正規表現で行パターンを検索する「行パターン認識」をPostgreSQLに実装しよう!

2023/11/24
JPUG PostgreSQL
カンファレンス

SRA OSS合同会社
石井　達夫
陳　寧イ

自己紹介

- 石井達夫
  - SRA OSS合同会社顧問
  - PostgreSQLコミッター
    - PostgreSQLの国際化
    - pgbench, pgstattuple, pgrowlocks, Pgpool-II
- 陳寧イ
  - SRA OSS合同会社データベース技術グループ所属
  - 社内データベースアプリメインテナンス作業担当
  - 現在はPostgreSQL開発に向けて勉強中

SRA OSS

本日の内容

- Row Pattern Recognition(行パターン認識)とは
- Row Pattern Recognitionの構文
- Row Pattern Recognitionの実装と今後の計画
- デモ

SRA OSS

Row Pattern Recognitionとは

- SQL標準で定義されている機能の一つ。行の並び（時系列データなど）に関する「パターン」を指定して検索することができる
- パターンは論理式で柔軟に記述できる
  - LOWPRICE AS price < 100
  - UP AS price > PREV(price)
  - DOWN AS price < PREV(price)
- パターンの並びを正規表現で記述できる
  - LOWPRICE UP+ DOWN+
- 株価の変動パターンの検索、異常値の検出など、幅広い応用が考えられる

行パターン認識とWHERE句の比較

tdate	price
2023-07-01	100
2023-07-02	200
2023-07-03	150
2023-07-04	140
2023-07-05	150
2023-07-06	90
2023-07-07	110
2023-07-08	130
2023-07-09	120
2023-07-10	130

WHERE price = 150

tdate	price
2023-07-03	150
2023-07-05	150

LOWPRICE UP+ DOWN+

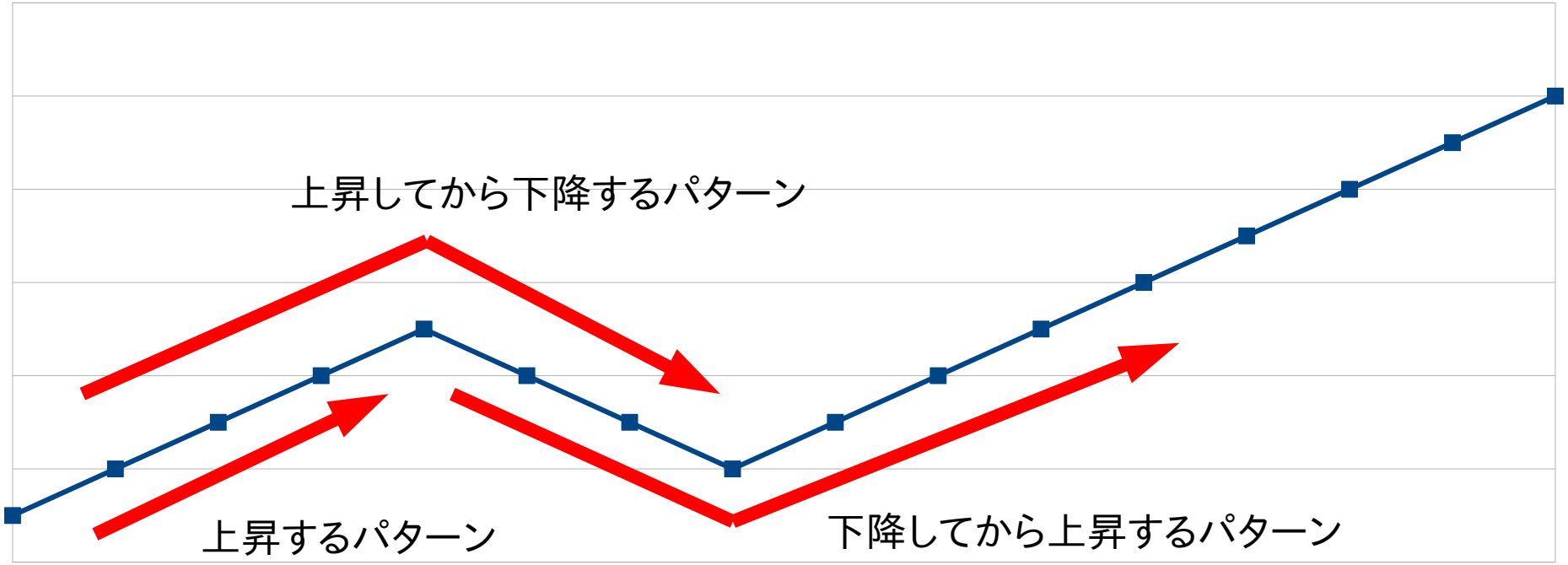
行パターン認識

tdate	price	first_value	last_value
2023-07-06	90	2023-07-06	2023-07-09
2023-07-07	110		
2023-07-08	130		
2023-07-09	120		

