

ORACLE
CODE

Simplified and fast Fraud Detection

ORACLE®

[developer.oracle.com/
code](http://developer.oracle.com/code)

Live for
the
Code



**ORACLE
CODE**



About me...

Keith Laker

Senior Principal Product Management
SQL and Data Warehousing

Marathon runner, mountain biker and
coffee drinker



@ASQLBarista

[developer.oracle.com/
code](http://developer.oracle.com/code)

**Live for
the Code**



ORACLE®

Agenda

- 1 Finding patterns in your data
- 2 Looking for fraudulent transfers
- 3 Dealing with new business requirements
- 4 Combining different types of analytics: Mash-Ups
- 5 Summary



**ORACLE
CODE**

Finding Patterns in Your Data

[developer.oracle.com/
code](http://developer.oracle.com/code)



**Live for
the
Code**



ORACLE®

Finding Patterns In Your Data

Basic Concepts and Languages

- Lots of languages natively support “pattern matching”

A

- AWK

C

- COMIT

E

- Elixir (programming language)
- Elm (programming language)
- Erlang (programming language)

F

- F Sharp (programming language)

H

- Haskell (programming language)

I

- Icon (programming language)

L

- LFE (programming language)

M

- Metacompiler
- ML (programming language)

O

- OCaml

P

- Prolog

R

- Refal

- Rust (programming language)

S

- Scala (programming language)
- CSMC (programming language)
- Sed
- SNOBOL
- SPITBOL
- Swift (programming language)

T

- Tom (pattern matching language)

U

- Unicon (programming language)

W

- Wolfram Language

Source: [wikipedia](#)

Finding Patterns In Your Data

Basic Concepts and Languages

- Lots of languages natively support “pattern matching”
- And so does SQL:

• AWK

— Row-level regular expression processing

• COMMIT

— “NEW”: Pattern recognition in sequences of events

E

• Elixir (programming language)

• Elm (programming language)

• Erlang (functional programming language)

F

• “Inter-row” pattern recognition

• F# (programming language)

H

• Haskell (programming language)

I

• Icon (programming language)

M

• Metacompiler

• ML (programming language)

O

• OCaml

P

• Prolog

R

• Refal

• Rust (programming language)

S

• Scala (programming language)

• CSCM (programming language)

• Sed

• SNOBOL

• SPITBOL

• Swift (programming language)

T

• Tom (pattern matching language)

U

• Unicon (programming language)

W

• Wolfram Language

Source: [wikipedia](#)

Pattern Recognition In Sequences of Rows

SQL - a new language for pattern matching

Provide native SQL language construct

With intuitive processing

Pattern Recognition In Sequences of Rows

SQL - a new language for pattern matching

Provide native SQL language construct

- New SQL construct MATCH_RECOGNIZE
 - Added as part of the ANSI-2016 SQL standard

With intuitive processing

Pattern Recognition In Sequences of Rows

SQL - a new language for pattern matching

Provide native SQL language construct

- New SQL construct MATCH_RECOGNIZE
 - Added as part of the ANSI-2016 SQL standard

With intuitive processing

- Four logical concepts:
 - Logically partition and order the data
 - Define pattern using regular expression and pattern variables
 - Regular expression is matched against a sequence of rows
 - Each pattern variable is defined using conditions on rows and aggregates

**ORACLE
CODE**

Searching for Fraud

ORACLE®

[developer.oracle.com/
code](http://developer.oracle.com/code)

**Live for
the Code**

Demo is on <http://livesql.oracle.com>

The screenshot shows the Oracle SQL Worksheet interface with a sidebar on the left containing navigation links: Home, SQL Worksheet, My Session, Schema, Design, My Scripts, My Tutorials (which is selected), and Code Library. The main area is titled "My Tutorials" and features a search bar. Below the search bar is a grid of tutorial cards. One card, "MATCH_RECOGNIZE - Fraud demo for OracleCODE events", is highlighted with a red circle.

