Using Normalized Status Change Events Data in Business Intelligence

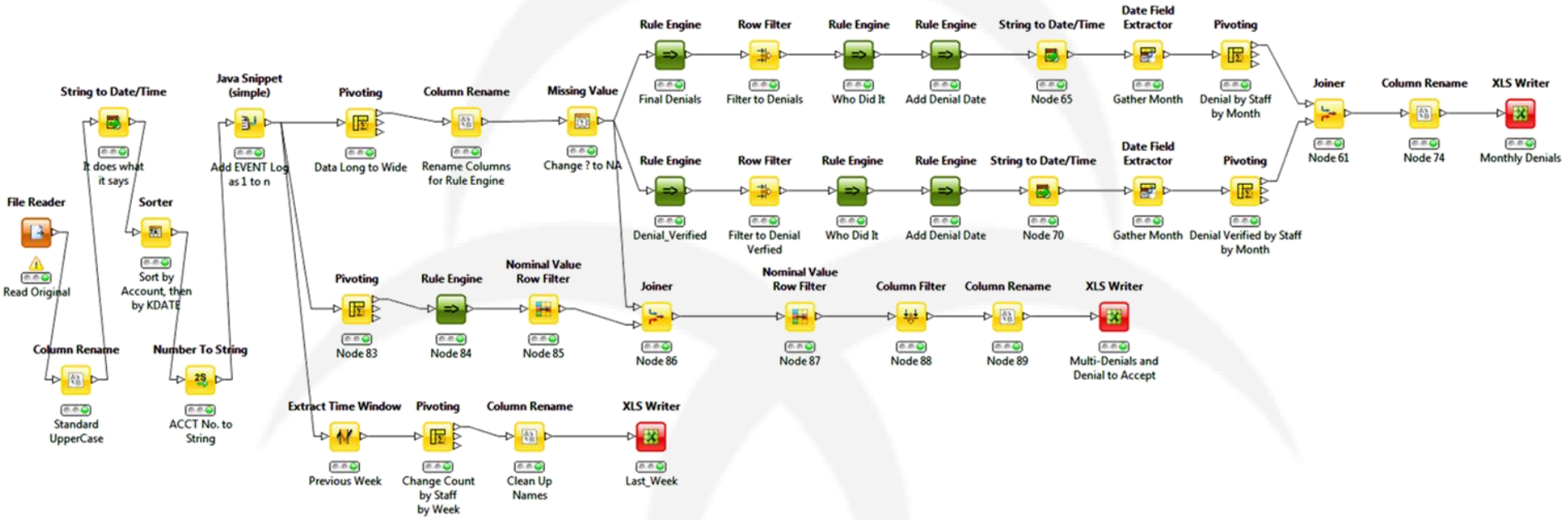
Mark C. Cooke, Ph.D.

Tax Management Associates, Inc.

A note to the audience

- TMA is a company that serves state and local government, providing auditing and fraud detection services
- I am an anthropologist by education, which means I love behavior and meaning (data as a referent and remnant). By business experience, I am involved in data analysis, solving problems with data transformations, aggregations, and puzzles of all sorts and shapes

The Solution



Thanks to Rosaria Silipo for making this a complete KNIME solution. By providing a Java Snippet node suggestion she resolved two problems including relying on an external R script. Bravissimo!

The Problem

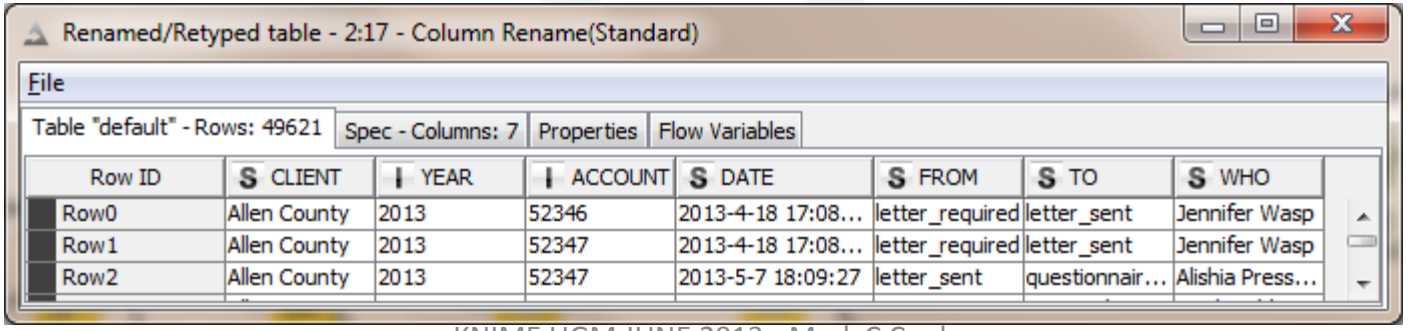
- Software developers like to build software that serves a purpose
- The database (MySQL) that drives the software is designed to follow two principles:
  - It is normalized to enhance functionality
  - It stores data that helps serve the purpose of the product – in other words, a lot about the product, not so much about how the software is used

