Getting Started with SQL Pattern Matching in Oracle 12c

Galo Balda

June 25 2014 – Seattle, WA
About Me

Originally From

Database Engineer at the Medicaid Applications Group

galo.balda@gmail.com

@GaloBalda

galobalda.wordpress.com
Agenda

- Evolution of Pattern Matching in Oracle.

- Where can I use it?

- How does it work?

- Examples
  - V-Shape Pattern in Stock Prices.
  - Web Log Analysis: Sessionization.
  - Financial Tracking.
Evolution of Pattern Matching in Oracle

- Like Operator (Before 10g)
- Regular Expressions (10g)
- SQL Pattern Matching (12c)
Where can I use Pattern Matching?

- Financial Services
  - Fraud
  - Tracking Stock Market
  - Money Laundering

- Utilities
  - Fraud
  - Network Analysis
  - Unusual Usage

- Law & Order
  - Monitoring Suspicious Activities

- Retail
  - Buying Patterns
  - Returns Fraud
  - Session-ization

- Telcos
  - SIM Card Fraud
  - Call Quality

How does it work?

Basic Syntax:

```
FROM [row pattern input table]
MATCH_RECOGNIZE
 ( [ PARTITION BY <cols> ]
     [ ORDER BY <cols> ]
     [ MEASURES <cols> ]
     [ ONE ROW PER MATCH | ALL ROWS PER MATCH ]
     [ SKIP_TO_option ]
     PATTERN ( <row pattern> )
     [ SUBSET <subset list> ]
     DEFINE <definition list>
 )
```
Building Regular Expressions

Some of the supported operators:

- **Concatenation**: No operator between elements.
- **Quantifiers**:
  - `*`: 0 or more matches.
  - `+`: 1 or more matches
  - `?`: 0 or 1 match.
  - `{n}`: Exactly `n` matches.
  - `{n,}`: `n` or more matches.
  - `{n, m}`: Between `n` and `m` (inclusive) matches.
  - `{, m}`: Between 0 and `m` (inclusive) matches.
- **Alternation**: `|`
- **Grouping**: `()`
Functions

- **CLASIFFIER()**: Which rows are members of which match.
- **MATCH_NUMBER()**: Which pattern variable applies to which rows.
- **PREV()**: Access to a column/expression in a previous row.
- **NEXT()**: Access to a column/expression in a next row.
- **LAST()**: Last value within the pattern match.
- **COUNT(), AVG(), MAX(), MIN(), SUM()**
Example #1

FIND ALL THE CASES WHERE STOCK PRICES DIPPED TO A BOTTOM PRICE AND THEN ROSE
V-Shape Pattern in Stock Prices

CREATE TABLE Ticker
(
  symbol VARCHAR2(10),
  tstamp DATE,
  price NUMBER
);

INSERT INTO Ticker VALUES('ACME', '01-Apr-11', 12);
INSERT INTO Ticker VALUES('ACME', '02-Apr-11', 17);
INSERT INTO Ticker VALUES('ACME', '03-Apr-11', 19);
INSERT INTO Ticker VALUES('ACME', '04-Apr-11', 21);
INSERT INTO Ticker VALUES('ACME', '05-Apr-11', 25);
INSERT INTO Ticker VALUES('ACME', '06-Apr-11', 12);
INSERT INTO Ticker VALUES('ACME', '07-Apr-11', 15);
INSERT INTO Ticker VALUES('ACME', '08-Apr-11', 20);
INSERT INTO Ticker VALUES('ACME', '09-Apr-11', 24);
INSERT INTO Ticker VALUES('ACME', '10-Apr-11', 25);
INSERT INTO Ticker VALUES('ACME', '11-Apr-11', 19);
INSERT INTO Ticker VALUES('ACME', '12-Apr-11', 15);
INSERT INTO Ticker VALUES('ACME', '13-Apr-11', 25);
INSERT INTO Ticker VALUES('ACME', '14-Apr-11', 25);
INSERT INTO Ticker VALUES('ACME', '15-Apr-11', 14);
INSERT INTO Ticker VALUES('ACME', '16-Apr-11', 12);
INSERT INTO Ticker VALUES('ACME', '17-Apr-11', 14);
INSERT INTO Ticker VALUES('ACME', '18-Apr-11', 24);
INSERT INTO Ticker VALUES('ACME', '19-Apr-11', 23);
INSERT INTO Ticker VALUES('ACME', '20-Apr-11', 22);
V-Shape Pattern in Stock Prices
V-Shape Pattern in Stock Prices