Tutorial Title	Description	Category
MATCH_RECOGNIZE - Empty Matches and Unmatched Rows	The aim of this tutorial is to explain the difference between the various row output options within MATCH_RECOGNIZE, specifically the EMPTY_MATCHES and UNMATCHED_ROWS clauses.	Analytics / Public
MATCH_RECOGNIZE - SKIP TO where exactly?	We use the AFTER MATCH SKIP clause to determine the precise point to resume row pattern matching after a non-empty match is found. If you don't supply an AFTER MATCH SKIP clause then	Analytics / Public
Incorrect CLASSIFIER error	This tutorial shows incorrect error message when using CLASSIFIER function with FIRST/LAST functions	SQL General / Private
MATCH_RECOGNIZE - importance of PARTITION BY and ORDER BY	The aim of this tutorial is to explain the importance of using PARTITION BY and ORDER BY to ensure the correct results are returned and explores how and when predicates are applied.	Analytics / Public
MATCH_RECOGNIZE - What to include in the MEASURES clause	This tutorial will help you understand why you might get errors such as ORA-904 "%: invalid identifier" or ORA-918 "column ambiguously defined" when you try to run a	SQL Analytics / Public
MATCH_RECOGNIZE - Using Built-In Measures	In this tutorial we will review the two built-in measures that are part of MATCH_RECOGNIZE. These measures are designed to help you understand how your data set is mapped to the pattern	Analytics / Public
Managing very long lists in 12.2 with LISTAGG	The tutorial will help explain the new 12.2 syntax for LISTAGG which provides developers with additional keywords for managing very long lists (i.e. those that exceed the current 4K)	SQL General / Private
New 12.2 Data Validation Features - CAST and VALIDATE_CONVERSION	The enhanced CAST function (along with TO_NUMBER, TO_BINARY_FLOAT, TO_BINARY_DOUBLE, TO_DATE, TO_TIMESTAMP, TO_TIMESTAMP_TZ, TO_DSINTERVAL, and	SQL General / Public

Business Problem: Finding Suspicious Money Transfers

- Suspicious money transfer pattern for an account is:
 - 3 or more small (<2K) money transfers within 30 days
 - Large transfer ($\geq 1M$) within 10 days of last small transfer
- Report
 - account
 - date of first small transfer
 - date of last large transfer
 - amount of large transfer

Business Problem: Find Suspicious Money Transfers

- Data stored in JSON logs ..

```
1  [
2   "time_id": "01-JAN-12",
3   "user_id": "John",
4   "event_id": "Deposit",
5   "trans_amount": 1000000
6 },
7 {
8   "time_id": "02-JAN-12",
9   "user_id": "John",
10  "event_id": "Transfer",
11  "trans_amount": 1000
12 },
13 {
14  "time_id": "05-JAN-12",
15  "user_id": "John",
16  "event_id": "Withdrawal",
17  "trans_amount": 2000
18 },
19 {
20  "time_id": "05-JAN-12",
21  "user_id": "John",
22  "event_id": "Transfer",
23  "trans_amount": 1500
24 }
```

Business Problem: Find Suspicious Money Transfers

- Data stored in JSON logs processed as relational table structure

```
1  {
2    "time_id": "01-JAN-12",
3    "user_id": "John",
4    "event_id": "Deposit",
5    "trans_amount": 1000000
6  },
7  {
8    "time_id": "02-JAN-12",
9    "user_id": "John",
10   "event_id": "Transfer",
11   "trans_amount": 1000
12 },
13 {
14   "time_id": "05-JAN-12",
15   "user_id": "John",
16   "event_id": "Withdrawal",
17   "trans_amount": 2000
18 },
19 {
20   "time_id": "05-JAN-12",
21   "user_id": "John",
22   "event_id": "Transfer",
23   "trans_amount": 1500
24 }
```



TIME_ID	USER_ID	EVENT	AMOUNT
1/1/2012	John	Deposit	1,000,000
1/2/2012	John	Transfer	1,000
1/5/2012	John	Withdrawal	2,000
1/10/2012	John	Transfer	1,500
1/20/2012	John	Transfer	1,200
1/25/2012	John	Deposit	1,200,000
1/27/2012	John	Transfer	1,000,000
2/2/2012	John	Deposit	500,000