The Real Problem

- Management likes to know what's going on
  - In order to know, they need to know not just what is happening, but who is doing it, how it is happening, how long it is taking, etc.
- Management is concerned with three things:
  - How many widgets did we make in this period?
  - How many resulted from each person's efforts?
  - How much work did each person do, and was it efficient?

Some Clues

- The application in question uses a state machine model to force widgets down a production line, metaphorically
- Every user initiated state change in the system is recorded as a normalized instance of who did it, when they did it, and what state it was changed from and to:

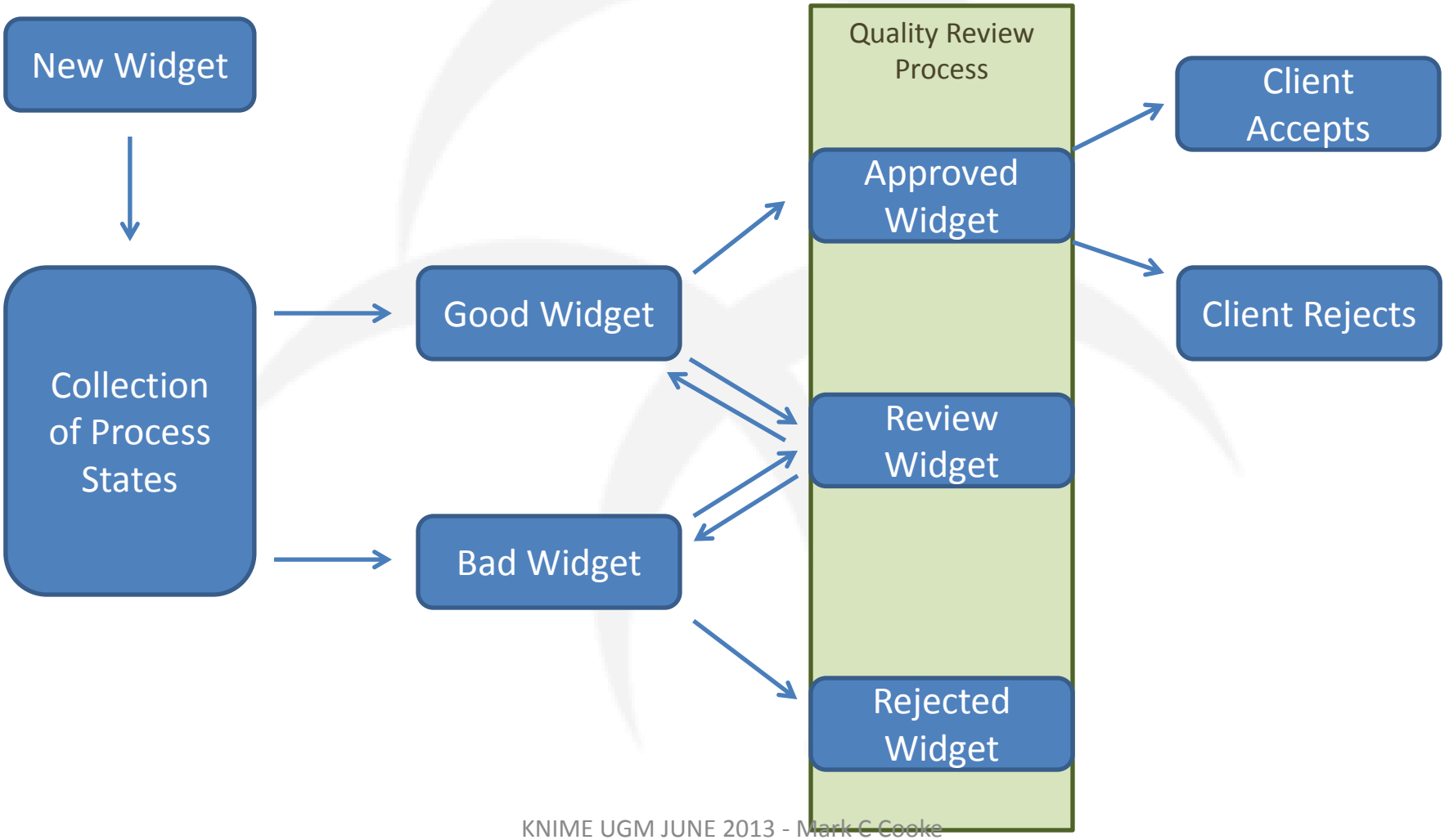

Renamed/Retyped table - 2:17 - Column Rename(Standard)

File

Table "default" - Rows: 49621 | Spec - Columns: 7 | Properties | Flow Variables

Row ID	S CLIENT	I YEAR	I ACCOUNT	S DATE	S FROM	S TO	S WHO
Row0	Allen County	2013	52346	2013-4-18 17:08...	letter_required	letter_sent	Jennifer Wasp
Row1	Allen County	2013	52347	2013-4-18 17:08...	letter_required	letter_sent	Jennifer Wasp
Row2	Allen County	2013	52347	2013-5-7 18:09:27	letter_sent	questionnair...	Alishia Press...

