Unité d'Enseignement en Informatique
Année 2014-2015

Master M1 EFREI – ASI – BI
Devoir Écrit de ERP – Second Session June, 2015
(1h00 duration, no document allowed)

Rule: *All answers must be written in English.*

Exercise 1: (2 points)

1) What do the letters MRP stand for? What does it allow to manage?
2) What are the 3 main technical particularities which characterize ERP software?
3) What language is used to develop the modules of SAP ERP?
4) What language is used to develop the modules of OpenERP?

Exercise 2: (2 points)

An analysis of three ERP software solutions, showing which business applications are integrated (I) to the central core of the ERP and which are installable modules (M), and showing different technical features, is presented below:

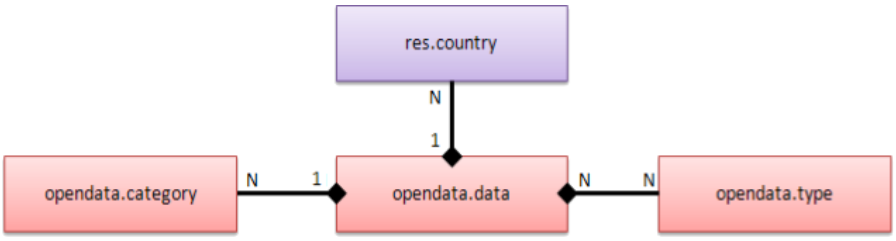
Business Application	ERP1	ERP2	ERP3
Sales	I	M	I
Purchasing		M	I
Warehouse Management	I	M	I
CRM		M	I
MRP	I	M	I
Accounting		M	M
Human Resources		M	M
CMMS	I	M	M

Features	ERP1	ERP2	ERP3
DBMS	SQL Server	Oracle	Oracle; PostgreSQL
Workflow Engine	No	Yes	Yes
Webservice	No	XML-RPC; JSON	JSON

1) Which of these ERP would you not consider as complete ERP software? Why?
2) Which one would you recommend for a big international manufacturing company? Why?
3) Which one would you recommend for a national medium-sized manufacturing company? Why?

Exercise 3: (6 points)

Some companies are leaders on Big Data market and are specialized on Open Data treatment and analysis. In order to help these companies managing a huge number of Open Data download points, we started to develop a new OpenERP module. The data model of this new module is detailed below. An Open Data is hosted and produced in a country, and a country can host and produce several Open Data. Moreover, different types of Open Data exist and it can be available under different formats (CSV, XML, JSON …). Thus, each Open Data can belong to different types, and several Open Data can be of the same type. Finally, each Open Data belongs to one category, and a category can gather several Open Data.



The writing of source code of `opendata.category` and `opendata.data` has already begun. See the two files `opendata.py` and `opendata_view.xml` below, and also the figure 2 illustrating the workflow of an Open Data. In order to continue this development, you have to answer the following questions.

1) The line 34 of the file `opendata.py` was erased. What column declaration is missing there? Give the code of this line.

2) We defined the column `state` for the business object `opendata.data`. Why is this column important? What is it used for?

3) Now, we want to record the dates since an Open Data has been opened or been maintained. Add the two new fields `date_open` and `date_maintain` to the `opendata.data` model.

4) The methods `data_open()` and `data_maintain()` are called when an Open Data enters the states *open* and *maintain*, respectively. Modify these functions in order to initialize the two fields `date_start` and `date_end` in there.

5) Create the search view for the business object `opendata.data` and define the following filtering:

a) Filter Open Data which entered the state *maintain* on the current day

b) Filter Open Data which entered the state *maintain* less than one year ago

c) Filter Open Data which entered the state *maintain* more than one year ago

opendata.py

```
1  from openerp.osv import osv
2  from openerp.osv import fields
3  from openerp.tools.translate import _
4  import time
5
6  listFMT = [('xml', 'Extensible Markup Language'), ('csv', 'Comma-separated values'), \
7            ('json', 'JavaScript Object Notation'), ('xls', 'Microsoft Excel'), \
8            ('rdf', 'Resource Description Framework')]
9
10 class opendata_category(osv.osv):
11     """ The category of Open Data objects """
12     _name = "opendata.category"
13     _description = "The category of Open Data objects"
14     _columns = {
15         'name': fields.char('Category', size=64, required=True),
16         'description': fields.char('Description', size=512, required=True),
17         'public': fields.boolean('Public organism', required=True),
18         'license': fields.char('License', size=64, required=True),
19     }
20     _sql_constraints = [
21         ('name', 'unique(name)', 'The name of a category must be unique')
22     ]
23     _order = 'name asc'
24
```

```
class opendata_data(osv.osv):
    """ A data which is an Open Data """
    _name = "opendata.data"
    _description = "A data which is an Open Data"
    _columns = {
        'name': fields.char('Name', size=64, required=True),
        'url': fields.char('Uniform Resource Locator', size=256, required=True),
        'country_id': fields.many2one('res.country', 'Country', required=False),
        'type_ids': fields.many2many('opendata.type', string='Type', required=False),

        'periodicity': fields.integer('Periodity of updates', required=True),
        'state': fields.selection(
            [
                ('draft','Draft'),
                ('open','Open'),
                ('maintain','Maintain'),
                ('close','Close'),
                ],
            'Status', readonly=True, track_visibility='onchange',
            )
    }
    _sql_constraints = [
        ('name', 'unique(name)', 'The name of a data must be unique')
    ]
    _order = 'name asc'


    def data_draft (self, cr, uid, ids, context={}):
        return self.write(cr, uid, ids, {'state': 'draft'}, context=context)

    def data_open (self, cr, uid, ids, context={}):
        return self.write(cr, uid, ids, {'state': 'open'}, context=context)

    def data_maintain (self, cr, uid, ids, context={}):
        return self.write(cr, uid, ids, {'state': 'maintain'}, context=context)

    def data_close (self, cr, uid, ids, context={}):
        return self.write(cr, uid, ids, {'state': 'close'}, context=context)

```

opendata_view.xml

```
<?xml version="1.0"?>
<openerp>
  <data>
    <!-- Opendata Category: Form View -->
    <record model="ir.ui.view" id="view_opendata_category_form">
      <field name="name">opendata.category.form</field>
      <field name="model">opendata.category</field>
      <field name="arch" type="xml">
       <form string="Category of Open Data" version="7.0">
         <label for="name"/><field name="name"/>
         <label for="description"/><field name="description"/>
         <label for="license"/><field name="license"/>
         <label for="public"/><field name="public"/>
       </form>
      </field>
    </record>

    <!-- Opendata Category: Tree View -->
    <record model="ir.ui.view" id="view_opendata_category_tree">
      <field name="name">opendata.category.tree</field>
      <field name="model">opendata.category</field>
      <field name="field_parent"></field>
      <field name="arch" type="xml">
       <tree string="Category of Open Data">
         <field name="name"/>
         <field name="license"/>
          <field name="public"/>
       </tree>
      </field>
    </record>
```

```
31
32      <!-- Opendata Category: Search View  -->
33      <record model="ir.ui.view" id="view_opendata_category_search">
34        <field name="name">opendata.category.search</field>
35        <field name="model">opendata.category</field>
36        <field name="arch" type="xml">
37          <search string="Models of Open Data">
38            <filter string="Public organism" domain="[('public','=', True)]" help="Data
    producer"/>
39            <filter string="Private organism" domain="[('public','=', False)]" help="Data
    producer"/>
40            <group expand="0" string="Group By...">
41              <filter string="licence" help="Access license" context="{'group_by':'license'}"/>
42            </group>
43          </search>
44        </field>
45      </record>
46
47      <!-- Opendata Category: Action -->
48      <record model="ir.actions.act_window" id="action_opendata_category">
49        <field name="name">Categories</field>
50        <field name="res_model">opendata.category</field>
51        <field name="view_type">form</field>
52        <field name="view_mode">tree,form</field>
53        <field name="search_view_id" ref="view_opendata_category_search"/>
54      </record>
55
56      <!-- Top menu item -->
57      <menuitem name="Open Data" id="base.menu_opendata_root" sequence="120"
    groups="base.group_user"/>
58
59      <!-- Menus sections -->
60      <menuitem name="Configuration" id="menu_opendata_configuration"
    parent="base.menu_opendata_root" sequence="2"/>
61
62      <!-- Menus items -->
63      <menuitem name="Categories" id="menu_opendata_categories"
    parent="menu_opendata_configuration" action="action_opendata_category" sequence="1"/>
64
65    </data>
66  </openerp>
```