Business Problem: Finding Suspicious Money Transfers

TIME_ID	USER_ID	EVENT	AMOUNT
1/1/2012	John	Deposit	1,000,000
1/2/2012	John	Transfer	1,000
1/5/2012	John	Withdrawal	2,000
1/10/2012	John	Transfer	1,500
1/20/2012	John	Transfer	1,200
1/25/2012	John	Deposit	1,200,000
1/27/2012	John	Transfer	1,000,000
2/2/2012	John	Deposit	500,000

Three small transfers within 30 days

Business Problem: Finding Suspicious Money Transfers

TIME_ID	USER_ID	EVENT	AMOUNT
1/1/2012	John	Deposit	1,000,000
1/2/2012	John	Transfer	1,000
1/5/2012	John	Withdrawal	2,000
1/10/2012	John	Transfer	1,500
1/20/2012	John	Transfer	1,200
1/25/2012	John	Deposit	1,200,000
1/27/2012	John	Transfer	1,000,000
2/2/2012	John	Deposit	500,000

Three small transfers within 30 days

Large transfer within 10 days of last small transfer



Declarative Pattern Matching using SQL

New syntax for discovering patterns using SQL: **finding suspicious money transfers...**

Input into the pattern matching process determined by **FROM** clause

MATCH_RECOGNIZE () – indicates start of pattern matching definition

```
SELECT . . .
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
```

Declarative Pattern Matching using SQL

STEP 1

Need to group and order the data to make the pattern “*visible*” within the sequence of rows...

Set the **PARTITION BY** and **ORDER BY** clauses

```
SELECT . . .
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY user_id ORDER BY time_id
```

Declarative Pattern Matching using SQL

STEP 2

Define the **MEASURES** –

List the columns to be returned

- calculated columns
- existing columns

Report account, date of first small transfer, date of last large transfer

```
SELECT . . .
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY user_id ORDER BY time_id
  MEASURES FIRST(x.time_id) AS first_t,
            LAST(y.time_id) AS last_t,
            y.amount amount
```

Declarative Pattern Matching using SQL

STEP 3

Control the output in terms of **ROWS** returned

Output **one row** each time we find a match to our pattern

```
SELECT . . .
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY userid ORDER BY time
  MEASURES FIRST(x.time_id) AS first_t,
            LAST(y.time_ID) AS last_t,
            y.amount amount
  ONE ROW PER MATCH
```

Controlling the Output: Summary vs. Detailed

- Which rows to return

- ONE ROW PER MATCH
- ALL ROWS PER MATCH
- ALL ROWS PER MATCH WITH UNMATCHED ROWS

- After match SKIP option :

- SKIP PAST LAST ROW
- SKIP TO NEXT ROW
- SKIP TO <VARIABLE>
- SKIP TO FIRST(<VARIABLE>)
- SKIP TO LAST (<VARIABLE>)

Declarative Pattern Matching using SQL

STEP 4

Describe how to search for the **VARIABLE X** –

Searching for three or more occurrences of **X**

```
SELECT . . .
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY user_id ORDER BY time_id
  MEASURES FIRST(x.time_id) AS first_t,
            LAST(y.timeid) AS last_t,
            y.amount amount
  ONE ROW PER MATCH
  PATTERN ( X{3,} )
```

Using Regular Expressions To Control Matching

- Concatenation: no operator
- 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
 - Reluctant quantifier – an additional ?

