# Oracle SQL – Row Pattern Matching

Stephan Karrer

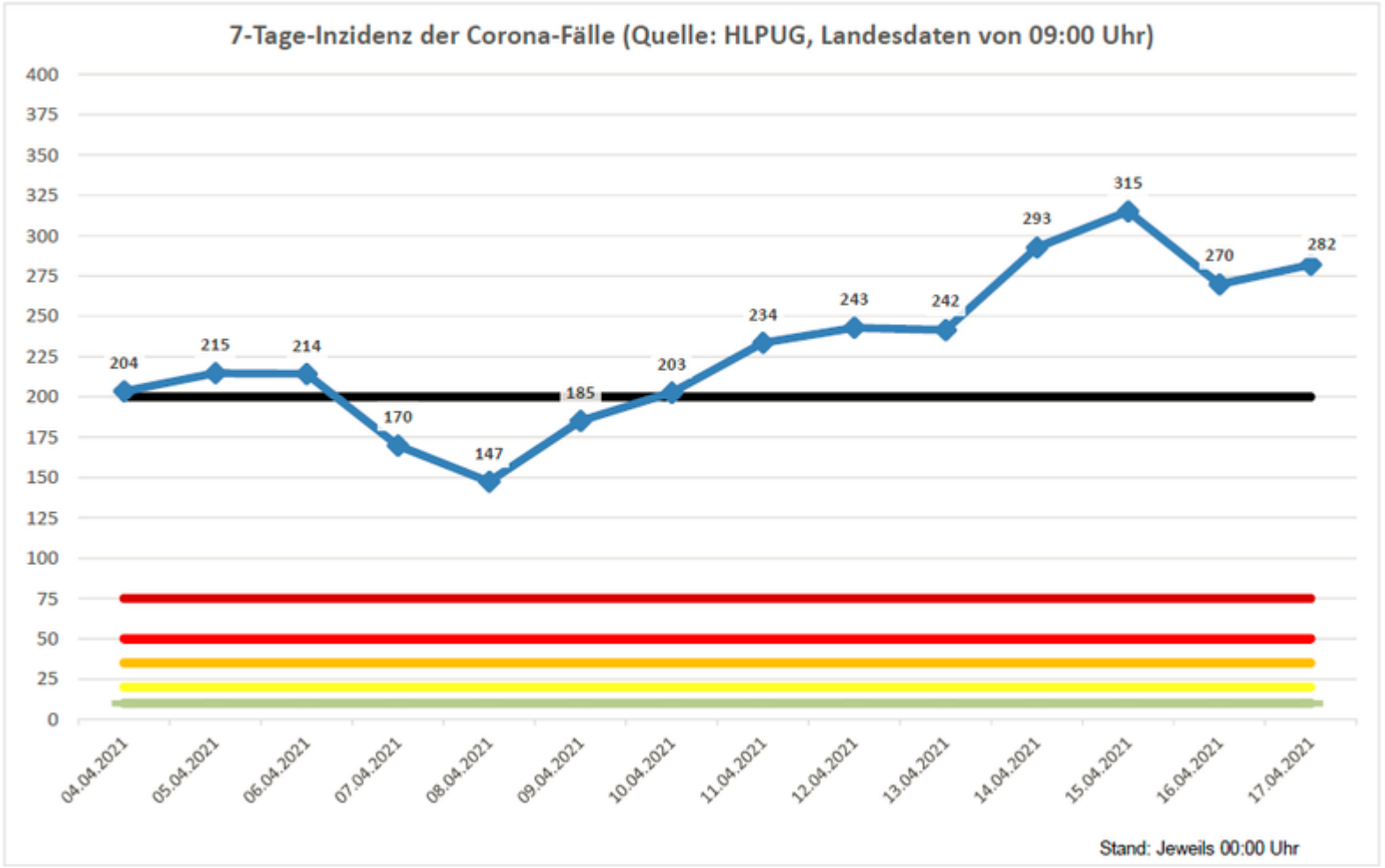
# Worum geht es

- Wir suchen nach Mustern, denen gewisse Spaltenwerte folgen, in einer Sequenz von Zeilen.
- Für jede durch das Muster getroffene Sequenz von Zeilen wollen wir natürlich bestimmte Werte bzw. Aggregate (Measures) ausgeben.
- Dazu definiert der ANSI 2016 Standard 3 optionale Features:
  - R010 ist die Basis: MATCH_RECOGNIZE_Klausel mit den Aggregatfunktionen min, max, sum, count, avg.
  - R020 erlaubt die Verwendung von Patterns zur Definition von Frames in der over-Klausel.
  - R030 Unterstützt der restlichen Aggregatfunktionen (z. B. stddev_pop, …).
- Dadurch können die bisherigen Lösungsansätze mittels geschachtelter Abfragen unter Nutzung von Aggregaten und analytischen Funktionen ersetzt werden.
  - Potentieller Performanzgewinn

# Unterstützung durch DBMS-Hersteller

- Momentane Unterstützung seitens DBMS-Hersteller (nach bestem Gewissen, Stand April 2021):
  - Oracle 12c: R010
  - Microsoft Azure SQL Server): R010 als Preview-Feature
  - Sonst herrscht Flaute.
  - R020 und R030 werden momentan gar nicht umgesetzt.

# Ein erstes Beispiel:
# Coronafallzahlen für den Landkreis Fulda




- Als Muster soll ein V-Shape dienen: fallende Zahlen, gefolgt von steigenden Zahlen.

# Eine (nicht aktuelle) Tabelle der Fallzahlen

- V-Shapes:
  - 14.3. - 18.3.
  - 18.3. - 23.3.
  - 23.3. - 25.3.
  - …
