ORACLE

SQL - the best development language for Big Data?

Exploring the Analytical Power of SQL in Oracle Database 12c

ORACLE OPEN WORLD

HARDWARE AND SOFTWARE ENGINEERED TO WORK TOGETHER

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract.

It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.

ORACLE

ORACLE OPEN WORLD

Keith Laker
Senior Principal Product Manager

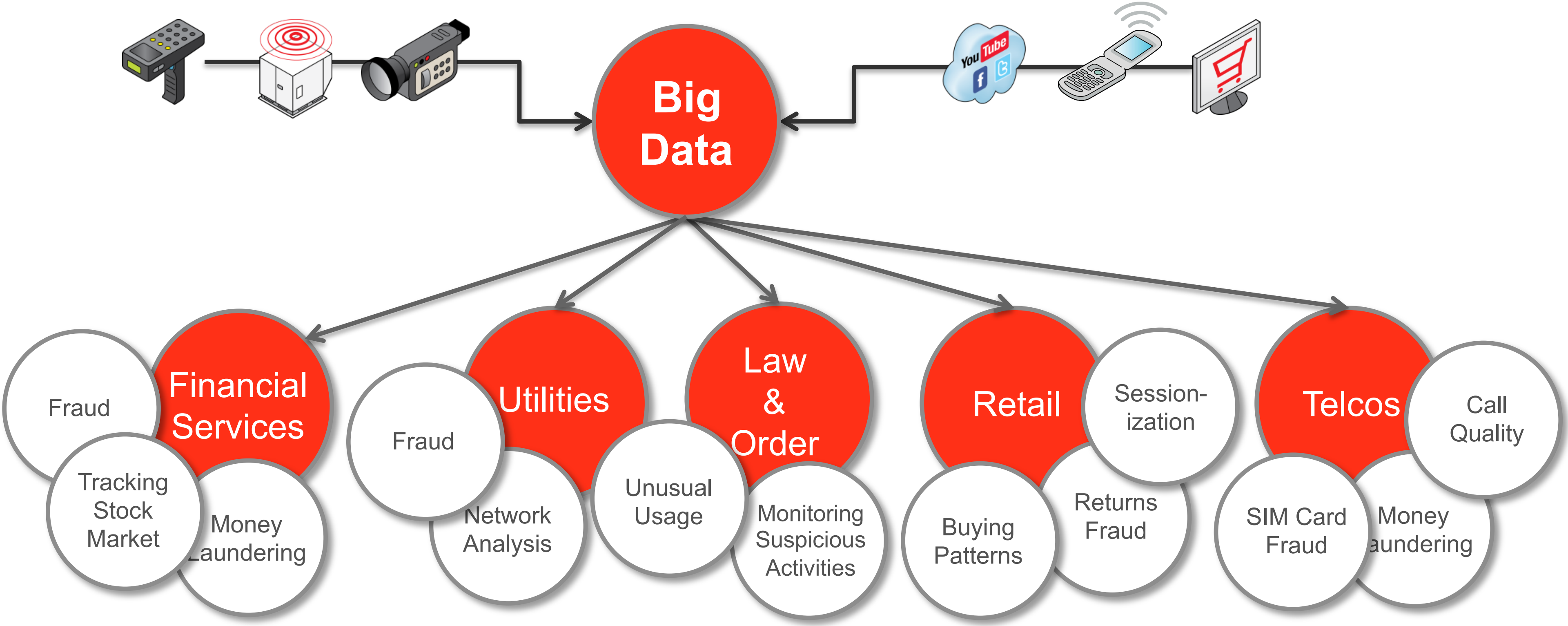
Andrew Witkowski
Architect

Sankar Subramanian
Senior Director of Development

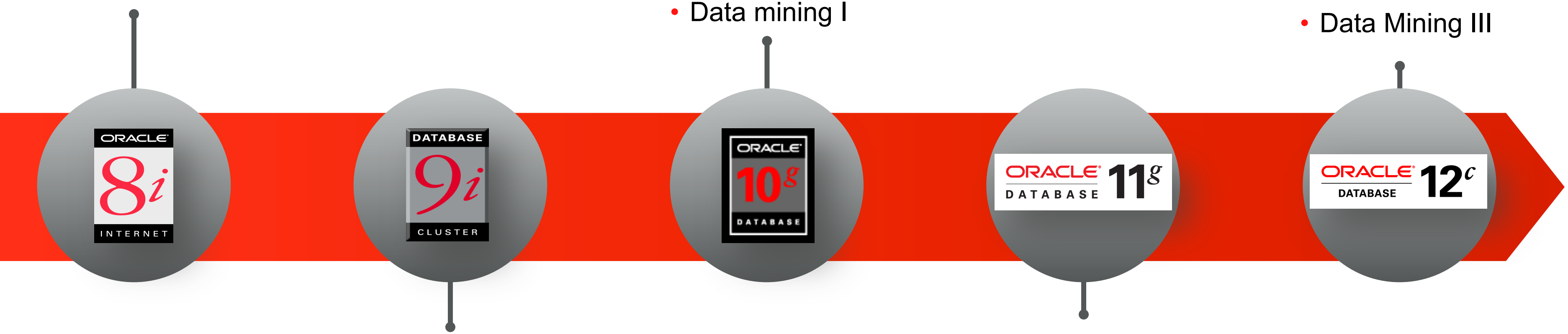
HARDWARE
AND SOFTWARE
ENGINEERED
TO WORK
TOGETHER

Finding Patterns in Big Data

Typical use cases in today's world of fast exploration of big data



The On-Going Evolution of SQL



- Introduction of Window functions

- Enhanced Window functions (percentile,etc)
- Rollup, grouping sets, cube

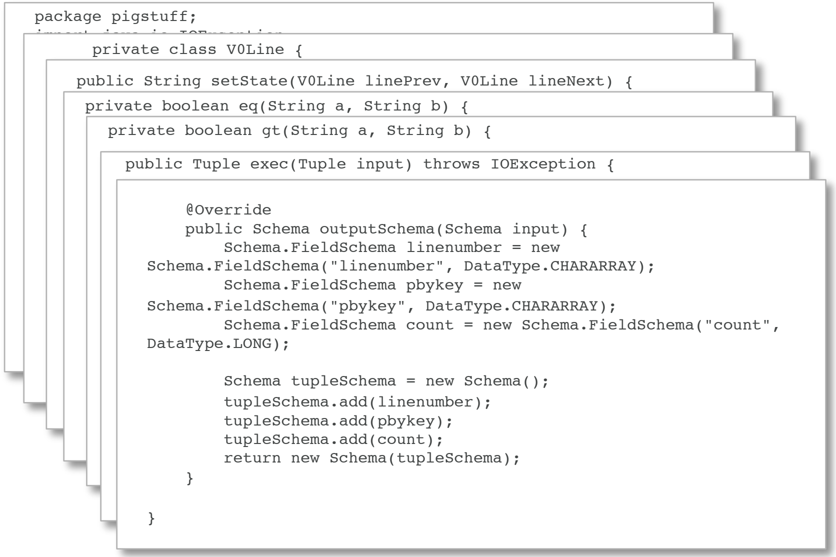
- Statistical functions
- SQL model clause
- Partition Outer Join
- Data mining I

- Data mining II
- SQL Pivot
- Recursive WITH
- ListAgg, Nth value window

- **Pattern matching**
- **Top N clause**
- Identity Columns
- Column Defaults
- Data Mining III

Pattern Matching with SQL Analytics

Java vs. SQL: Stock Markets - Searching for ‘W’ Patterns in Trade Data



```
package pigstuff;
