A better way to enforce a data model

Suggestions to improve Autofix

Camillo Carlo Pellizzari di San Girolamo (user:Epìdosis)
Scuola Normale Superiore

This session is recorded: Please mute your microphone and camera when you're not speaking.

DATA MODELLING DAYS
November 30
December 2
2023
WIKIDATA

Summary

1. Modelling data: **detect, decide, enforce**
2. Enforcing a data model: overview of the **methods**
3. The most used enforcement method, **Autofix**, and its issues
4. Proposals of **improvement**, towards a new Autofix

1) Modelling data: detect, decide, enforce

DATA MODELLING DAYS
November 30
December 2
2023
WIKIDATA

The aim: uniformity

The fundamental assumption is:

- structured data is great, but ...
- ... if everyone structures them in a different way, then they are basically useless, because it's nearly impossible to query them

So, we need to agree on data models: the **same information** should always be structured in the **same way**

Detect, decide, enforce

Divergent data models naturally arise over time. In order to reduce (and, ideally, completely suppress) these divergences, we can follow the following process (see Wikidata:Events/Data Quality Days 2022/Modeling data for a more complete description and examples):

1. **detect** the existence of divergent data models
2. **decide** which data model is the standard one
3. **enforce** the standard data model, converting the others into it

One example: martyrs

1. **detection**: a person who is considered a martyr by a certain religion can be modelled as “instance of” (P31) = “martyr” or “occupation” (P106) = “martyr” or “role of subject” (P2868) = “martyr” (etc.)
2. **decision**: a discussion at WikiProject Religions chooses the data model “role of subject” = “martyr” as standard
3. **enforcement**: Autofix is used to automatically convert occurrences of “instance of” and “occupation” to “role of subject”

2) Enforcing a data model: overview of the methods

DATA MODELLING DAYS
November 30
December 2
2023
WIKIDATA

Overview of enforcement methods

Method	What can it fix?	Pros	Cons
(manual editing)	Everything	Versatility	Dramatic inefficiency
property constraints	A few basic patterns	Queryable	They don't edit items!
Autofix	A few basic patterns	Efficiency	*Many* (see part 3)
DeltaBot fixClaims	Many more patterns than Autofix	Versatility; efficiency	*Some shared with Autofix* (see part 3)
bot (appositely programmed)	Everything	Versatility; efficiency	One-time; needs programming skills

3) The most used enforcement method, Autofix, and ...

DATA MODELLING DAYS
November 30
December 2
2023
WIKIDATA

Template:Autofix – overview

- programmed by user:Ivan A. Krestinin in 2017
- it is added in the talk pages of properties; once added, consequent edits are made by KrBot (a bot managed by user:Ivan A. Krestinin)
- it is used for properties having as datatype “item” or a textual datatype (“external-id”, “string” etc.)

Template:Autofix – two examples

Example 1:

- Property talk:P106 (occupation)

  {{Autofix|pattern=Q6498826|replacement=Q6498826|move_to=P2868}}

this means that “martyr” (Q6498826) is moved from P106 to P2868 (“role of subject”)

Example 2:

- Property talk:P214 (VIAF ID)

  {{Autofix|pattern=<nowiki>([1-9]\d{1,8}|[1-9]\d{18,21})\/</nowiki>|replacement=\1}}

this means that the final “/” is cut from VIAF ID values

... its issues

1) place of storage

Being stored as a template in property talk pages, Autofix:

1. is often a partial duplication of the information contained in property constraints
2. cannot be queried
3. is not much visible
4. requires some (low) programming skills to be understood
5. can be edited by IPs (which from 2020 cannot edit properties)

The first 4 points can also be applied to DeltaBot

2) periodicity and tracking of edits

1. The periodicity of edits through which KrBot applies the Autofix templates is not clearly stated; usually it is daily (***for DeltaBot it is clearly stated as hourly***)
2. The batches of edits triggered by each Autofix template are not tracked through EditGroups, which makes it very difficult to undo them massively if necessary (***same for DeltaBot***)

3) prevention of bot wars

- If a user (human or bot) continues to add a statement which is then autofixed, Autofix has no mechanism which at some point stops it, so the edit war could be potentially infinite (example)

4.1) limitations: patterns which can be autofixed

Autofix can work on the following patterns (for datatype “item”):

- X:A -> X:B
- X:A -> Y:A
- X:A -> Y:B
- X:A -> X:A + X:B or X:A + Y:B

X or Y is main statement and qualifier and reference

A or B is the value of main statement and qualifier and reference

4.2) limitations: patterns which cannot be autofixed

Autofix cannot work on the following patterns (for datatype “item”):

1. restrict a certain fix only on main statement and/or qualifiers and/or references instead of everywhere (***neither DeltaBot, if I see correctly***)
2. manage recursive subclasses (e.g. autofix X:A where A is a recursive subclass of C to Y:A) (***neither DeltaBot, if I see correctly***)
3. manage combinations of main statement(s) and qualifier(s), e.g. autofix X:A with qualifier Y:B to Y:B with qualifier X:A) (***DeltaBot manages these cases at least partially, if I see correctly***)

4) Proposals of improvement, towards a new Autofix

DATA MODELLING DAYS
November 30
December 2
2023
WIKIDATA

How a new Autofix should look like, IMHO

Premise: both Autofix and DeltaBot fixClaims function can (and should) be taken as inspiration, but I think the new Autofix will need to be **written *ex novo***, because it should be different from its predecessors at least in one fundamental aspect:
it should be a system that perform autofixes **on the basis of statements stored in content pages** (mainly property pages, along with constraints) and not on the basis of code lines stored elsewhere.

Proposed features

1. autofix “commands” are stored as **statements in property pages**, together with constraints (so that they don’t duplicate constraints and they are queryable and they are immediately visible and they don’t require programming skills)
2. autofix edits based on the “commands” are performed by a bot with a **clear periodicity** and their batches are tracked in **EditGroups** (so that they can easily be undone if necessary)

Proposed features

3. The bot performing autofix edits should incorporate some mechanism that, **in case of edit wars**, stops the edits on the contended item and sends a message to the involved users
4. The autofix “commands” should support the **widest possible range of autofixes**, including all the ones presently supported by Autofix and DeltaBot plus the ones required above (mainly the case of recursive subclasses; e.g. autofixing “martyr” + all its recursive subclasses from “occupation” to “role of subject”)

Practically, what do we need?

- The main part of the new Autofix which still needs to be constructed is a group of **properties** which should allow to **store as statements the autofix "commands"** which are presently supported by Autofix and by DeltaBot
- When we will be able to store in statements all the autofix "commands", creating a bot executing periodically edit batches on the basis of them will probably be easy

How could we proceed?

- Comment on the detailed proposal of the new Autofix: https://www.wikidata.org/wiki/Wikidata_talk:Events/Data_Quality_Days_2022/Modeling_data (see also phab:T341405)

- Reflect on the properties we are missing in order to store autofix “commands” as statements and propose them

Thanks for your attention!

Get in touch with me:

Camillo Carlo Pellizzari di San Girolamo
camillo.pellizzaridisangirolamo@sns.it

DATA MODELLING DAYS
November 30
December 2
2023
WIKIDATA