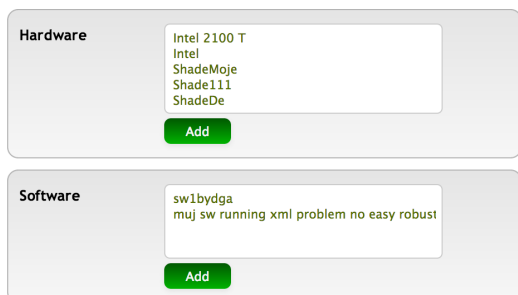


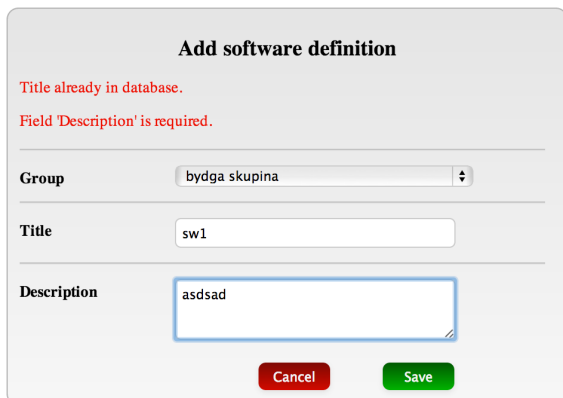
Analýza migrace Postgres -> Elasticsearch

Většina parametrů Experimentu (asi všechny) jsou nachlup stejné. Všechny mají shodně textový titulek a popis, dále referenci na Experiment ke kterému patří a na ResearchGroup. Ta je tam duplicitní - dostanu se na skupinu jednoduše přes experiment. Jediné ospravedlnění je pro našeptávání z již existujících parametrů (našeptávají se jen parametry z researchgrupy):



Tato informace by ale šla vytáhnout nějakým dotazem (v případě ES facet query) a není nutné kvůli tomu držet cca 15 vazebních tabulek parameterX_researchgroup. ResearchGroup vazbu bych tedy z parametrů úplně vyhodil (časem, až dojde na migraci stávajících parametrů - viz dále).

Navíc s tímto propojováním na ResearchGroup jsem objevil zajímavý bug. Není možné vytvořit parametr, jehož titulek již v databázi je. Protože ale aplikace zobrazuje existující parametry jen z aktuální skupiny, cizí parametry nejsou vidět a aplikace nedovolí “nový” parametr vytvořit.



Toto aplikace vypíše, přestože já v seznamu žádný software s názvem sw1 nemám a nevidím.

Postupoval bych tedy tak, že se nadefinuje u experimentu nový field “genericParameter”, na něm se vyzkouší nová funkčnost, ukládání různých dynamických parametrů, hledání nad nimi a optimalizují se query do elasticu. Poté (podle toho jak bude čas) se začnou převádět jednotlivé stávající parametry z postgres do elasticu (na ten GenericParam). Toto bude neméně náročná práce, protože se bude muset upravit mnoho (desítek) tříd, které se stávajícími parametry (pojos + dao) pracují.

Konkrétně se tedy jedná o přesunutí těchto položek do ES a jejich postupné rušení v PG:

```
private Weather weather;
private Digitization digitization;
private SubjectGroup subjectGroup;
private Artifact artifact;
private Timestamp startTime;
private Timestamp endTime;
private int temperature;
private String environmentNote;
private Set<Hardware> hardwares
private Set<Pharmaceutical> pharmaceuticals
private Set<Disease> diseases
private Set<ProjectType> projectTypes
private Set<Software> softwares
private Set<ArtifactRemoveMethod> artifactRemoveMethods
private Set<ExperimentOptParamVal> experimentOptParamVals
```

Je otázka, jestli by nebylo vhodné i spolupracovníky a testovaný subjekt ukládat také do ES. Nyní je to přímo propojené s objektem Person v systému (tzn že nemůžu testovat člověka co není v DB). Ovšem nevím, jestli se podle toho (ted' nebo do budoucna) nebude nějak řešit oprávnění/sdílení, nebo něco jiného... Pokud by jméno subjektu a spolupracovníků měla být jenom doplňující informace k experimentu (nad kterou např. chceme jen hledat), pak by bylo vhodné to taky dát do ES. Jinak to musí zůstat v PG.

Další otazník vizí nad objektem ElectrodeConf. Ten se dál vazbí na spoustu dalších objektů a nevím, jestli jsou takto strukturovaná data k něčemu užitečná. Zatím ho tedy nechávám v pg.

