

O'REILLY®

# ADVANCED SQL SERIES

## Proximal and Linear Interpolations

# CHALLENGE DESCRIPTION

We have a production line with multiple sensors.

Sensors are supposed to provide daily readings.

Unfortunately, the sensors are unreliable, and frequently break down. Therefore, the time series is incomplete.

Our task is to fill in the gaps for reporting purposes.

# CURVE FITTING

## ❖ Interpolation

Constructing new data points within the range of a discrete set of known data points

## ❖ Extrapolation

Estimating, beyond the original observation range, the value of a variable on the basis of its relationship with another variable

## ❖ Smoothing

Creating an approximating function that attempts to capture important patterns in the data

# INTERPOLATION

- ❖ Proximal interpolation

- Locate the nearest data value, and assign the same value  
AKA “Piecewise constant interpolation” or “Nearest-neighbor interpolation”

- ❖ Polynomial interpolation

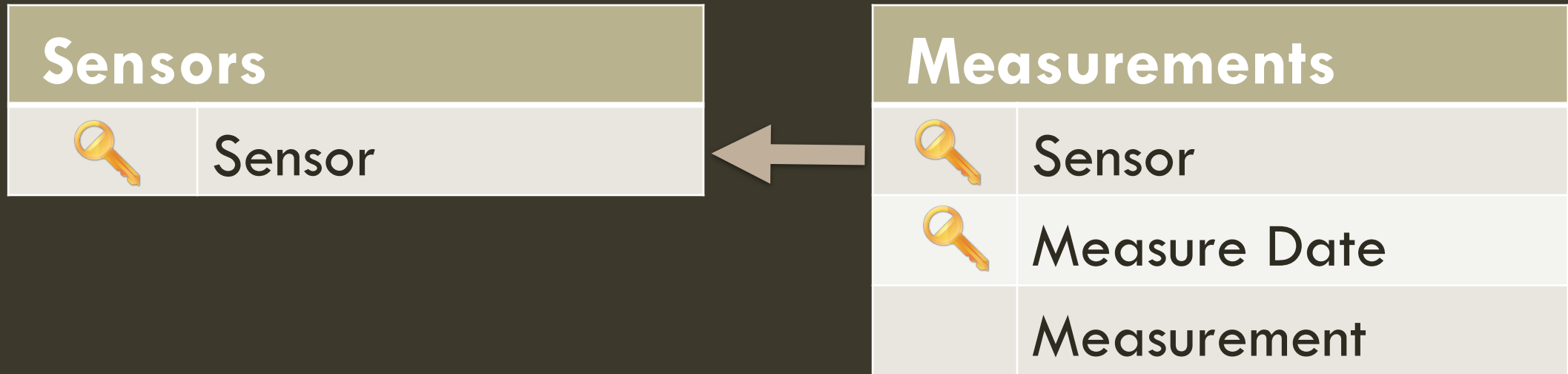
- Using polynomials to construct new data points within the range of a discrete set of known data points

- ❖ Linear interpolation

- A specific case of polynomial interpolation, which uses only linear polynomials; i.e. *exponentials with a degree of one or zero*

- ❖ Bi-linear, Spline, Gaussian, rational and many more...

