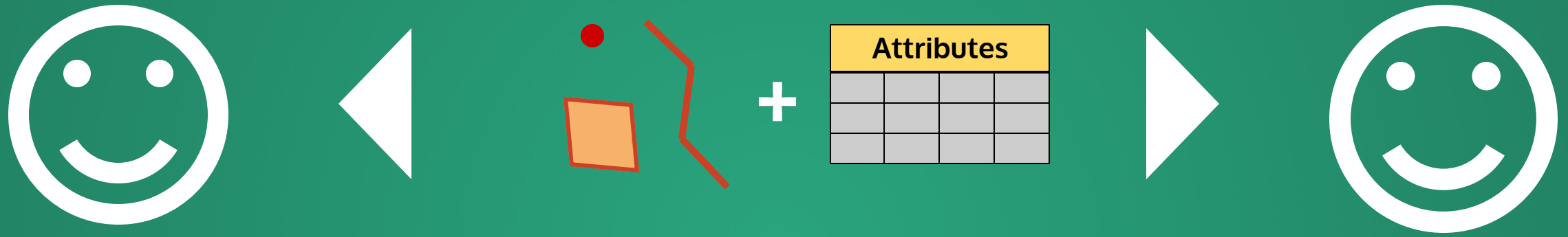
Bringing complex features from extendable GML application schemas into a compact database representation

Felix Kunde

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Exchanging Simple Features



Shapefiles

Keyhole Markup Language

GeoJSON

GIS

CSV

GeoDB

WebGIS

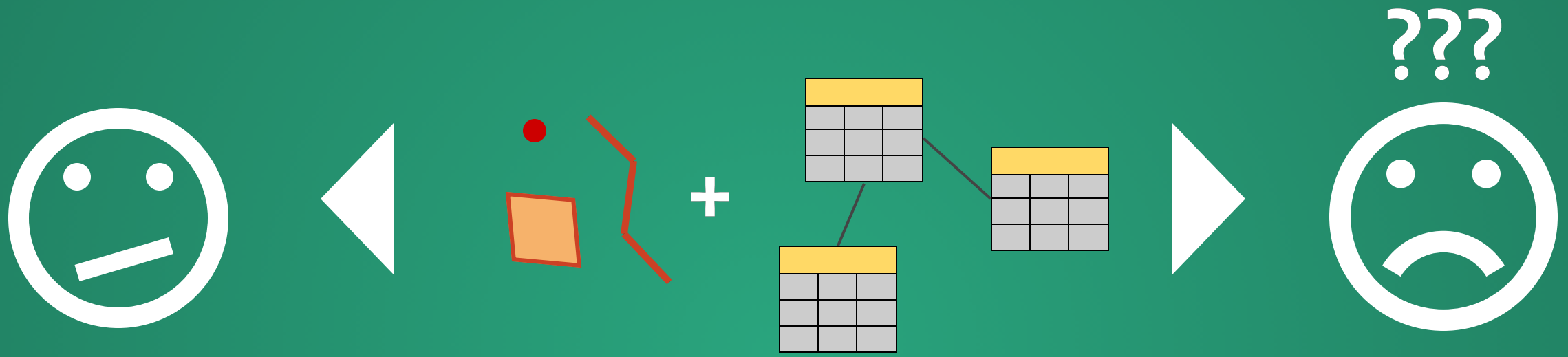
Web

GeoPackage

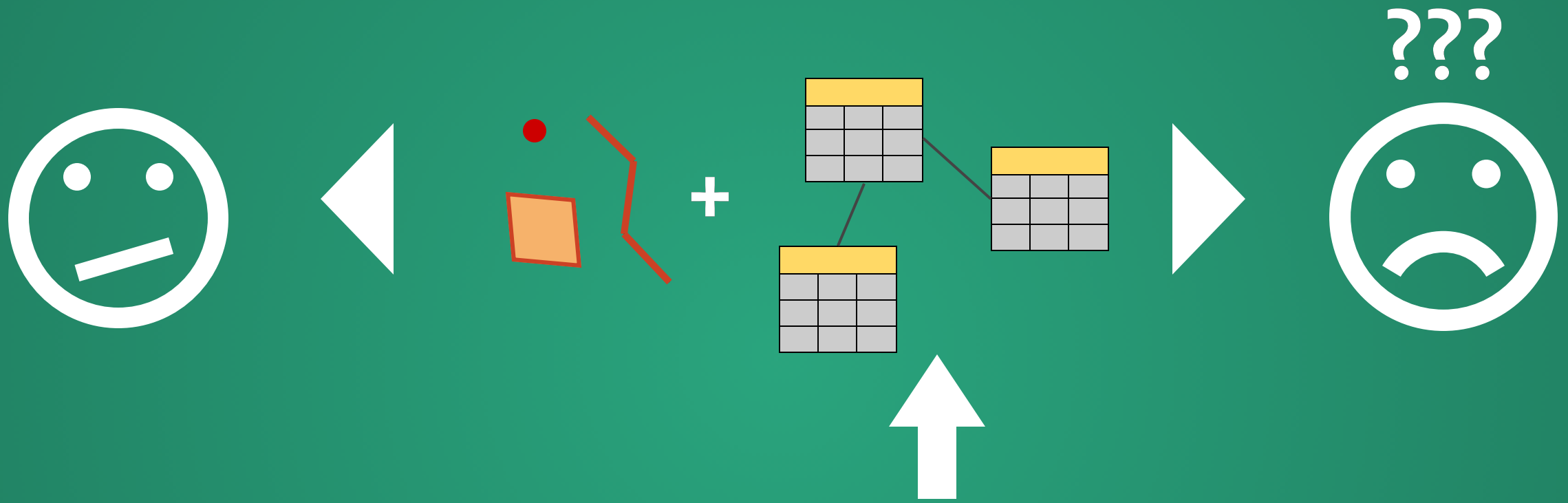
Geography Markup Language

TopoJSON

Exchanging Complex Features



Exchanging Complex Features



Why do we want that?!

