ESRF | The European Synchrotron

SAXS, EM, SSX

Historically two fundamentally different approaches:

- **ISPyB/Java/EXI - Completely separate developments**
  - New routes, tables, specifically designated as a `type` i.e. EM

- **SynchWeb - "Feature" development**
  - Create features to fulfil requirements, but not directly tied to a `type`

Separated Development

Code is well isolated

    Easy to add / remove without impacting other parts of codebase

Code duplication

    Increases maintenance cost

Hard to share features

Feature development

More complex code base

    Higher cognitive load

Feature sharing

Reduced code duplication

    Improved maintenance

The European Synchrotron | ESRF

SSX Requirements

sampleSupport: Optional[str]

jetMaterial: Optional[str]

avgXtalSize: Optional[float]

crystalConcentration: Optional[float]

acronym: Optional[str]

bufferName: Optional[str]   **=> Extended sample description**

bufferComposition: Optional[str]

ligandName: Optional[str]

ligandConcentration: Optional[float]

The European Synchrotron | ESRF

Abstraction 101

class Cat

   class Dog

=> Don't copy and paste classes

class Animal:

   class Cat(Animal), class Dog(Animal)

Abstract up commonalities, and specialise in children

class Animal:

   animal_type: str

The European Synchrotron | ESRF

Protein, Buffer, Ligand, Additive, Precipitant, Stock Solution

=> All the same thing.

=> A something with a short name and a chemical description

   +   concentration, maybe pH

Can all be represented by `Protein`

   Can't change the name, only need to care about what it represents

   `Component` => Documentation

In MX users put things other than `Proteins` in the beam

- **dna, rna, viruses, small molecules**

Database Abstraction

Add qualifiers to designate what each entry in Protein is

Commonalities

   Name, Acronym, Sequence (SMILES)

