Manuel de l'utilisateur de Mocodo 1.4

Aristide Grange*

21 mars 2011

Résumé

Mocodo est un programme Python qui génère des diagrammes entités-associations, ou MCD (modèles conceptuels de données), ainsi que des schémas relationnels, ou MLD (modèles logiques de données), et des tables SQL.

Il prend en entrée une liste d'entités et d'associations décrites dans une syntaxe minimale.

Il produit, pour le MCD, une figure en SVG (Windows, Linux, Mac OS X), ou un dessin Nodebox 1.x (Mac OS X seulement) ; et pour les tables, un ou plusieurs fichiers aux formats html, txt2tags, LaTeX, MySQL, etc.

Son slogan, « nickel, ni souris », en résume les points forts :

- la description textuelle des données. L'utilisateur n'a pas à renseigner, placer et déplacer des éléments comme sous un éditeur WYSIWYG. Il saisit simplement les titres, attributs et cardinalités décrivant son MCD. L'outil s'occupe du plongement ;
- la propreté du rendu. La sortie se fait en vectoriel, prête à être affichée, imprimée, agrandie, exportée dans une multitude de formats sans perte de qualité ;
- la rapidité des retouches. L'utilisateur rectifie les alignements en dupliquant des coordonnées ou en ajustant des facteurs mutiplicatifs : là encore, il travaille sur une description textuelle, et non directement sur le dessin.

Mocodo est libre, gratuit et multiplateforme. Si vous l'utilisez, transmettez la bonne nouvelle en en reproduisant le logo (inclus dans la distribution) sur votre support : cela multipliera ses chances d'attirer des contributeurs qui le feront évoluer.

1 Installation

Pour...	**Windows**	**Debian (Ubuntu, etc.)**	**Mac OS X**
ouvrir l'archive `.tar.gz`	7-zip		
éditer le MCD d'entrée	Notepad++		
lancer le script `mocodo`	Python 2.6 ou 2.7	`sudo apt-get install python-tk`	
voir le MCD produit	Firefox		
le retoucher et l'exporter	Inkscape	`sudo apt-get install inkscape`	Nodebox 1.x

*Université Paul-Verlaine de Metz. Courrier électronique: `grange` (escargot) `univ-metz.fr`

Mocodo lui-même est un simple petit script, qui peut être lancé directement à partir de l'endroit où vous aurez décompressé l'archive téléchargée. Mais il est juché sur des épaules de géants, notamment Python et Nodebox ou Inkscape. Le tableau ci-dessus résume ce que vous devriez avoir à installer selon votre système. Les cases vides signifient que le logiciel requis est pré-installé.

N.B.. Sur Mac et Linux, il se peut que vous ayez à mettre à jour en 2.6 ou 2.7 votre version de Python. Si vous tenez à faire fonctionner Mocodo avec Python 2.5, vous devrez installer le paquetage supplémentaire simplejson.

2 Lancement de l'application

2.1 Au plus simple

Le programme se lance à partir d'une console en tapant :

```
python mocodo.py
```

2.2 Paramétrage occasionnel

Les arguments suivants sont reconnus :

option	description
`-attraction`	Distance horizontale (par défaut 40 pixels) en-deçà de laquelle les centres des boîtes seront alignés verticalement.
`-cleanup`	Recrée dans le répertoire principal les fichiers `default.json` et `sandbox.mcd` à partir de ceux conservés dans le répertoire `pristine`, comme au premier lancement (voir plus loin).
`-colors=...`	Le nom de la palette de couleurs à utiliser, par défaut `bw`. Les palettes sont stockées dans le répertoire `colors`.
`-df=...`	Le sigle à afficher dans une dépendance fonctionnelle, par défaut `DF`.
`-encoding=...`	L'encodage du fichier d'entrée, par défaut `utf8`.
`-help`	Un court message d'aide.
`-input=...`	Le nom du fichier d'entrée, par défaut `sandbox.mcd`.
`-table=...`	Le format de sortie tabulaire (MLD ou MPD), cf. répertoire `tables`.
`-output=...`	Le nom du fichier de sortie, par défaut `sandbox-svg.py` ou `sandbox-nodebox.py` selon la plateforme.
`-sep=...`	Le séparateur à afficher entre les cardinalités minimales et maximales, par défaut une virgule.
`-shapes=...`	Le nom du dictionnaire de définition des polices, dimensions, etc., cf. répertoire `shapes`.
`-version`	Le numéro de version.