The States, generally



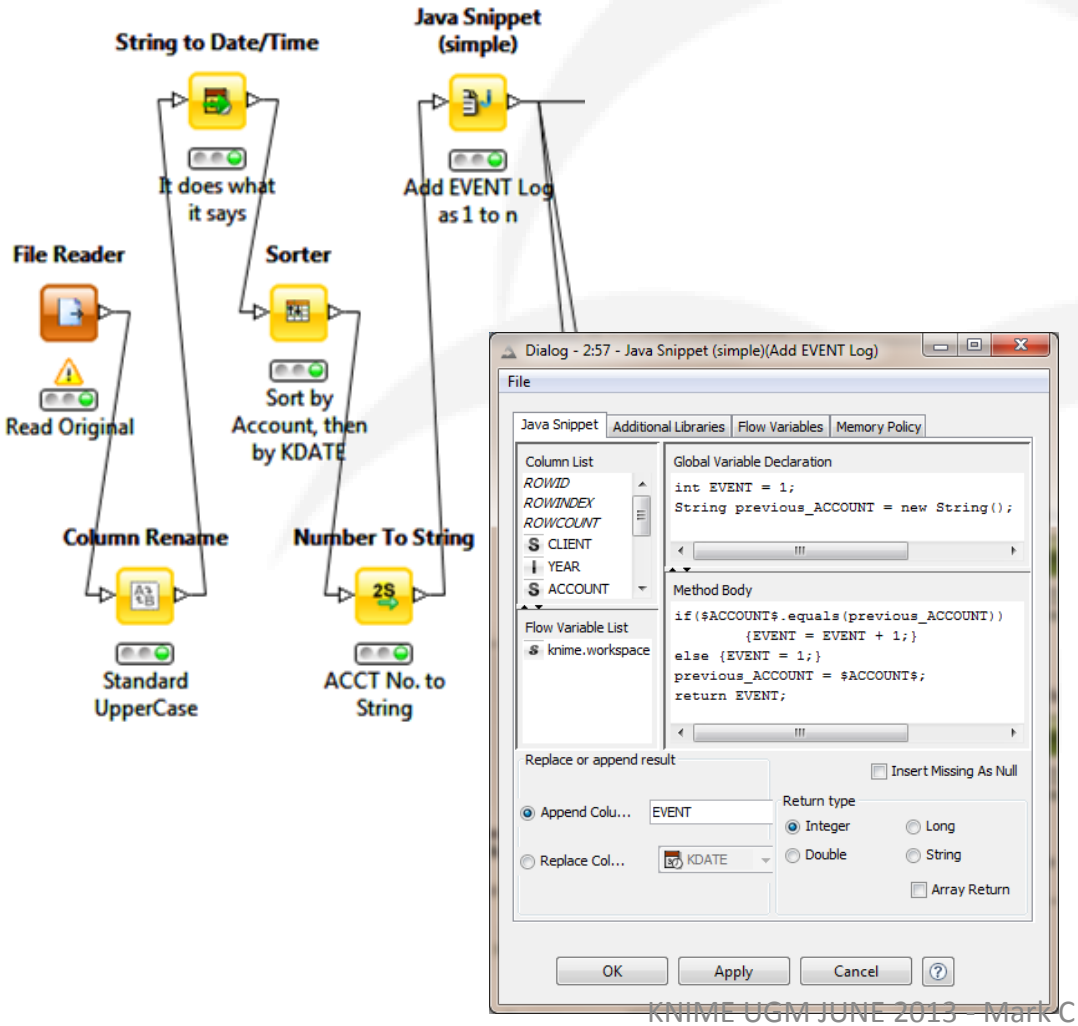
More Problems

- A widget (single entity) may be deemed good multiple times over it's life cycle (same state) by different people…within the same time period (even with the same day)
- Therefore, a query of the number of "good" widgets will always return more than the unique number of widgets with a "good" state
- Widgets can go from good to bad to good again, or just bad to good or good to bad…
- We still want to know who is doing what and where all of these state changes are coming from…

