрегаты

```
-- Подсчитать количество машин других цветов между машинами одного цвета
SELECT * from parking_example
MATCH_RECOGNIZE (
     ORDER BY place
     MEASURES  COLOUR.car_colour AS car_colour,
               COLOUR.place AS car_place,
               COUNT(DIF_COLOUR.*) AS cars_between,
               SAME_COLOUR.place AS next_car_place
     ONE ROW PER MATCH
     AFTER MATCH SKIP TO FIRST DIF_COLOUR
     PATTERN (COLOUR DIF_COLOUR+ SAME_COLOUR+)
     DEFINE
        DIF_COLOUR AS DIF_COLOUR.car_colour != COLOUR.car_colour,
        SAME_COLOUR AS SAME_COLOUR.car_colour = COLOUR.car_colour);
```

	CAR_COLOUR	CAR_PLACE	CARS_BETWEEN	NEXT_CAR_PLACE
1	red	1	2	5
2	yellow	2	3	6
3	blue	3	3	7
4	red	5	3	9
5	yellow	6	1	8
6	blue	7	5	13
7	yellow	8	3	12
8	red	9	4	14
9	white	11	3	15
10	yellow	12	3	17
11	blue	13	5	19
12	white	15	2	18



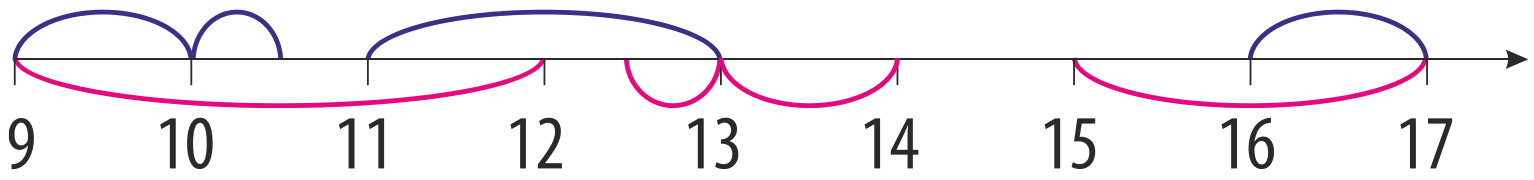
match_recognize () – итоги

- **define**
- **pattern**
- **partition**
- **order by**
- **measures**
- **after match**
- **rows per match**
- **prev / next**
- **first / last**

Анализ закономерностей – пример 2

```
CREATE TABLE meeting_attendees (
  attendee_id INTEGER NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE,
  PRIMARY KEY ( attendee_id, start_date ));
```

ATTENDEE_ID	START_DATE	END_DATE
1	17/06/20 09:00	17/06/20 10:00
2	17/06/20 09:00	17/06/20 12:00
1	17/06/20 10:00	17/06/20 10:30
1	17/06/20 11:00	17/06/20 13:00
2	17/06/20 12:30	17/06/20 13:00
2	17/06/20 13:00	17/06/20 14:00
2	17/06/20 15:00	17/06/20 17:00
1	17/06/20 16:00	17/06/20 17:00



Анализ закономерностей – пример 2

```
-- свободное время по каждому
SELECT *
FROM   meeting_attendees
MATCH_RECOGNIZE (
    PARTITION BY attendee_id
    ORDER BY start_date
    MEASURES
      end_date start_ft,
      NEXT(start_date) end_ft,
      classifier() AS cls
    ONE ROW PER MATCH
    PATTERN (FREE_TIME)
    DEFINE
    FREE_TIME AS end_date < NEXT(start_date));
```

	ATTENDEE_ID	START_FT	END_FT	CLS
1	1	17/06/20 10:30	17/06/20 11:00	FREE TIME
2	1	17/06/20 13:00	17/06/20 16:00	FREE TIME
3	2	17/06/20 12:00	17/06/20 12:30	FREE TIME
4	2	17/06/20 14:00	17/06/20 15:00	FREE TIME