Exemple.

```
python mocodo.py --input="test.mcd" --colors="mondrian" --output="test-svg.py" --df="CIF"
```

2.3 Paramétrage à long terme

Au premier lancement, Mocodo essaie de déterminer s'il opère sous Windows, Linux ou Mac OS X. Dans ce dernier cas, il cherche en outre si Nodebox est installé. Il crée alors un fichier de paramètres `default.json` adapté, fichier qu'il relira à chaque lancement ultérieur. Vous êtes encouragés à modifier ce fichier selon vos goûts. De la sorte, le style de vos MCD pourra être maintenu à moindre frais à travers tous vos documents. En cas de besoin, vous pourrez toujours ponctuellement passer outre ces réglages en en précisant d'autres en ligne de commande.

2.4 Accents et symboles spéciaux

Les accents présents dans votre fichier d'entrée sont correctement gérés pourvu que vous ayez enregistré celui-ci dans l'encodage attendu par Mocodo en entrée : c'est par défaut *Unicode utf8*. Une tolérance existe : si Mocodo échoue à décoder votre fichier avec *utf8*, il se rabattra sur le codec d'Europe de l'Ouest associé historiquement à votre plateforme : *iso-8859-15* pour Windows et Linux, *mac-roman* pour Mac OS X.
Si les accents n'apparaissent pas correctement, vous aurez encore trois solutions :
– soit, ré-enregistrer votre fichier en *utf8* (conseillé) ou dans un encodage historique ;
– soit (à long terme), modifier dans `default.json` la liste des encodages pris en charge ;
– soit (ponctuellement), passer l'encodage de votre fichier en argument à `mocodo.py`.

2.5 Autres

Le script `mocodo.py` se borne à analyser ses paramètres et à passer la main à l'un des deux programmes `mcd2nodebox.py` ou `mcd2svg.py`, que vous pouvez donc appeler directement : ils lisent le même fichier `default.json`, mais génèrent un script destiné respectivement à NodeBox ou à générer un fichier SVG.

3 Syntaxe de description du MCD

3.1 Besoins élémentaires

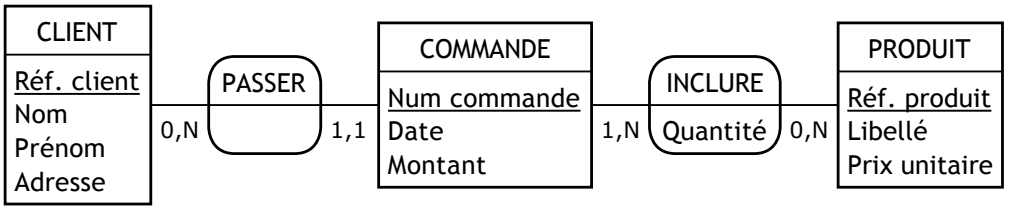
3.1.1 Entités, associations, attributs, identifiants, cardinalités

Fichier d'entrée.

```
CLIENT: Réf. client, Nom, Prénom, Adresse
PASSER, ON CLIENT, 11 COMMANDE
COMMANDE: Num commande, Date, Montant
```

```
INCLURE, 1N COMMANDE, 0N PRODUIT: Quantité
PRODUIT: Réf. produit, Libellé, Prix unitaire
```

Sortie brute. *Cf. diagramme ci-dessous.*

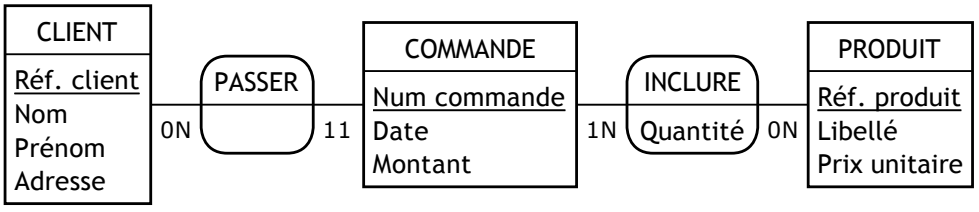


La syntaxe ne devrait pas poser problème. À noter :
- la liste des cardinalités / entités mises en jeu par une association s'intercale entre le nom et l'éventuelle liste d'attributs de celle-ci ;
- le premier attribut d'une entité est considéré par défaut comme son identifiant, et donc souligné ;
- pour les associations sans attributs, le deux-points est facultatif.