One Row Per Match

```
SELECT * FROM ticker
MATCH_RECOGNIZE
(
  PARTITION BY symbol ORDER BY tstamp
  MEASURES
    strt.tstamp as start_tstamp, strt.price as start_price, LAST(down.tstamp) as bottom_tstamp, LAST(down.price) as bottom_price, LAST(up.tstamp) as end_tstamp, LAST(up.price) as end_price
  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;
```
V-Shape Pattern in Stock Prices

One Row Per Match
V-Shape Pattern in Stock Prices

One Row Per Match

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>START_TSTAMP</th>
<th>START_PRICE</th>
<th>BOTTOM_TSTAMP</th>
<th>BOTTOM_PRICE</th>
<th>END_TSTAMP</th>
<th>END_PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACME</td>
<td>05-APR-11</td>
<td>25</td>
<td></td>
<td>12</td>
<td>10-APR-11</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>10-APR-11</td>
<td>25</td>
<td>12-APR-11</td>
<td>15</td>
<td>13-APR-11</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>14-APR-11</td>
<td>25</td>
<td>16-APR-11</td>
<td>12</td>
<td>18-APR-11</td>
<td>24</td>
</tr>
</tbody>
</table>
V-Shape Pattern in Stock Prices
V-Shape Pattern in Stock Prices

All Rows Per Match

```sql
SELECT * FROM ticker
MATCH_RECOGNIZE
(
    PARTITION BY symbol ORDER BY tstamp
    MEASURES
    strt.tstamp as start_tstamp, FINAL LAST(down.tstamp) as bottom_tstamp,
    FINAL LAST(up.tstamp) as end_tstamp, MATCH_NUMBER() as match_num,
    CLASSIFIER() as var_match
    ALL ROWS 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.match_num, mr.tstamp;
```
V-Shape Pattern in Stock Prices