Анализ закономерностей – пример 2

```
-- все периоды - свободные и занятые
SELECT *
FROM   meeting_attendees
MATCH_RECOGNIZE (
    ORDER BY start_date, end_date
    MEASURES
      MAX(end_date) start_free,
      NEXT(start_date) end_free,
      classifier() AS cls
    ALL ROWS PER MATCH
    PATTERN ( (FREE|BUSY)+ )
    DEFINE
    FREE AS MAX(end_date) < NEXT(start_date),
    BUSY AS 1 = 1);
```

	START_DATE	END_DATE	START_FREE	END_FREE	CLS	ATTENDEE_ID
1	17/06/20 09:00	17/06/20 10:00	17/06/20 10:00	17/06/20 09:00	BUSY	1
2	17/06/20 09:00	17/06/20 12:00	17/06/20 12:00	17/06/20 10:00	BUSY	2
3	17/06/20 10:00	17/06/20 10:30	17/06/20 12:00	17/06/20 11:00	BUSY	1
4	17/06/20 11:00	17/06/20 13:00	17/06/20 13:00	17/06/20 12:30	BUSY	1
5	17/06/20 12:30	17/06/20 13:00	17/06/20 13:00	17/06/20 13:00	BUSY	2
6	17/06/20 13:00	17/06/20 14:00	17/06/20 14:00	17/06/20 15:00	FREE	2
7	17/06/20 15:00	17/06/20 17:00	17/06/20 17:00	17/06/20 16:00	BUSY	2
8	17/06/20 16:00	17/06/20 17:00	17/06/20 17:00	(null)	BUSY	1

Анализ закономерностей – пример 2

```
-- только свободные
SELECT *
FROM   meeting_attendees
MATCH_RECOGNIZE (
    ORDER BY start_date, end_date
    MEASURES
      MAX(end_date) start_free,
      NEXT(start_date) end_free
    ALL ROWS PER MATCH
    PATTERN ( (FREE|{-BUSY-})+ )
    DEFINE
    FREE AS MAX(end_date) < NEXT(start_date));
```

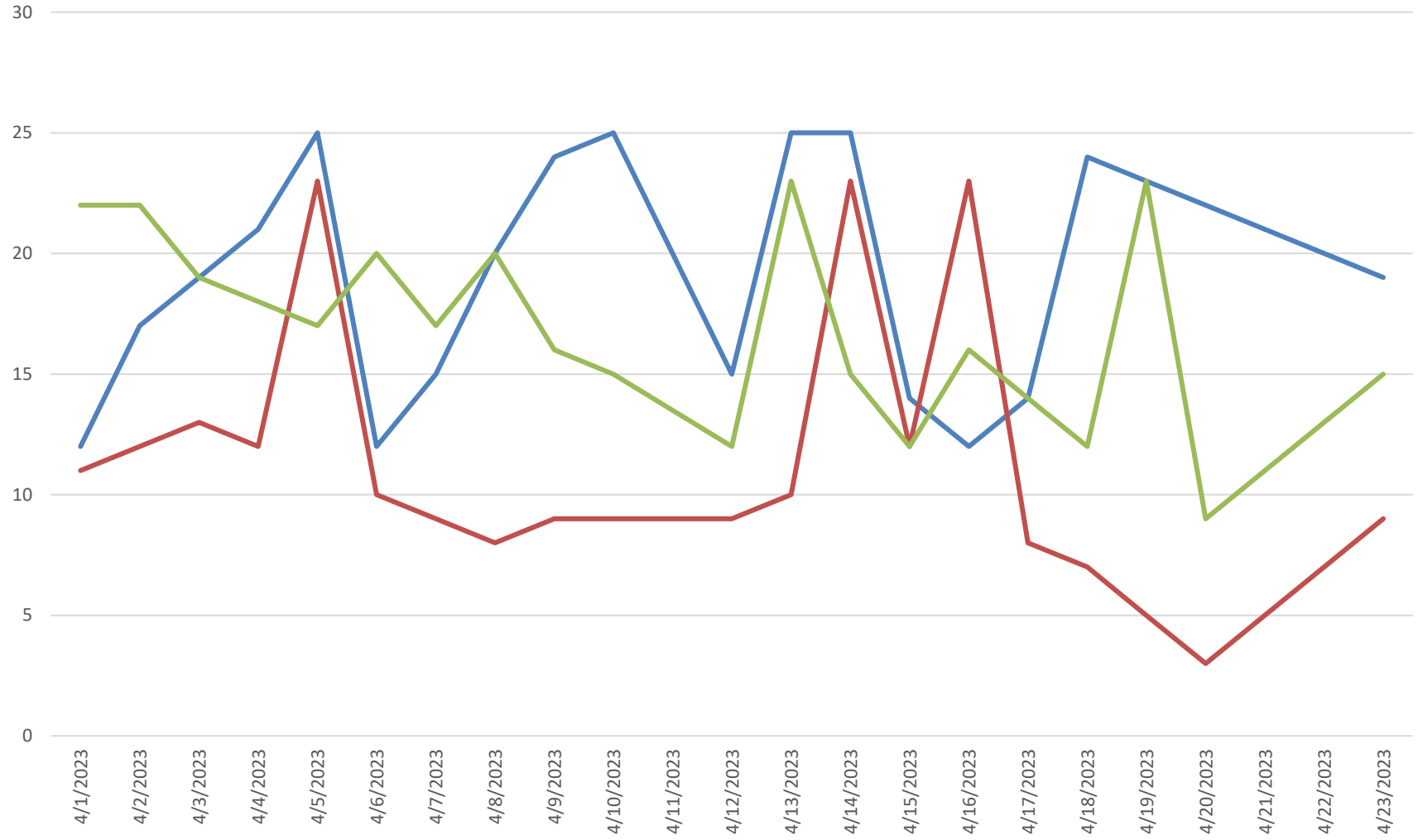
	START_DATE	END_DATE	START_FREE	END_FREE	ATTENDEE_ID
1	17/06/20 13:00	17/06/20 14:00	17/06/20 14:00	17/06/20 15:00	2

