Visualizing Data Lineages

David Manley
Senior Institutional Research Analyst
Rowan University
July 17, 2024

RowanUniversity

njair
NEW JERSEY
ASSOCIATION for
INSTITUTIONAL
RESEARCH

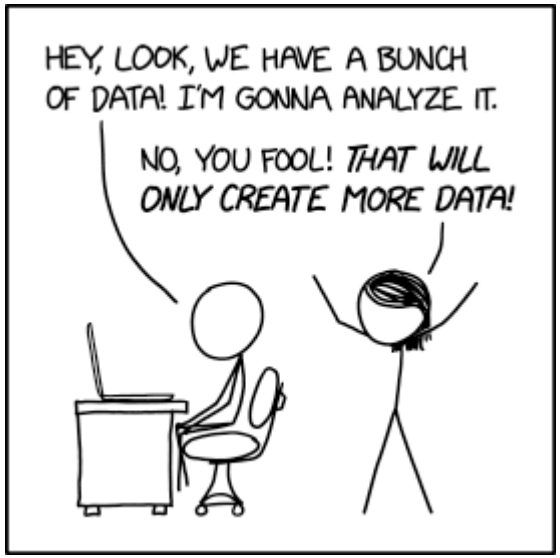
What to Expect

- A light survey of data lineage visualization tools (by no means exhaustive)
- A brief tutorial of how I draw data lineages (from very general to very technical)
- Comments on my experiences trying to find the “perfect” method for documenting lineages

Who Am I

- Senior Institutional Research Analyst at Rowan University
- Formerly a Statistics Instructor at Rowan University
- In a previous life I was a Quality & Reliability Engineer in manufacturing
- Data roots in:
  - Excel (with VBA to handle 65,000+)
  - JMP v3
  - Minitab v12
  - R (before Tidyverse and ggplot2)
- **In all job positions that I have held, I have used flow charts**

Data Trap



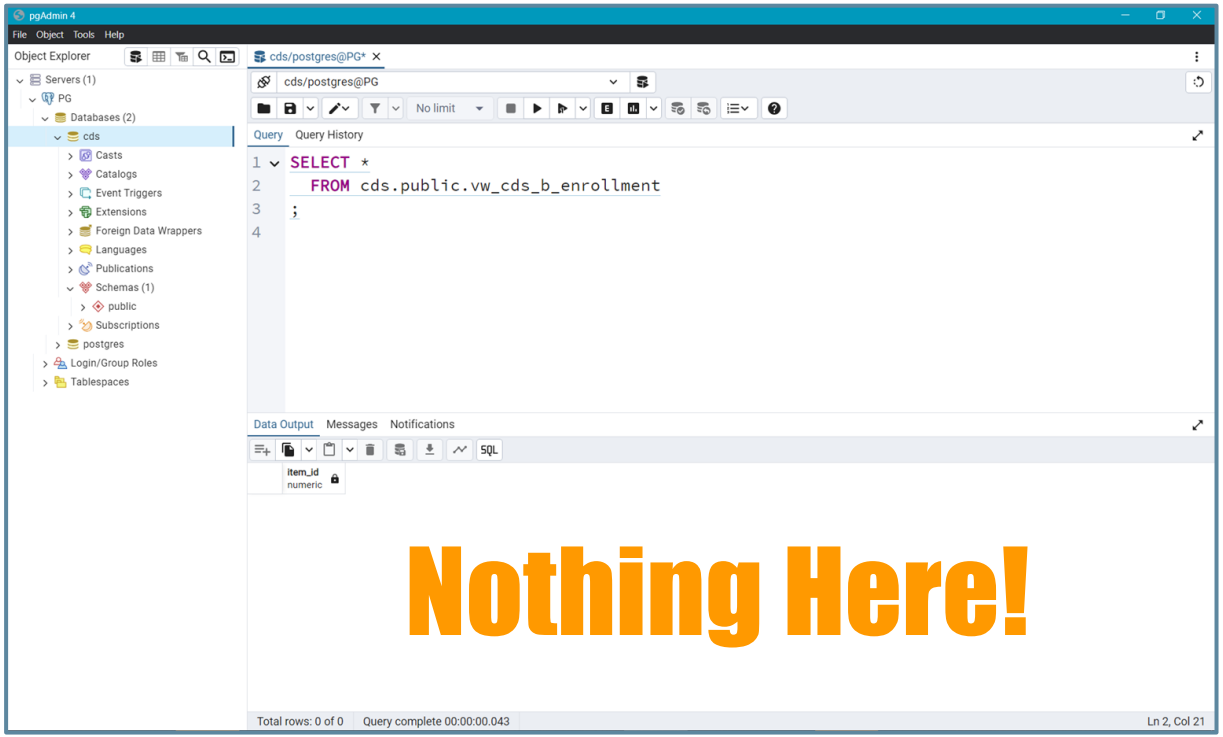
https://xkcd.com/2582/
xkcd by Randall Munroe

Don't worry
we will only discuss

METADATA

in this presentation!

In Fact, My Example Data Set Is Completely Empty



Before we go any further...

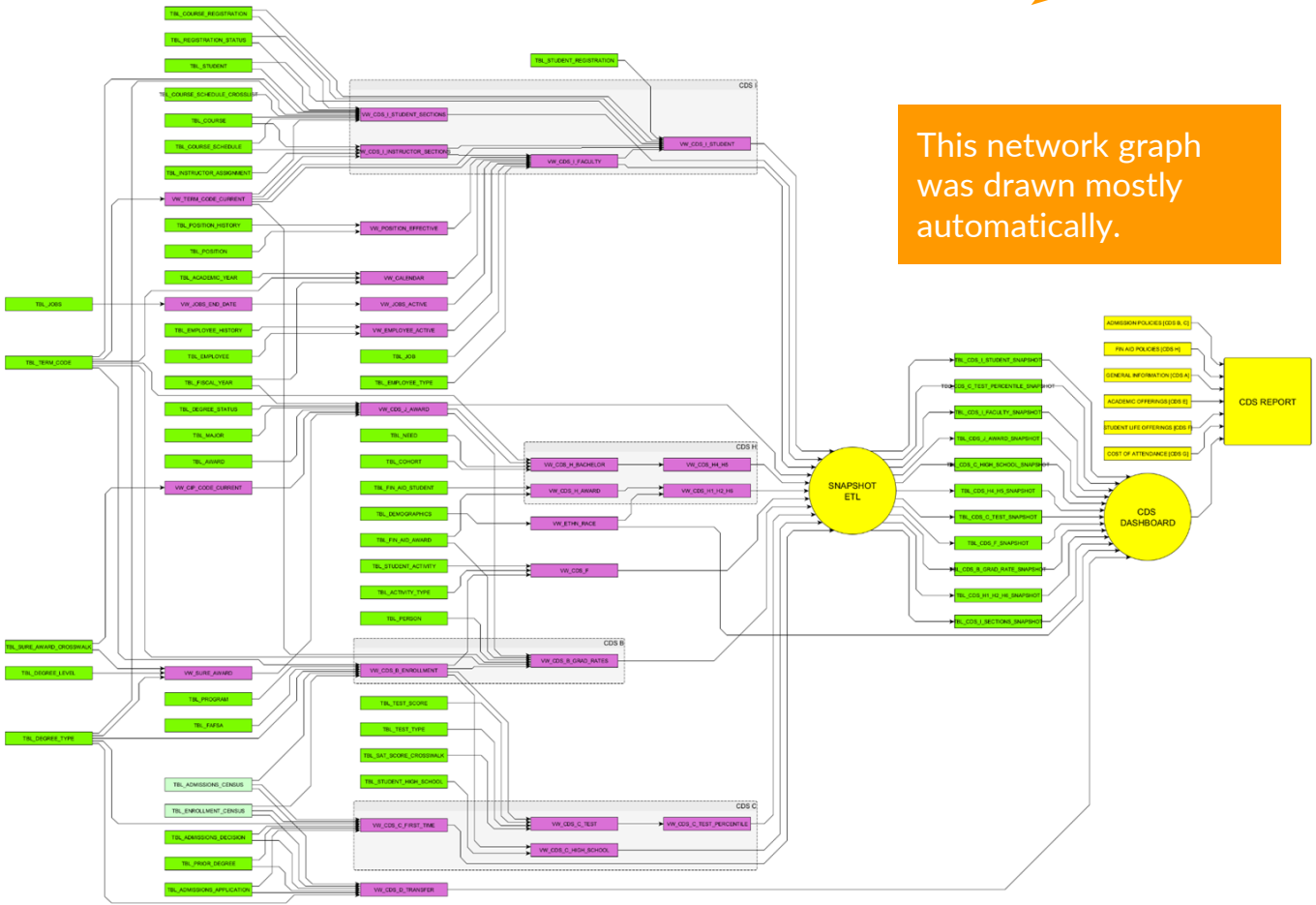
Callouts will have relevant file names for the slide

All example files used in this presentation are available on my Google Drive

https://drive.google.com/drive/folders/1j03hiCeEVpzQkp9p1EnrOREmXYuaGdLD?usp=sharing

cds_network_final.graphml

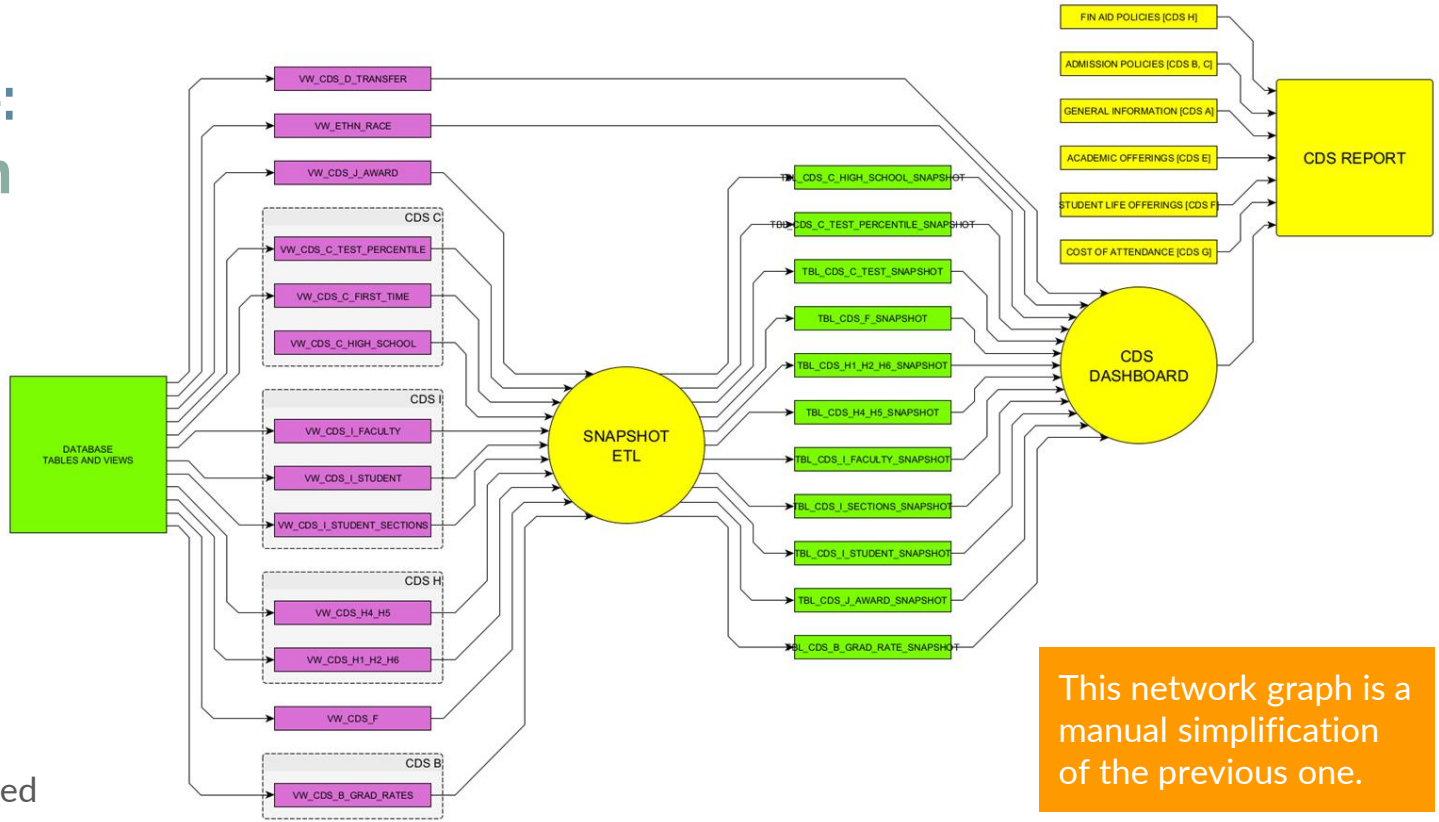
Today's Example: Common Data Set (CDS)



This network graph was drawn mostly automatically.

- Tables (tbl_???)
- Views (vw_???)
- Manually added

Today's Example: Common Data Set (CDS) [simpler]

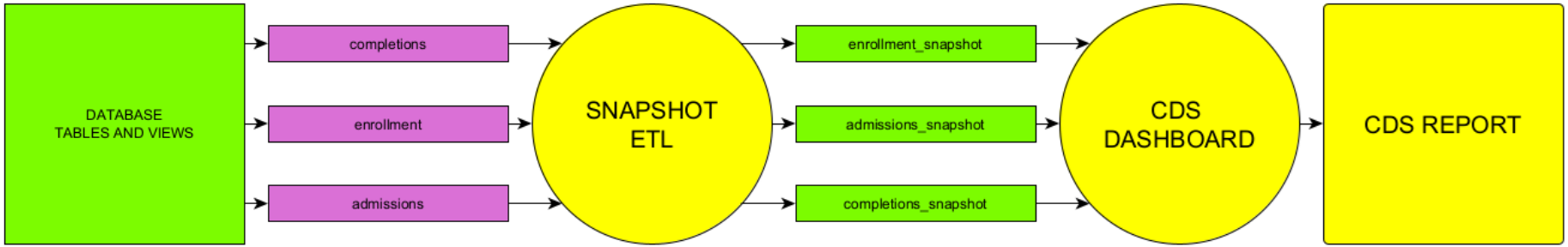


- Tables (tbl_???)
- Views (vw_???)
- Manually added

This network graph is a manual simplification of the previous one.

cds_network_final_simplest.graphml

Today's Example:
Common Data Set (CDS)
[simplest]



Tables
(tbl_???)