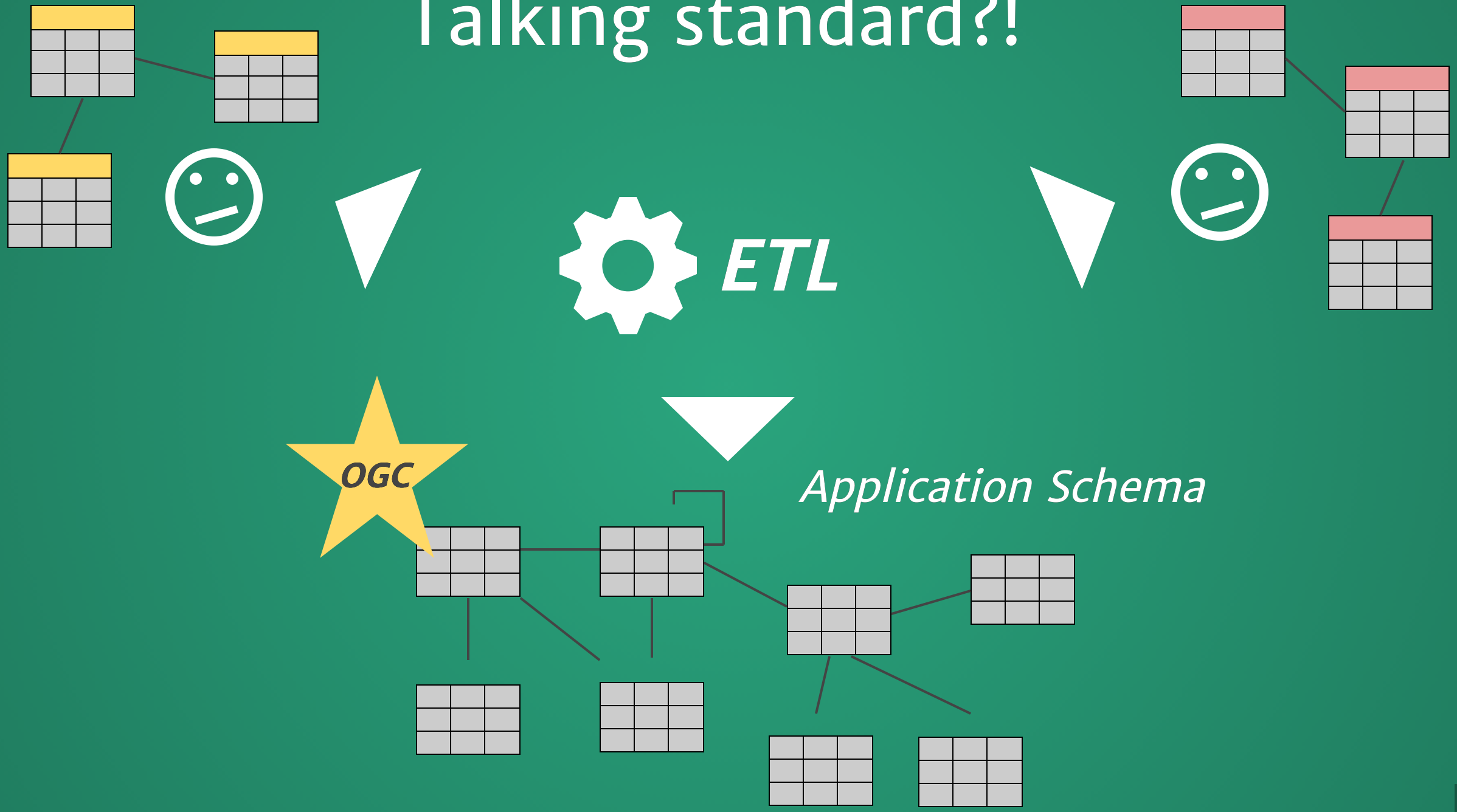
Cleaner modelling

Data consistency

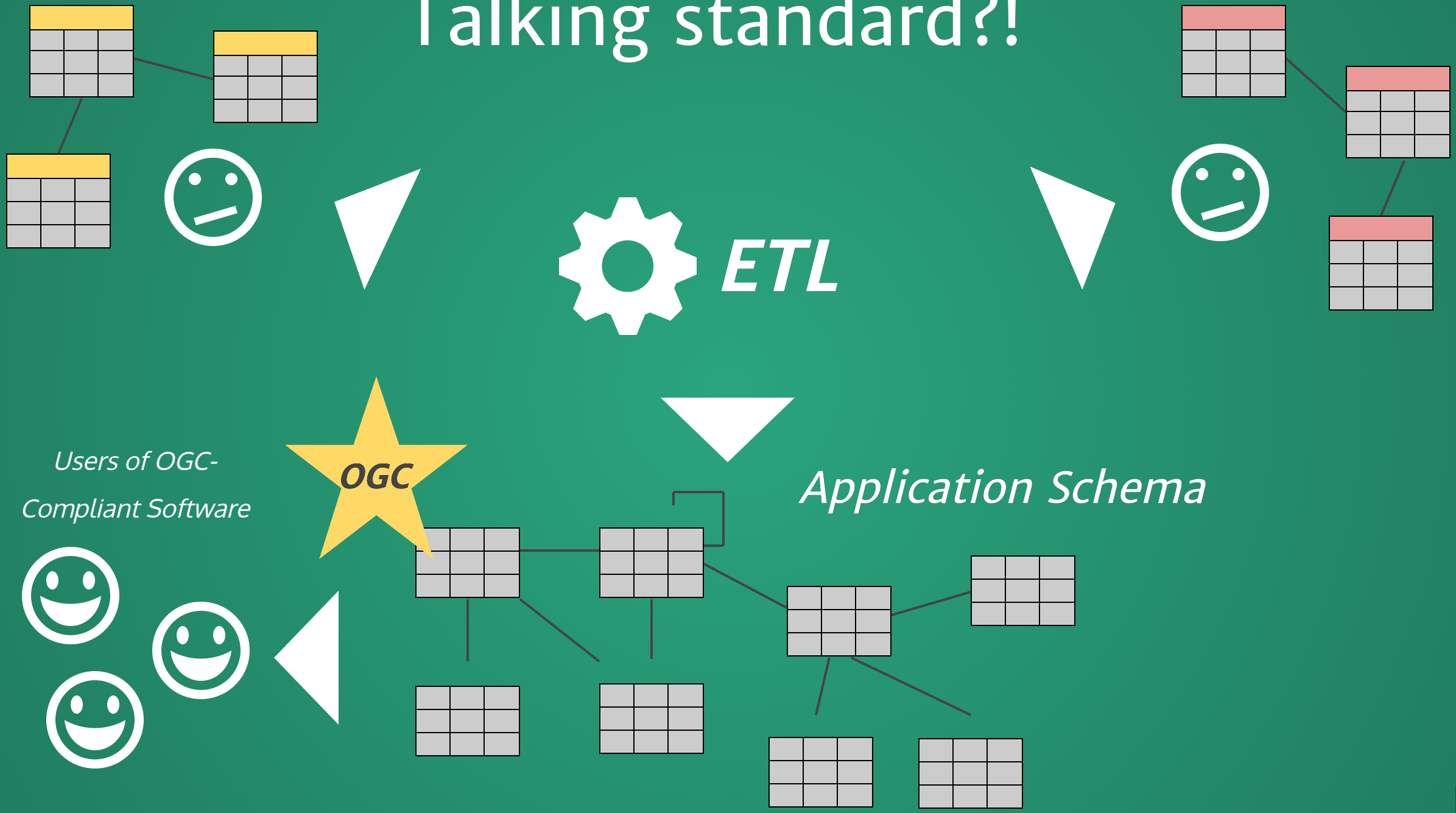
Less redundancy

Talking standard?!

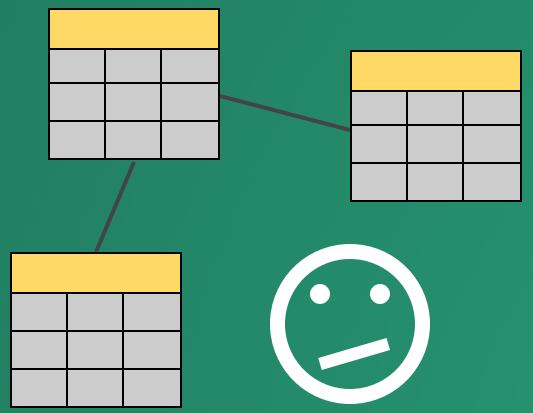
Talking standard?!

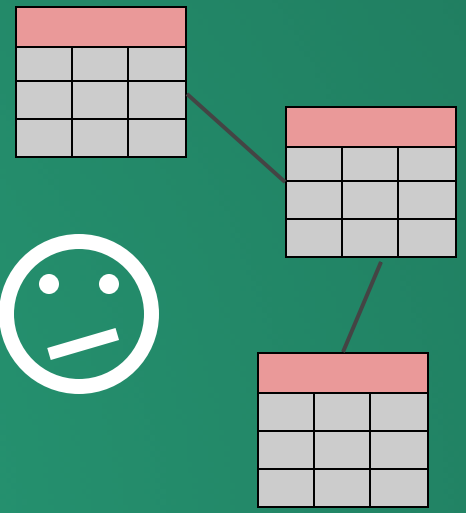


Talking standard?!



Talking standard?!



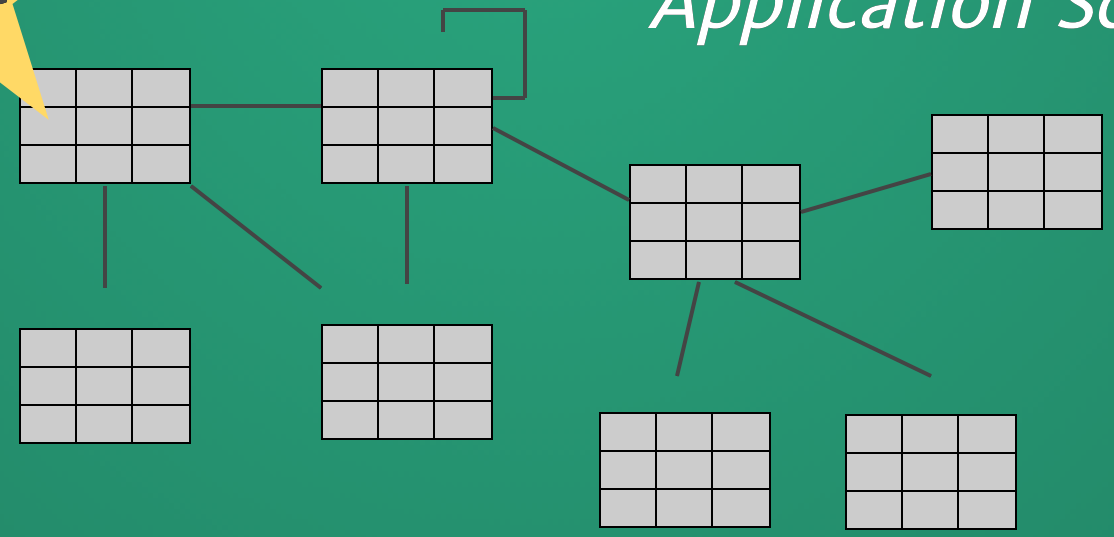


ETL

OGC

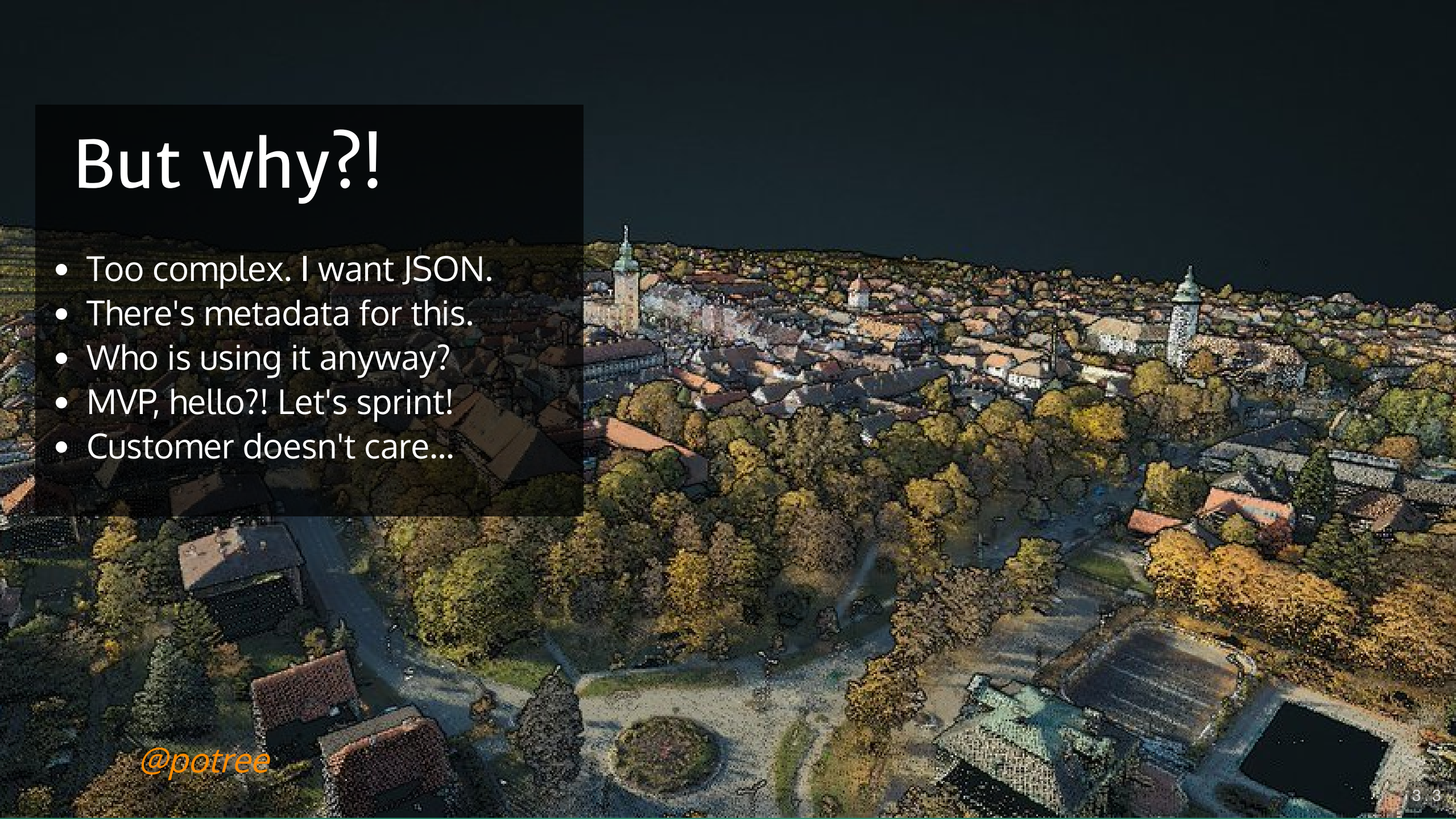
Users of OGC-Compliant Software

Application Schema



But why?!