All Rows Per Match

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>TSTAMP</th>
<th>START_TSTAMP</th>
<th>BOTTOM_TSTAMP</th>
<th>END_TSTAMP</th>
<th>MATCH_NUM</th>
<th>VAR_MATCH</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACME</td>
<td>05-APR-11</td>
<td>05-APR-11</td>
<td>06-APR-11</td>
<td>10-APR-11</td>
<td>1</td>
<td>STRT</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>06-APR-11</td>
<td>05-APR-11</td>
<td>06-APR-11</td>
<td>10-APR-11</td>
<td>1</td>
<td>DOWN</td>
<td>12</td>
</tr>
<tr>
<td>ACME</td>
<td>07-APR-11</td>
<td>05-APR-11</td>
<td>06-APR-11</td>
<td>10-APR-11</td>
<td>1</td>
<td>UP</td>
<td>15</td>
</tr>
<tr>
<td>ACME</td>
<td>08-APR-11</td>
<td>05-APR-11</td>
<td>06-APR-11</td>
<td>10-APR-11</td>
<td>1</td>
<td>UP</td>
<td>20</td>
</tr>
<tr>
<td>ACME</td>
<td>09-APR-11</td>
<td>05-APR-11</td>
<td>06-APR-11</td>
<td>10-APR-11</td>
<td>1</td>
<td>UP</td>
<td>24</td>
</tr>
<tr>
<td>ACME</td>
<td>10-APR-11</td>
<td>05-APR-11</td>
<td>06-APR-11</td>
<td>10-APR-11</td>
<td>1</td>
<td>UP</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>10-APR-11</td>
<td>10-APR-11</td>
<td>12-APR-11</td>
<td>13-APR-11</td>
<td>2</td>
<td>STRT</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>11-APR-11</td>
<td>10-APR-11</td>
<td>12-APR-11</td>
<td>13-APR-11</td>
<td>2</td>
<td>DOWN</td>
<td>19</td>
</tr>
<tr>
<td>ACME</td>
<td>12-APR-11</td>
<td>10-APR-11</td>
<td>12-APR-11</td>
<td>13-APR-11</td>
<td>2</td>
<td>DOWN</td>
<td>15</td>
</tr>
<tr>
<td>ACME</td>
<td>14-APR-11</td>
<td>14-APR-11</td>
<td>16-APR-11</td>
<td>18-APR-11</td>
<td>3</td>
<td>STRT</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>15-APR-11</td>
<td>14-APR-11</td>
<td>16-APR-11</td>
<td>18-APR-11</td>
<td>3</td>
<td>DOWN</td>
<td>14</td>
</tr>
<tr>
<td>ACME</td>
<td>16-APR-11</td>
<td>14-APR-11</td>
<td>16-APR-11</td>
<td>18-APR-11</td>
<td>3</td>
<td>DOWN</td>
<td>12</td>
</tr>
<tr>
<td>ACME</td>
<td>17-APR-11</td>
<td>14-APR-11</td>
<td>16-APR-11</td>
<td>18-APR-11</td>
<td>3</td>
<td>UP</td>
<td>14</td>
</tr>
<tr>
<td>ACME</td>
<td>18-APR-11</td>
<td>14-APR-11</td>
<td>16-APR-11</td>
<td>18-APR-11</td>
<td>3</td>
<td>UP</td>
<td>24</td>
</tr>
</tbody>
</table>
V-Shape Pattern in Stock Prices
V-Shape Pattern in Stock Prices

Aggregate on a Variable

```sql
SELECT * FROM ticker
MATCH_RECOGNIZE
(
    PARTITION BY symbol ORDER BY tstamp
    MEASURES
        MATCH_NUMBER() as match_num, CLASSIFIER() as var_match, FINAL COUNT(up.tstamp) as up_days, FINAL COUNT(tstamp) as total_days, COUNT(tstamp) as cnt_days, price - strt.price as price_dif
    ALL ROWS 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.match_num, mr.tstamp;
```
## V-Shape Pattern in Stock Prices

**Aggregate on a Variable**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>TSTAMP</th>
<th>MATCH_NUM</th>
<th>VAR_MATCH</th>
<th>UP_DAYS</th>
<th>TOTAL_DAYS</th>
<th>CNT_DAYS</th>
<th>PRICE_DIF</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACME</td>
<td>05-APR-11</td>
<td>1</td>
<td>STRT</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>06-APR-11</td>
<td>1</td>
<td>DOWN</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>-13</td>
<td>12</td>
</tr>
<tr>
<td>ACME</td>
<td>07-APR-11</td>
<td>1</td>
<td>UP</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>-10</td>
<td>15</td>
</tr>
<tr>
<td>ACME</td>
<td>08-APR-11</td>
<td>1</td>
<td>UP</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>-5</td>
<td>20</td>
</tr>
<tr>
<td>ACME</td>
<td>09-APR-11</td>
<td>1</td>
<td>UP</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>-1</td>
<td>24</td>
</tr>
<tr>
<td>ACME</td>
<td>10-APR-11</td>
<td>1</td>
<td>UP</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>10-APR-11</td>
<td>2</td>
<td>STRT</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>11-APR-11</td>
<td>2</td>
<td>DOWN</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>-6</td>
<td>19</td>
</tr>
<tr>
<td>ACME</td>
<td>12-APR-11</td>
<td>2</td>
<td>DOWN</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>-10</td>
<td>15</td>
</tr>
<tr>
<td>ACME</td>
<td>13-APR-11</td>
<td>2</td>
<td>UP</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>14-APR-11</td>
<td>3</td>
<td>STRT</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>ACME</td>
<td>15-APR-11</td>
<td>3</td>
<td>DOWN</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>-11</td>
<td>14</td>
</tr>
<tr>
<td>ACME</td>
<td>16-APR-11</td>
<td>3</td>
<td>DOWN</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>-13</td>
<td>12</td>
</tr>
<tr>
<td>ACME</td>
<td>17-APR-11</td>
<td>3</td>
<td>UP</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>-11</td>
<td>14</td>
</tr>
<tr>
<td>ACME</td>
<td>18-APR-11</td>
<td>3</td>
<td>UP</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>-1</td>
<td>24</td>
</tr>
</tbody>
</table>
Example #2