Declarative Pattern Matching using SQL

STEP 4

Describe how to search
for the **VARIABLE Y** –

Searching for only one
instance of **Y**

```
SELECT . . .
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY user_id ORDER BY time_id
  MEASURES FIRST(x.time_id) AS first_t,
            LAST(y.time_id) AS last_t,
            y.amount amount
  ONE ROW PER MATCH
  PATTERN ( x{3,} y)
```

Declarative Pattern Matching using SQL

STEP 5

DEFINE each variable in terms of what to search for:

For a match on **variable X** we are searching for small transfers of amounts less than 2K and all three transfers must occur within a 30 day window

```
SELECT . . .
FROM
  (SELECT ... FROM json_transactions WHERE event ='transfer')
MATCH_RECOGNIZE (
  PARTITION BY user_id ORDER BY time_id
  MEASURES FIRST(x.time_id) AS first_t,
    LAST(y.time_id) AS last_t,
    y.amount amount
  ONE ROW PER MATCH
  PATTERN ( X{3,} Y)
  DEFINE
    X as (amount < 2000) AND
      LAST(time_id) - FIRST(time_id) < 30,
```

Declarative Pattern Matching using SQL

STEP 5

DEFINE each variable in terms of what to search for:

For a match on **variable Y** we are searching for a large transfer of more than 10K

```
SELECT . . .
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY user_id ORDER BY time_id
  MEASURES FIRST(x.time_id) AS first_t,
    LAST(y.time_id) AS last_t,
    y.amount amount
  ONE ROW PER MATCH
  PATTERN ( X{3,} Y)
  DEFINE
    X as (amount < 2000) AND
      LAST(time_id) - FIRST(time_id) < 30,
    Y as (amount >= 1000000)
```

Declarative Pattern Matching using SQL

STEP 5

DEFINE each variable in terms of what to search for:

For a match on **variable Y** we are searching for a large transfer of more than 10K...

...and the large transfer must occur within 10 days of last small transfer

```
SELECT . . .
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY user_id ORDER BY time_id
  MEASURES FIRST(x.time_id) AS first_t,
    LAST(y.time_id) AS last_t,
    y.amount amount
  ONE ROW PER MATCH
  PATTERN ( X{3,} Y)
  DEFINE
    X as (amount < 2000) AND
      LAST(time_id) - FIRST(time_id) < 30,
    Y as (amount >= 1000000 AND
      time_id - LAST(x.time_id) < 10 ))
```

Declarative Pattern Matching using SQL

Finally list columns to return by the pattern matching process to the calling application

*SELECT * FROM... is acceptable but should really be avoided!*

```
SELECT user_id, first_t, last_t, amount
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY user_id ORDER BY time_id
  MEASURES FIRST(x.time_id) AS first_t,
            LAST(y.time_id) AS last_t,
            y.amount AS amount
  ONE ROW PER MATCH
  PATTERN ( X{3,} Y)
  DEFINE
    X as (amount < 2000) AND
        LAST(time_id) - FIRST(time_id) < 30,
    Y as (amount >= 1000000 AND
          time_id - LAST(x.time_id) < 10 ))
```

Live demonstration – Simple Fraud Detection





**ORACLE
CODE**

New Requirements

Managing new business requirements, quickly and efficiently with no application code changes

[developer.oracle.com/
code](http://developer.oracle.com/code)

**Live for
the
Code**

ORACLE®

Refining The “Suspicious Money Transfers” Pattern

Dealing With New Business Requirements

- **Additional requirement:**

- Check for transfers are to different accounts
- Total sum of small transfers must be less than 20K

```
1 [{  
2   "time_id": "01-JAN-12",  
3   "user_id": "John",  
4   "event_id": "Deposit",  
5   "trans_amount": 1000000  
6 },  
7 {  
8   "time_id": "02-JAN-12",  
9   "user_id": "John",  
10  "event_id": "Transfer",  
11  "transfer_id": "John",  
12  "trans_amount": 100  
13 },  
14 {  
15  "time_id": "02-JAN-12",  
16  "user_id": "John",  
17  "event_id": "Transfer",  
18  "transfer_id": "Bob",  
19  "trans_amount": 1000  
20 },  
21 {  
22  "time_id": "05-JAN-12",  
23  "user_id": "John",  
24  "event_id": "Withdrawal",  
25  "trans_amount": 2000  
26 }
```