- Die Ausgangsdaten mussen natürlich sortiert sein, hier anhand des Datums, um stabile Mustererkennung zu garantieren.

| | BUNDESLAND | LANDKREIS | MELDEDATUM | FALLZAHL |
|---|---|---|---|---|
| 1 | Hessen | LK Fulda | 09.03.20 | 2 |
| 2 | Hessen | LK Fulda | 11.03.20 | 5 |
| 3 | Hessen | LK Fulda | 12.03.20 | 2 |
| 4 | Hessen | LK Fulda | 13.03.20 | 2 |
| 5 | Hessen | LK Fulda | 14.03.20 | 14 |
| 6 | Hessen | LK Fulda | 16.03.20 | 9 |
| 7 | Hessen | LK Fulda | 17.03.20 | 3 |
| 8 | Hessen | LK Fulda | 18.03.20 | 11 |
| 9 | Hessen | LK Fulda | 19.03.20 | 9 |
| 10 | Hessen | LK Fulda | 20.03.20 | 6 |
| 11 | Hessen | LK Fulda | 23.03.20 | 24 |
| 12 | Hessen | LK Fulda | 24.03.20 | 5 |
| 13 | Hessen | LK Fulda | 25.03.20 | 9 |
| 14 | Hessen | LK Fulda | 26.03.20 | 4 |
| 15 | Hessen | LK Fulda | 27.03.20 | 10 |
| 16 | Hessen | LK Fulda | 28.03.20 | 12 |
| 17 | Hessen | LK Fulda | 29.03.20 | 4 |
| 18 | Hessen | LK Fulda | 31.03.20 | 15 |
| 19 | Hessen | LK Fulda | 01.04.20 | 19 |

# Syntax für die MATCH_RECOGNIZE-Klausel

```
SELECT <select list> FROM <source table>
  MATCH_RECOGNIZE (
    [ PARTITION BY <partition list> ]
    [ ORDER BY <order by list> ]
    [ MEASURES <measure list> ]
    [ <row pattern rows per match> ::= ONE ROW PER MATCH |
                                       ALL ROWS PER MATCH ]
    [ AFTER MATCH <skip to option>
    PATTERN ( <row pattern> )
    [ SUBSET <subset list> ]
    DEFINE <definition list>
  ) <table alias>;
```