WEB LOG ANALYSIS: DEFINE SESSIONS OF USER ACTIVITY
CREATE TABLE Events
(
    Time_Stamp NUMBER,
    User_ID VARCHAR2(10)
);

INSERT INTO Events VALUES(1,'Mary');
INSERT INTO Events VALUES(11,'Mary');
INSERT INTO Events VALUES(23,'Mary');
INSERT INTO Events VALUES(34,'Mary');
INSERT INTO Events VALUES(44,'Mary');
INSERT INTO Events VALUES(53,'Mary');
INSERT INTO Events VALUES(63,'Mary');
INSERT INTO Events VALUES(3,'Richard');
INSERT INTO Events VALUES(13,'Richard');
INSERT INTO Events VALUES(23,'Richard');
INSERT INTO Events VALUES(33,'Richard');
INSERT INTO Events VALUES(43,'Richard');
INSERT INTO Events VALUES(47,'Sam');
INSERT INTO Events VALUES(48,'Sam');
INSERT INTO Events VALUES(59,'Sam');
INSERT INTO Events VALUES(60,'Sam');
INSERT INTO Events VALUES(68,'Sam');

Web Log Analysis: Sessionization
Web Log Analysis: Sessionization

```sql
SELECT time_stamp, user_id, session_id
FROM Events
MATCH RECOGNIZE
(
  PARTITION BY user_id ORDER BY time_stamp
  MEASURES
    MATCH_NUMBER() as session_id
  ALL ROWS PER MATCH
  PATTERN (b s*)
  DEFINE
    s as (s.time_stamp - PREV(time_stamp) <= 10)
)
ORDER BY user_id, time_stamp;
```

<table>
<thead>
<tr>
<th>TIME_STAMP</th>
<th>USER_ID</th>
<th>SESSION_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mary</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Mary</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>Mary</td>
<td>2</td>
</tr>
<tr>
<td>34</td>
<td>Mary</td>
<td>3</td>
</tr>
<tr>
<td>44</td>
<td>Mary</td>
<td>3</td>
</tr>
<tr>
<td>53</td>
<td>Mary</td>
<td>3</td>
</tr>
<tr>
<td>63</td>
<td>Mary</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Richard</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>Richard</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>Richard</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>Richard</td>
<td>1</td>
</tr>
<tr>
<td>43</td>
<td>Richard</td>
<td>1</td>
</tr>
<tr>
<td>54</td>
<td>Richard</td>
<td>2</td>
</tr>
<tr>
<td>63</td>
<td>Richard</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Sam</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Sam</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>Sam</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>Sam</td>
<td>1</td>
</tr>
<tr>
<td>43</td>
<td>Sam</td>
<td>2</td>
</tr>
<tr>
<td>47</td>
<td>Sam</td>
<td>2</td>
</tr>
<tr>
<td>48</td>
<td>Sam</td>
<td>2</td>
</tr>
<tr>
<td>59</td>
<td>Sam</td>
<td>3</td>
</tr>
<tr>
<td>60</td>
<td>Sam</td>
<td>3</td>
</tr>
<tr>
<td>68</td>
<td>Sam</td>
<td>3</td>
</tr>
</tbody>
</table>