- Too complex. I want JSON.
- There's metadata for this.
- Who is using it anyway?
- MVP, hello?! Let's sprint!
- Customer doesn't care...



@potree

But why?!

- Too complex. I want JSON.
- There's metadata for this.
- Who is using it anyway?
- MVP, hello?! Let's sprint!
- Customer doesn't care...

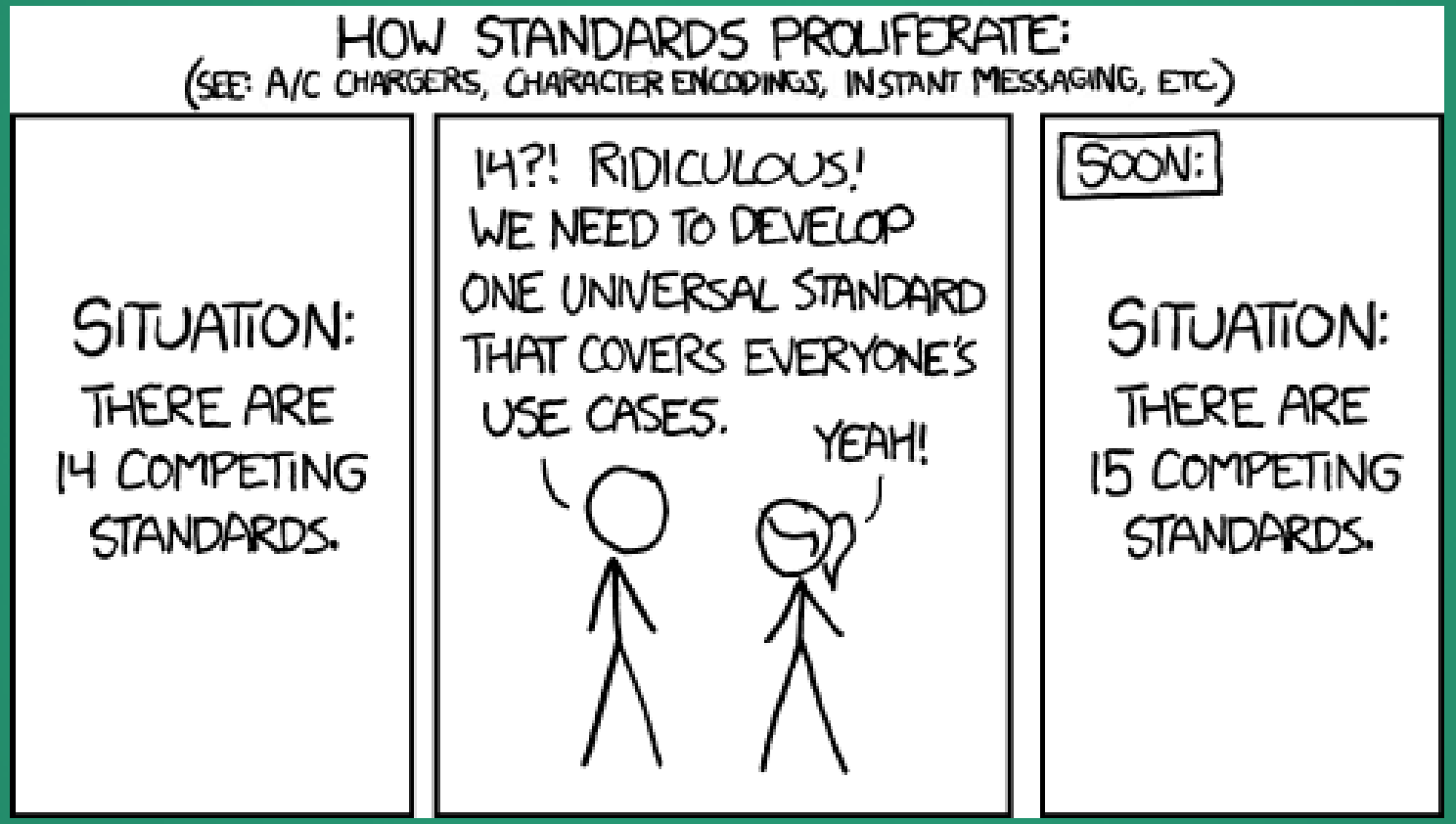
Darum

- Exit data integration hell
- Watch out data producers!
- Yes, there are glitches ... but tests are getting better and easier.
- Protection shield against start ups
- Standardized labeling to use with Machine Learning (just think about it...)

Requirememts vs. Standard

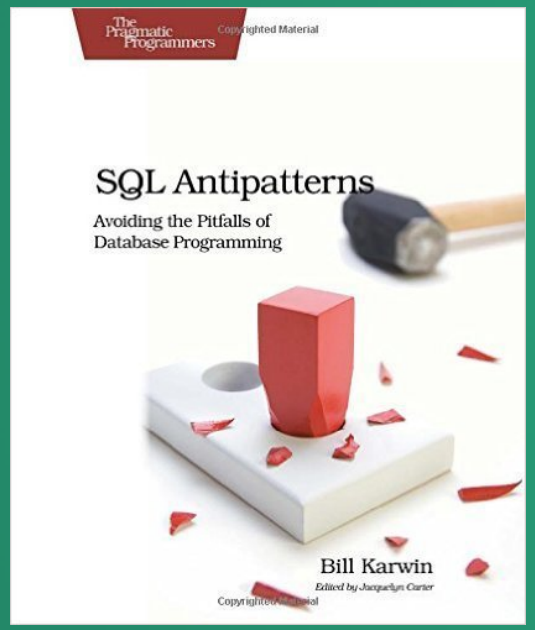
What if the standard doesn't go far enough?

Option 1 – DIY



https://xkcd.com/927/

Option 2 – Make it fit somehow

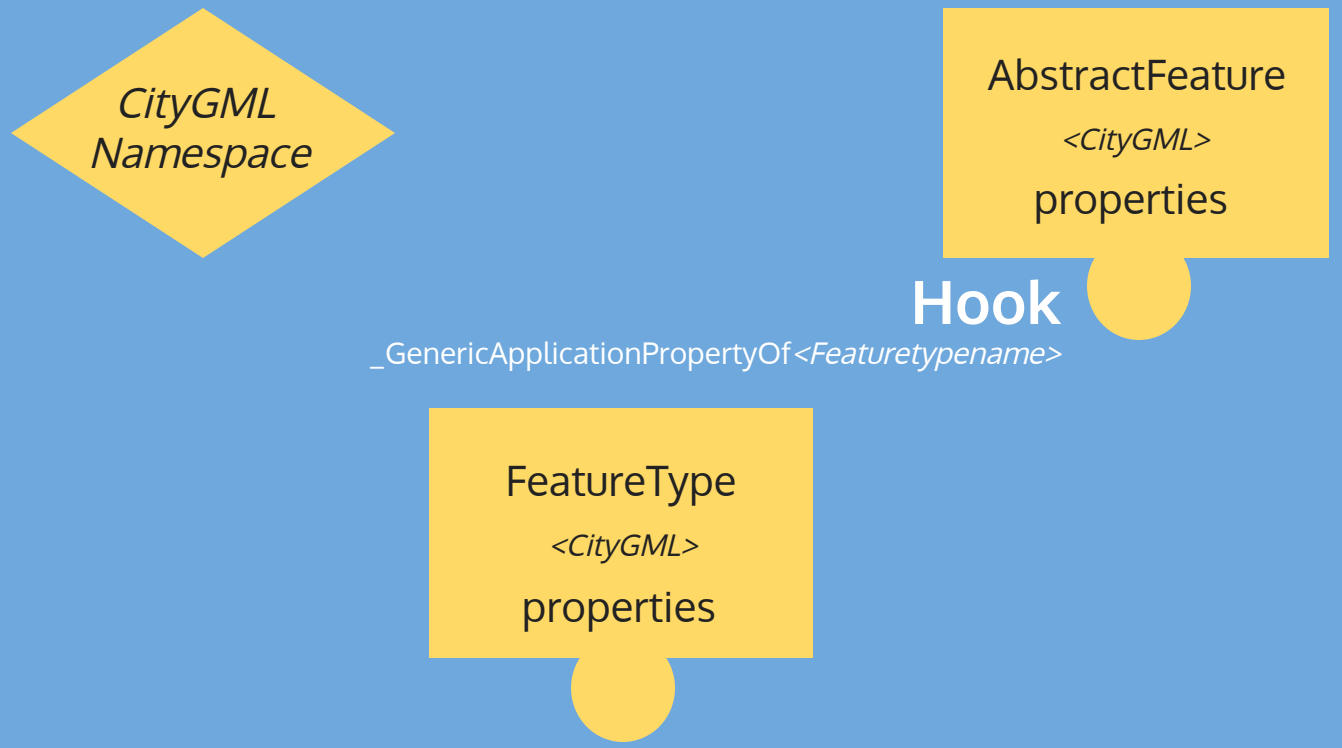


- Use generic classes in case the standard provides them
- Pro: Existing tools can be used
- Con: Cannot validate the semantic content