# Beispiel mit ONE ROW PER MATCH

```
SELECT * FROM CORONAZAHLENFULDA MATCH_RECOGNIZE (
    ORDER BY MELDEDATUM
    MEASURES STRT.meldedatum AS start_day,
             LAST(DOWN.meldedatum) AS bottom_day,
             LAST(UP.meldedatum) AS high_day
    ONE ROW PER MATCH
    AFTER MATCH SKIP TO LAST UP
    PATTERN (STRT DOWN+ UP+)
    DEFINE DOWN AS DOWN.fallzahl < PREV(DOWN.fallzahl),
           UP AS UP.fallzahl > PREV(UP.fallzahl)
);
```

Sortierung

Werte

Eine Zeile je Treffer

Wo geht's weiter

Muster: regulärer Ausdruck auf Basis von Pattern -Variablen

# Beispiel mit ONE ROW PER MATCH: Ergebnis

- V-Shapes:
  - 14.3. - 18.3.
  - 18.3. - 23.3.
  - 23.3. - 25.3.
  - …

| | START_DAY | BOTTOM_DAY | HIGH_DAY |
|---|---|---|---|
| 1 | 14.03.20 | 17.03.20 | 18.03.20 |
| 2 | 18.03.20 | 20.03.20 | 23.03.20 |
| 3 | 23.03.20 | 24.03.20 | 25.03.20 |
| 4 | 25.03.20 | 26.03.20 | 28.03.20 |
| 5 | 28.03.20 | 29.03.20 | 01.04.20 |
| 6 | 01.04.20 | 02.04.20 | 03.04.20 |
| 7 | 03.04.20 | 05.04.20 | 07.04.20 |
| 8 | 07.04.20 | 12.04.20 | 14.04.20 |
| 9 | 18.04.20 | 20.04.20 | 21.04.20 |
| 10 | 21.04.20 | 22.04.20 | 23.04.20 |
| 11 | 23.04.20 | 24.04.20 | 27.04.20 |
| 12 | 27.04.20 | 30.04.20 | 04.05.20 |
| 13 | 28.05.20 | 29.05.20 | 02.06.20 |
| 14 | 02.06.20 | 03.06.20 | 04.06.20 |
| 15 | 04.06.20 | 05.06.20 | 08.06.20 |

# Pattern-Definition

```
    PATTERN (STRT DOWN+ UP+)

    DEFINE DOWN AS DOWN.fallzahl < PREV(DOWN.fallzahl),

           UP AS UP.fallzahl > PREV(UP.fallzahl)

);
```

- Das Muster wid in Form regulärer Ausdrücke auf Basis von Pattern-Variablen definiert.
- Die Pattern-Variablen nehmen bei der Abarbeitung jeweils die aktuelle Zeile als Wert an.
- Die Variablen werden in der DEFINE-Klausel mit logischen Bedingungen verknüpft.
- Die getroffenen Zeilen bilden analog zu den analytischen Funktionen ein Fenster, in dem navigiert und aggregiert werden kann.
- Wird eine Variable (hier STRT) nicht definiert, so trifft sie alle Zeilen.

# Ausgabe

```
MEASURES STRT.meldedatum AS start_day,
         LAST(DOWN.meldedatum) AS bottom_day,
         LAST(UP.meldedatum) AS high_day
ONE ROW PER MATCH
```

- Bei ONE ROW PER MATCH wird je Treffer nur eine Zeile produziert. Diese enthält neben den eventuellen Partitionierungsspalten nur die in der Measures-Klausel definierten Spalten.
- In der Measures-Klausel können ebenfalls die Navigations-Funktionen (FIRST, LAST, PREV, NEXT) und die Pattern-Variablen verwendet werden.
- Als Aggregatsfunktionen stehen zur Verfügung:
  MIN, MAX, SUM, COUNT, AVG (kein DISTINCT !)