Views
(vw_???)

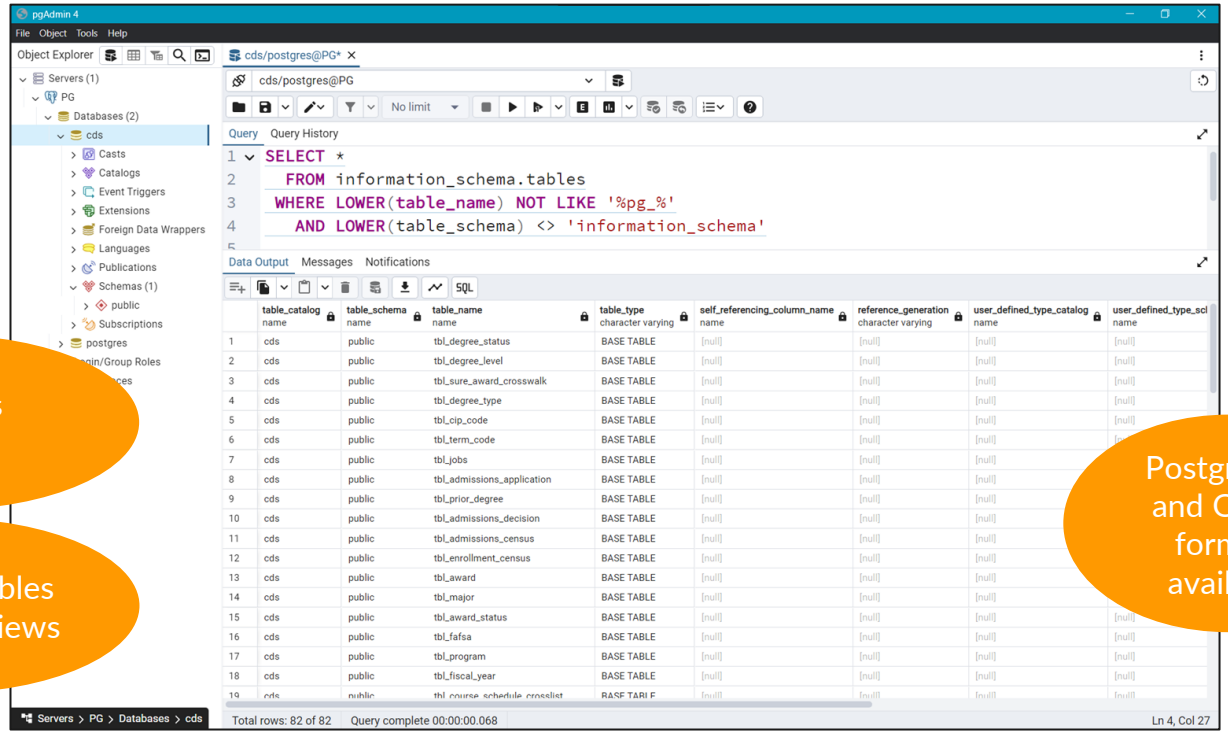
Manually added

This network graph was drawn completely manually.

cds_create_tables_view_postgres.sql
cds_create_tables_views_oracle.sql

cds_drop_views_tables_postgres.sql
cds_drop_views_tables_oracle.sql

Example CDS Database



Contains no data

56 empty tables
26 dummy views

PostgreSQL and Oracle formats available

Scope and Goals

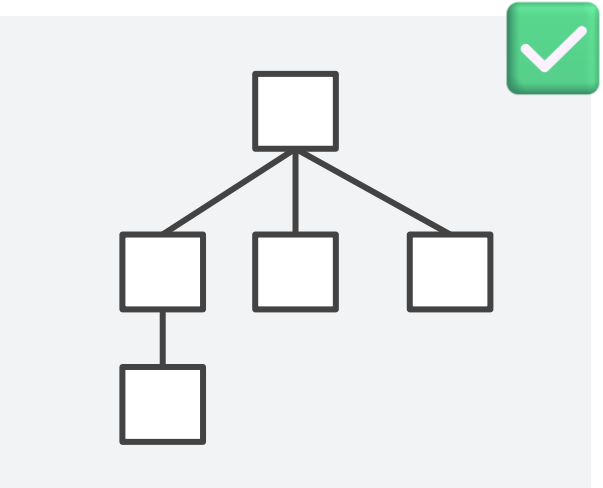
Why Visualize Data Lineages?

1. Design data flows (intent)
2. Monitor data systems (reality)
3. Documenting (consistency)
4. Training (speed)
5. Explaining (common understanding)

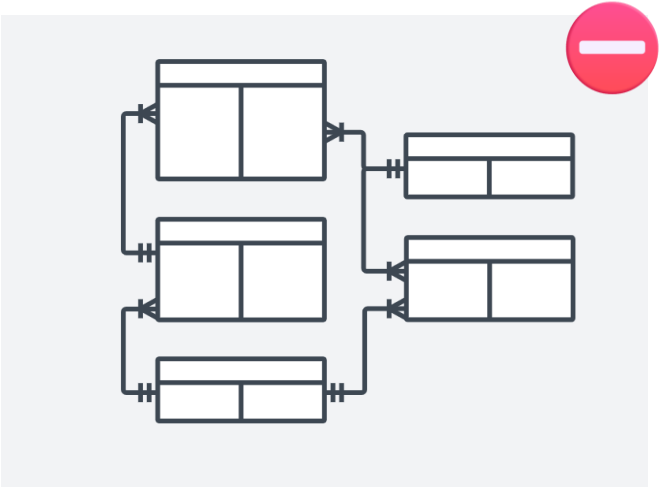
Bonus: Change of pace from bar charts, line graphs, heatmaps, etc.

For This Presentation We Want to

Visualize Data Flows (Hierarchies)



Not Entity Relationship (ER) Diagrams



Drawing Lineages Can Be Tedious…
Rearranging a Lineage Diagram is Even Worse!

I Want to...

- ...draw a data lineage network graph quickly
  - table-level lineages
  - column-level lineages
- ...update and rearrange an existing lineage network graph effortlessly
- ...use database and data tool metadata to help draw my lineage
- ...share my my lineage diagram easily

Cost matters, but I will not consider heavily during this presentation

Disclaimer

I respect and appreciate the effort of the developers that created the tools I mention on the following slides. Thanks to them I have a variety of choices, accessible with only a few clicks. The tiers on the following slides only consider the aspects of the software related to creating data lineages and do not assess the tools in their entirety.

Thank you developers! 🙏

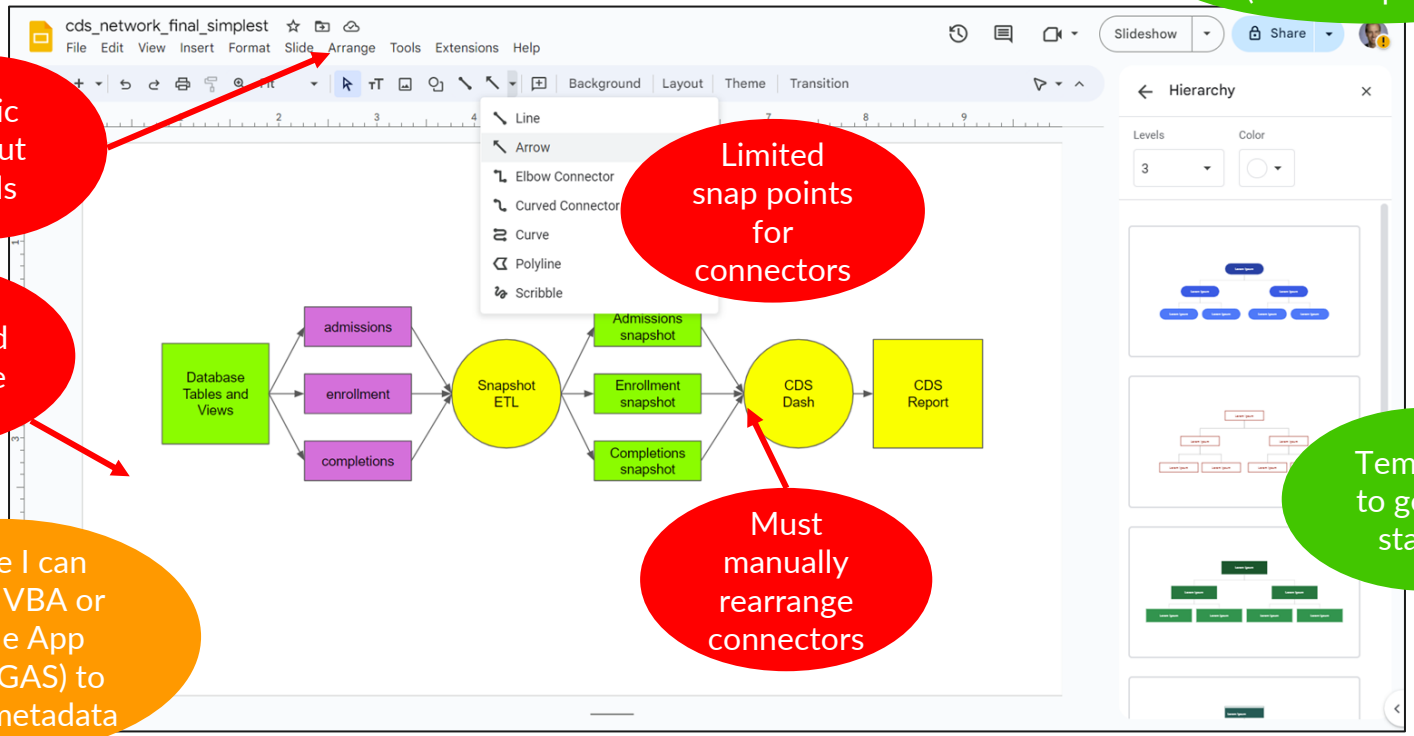
Tiers of Lineage Tools

My Tiers of Lineage Visualization Tools

Tier	Example Tools
5. Continuous lineage monitoring	Informatica, Collibra, Octopai
4. Automated metadata import, advanced automated layout	iGraph for R and Python, SQL Flow
3. Assisted drawing, advanced automated layout	yEd/yEd Live
2. Assisted drawing, basic automated layout	MS Visio, Draw.io, Lucidchart/LucidSpark
1. Manual drawing, manual layout	MS Office Apps, Google Apps

Tier 1
Manual drawing
Manual layout

Office Apps/Google Apps



Ubiquitous (rather important)