Row Pattern Recognitionの規格と実装

- SQL:2016に最初に登場した比較的新しい機能
  - “SQL/RPR”と呼ぶ
- 汎用RDBMSでは、唯一Oracleにのみ実装。汎用OSS RDBMSではまだ実装例がない
  - 汎用RDBMSではないが、Analytics の世界では”Trino”などが実装済み
    - https://trino.io/
  - 汎用RDBMSでWindow句でのRPRの実装例はない
- 以後Row Pattern Recognition、SQL/RPRを略して “RPR”と本スライドでは呼称する

RPRでできることの例1：時系列データから変化パターンを検索



上昇してから下降するパターンの検索

company	tdate	price	
Company1	2023-07-01	100	START
Company1	2023-07-02	200	UP
Company1	2023-07-03	150	DOWN
Company1	2023-07-04	140	DOWN
company1	2023-07-05	150	
Company1	2023-07-06	90	START
Company1	2023-07-07	110	UP
Company1	2023-07-08	130	UP
Company1	2023-07-09	120	DOWN
company1	2023-07-10	130	

priceが1回以上上昇し、その後1回以上下降するような行の並び(sequence)を見つける

↓ パターン変数の定義

START AS TRUE,
UP AS price > PREV(price),
DOWN AS price < PREV(price)

↓

START
UP+
DOWN+

パターンの並びを正規表現で定義

実際のクエリの例

```
SELECT company, tdate, price,
first_value(price) OVER w,  -- フレームの最初の行を返す関数
last_value(price) OVER w    -- フレームの最後の行を返す関数
 FROM stock
 WINDOW w AS (
 PARTITION BY company
 ORDER BY tdate
 ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
 AFTER MATCH SKIP PAST LAST ROW
 INITIAL
 PATTERN (START UP+ DOWN+) -- パターンの定義
 DEFINE                    -- パターン変数の定義
  START AS TRUE,
  UP AS price > PREV(price),
  DOWN AS price < PREV(price)
);
```

検索結果の例

company	tdate	price	first_value	last_value
company1	2023-07-01	100	100	140
company1	2023-07-02	200		
company1	2023-07-03	150		
company1	2023-07-04	140		
company1	2023-07-05	150		
company1	2023-07-06	90	90	120
company1	2023-07-07	110		
company1	2023-07-08	130		
company1	2023-07-09	120		
company1	2023-07-10	130		

(10 rows)

RPRでできることの例2：連続する特定事象の検出

alert_time	alert_msg	first_alert	last_alert
2023-11-24 09:00:00	Warning: device is not ready		
2023-11-24 09:01:00	Log: device is busy		
2023-11-24 09:02:00	Log: device is busy		
2023-11-24 09:03:00	Warning: device is not ready		
2023-11-24 09:04:00	Warning: device is not ready		
2023-11-24 09:05:00	Warning: device is not ready		
2023-11-24 09:10:00	Log: device is busy		
2023-11-24 09:20:00	Log: device is busy		
2023-11-24 09:21:00	Warning: device is not ready	2023-11-24 09:21:00	2023-11-24 09:21:20
2023-11-24 09:21:10	Warning: device is not ready		
2023-11-24 09:21:20	Warning: device is not ready		
2023-11-24 09:22:00	Log: device is busy		

30秒以内に“Warning”メッセージが
3回以上連続している事象を検出

RPRでの表現

alert_time	alert_msg
2023-11-24 09:00:00	Warning: device is not ready
2023-11-24 09:01:00	Log: device is busy
2023-11-24 09:02:00	Log: device is busy
2023-11-24 09:03:00	Warning: device is not ready
2023-11-24 09:04:00	Warning: device is not ready
2023-11-24 09:05:00	Warning: device is not ready
2023-11-24 09:10:00	Log: device is busy
2023-11-24 09:20:00	Log: device is busy
2023-11-24 09:21:00	Warning: device is not ready
2023-11-24 09:21:10	Warning: device is not ready
2023-11-24 09:21:20	Warning: device is not ready
2023-11-24 09:22:00	Log: device is busy

