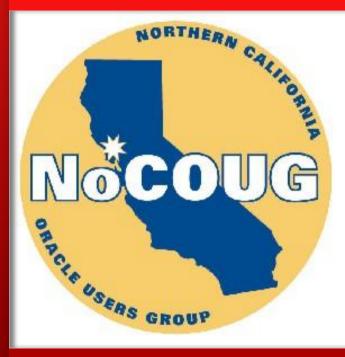


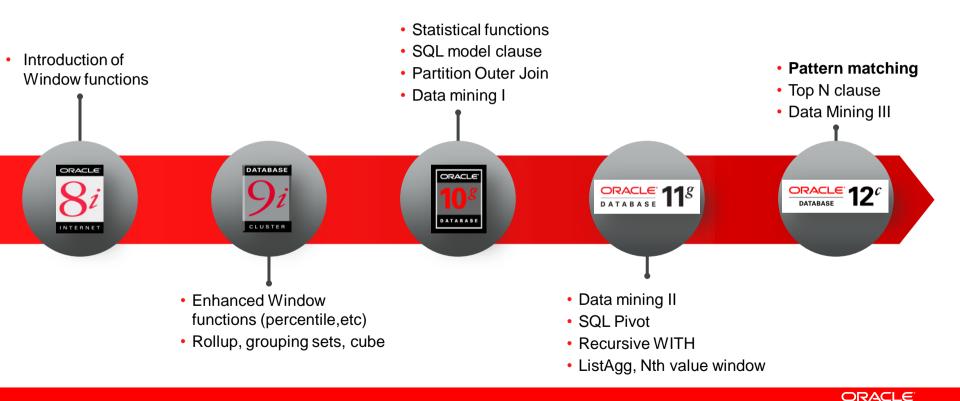
# SQL - the best analysis language for Big Data!

**NoCOUG Winter Conference 2014** 

Hermann Bär, hermann.baer@oracle.com Data Warehousing Product Management, Oracle



### The On-Going Evolution of SQL

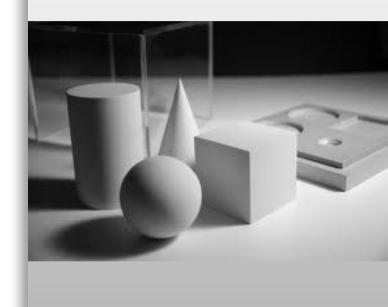


#### SQL for Analysis and Reporting Benefits

- Dramatically enhanced analysis capabilities with SQL
  - Native support, e.g. OBI EE
  - Embedding into SQL views
- Simplified development
  - Investment protection through ANSI standard compliance

- Increased performance
  - New language constructs enable more efficient plans
  - Internal optimizations

#### **Fundamental Concepts**





## **Key Concepts**

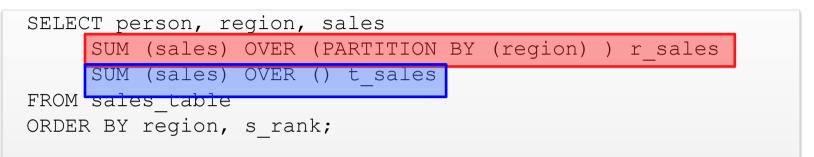
Same for all functions - Unified SQL

#### Partitions

- Groupings of rows within a query result set
- Orderings
  - Rows can be ordered within a partition
- Windows (logical or physical)
  - A moving group of rows within a partition
  - Defines the range of an aggregate calculation
- Current Row

### **Reporting Aggregates**

#### Compare total sales of regions with total sales



PERSON	REGION	SALES	R_SALES	T_SALES
Adams	East	200	530	1130
Connor	East	180	530	1130
Baker	East	150	530	1130
Donner	West	300	600	1130
Edward	West	200	600	1130
Witkowski	West	100	600	1130

### Lag/Lead Functions

How does sales compare versus this month last year?

	timekey, sales,
	LAG(sales,12)OVER(ORDER BY timekey) as sales_last_year,
	(sales - sales_last_year) as sales_change
FROM s	ales;

TIMEKEY	SALES	SALES LAST YEAR	SALES CHANGE
2009-01	1100	-	-
2010-01	2000	1100	900
2010-02	1800	1200	600
2011-01	1900	2000	-100