Анализ закономерностей – пример 3

```
CREATE TABLE ticker (
        SYMBOL VARCHAR2(10),
        tstamp DATE,
        price NUMBER);
```

	SYMBOL	TSTAMP	PRICE
1	ACME	01-04-23	12
2	ACME	02-04-23	17
3	ACME	03-04-23	19
4	ACME	04-04-23	21
5	ACME	05-04-23	25
6	ACME	06-04-23	12
7	ACME	07-04-23	15
8	ACME	08-04-23	20
9	ACME	09-04-23	24
10	ACME	10-04-23	25
11	ACME	12-04-23	15
12	ACME	13-04-23	25
13	ACME	14-04-23	25
14	ACME	15-04-23	14
15	ACME	16-04-23	12
16	ACME	17-04-23	14
17	ACME	18-04-23	24
18	ACME	19-04-23	23
19	ACME	20-04-23	22
20	ACME	23-04-23	19
21	GLOBEX	01-04-23	11
22	GLOBEX	02-04-23	12
23	GLOBEX	03-04-23	13
24	GLOBEX	04-04-23	12
25	GLOBEX	05-04-23	23
26	GLOBEX	06-04-23	10
27	GLOBEX	07-04-23	9
28	GLOBEX	08-04-23	8
29	GLOBEX	09-04-23	9
30	GLOBEX	10-04-23	9
31	GLOBEX	12-04-23	9
32	GLOBEX	13-04-23	10
33	GLOBEX	14-04-23	23
34	GLOBEX	15-04-23	12
35	GLOBEX	16-04-23	23
36	GLOBEX	17-04-23	8
37	GLOBEX	18-04-23	7
38	GLOBEX	19-04-23	5
39	GLOBEX	20-04-23	3
40	GLOBEX	23-04-23	9
41	OSCORP	01-04-23	22
42	OSCORP	02-04-23	22
43	OSCORP	03-04-23	19
44	OSCORP	04-04-23	18
45	OSCORP	05-04-23	17
46	OSCORP	06-04-23	20
47	OSCORP	07-04-23	17
48	OSCORP	08-04-23	20
49	OSCORP	09-04-23	16
50	OSCORP	10-04-23	15
51	OSCORP	12-04-23	12
52	OSCORP	13-04-23	23
53	OSCORP	14-04-23	15
54	OSCORP	15-04-23	12
55	OSCORP	16-04-23	16
56	OSCORP	17-04-23	14
57	OSCORP	18-04-23	12
58	OSCORP	19-04-23	23
59	OSCORP	20-04-23	9
60	OSCORP	23-04-23	15