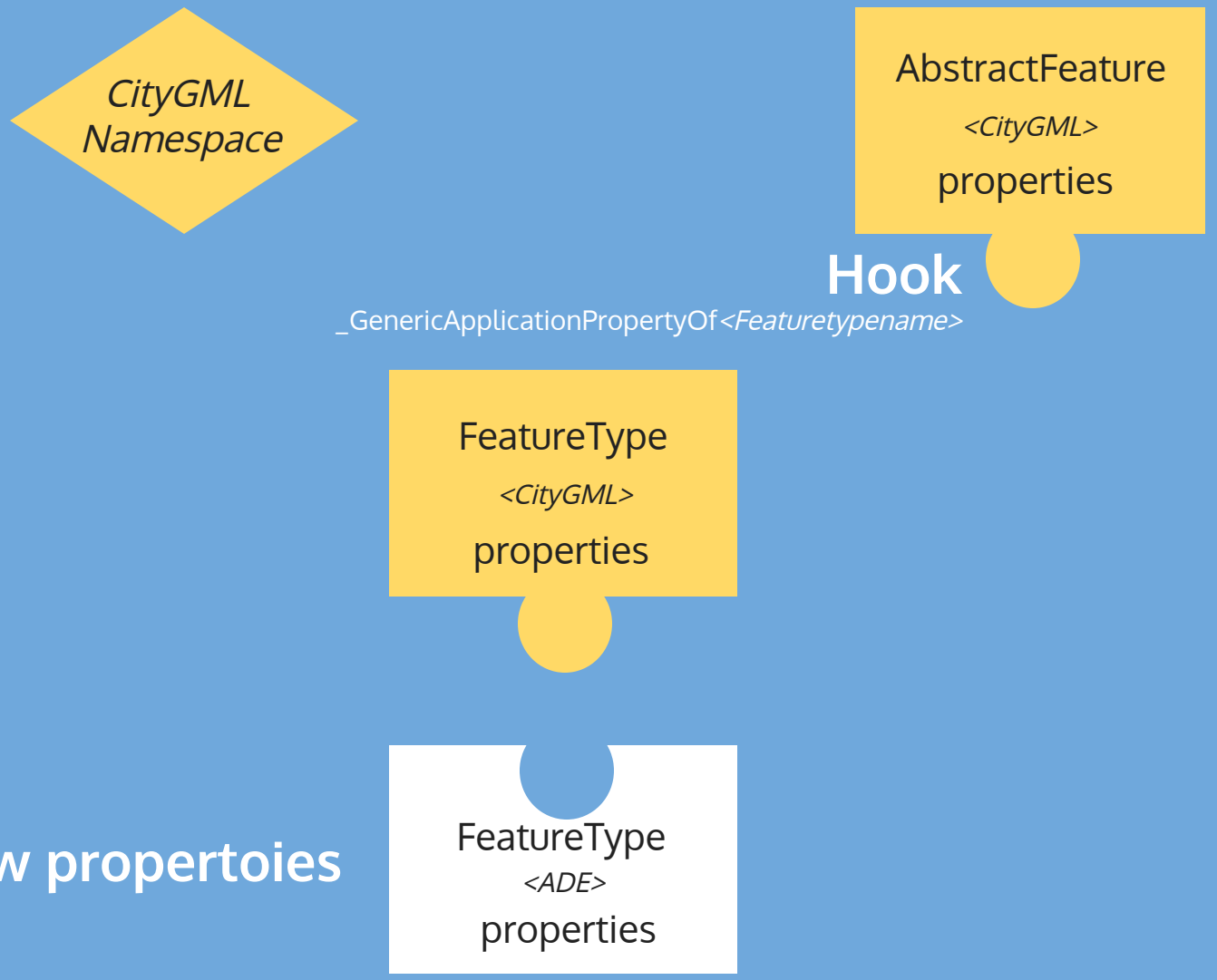
Option 3 – Join SWG

http://dilbert.com/strip/2011-08-02

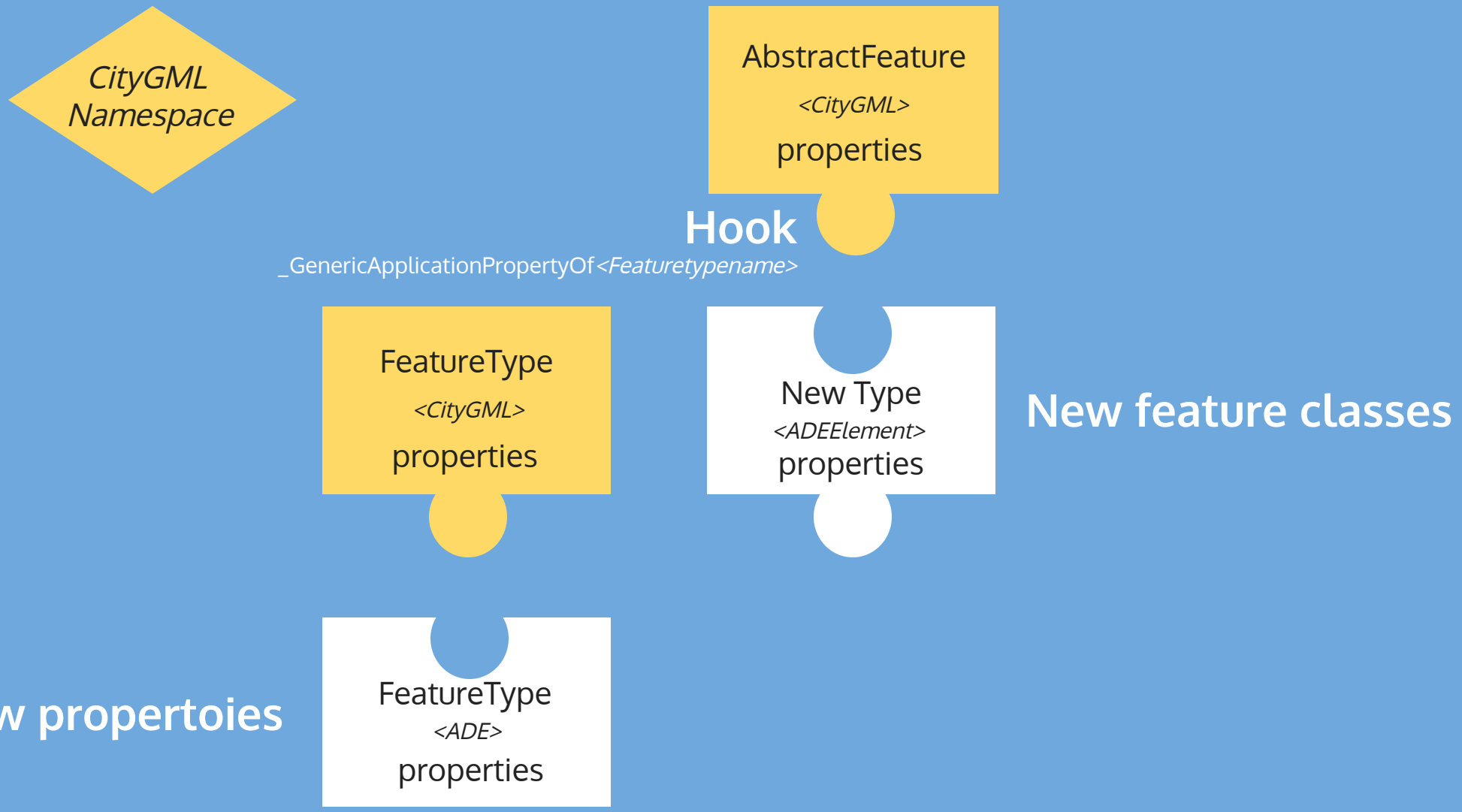
Option 4 – Get hooked



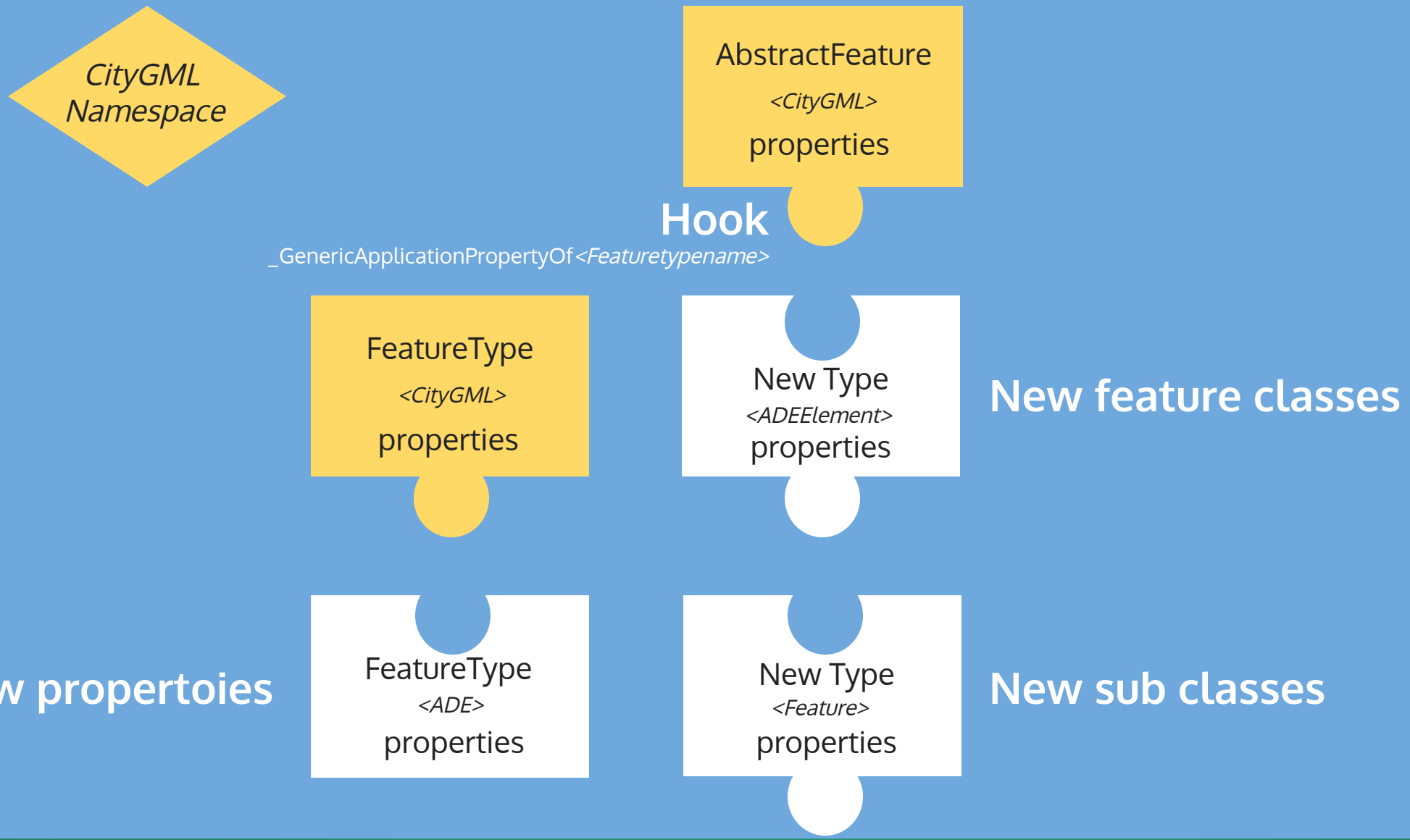
Option 4 – Get hooked



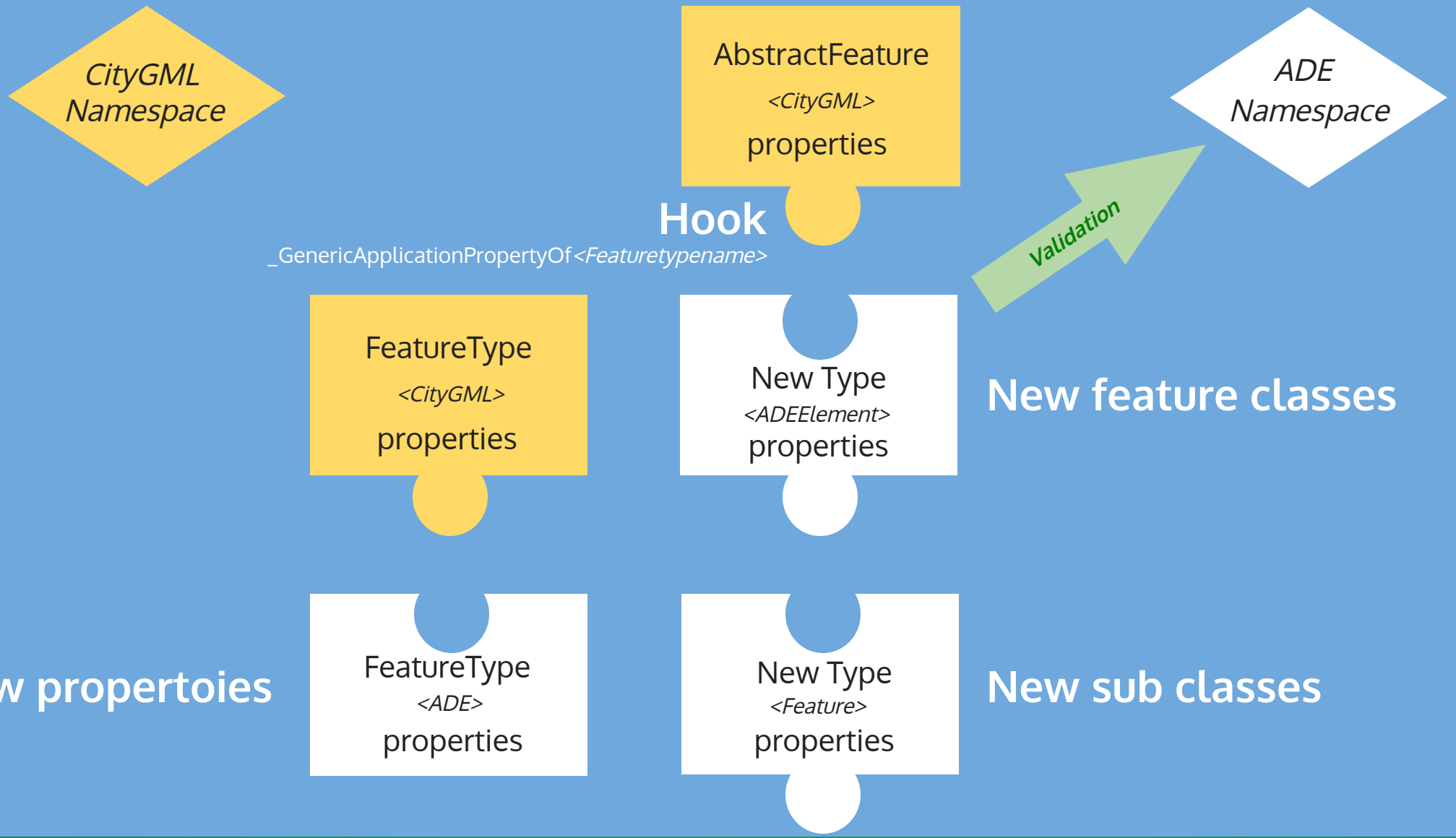
Option 4 – Get hooked

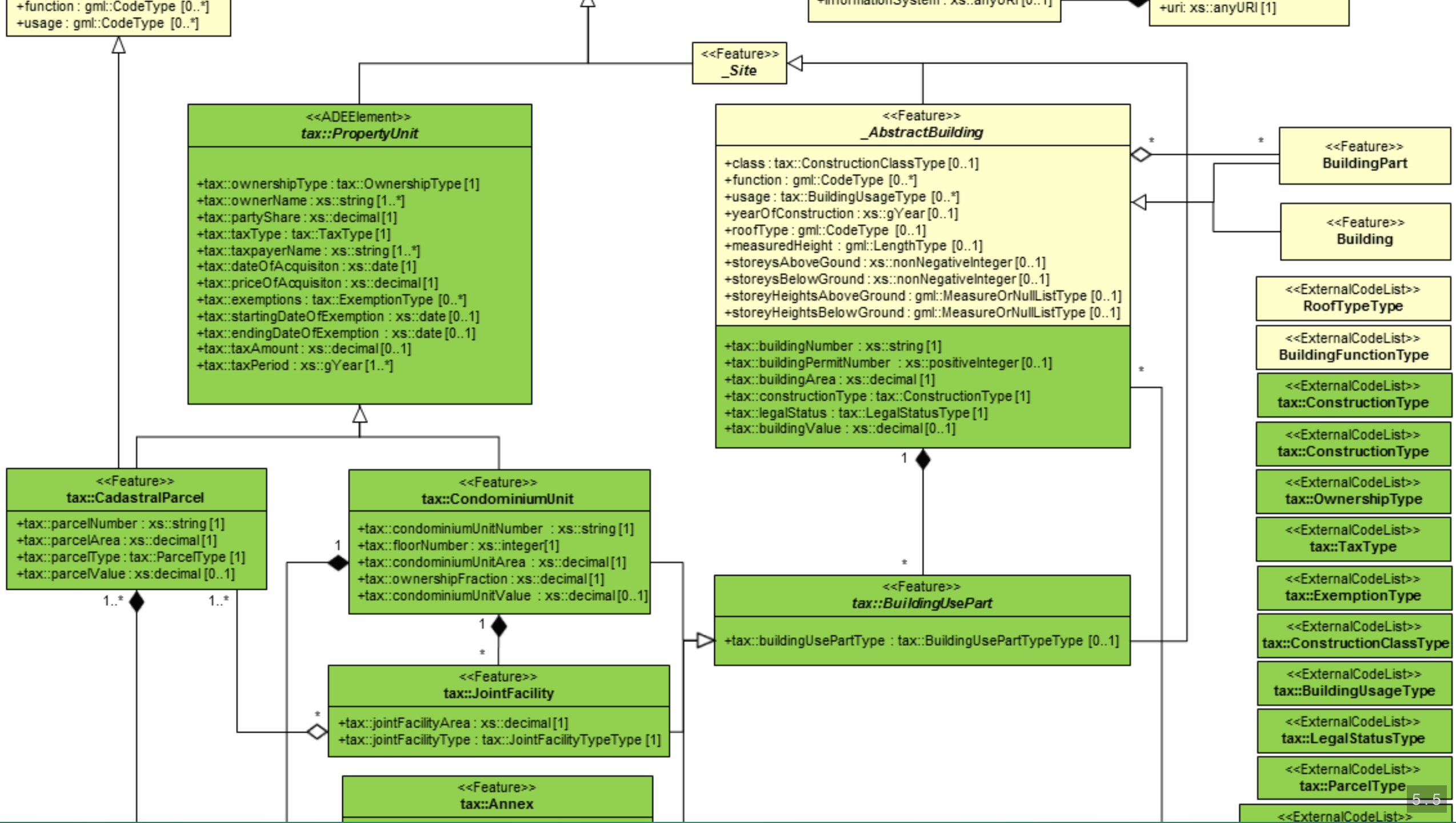


Option 4 – Get hooked



Option 4 – Get hooked




+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
+uri: xs::anyURI [1]
<<Feature>>
_Site
<<ADEElement>>
tax::PropertyUnit
+tax::ownershipType : tax::OwnershipType [1]
+tax::ownerName : xs::string [1..*]
+tax::partyShare : xs::decimal [1]
+tax::taxType : tax::TaxType [1]
+tax::taxpayerName : xs::string [1..*]
+tax::dateOfAcquisiton : xs::date [1]
+tax::priceOfAcquisiton : xs::decimal [1]
+tax::exemptions : tax::ExemptionType [0..*]
+tax::startingDateOfExemption : xs::date [0..1]
+tax::endingDateOfExemption : xs::date [0..1]
+tax::taxAmount : xs::decimal [0..1]
+tax::taxPeriod : xs::gYear [1..*]
<<Feature>>
_AbstractBuilding
+class : tax::ConstructionClassType [0..1]
+function : gml::CodeType [0..*]
+usage : tax::BuildingUsageType [0..*]
+yearOfConstruction : xs::gYear [0..1]
+roofType : gml::CodeType [0..1]