メッセージが“Warning”で始まり、前のメッセージから30秒以内に3回以上連続している

↓ パターン変数の定義

START AS alert_msg LIKE ‘Warning%’,
WARNING AS alert_msg LIKE ‘Warning%’ AND (alert_time – PREV (alert_time) < interval ‘30 seconds’

↓

START
WARNING
WARNING+

パターンの並びを正規表現で定義

実際のクエリの例

```
SELECT alert_time, alert_msg,
first_value(alert_time) OVER w AS first_alert,
last_value(alert_time) OVER w AS last_alert
 FROM alerts
 WINDOW w AS (
 PARTITION BY device_id
 ORDER BY alert_time
 ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
 INITIAL
 PATTERN (START WARNING WARNING+)
 DEFINE
  START AS alert_msg LIKE 'Warning%',
  WARNING AS alert_msg LIKE 'Warning%' AND
  (alert_time - PREV(alert_time)) < interval '30 seconds'
);
```

検索結果の例

alert_time	alert_msg	first_alert	last_alert
2023-11-24 09:00:00	Warning: device is not ready		
2023-11-24 09:01:00	Log: device is busy		
2023-11-24 09:02:00	Log: device is busy		
2023-11-24 09:03:00	Warning: device is not ready		
2023-11-24 09:04:00	Warning: device is not ready		
2023-11-24 09:05:00	Warning: device is not ready		
2023-11-24 09:10:00	Log: device is busy		
2023-11-24 09:20:00	Log: device is busy		
2023-11-24 09:21:00	Warning: device is not ready	2023-11-24 09:21:00	2023-11-24 09:21:20
2023-11-24 09:21:10	Warning: device is not ready		
2023-11-24 09:21:20	Warning: device is not ready		
2023-11-24 09:22:00	Log: device is busy		

なぜWindow句？

- 実はRPRには2種類ある
  - R010: Row pattern recognition: FROM clause
    - FROMの後にMATCH_RECOGNIZEという句を追加、そこにRPRの定義を書く
  - R020: Row pattern recognition: WINDOW clause
    - Window句にRPRを書く
  - 共通部分が多く、記述能力は大差ない
- RPRをWindow句で実装した理由
  - RPRは、入力行の集合を何度もスキャンする必要がある
    - FROM句の普通のスキャンでは難しい
  - Window句では、すでにWindow句の入力行の集合を何度もスキャンできるインフラが実装されている
  - 汎用RDBMSでの実装例がまだない(一番乗りしたい!)

R010によるクエリの例

```
SELECT company, tdate, price, m.first_val, m.last_val
FROM stock
MATCH_RECOGNIZE
(
 PARTITION BY company
 ORDER BY tdate
 MEASURES
  FIRST(price) AS first_val,
  LAST(price) AS last_val,
 AFTER MATCH SKIP PAST LAST ROW
 PATTERN (START UP+ DOWN+)  -- パターンの定義
 DEFINE                     -- パターン変数の定義
  START AS TRUE,
  UP AS price > PREV(price),
  DOWN AS price < PREV(price)
) AS m;
```

RPR(R020)の構文詳細

```
WINDOW window_name AS (
[ PARTITION BY ... ]
[ ORDER BY... ]
[ MEASURES ... ]
ROWS BETWEEN CURRENT ROW AND ...
[ AFTER MATCH SKIP ... ]
[ INITIAL|SEEK ]
PATTERN (...)
[ SUBSET ... ]
DEFINE ...
)
```

- PARTITION BY, ORDER BYはRPRなしのWindow句と同じなので説明省略
- MEASURESとSUBSETは未実装なので説明省略

SRA OSS

ROWS BETWEEN CURRENT ROW...

