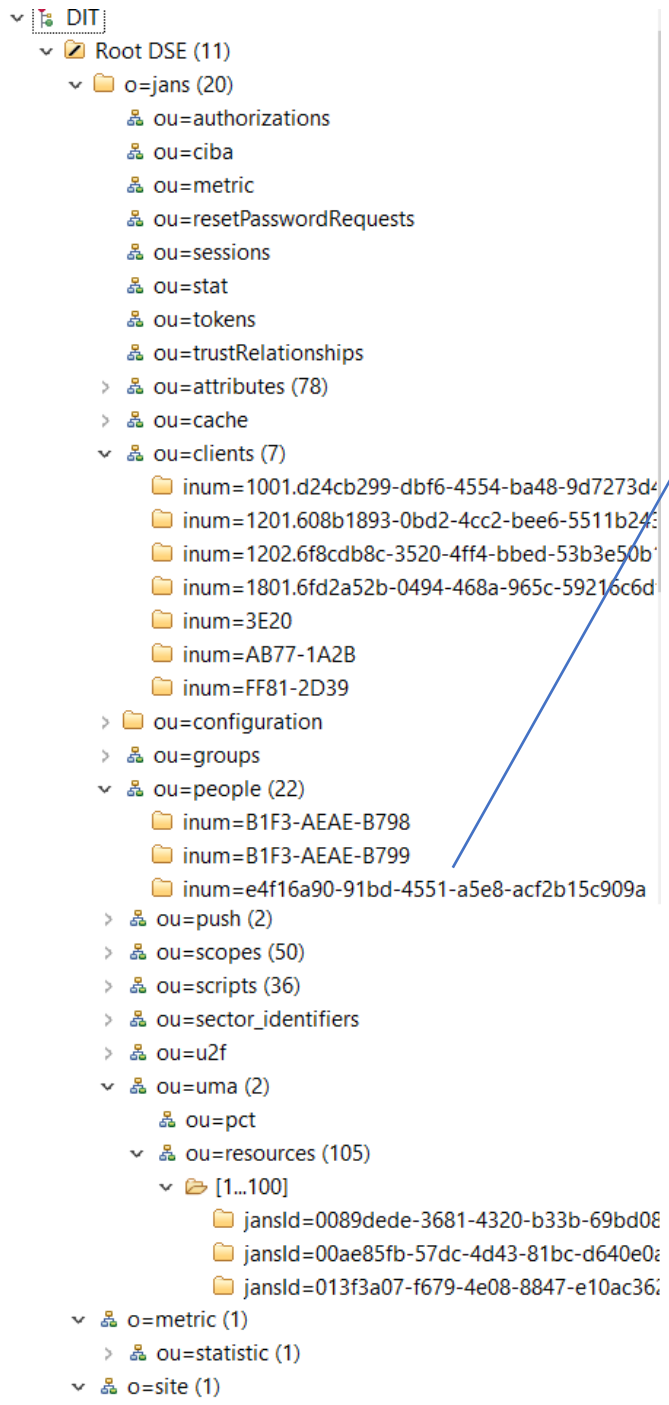

Convert object classes to tables

Table name = objectClass name			
doc_id	e4f16a90-91bd-4551-a5e8-acf2b15c909a	Last RDN value	VARCHAR(48) Primary key, Not null, Unique
objectClass	jansPerson	Last objectClass	VARCHAR(48)
dn	inum=e4f16a90-91bd-4551-a5e8-acf2b15c909a,ou=people,o=jans	LDAP DN	VARCHAR(128)
attr1 (from OC)	attr1 value	Attribute value	
attr2 (from OC)	attr2 value		
…			

Convert attributes based on these rules

Schema type	SQL Type	
1.3.6.1.4.1.1466.115.121.1.7	Boolean syntax	TINYINT
1.3.6.1.4.1.1466.115.121.1.12	Distinguished name syntax	TINYTEXT
1.3.6.1.4.1.1466.115.121.1.15	Directory string syntax	VARCHAR(127) / TINYTEXT / TEXT / Type from schema if defined
1.3.6.1.4.1.1466.115.121.1.24	Generalized Time syntax	DATETIME(3)
1.3.6.1.4.1.1466.115.121.1.27	Integer syntax	INT
Each type above with multivalued=true	json	Store values as JSON types in JSON array

Extended rules to specify required type

```
{
    "desc": "jans user id",
    "equality": "caseIgnoreMatch",
    "names": [
      "jansUsrId",
      "usrId"
    ],
    "oid": "jansAttr",
    "substr": "caseIgnoreSubstringsMatch",
    "syntax": "1.3.6.1.4.1.1466.115.121.1.15",
    "x_origin": "Jans created attribute“,
    “db_data_type": {“db” : “mysql”, “type” : “VARCHAR”, “size” : 47}
  },
```

Extended rules to convert Directory Strings

1. Default string size is 47.
2. Override string size with size from “db_data_type” if exists.

3. Use VARCHAR(size) if length <= 127
4. Use TINYTEXT if 128 < length <= 255
5. Use TEXT if length > 255

Case when attribute defined in system schema

```
{
    "names": [
      “userPassword"
    ],
    "syntax": "1.3.6.1.4.1.1466.115.121.1.15",
    "x_origin": "Jans created attribute“,
    “db_data_type": {“db” : “mysql”, “type” : “VARCHAR”, “size” : 63}
  },
```

JSON attributes

In order to avoid MySQL limitation:
Error Code: 1235. This version of MySQL doesn't yet support 'CAST-ing JSON OBJECT type to array' 0.047 sec

SQL ORM stores data in JSON in next format instead of storing them as JSON array:
'{"v": [94582,94536]}’
'{"v": [2021-02-01T21:18:28.382, 2021-03-01T21:18:28.382]}’
'{"v": [“value_1", “value_2"]}'

JSON indexes

This index will be used to check if array contains specific value

```
ALTER TABLE jans.jansPerson ADD INDEX jansExtUidValues( (CAST(jansExtUid->’$.v' AS UNSIGNED ARRAY)) );
ALTER TABLE jans.jansPerson ADD INDEX address( (CAST(address->'$.v' AS CHAR(48)  ARRAY)) );
```

UNSIGNED/ CHAR(48) are used here as reference. Instead of it valid data type should be used.

Sample query:

```
SELECT COUNT(*) FROM jans.jansPerson doc
WHERE doc.objectClass = 'jansPerson' AND JSON_CONTAINS(doc. jansExtUid->'$.v', CAST(‘[243]' AS JSON))
```

This index is need to compare specific value from array

```
ALTER TABLE jans.jansPerson ADD INDEX jansExtUidValue0( (CAST(jansExtUid->’$.v[0]' AS UNSIGNED)) );
ALTER TABLE jans.jansPerson ADD INDEX jansExtUidValue1( (CAST(jansExtUid->’$.v[1]' AS UNSIGNED)) );
ALTER TABLE jans.jansPerson ADD INDEX jansExtUidValue2( (CAST(jansExtUid->’$.v[2]' AS UNSIGNED)) );
```

This index setup should create for first 3 elements of array only. Admin can add indexes for next elements later if needed.

Sample query:

```
SELECT COUNT(*) FROM jans.jansPerson doc
WHERE doc.objectClass = 'jansPerson' AND jansExtUid->’$.v[0]'  > 3 or jansExtUid->’$.v[1]'  > 3 or jansExtUid->’$.v[2]'  > 3
```