Options. Il est possible de remplacer la virgule séparatrice des cardinalités par une chaîne quelconque, par exemple vide :

```
python mocodo.py --sep=""
```

Sortie brute. *Cf. diagramme ci-dessous.*

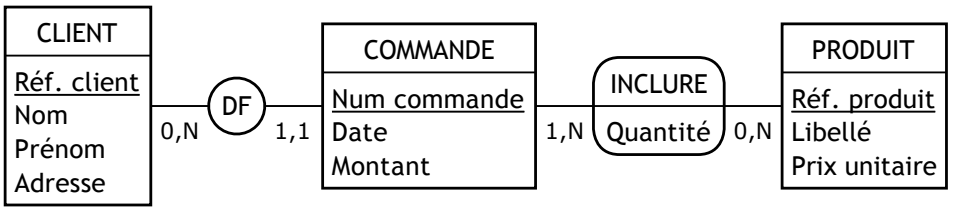


3.1.2 Dépendances fonctionnelles

Fichier d'entrée.

```
CLIENT: Réf. client, Nom, Prénom, Adresse
DF, 0N CLIENT, 11 COMMANDE
COMMANDE: Num commande, Date, Montant
INCLURE, 1N COMMANDE, 0N PRODUIT: Quantité
PRODUIT: Réf. produit, Libellé, Prix unitaire
```

Sortie brute. *Cf. diagramme page ci-contre.*



Options. Il est possible d'activer l'encerclement d'un autre sigle que DF, par exemple :

```
python mocodo.py --df=CIF
```

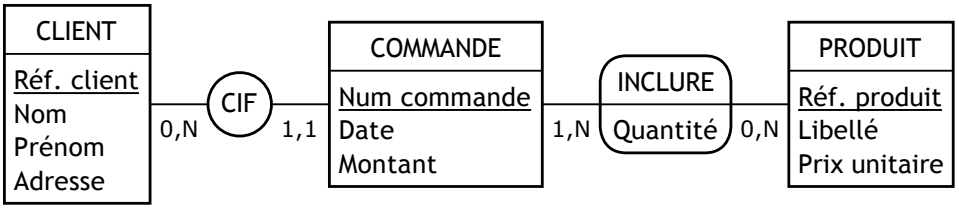
C'est ce sigle qui devra alors apparaître en entrée :

```
...
CIF, ON CLIENT, 11 COMMANDE
...
```

Pour placer le sigle à la bonne hauteur dans ce cercle plus grand, il est de plus nécessaire de modifier (a priori une fois pour toutes) le ratio défini dans le dictionnaire `trebuchet` de `shapes` :

```
"dfTextHeightRatio"             : 1.00,
```

Sortie brute. *Cf. diagramme ci-dessous.*

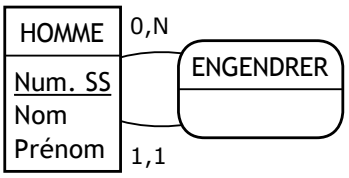


3.1.3 Associations réflexives

Fichier d'entrée.

```
HOMME: Num. SS, Nom, Prénom
ENGENDRER, ON HOMME, 11 HOMME
```

Sortie brute. *Cf. diagramme page suivante.*



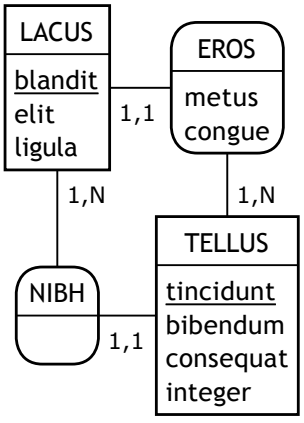
3.1.4 Placement sur plusieurs lignes

Fichier d'entrée.

```
LACUS: blandit, elit, ligula
EROS, 11 LACUS, 1N TELLUS: metus, congue

NIBH, 1N LACUS, 11 TELLUS
TELLUS: tincidunt, bibendum, consequat, integer
```

Sortie brute. *Cf. diagramme ci-dessous.*



L'ordre et la séparation des lignes de la description permet de spécifier à coût zéro un plongement grossier, mais qui se révèle souvent suffisant :
– les éléments sont placés de gauche à droite dans l'ordre de leur énumération ;
– le retour à la ligne se fait en insérant une ligne blanche dans la description.

Par défaut, les centres des entités et des associations d'une même ligne sont alignés horizontalement. Les lignes sont centrées. Depuis la version 1.2, le système tente d'aligner verticalement les centres dont les abscisses sont égales à une constante près. Cette constante, nommée `attraction` peut être spécifiée, soit en argument, soit dans le fichier de paramètres `default.json`. Une valeur de 0 désactive l'attraction.

Les autres types d'alignement ou de décalage demandent à intervenir manuellement sur le fichier de sortie.