Crafting a Solution

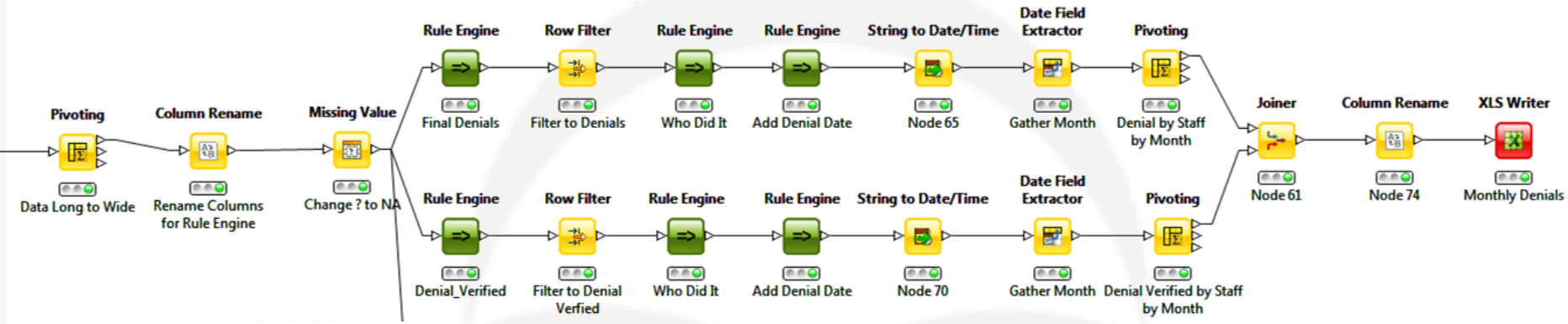
- First Step: Recognize the resource, i.e. the state table and the who, what, when
- Second Step: Recognize the limitations of a normalized database structure so that SQL querying was “inopportune”
- KNIME is the best possible resource because it allows a functional workflow that can be integrated and run on demand to produce business intelligence when required
- In our case, this report is run at minimum once a week, sometimes more frequently