**SQL for Analysis and Reporting** 

#### Simplified SQL development

#### Without analytical function

```
select avg(b.event datetime - a.prior event datetime) as avg wait
from
  (select
    a.*
     , rownum as row number
   from
     (select
       order id
       , event datetime
            as prior event datetime
        , new event cd as prior event cd
      from order pipeline events
      where
       warehouse id = 'RNO1'
       and event datetime > sysdate - 2
      order bv
       order id
       , prior event datetime
        , prior event cd
     ) a
  ) a .
 (select
    b.*
     , rownum as row number
   from
     (select
       order id
       , event datetime
       , new event cd as event cd
      from order pipeline events
      where
       warehouse id = 'RNO1'
       and event datetime > sysdate - 2
     order by
       order id
       , event datetime
        , event cd
      ) b
  ) b
where
 a.order id = b.order id
 and a.prior event cd = '1001'
  and b.event cd = "1002"
  and a.row number = b.row number - 1;
```

#### With analytical function

```
Select avg(event datetime - prior event datetime) as
       avg wait
from
  (select
    new event cd as event cd
    , event datetime
    , lag(new event cd) over
      (partition by order id order by event datetime,
       new event cd)
       as prior event cd
    , lag(event datetime) over
      (partition by order id order by event datetime,
       new event cd)
       as prior event datetime
  from order pipeline events
  where
    warehouse id = 'RNO1'
    and event datetime > sysdate - 2
where
  prior event cd = '1001'
  and event cd = '1002';
```

## **SQL Pattern Matching**

#### "What's this about?"





#### Pattern Matching in Sequences of Rows

The Challenge – a real-world business problem

... detect if a phone card went from phone A to phone B to phone C... and back to phone A within 'N' hours..."

"... and detect if pattern above occurs at least 'N' times within 7 days ...

Prior to Oracle Database 12c pattern recognition in SQL is difficult

- Use multiple self joins (not good for \*)
  - T1.handset\_id <> T2.handset\_id <> T3.handset\_id AND.... T1.sim\_id= 'X' AND T2.time BETWEEN T1.time and T1.time+2....

ORACLE

- Use recursive query for \* (WITH clause, CONNECT BY)
- Use Window Functions (likely with multiple query blocks)

"

#### Pattern Matching in Sequences of Rows Objective

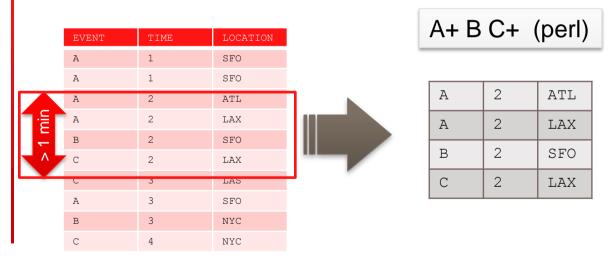
Provide native SQL language construct

Align with well-known regular expression declaration (PERL)

Apply expressions across rows

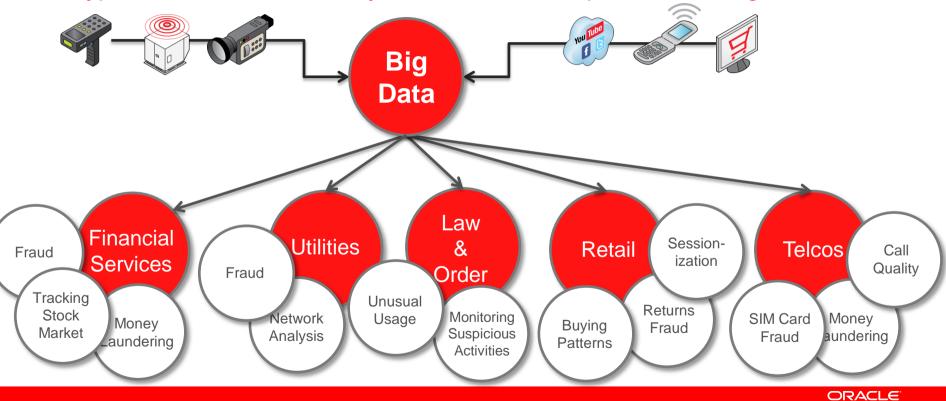
Soon to be in ANSI SQL Standard

*"Find one or more event A followed by one B followed by one or more C in a 1 minute interval"* 