Pro zajímavost přikládám stávající model databáze - [model.svg](#)

Nástin mappingu Experiment do ES (tak jak bude vytvořeno mapování přes PUT Mapping API):

```
experiment: {
  properties: {
    _id: { type: "string" }, //stejně ID jako v pg
    title: { type: "string" }, //titulek experimentu - fulltext
    userId: { type: "integer" }, //kdyby náhodou
    researchGroupId: { type: "integer" }, // pro našepřávání stávajících parametrů při
    //vytváření nových experimentů
    description: { type: "string" }, //popis, fulltext
    startDate: { type: "date" }, //jasný
    endDate: { type: "date" }, //jasný
    parameters: { //PODSTATNÉ: to jsou ony generické parametry
      type: "nested", //nested document - aby nad nimi šlo hledat
      //jak potřebujeme - viz dokumentace
      properties: {
        name: { type: "string" }, //název/typ parametru (např. HW/SW/teplota)
        value_integer: { type: "integer" }, //číselná hodnota (pokd existuje (např. teplota)) -
        //aby šly provádět číselné operace (range query)
        value_string: { type: "string" }, //hodnota - vyhledatelná
        additionalInfo: { //libovolné doplňující informace k parametru
          properties: {
            description: { type: "string" },
            someOther: { type: "string" }
            //.... cokoliv
          }
        }
      }
    }
  }
}
```

```
}
```

V mapování chybí definice fulltext indexů a analyzerů/tokenizerů, ale to není pro tuto ukázkou potřeba. Konkrétní experiment potom může vypadat následovně:

```
{
  "title": "unava za volantem",
  "description": "text nejaky slovní popis o co jde",
  "userId": 12346,
  "researchGroupId": 543,
  "parameters": [
    {"name": "temperature", "value_integer": 28, "value_string": "28"},
    {"name": "hardware", "value_string": "cepicka", "additionalInfo": {"description": "červená čepička, suší moc", "someOther": "other unnecessary value"}},
    {"name": "hardware", "value_string": "kreslo1", "additionalInfo": {"description": "ss"}},
    {"name": "hardware", "value_string": "Intel Core i5"},
    {"name": "software", "value_string": "corel draw"},
    {"name": "tag", "value_string": "unava"},
    {"name": "tag", "value_string": "rizeni"},
    {"name": "tag", "value_string": "spanek"}
  ]
}
```

Query na hledání podle nějakého parametru vypadá takto (vím, vypadá to hrozně, ale to je elasticsearch :))

Zde je hledání podle dvou kritérií: všechny experimenty, kde teplota je > 21 a < 80 (samozřejmě se lze ptát na různé fieldy zároveň (např. teplota+tagy), jenom zde jsem náhodou uvedl stejné:

```
{
  "filter": {
    "nested": {
      "path": "parameters",
      "filter": {
        "and": [
          {
            "bool": {
              "must": [
                {"term": {"parameters.name": "temperature"}},
                {"range": {"parameters.value_integer": {"gt": 21}}}
              ]
            }
          },
          {
            "bool": {
              "must": [
                {"term": {"parameters.name": "temperature"}},
                {"range": {"parameters.value_integer": {"lt": 80}}}
              ]
            }
          }
        ]
      }
    }
  }
}
```

Pro úplnost uvádím i facet (něco jako agregace v SQL) query na získání všech hardware parametrů pro konkrétní researchgroupu (našeptávání a jako náhrada za zrušenou vazbu parametr-group):

```
{
  "query": {
    "bool": {"must": [{"term": {"researchGroupId": 1234}}]}
  },
}
```

```

"facets": {
  "hwFacet": {
    "terms": { "field": "value_string" },
    "nested": "parameters",
    "facet_filter": { "term": { "name": "hardware" } }
  }
}

```

Všechny výše uvedené ES mapování a dotazy mám odzkoušené, včetně funkčního hledání nad malým vzorkem dat.

Ještě nevím, jak bude vypadat načítání dat z ES do bussiness java objektů (nikdy jsem to nedělal :)) Zatím jsem koukal na spring-data, o kterém jsme se bavili - ten vypadá nadějně, v repu na githubu je poslední commit cca 2měs. zpátky a obsahuje jednak podporu pro anotace pojo entit na ES mapování, a pak interface pro tvorbu dao/service vrstvy.

Dále jsem našel <https://github.com/dadoonet/es-hibernate-connector> - ale to vypadá že je mrtvé (2 roky nic, celkem 12 commitů).

Závěr

Sjednotíme tedy všechny dodatečné informace o experimentech do elasticu. Časem do této struktury můžeme i přidat automaticky parsovaná data z .eeg souborů. Na všechny tyto úpravy a přídávky bude elastic struktura připravená. Filtrování podle parametrů a fulltext bude také fungovat. Nested documents nám umožní efektivně vyhledávat jak nad “typovanými” parametry experimentu (autor, titulek), tak i nad generickými (hw, sw...), případně nad kombinací obou.

Pokud si odsouhlasíme tuhle strukturu, začal bych dělat na tomto java-hibernate-es propojení.