# OUR BASE SCHEMA



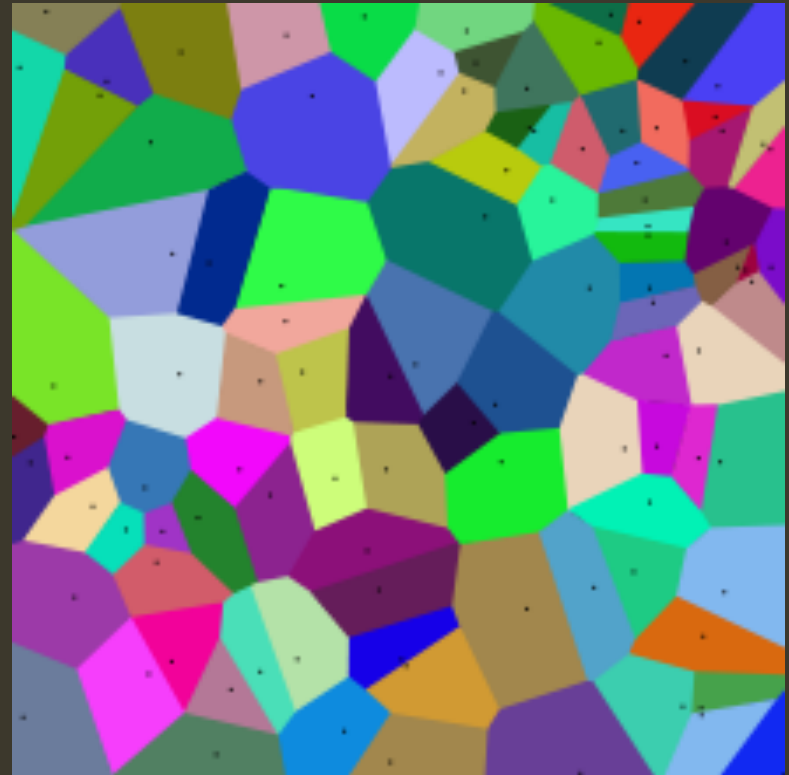
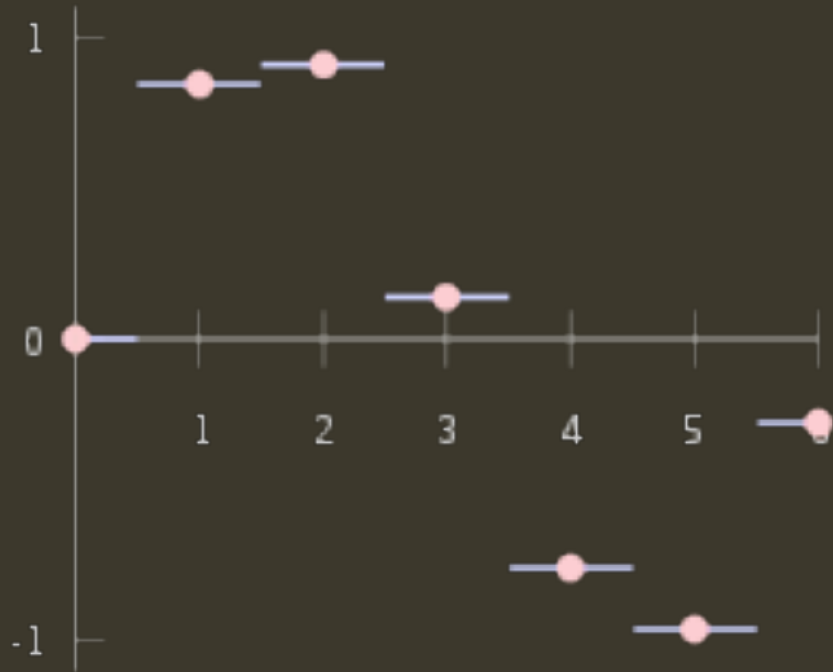
# OUR SAMPLE DATA

Sensor	Measure Date	Measurement
Sensor A	2016-01-01	10
Sensor A	2016-01-05	12
Sensor A	2016-01-11	14
Sensor A	2016-01-21	12
Sensor A	2016-01-27	8
Sensor B	2016-01-10	9
Sensor B	2016-01-11	22
Sensor B	2016-01-23	10
Sensor C	2016-01-01	16
Sensor C	2016-01-03	14
Sensor C	2016-01-21	19

# PHASE 1 – GENERATE SOURCE DATA



# PROXIMAL INTERPOLATION



**Voronoi Diagram**



# SQL SOLUTION METHOD

- ❖ Write down the expected result set
- ❖ Generate the necessary source data set rows
- ❖ Consider the required output columns, and think of all the additional values you need to get there