KNIME Highlights



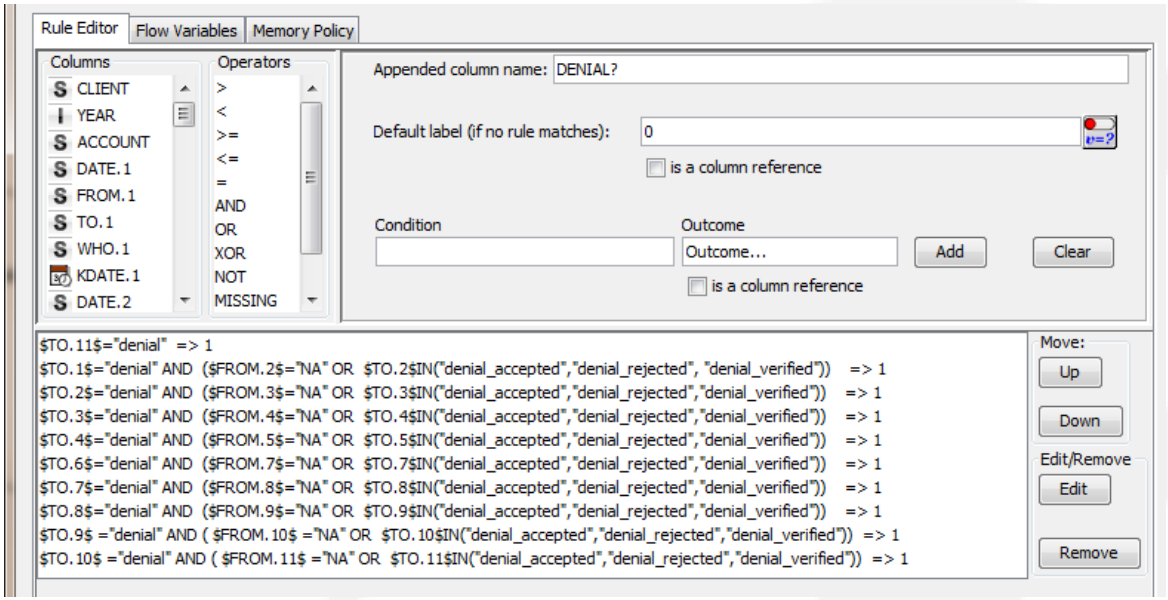
1. File read from copy of table
2. Column names cleaned
3. Date Extracted to KNIME format
4. Sort by ACCOUNT + KDATE as the ORDER BY operation
5. Use the Java Snippet to add an EVENT as a new column

KNIME Highlights



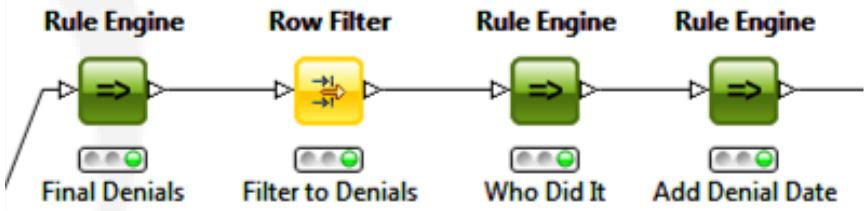
- A Pivoting Node allows us to take the data from "long" to "wide" or normalized to de-normalized table structure – very powerful as now we have the opportunity to complete many analyses including time series, association pattern mining, etc.
- We are interested here in looking for widgets made and widgets approved by individuals, two analyses which we can rejoin at the end by employee name and for any given period
- Rule Engines do a lot of the heavy work in this branch to organize the data and reduce it to what we need