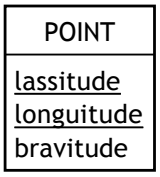
3.2 Besoins plus avancés

3.2.1 Identifiants multiples

Fichier d'entrée.

```
POINT: lassitude, _longuitude, bravitude
```

Sortie brute. *Cf. diagramme ci-dessous.*

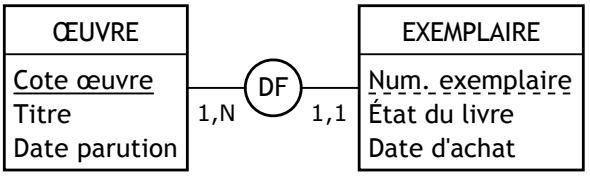


3.2.2 Identifiants faibles

Fichier d'entrée.

```
ŒUVRE: Cote œuvre, Titre, Date parution
DF, 1N ŒUVRE, 11 EXEMPLAIRE
EXEMPLAIRE: -Num. exemplaire, État du livre, Date d'achat
```

Sortie brute. *Cf. diagramme ci-dessous.*



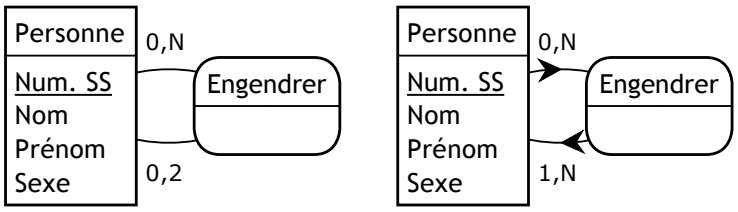
Un soulignement pointillé dénote les identifiants faibles.

3.2.3 Étiquettes et flèches sur les pattes

Fichier d'entrée.

```
Personne1: Num. SS, Nom, Prénom, Sexe
Engendrer1, 0Nenfant Personne1, 02parent Personne1
Personne2: Num. SS, Nom, Prénom, Sexe
Engendrer2, 0N< Personne2, 1N> Personne2
```

Sortie brute. *Cf. diagramme page suivante.*
Les définitions d'étiquettes sont prévues dans la syntaxe, mais encore ignorées par la version actuelle. À partir de la version 1.3, les flèches sont effectivement tracées.



3.2.4 Styles

Plusieurs styles prédéfinis sont distribués avec l'application. Un style se définit comme la combinaison d'une palette de couleurs (répertoire `colors`) avec un dictionnaire de polices et de dimensions (répertoire `shapes`). Vous pouvez bien sûr créer les vôtres en vous inspirant des fichiers fournis. Si vous êtes particulièrement content d'un style, soumettez-le pour inclusion dans une prochaine distribution.

3.2.5 Types

À partir de la version 1.4, chaque attribut peut être assorti d'un type (entre crochets) :

```
CLIENT: Réf. client [varchar(8)], Nom [varchar(20)], Adresse [varchar(40)]
DF, ON CLIENT, 11 COMMANDE
COMMANDE: Num commande [tinyint(4)], Date [date], Montant [decimal(5,2) DEFAULT '0.00']
INCLURE, 1N COMMANDE, ON PRODUIT: Quantité [tinyint(4)]
PRODUIT: Réf. produit [varchar(8)], Libellé [varchar(20)], Prix unitaire [decimal(5,2)]
```

Actuellement ignorés au niveau du MCD, les types sont utiles à la génération d'un modèle physique de données. À titre d'exemple, la distribution inclut le fichier de style `mysql.json` qui sert à établir la sortie suivante, directement importable par MySQL :

```
CREATE TABLE `PRODUIT` (
  `Réf. produit` varchar(8),
  `Libellé` varchar(20),
  `Prix unitaire` decimal(5,2),
  PRIMARY KEY(`Réf. produit`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

CREATE TABLE `CLIENT` (
  `Réf. client` varchar(8),
  `Nom` varchar(20),
  `Adresse` varchar(40),
  PRIMARY KEY(`Réf. client`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

CREATE TABLE `COMMANDE` (
  `Num commande` tinyint(4),
  `Date` date,
  `Montant` decimal(5,2) DEFAULT '0.00',
  `Réf. client` varchar(8),
```

```
  PRIMARY KEY(‘Num commande‘)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

CREATE TABLE ‘INCLURE‘ (
  ‘Num commande‘ tinyint(4),
  ‘Réf. produit‘ varchar(8),
  ‘Quantité‘ tinyint(4),
  PRIMARY KEY(‘Num commande‘, ‘Réf. produit‘)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

ALTER TABLE ‘COMMANDE‘ ADD FOREIGN KEY (‘Réf. client‘) REFERENCES ‘CLIENT‘ (‘Réf. client‘);

ALTER TABLE ‘INCLURE‘ ADD FOREIGN KEY (‘Num commande‘) REFERENCES ‘COMMANDE‘ (‘Num commande‘);

ALTER TABLE ‘INCLURE‘ ADD FOREIGN KEY (‘Réf. produit‘) REFERENCES ‘PRODUIT‘ (‘Réf. produit‘);
```

