How to Evolve a Knowledge Graph?

Outline

- Overview
- Examples requiring Change
- Change Management Techniques
    - Schema evolution
    - View maintenance
    - Truth Maintenance
- Summary

Overview

Only constant in life is change	Change in the real-world Change in the business requirements
Changes can require	Revising the schema Revising the ground facts
Approaches to handle change must address	Technical challenges Social challenges

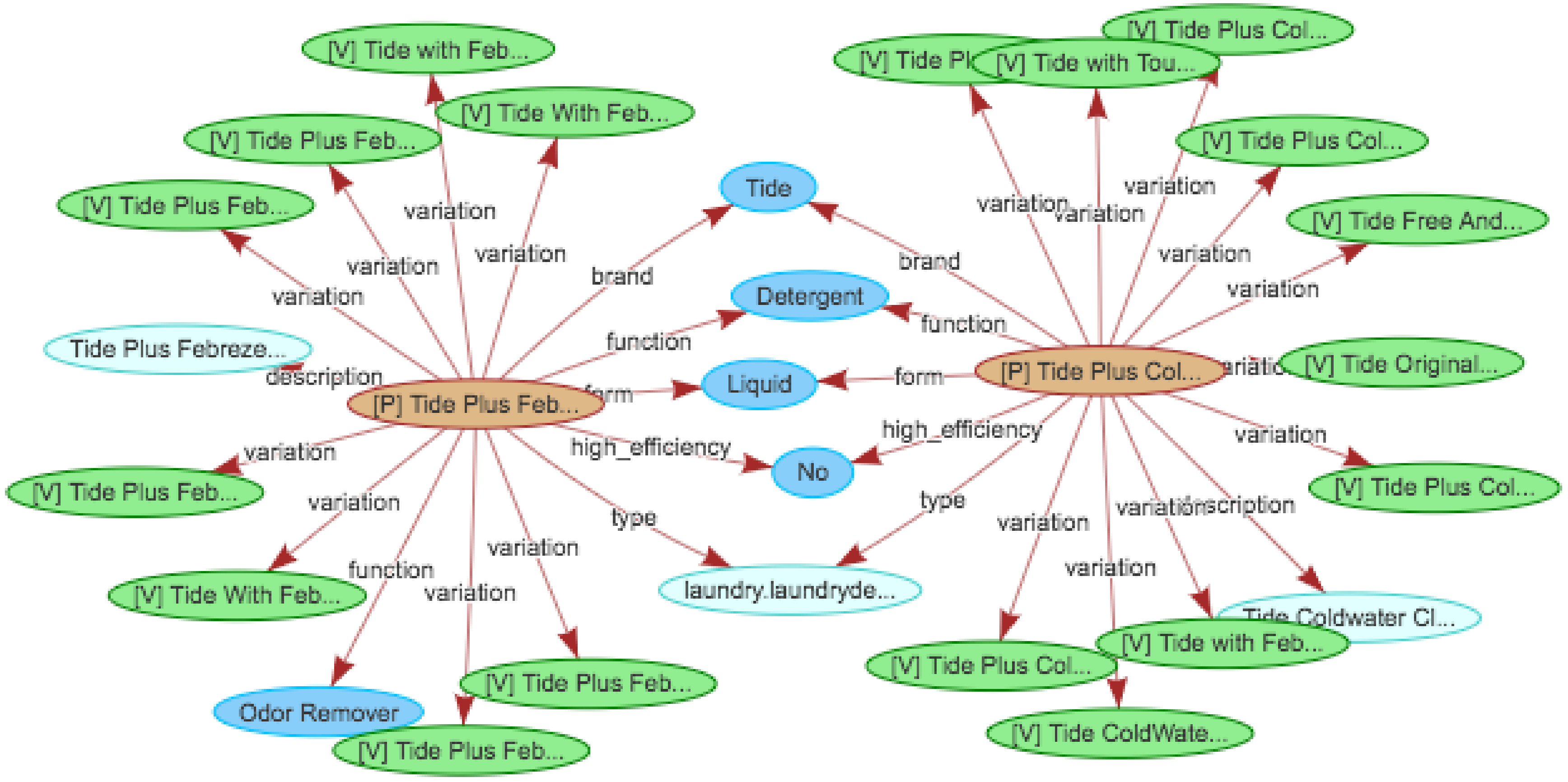
Examples Requiring Change

- Changing world
- Changing requirements
- Changing sources
- Changes affecting previous inferences
- Changes requiring redesign