*Even if you don't know how to get there yet.*

*That will be much easier once written down...*

# EXPECTED RESULT SET

Sensor	Date	Measurement	Measurement Type
Sensor A	1/1/2016	10	Measured
Sensor A	1/2/2016	10	Interpolated
Sensor A	1/3/2016	12	Interpolated
Sensor A	1/4/2016	12	Interpolated
Sensor A	1/5/2016	12	Measured
Sensor A	1/6/2016	12	Interpolated
Sensor A	1/7/2016	12	Interpolated
Sensor A	1/8/2016	14	Interpolated
Sensor A	1/9/2016	14	Interpolated
Sensor A	1/10/2016	14	Interpolated
Sensor A	1/11/2016	14	Measured
Sensor A	1/12/2016	14	Interpolated
... <TRIMMED>...			
Sensor A	1/26/2016	8	Interpolated
Sensor A	1/27/2016	8	Measured
Sensor B	1/10/2016	9	Measured
Sensor B	1/11/2016	22	Measured
Sensor B	1/12/2016	22	Interpolated

# FIRST, LET'S GENERATE THE MISSING ROWS

- ❖ We need a Calendar table with sequential dates
- ❖ We'll use a Numbers table to populate the calendar

Both can be generated 'on-the-fly' with CTEs or derived tables, but you'll soon see how useful they are for solving many types of challenges.

*Highly recommended for every database*

# PHASE 2 – AUXILIARY DATA TABLES



# PHASE 3 – CONSTRUCTING OUR ROW SET



# FIGURE OUT WHICH DATA COLUMNS WE NEED

Sensor	Date	Measure	Measurement Type
Sensor A	1/1/2016	10	Measured
Sensor A	1/2/2016	NULL	Interpolated
Sensor A	1/3/2016	NULL	Interpolated
Sensor A	1/4/2016	NULL	Interpolated
Sensor A	1/5/2016	12	Measured
Sensor A	1/6/2016	NULL	Interpolated
Sensor A	1/7/2016	NULL	Interpolated
Sensor A	1/8/2016	NULL	Interpolated
Sensor A	1/9/2016	NULL	Interpolated
Sensor A	1/10/2016	NULL	Interpolated
Sensor A	1/11/2016	14	Measured



# LAST / NEXT NON-NULL VALUE

## Exercise

Get the last and next non-null values using correlated sub queries

# LAST / NEXT NON-NULL VALUE

- ❖ Here things get a bit tricky, so let's use a simple example to demonstrate the solution
- ❖ We will use two different solution to solve this challenge

*See if you can find more...*



LAST / NEXT NON NULL  
VALUE

*SOLUTION #1 – SIMPLE EXAMPLE*



# LAST AND NEXT NON NULL VALUE DATES

```
MAX (CASE WHEN Measurement IS NOT NULL THEN Date ELSE NULL END)
OVER (PARTITION BY Sensor ORDER BY Date ASC
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
AS [Last Date With Measurement Value],
MIN (CASE WHEN Measurement IS NOT NULL THEN Date ELSE NULL END)
OVER (PARTITION BY Sensor ORDER BY Date ASC
      ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING)
AS [Next Date With Measurement Value]
```

# LAST AND NEXT NON NULL VALUES

MAX ([Measurement])

OVER (PARTITION BY [Sensor], [Last Date With Measurement  
Value]) AS [Last Value],

MAX ([Measurement])

OVER (PARTITION BY [Sensor], [Next Date With Measurement  
Value]) AS [Next Value]

# PHASE 4 – LAST / NEXT NON NULL VALUE



# AND THE FINAL SOLUTION IS NOW EASY

```
ISNULL(Measurement,  
    CASE WHEN DATEDIFF(DAY, Date, [Next Date With Value])  
          > DATEDIFF(DAY, [Last Date With Value], Date)  
        THEN [Last Value] ELSE [Next Value]  
    END ) AS Measurement,  
CASE WHEN Measurement IS NULL  
    THEN N'Interpolated' ELSE N'Measured'  
END AS [Measurement Type]
```