Basic layout tools

Limited snap points for connectors

Constrained to page size

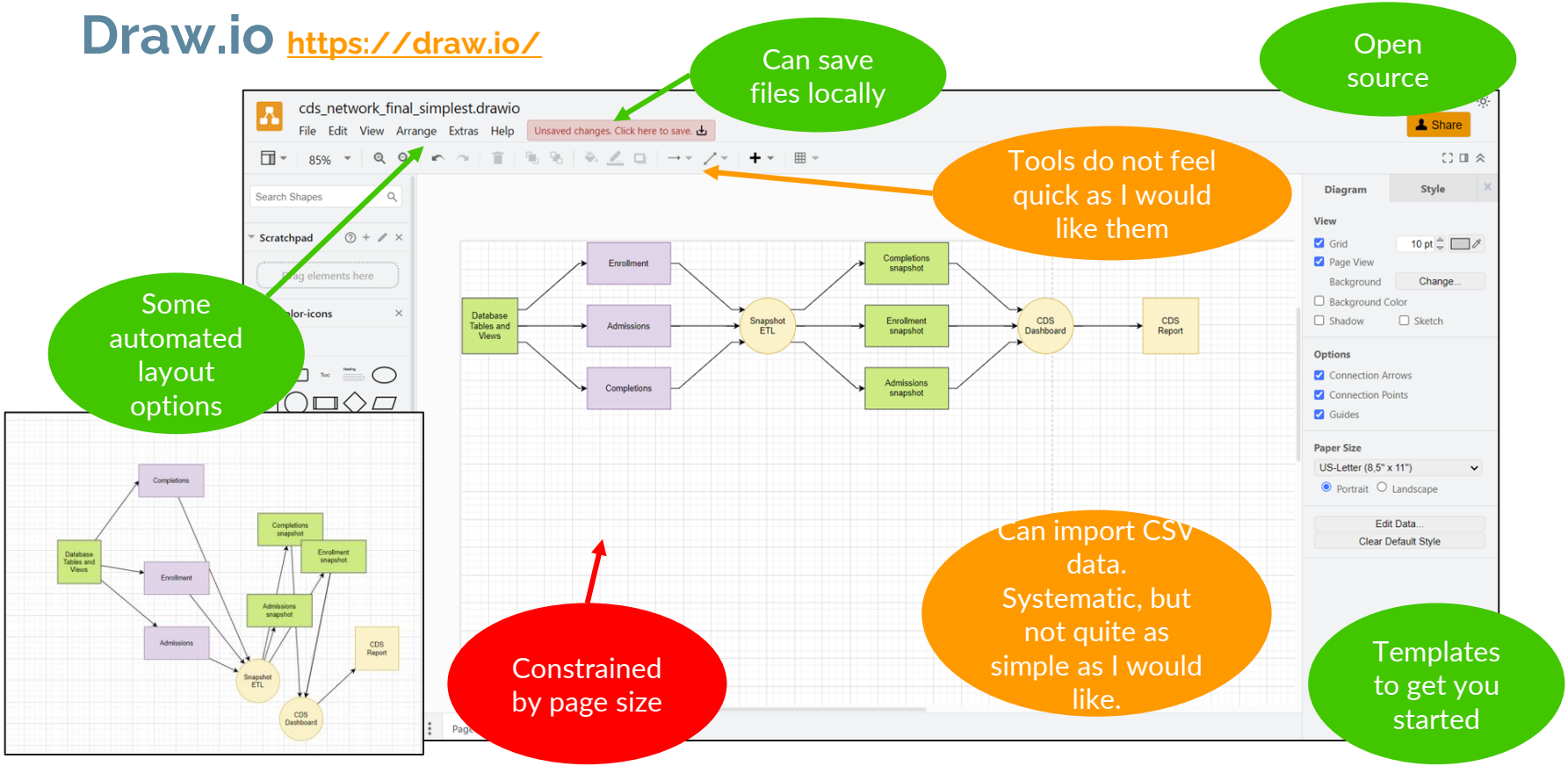
Templates to get you started

Must manually rearrange connectors

Maybe I can write a VBA or Google App Script (GAS) to import metadata

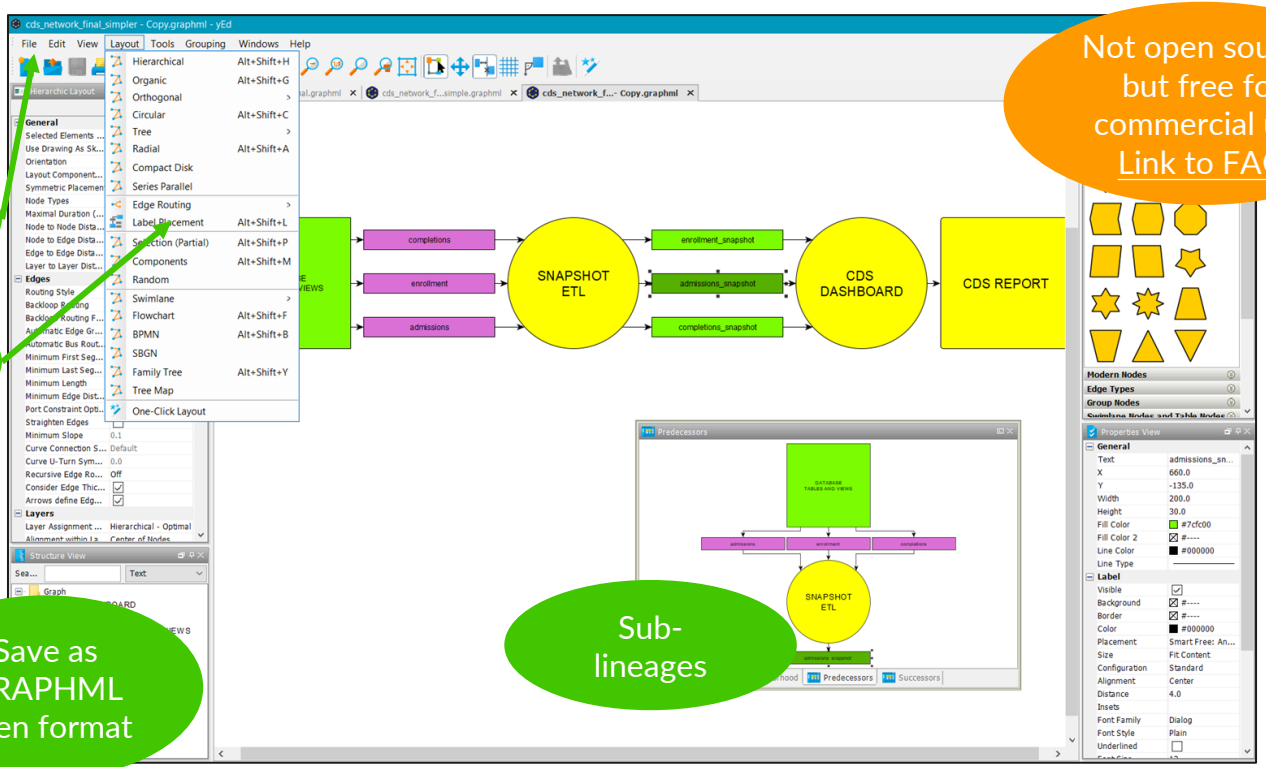
Tier 2
Assisted drawing
Basic automated layout

Draw.io https://draw.io/



Tier 3
Assisted drawing
Advanced automated layout

yEd Desktop/yEd Live https://www.yworks.com/yed-live/



Can import metadata

Advanced layout tools

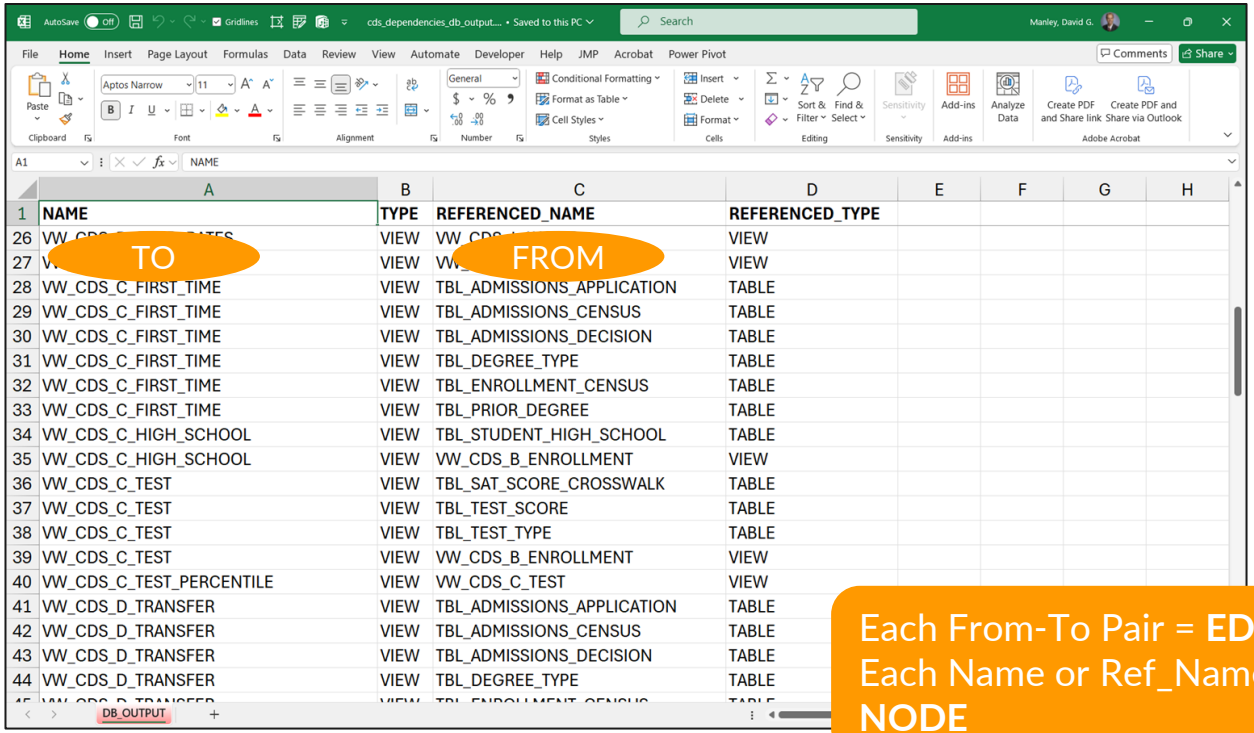
Save as GRAPHML open format

Sub-lineages

Not open source, but free for commercial use
Link to FAQ

cds_dependencies_db_output.xlsx

What do I consider a simple metadata export?



	A	B	C	D
1	NAME	TYPE	REFERENCED_NAME	REFERENCED_TYPE
26	VW_CDS_C_FIRST_TIME	VIEW	VW_CDS_B_ENROLLMENT	VIEW
27	VW_CDS_C_FIRST_TIME	VIEW	VW_CDS_C_TEST	VIEW
28	VW_CDS_C_FIRST_TIME	VIEW	TBL_ADMISSIONS_APPLICATION	TABLE
29	VW_CDS_C_FIRST_TIME	VIEW	TBL_ADMISSIONS_CENSUS	TABLE
30	VW_CDS_C_FIRST_TIME	VIEW	TBL_ADMISSIONS_DECISION	TABLE
31	VW_CDS_C_FIRST_TIME	VIEW	TBL_DEGREE_TYPE	TABLE
32	VW_CDS_C_FIRST_TIME	VIEW	TBL_ENROLLMENT_CENSUS	TABLE
33	VW_CDS_C_FIRST_TIME	VIEW	TBL_PRIOR_DEGREE	TABLE
34	VW_CDS_C_HIGH_SCHOOL	VIEW	TBL_STUDENT_HIGH_SCHOOL	TABLE
35	VW_CDS_C_HIGH_SCHOOL	VIEW	VW_CDS_B_ENROLLMENT	VIEW
36	VW_CDS_C_TEST	VIEW	TBL_SAT_SCORE_CROSSWALK	TABLE
37	VW_CDS_C_TEST	VIEW	TBL_TEST_SCORE	TABLE
38	VW_CDS_C_TEST	VIEW	TBL_TEST_TYPE	TABLE
39	VW_CDS_C_TEST	VIEW	VW_CDS_B_ENROLLMENT	VIEW
40	VW_CDS_C_TEST_PERCENTILE	VIEW	VW_CDS_C_TEST	VIEW
41	VW_CDS_D_TRANSFER	VIEW	TBL_ADMISSIONS_APPLICATION	TABLE
42	VW_CDS_D_TRANSFER	VIEW	TBL_ADMISSIONS_CENSUS	TABLE
43	VW_CDS_D_TRANSFER	VIEW	TBL_ADMISSIONS_DECISION	TABLE
44	VW_CDS_D_TRANSFER	VIEW	TBL_DEGREE_TYPE	TABLE

Each From-To Pair = **EDGE**
Each Name or Ref_Name = **NODE**

Process to Draw Lineage from Metadata with yEd

Export data from database or data tool

Reshape metadata for yEd (nodes and edges)

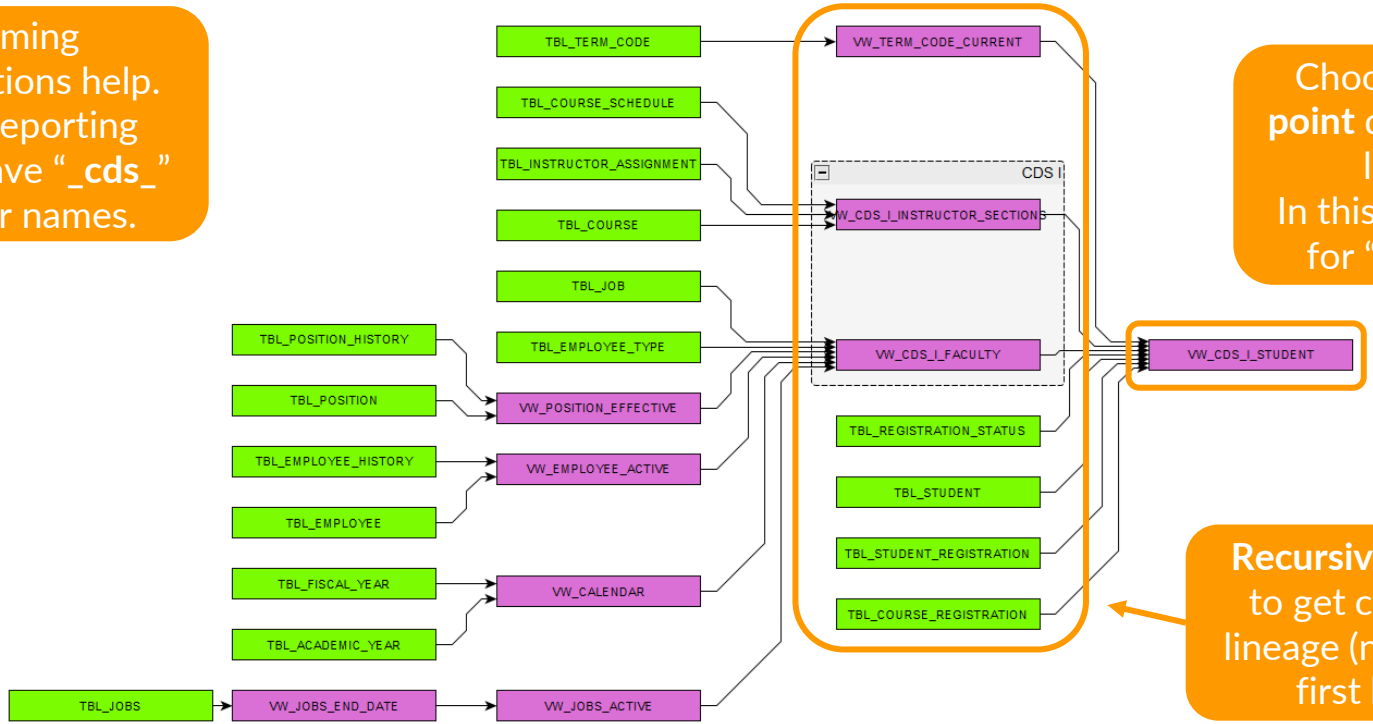
Map node and edge columns to yEd fields

Map additional properties to yEd fields

Layout and customize diagram

STEP 1: A Quick Note About Dependency Queries

Naming conventions help. Here reporting views have "**_cds_**" in their names.