Examples Requiring Change

- Changing world: Amazon Product Knowledge Graph

New products
New product categories
New features
Discontinued products

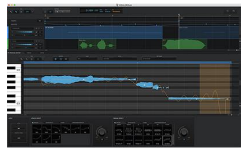


Examples Requiring Change

- Changing requirements: Google Knowledge Graph

An artist must be a person

An artist must be a person OR a vocaloid



Examples Requiring Change

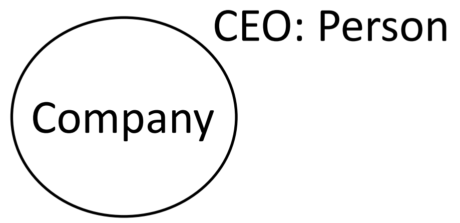
- Changing sources (Google Knowledge Graph)
  - The artists of music albums are obtained from different sources
  - These sources keep changing their data feed
  - The sources used also change over a period of time

Examples Requiring Change

- Changes affecting previous inferences
  - Consider the constraint that a movie theater only shows movies
    - Using this constraint a KG might have previously inferred that certain events are movies
  - More recently the movie theaters are being used for operas, and social events
    - If we had previously derived such events to be movies, we must update them

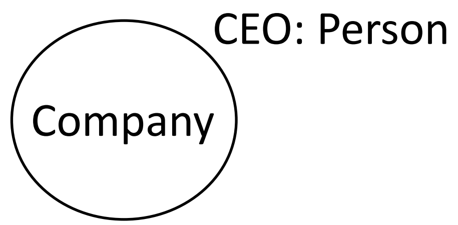
Examples Requiring Change

- Changes requiring redesign
  - Initial design: Every company has a CEO. A CEO is represented by name.

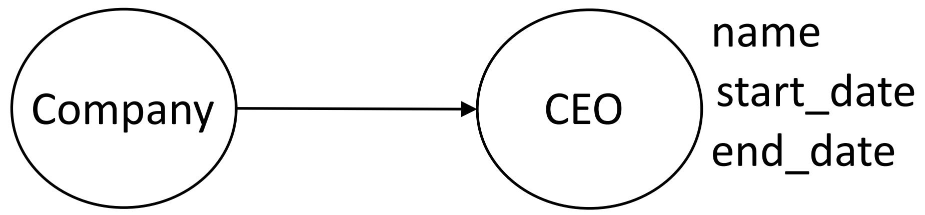


Examples Requiring Change

- Changes requiring redesign
  - Initial design: Every company has a CEO. A CEO is represented by name.



  - Revised design: Every company has a CEO. A CEO is represented by an object that can also record the duration for which the person was a CEO



Outline

- Overview
- Examples requiring Change
- Change Management Techniques
  - Schema evolution
  - View maintenance
  - Truth Maintenance
- Summary

Schema Evolution

- For a relational database
  - Adding/removing a column, renaming an attribute
    - Known as database reorganization

Schema Evolution

- For a relational database
  - Adding/removing a column, renaming an attribute
    - Known as database reorganization
- For a knowledge graph
  - Adding/removing a class
  - Adding/removing a superclass
  - Adding/removing a property
  - Adding/removing a constraint

Approach is to maintain invariants, and make system-specific decisions

Schema Evolution

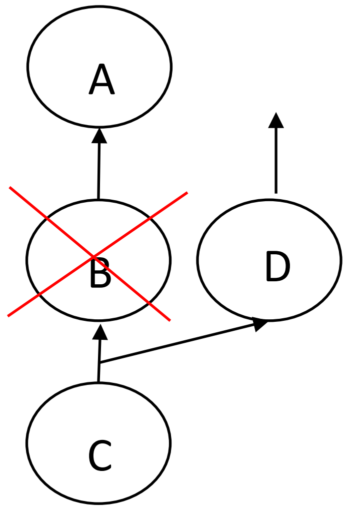
- Remove/rename a property
  - The change must be propagated through the graph
  - Generate a summary for review by the user