+measuredHeight : gml::LengthType [0..1]
+storeysAboveGound : xs::nonNegativeInteger [0..1]
+storeysBelowGround : xs::nonNegativeInteger [0..1]
+storeyHeightsAboveGround : gml::MeasureOrNullListType [0..1]
+storeyHeightsBelowGround : gml::MeasureOrNullListType [0..1]
+tax::buildingNumber : xs::string [1]
+tax::buildingPermitNumber : xs::positiveInteger [0..1]
+tax::buildingArea : xs::decimal [1]
+tax::constructionType : tax::ConstructionType [1]
+tax::legalStatus : tax::LegalStatusType [1]
+tax::buildingValue : xs::decimal [0..1]
<<Feature>>
BuildingPart
<<Feature>>
Building
<<Feature>>
tax::CadastralParcel
+tax::parcelNumber : xs::string [1]
+tax::parcelArea : xs::decimal [1]
+tax::parcelType : tax::ParcelType [1]
+tax::parcelValue : xs:decimal [0..1]
<<Feature>>
tax::CondominiumUnit
+tax::condominiumUnitNumber : xs::string [1]
+tax::floorNumber : xs::integer[1]
+tax::condominiumUnitArea : xs::decimal [1]
+tax::ownershipFraction : xs::decimal [1]
+tax::condominiumUnitValue : xs::decimal [0..1]
<<Feature>>
tax::BuildingUsePart
+tax::buildingUsePartType : tax::BuildingUsePartTypeType [0..1]
<<Feature>>
tax::JointFacility
+tax::jointFacilityArea : xs::decimal [1]