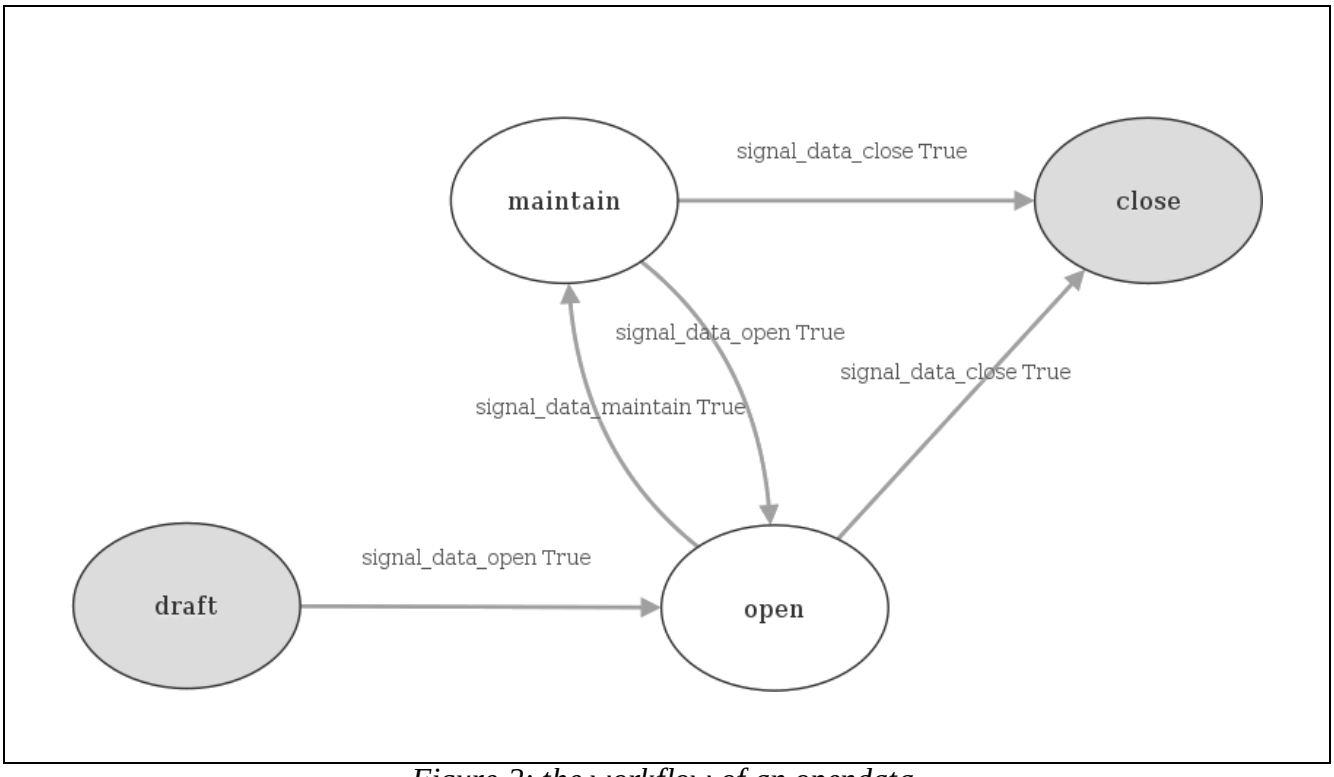


Figure 2: the workflow of an opendata.

Documentation:

a) Type of fields: date

```
fields.date('Field Name' [, Optional Parameters]),
```

b) Get the current date in Python:

```
from datetime import datetime
datetime.now()
```

c) The write method of OpenERP objects, which allows to affect values of fields:
(for example, it is called on lines 48, 51, 54 and 57 of `idea.py` in order to set the value of the field `state`)

`write``(cr, user, ids, vals, context=None)`

Update records with given ids with the given field values

Parameters:
- **cr** – database cursor
- **user** (integer) – current user id
- **ids** – object id or list of object ids to update according to vals
- **vals** (dictionary) – field values to update, e.g {'field_name': new_field_value, ...}
- **context** (dictionary) – (optional) context arguments, e.g. {'lang': 'en_us', 'tz': 'UTC', ...}

Returns: `True`

Raises:
- **AccessError** –
  - if user has no write rights on the requested object
  - if user tries to bypass access rules for write on the requested object
- **ValidateError** – if user tries to enter invalid value for a field that is not in selection
- **UserError** – if a loop would be created in a hierarchy of objects a result of the operation (such as setting an object as its own parent)

d) Comparison operators in XML files:

operator	xml
=	=
≠	!=

operator	xml
<	<
≤	<=

operator	xml
>	>
≥	>=

e) For the filtering of dates:

- Getting the current date:
  `context_today().strftime('%%Y-%%m-%%d')`
- Getting the date 1 year ago from current date:
  `(context_today() - datetime.timedelta(weeks=52)).strftime('%%Y-%%m-%%d')`