import java.io.IOException
    private class V0Line {
  public String setState(V0Line linePrev, V0Line lineNext) {
  private boolean eq(String a, String b) {
    private boolean gt(String a, String b) {
    public Tuple exec(Tuple input) throws IOException {

       @Override
       public Schema outputSchema(Schema input) {
           Schema.FieldSchema linenumber = new
   Schema.FieldSchema("linenumber", DataType.CHARARRAY);
           Schema.FieldSchema pbykey = new
   Schema.FieldSchema("pbykey", DataType.CHARARRAY);
           Schema.FieldSchema count = new Schema.FieldSchema("count",
   DataType.LONG);

           Schema tupleSchema = new Schema();
           tupleSchema.add(linenumber);
           tupleSchema.add(pbykey);
           tupleSchema.add(count);
           return new Schema(tupleSchema);
       }

   }
```

```
SELECT first_x, last_z
FROM ticker MATCH_RECOGNIZE (
       PARTITION BY name ORDER BY time
       MEASURES FIRST(x.time) AS first_x,
                LAST(z.time)  AS last_z
       ONE ROW PER MATCH
       PATTERN (X+ Y+ W+ Z+)
       DEFINE X AS (price < PREV(price)),
              Y AS (price > PREV(price)),
              W AS (price < PREV(price)),
              Z AS (price > PREV(price)  AND
                  z.time - FIRST(x.time) <= 7 ))
```