+tax::jointFacilityType : tax::JointFacilityTypeType [1]
<<Feature>>
tax::Annex
<<ExternalCodeList>>
RoofTypeType
<<ExternalCodeList>>
BuildingFunctionType
<<ExternalCodeList>>
tax::ConstructionType
<<ExternalCodeList>>
tax::ConstructionType
<<ExternalCodeList>>
tax::OwnershipType
<<ExternalCodeList>>
tax::TaxType
<<ExternalCodeList>>
tax::ExemptionType
<<ExternalCodeList>>
tax::ConstructionClassType
<<ExternalCodeList>>
tax::BuildingUsageType
<<ExternalCodeList>>
tax::LegalStatusType
<<ExternalCodeList>>
tax::ParcelType
1
*
1..*
1..*

So what about the database?

Objekt <-> Relational

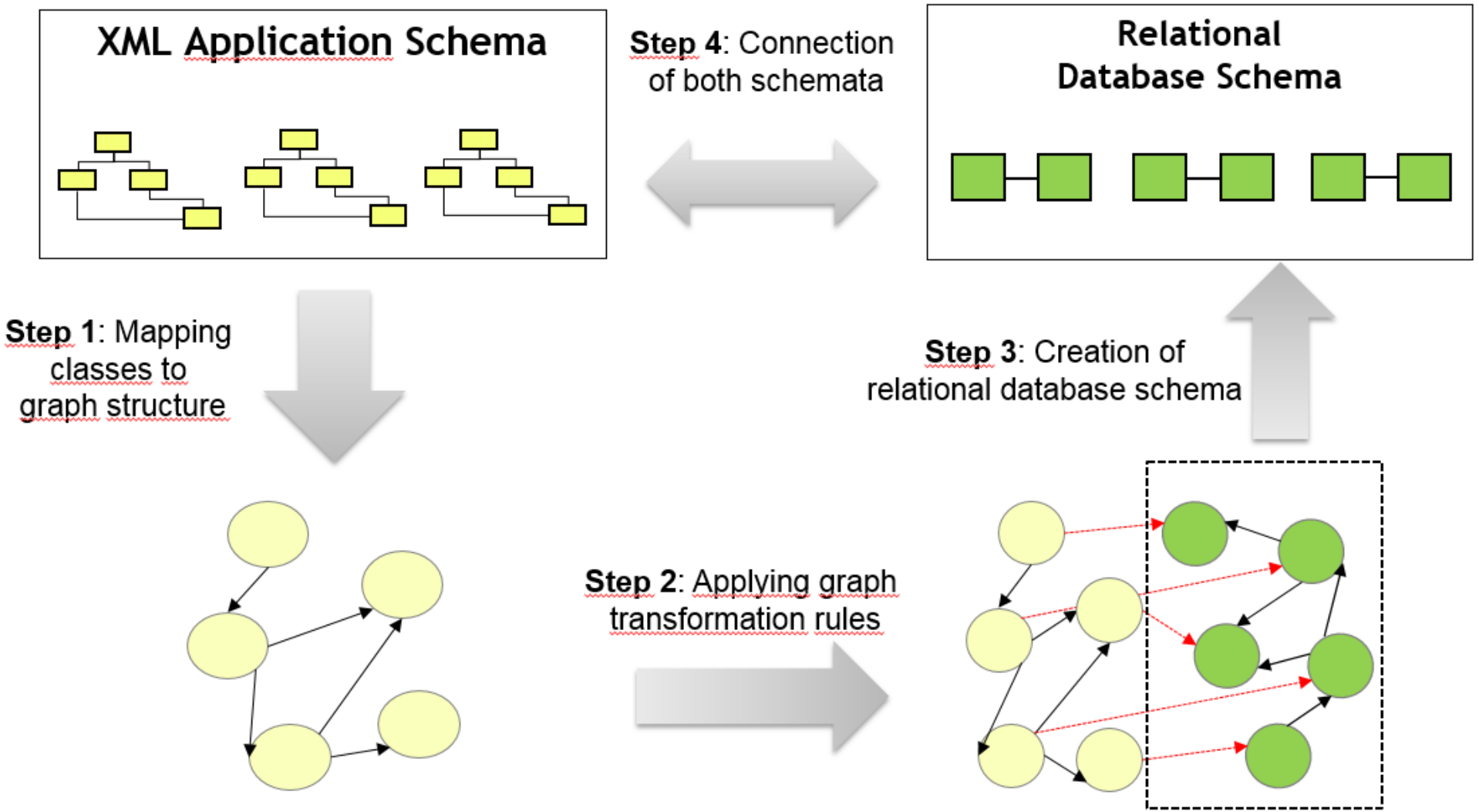
- Oooold topic (impedence mismatch) ...
- One table for each class and complex type etc.?
- Mapping of hierachies and inheritance?
- De-/Normalization (Avoiding JOINs)?
- Extendable XML schema: Map once vs. map often

3D City Database

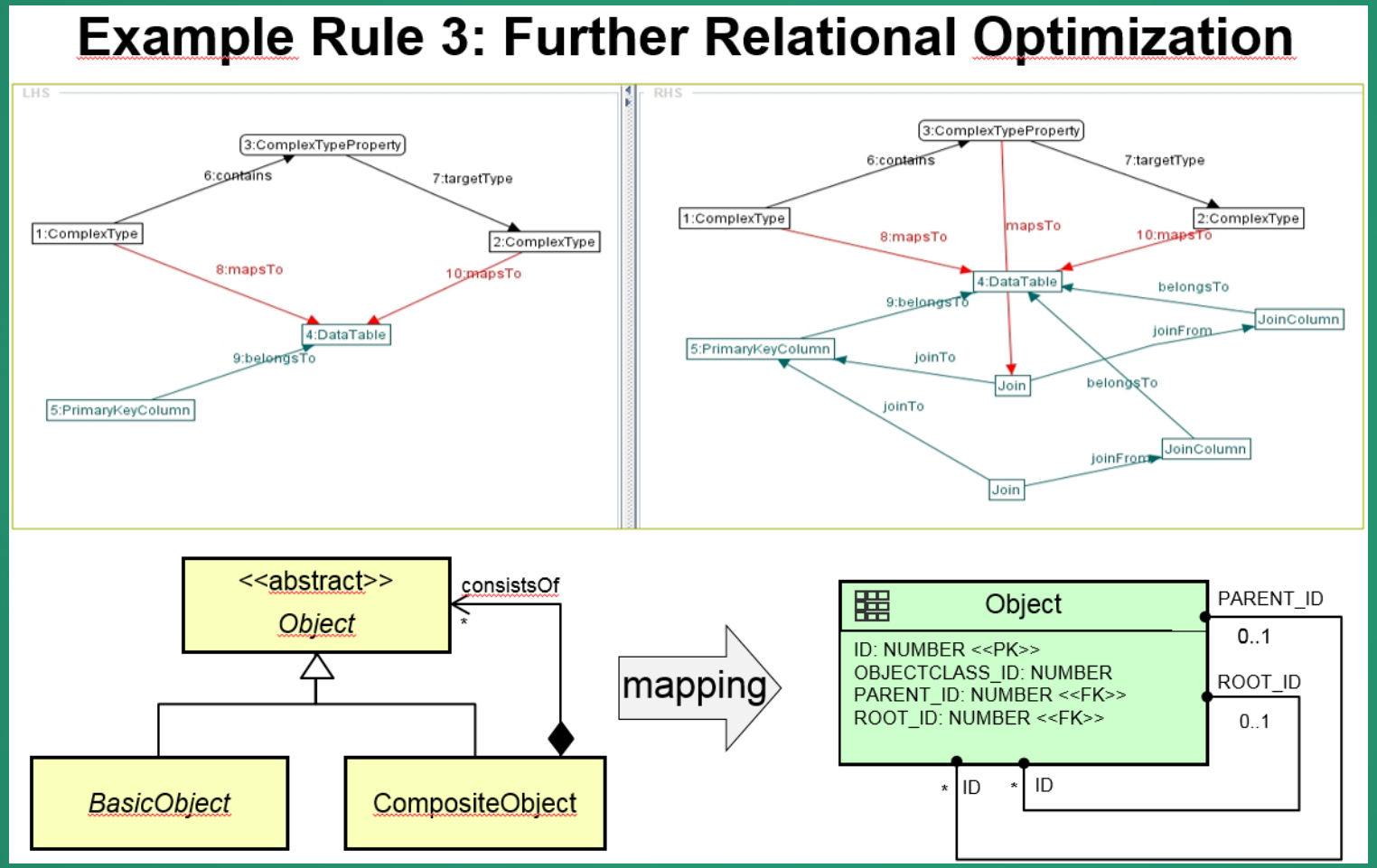
- UML classes edited manually
- Database schema edited manually
- ~ 60 table for CityGML v2.0
- Hard-coded Im/Exporter in Java
- WFS-to-SQL using an XML config file
- **New**: Extendable by abritrary CityGML ADEs
- Repo: github.com/3dcitydb
- hub.docker.com/r/tumgis/3dcitydb-postgis