Business Problem: Find Suspicious Money Transfers

- Data stored in JSON logs processed as relational table structure

```
1  [
2    {
3      "time_id": "01-JAN-12",
4      "user_id": "John",
5      "event_id": "Deposit",
6      "trans_amount": 1000000
7    },
8    {
9      "time_id": "02-JAN-12",
10     "user_id": "John",
11     "event_id": "Transfer",
12     "transfer_id": "John",
13     "trans_amount": 100
14   },
15   {
16     "time_id": "02-JAN-12",
17     "user_id": "John",
18     "event_id": "Transfer",
19     "transfer_id": "Bob",
20     "trans_amount": 1000
21   },
22   {
23     "time_id": "05-JAN-12",
24     "user_id": "John",
25     "event_id": "Withdrawal",
26     "trans_amount": 2000
27   }
```



TIME_ID	USER_ID	EVENT	TRANSFER_ID	AMOUNT
1/1/2012	John	Deposit		1,000,000
1/2/2012	John	Transfer	John	1,000
1/2/2012	John	Transfer	Bob	1,000
1/5/2012	John	Withdrawal		2,000
1/10/2012	John	Transfer	Allen	1,500
1/20/2012	John	Transfer	Tim	1,200
1/25/2012	John	Deposit		1,200,000
1/27/2012	John	Transfer	Tim	1,000,000
2/2/2012	John	Deposit		500,000

Refining The “Suspicious Money Transfers” Pattern

TIME_ID	USER_ID	EVENT	TRANSFER_TO	AMOUNT
1/1/2012	John	Deposit	-	1,000,000
1/2/2012	John	Transfer	John	100
1/2/2012	John	Transfer	Bob	1,000
1/5/2012	John	Withdrawal	-	2,000
1/10/2012	John	Transfer	Allen	1,500
1/20/2012	John	Transfer	Tim	1,200
1/25/2012	John	Deposit		1,200,000
1/27/2012	John	Transfer	Tim	1,000,000
2/2/2012	John	Deposit	-	500,000

Refining The “Suspicious Money Transfers” Pattern

TIME_ID	USER_ID	EVENT	TRANSFER_TO	AMOUNT
1/1/2012	John	Deposit	-	1,000,000
1/2/2012	John	Transfer	John	100
1/2/2012	John	Transfer	Bob	1,000
1/5/2012	John	Withdrawal	-	2,000
1/10/2012	John	Transfer	Allen	1,500
1/20/2012	John	Transfer	Tim	1,200
1/25/2012	John	Deposit		1,200,000
1/27/2012	John	Transfer	Tim	1,000,000
2/2/2012	John	Deposit	-	500,000

Refining The “Suspicious Money Transfers” Pattern

TIME_ID	USER_ID	EVENT	TRANSFER_TO	AMOUNT
1/1/2012	John	Deposit	-	1,000,000
1/2/2012	John	Transfer	John	100
1/2/2012	John	Transfer	Bob	1,000
1/5/2012	John	Withdrawal	-	2,000
1/10/2012	John	Transfer	Allen	1,500
1/20/2012	John	Transfer	Tim	1,200
1/25/2012	John	Deposit		1,200,000
1/27/2012	John	Transfer	Tim	1,000,000
2/2/2012	John	Deposit	-	500,000

Three small transfers within
30 days
**to different acct and total
sum < 20K**

SQL Pattern Matching in action

Modify the pattern variables
DEFINE

- Check the transfer account

```
SELECT user_id, first_t, last_t, amount
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY userid ORDER BY time
  MEASURES FIRST(x.time_id) AS first_t,
            LAST(y.time_id) AS last_t,
            y.amount AS amount
  ONE ROW PER MATCH
  PATTERN (X{3,} Y)
  DEFINE
    X as (amount < 2000) AND
        LAST(time_id) - FIRST(time_id) < 30
```

Refining The “Suspicious Money Transfers” Pattern

TIME_ID	USER_ID	EVENT	TRANSFER_TO	AMOUNT	
1/1/2012	John	Deposit	-	1,000,000	
1/2/2012	John	Transfer	John	100	
1/2/2012	John	Transfer	Bob	1,000	Three small transfers within 30 days to different acct and total sum < 20K
1/5/2012	John	Withdrawal	-	2,000	
1/10/2012	John	Transfer	Allen	1,500	
1/20/2012	John	Transfer	Tim	1,200	
1/25/2012	John	Deposit		1,200,000	
1/27/2012	John	Transfer	Tim	1,000,000	Large transfer within 10 days of last small transfer
2/2/2012	John	Deposit	-	500,000	