250+ Lines of Java and PIG

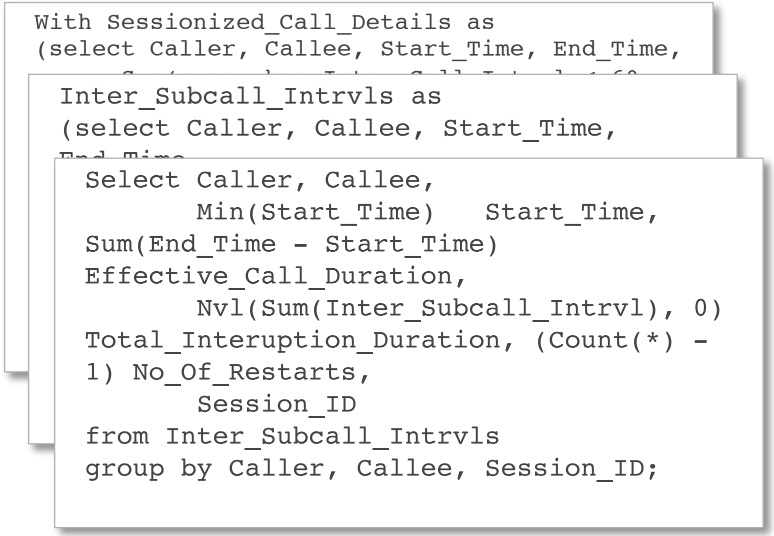
12 Lines of SQL

SQL - 20x less code, 5x faster

Pattern Matching with SQL Analytics

11g vs. 12c: Call Quality Analysis - Looking for Dropped Calls

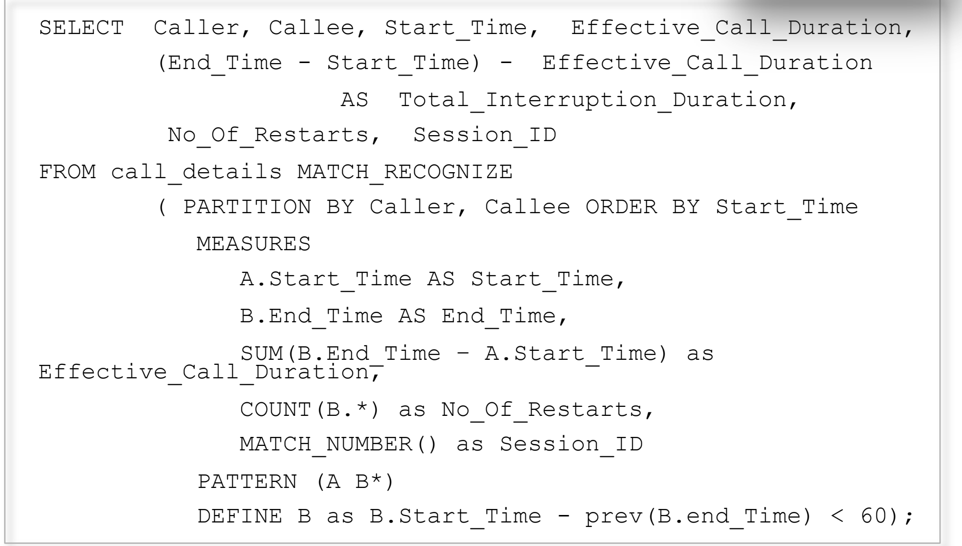
ORACLE DATABASE 12c



```
With Sessionized_Call_Details as
(select Caller, Callee, Start_Time, End_Time,
```

```
Inter_Subcall_Intrvls as
(select Caller, Callee, Start_Time,
End_Time
```

```
Select Caller, Callee,
       Min(Start_Time)   Start_Time,
Sum(End_Time - Start_Time)
Effective_Call_Duration,
       Nvl(Sum(Inter_Subcall_Intrvl), 0)
Total_Interuption_Duration, (Count(*) -
1) No_Of_Restarts,
       Session_ID
from Inter_Subcall_Intrvls
group by Caller, Callee, Session_ID;
```



```
SELECT  Caller, Callee, Start_Time,  Effective_Call_Duration,
        (End_Time - Start_Time) -  Effective_Call_Duration
                    AS  Total_Interruption_Duration,
         No_Of_Restarts,  Session_ID
FROM call_details MATCH_RECOGNIZE
        ( PARTITION BY Caller, Callee ORDER BY Start_Time
          MEASURES
             A.Start_Time AS Start_Time,
             B.End_Time AS End_Time,
             SUM(B.End_Time - A.Start_Time) as
Effective_Call_Duration,
             COUNT(B.*) as No_Of_Restarts,
             MATCH_NUMBER() as Session_ID
          PATTERN (A B*)
          DEFINE B as B.Start_Time - prev(B.end_Time) < 60);
```

24+ lines of multi-select, sophisticated SQL

14 lines of simple SQL

50% less code - easier to understand, test, deploy and manage

ORACLE

SQL Pattern Matching

Key Concepts

Pattern Recognition In Sequences of Rows

“SQL Pattern Matching” - Concept

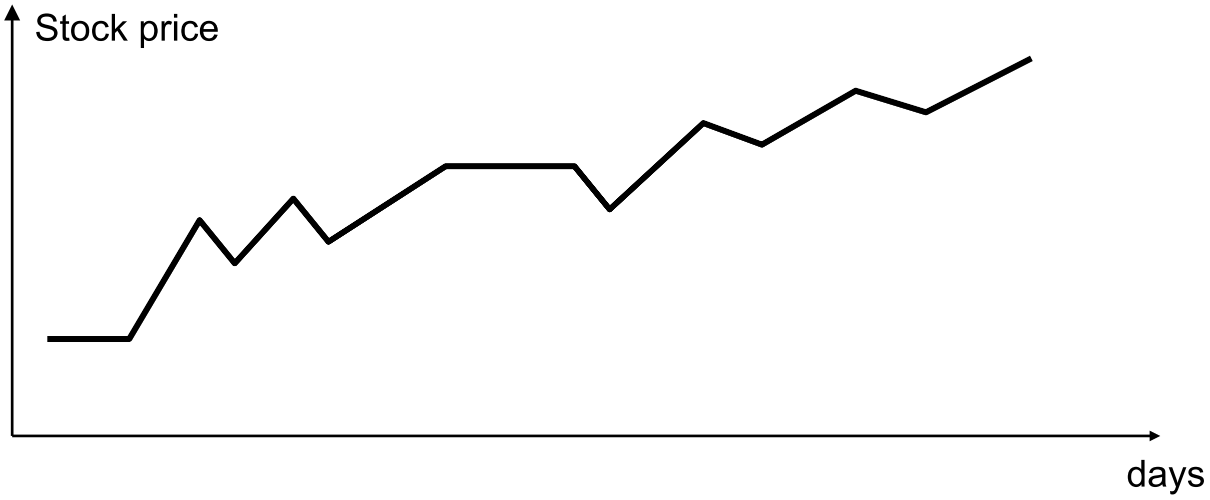
- Recognize patterns in sequences of events using SQL
  - Sequence is a stream of rows
  - Event equals a row in a stream
- New SQL construct MATCH_RECOGNIZE
  - Logically partition and order the data
    - ORDER BY mandatory (optional PARTITION BY)
  - Pattern defined using regular expression using variables
  - Regular expression is matched against a sequence of rows
  - Each pattern variable is defined using conditions on rows and aggregates