Choose **starting point** deep into the lineage.
In this case search for "**vw_cds%**"

Recursively query to get complete lineage (no just the first layer)

cds_dependencies_postgres.sql
cds_dependencies_oracle.sql

STEP 1: Export Data from Database or Data Tool

Oracle

```
/* Oracle
   Retrieve CDS views and their dependencies.
   Get the dependencies of all sub-views and until you reach a base table.
*/

/* View dependencies */
SELECT DISTINCT name
     , type
     , referenced_name
     , referenced_type
  START WITH LOWER(name) LIKE '%vw_cds%' /* Starting point = objects that contain "vw_
CONNECT BY NOCYCLE PRIOR referenced_name = NAME
               AND PRIOR referenced_owner = OWNER

  UNION

/* Tables with "tbl_cds" in their names, but do not have database dependencies (i.e.,
SELECT table_name AS name
     , 'TABLE' AS type
     , NULL AS referenced_name
     , NULL AS referenced_type
  FROM all_tables
 WHERE LOWER(table_name) LIKE '%tbl_cds%'
```

PostgreSQL

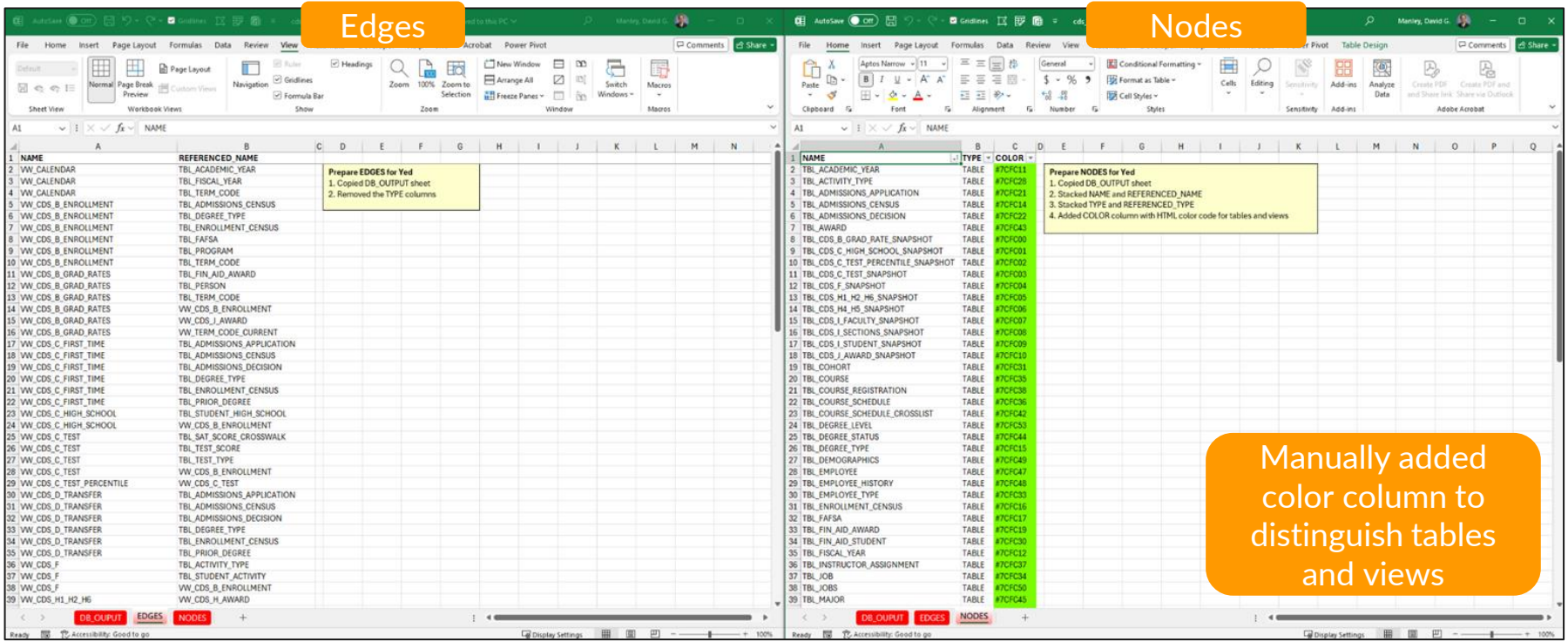
```
/* PostgreSQL
   Retrieve CDS views and their dependencies.
   Get the dependencies of all sub-views and until you reach a base table.
*/
WITH RECURSIVE lineage AS (
    SELECT a.*
      FROM information_schema.view_table_usage AS a
     WHERE LOWER(a.view_name) LIKE '%vw_cds%' /* Starting point = objects th

    UNION ALL

    SELECT b.*
      FROM information_schema.view_table_usage AS b
           JOIN lineage AS x
             ON x.table_schema = b.view_schema
            AND x.table_name = b.view_name
```

Each DBMS has different syntax for querying table and view dependencies. I want recursive dependencies so that we get all related objects (not just the first layer).

