Unité d'Enseignement en Informatique
Année 2014-2015

Master M1 EFREI – ASI – ISI
Devoir Écrit de ERP – Deuxième Session de Juin 2015
(durée 1h00, aucun document autorisé)

Consigne: *Réponses en Français ou en Anglais, terminologie Anglophone autorisée dans les deux cas.*

Exercice 1: (2 points)

1) Que signifie le sigle MRP ? Que permet-il de gérer ?
2) Quelles sont les trois particularités techniques qui caractérisent les logiciels ERP ?
3) Quel est le langage de développement des modules pour l'ERP de SAP ?
4) Quel est le langage de développement des modules pour OpenERP ?

Exercice 2: (2 points)

Une analyse de trois ERP, montrant quelles applications métier sont intégrées (I) au composant central de l'ERP et lesquelles sont installables entant que module (M), et montrant différentes options techniques, est présentée dans le tableau suivant :

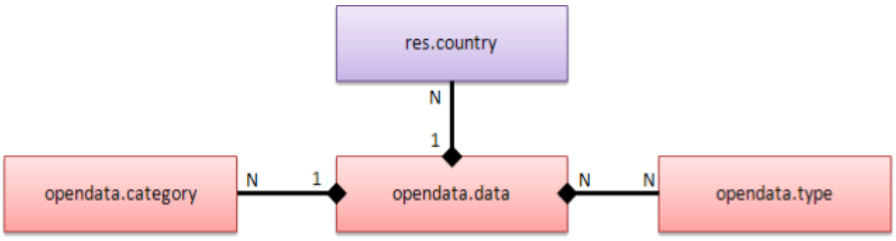
Business Application	**ERP1**	**ERP2**	**ERP3**
Sales	I	M	I
Purchasing		M	I
Warehouse Management	I	M	I
CRM		M	I
MRP	I	M	I
Accounting		M	M
Human Resources		M	M
CMMS	I	M	M

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

1) Lequel de ces ERP ne considéreriez-vous pas comme un logiciel ERP complet ? Pourquoi ?
2) Lequel recommanderiez-vous pour une grande entreprise manufacturière multinationale ? Pour quelles raisons ?
3) Lequel recommanderiez-vous pour une entreprise manufacturière nationale de taille moyenne ? Pour quelles raisons ?

Exercice 3: (6 points)

Des entreprises en pointe sur les Données Massives (ou *Big Data*) sont spécialisées dans le traitement et l'analyse de Données Ouvertes (ou *Open Data*). Afin d'aider ces entreprises à gérer un grand nombre de points de téléchargement de Données Ouvertes, nous avons commencé le développement d'un nouveau module OpenERP. Le modèle de donnée de ce nouveau module est décrit ci-dessous. Une Donnée Ouverte est hébergée et produite dans un pays et un pays peut héberger et produire plusieurs Données Ouvertes. De plus, différents types de Données Ouvertes existent et elles peuvent être disponibles dans différents formats (CSV, XML, JSON …). Ainsi, chaque Donnée Ouverte peut appartenir à différents types et plusieurs Données Ouvertes peuvent être du même type. Enfin, chaque Donnée Ouverte appartient à une catégorie et une catégorie peut regrouper plusieurs Données Ouvertes.



L'écriture du code source de `opendata.category` et `opendata.data` a déjà commencé. Voir les deux fichiers `opendata.py`, et `opendata_view.xml` ci-après, ainsi que la figure 2 illustrant le flux de travail (*workflow*) d'une Donnée Ouverte. Afin de poursuivre ce développement, vous devez répondre aux questions suivantes.

1) La ligne 34 du fichier `opendata.py` a été effacée. Quelle déclaration de colonne manque ici ? Donnez le code de cette ligne.

2) Nous avons définit la colonne `state` de l'objet métier `opendata.data`. Pourquoi cette colonne est-elle importante ? À quoi sert-elle ?