SQL Pattern Matching in action

Example: Find a double bottom pattern (W-shape) in ticker stream

Find a W-shape pattern in a ticker stream:

- Output the **beginning** and **ending** date of the pattern
- Calculate **average price** in the second ascent
- Find only patterns that **lasted less than a week**

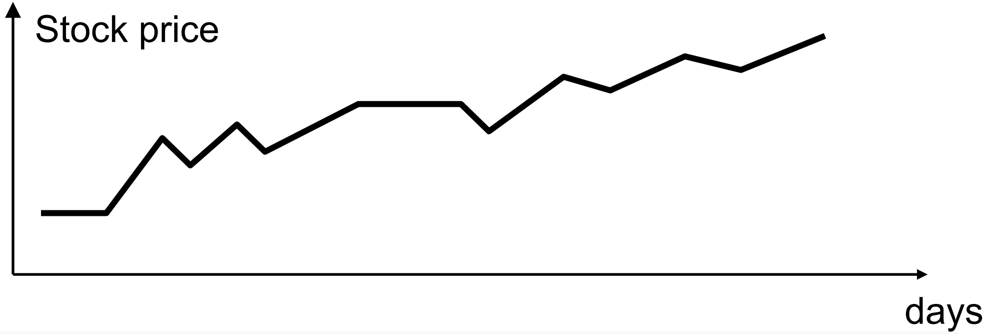


SQL Pattern Matching in action

Example: Find W-Shape

New syntax for discovering patterns using SQL:

MATCH_RECOGNIZE ( )



```
SELECT . . .
FROM ticker MATCH_RECOGNIZE (
      . . .
)
```

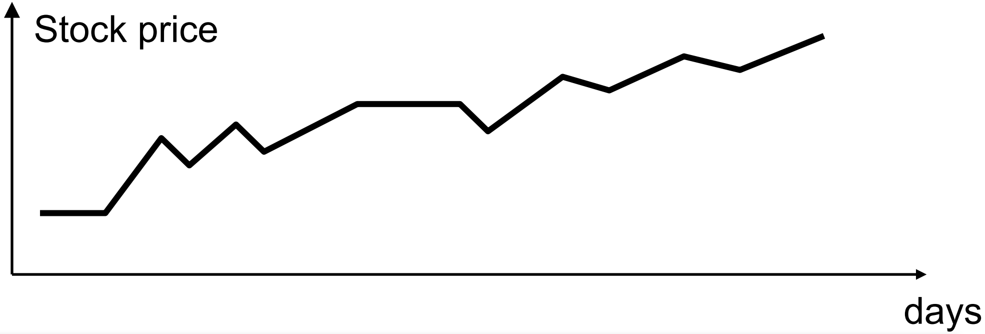
SQL Pattern Matching in action

Example: Find W-Shape

Find a W-shape pattern in a ticker stream:

- Set the PARTITION BY and ORDER BY clauses

We will continue to look at the black stock only from now on



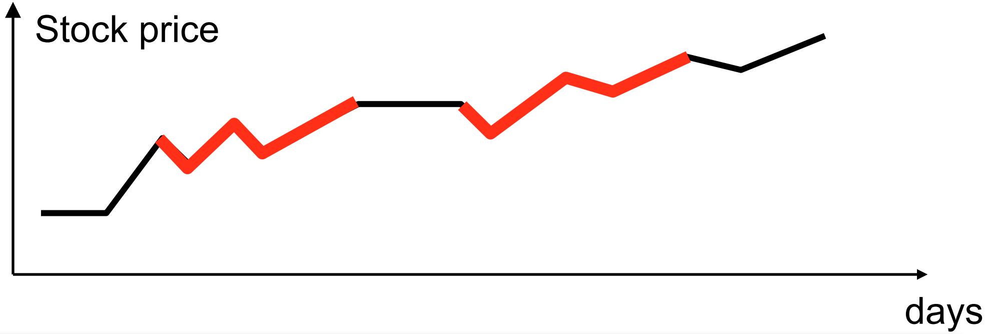
```
SELECT …
FROM ticker MATCH_RECOGNIZE (
       PARTITION BY name ORDER BY time
```

SQL Pattern Matching in action

Example: Find W-Shape

Find a W-shape pattern in a ticker stream:

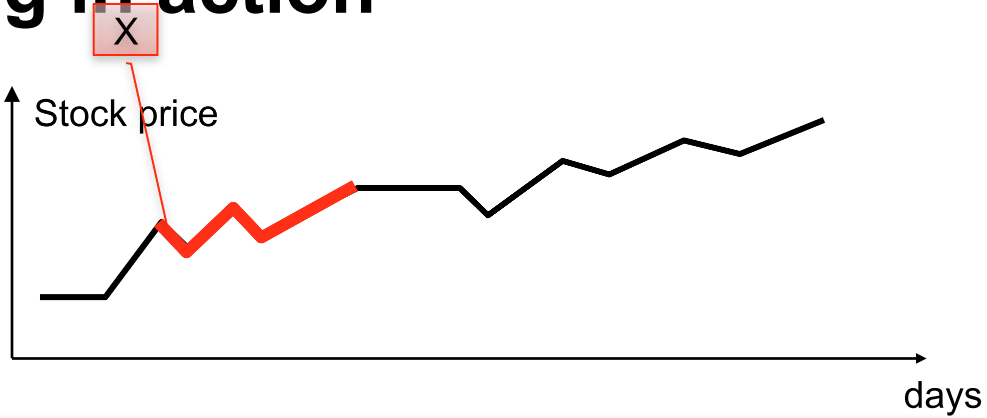
- Define the **pattern** – the “W-shape”



```
SELECT …
FROM ticker MATCH_RECOGNIZE (
       PARTITION BY name ORDER BY time



       PATTERN (X+ Y+ W+ Z+)
```

SQL Pattern Matching in action

Example: Find W-Shape



Find a W-shape pattern in a ticker stream:

- Define the **pattern** – the first down part of the "W-shape"

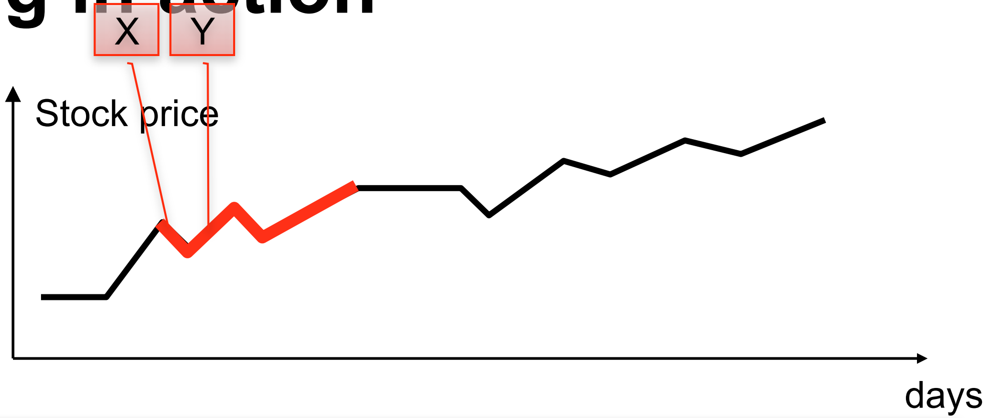
```
SELECT …
FROM ticker MATCH_RECOGNIZE (
       PARTITION BY name ORDER BY time



       PATTERN (X+ Y+ W+ Z+)
       DEFINE X AS (price < PREV(price)),
```

SQL Pattern Matching in action

Example: Find W-Shape

Find a W-shape pattern in a ticker stream:

- Define the **pattern** – the first up part of “W-shape”



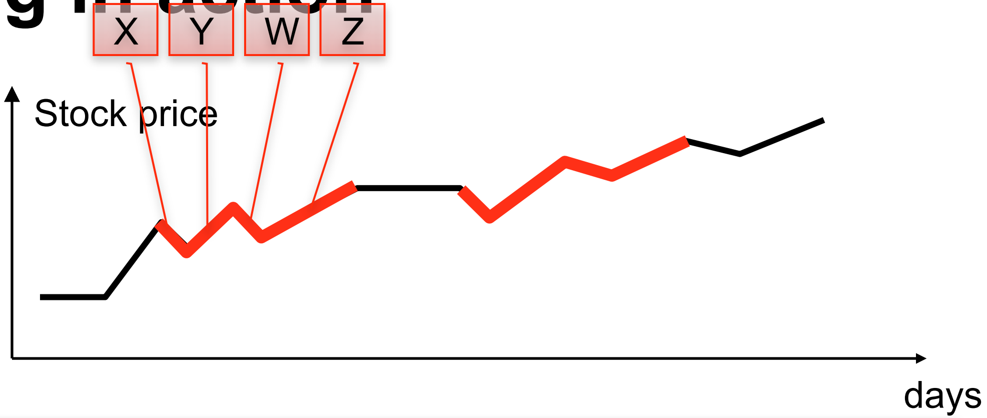
```
SELECT …
FROM ticker MATCH_RECOGNIZE (
       PARTITION BY name ORDER BY time



       PATTERN (X+ Y+ W+ Z+)
       DEFINE  X AS (price < PREV(price)),
               Y AS (price > PREV(price)),
```