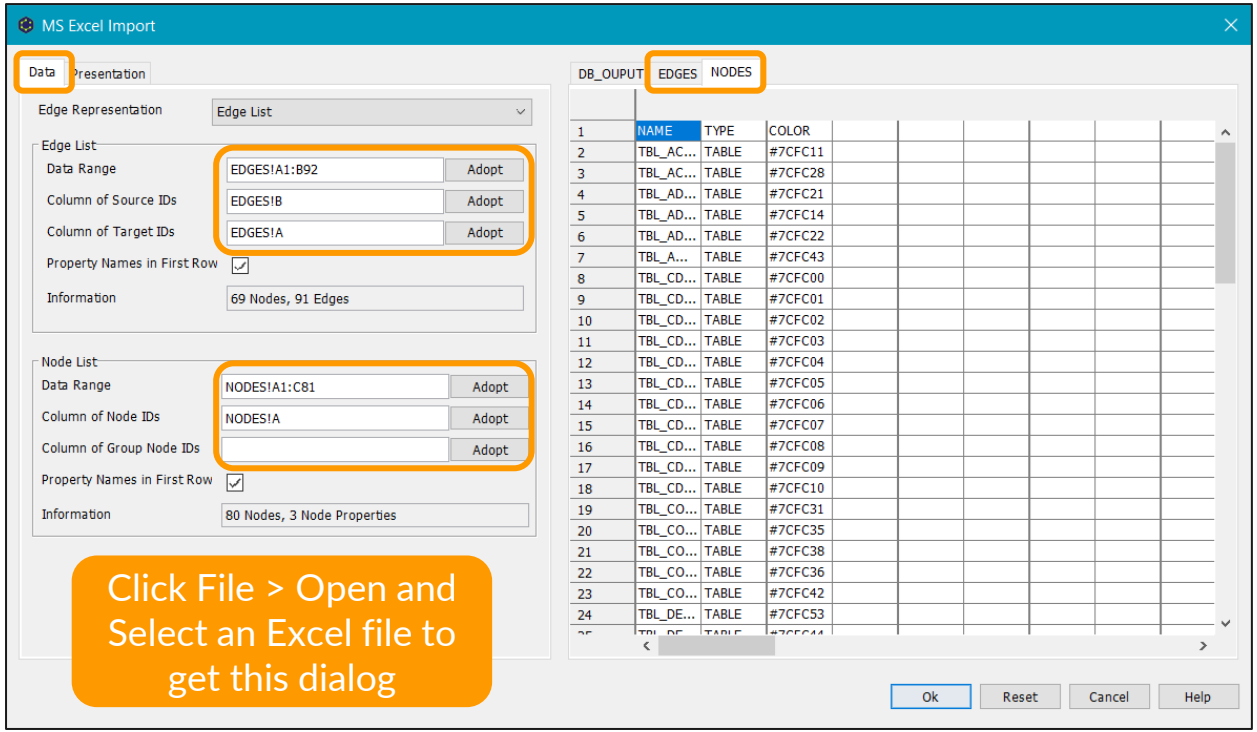
STEP 2: Reshape metadata for yEd

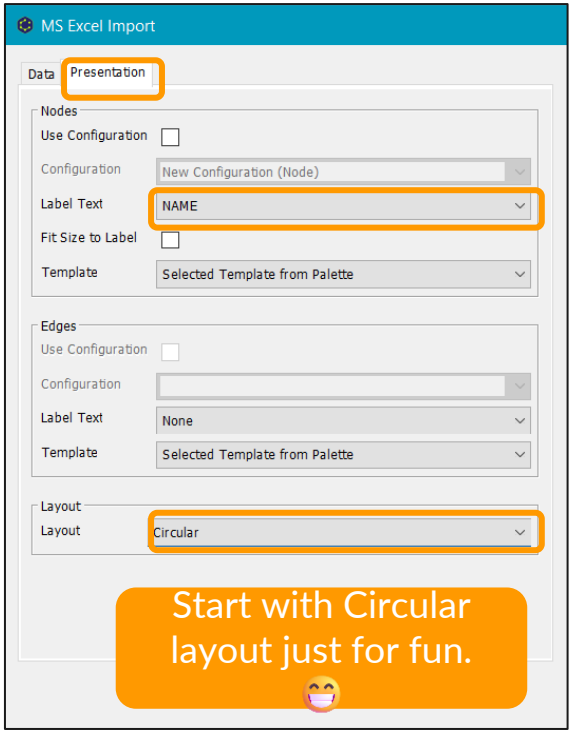
Edges

Nodes

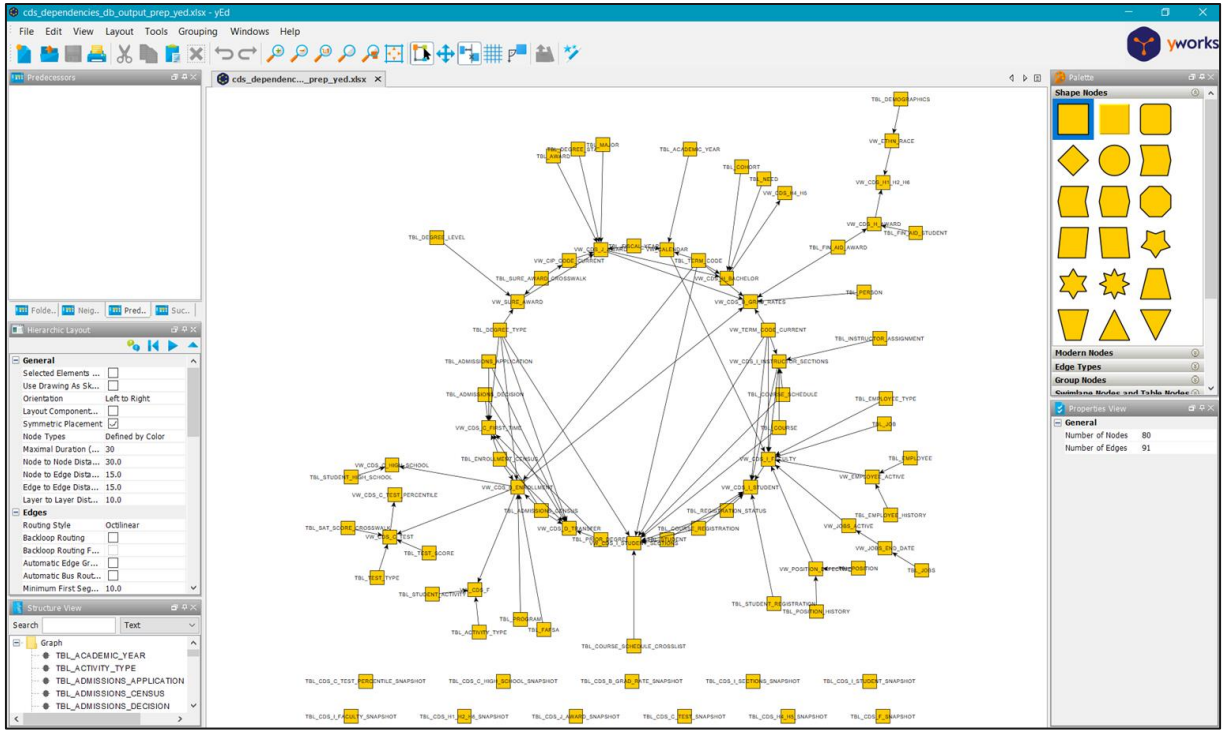


STEP 3: Map node and edge columns to yEd fields

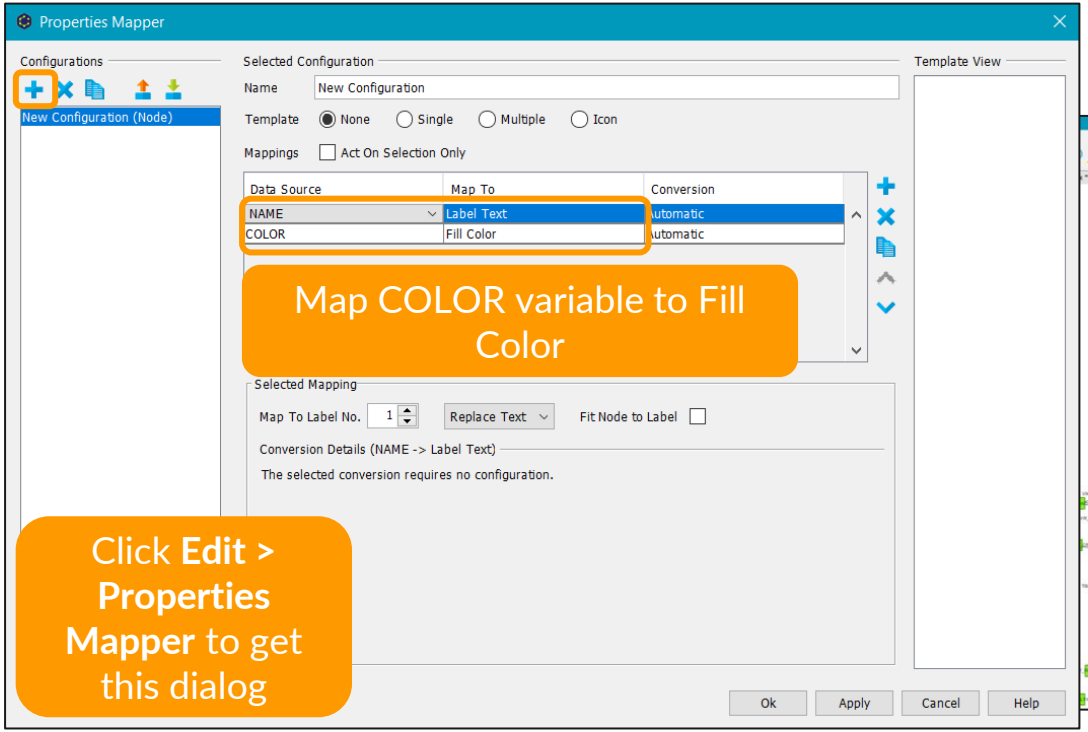


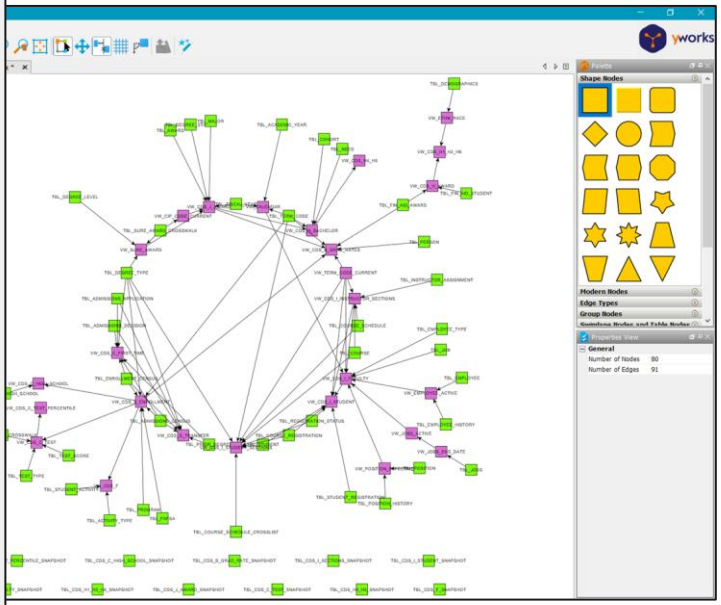


STEP 3: Results So Far



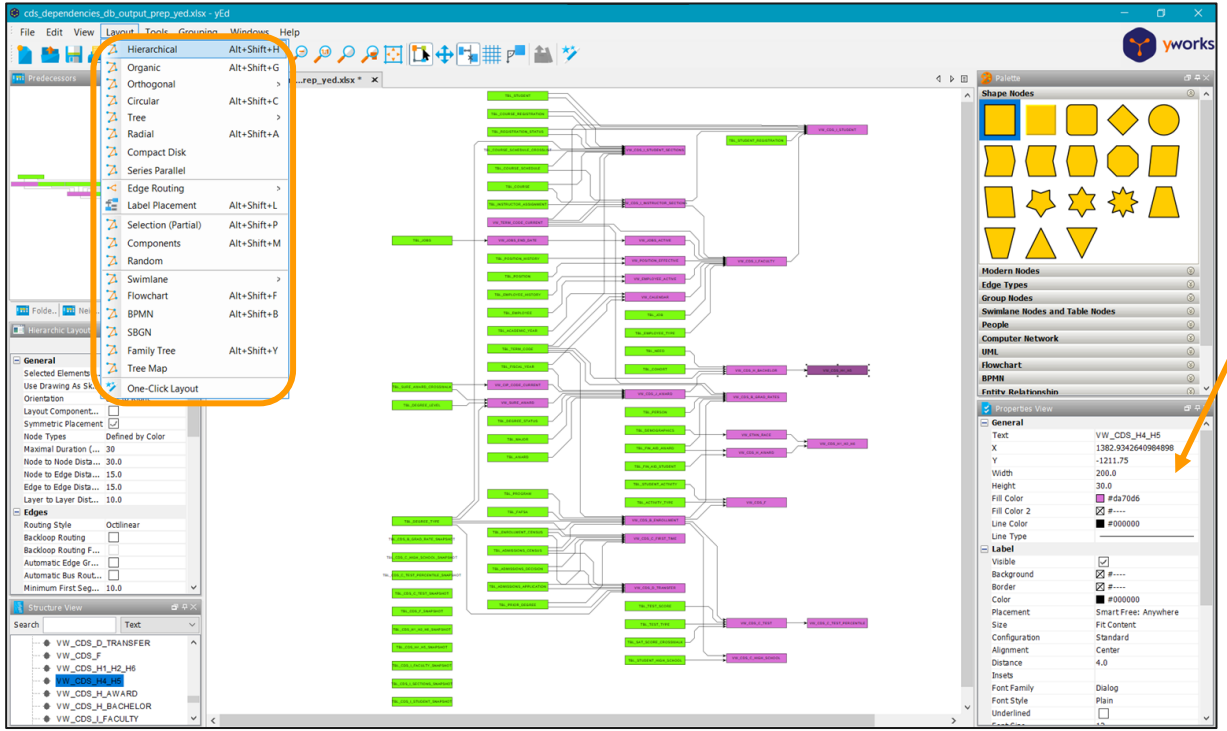
STEP 4: Map additional properties to yEd fields





STEP 5: Layout and customize diagram

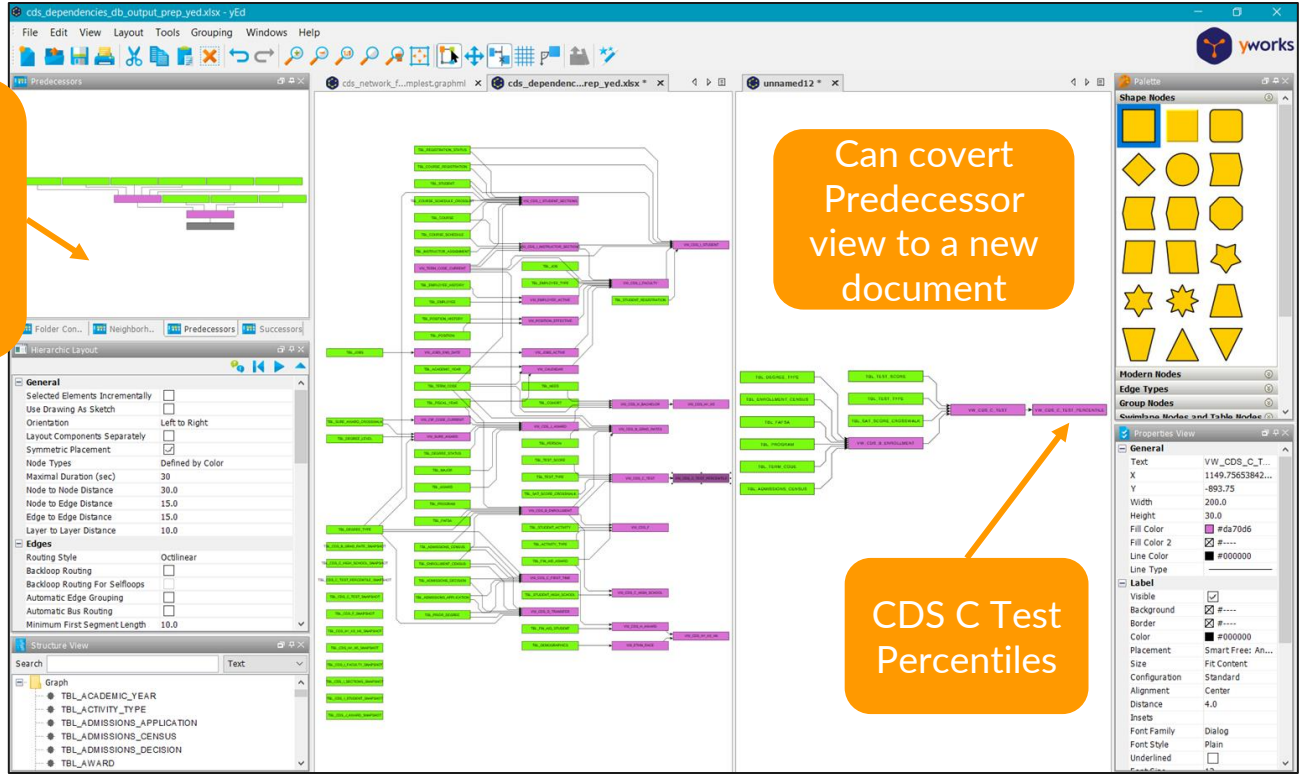
Layout menu has many choices. Used **Hierarchical** here.



TIP: Select a single node a press CTRL+A to select all nodes. Can then edit properties of all nodes without affecting edges.

STEP 5: Sub Lineages

Predecessors and Successors view

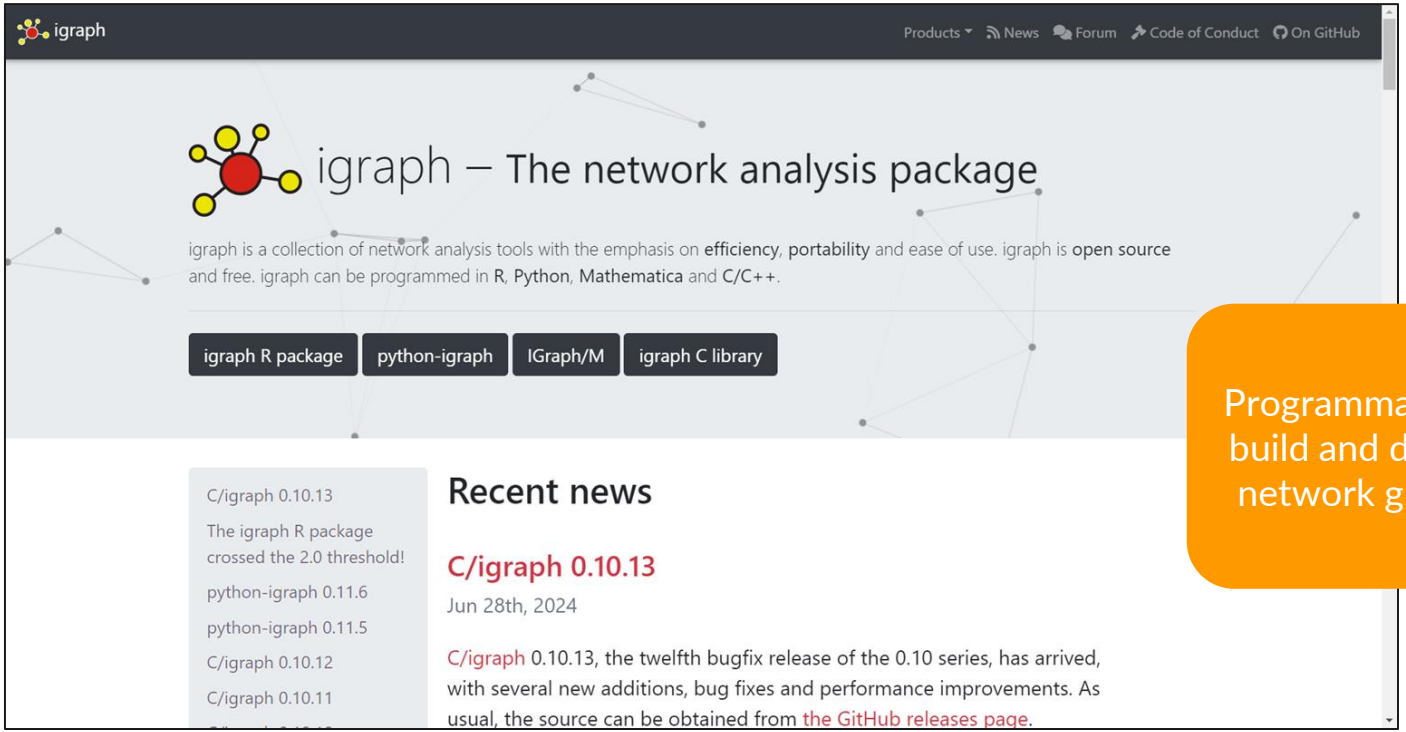


Can covert Predecessor view to a new document

CDS C Test Percentiles

Tier 4
Automated metadata import
Advanced automated layout

iGraph https://igraph.org/



Programmatically build and display network graphs

Process to Draw Lineage from Metadata with iGraph

STEP A | STEP B | STEP C

R/Python Script

1. Get metadata directly from database
2. Parse metadata for edges
3. Parse metadata for nodes
4. Use iGraph package to create network
5. Colorize and plot network
6. Export to GRAPHML

Map GRAPHML to yEd fields

Layout and customize diagram

Why I am still using yEd?

I find it has better layouts and often I need to manually customize something.

dependency_network_script_r.R
dependency_network_script_py.py

R/Python Scripts

R

```r
library(readxl)      #Import Excel files
library(RODBC)       #Connect to database
library(tidyverse)   #Data manipulations
library(igraph)      #Create the network data structure
library(visNetwork)  #Visualize the network interactively

# 1. Read network components Excel file
df_components <- read_excel(file.choose())

# #OPTIONAL: 1. READ DIRECTLY FROM DATABASE
# #Query for network components (edges and nodes)
# sql_components  <- "
# /* View dependencies */
#  SELECT DISTINCT name
#       , type
#       , referenced_name
#       , referenced_type
#    FROM all_dependencies
#   START WITH LOWER(name) LIKE '%vw_cds%'
# CONNECT BY NOCYCLE PRIOR referenced_name = NAME
#                AND PRIOR referenced_owner = OWNER
#
#   UNION
#
# /* Stand-alone tables */
#  SELECT table_name AS name
#       , 'TABLE' AS type
#       , NULL AS referenced_name
#       , NULL AS referenced_type
#    FROM all_tables
#   WHERE LOWER(table_name) LIKE '%tbl_cds%'
# ;"
#
# #Open database connection, query components, close database connection
# channel <- odbcConnect("<DATABASE DSN>", uid="<USERID>", pwd="<PASSWORD>")
# df_components <- sqlQuery(channel, sql_components)
# odbcClose(channel)

# 2. Parse the edges from the component data set
df_edge <- df_components %>%
  filter(!is.na(REFERENCED_NAME)) %>%
  select(REFERENCED_NAME, NAME)

# 3. Parse the nodes from the component data set
df_node <- df_components %>%
  select(NAME, TYPE) %>%
  dplyr::union(df_components %>%
```

Python

```python
from tkinter.filedialog import askopenfilename, asksaveasfilename
import pandas as pd
import numpy as np
import igraph as ig
import matplotlib.pyplot as plt

# 1. Read network components Excel file
filename = askopenfilename()
df_components = pd.read_excel(filename)

# 2. Parse the edges from the component data set
df_edge = df_components.dropna() [["REFERENCED_NAME", "NAME"]]

# 3. Parse the nodes from the component data set
df_n1 = df_components[["NAME", "TYPE"]]
df_n2 = df_components[["REFERENCED_NAME", "REFERENCED_TYPE"]]
df_n2 = df_n2.rename(columns = {"REFERENCED_NAME" : "NAME", "REFERENCED_TYPE" : "TYPE"})
df_n2 = df_n2.dropna()
df_node = pd.concat([df_n1, df_n2])
df_node = df_node.drop_duplicates()

# 3a. Convert TYPE variable from text to a number for each node
df_node["TYPE_NUMBER"] = np.where(df_node["TYPE"] == "TABLE", 1,
                         np.where(df_node["TYPE"] == "VIEW", 2,
                         np.where(df_node["TYPE"] == "FUNCTION", 3, 4)))

# 4. Create the network
net = ig.Graph(directed = True)
net.add_vertices(df_node["NAME"].tolist())
net.add_edges(list(zip(df_edge["REFERENCED_NAME"], df_edge["NAME"])))
net.vs["label"] = df_node["NAME"].tolist()

# 5. Define colors for nodes (V) and edges (E)
type_colrs = { 1: "#7CFC00",
               2: "#DA7006",
               3: "#fc9e23",
               4: "#4169E1" }

colrs = [type_colrs[x] for x in df_node["TYPE_NUMBER"].tolist()]
net.vs['color'] = colrs

# 5a. Plot a static image of the network
fig, ax = plt.subplots()
```

CAUTION!
Be careful of database passwords stored in scripts!