### **Finding Patterns in Big Data**

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



## **SQL Pattern Matching**

#### **Conceptual Example**





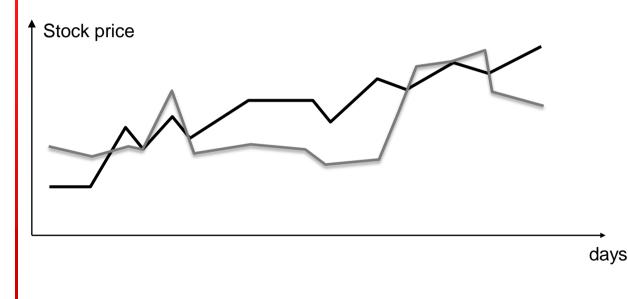
Basic steps for building the SQL command

- 1. Define the partitions/buckets and ordering needed to identify the 'stream of events' you are analyzing
- 2. Define the pattern of events and pattern variables identifying the individual events within the pattern
- 3. Define measures: source data points, pattern data points and aggregates related to a pattern
- 4. Determine how the output will be generated

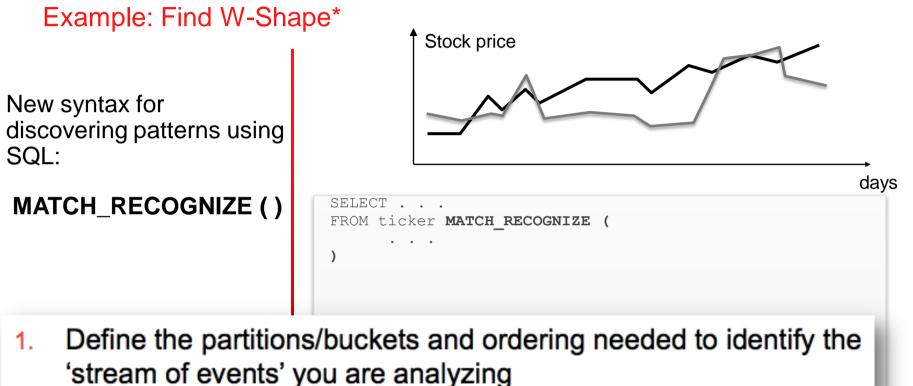
Example: Find A Double Bottom Pattern (W-shape) in ticker stream

Find a W-shape pattern in a ticker stream:

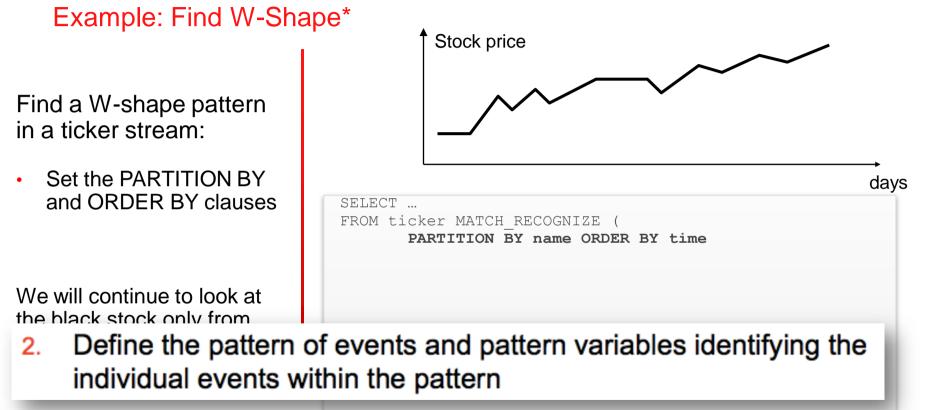
- Output the beginning and ending date of the pattern
- Calculate average price
   in the second ascent
- Find only patterns that lasted less than a week







#### ORACLE

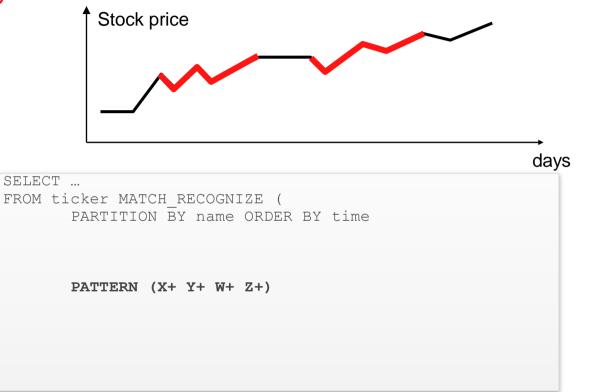


#### ORACLE

Example: Find W-Shape\*

Find a W-shape pattern in a ticker stream:

 Define the pattern – the "W-shape"

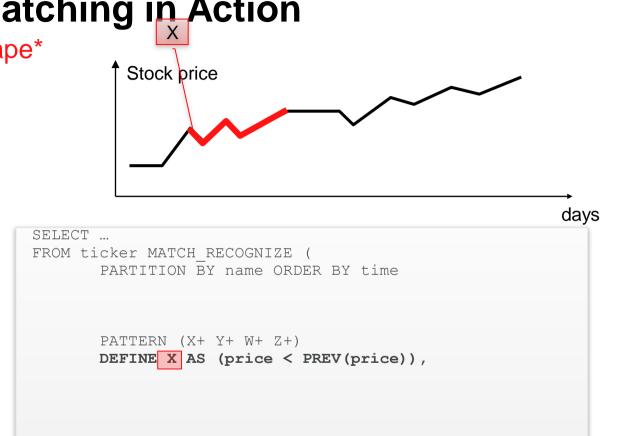


#### ORACLE

Example: Find W-Shape\*

Find a W-shape pattern in a ticker stream:

 Define the pattern – the first down part of the "Wshape"

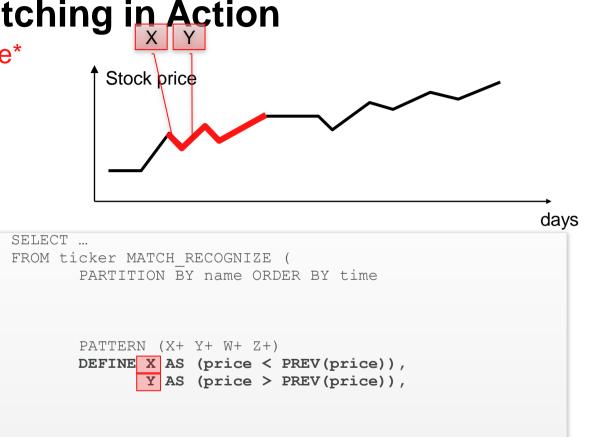


#### ORACLE

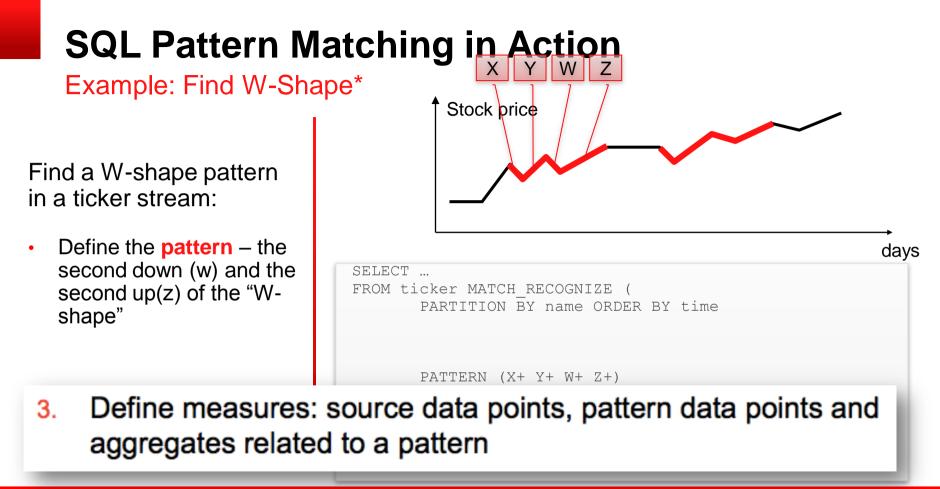
Example: Find W-Shape\*

Find a W-shape pattern in a ticker stream:

 Define the pattern – the first up part of "W-shape"