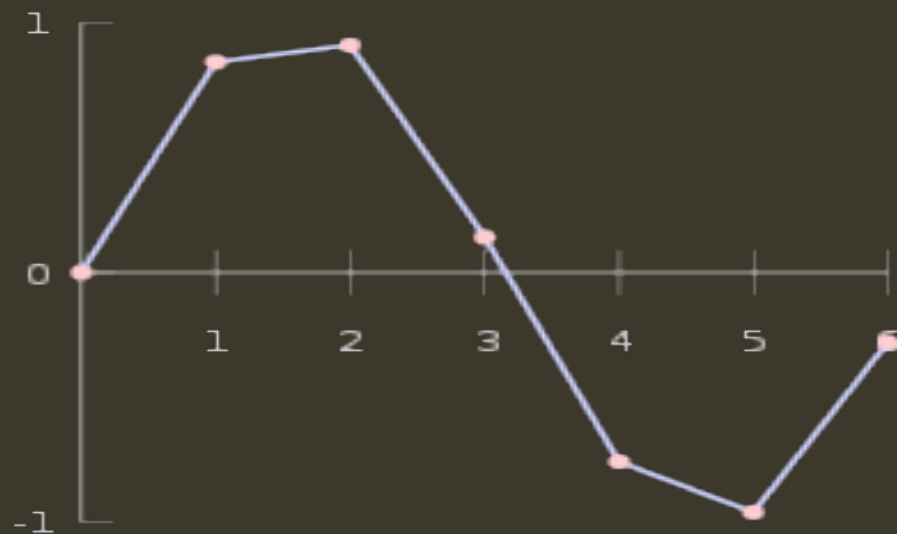
# PHASE 5 – PROXIMAL INTERPOLATION SOLUTION



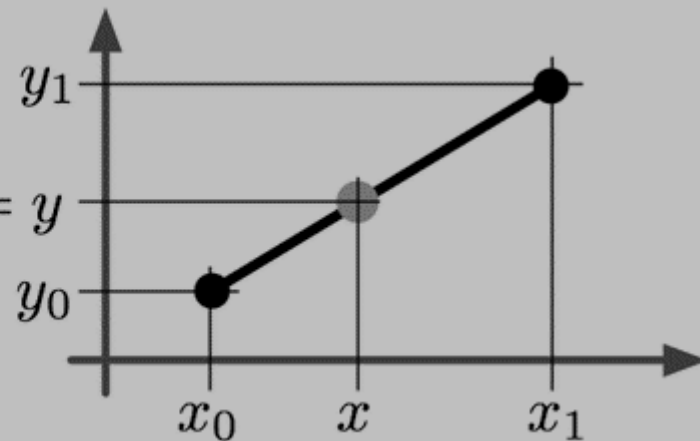
# PROXIMAL INTERPOLATION



# LINEAR INTERPOLATION



$$y_0 + \frac{(x - x_0)(y_1 - y_0)}{x_1 - x_0} = y$$





# EXPECTED RESULT SET

Sensor	Date	Measurement	Measurement Type
Sensor A	1/1/2016	10.00	Measured
Sensor A	1/2/2016	10.50	Interpolated
Sensor A	1/3/2016	11.00	Interpolated
Sensor A	1/4/2016	11.50	Interpolated
Sensor A	1/5/2016	12.00	Measured
Sensor A	1/6/2016	12.33	Interpolated
Sensor A	1/7/2016	12.67	Interpolated
Sensor A	1/8/2016	13.00	Interpolated
Sensor A	1/9/2016	13.33	Interpolated
Sensor A	1/10/2016	13.67	Interpolated
Sensor A	1/11/2016	14.00	Measured
Sensor A	1/12/2016	13.80	Interpolated
... <TRIMMED>...			
Sensor A	1/26/2016	8.67	Interpolated
Sensor A	1/27/2016	8.00	Measured
Sensor B	1/10/2016	9.00	Measured
Sensor B	1/11/2016	22.00	Measured
Sensor B	1/12/2016	21.00	Interpolated