SQL Pattern Matching in action

Example: Find W-Shape

Find a W-shape pattern in a ticker stream:

- Define the **pattern** – the second down (w) and the second up(z) of the “W-shape”



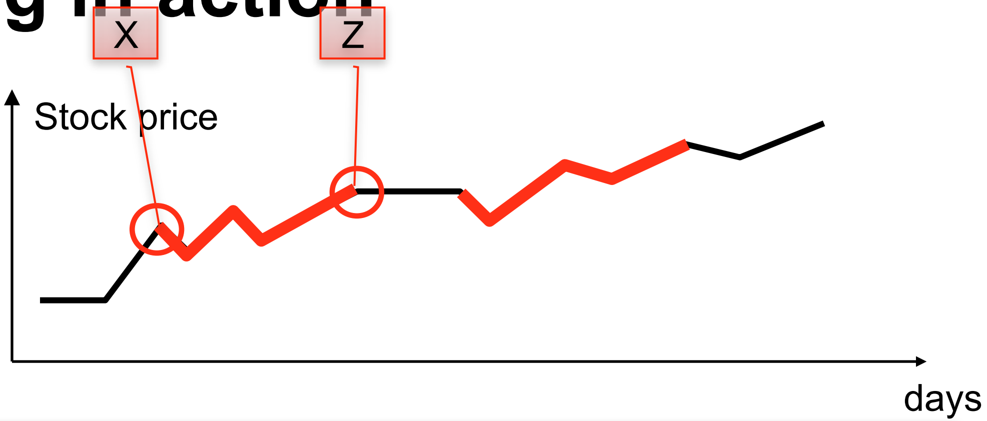
```
SELECT …
FROM ticker MATCH_RECOGNIZE (
       PARTITION BY name ORDER BY time



       PATTERN (X+ Y+ W+ Z+)
       DEFINE X AS (price < PREV(price)),
              Y AS (price > PREV(price)),
              W AS (price < PREV(price)),
              Z AS (price > PREV(price)))
```

SQL Pattern Matching in action

Example: Find W-Shape

Find a W-shape pattern in a ticker stream:

- Define the measures to output once a pattern is matched:
  - **FIRST: beginning date**
  - **LAST: ending date**



```
SELECT …
FROM ticker MATCH_RECOGNIZE (
       PARTITION BY name ORDER BY time
       MEASURES FIRST(x.time) AS first_x,
                LAST(z.time)  AS last_z

       PATTERN (X+ Y+ W+ Z+)
       DEFINE X AS (price < PREV(price)),
              Y AS (price > PREV(price)),
              W AS (price < PREV(price)),
              Z AS (price > PREV(price))
```

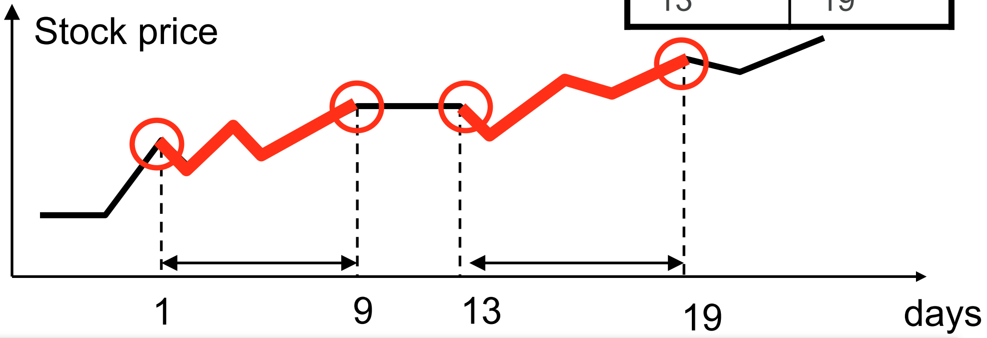
SQL Pattern Matching in action

Example: Find W-Shape

First_x	Last_z
1	9
13	19

Find a W-shape pattern in a ticker stream:

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

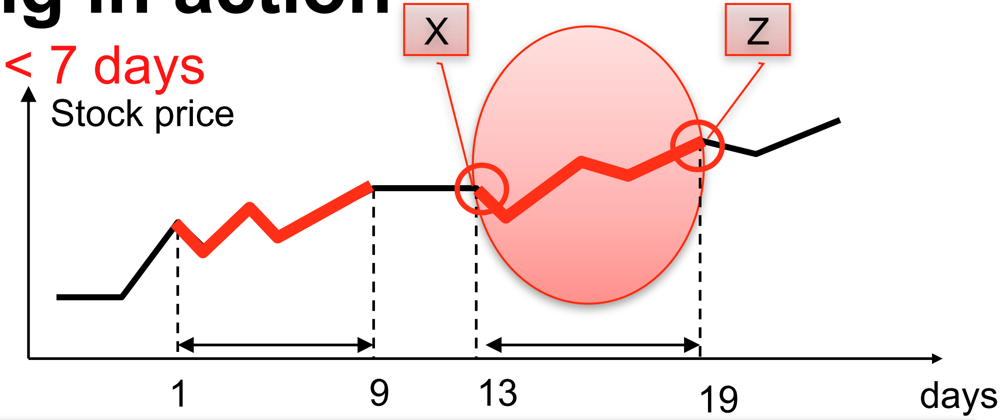


```
SELECT first_x, last_z
FROM ticker MATCH_RECOGNIZE (
       PARTITION BY name ORDER BY time
       MEASURES FIRST(x.time) AS first_x,
                LAST(z.time)  AS last_z
       ONE ROW PER MATCH
       PATTERN (X+ Y+ W+ Z+)
       DEFINE X AS (price < PREV(price)),
              Y AS (price > PREV(price)),
              W AS (price < PREV(price)),
              Z AS (price > PREV(price)))
```