#### ORACLE



#### ORACLE

21 Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

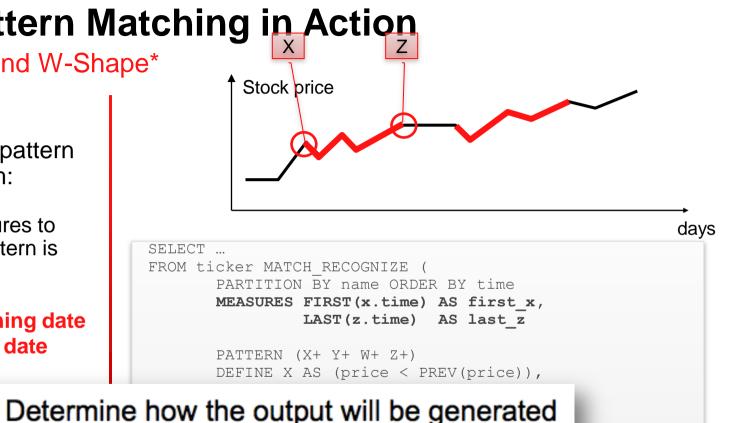
Example: Find W-Shape\*

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

> > 4.

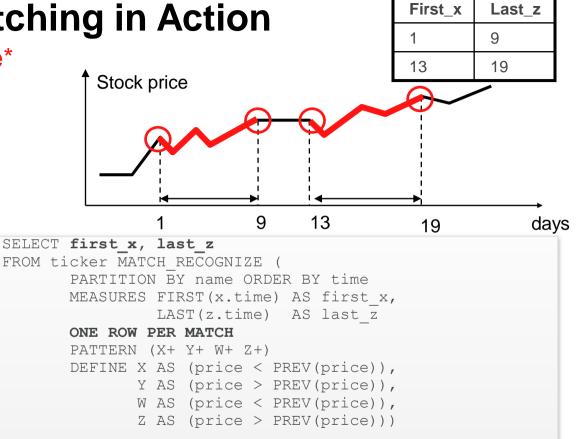


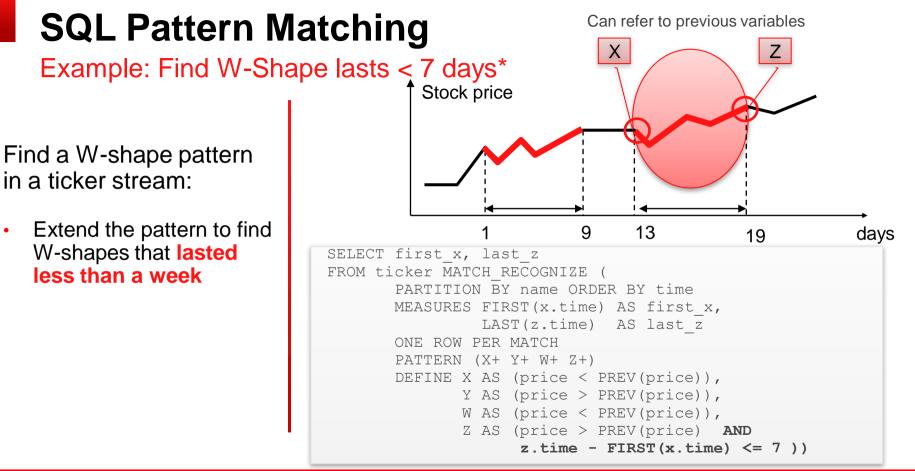
#### ORACLE

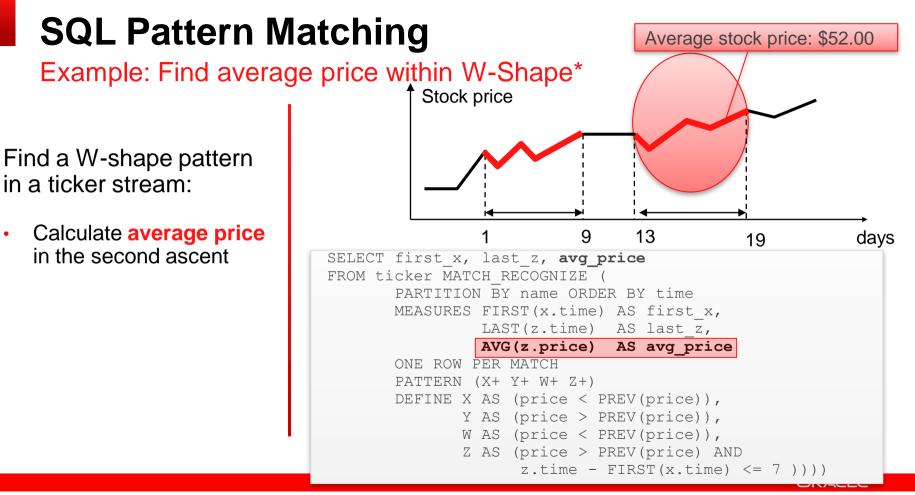
Example: Find W-Shape\*

Find a W-shape pattern in a ticker stream:

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







## **SQL Pattern Matching**

"Declarative" Pattern Matching

- 1. Define the partitions/buckets and ordering needed to identify the 'stream of events' you are analyzing
  - Matching within a stream of events (ordered partition of data)
  - MATCH\_RECOGNIZE (PARTITION BY stock\_name ORDER BY time MEASURES ...
- 2. Define the pattern of events and pattern variables identifying the individual events within the pattern

- Use framework of Perl regular expressions (conditions on rows)
  - PATTERN (X+ Y+ W+ Z+)
- Define matching using Boolean conditions on rows
  - DEFINE X AS (price > 15)

### SQL Pattern Matching, cont.

"Declarative" Pattern Matching

3. Define measures: source data points, pattern data points and aggregates related to a pattern

 MEASURES FIRST(x.time) AS first\_x, LAST(z.time) AS last\_z, AVG(z.price) AS avg\_price

- 4. Determine how the output will be generated
  - ONE ROW PER MATCH

#### **SQL** Pattern Matching

MATCH\_RECOGNIZE Syntax

<table\_expression> := <table\_expression> 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>
```



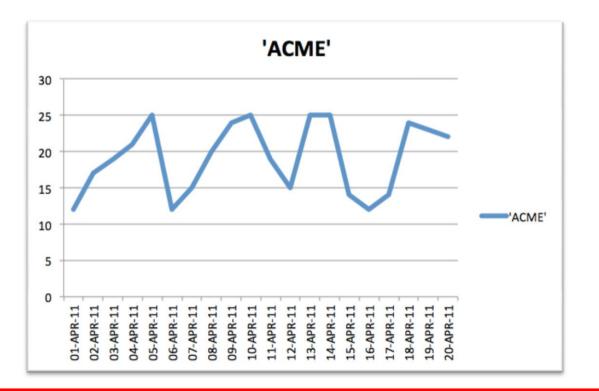
## **SQL Pattern Matching**

I'll get my hands dirty ...





#### **ACME Data Set**



#### **SQL** Pattern Matching

#### Real world use cases





#### Hot Off the Press ...

#### OTN Forum SQL and PL/SQL, 01/31/2014

			ve query
i nis questi	on is Not Answ	verea.	
Hi,			
i've a table	with 4 columns	5	
field1 num	nber		
field2 num	nber		
field3 cha			
field4 date	e		
i want to ex	tract field1 and	d field2 (field1 a	and 2 must be the same for the rows) where there are 3 consecutive field3='X' ordered by field4 desc.
i want to ex I want to ex		d field2 (field1 a	and 2 must be the same for the rows) where there are 3 consecutive field3='X' ordered by field4 desc.
		field2 (field1 a	and 2 must be the same for the rows) where there are 3 consecutive field3='X' ordered by field4 desc.
I want to ex	tract this:	field3	
l want to ex field1	tract this:	field3	field4
<b>I want to ex</b> field1 1	tract this:	field3 X	field4 10/10/2013
l want to ex field1 1 1	tract this: field2 1 1 1	field3 X X	field4 10/10/2013 10/9/2013
l want to ex field1 1 1 1	tract this: field2 1 1 1	field3 X X	field4 10/10/2013 10/9/2013
l want to ex field1 1 1 But not f	tract this: field2 1 1 this	field3 X X X	field4 10/10/2013 10/9/2013 10/8/2013
l want to ex field1 1 1 But not field1	tract this: field2 1 1 this	field3 X X X field3	field4 10/10/2013 10/9/2013 10/8/2013 field4

#### Hot Off the Press ...

#### OTN Forum SQL and PL/SQL, 01/31/2014

Depending on your answer, you might want something like this:

```
WITH
    got grp AS
1
   SELECT field1, field2, field3, field4
          ROW NUMBER () OVER ( PARTITION BY
   1
                                         The SQL statement to find the exact three records for each pair (field1, field2) looks as follows:
                            ORDER BY
                                         SOL> select * from tt
        - ROW NUMBER () OVER ( PARTITION BY
                                         match recognize (partition by field1, field2 order by field4 desc
                            ORDER BY
                                         all rows per match
                          ) AS grp
   FROM a
                                         pattern (strt a{2})
      got crp cnt AS
1.
                                         define
                                         a as field3=prev(field3));
   SELECT field1, field2, field3, field4
   ,
          COUNT (*) OVER (PARTITION BY fiel
   FROM got_grp
                                                          FIELD2 FIELD4 F
                                             FIELD1
   WHERE field3 = 'X'
                                             _____ _
        field1, field2, field3, field4
                                             20 10-OCT-13 X
SELECT
                                           2
FROM
        got grp cnt
                                           2 20 09-OCT-13 X
        grrp cnt >= 3
WHERE
                                           2 20 08-OCT-13 X
ORDER BY field1, field2, field4 DESC
```

#### **Pattern Matching with SQL Analytics**

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

```
if (!q.isEmpty() && (next.isEmpty() || (qt(q, prev) && eq(q, next)))) {
        state = "E"
        return state
    if (q.isEmpty() || eq(q, prev)) {
        state = "F";
        return state;
    return state;
private boolean eq(String a, String b) {
    if (a, isEmpty() || b, isEmpty()) {
        return false;
    return a.equals(b);
private boolean gt(String a, String b) {
    if (a.isEmpty() || b.isEmpty()) {
        return false:
    return Double.parseDouble(a) > Double.parseDouble(b);
private boolean lt(String a, String b) {
    if (a.isEmpty() || b.isEmpty()) {
        return false.
    return Double.parseDouble(a) < Double.parseDouble(b)
3
public String getState() {
    return this.state;
```

```
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 ))</pre>
```

#### 250+ Lines of Java and PIG

12 Lines of SQL

ORACLE

#### 20x less code, 5x faster

Analytical SQL in the Database

#### Summary

- Ranking functions
  - rank, dense\_rank, cume\_dist, percent\_rank, ntile
- Window Aggregate functions (moving and cumulative)
  - Avg, sum, min, max, count, variance, stddev, first\_value, last\_value
- LAG/LEAD functions
  - Direct inter-row reference using offsets
- Reporting Aggregate functions
  - Sum, avg, min, max, variance, stddev, count, ratio\_to\_report
- Statistical Aggregates
  - Correlation, linear regression family, covariance
- Linear regression
  - Fitting of an ordinary-least-squares regression line to a set of number pairs.
  - Frequently combined with the COVAR\_POP, COVAR\_SAMP, and CORR functions

- Descriptive Statistics
  - DBMS\_STAT\_FUNCS: summarizes numerical columns of a table and returns count, min, max, range, mean, stats\_mode, variance, standard deviation, median, quantile values, +/- n sigma values, top/bottom 5 values
- Correlations
  - Pearson's correlation coefficients, Spearman's and Kendall's (both nonparametric).
- Cross Tabs
  - Enhanced with % statistics: chi squared, phi coefficient, Cramer's V, contingency coefficient, Cohen's kappa
- Hypothesis Testing
  - Student t-test, F-test, Binomial test, Wilcoxon Signed Ranks test, Chi-square, Mann Whitney test, Kolmogorov-Smirnov test, One-way ANOVA
  - Distribution Fitting
    - Kolmogorov-Smirnov Test, Anderson-Darling Test, Chi-Squared Test, Normal, Uniform, Weibull, Exponential

## Summary

#### New Database 12c SQL Analytics

- Comprehensive analysis with SQL out of the box
  - ANSI compliant features with some additional extensions
- Common language SQL speeds up adoption
  - Widely known and used
  - 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

#### SQL - the best development language for Big Data?

#### Yes, because SQL is....





### Where to get more information

SQL Analytics Home Page on OTN

SQL PATTERN MATCHING IN ORACLE DATABASE 12c Patterns everywhere – Find them fast!



ATARASE

- <u>http://www.oracle.com/technetwork/database/bi-</u> <u>datawarehousing/sql-analytics-index-1984365.html</u>
- 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/





## **Hardware and Software**

#### ORACLE

## **Engineered to Work Together**