Workflow of Model Transformation using GTS

http://www.dgpf.de/src/tagung/jt2017/proceedings/proceedings/papers/30_DGPF2017_Yao_Kolbe.pdf



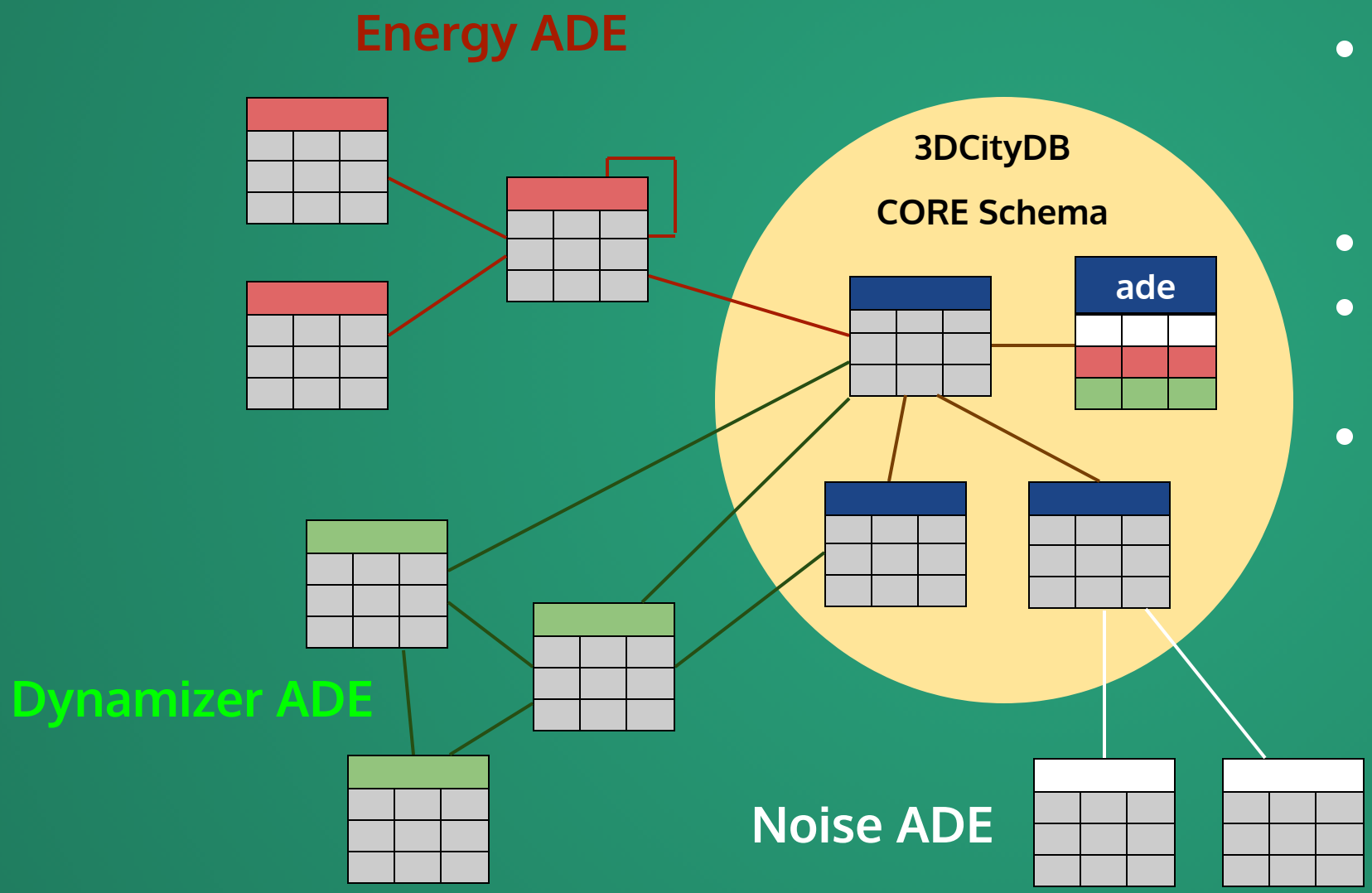
Atributed Graph Grammar



Homepage: http://www.user.tu-berlin.de/o.runge/agg/index.html

Repo(?): https://github.com/de-tu-berlin-tfs/Henshin-Editor/tree/master/de.tub.tfs.agg

ADE Management



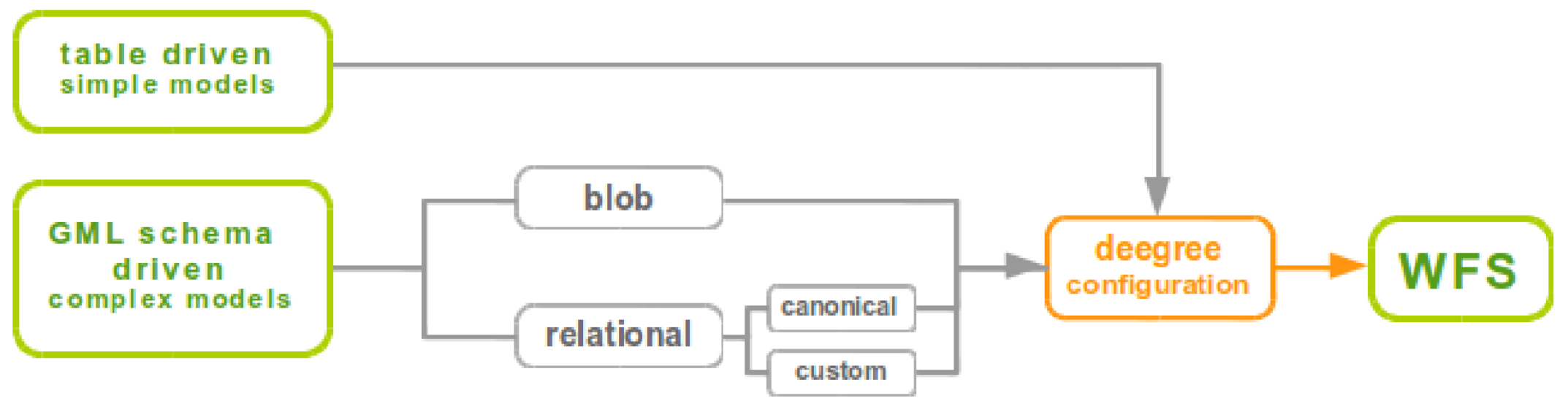
- Automatic mapping into tables and generation of stored procedures*
- Core schema isn't changed
- Class extensions are stored in new tables
- Some implementation still required for enabling imports and exports

deegree Feature Store

- One-time mapping via CLI (v3.4+)
- CityGML = 422 tables (maybe more)
- BLOB-Mode for faster exports
- Configurable mapping (XML)
- Support for more GMLAS (XPlan Toolbox, AIXM)
- PostGIS, Oracle, MS-SQL and (soon) GeoPackage