Анализ закономерностей – пример 3



Анализ закономерностей – пример 3

```
-- угол вниз
SELECT *
FROM Ticker MATCH_RECOGNIZE (
     PARTITION BY symbol
     ORDER BY tstamp
     MEASURES  STRT.tstamp AS start_tstamp,
               LAST(DOWN.tstamp) AS bottom_tstamp,
               LAST(UP.tstamp) AS end_tstamp
     ONE ROW PER MATCH
     AFTER MATCH SKIP TO LAST UP
     PATTERN (STRT DOWN+ UP+)
     DEFINE
        DOWN AS DOWN.price < PREV(DOWN.price),
        UP AS UP.price > PREV(UP.price)
     ) MR
ORDER BY MR.symbol, MR.start_tstamp;
```

	SYMBOL	START_TSTAMP	BOTTOM_TSTAMP	END_TSTAMP
1	ACME	05-04-23	06-04-23	10-04-23
2	ACME	10-04-23	12-04-23	13-04-23
3	ACME	14-04-23	16-04-23	18-04-23
4	GLOBEX	03-04-23	04-04-23	05-04-23
5	GLOBEX	05-04-23	08-04-23	09-04-23
6	GLOBEX	14-04-23	15-04-23	16-04-23
7	GLOBEX	16-04-23	20-04-23	23-04-23
8	OSCORP	02-04-23	05-04-23	06-04-23
9	OSCORP	06-04-23	07-04-23	08-04-23
10	OSCORP	08-04-23	12-04-23	13-04-23
11	OSCORP	13-04-23	15-04-23	16-04-23
12	OSCORP	16-04-23	18-04-23	19-04-23
13	OSCORP	19-04-23	20-04-23	23-04-23

Анализ закономерностей – пример 3

```
-- угол вверх
SELECT *
FROM Ticker MATCH_RECOGNIZE (
     PARTITION BY symbol
     ORDER BY tstamp
     MEASURES  STRT.tstamp AS start_tstamp,
               LAST(UP.tstamp) AS peak_tstamp,
               LAST(DOWN.tstamp) AS end_tstamp
     ONE ROW PER MATCH
     AFTER MATCH SKIP TO LAST UP
     PATTERN (STRT UP+ DOWN+)
     DEFINE
        DOWN AS DOWN.price < PREV(DOWN.price),
        UP AS UP.price > PREV(UP.price)
     ) MR
ORDER BY MR.symbol, MR.start_tstamp;
```

	SYMB...	START_TSTAMP	PEAK_TSTAMP	END_TSTAMP
1	ACME	01-04-23	05-04-23	06-04-23
2	ACME	06-04-23	10-04-23	12-04-23
3	ACME	16-04-23	18-04-23	23-04-23
4	GLOBEX	01-04-23	03-04-23	04-04-23
5	GLOBEX	04-04-23	05-04-23	08-04-23
6	GLOBEX	12-04-23	14-04-23	15-04-23
7	GLOBEX	15-04-23	16-04-23	20-04-23
8	OSCORP	05-04-23	06-04-23	07-04-23
9	OSCORP	07-04-23	08-04-23	12-04-23
10	OSCORP	12-04-23	13-04-23	15-04-23
11	OSCORP	15-04-23	16-04-23	18-04-23
12	OSCORP	18-04-23	19-04-23	20-04-23

MATCH_RECOGNIZE – проверочная работа:

- Используется таблица TICKER и слайд 3
- Добавить данные, если недостаточно имеющихся
- 1 группа: найти образец «Голова и плечи»
- 2 группа: найти образец «Двойное дно»
- 3 группа: найти образец «Бычий флаг»
- 7 группа: найти образец «Двойная вершина»
- 8 группа: найти образец «Медвежий флаг»
- + предложить решение для любого вида клинов

- Срок – до следующей лекции

Вопросы?