# Beispiel mit Partitionierung: Ausgangstabelle

- V-Shapes sollen je Geschlecht gefunden werden:
  - W: 14.03.20 - 18.03.20
  - M: 14.03.20 - 18.03.20
  - M: 18.03.20 - 23.03.20
  - M: 23.03.20 - 25.03.20
  - W: 23.03.20 - 27.03.20
  - …

| | BUNDESLAND | LANDKREIS | MELDEDATUM | GESCHLECHT | FALLZAHL |
|---|---|---|---|---|---|
| 1 | Hessen | LK Fulda | 09.03.20 | M | 2 |
| 2 | Hessen | LK Fulda | 11.03.20 | W | 2 |
| 3 | Hessen | LK Fulda | 11.03.20 | M | 3 |
| 4 | Hessen | LK Fulda | 12.03.20 | W | 2 |
| 5 | Hessen | LK Fulda | 13.03.20 | W | 2 |
| 6 | Hessen | LK Fulda | 14.03.20 | W | 7 |
| 7 | Hessen | LK Fulda | 14.03.20 | M | 7 |
| 8 | Hessen | LK Fulda | 16.03.20 | M | 4 |
| 9 | Hessen | LK Fulda | 16.03.20 | W | 5 |
| 10 | Hessen | LK Fulda | 17.03.20 | W | 2 |
| 11 | Hessen | LK Fulda | 17.03.20 | M | 1 |
| 12 | Hessen | LK Fulda | 18.03.20 | W | 3 |
| 13 | Hessen | LK Fulda | 18.03.20 | M | 8 |
| 14 | Hessen | LK Fulda | 19.03.20 | M | 6 |
| 15 | Hessen | LK Fulda | 19.03.20 | W | 3 |
| 16 | Hessen | LK Fulda | 20.03.20 | M | 3 |
| 17 | Hessen | LK Fulda | 20.03.20 | W | 3 |
| 18 | Hessen | LK Fulda | 23.03.20 | W | 12 |

# Beispiel mit Partitionierung: Anweisung

```
SELECT * FROM CORZAHLENFULDAGESCHLECHT MATCH_RECOGNIZE (
    PARTITION BY geschlecht
    ORDER BY MELDEDATUM
    MEASURES STRT.meldedatum AS start_day,
             STRT.fallzahl AS start_zahl,
             DOWN.meldedatum AS bottom_day,
             DOWN.fallzahl AS bottom_zahl,
             UP.meldedatum AS high_day,
             UP.fallzahl AS high_zahl,
             round(avg(fallzahl),2) AS average
    ONE ROW PER MATCH
    AFTER MATCH SKIP TO LAST UP
    PATTERN (STRT DOWN+ UP+)
    DEFINE DOWN AS DOWN.fallzahl < PREV(DOWN.fallzahl),
           UP AS UP.fallzahl > PREV(UP.fallzahl)
    ) m ORDER BY m.start_day;
```