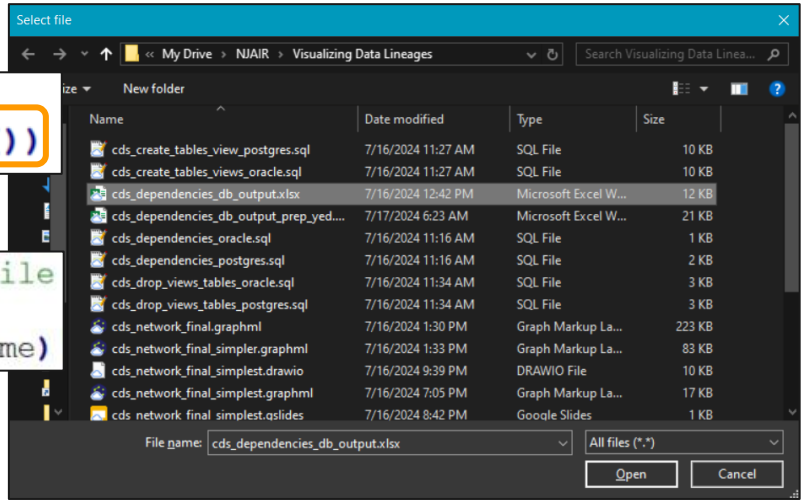
Forgive my Python…I am an R guy. 😁

STEP 1: Run the Script

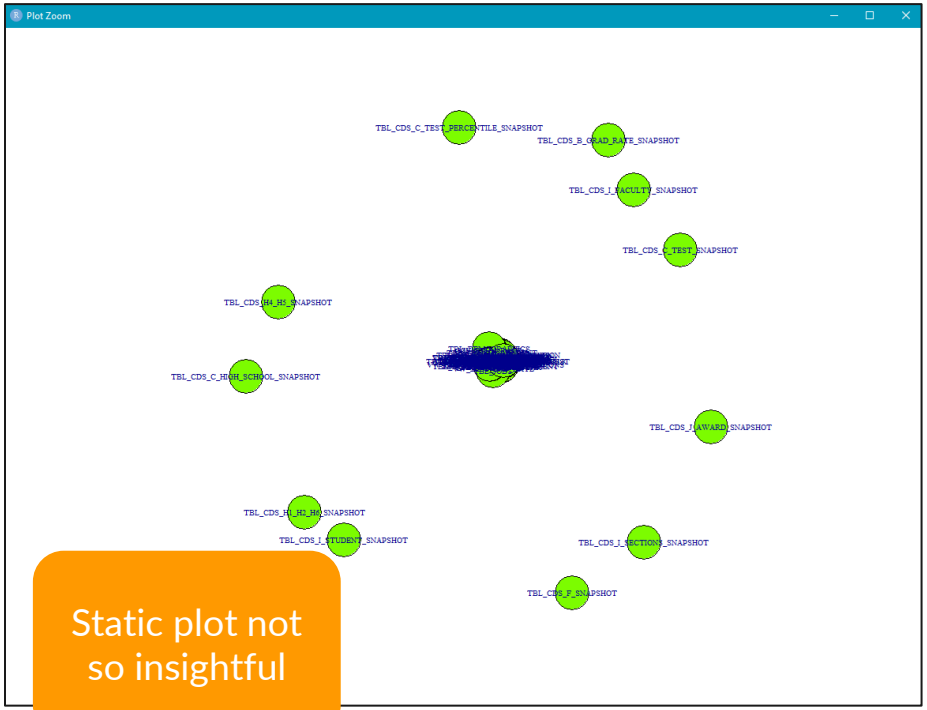
- For the sake of security, I will use Excel exports in this presentation and not db connections.
- The example scripts are meant to be **stepped through** not run all at once.
- **File dialogs** rather than file paths are coded to make file selection a more convenient.

```
# 1. Read network components Excel file
df_components <- read_excel(file.choose())
```

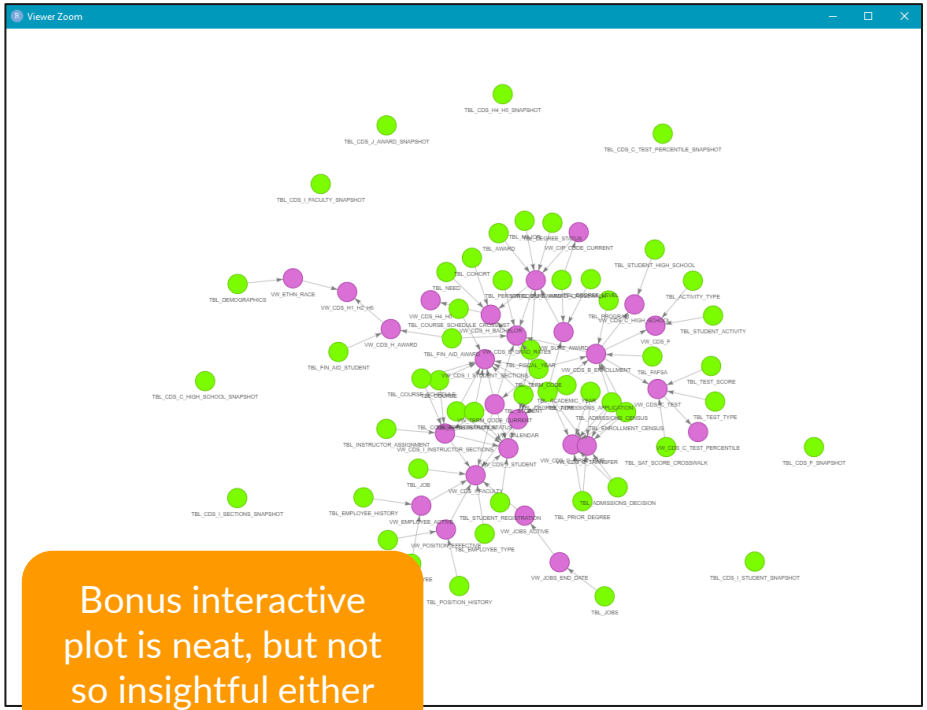
```
# 1. Read network components Excel file
filename = askopenfilename()
df_components = pd.read_excel(filename)
```



STEP 1: Plot Output



Static plot not so insightful



Bonus interactive plot is neat, but not so insightful either

cds_network_r_output.graphml
cds_network_py_output.graphml

STEP 1: Send iGraph to GRAPHML File

Text Editor

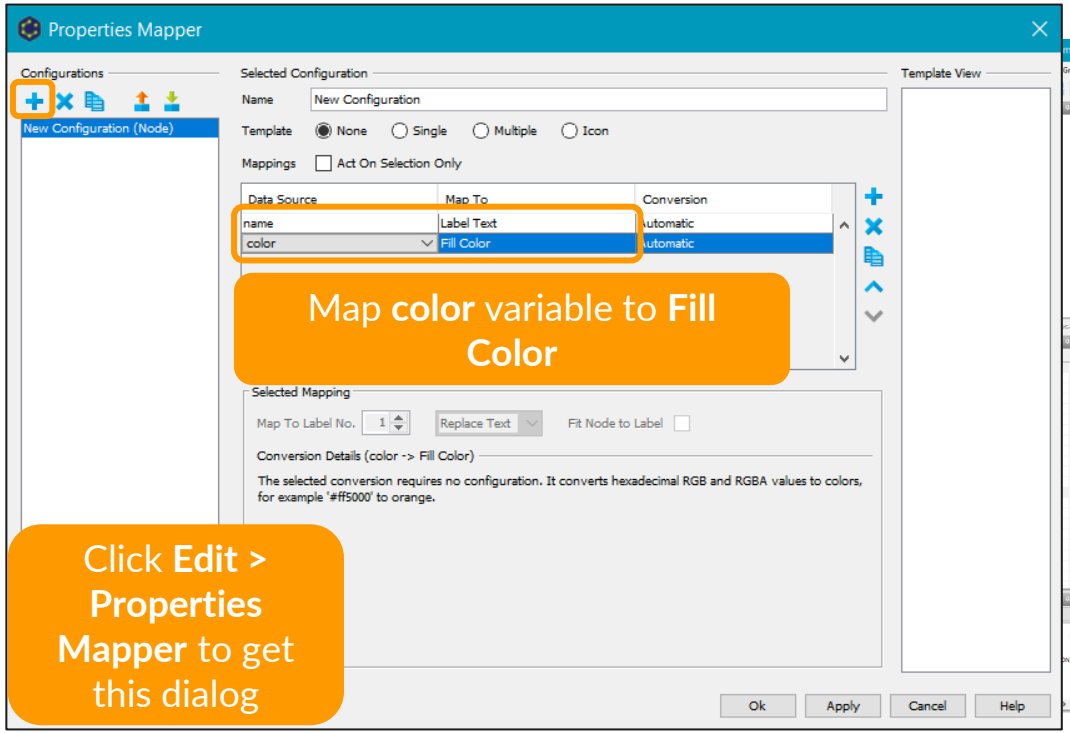
```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
         http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<!-- Created by igraph -->
  <key id="v_name" for="node" attr.name="name" attr.type="string"/>
  <key id="v_TYPE" for="node" attr.name="TYPE" attr.type="string"/>
  <key id="v_TYPE_NUMBER" for="node" attr.name="TYPE_NUMBER" attr.type="double"/>
  <key id="v_color" for="node" attr.name="color" attr.type="string"/>
  <key id="e_color" for="edge" attr.name="color" attr.type="string"/>
  <graph id="G" edgedefault="directed">
    <node id="n0">
      <data key="v_name">TBL_CDS_B_GRAD_RATE_SNAPSHOT</data>
      <data key="v_TYPE">TABLE</data>
      <data key="v_TYPE_NUMBER">1</data>
      <data key="v_color">#7CFC00</data>
    </node>
    <node id="n1">
      <data key="v_name">TBL_CDS_C_HIGH_SCHOOL_SNAPSHOT</data>
      <data key="v_TYPE">TABLE</data>
      <data key="v_TYPE_NUMBER">1</data>
      <data key="v_color">#7CFC00</data>
    </node>
    <node id="n2">
      <data key="v_name">TBL_CDS_C_TEST_PERCENTILE_SNAPSHOT</data>
      <data key="v_TYPE">TABLE</data>
      <data key="v_TYPE_NUMBER">1</data>
      <data key="v_color">#7CFC00</data>
    </node>
    <node id="n3">
      <data key="v_name">TBL_CDS_C_TEST_SNAPSHOT</data>
      <data key="v_TYPE">TABLE</data>
      <data key="v_TYPE_NUMBER">1</data>
      <data key="v_color">#7CFC00</data>
    </node>
    <node id="n4">
      <data key="v_name">TBL_CDS_F_SNAPSHOT</data>
      <data key="v_TYPE">TABLE</data>
```

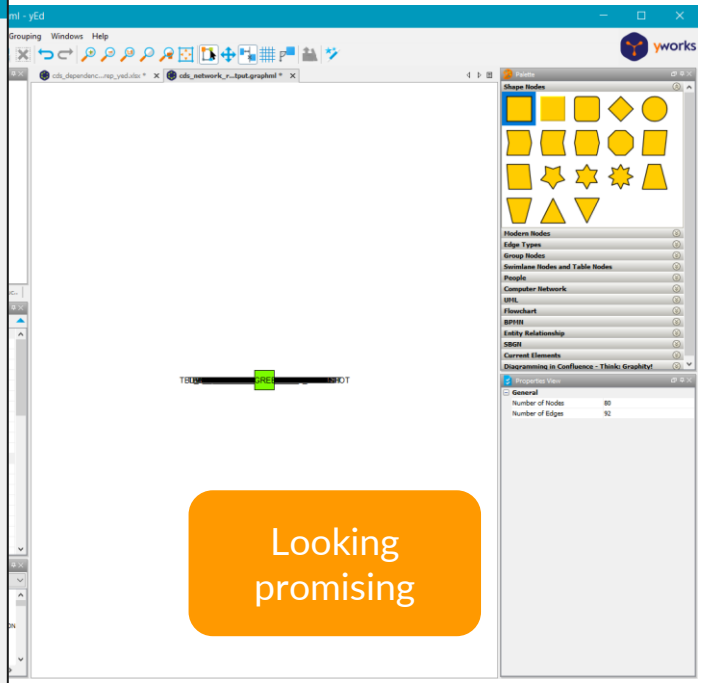
GRAPHML is just a type of XML file

yEd

Hello? Is my data lineage there?