# LET'S REUSE WHAT WE ALREADY HAVE

- ❖ The numbers and calendar tables
- ❖ The needed source rows with the full date ranges
  - ❖ The “Measurements With Sequential Date Ranges” view

*But this time we will use a different solution to  
get the last and next non null values...*

# LAST / NEXT NON-NULL VALUE

- ❖ In the previous solution, we used **MAX** and **MIN** over a **CASE** expression to identify the last and next **dates** where **measurement** was **NOT NULL**
- ❖ That in turn, provided us with a unique **grouping value** we could use for the **PARTITION BY** clause to find the last and next measurement values for any date.

*The date was convenient because we also needed it for proximity.*

*But, can you think of other 'grouping' values we can use?*

# LAST / NEXT NON-NULL VALUE

Note that we don't necessarily need the dates for the last and next non null values, but we do need something to calculate the distance from both neighboring known data points


*Can you guess what it is?*

LAST / NEXT NON NULL  
VALUE

*SOLUTION #2 – SIMPLE EXAMPLE*



# HERE WE NEED SLIGHTLY DIFFERENT COLUMNS

Sensor	Date	Meas...	Last NN Meas...	Next NN Meas..		Measurement Type
Sensor A	1/1/2016	10	10	10		Measured
Sensor A	1/2/2016	NULL	10	12		Interpolated
Sensor A	1/3/2016	NULL	10	12		Interpolated
Sensor A	1/4/2016	NULL	10	12		Interpolated
Sensor A	1/5/2016	12	12	12		Measured
Sensor A	1/6/2016	NULL	12	14		Interpolated
Sensor A	1/7/2016	NULL	12	14		Interpolated
Sensor A	1/8/2016	NULL	12	14		Interpolated
Sensor A	1/9/2016	NULL	12	14		Interpolated
Sensor A	1/10/2016	NULL	12	14		Interpolated
Sensor A	1/11/2016	14	14	14		Measured

# GENERATING THE COUNTS FOR GROUPING VALUES

```
COUNT([Measurement])  
OVER(PARTITION BY [Sensor] ORDER BY [Date] ASC  
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW  
) AS [Count Backwards] ,  
COUNT([Measurement])  
OVER(PARTITION BY [Sensor] ORDER BY [Date] DESC  
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW  
) AS [Count Forward]
```

# GETTING THE LAST / NEXT NON NULL VALUES + SEQUENCE POSITION

```
MAX([Measurement])  
OVER (PARTITION BY [Sensor], [Count Backwards])  
AS [Last Non Null Measurement],  
MAX([Measurement])  
OVER (PARTITION BY [Sensor], [Count Forward])  
AS [Next Non Null Measurement] ,  
ROW_NUMBER() OVER(PARTITION BY [Sensor], [Count Backwards]  
                    ORDER BY [Date] ASC  
                    ) AS [Sequence Within Group]
```



# PHASE 4 – LAST / NEXT NON NULL VALUE + POSITION



# AND THE FINAL SOLUTION IS NOW EASY

```
ISNULL ([Measurement],  
[Last Non Null Measurement] + ( (  
([Next Non Null Measurement] - [Last Non Null Measurement])  
    / MAX([Sequence Within Group])  
    OVER( PARTITION BY [Sensor], [Count Backwards])  
    ) * ([Sequence Within Group] - 1)  
    ) ) AS [Measurement]
```

# PHASE 5 – LINEAR INTERPOLATION SOLUTION



| THANK  
YOU!

[amilevin@gmail.com](mailto:amilevin@gmail.com)

O'REILLY®

# ADVANCED SQL SERIES

## Proximal and Linear Interpolations