- RPRのないWindow句と同じで、フレームの開始、終了を指定する
- フレームの開始は現在行(ROWS BETWEEN CURRENT ROW)でなければならない
- フレームの終了はフレームの終端(UNBOUNDED FOLLOWING)か、現在行からn行(n FOLLOWING)のみが許される
- 結局RPRでは以下の2つのみが許されている
  - ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
  - ROWS BETWEEN CURRENT ROW AND n FOLLOWING

ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING

パーティション全体

company	tdate	price
company1	07-01-2023	100
company1	07-02-2023	200
company1	07-03-2023	150
company1	07-04-2023	140
company1	07-05-2023	150
company1	07-06-2023	90
company1	07-07-2023	110
company1	07-08-2023	130
company1	07-09-2023	120
company1	07-10-2023	130

最初のフレームはこの行から最後まで
次のフレームはこの行から最後まで
そのフレームはこの行から最後まで

RPR(R020)の構文詳細

```
WINDOW window_name AS (
[ PARTITION BY ... ]
[ ORDER BY... ]
[ MEASURES ... ]
ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
[ AFTER MATCH SKIP ... ]
[ INITIAL|SEEK ]
PATTERN (...)
[ SUBSET ... ]
DEFINE ...
)
```

AFTER MATCH SKIP...

- RPR用にのみ存在する
- パターンマッチを行った後、次のパターンマッチをどの行から開始するかを指定する
  - AFTER MATCH SKIP PAST LAST ROW
    - デフォルト。たとえば1行目からパターンマッチを開始し、4行マッチしたら、次のパターンマッチは5行目から開始する
  - AFTER MATCH SKIP TO NEXT ROW
    - たとえば1行目からパターンマッチを開始したら、次は2行目からパターンマッチを開始する
  - AFTER MATCH SKIP TO FIRST|LAST パターン変数名
    - パターンマッチした変数名が存在する最初(FIRST)あるいは最後(LAST)の行からパターンマッチを開始する
    - 現在のパッチでは未サポート

RPR(R020)の構文詳細

```
WINDOW window_name AS (
[ PARTITION BY ... ]
[ ORDER BY... ]
[ MEASURES ... ]
ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
[ AFTER MATCH SKIP ... ]
[ INITIAL|SEEK ]
PATTERN (...)
[ SUBSET ... ]
DEFINE ...
)
```

INITIAL|SEEK

- RPR用にのみ存在する
- INITIAL
  - マッチした行の集合がフレームの先頭行から始まるときにのみマッチが成功したと見なす
  - デフォルト
- SEEK
  - マッチした行の集合の先頭行がフレームの先頭でなくてもマッチが成功したと見なす
  - 現在のパッチではサポートされていない

RPR(R020)の構文詳細

```
WINDOW window_name AS (
[ PARTITION BY ... ]
[ ORDER BY... ]
[ MEASURES ... ]
ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
[ AFTER MATCH SKIP ... ]
[ INITIAL|SEEK ]
PATTERN (...)
[ SUBSET ... ]
DEFINE ...
)
```

DEFINE(1)

- RPR用にのみ存在する
- パターンマッチ変数を定義する
  - DEFINE 変数名 AS 論理式, …
  - 論理式は任意の論理式が書ける
    - START AS TRUE
    - LOWPRICE AS price < 100
    - UP AS price > PREV(price)
    - UP AS price > 100 AND price PREV(price)

DEFINE(2)

- DEFINE句では、row pattern navigation operationと呼ばれる関数的なものが利用できる
  - FIRST, LAST, PREV, NEXT
  - 現在のパッチでは、PREV/NEXTのみサポート
- 以下は現在のパッチでは未サポート
  - CLASSIFIER(パターンマッチした変数名を文字列で返す関数)
  - 集約関数
  - 副問合せ

RPR(R020)の構文詳細

```
WINDOW window_name AS (
[ PARTITION BY ... ]
[ ORDER BY... ]
[ MEASURES ... ]
ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
[ AFTER MATCH SKIP ... ]
[ INITIAL|SEEK ]
PATTERN (...)
[ SUBSET ... ]
DEFINE ...
)
```

PATTERN(1)

- DEFINEで定義した変数を使って、一致させたいパターンを記述する
  - PATTERN (START UP+ DOWN+)
    - STARTにマッチする行(1行目)
    - UPにマッチする行が1行以上続く
    - DOWNにマッチする行が1行以上続く
  - DEFINEで記述していない変数Aが出現したら、「A IS TRUE」と見なす

PATTERN(2)

- 記述には正規表現も使える
  - +: 1行以上
  - *: 0行以上
  - ?: 0または1行
  - A | B: OR条件
  - (A B): グルーピング
  - {n}: n行
  - {n,}: n行以上
  - {n,m}: n行以上m行以下
  - {,m}: 0行以上m行以下
- 現在のパッチでは “+” と “*” のみサポート