STEP 2: Map GRAPHML to yEd fields

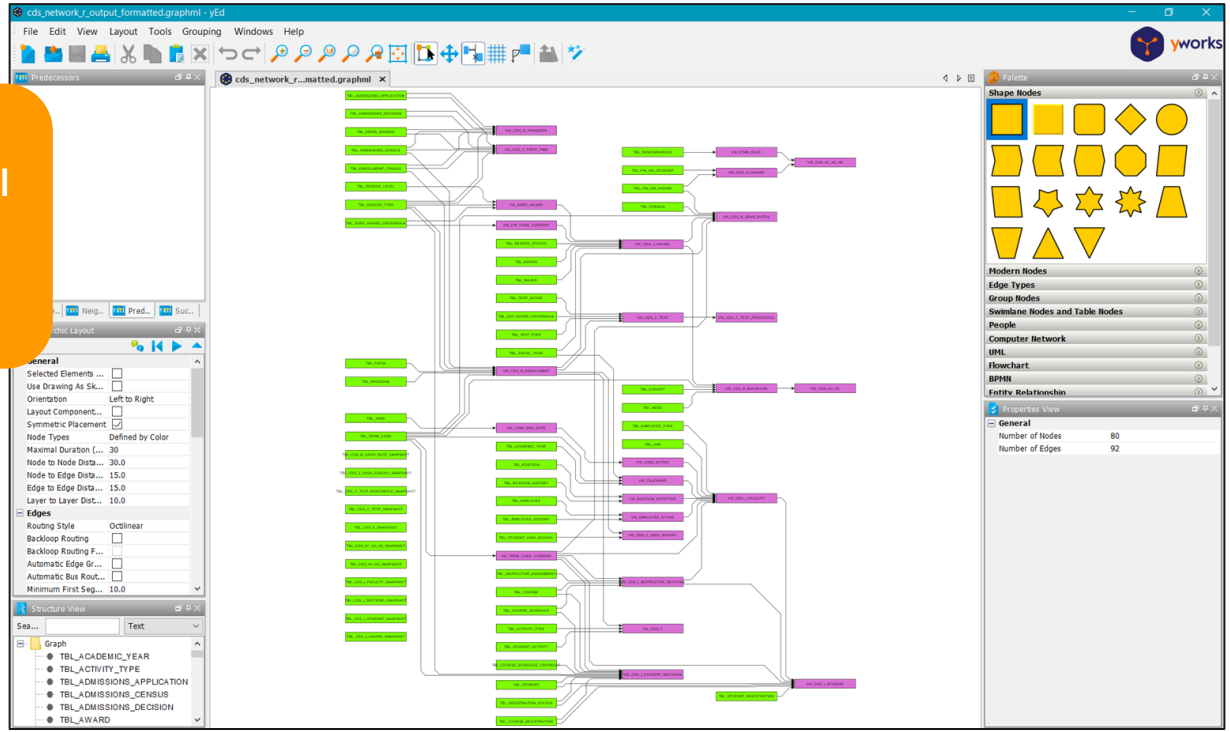




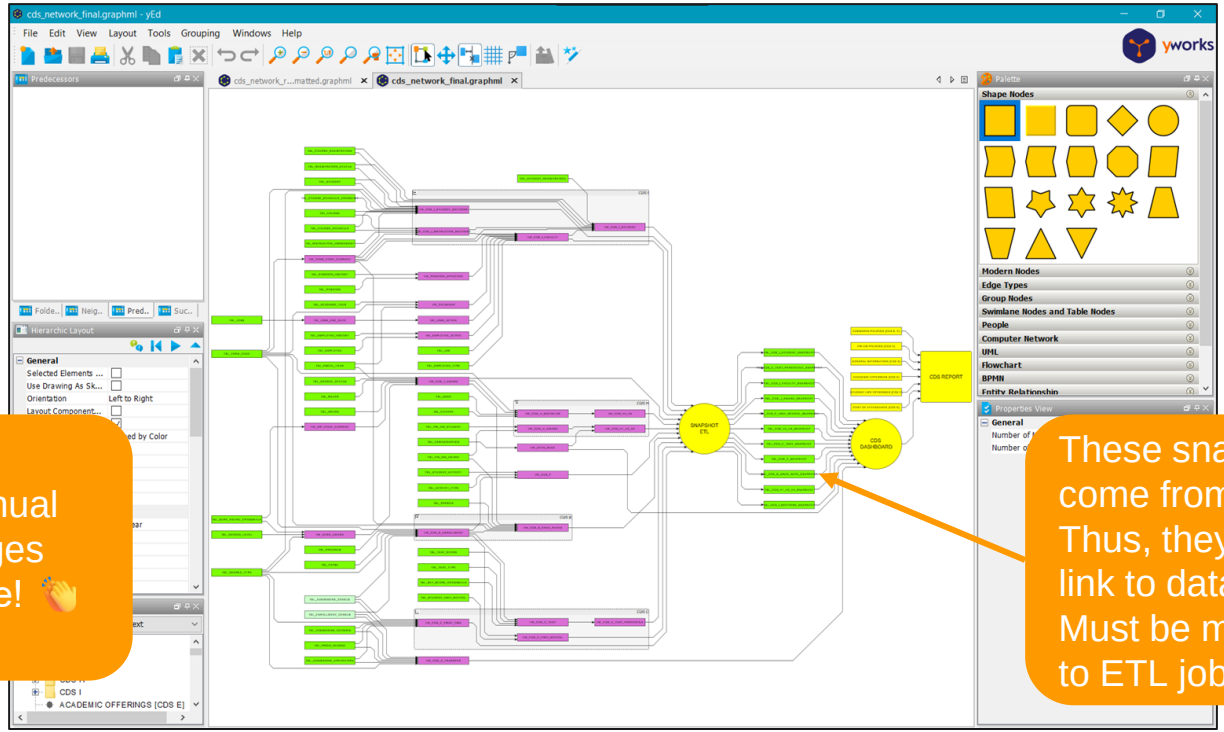
cds_network_r_output_formatted.graphml
cds_network_py_output_formatted.graphml

STEP 3: Layout and customize diagram

Chose **Hierarchical** from the Layout menu.



STEP 3: Layout and customize diagram



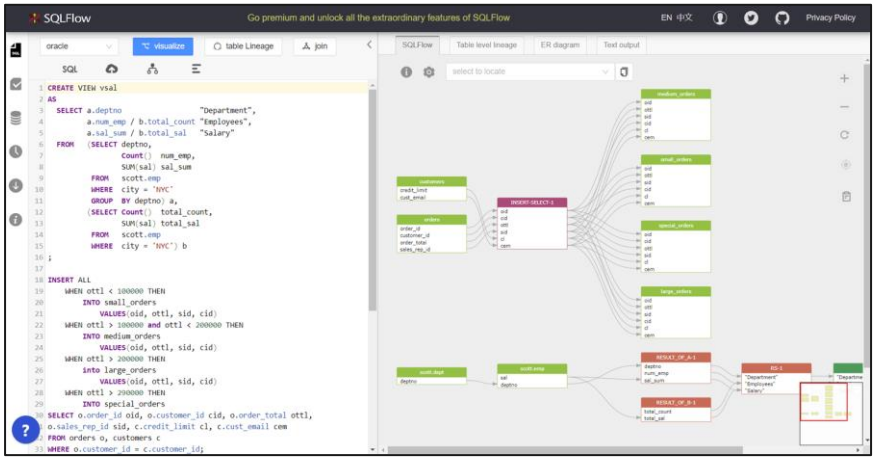
Add some manual nodes and edges and we're done! 👏

These snapshot tables come from an ETL job. Thus, they do not auto-link to database views. Must be manually linked to ETL job node.

Before we leave Tier 4: SQL FLow

https://sqlflow.gudusoft.com/#/

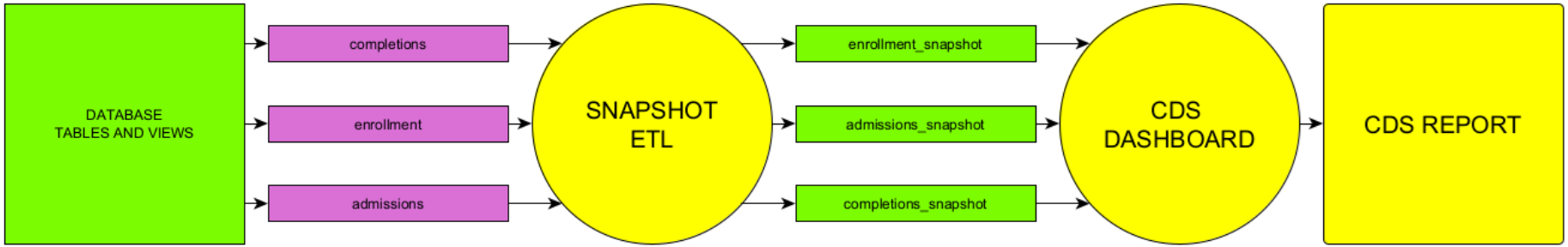
- There are times when I want to look at 500+ lines SQL code and understand the data lineage within that file.
- I have not found solid free or open source tools that address this situation, but SQL Flow (a commercial tool) looks like it could.
- My institution has not purchased this tool, nor does it intend to. I have no real experience with it…just passing along what I have seen on my data lineage journey.



Tier 5
Continuous lineage monitoring

What Have We Done To This Point?

Essentially we have built a scanner the retrieves the metadata from a single database and displays it as a network diagram.



Can I handle multiple database and database links?

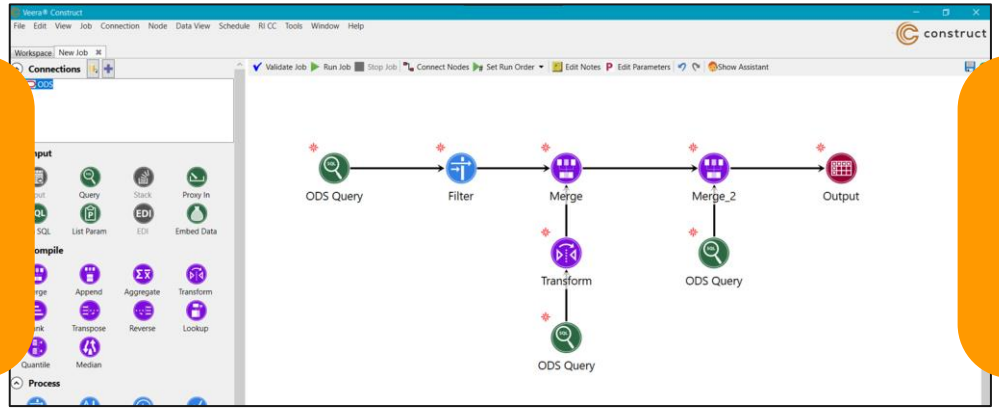
Can I parse the metadata lineage in my ETL tool?

Can I parse the metadata lineage in my Dashboard?

A Question I Regularly Think About

- **At what layer in my report lineage do I make a transformation/calculation?**
  - It would be convenient if we only had one layer or one tool
  - Our ETL tool is very visual…unfortunately it has limitations

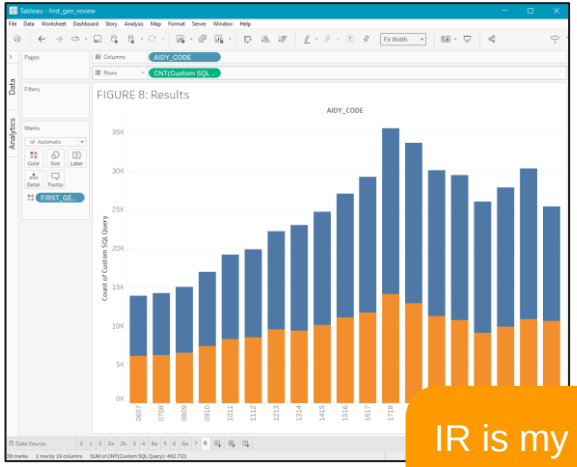
We Use Rapid Insight Veera Construct for ETL https://eab.com/solutions/rapid-insight/



Funny note: My color scheme for tables and views in the presentation is based the colors in Veera Construct.

Parsing Dashboard

Tableau workbooks are essentially XML files. Why don't parse those and add to my lineage detail?