SQL Pattern Matching in action

Example: Find W-Shape lasts < 7 days

Can refer to previous variables



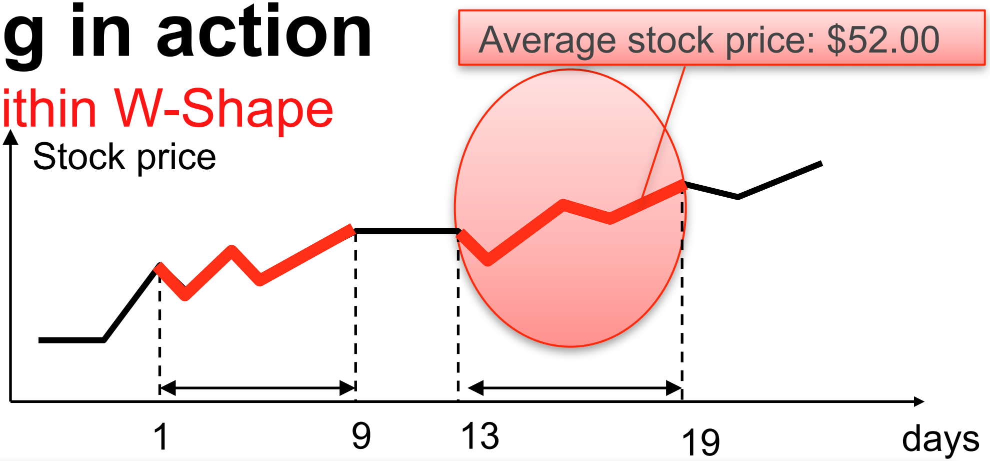
Find a W-shape pattern in a ticker stream:

- Extend the pattern to find W-shapes that **lasted less than a week**

```
SELECT first_x, last_z
FROM ticker MATCH_RECOGNIZE (
       PARTITION BY name ORDER BY time
       MEASURES FIRST(x.time) AS first_x,
                LAST(z.time)  AS last_z
       ONE ROW PER MATCH
       PATTERN (X+ Y+ W+ Z+)
       DEFINE X AS (price < PREV(price)),
              Y AS (price > PREV(price)),
              W AS (price < PREV(price)),
              Z AS (price > PREV(price)  AND
                    z.time - FIRST(x.time) <= 7 ))
```

SQL Pattern Matching in action

Example: Find average price within W-Shape



Find a W-shape pattern in a ticker stream:

- Calculate **average price** in the second ascent

```
SELECT first_x, last_z, avg_price
FROM ticker MATCH_RECOGNIZE (
       PARTITION BY name ORDER BY time
       MEASURES FIRST(x.time) AS first_x,
                LAST(z.time)  AS last_z,
                AVG(z.price)  AS avg_price
       ONE ROW PER MATCH
       PATTERN (X+ Y+ W+ Z+)
       DEFINE X AS (price < PREV(price)),
              Y AS (price > PREV(price)),
              W AS (price < PREV(price)),
              Z AS (price > PREV(price) AND
                    z.time - FIRST(x.time) <= 7 ))))
```

SQL Pattern Matching in action

Example: Sessionization for user log

- Define a session as a sequence of one or more events with the same partition key where the inter-timestamp gap is less than a specified threshold
- Example “user log analysis”
  - Partition key: User ID, Inter-timestamp gap: 10 (seconds)
  - Detect the sessions
  - Assign a within-partition (per user) surrogate Session_ID to each session
  - Annotate each input tuple with its Session_ID

SQL Pattern Matching in action

Example Sessionization for user log

TIME	USER ID
1	Mary
2	Sam
11	Mary
12	Sam
22	Sam
23	Mary
32	Sam
34	Mary
43	Sam
44	Mary
47	Sam
48	Sam
53	Mary
59	Sam
60	Sam
63	Mary
68	Sam

Identify sessions

TIME	USER ID
1	Mary
11	Mary
23	Mary
34	Mary
44	Mary
53	Mary
63	Mary
2	Sam
12	Sam
22	Sam
32	Sam
43	Sam
47	Sam
48	Sam
59	Sam
60	Sam
68	Sam

Number Sessions per user

TIME	USER ID	SESSION
1	Mary	1
11	Mary	1
23	Mary	2
34	Mary	3
44	Mary	3
53	Mary	3
63	Mary	3
2	Sam	1
12	Sam	1
22	Sam	1
32	Sam	1
43	Sam	2
47	Sam	2
48	Sam	2
59	Sam	3
60	Sam	3
68	Sam	3

SQL Pattern Matching in action

Example Sessionization for user log: MATCH_RECOGNIZE

```
.   .   .
FROM Events MATCH_RECOGNIZE
      (PARTITION BY user_ID ORDER BY time
       MEASURES match_number() as session_id
       ALL ROWS PER MATCH
       PATTERN (b s*)
       DEFINE
              s as (s.time - prev(s.time) <= 10)
       );
```

SQL Pattern Matching in action

Example Sessionization – Aggregation of sessionized data

- Primitive sessionization only a foundation for analysis
  - Mandatory to logically identify related events and group them
- Aggregation for the first data insight
  - How many “events” happened within an individual session?
  - What was the total duration of an individual session?