Rule Engines

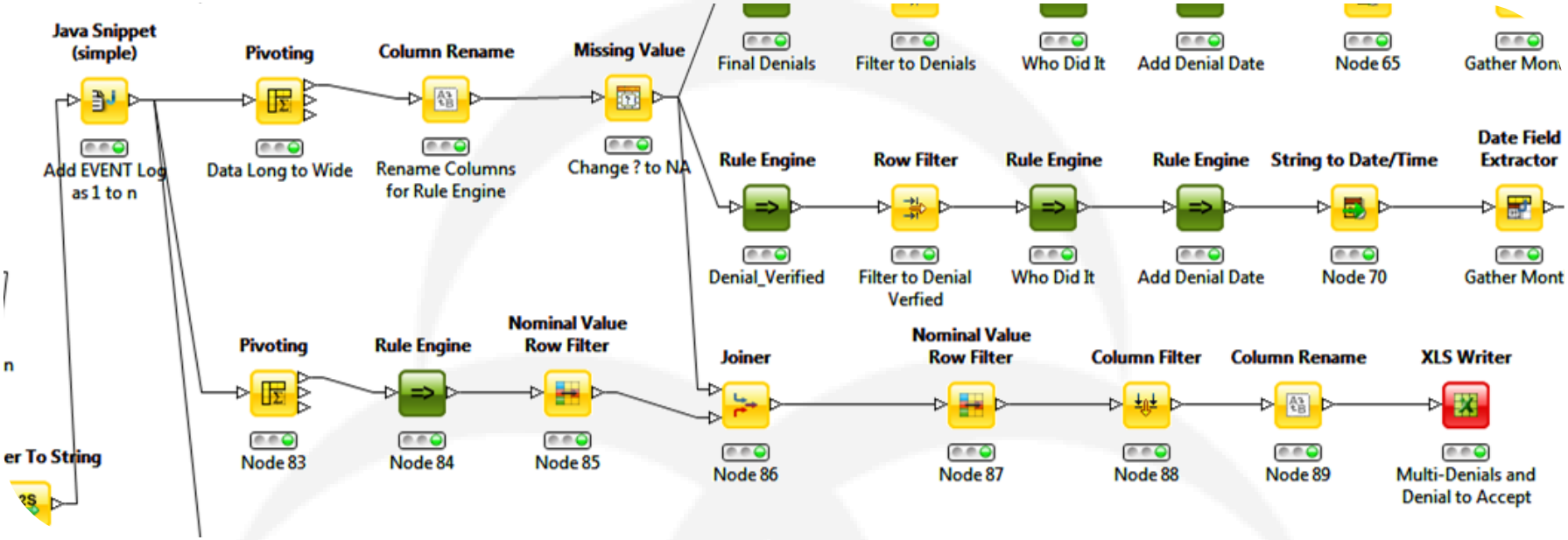


- Used to logically review the de-normalized data to find states of interest that either never shifted out of that state, only progressed forward, or at least ended up in the state of interest with no further changes

- They are then also used to pluck out the WHO column entry and the KDATE for each line of data which is the last entry in the time series to meet the logical condition

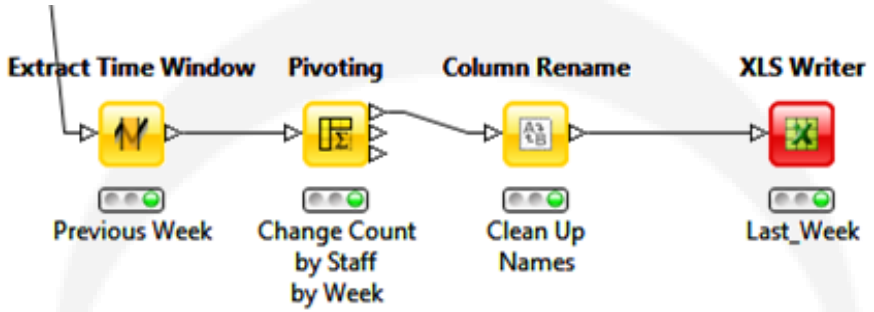


Second Workflow



- Pivoting off the long data grouping by ACCOUNT and pivoting on TO so that we have a COUNT BY states changed to by state name
- The Rule Engine is used to create a column of WHAT? types identifying accounts which were declared good more than once or went good to bad
- Filtering by types and joining with the wide table now gives me a class of objects with their history for analyses (cf. patterns)

Third Workflow



- Branching from the “long” data structure
- Reduce data down to previous week
- The Pivoting Node is used again, but this time grouping by WHO and pivoting on TO achieving a table with counts by staff by state change made

What We Achieve

- Happy managers, hooray! (okay, that depends on what the numbers actually are…)
- Measure of product velocity – the speed at which an account moves from “new” to “billable”
- Measure of project velocity – the speed at which a client moves from start to finish
- Employee Productivity – Individual and Period on Period
- Review of behavioral patterns which might be indicative of a bottleneck, systemic issue, or possible enhancement
- Eventually analyses of predictive or interpretive analytics related to account success measures (survival and time

Thank you!

If you have any ideas on how this process could be improved, or would like to discuss how it was possible now is the time! If you would like to contact me in the future:

Email: Mark.Cooke@tma1.com
Skype: markccooke
LinkedIn: http://lnkd.in/vYxfaW