py-ispyb Working Group - User Portal Sync

Responsible: DESY
Participants: ESRF, ALBA

1. Scope

Development of a fairly generic mechanism to ingest the data from the User Portal (UP). The synchronization process will include the sync process for the following entities: proposals (and person relations), sessions (and person relations?), proteins, persons, permissions, laboratories and lab contacts.

2. Deliverables

The deliverables will be the definition of a common format (JSON-based) to import the entities mentioned within the scope, the implementation of the synchronization process together with automatic testings, and a basic documentation. The definition of the schema may be done with a JSON schema (https://json-schema.org/) or maybe generated via pydantic: https://pydantic-docs.helpmanual.io/usage/schema/

3. Requirements

a. Define common format

The definition of the common format should use field names matching the ISPyB database so it is easier to understand for developers (pending for a common agreement).

b. Details of the synchronization process implementation

- i. Define permissions: we need to define which role/permission will be able to execute the User Portal synchronization process (Ex: **User Group**: manager, **Permission**: sync_user_portal). I guess we should start defining and making documentation of the roles/permissions from the beginning.
- ii. Sync of User permissions (Ex: Insert into UserGroup_has_Person table). I guess Groups and Permissions should be defined/agreed by the collaboration? (same from above)
- iii. Define the synchronization process according to the established/agreed format.
  1. Should we implement the sync process per specific entities (Ex: ability to sync only labcontacts), more or less as it was done here (but with a different format):

https://github.com/ispyb/ispyb-client/tree/master/python/userportal/json? (specific endpoints to sync sessions, proposals, lab contacts, etc - independently)

2. Or should we create a general json format including all the related entities to sync per proposal. Ex: https://jsonblob.com/971318471027867648, and an unique endpoint to sync per proposal?

iv. Define a flag to sync the proposal main owner (the “*personId”* within the Proposal table), and the rest of “participants” should go to the “*ProposalHasPerson*” table (Example: participant → "type": "pi").

v. Proposal participants vs session participants: should there be a mechanism to sync also session participants (into Session_has_Person table)?

vi. Dates format: should we use the ISO 8601 for dates in JSON

vii. Difference between siteID, PersonUUID, externalid. I saw that externalid can change, and it is used as a flag to update a person (among other entities).

viii. There should be an endpoint to update a single proposal.

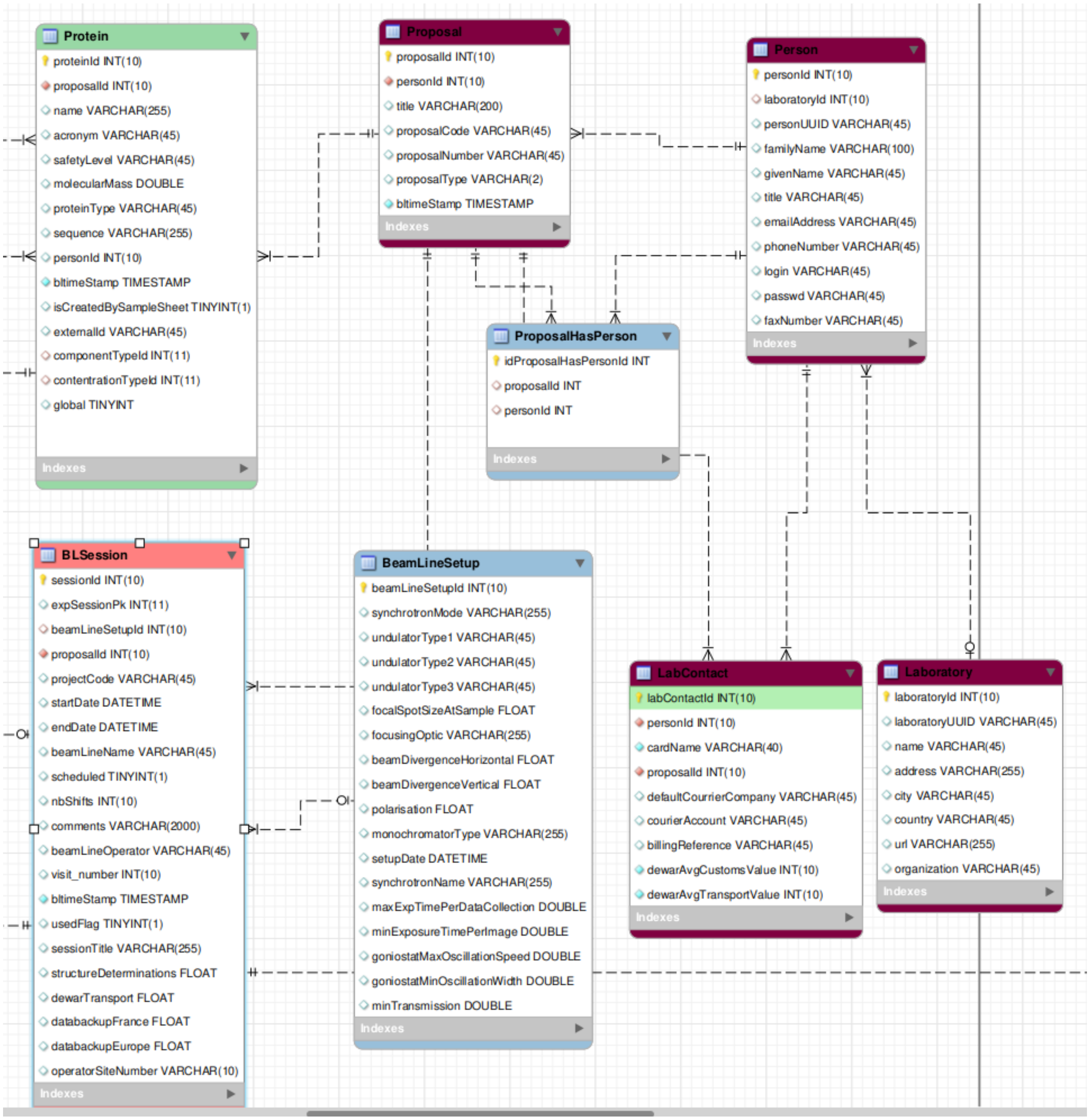
ix. The synchronization must be transactional, meaning that if for example adding a person fails, the whole process of proposal synchronization is rolled back and nothing is committed or inserted into the DB.
https://docs.sqlalchemy.org/en/14/orm/session_transaction.html