Schema Evolution

- Add a class
  - By default, its parent is the system defined root class

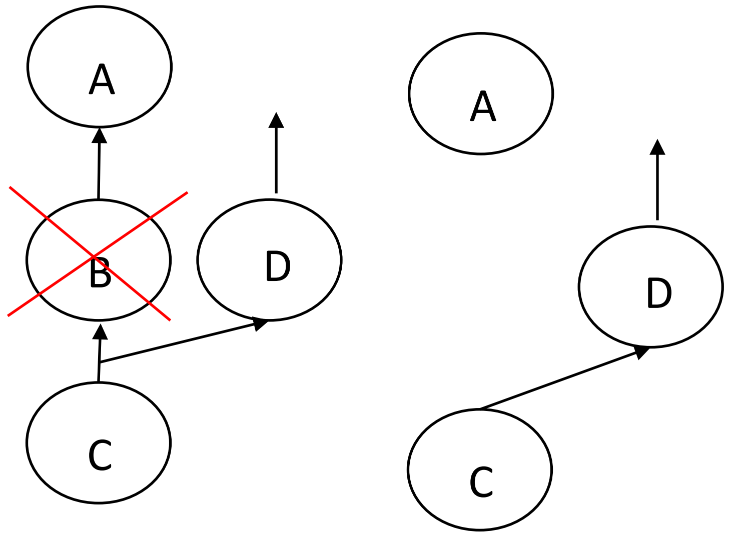
Schema Evolution

- Remove a class
  - What to do about its subclasses and instances
    - If its subclass has another parent, do nothing
      - Otherwise, make it a subclass of immediate parent
    - A more extreme step is to delete the subclasses and instances



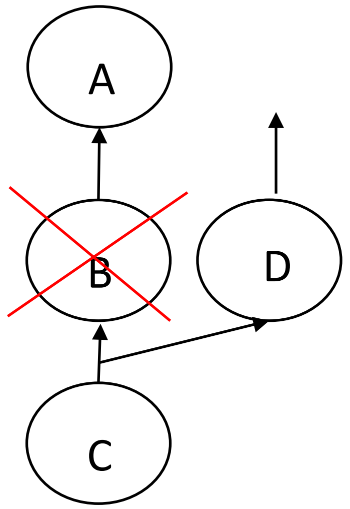
Schema Evolution

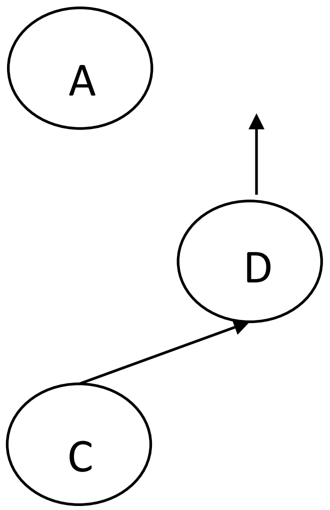
- Remove a class
  - What to do about its subclasses and instances
    - If its subclass has another parent, do nothing
      - Otherwise, make it a subclass of immediate parent
    - A more extreme step is to delete the subclasses and instances

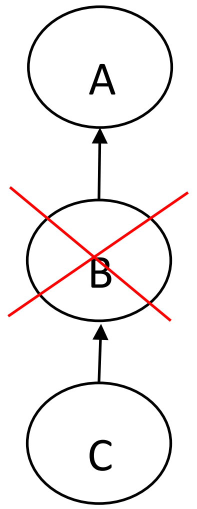


Schema Evolution

- Remove a class
  - What to do about its subclasses and instances
    - If its subclass has another parent, do nothing
      - Otherwise, make it a subclass of immediate parent
    - A more extreme step is to delete the subclasses and instances