ComponentType

   protein dna, rna, small molecule

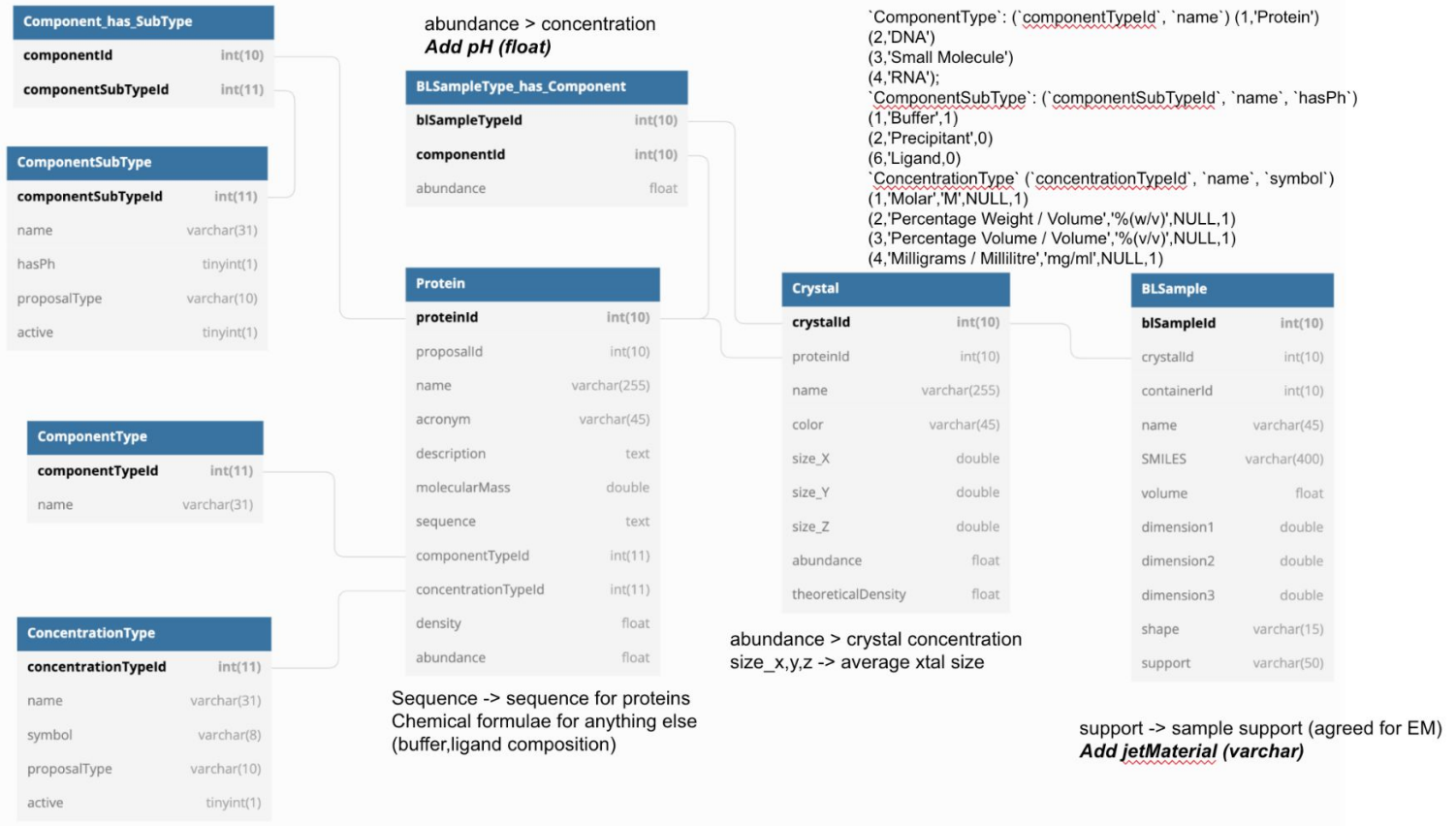**ComponentSubType**

   buffer, ligand, precipitant

CrystalHasProtein

   BLSample -> Crystal -> Protein

Dewar.type = enum('Dewar','Toolbox')

The European Synchrotron | ESRF

Already have the means to store this information!

Document how all of this works (!)

abundance > concentration
Add pH (float)

`ComponentType`: (`componentTypeId`, `name`) (1,'Protein')
(2,'DNA')
(3,'Small Molecule')
(4,'RNA');
`ComponentSubType`: (`componentSubTypeId`, `name`, `hasPh`)
(1,'Buffer',1)
(2,'Precipitant',0)
(6,'Ligand',0)
`ConcentrationType` (`concentrationTypeId`, `name`, `symbol`)
(1,'Molar','M',NULL,1)
(2,'Percentage Weight / Volume','%(w/v)',NULL,1)
(3,'Percentage Volume / Volume','%(v/v)',NULL,1)
(4,'Milligrams / Millilitre','mg/ml',NULL,1)

abundance > crystal concentration
size_x,y,z -> average xtal size

Sequence -> sequence for proteins
Chemical formulae for anything else
(buffer,ligand composition)

support -> sample support (agreed for EM)
Add jetMaterial (varchar)

BLSample ->Crystal -> Protein | Seq = ABVFFRG... | Component Type = Protein | ComponentSubType = None | ConcentrationType = mg/ml
-> BLSampleTypeHasComponent -> Protein | Seq = HEPES | Component Type = Small Molecule | ComponentSubType = Buffer | ConcentrationType = Molar (M)
-> BLSampleTypeHasComponent -> Protein | Seq = CH3CO2H | Component Type = Small Molecule | ComponentSubType = Ligand | ConcentrationType = Molar (M)

Current proposal to link SSX to BioSAXS tables

Duplicate big chunks of core tables

- **Understand why decisions were made at the time**
  - Experiment/SAXSDataCollection/Measurement
  - SamplePlate -> Container
  - Buffer, Additive, Macromolecule -> Protein

In the long (long) term these tables should be refactored

Avoid tying new developments to these tables

   => Can't refactor if they have dependencies

The European Synchrotron | **ESRF**

Try not to tie names to disciplines, rather what they represents

SSXSample

 CrystalSlurry? (Hamburg Proposal)

 Crystal.type = ["single", "slurry"]

SSXDataCollection

 …

 monoBandwidth / monoStripe => DataCollection

Sequence

 Pump / probe

 Sequence.dataCollectionId

The European Synchrotron | **ESRF**

Resource Design

/mx/datacollections

/em/datacollections

/ssx/datacollections

> => All query the same table

> => Will all need similar filters (beamline, session, time spans)

> => Increased cost for UI development

/datacollections

> => Better to have a single resource

> => Use a sample in MX, EM, SSX

>> => Can't view on a single page easily

> => Can lazily query additional info (i.e. GridInfo, EM/SSX Specific info)

The European Synchrotron | ESRF

Resource Design

/mx/sample

/em/sample

/ssx/sample

/samples

> **=> If we can use the Core tables we can reduce code duplication**

> **=> Features benefit everyone**

> **=> Can also reuse existing features**
>
> > **Shipping**
> >
> > **Stats**

Creating SSXSample/SSXBuffer, etc

> **Very tightly coupled to SSX, hard to share**

Every sample in MX uses a buffer and a precipitant

Develop the ability to store this sample information for SSX as a "feature"

- **Using core tables if we can**
  - Can be used by MX, EM, etc

With SQLAlchemy we can reduce complexity of queries

> **Reduced cognitive load**

The European Synchrotron | ESRF