3) Nous voulons maintenant enregistrer les dates quand une Données Ouverte a été ouverte ou est passée en maintenance. Ajoutez deux nouveaux champs `date_open` et `date_maintain` au modèle `opendata.data`.

4) Les méthodes `data_open()` et `data_maintain()` sont appellées quand une Donnée Ouverte entre respectivement dans les états *open* et *maintain*. Modifiez ces fonctions pour que les champs `date_open` et `date_maintain` y soient initialisés.

5) Créer la vue `search` de l'objet métier `opendata.data` et définir les filtrages suivant :

a) Filtrer les Données Ouvertes qui sont entrées dans l'état *maintain* le jour même

b) Filtrer les Données Ouvertes qui sont entrées dans l'état *maintain* moins de 1 an auparavant

c) Filtrer les Données Ouvertes qui sont entrées dans l'état *maintain* plus de 1 an auparavant

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
```

```
25 class opendata_data(osv.osv):
26     """ A data which is an Open Data """
27     _name = "opendata.data"
28     _description = "A data which is an Open Data"
29     _columns = {
30         'name': fields.char('Name', size=64, required=True),
31         'url': fields.char('Uniform Resource Locator', size=256, required=True),
32         'country_id': fields.many2one('res.country', 'Country', required=False),
33         'type_ids': fields.many2many('opendata.type', string='Type', required=False),
34 
35         'periodicity': fields.integer('Periodity of updates', required=True),
36         'state': fields.selection(
37             [
38                 ('draft','Draft'),
39                 ('open','Open'),
40                 ('maintain','Maintain'),
41                 ('close','Close'),
42                 ],
43             'Status', readonly=True, track_visibility='onchange',
44             )
45     }
46     _sql_constraints = [
47         ('name', 'unique(name)', 'The name of a data must be unique')
48     ]
49     _order = 'name asc'
50 
51 
52     def data_draft (self, cr, uid, ids, context={}):
53         return self.write(cr, uid, ids, {'state': 'draft'}, context=context)
54 
55     def data_open (self, cr, uid, ids, context={}):
56         return self.write(cr, uid, ids, {'state': 'open'}, context=context)
57 
58     def data_maintain (self, cr, uid, ids, context={}):
59         return self.write(cr, uid, ids, {'state': 'maintain'}, context=context)
60 
61     def data_close (self, cr, uid, ids, context={}):
62         return self.write(cr, uid, ids, {'state': 'close'}, context=context)
63 
```

opendata_view.xml