Docs:
http://download.deegree.org/documentation/3.3.0/html/featurestores.html#application-schemas

deegree Feature Store



Stenger 2017, FOSSGIS Passau

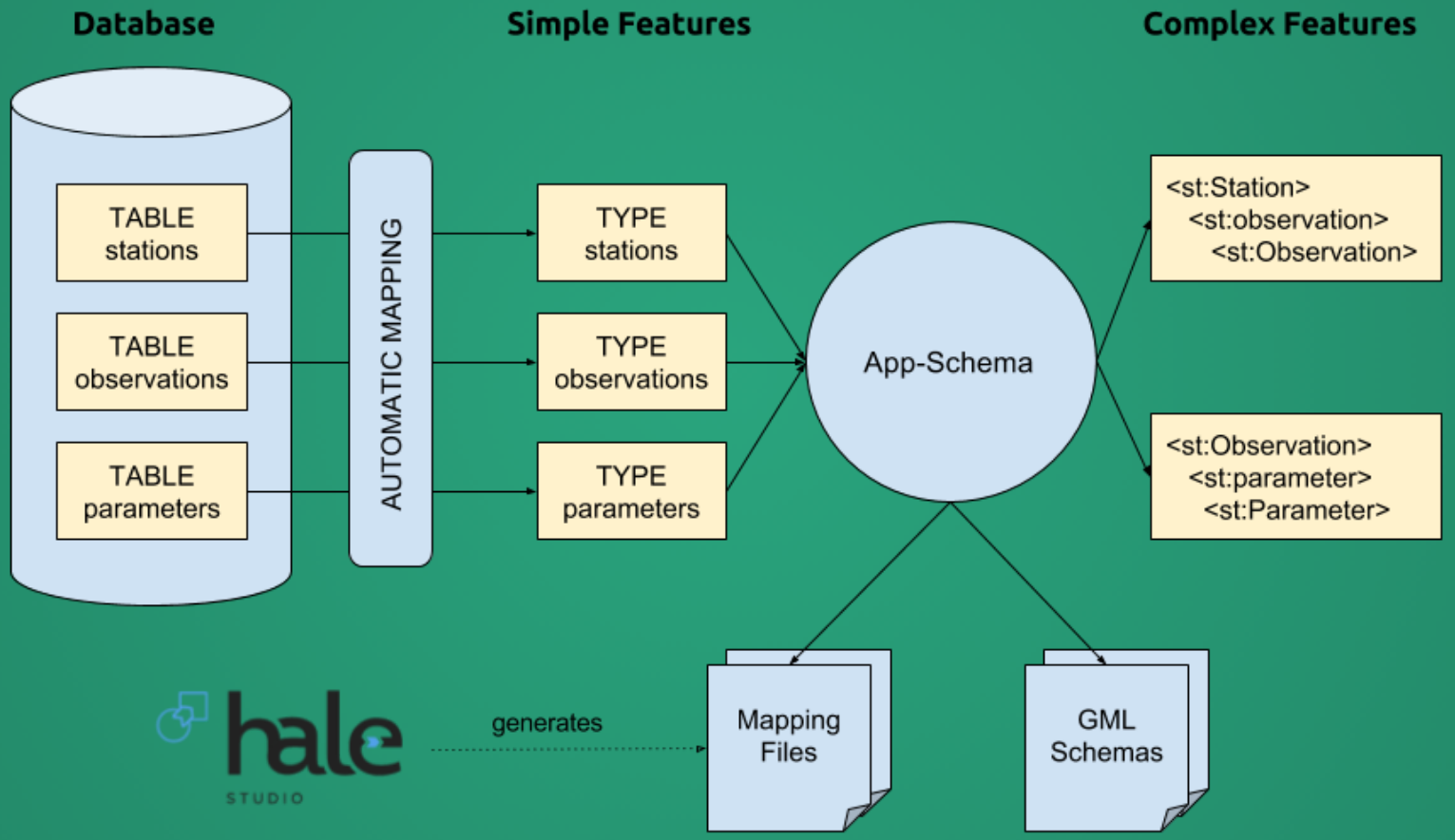
CLI interface: https://github.com/lat-lon/deegree-cli-utility

Geoserver app-schema

- Separate extension to Geoserver
- Database schema must already exist
- 1:n relationships: ***Feature Chaining*** and ***Joining***
- n:m relationships: ***Denormalized Views***
- Use ***Hale*** to create the mapping and export as GeoServer *app-schema*
- PostGIS, Oracle, MongoDB, SolR

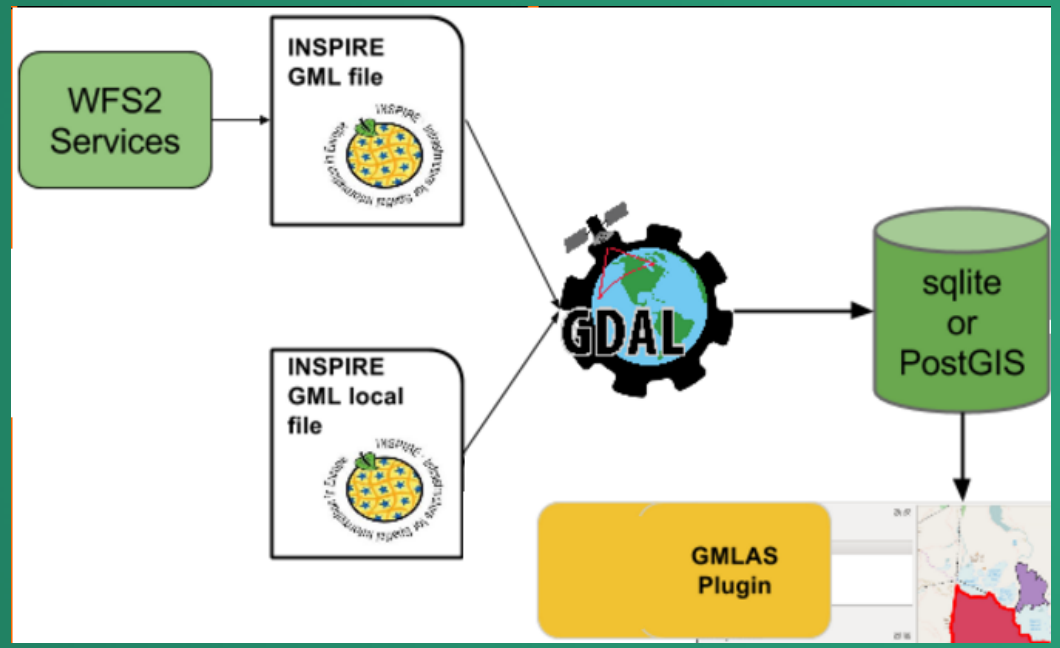
Docs: http://docs.geoserver.org/latest/en/user/data/app-schema/tutorial.html

Geoserver app-schema



Tutorial: https://geoserver.geo-solutions.it/edu/en/complex_features/index.html

BRGM QGIS GMLAS Toolbox



- Based on **GDAL GMLAS** driver
- Very generic, not much to configure
- CityGML = 1412 tables
- Modes: Create, Update, Append, Overwrite
- PostGIS, SQLite

GDAL: http://www.gdal.org/drv_gmlas_mapping_examples.html

BRGM: https://github.com/BRGM/gml_application_schema_toolbox

gml_brgm - QGIS
Bearbeiten Ansicht Layer Einstellungen Erweiterungen Vektor Raster Datenbank Web Verarbeitung Hilfe
bldg_window__generiapplicproperofcityobject_generiattribuset
bldg_window (lod4multisurface)
bldg_window
bldg_wallsurface_opening
bldg_wallsurface_name
bldg_wallsurface_metadataproperty
bldg_wallsurface_generalizesto
bldg_wallsurface_externalreference_externalreference
bldg_wallsurface (lod4multisurface)
bldg_wallsurface (lod3multisurface)
bldg_wallsurface
bldg_wallsurfac__generiapplicproperofcityobject_uriattribute
bldg_wallsurfac__generiapplicproperofcityobject_intattribute
bldg_wallsurfac__generiapplicapropertofcityobject_appearance
bldg_wallsurfa__generiapplicproperofcityobject_dateattribute
bldg_wallsurfa__generappliproperofcityobject_stringattribute
bldg_wallsurfa__generappliproperofcityobject_measurattribute
bldg_wallsurfa__generappliproperofcityobject_generiattribset
bldg_wallsurfa__generappliproperofcityobject_doubleattribute
bldg_roofsurface_opening
bldg_roofsurface_name
bldg_roofsurface_metadataproperty
bldg_roofsurface_generalizesto
bldg_roofsurface_externalreference_externalreference
bldg_roofsurface (lod4multisurface)
bldg_roofsurface (lod3multisurface)
bldg_roofsurface
bldg_roofsurfac__generiapplicproperofcityobject_uriattribute
bldg_roofsurfac__generiapplicproperofcityobject_intattribute
bldg_roofsurfac__generiapplicapropertofcityobject_appearance
bldg_roofsurfa__generiapplicproperofcityobject_dateattribute
bldg_roofsurfa__generappliproperofcityobject_stringattribute
GML App Schema Toolbox
Download
Convert
Export
Help
Load layer list
707 layer(s) found:
powerdistributionsystem (0)
powerdistributionsystem_distributes (0)
powerdistributionsystem_externalreference_externalreference (0)
powerdistributionsystem_generalizesto (0)
Filter by extent
xmin,ymin,xmax,ymax,EPSG:srs_id
Miscellaneous options
Create OGR metadata tables
Create null constraints
Convert to linear geometry
Skip failures
Swap coordinates
AUTO
Language
en
Target database
SQLite
PostgreSQL
Database
pg_bgrm
Schema
public
Foreign keys
Source SRS
Ungültige Projektion
Reproject to
EPSG:28992 - Amersfoort / RD New
Write mode
Create
Update
Append
Overwrite
Convert
Load and configure layers
suchender Typ (Strg+K)
Koordinate
57.27,778.44
Maßstab
1:113
Vergrößerung
100%
Drehung
0,0 °
Zeichnen
EPSG:28992

More OS solutions

UML to DB schema:

ShapeChange: https://github.com/ShapeChange

xmi2db: https://github.com/pkorduan/xmi2db

ili2db: https://github.com/claeis/ili2db

ETL:

stetl: https://github.com/geopython/stetl

You know more? Please, tell me!

NoSQL

- Schemaless = support for extendable GMLAS out-of-the-box
- Fast I/O, ideal as application state, e.g. GeoRocket (MongoDB)
- **Not** relational vs. NoSQL, relational **AND** NoSQL
- Existing research: MongoDB, BaseX, Neo4j, ArangoDB

Conclusion

- Customizable mapping with graphs
- Still a very complex topic
- But entrance barrier is getting lower
- ADEs: More independence, more reliable
- What is the right tool?
  - CityGML? > 3DCityDB
  - GML to DB? > GDAL
  - GML to DB + WFS > deegree
  - DB to GML > HALE + Geoserver
  - Just storage > NoSQL

That's it. Questions?

Felix Kunde

Beuth Hochschule

@FlxKu

Slides: **https://slides.com/fxku/gmlas_db_foss4g**