Partitionierung

LAST ist default

Aggregat

# Beispiel mit Partitionierung: Ergebnis

| | GESCHLECHT | START_DAY | START_ZAHL | BOTTOM_DAY | BOTTOM_ZAHL | HIGH_DAY | HIGH_ZAHL | AVERAGE |
|---|---|---|---|---|---|---|---|---|
| 1 | W | 14.03.20 | 7 | 17.03.20 | 2 | 18.03.20 | 3 | 4,25 |
| 2 | M | 14.03.20 | 7 | 17.03.20 | 1 | 18.03.20 | 8 | 5 |
| 3 | M | 18.03.20 | 8 | 20.03.20 | 3 | 23.03.20 | 12 | 7,25 |
| 4 | M | 23.03.20 | 12 | 24.03.20 | 2 | 25.03.20 | 7 | 7 |
| 5 | W | 23.03.20 | 12 | 26.03.20 | 1 | 27.03.20 | 7 | 5 |
| 6 | M | 28.03.20 | 5 | 29.03.20 | 1 | 01.04.20 | 10 | 5,5 |
| 7 | W | 28.03.20 | 7 | 29.03.20 | 3 | 31.03.20 | 9 | 6,33 |
| 8 | M | 01.04.20 | 10 | 05.04.20 | 1 | 07.04.20 | 5 | 4,67 |
| 9 | W | 01.04.20 | 9 | 02.04.20 | 4 | 03.04.20 | 9 | 7,33 |
| 10 | W | 03.04.20 | 9 | 05.04.20 | 1 | 07.04.20 | 10 | 5,75 |
| 11 | W | 07.04.20 | 10 | 12.04.20 | 1 | 14.04.20 | 6 | 5,17 |
| 12 | M | 07.04.20 | 5 | 08.04.20 | 4 | 09.04.20 | 6 | 5 |
| 13 | M | 09.04.20 | 6 | 12.04.20 | 1 | 14.04.20 | 4 | 3,5 |
| 14 | W | 14.04.20 | 6 | 16.04.20 | 1 | 17.04.20 | 2 | 3 |
| 15 | W | 17.04.20 | 2 | 18.04.20 | 1 | 21.04.20 | 2 | 1,67 |
| 16 | W | 21.04.20 | 2 | 22.04.20 | 1 | 23.04.20 | 2 | 1,67 |
| 17 | W | 27.04.20 | 2 | 30.04.20 | 1 | 11.05.20 | 2 | 1,67 |
| 18 | M | 12.05.20 | 3 | 14.05.20 | 1 | 28.05.20 | 5 | 2,8 |
| 19 | M | 28.05.20 | 5 | 29.05.20 | 1 | 02.06.20 | 4 | 3,33 |

# Ohne Geschlecht aber mit ALL ROWS PER MATCH

```
SELECT * FROM CORONAZAHLENFULDA MATCH_RECOGNIZE (
    ORDER BY MELDEDATUM
    MEASURES MATCH_NUMBER() AS match_num,
             CLASSIFIER() AS patternvar,
             STRT.meldedatum AS start_day,
             FINAL LAST(DOWN.meldedatum) AS bottom_day,
             FINAL LAST(UP.meldedatum) AS high_day,
             RUNNING COUNT(meldedatum) AS running_days,
             FINAL COUNT(down.meldedatum) AS down_count,
             FINAL COUNT(meldedatum) AS total_count
    ALL ROWS PER MATCH
    AFTER MATCH SKIP TO LAST UP
    PATTERN (STRT DOWN+ UP+)
    DEFINE DOWN AS DOWN.fallzahl < PREV(DOWN.fallzahl),
           UP AS UP.fallzahl > PREV(UP.fallzahl)
    ) m ORDER BY m.match_num, m.meldedatum;
```