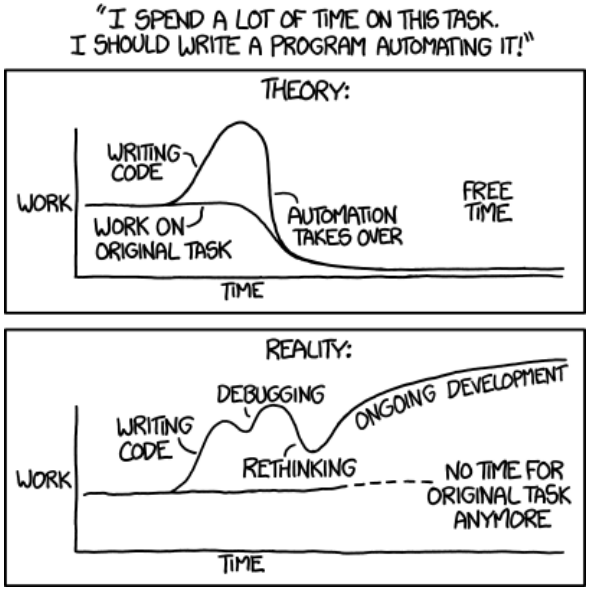


IR is my primary job…not taking apart Tableau 🔨

What Am I Still Missing

- Cross-platform data lineage
- Column-level data lineage
- Linking my technical data terms to business data terms

Automation



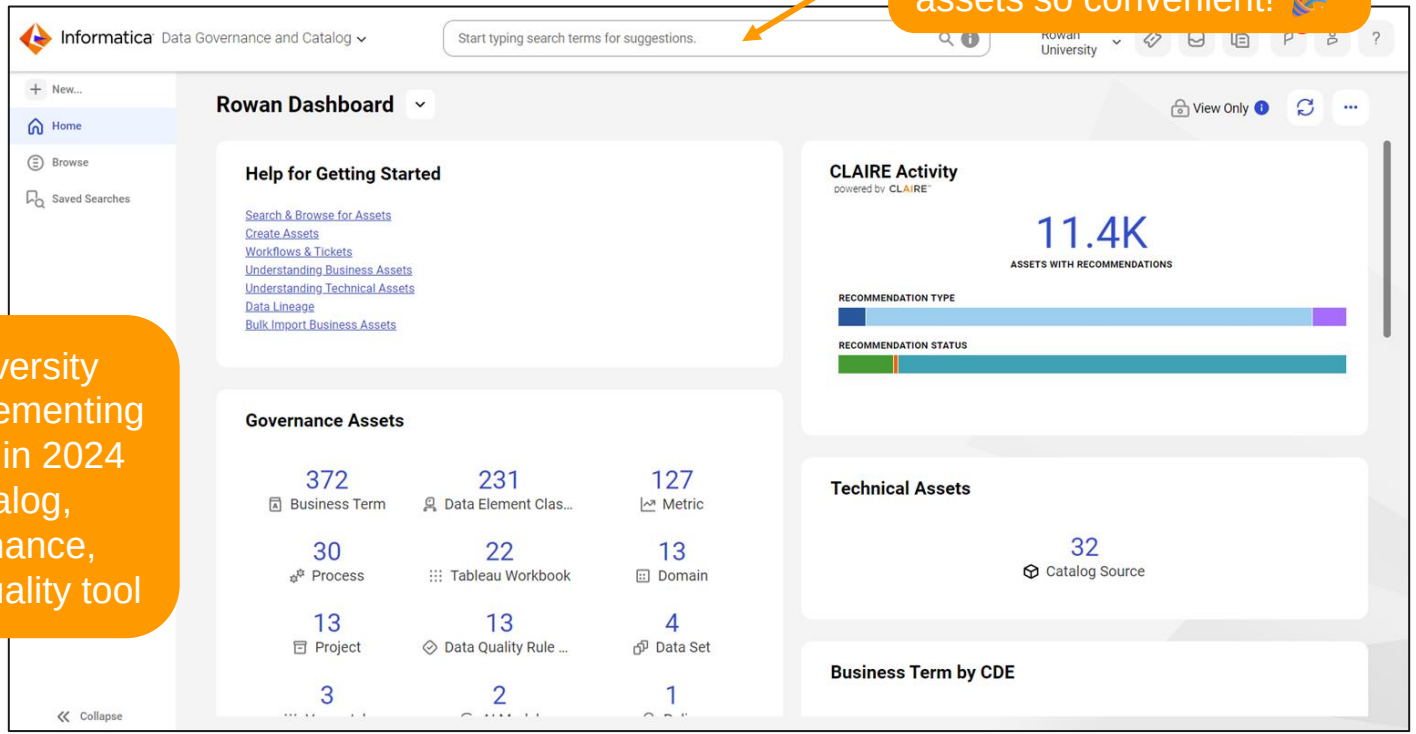
https://xkcd.com/1319/
xkcd by Randall Munroe

It can be a full-time job picking apart

METADATA

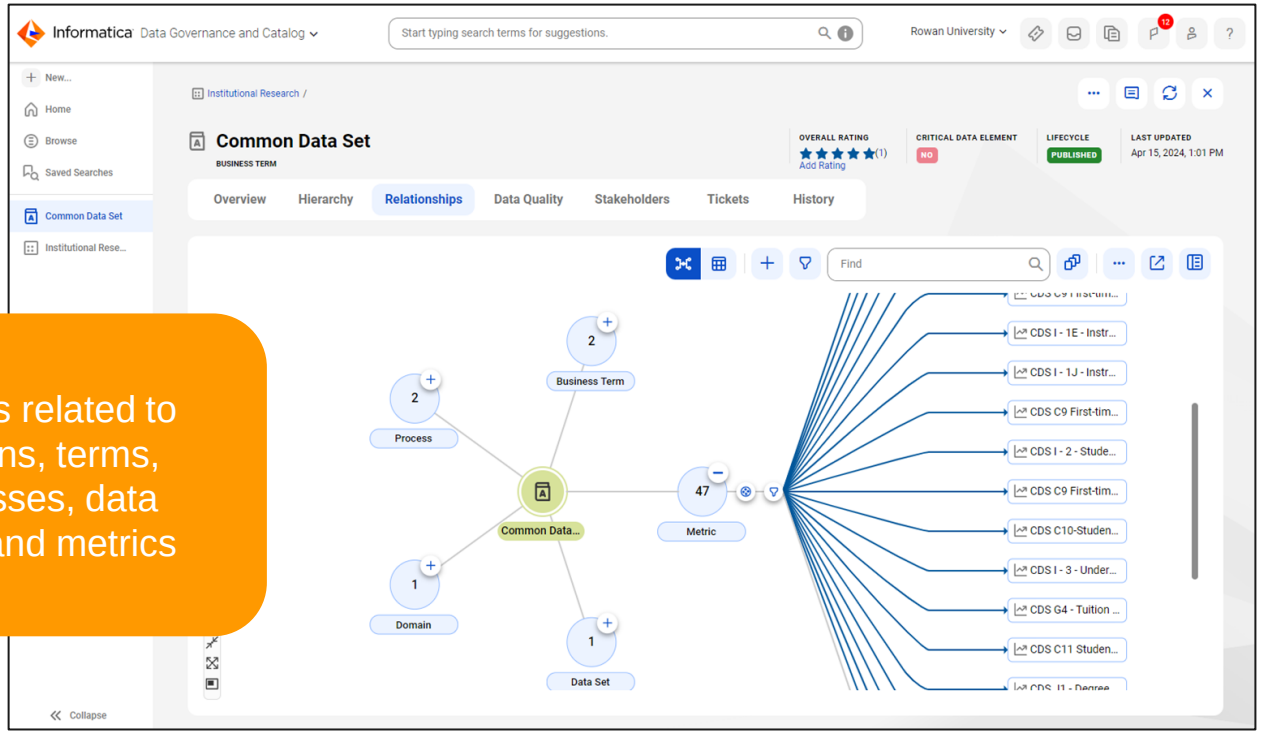
Informatica

Google-like search bar makes finding data assets so convenient! 🎉



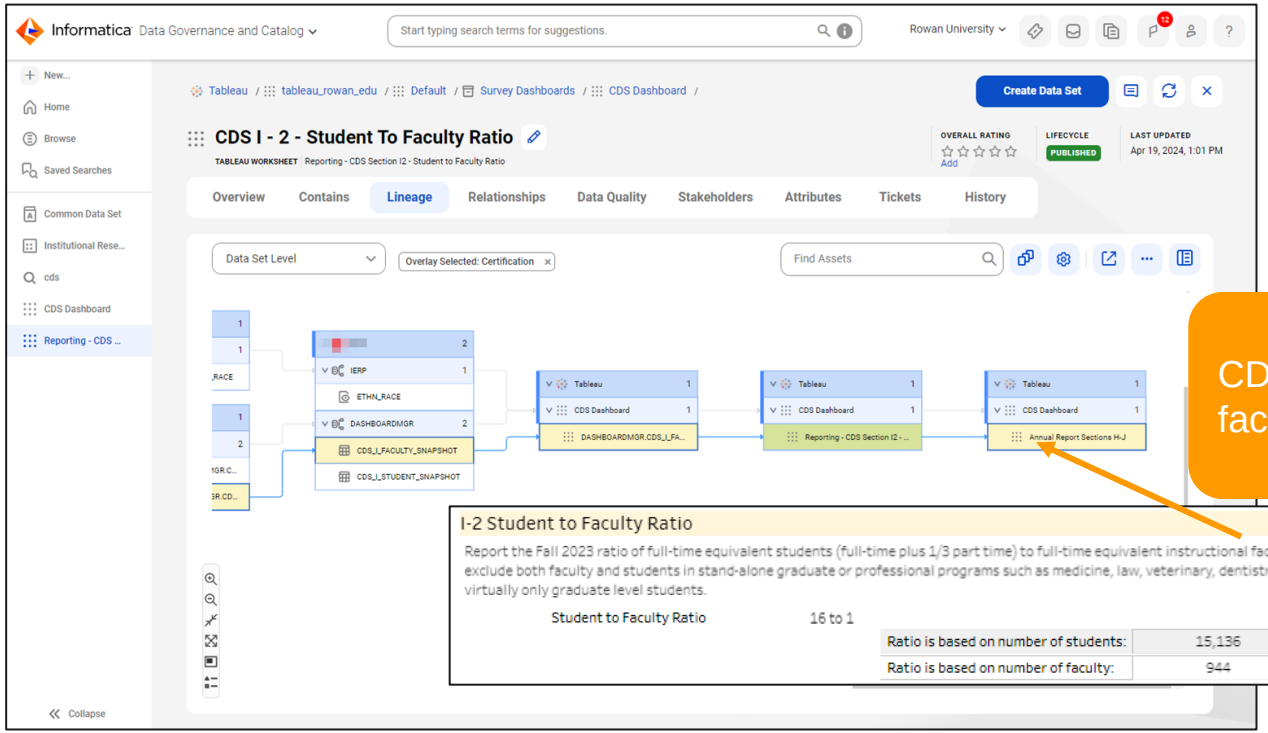
Rowan University began implementing Informatica in 2024 as data catalog, data governance, and data quality tool

Informatica: Common Data Set Asset Relationships



CDS is related to domains, terms, processes, data sets, and metrics

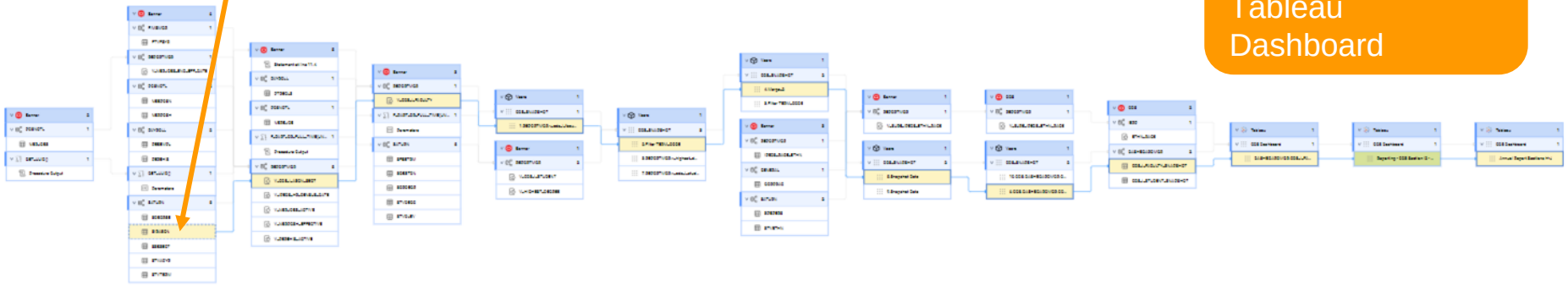
Informatica: CDS I2 Lineage from Tableau Dashboard



Informatica: CDS I2 Lineage Expanded

Yellow represents propagation of faculty assignment through lineage

CDS I2 Student-to-faculty ratio in Tableau Dashboard



Database

ETL Lineage

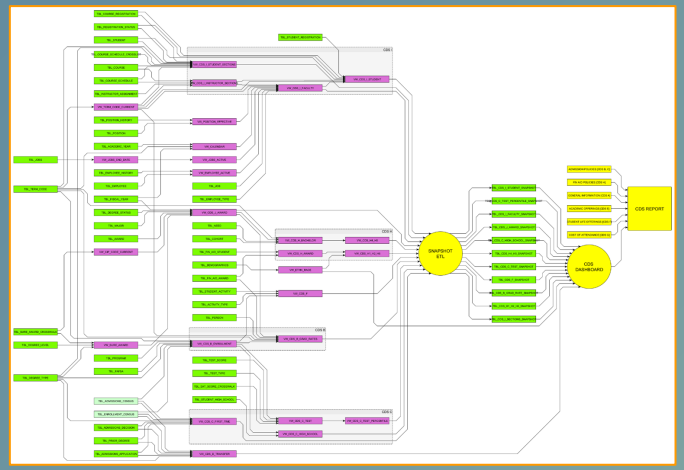
Snapshot Table

Tableau Lineage

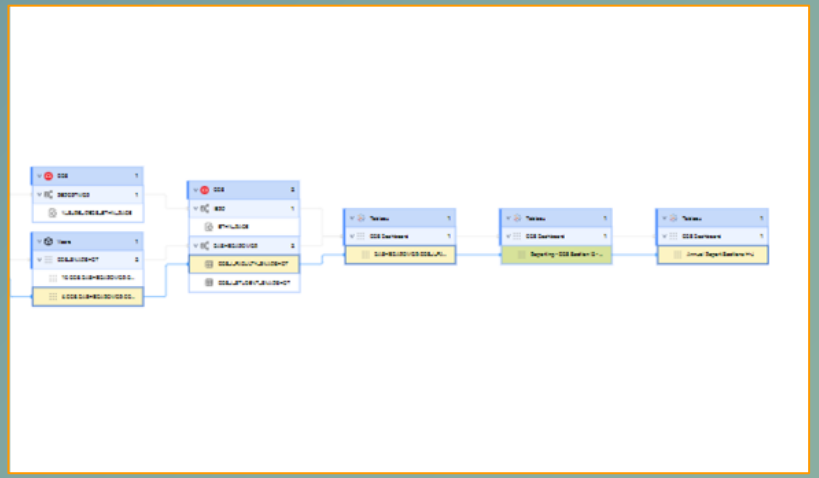
Informatic at Rowan University

- This is just the beginning
- Still lots to learn
- Lots of potential
  - Align business terms and technical terms
  - Track and consolidate reporting data sets
  - Ensure that data quality meets specified threshold for reporting
  - Assign stakeholders to review underlying data or approve metrics for reporting

What does this mean for the non-continuous lineage we have been drawing?



VS



Design and Intent

Monitoring Reality

Thank You

Please send questions and comments to manleyd@rowan.edu