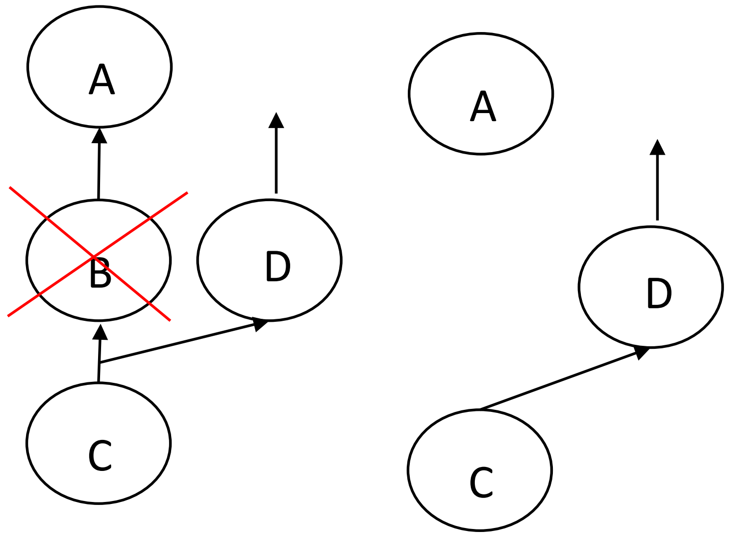


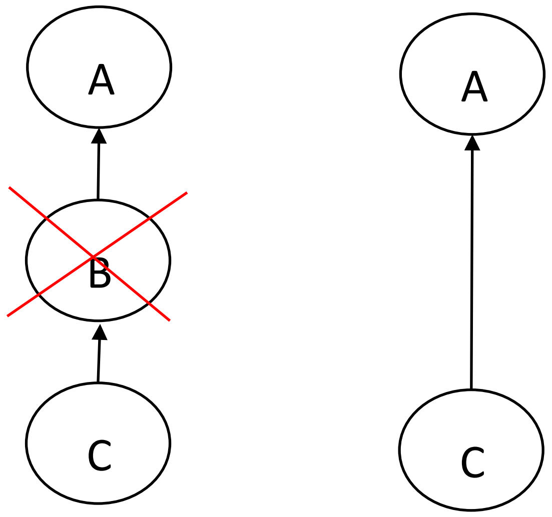




Schema Evolution

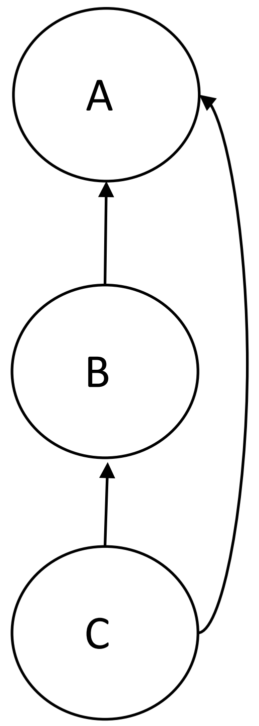
- Remove a class
  - What to do about its subclasses and instances
    - If its subclass has another parent, do nothing
      - Otherwise, make it a subclass of immediate parent
    - A more extreme step is to delete the subclasses and instances





Schema Evolution

- Add a super class
  - How to handle redundant links?
  - How to handle cycles?



View Maintenance

- A mechanism from databases to name a query
  - Query is defined with respect to one or more tables (known as ***base tables***)
  - If we store the results of the query, the stored data is called ***materialized*** view
- If the base data changes, the materialized view must be updated
  - Incremental view maintenance

View Maintenance

- A mechanism from databases to name a query
  - Query is defined with respect to one or more tables (known as ***base tables***)
  - If we store the results of the query, the stored data is called ***materialized*** view
- If the base data changes, the materialized view must be updated
  - Incremental view maintenance

Use of view maintenance is not prevalent in current Knowledge Graph engines

Truth Maintenance

- A mechanism from rule-based systems
  - Tracks how each conclusion was derived
- A popular implementation: Justification based system
  - Each derived conclusion records the fact or rule that was used in derivation
  - Any time that fact or rule updates, the conclusion must be revised

Use of truth maintenance is not prevalent in current Knowledge Graph engines

Summary

- Knowledge Graphs have a life-cycle
  - Must evolve over a period of time
  - Must address both social and technical concerns
- Techniques and algorithms
  - Schema evolution
  - View maintenance
  - Truth maintenance