Einzelzeilen ↔ Match

Aktuelle Pattern-Variable

Aggregate sind fortlaufend

# Ohne Geschlecht aber mit ALL ROWS PER MATCH: Ergebnis

| | MELDEDATUM | M_NUM | P_VAR | START_DAY | BOTTOM_DAY | HIGH_DAY | R_DAYS | D_COUNT | T_COUNT | BUNDESLAND | LANDKREIS | FALLZAHL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 14.03.20 | 1 | STRT | 14.03.20 | 17.03.20 | 18.03.20 | 1 | 2 | 4 | Hessen | LK Fulda | 14 |
| 2 | 16.03.20 | 1 | DOWN | 14.03.20 | 17.03.20 | 18.03.20 | 2 | 2 | 4 | Hessen | LK Fulda | 9 |
| 3 | 17.03.20 | 1 | DOWN | 14.03.20 | 17.03.20 | 18.03.20 | 3 | 2 | 4 | Hessen | LK Fulda | 3 |
| 4 | 18.03.20 | 1 | UP | 14.03.20 | 17.03.20 | 18.03.20 | 4 | 2 | 4 | Hessen | LK Fulda | 11 |
| 5 | 18.03.20 | 2 | STRT | 18.03.20 | 20.03.20 | 23.03.20 | 1 | 2 | 4 | Hessen | LK Fulda | 11 |
| 6 | 19.03.20 | 2 | DOWN | 18.03.20 | 20.03.20 | 23.03.20 | 2 | 2 | 4 | Hessen | LK Fulda | 9 |
| 7 | 20.03.20 | 2 | DOWN | 18.03.20 | 20.03.20 | 23.03.20 | 3 | 2 | 4 | Hessen | LK Fulda | 6 |
| 8 | 23.03.20 | 2 | UP | 18.03.20 | 20.03.20 | 23.03.20 | 4 | 2 | 4 | Hessen | LK Fulda | 24 |
| 9 | 23.03.20 | 3 | STRT | 23.03.20 | 24.03.20 | 25.03.20 | 1 | 1 | 3 | Hessen | LK Fulda | 24 |
| 10 | 24.03.20 | 3 | DOWN | 23.03.20 | 24.03.20 | 25.03.20 | 2 | 1 | 3 | Hessen | LK Fulda | 5 |
| 11 | 25.03.20 | 3 | UP | 23.03.20 | 24.03.20 | 25.03.20 | 3 | 1 | 3 | Hessen | LK Fulda | 9 |
| 12 | 25.03.20 | 4 | STRT | 25.03.20 | 26.03.20 | 28.03.20 | 1 | 1 | 4 | Hessen | LK Fulda | 9 |
| 13 | 26.03.20 | 4 | DOWN | 25.03.20 | 26.03.20 | 28.03.20 | 2 | 1 | 4 | Hessen | LK Fulda | 4 |
| 14 | 27.03.20 | 4 | UP | 25.03.20 | 26.03.20 | 28.03.20 | 3 | 1 | 4 | Hessen | LK Fulda | 10 |
| 15 | 28.03.20 | 4 | UP | 25.03.20 | 26.03.20 | 28.03.20 | 4 | 1 | 4 | Hessen | LK Fulda | 12 |
| 16 | 28.03.20 | 5 | STRT | 28.03.20 | 29.03.20 | 01.04.20 | 1 | 1 | 4 | Hessen | LK Fulda | 12 |
| 17 | 29.03.20 | 5 | DOWN | 28.03.20 | 29.03.20 | 01.04.20 | 2 | 1 | 4 | Hessen | LK Fulda | 4 |
| 18 | 31.03.20 | 5 | UP | 28.03.20 | 29.03.20 | 01.04.20 | 3 | 1 | 4 | Hessen | LK Fulda | 15 |
| 19 | 01.04.20 | 5 | UP | 28.03.20 | 29.03.20 | 01.04.20 | 4 | 1 | 4 | Hessen | LK Fulda | 19 |

# ALL ROWS PER MATCH

- Hier wird folgende Ausgabe je Zeile in dieser Reihenfolge erzeugt:
  - Eventuelle Partitionierungsspalten
  - Sortierspalten
  - Measures-Spalten
  - Alle sonstigen Spalten der Basistabelle
- Zur besseren Zuordnung der Spalten existieren die beiden Funktionen CLASSIFIER und MATCH_NUMBER.
- Die Aggregatsberechnung erfolgt wie bei den analytischen Funktionen standardmäßig fortlaufend.
  - RUNNING ist default
  - FINAL ist Pflicht für den jeweiligen Endwert

# Interner Ablauf beim Pattern Matching

1) Die Eingangsmenge wird, falls vorgesehen, partitioniert.