ターゲットリストでの集約の扱い

- 基本的にはWindow句における集約と同じ扱い
- ただしRPRが適用されると、集約結果はマッチした行内に限定される

集約関数の実行例

```
SELECT company, tdate, price,
count(*) OVER w1 AS count1,
count(*) OVER w2 AS count2
 FROM stock
 WINDOW w1 AS (
 PARTITION BY company
 ORDER BY tdate
),
 w2 AS (w1
 ROWS BETWEEN CURRENT ROW AND
 UNBOUNDED FOLLOWING
 AFTER MATCH SKIP PAST LAST ROW
 INITIAL
 PATTERN (START UP+ DOWN+)
 DEFINE
  START AS TRUE,
  UP AS price > PREV(price),
  DOWN AS price < PREV(price)
);
```

company	tdate	price	count1	count2
company1	2023-07-01	100	10	4
company1	2023-07-02	200	9	
company1	2023-07-03	150	8	
company1	2023-07-04	140	7	
company1	2023-07-05	150	6	0
company1	2023-07-06	90	5	4
company1	2023-07-07	110	4	
company1	2023-07-08	130	3	
company1	2023-07-09	120	2	
company1	2023-07-10	130	1	0

(10 rows)

RPRのPostgreSQL実装

- Window句に対して実装
- 開発中のPostgreSQL 17に対するパッチをpgsql-hackersで提案中
  - https://www.postgresql.org/message-id/20230625.210509.1276733411677577841.t-ishii%40sranhm.sra.co.jp
  - パッチは、パーサ、プランナ、エグゼキュータに渡る修正となっており、パッチ行数で約3,000行(テスト、ドキュメントのパッチを除く)

正規表現検索の実装

```
PATTERN (START UP+ DOWN+)
DEFINE
  START AS TRUE,
  UP AS price > PREV(price),
  DOWN AS price < PREV(price)
```

```
price    マッチした変数
-----------------------------
100      START
200      START UP
150      START DOWN
140      START DOWN
150      START UP
```

可能な組み合わせを生成

```
START START START START START
START UP START START START
START UP DOWN START START
START UP DOWN START START
:
:
START UP DOWN DOWN UP
```

正規表現検索
START UP+ DOWN+

パターンマッチの結果を
文字列にして表現、
正規表現検索する

最長マッチが検索結果

START UP DOWN DOWN UP

今後の予定

- どのレベルの実装でPostgreSQLに取り込むか検討中
  - 未実装の機能はまだまだあるが、これ以上実装を増やすとパッチが巨大化する懸念がある
    - パッチが巨大化するとレビューが困難になり、取り込んでもらえなくなる
  - 今の実装レベルでも実用的には使える局面も多いので、今の機能レベルでとりあえずPostgreSQL 17か18に取り込みたい
  - その後は、残りの実装に取り組みたい
    - MEASURE句などの実装
    - MATCH RECOGNIZE (R010)の実装

ここまでのまとめ

- 行パターン認識(RPR)は、今までのSQLでは表現が難しかった高度な検索を可能にします
- RPRでは、正規表現を用いて行パターンを記述します
- RPRをWindow句で利用する場合の構文を紹介しました
- RPRをPostgreSQLに実装する試みを紹介しました

参考文献、URL

- ISO/IEC 19075-5 “Information technology – Guidance for the use of database language SQL – Part5: Row Pattern Recognition”
- SQLのビッグデータ対応
  - https://www.sraoss.co.jp/tech-blog/db-special-lecture/masunaga-db-special-lecture-3/
- Row Pattern Recognition in SQL
  - https://sqlperformance.com/2019/04/t-sql-queries/row-pattern-recognition-in-sql
- Specification of Row Pattern Recognition in the SQL Standard and its Implementations
  - https://link.springer.com/article/10.1007/s13222-022-00404-3
- Trino 426 Documentation (MATCH_RECOGNIZE)
  - https://trino.io/docs/current/sql/match-recognize.html
- Trino 426 Documentation (Row pattern recognition in window structures)
  - https://trino.io/docs/current/sql/pattern-recognition-in-window.html
- Oracle12cマニュアルより
  - https://docs.oracle.com/cd/F19136_01/dwhsg/sql-pattern-matching-data-warehouses.html#GUID-136DAC89-DA17-45C6-9E37-C9892723AC79

デモ

- 陳さんにバトンタッチ!