```
 1 <?xml version="1.0"?>
 2 <openerp>
 3   <data>
 4     <!-- Opendata Category: Form View -->
 5     <record model="ir.ui.view" id="view_opendata_category_form">
 6       <field name="name">opendata.category.form</field>
 7       <field name="model">opendata.category</field>
 8       <field name="arch" type="xml">
 9        <form string="Category of Open Data" version="7.0">
10          <label for="name"/><field name="name"/>
11          <label for="description"/><field name="description"/>
12          <label for="license"/><field name="license"/>
13          <label for="public"/><field name="public"/>
14        </form>
15       </field>
16     </record>
17 
18     <!-- Opendata Category: Tree View -->
19     <record model="ir.ui.view" id="view_opendata_category_tree">
20       <field name="name">opendata.category.tree</field>
21       <field name="model">opendata.category</field>
22       <field name="field_parent"></field>
23       <field name="arch" type="xml">
24        <tree string="Category of Open Data">
25          <field name="name"/>
26          <field name="license"/>
27           <field name="public"/>
28        </tree>
29       </field>
30     </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>
```

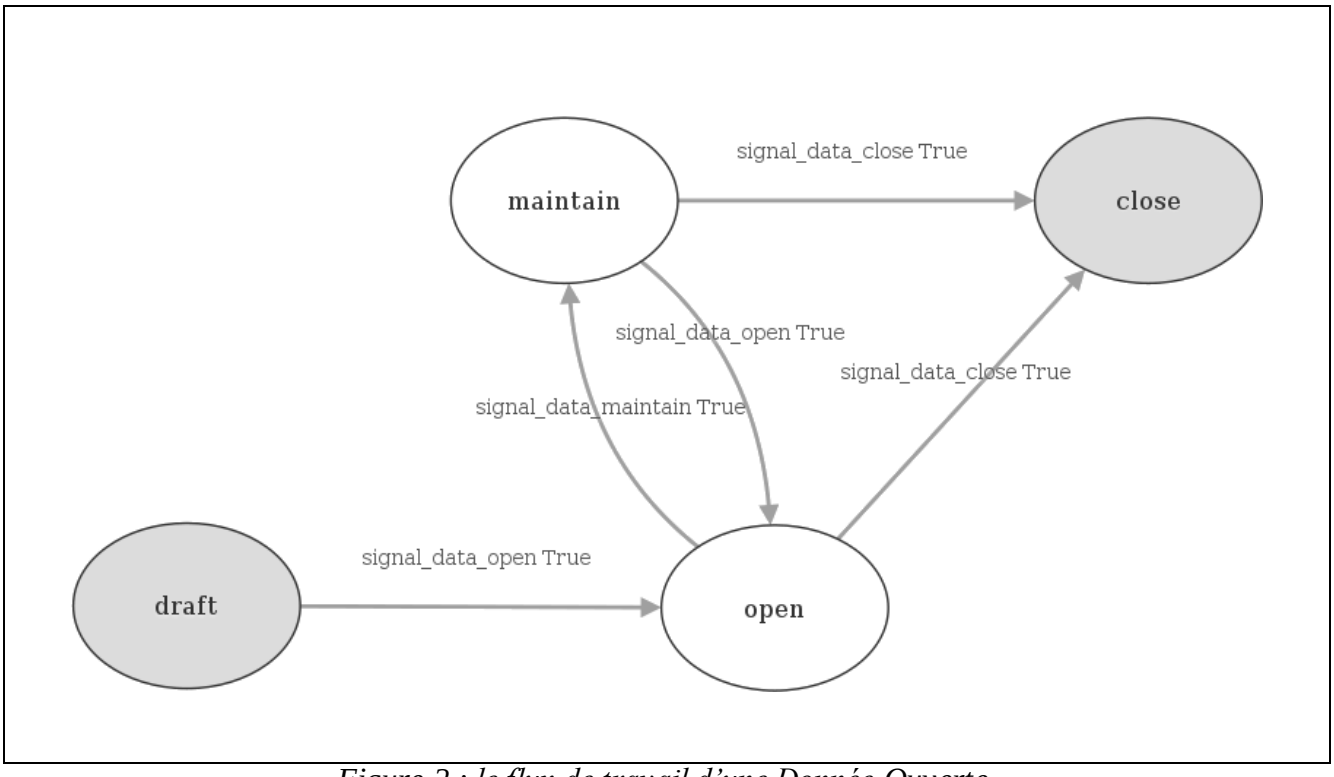


Figure 2 : le flux de travail d'une Donnée Ouverte.

Documentation:

a) Type de champ: date

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

b) Obtenir la date courante en Python :

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

c) La méthode write des objets OpenERP, qui permet d'affecter les valeurs des champs :
(par exemple, elle est appelée lignes 48, 51, 54 et 57 de `idea.py` pour donner sa valeur au champ `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) Opérateurs de comparaison dans les fichiers XML :

opérateur	xml
=	=
≠	!=

opérateur	xml
<	<
≤	<=

opérateur	xml
>	>
≥	>=

e) Pour le filtrage des dates :

- Obtenir la date courante :
  `context_today().strftime('%%Y-%%m-%%d')`
- Pour connaître la date 1 an auparavant :
  `(context_today() - datetime.timedelta(weeks=52)).strftime('%%Y-%%m-%%d')`