SQL Pattern Matching in action

Modify the pattern variables
DEFINE

- Check the transfer account

```
SELECT user_id, first_t, last_t, amount
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY userid ORDER BY time
  MEASURES FIRST(x.time_id) AS first_t,
            LAST(y.time_id) AS last_t,
            y.amount AS amount
  ONE ROW PER MATCH
  PATTERN (X{3,} Y)
  DEFINE
    X as (amount < 2000) AND
        PREV(transfer_id) <> transfer_id AND
        LAST(time_id) - FIRST(time_id) < 30
```

SQL Pattern Matching in action

Modify the pattern variables
DEFINE

- Check the total of the small transfers is less than 20K

```
SELECT user_id, first_t, last_t, amount
FROM
  (SELECT ... FROM json_transactions WHERE event = 'transfer')
MATCH_RECOGNIZE (
  PARTITION BY userid ORDER BY time
  MEASURES FIRST(x.time_id) AS first_t,
             LAST(y.time_id) AS last_t,
             y.amount amount
  ONE ROW PER MATCH
  PATTERN (X{3,} Y)
  DEFINE
    X as (amount < 2000) AND
         PREV(transfer_id) <> transfer_id AND
         LAST(time_id) - FIRST(time_id) < 30,
    Y as (amount >= 1000000 AND
          time_id - LAST(x.time_id) < 10 AND
          SUM(x.amount) < 20000 );
```

Live Demonstration – New Requirements





**ORACLE
CODE**

Combining Analytics

Why pattern matching with SQL is so powerful:

ANALYTICAL MASH-UPS



ORACLE®

[developer.oracle.com/
code](http://developer.oracle.com/code)

**Live for
the
Code**

Fraud Detection Mash-Up Using SQL

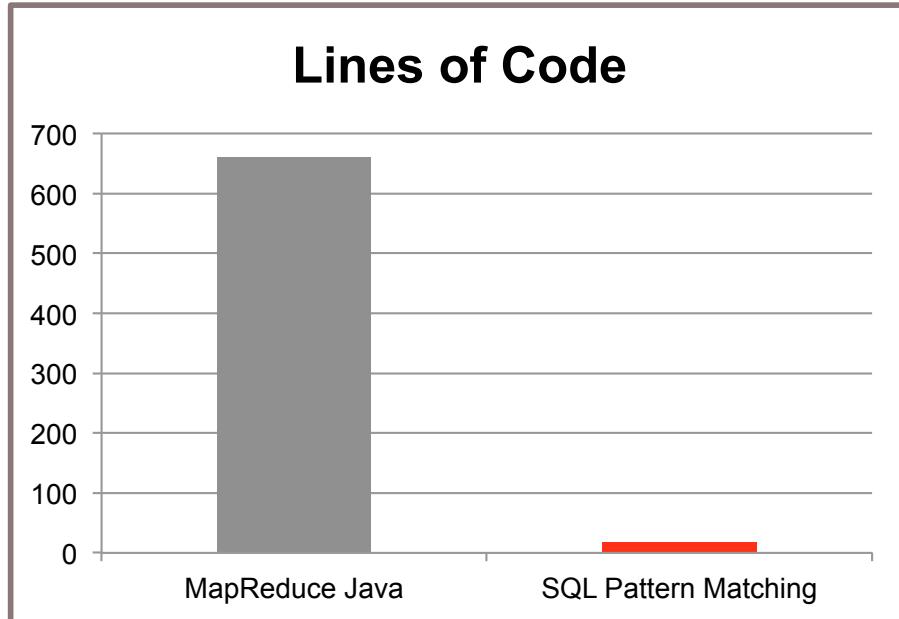
Key benefits of SQL – easy to combine different types of analytics