2) Jede Partition wird sortiert, sofern das nicht bereits der Fall ist

3) Jede Partition wird unabhängig von den anderen nach Übereinstimmung mit dem Muster durchsucht (Backtrace-Algorithmus):
   1) Beginnend mit der ersten Zeile werden iterativ solange die nachfolgende Zeilen abgearbeitet, bis entweder ein Treffer resultiert oder aber das Muster verletzt wird.
   2) Falls auf Basis der ersten Zeile kein Treffer resultiert, wird die nächste Zeile als Startpunkt gewählt und iteriert.
   3) Falls ein Treffer vorliegt, werden die Ausgabespalten berechnet.
   4) Anschließend wird der Prozess an einer zu definierenden Zeile erneut gestartet (AFTER MATCH SKIP …).

# AFTER MATCH SKIP

- AFTER MATCH SKIP PAST LAST ROW (default)
  - Wiederhole Pattern-Matching-Verfahren bei der ersten Zeile nach der letzten Zeile des aktuellen Treffers.
- AFTER MATCH SKIP TO NEXT ROW
  - Wiederhole Pattern-Matching-Verfahren bei der Zeile nach der ersten Zeile des aktuellen Treffers.
- AFTER MATCH SKIP TO FIRST *pattern_variable*
  - Wiederhole Pattern-Matching-Verfahren bei der ersten Zeile, die der Pattern-Variable zugewiesen wurde.
- AFTER MATCH SKIP TO LAST *pattern_variable*
  bzw. AFTER MATCH SKIP TO *pattern_variable*
  - Wiederhole Pattern-Matching-Verfahren bei der letzten Zeile, die der Pattern-Variable zugewiesen wurde.

# Reguläre Ausdrücke im Überblick

- Konkatenation, z.B: PATTERN (A B C)
- Quantifizierung
  - * beliebig viele, auch keines
  - + mindestens einmal
  - ? einmal oder keinmal
  - {n} n-mal
  - {n,} mindestens n-mal
  - {,n} maximal n-mal
  - {n,m} zwischen n- und m-mal (inklusive)
  - Default ist ***greedy***, kann auf ***reluctant*** geschaltet werden
- Alternative, z.B: PATTERN ( A | B | C )
- Gruppierung, z.B: PATTERN ( ( A B ){4} C )
- Absolute Anker
  - ^ vor der ersten Zeile der Partition
  - $ nach der letzten Zeile der Partition

# Etwas unsinnig: TOP-N hiermit

```
SELECT * FROM CORZAHLENFULDAGESCHLECHT MATCH_RECOGNIZE (
    PARTITION BY geschlecht
    ORDER BY fallzahl desc
    MEASURES RUNNING COUNT(*) AS r_count
    ALL ROWS PER MATCH
    -- AFTER MATCH SKIP PAST LAST ROW   -- default
    PATTERN ( ^STRT{1,4} )
    DEFINE    -- ist nicht optional
           STRT AS 1=1    -- dummy
    ) ORDER BY geschlecht;
```

Quantifizierer

# Begierig versus Zurückhaltend (Greedy versus Reluctant)

- Begierig: PATTERN ( X Y* Z)
  - Es werden solange Zeilen auf Y abgebildet, wie das maximal möglich ist.
  - Erfüllt eine Zeile sowohl die Bedingung für Y als auch Z, hat Y Vorrang, d.h. standardmäßig wird maximale Übereinstimmung gesucht.

- Zurückhaltend: PATTERN ( X Y*? Z)
  - Es werden solange Zeilen auf Y abgebildet, bis eine Zeile auf Z abgebildet werden kann.
  - Erfüllt eine Zeile sowohl die Bedingung für Y als auch Z, hat Z Vorrang, d.h. es wird die minimale Übereinstimmung gesucht.