SQL Pattern Matching in action

Example Sessionization – **Aggregation** of sessionized data

TIME	USER ID	SESSION
1	Mary	1
11	Mary	1
23	Mary	2
34	Mary	3
44	Mary	3
53	Mary	3
63	Mary	3
2	Sam	1
12	Sam	1
22	Sam	1
32	Sam	1
43	Sam	2
47	Sam	2
48	Sam	2
59	Sam	3
60	Sam	3
68	Sam	3

Aggregate sessions per user

TIME	SESSION_ID	START_TIME	NUM EVENTS	DURATION
Mary	1	1	2	10
Mary	2	23	1	0
Mary	3	34	4	29
Sam	1	2	4	30
Sam	2	43	3	5
Sam	3	59	3	9

SQL Pattern Matching

Example Sessionization – Aggregation: **ONE ROW PER MATCH**

```
. . .
FROM Events MATCH_RECOGNIZE
       ( PARTITION BY user_ID ORDER BY time ONE ROW PER MATCH
         MEASURES match_number() session_id,
                  count(*) as no_of_events,
                  first(time) start_time,
                  last(time) - first(time) duration
         PATTERN (b s*)
         DEFINE
                  s as (s.time - prev(time) <= 10)
       )
ORDER BY user_id, session_id;
```

Native Top N Support

Native Support for TOP-N Queries

Natively identify top N in SQL

Significantly simplifies code development

ANSI SQL:2008

"Who are the top 5 money makers in my enterprise?"

```
SELECT empno, ename, deptno
FROM emp
ORDER BY sal, comm FETCH FIRST 5 ROWS ONLY;
```

versus

```
SELECT empno, ename, deptno
FROM (SELECT empno, ename, deptno, sal, comm,
            row_number() OVER (ORDER BY sal,comm) rn
      FROM emp
      )
WHERE rn <=5
ORDER BY sal, comm;
```

Native Support for TOP-N Queries

New offset and fetch_first clause

- ANSI 2008/2011 compliant with some additional extensions
- Specify offset and number or percentage of rows to return
- Provisions to return additional rows with the same sort key as the last row (WITH TIES option)
- Syntax:

```
OFFSET <offset> [ROW | ROWS]
FETCH [FIRST | NEXT]
 [<rowcount> | <percent> PERCENT] [ROW | ROWS]
 [ONLY | WITH TIES]
```

Summary

New Database 12c SQL Analytics

- ANSI compliant features with some additional extensions
- Common syntax reduces learning curve
- Comprehensive support for SQL based pattern matching
  - Supports a wide range of use cases
  - Simplifies application development
  - Simplifies existing SQL code
- New TOP-N feature
  - Simplifies existing SQL code

SQL - the best development language for Big Data?

Yes, because SQL is….

SIMPLER

FASTER

RICHER

ORACLE
OPEN
WORLD

HARDWARE
AND SOFTWARE
ENGINEERED
TO WORK
TOGETHER

Other sessions…

Session	Date	Location
Pattern Matching Hands-on Lab	Tues - 12:00pm	Marriot Salon 3-4
Top Tips for Mastering Oracle Partitioning	Tues - 3:45pm	Moscone South 103
Oracle Optimizer Boot Camp	Tues - 5:15pm	Moscone South 102
In-Database MapReduce using SQL	Wed - 10:15am	Marriot Salon 7
Programming with Big Data Connectors	Wed – 3:30pm	Marriot Salon 7
Data Warehouse & Big Data – Customer panel	Wed – 3:30pm	Moscone South 300
Your Data is talking to you – Customer panel	Wed – 5:00pm	Moscone South 300

Where to get more information

SQL PATTERN MATCHING IN ORACLE DATABASE 12c

Patterns everywhere – Find them fast!

ORACLE DATABASE 12c

ORACLE MULTIMEDIA BOOK

- SQL Analytics Home Page on OTN
  - http://www.oracle.com/technetwork/database/bi-datawarehousing/sql-analytics-index-1984365.html
  - Oracle By Example – Pattern matching
  - Podcasts for pattern matching and SQL analytics
  - Data Sheet
  - Whitepapers
    - Patterns Everywhere - Find then fast!
    - Patterns Everywhere - Find then fast! (Apple iBook)
- Data Warehouse and SQL Analytics blog
  - http://oracle-big-data.blogspot.co.uk/

ORACLE®

ORACLE OPEN WORLD

THANK YOU FOR JOINING US TODAY

ENJOY OPENWORLD

HARDWARE AND SOFTWARE ENGINEERED TO WORK TOGETHER