- + Real-time streaming data
 - + Real-time geo-coding
 - + SQL Pattern Matching
 - + Spatial Analytics
- = Real-time fraud detection



SQL Pattern Matching vs. Java

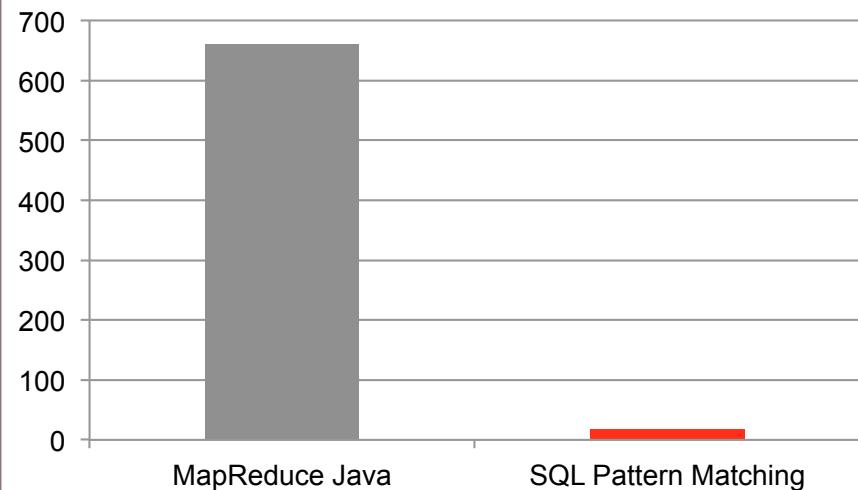
Fraud Detection: Fewer lines of code required....



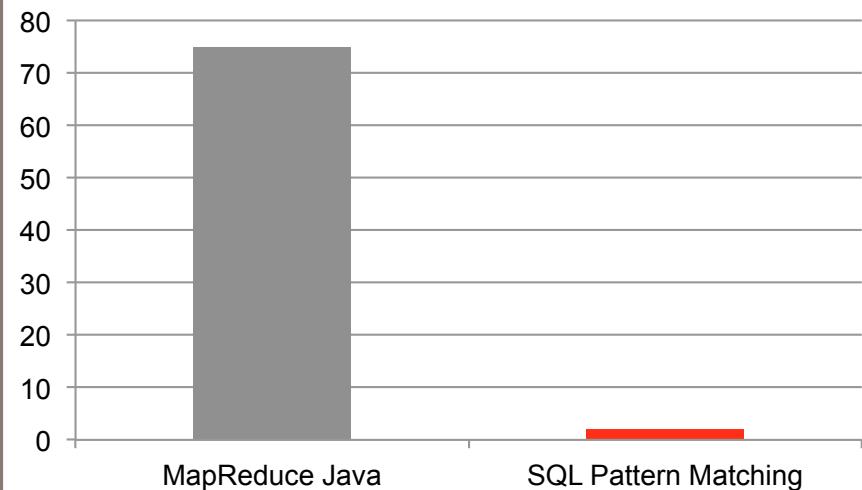
SQL Pattern Matching vs. Java

Fraud Detection: Fewer lines of code required....

Lines of Code



Execution Time (secs)





**ORACLE
CODE**

Summary



ORACLE®



[developer.oracle.com/
code](http://developer.oracle.com/code)

**Live for
the Code**

Summary – You Can Now....

- Construct a MATCH_RECOGNIZE statement
 - Build search criteria using pattern variables
 - Check if your pattern matching *really* is working correctly
 - Use built-in measures to get additional information
 - Understand the power and value of SQL pattern matching
-
- Go and use SQL Pattern Matching to your advantage!

SQL Pattern Matching

Where to get more information

- Analytical SQL Home Page on OTN with links to:
 - Training + Oracle By Example
 - Podcasts for pattern matching
 - Whitepapers
 - Sample scripts and simple tutorials for pattern matching on liveSQL
- Data Warehouse and SQL Analytics blog
 - <http://oracle-big-data.blogspot.co.uk/>



ORACLE®