# Navigations-Funktionen

- PREV, z.B: DEFINE A AS PREV ( A.fallzahl) > 10
  - Zugriff auf die Vorgängerzeile bzgl. physischer Zeilen, unabhängig vom Mapping auf Pattern-Variable
  - Falls kein Vorgänger existiert: NULL
  - Kann mit positiver Ganzzahl für Offset verwendet werden,
    z.B: PREV ( A.fallzahl, 2)

- NEXT, z.B: DEFINE A AS NEXT ( A.fallzahl) > 10
  - Analog PREV, aber Nachfolger-Zeile

- FIRST, z.B: FIRST ( A.fallzahl )
  - Zugriff auf die Vorgängerzeile, die auf die Pattern-Variable abgebildet wurde
  - Falls kein Vorgänger existiert: NULL
  - Kann mit positiver Ganzzahl für Offset verwendet werden,
    z.B: FIRST ( A.fallzahl, 2)
  - RUNNING-Semantik

- LAST, z.B: LAST ( A.fallzahl )
  - Analog FIRST, aber Nachfolger-Zeile

# Beispiel: Nutzung Navigationsfunktion

```
SELECT m.meldedatum, current_fallz, pre2_fallz FROM
CORONAZAHLENFULDA
    MATCH_RECOGNIZE (
    ORDER BY MELDEDATUM
    MEASURES  X.fallzahl AS current_fallz,
              PREV(X.fallzahl, 2) AS pre2_fallz
    ALL ROWS PER MATCH
    PATTERN (X)
    DEFINE X AS X.fallzahl > 2 * (PREV (X.fallzahl, 2))
    ) m ORDER BY m.meldedatum;
```

# Was ist mit leeren Treffern (empty match)

- Beispiel: PATTERN ( A* ) trifft 0 oder mehr Zeilen

- Bei einem leeren Treffer gilt bei ONE ROW PER MATCH:
  - Auch dieser bekommt eine MATCH_NUMBER zugeordnet (kann zur Identifikation benutzt werden).
  - COUNT-Aggregat liefert 0
  - Alle anderen Aggregate und die Navigationsfunktionen liefern NULL, Pattern-Variablen sind ebenso NULL.

- Bei einem leeren Treffer gilt bei ALL ROWS PER MATCH bzw. ALL ROWS PER MATCH SHOW EMPTY MATCHES (default):
  - Analog ONE ROW PER MATCH und zusätzlich:
  - CLASSIFIER-Funktion liefert NULL, sonstige Spaltenwerte aus der Ausgangsmenge werden 1:1 ausgegeben.

- Bei einem leeren Treffer gilt bei ALL ROWS PER MATCH OMIT EMPTY MATCHES:
  - Es wird keine Zeile generiert, d.h. MATCH_NUMBER hat Lücken.

- Empfehlung: Leere Treffer vermeiden.

# Einschränkungen

- Formulierung der Ausgangsmenge
  - Zwischen FROM und MATCH_RECOGNIZE ist keine andere Klausel erlaubt, auch kein WHERE und keine JOINS.
  - Sofern solche Anforderungen bestehen kann ein Subquery verwendet werden, z.B:

    ```
    FROM (SELECT * FROM tabX JOIN tabY ON tabx.z = tabY.z)
      MATCH_RECOGNIZE ( ...
    ```

- Geschachtelte MATCH_RECOGNIZE-Klauseln sind nicht möglich.
  - Bei Bedarf in Subqueries schachteln.

# Fragen / Diskussion

- ???

# Weitere Infos

- ANSI stellt einen entsprechenden Auszug aus dem Standard netterweise kostenfrei zur Verfügung: https://standards.iso.org/ittf/PubliclyAvailableStandards/c065143_ISO_IEC_TR_19075-5_2016.zip

- Oracle-Dokumentation: https://docs.oracle.com/en/database/oracle/oracle-database/21/dwhsg/sql-pattern-matching-data-warehouses.html#GUID-136DAC89-DA17-45C6-9E37-C9892723AC79

- Diverse Blogs

- Weitere Programmbeispiele bei ORACLE LIVE SQL: https://livesql.oracle.com/apex/livesql/file/tutorial_E4DB2E0Z0D5ZTUBGN6JWUPKAU.html