x. We will try to create the pydantic models automatically from the SQLAlchemy models
https://github.com/tiangolo/pydantic-sqlalchemy

Please feel free to add any specific implementation details you would like to have.

4. **Specification of synchronization per entity**
    a. **Sessions**
        i. What about the BeamLineSetupId and its relation with the BeamlineSetup Table?. Currently it has a foreign key constraint. Is this managed by MXCuBE? Perhaps we can exclude the BeamLineSetupId field for the import/sync process.
5. **Tables involved (are maybe some missing?)**

Next figure shows some of the tables that are filled during the synch process



Protein

proteinId INT(10)
proposalId INT(10)
name VARCHAR(255)
acronym VARCHAR(45)
safetyLevel VARCHAR(45)
molecularMass DOUBLE
proteinType VARCHAR(45)
sequence VARCHAR(255)
personId INT(10)
bltimeStamp TIMESTAMP
isCreatedBySampleSheet TINYINT(1)
externalId VARCHAR(45)
componentTypeId INT(11)
contentrationTypeId INT(11)
global TINYINT
Indexes

Proposal

proposalId INT(10)
personId INT(10)
title VARCHAR(200)
proposalCode VARCHAR(45)
proposalNumber VARCHAR(45)
proposalType VARCHAR(2)
bltimeStamp TIMESTAMP
Indexes

Person

personId INT(10)
laboratoryId INT(10)
personUUID VARCHAR(45)
familyName VARCHAR(100)
givenName VARCHAR(45)
title VARCHAR(45)
emailAddress VARCHAR(45)
phoneNumber VARCHAR(45)
login VARCHAR(45)
passwd VARCHAR(45)
faxNumber VARCHAR(45)
Indexes

ProposalHasPerson

idProposalHasPersonId INT
proposalId INT
personId INT
Indexes

BLSession

sessionId INT(10)
expSessionPk INT(11)
beamLineSetupId INT(10)
proposalId INT(10)
projectCode VARCHAR(45)
startDate DATETIME
endDate DATETIME
beamLineName VARCHAR(45)
scheduled TINYINT(1)
nbShifts INT(10)
comments VARCHAR(2000)
beamLineOperator VARCHAR(45)
visit_number INT(10)
bltimeStamp TIMESTAMP
usedFlag TINYINT(1)
sessionTitle VARCHAR(255)
structureDeterminations FLOAT
dewarTransport FLOAT
databackupFrance FLOAT
databackupEurope FLOAT
operatorSiteNumber VARCHAR(10)
Indexes

BeamLineSetup

beamLineSetupId INT(10)
synchrotronMode VARCHAR(255)
undulatorType1 VARCHAR(45)
undulatorType2 VARCHAR(45)
undulatorType3 VARCHAR(45)
focalSpotSizeAtSample FLOAT
focusingOptic VARCHAR(255)
beamDivergenceHorizontal FLOAT
beamDivergenceVertical FLOAT
polarisation FLOAT
monochromatorType VARCHAR(255)
setupDate DATETIME
synchrotronName VARCHAR(255)
maxExpTimePerDataCollection DOUBLE
minExposureTimePerImage DOUBLE
goniostatMaxOscillationSpeed DOUBLE
goniostatMinOscillationWidth DOUBLE
minTransmission DOUBLE
Indexes

LabContact

labContactId INT(10)
personId INT(10)
cardName VARCHAR(40)
proposalId INT(10)
defaultCourrierCompany VARCHAR(45)
courierAccount VARCHAR(45)
billingReference VARCHAR(45)
dewarAvgCustomsValue INT(10)
dewarAvgTransportValue INT(10)
Indexes

Laboratory

laboratoryId INT(10)
laboratoryUUID VARCHAR(45)
name VARCHAR(45)
address VARCHAR(255)
city VARCHAR(45)
country VARCHAR(45)
url VARCHAR(255)
organization VARCHAR(45)
Indexes