Web Log Analysis: Sessionization
Web Log Analysis: Sessionization

SELECT
  sess_id, user_id, start_time,
  no_of_events, duration
FROM Events

MATCH_RECOGNIZE

(PARTITION BY user_id ORDER BY time_stamp

MEASURES
  MATCH_NUMBER() as sess_id,
  COUNT(*) as no_of_events,
  FIRST(time_stamp) start_time,
  LAST(time_stamp) - FIRST(time_stamp) as duration

ONE ROW PER MATCH

PATTERN (b s*)

DEFINE
  s as (s.time_stamp - PREV(time_stamp) <= 10)
)

ORDER BY user_id, session_id;
## Web Log Analysis: Sessionization

<table>
<thead>
<tr>
<th>SESSI_ID</th>
<th>USER_ID</th>
<th>START_TIME</th>
<th>NO_OF_EVENTS</th>
<th>DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mary</td>
<td>1</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Mary</td>
<td>23</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Mary</td>
<td>34</td>
<td>4</td>
<td>29</td>
</tr>
<tr>
<td>1</td>
<td>Richard</td>
<td>3</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>Richard</td>
<td>54</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>Sam</td>
<td>2</td>
<td>4</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>Sam</td>
<td>43</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Sam</td>
<td>59</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>
Example #3

FINANCIAL TRACKING:
SUSPICIOUS MONEY TRANSFERS
CREATE TABLE event_log
(time DATE, userid VARCHAR2(30),
   amount NUMBER(10), event VARCHAR2(10),
   transfer_to VARCHAR2(10));

INSERT INTO event_log VALUES(TO_DATE('01-JAN-2012','DD-MON-YYYY'),'john',1000000,'deposit',NULL);
INSERT INTO event_log VALUES(TO_DATE('05-JAN-2012','DD-MON-YYYY'),'john',1200000,'deposit',NULL);
INSERT INTO event_log VALUES(TO_DATE('06-JAN-2012','DD-MON-YYYY'),'john',1000,'transfer','bob');
INSERT INTO event_log VALUES(TO_DATE('15-JAN-2012','DD-MON-YYYY'),'john',1500,'transfer','bob');
INSERT INTO event_log VALUES(TO_DATE('20-JAN-2012','DD-MON-YYYY'),'john',1500,'transfer','allen');
INSERT INTO event_log VALUES(TO_DATE('23-JAN-2012','DD-MON-YYYY'),'john',1000,'transfer','tim');
INSERT INTO event_log VALUES(TO_DATE('26-JAN-2012','DD-MON-YYYY'),'john',1000000,'transfer','tim');
INSERT INTO event_log VALUES(TO_DATE('27-JAN-2012','DD-MON-YYYY'),'john',500000,'deposit',NULL);
SELECT userid, first_t, last_t, amount
FROM (SELECT * FROM event_log WHERE event = 'transfer')
MATCH_RECOGNIZE
(
  PARTITION BY userid ORDER BY time
  MEASURES
    FIRST(x.time) first_t, y.time last_t, y.amount amount
  PATTERN ( x{3,} y )
  DEFINE
    x as (amount < 2000 AND LAST(x.time) - FIRST(x.time) < 30),
    y as (amount >= 1000000 AND y.time - LAST(x.time) < 10)
);

| USERID | FIRST_T   | LAST_T    | AMOUNT
|--------|-----------|-----------|--------
| john   | 06-JAN-12 | 26-JAN-12 | 1000000 |
Summary

- Recognizing patterns in a sequence of rows has been a capability that was widely desired, but not possible with SQL until now.

- SQL Pattern Matching combines the pattern processing flexibility of languages like Perl and Java with the declarative and analytical power of SQL.

- Reduces the need of identifying patterns on application servers and clients. Helps to follow the principle of “Move your code closer to the data”.
References & Reading Material


- Oracle Database Data Warehousing Guide 12c Release 1 (12.1)