3.3 Besoins spécifiques à la pédagogie

3.3.1 Laisser vierges des attributs ou des cardinalités

Les MCD à trous sont des exercices classiques d'introduction aux bases de données. Mocodo offre deux méthodes distinctes pour les produire.
Vous pouvez « trouer » n'importe quel MCD en rendant complètement transparentes les couleurs des attributs, associations et cardinalités. Le style `blank` a été prédéfini à cet effet :

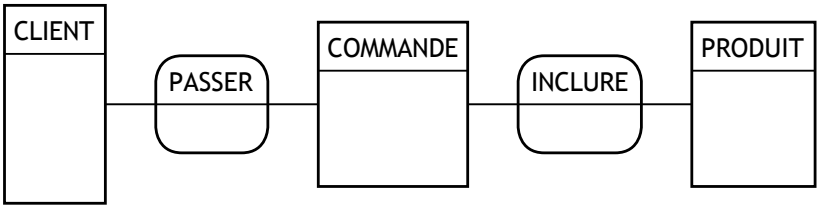
```
python mocodo.py --colors=blank
```

L'autre méthode (plus souple) demande à modifier le MCD d'entrée :

Fichier d'entrée.

```
CLIENT: , , ,
PASSER, XX CLIENT, XX COMMANDE
COMMANDE: , ,
INCLURE, XX COMMANDE, XX PRODUIT:
PRODUIT: , ,
```

Sortie brute. *Cf. diagramme ci-dessous.*



– Une liste de n virgules permet de réserver la place de $n+1$ attributs ;
– les cardinalités `XX` n'apparaissent pas dans le MCD résultant.

3.3.2 Dupliquer des entités ou des associations

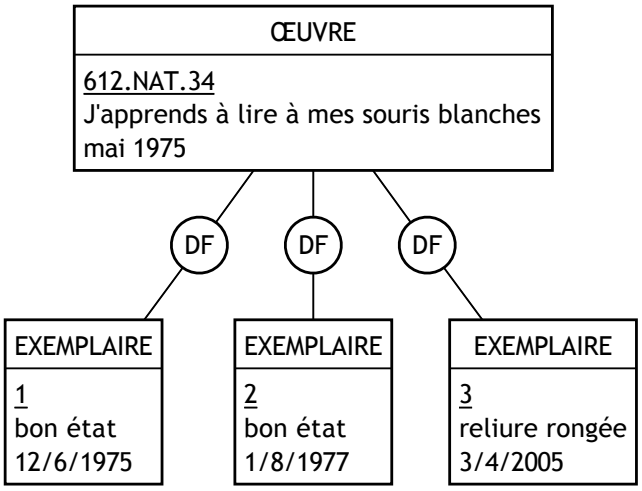
Fichier d'entrée.

```
ŒUVRE: 612.NAT.34, J'apprends à lire à mes souris blanches, mai 1975

DF1, XX ŒUVRE, XX EXEMPLAIRE1
DF2, XX ŒUVRE, XX EXEMPLAIRE2
DF3, XX ŒUVRE, XX EXEMPLAIRE3

EXEMPLAIRE1: 1, bon état, 12/6/1975
EXEMPLAIRE2: 2, bon état, 1/8/1977
EXEMPLAIRE3: 3, reliure rongée, 3/4/2005
```

Sortie brute. *Cf. diagramme ci-dessous.*



Cette fonctionnalité permet de préparer des vues en extension, ou de figurer par exemple par le même symbole DF plusieurs dépendances fonctionnelles d'un MCD en compréhension.

– Les suffixes numériques des entités ou des associations n'apparaissent pas dans le MCD résultant.

4 Post-traitement de la sortie graphique

Le plongement automatique réalisé par Mocodo pour des MCD de plusieurs lignes peut généralement être amélioré.

Fichier d'entrée.

```
Lacus, 01 Blandit, 1N Elit
Elit: ligula, tellus
```

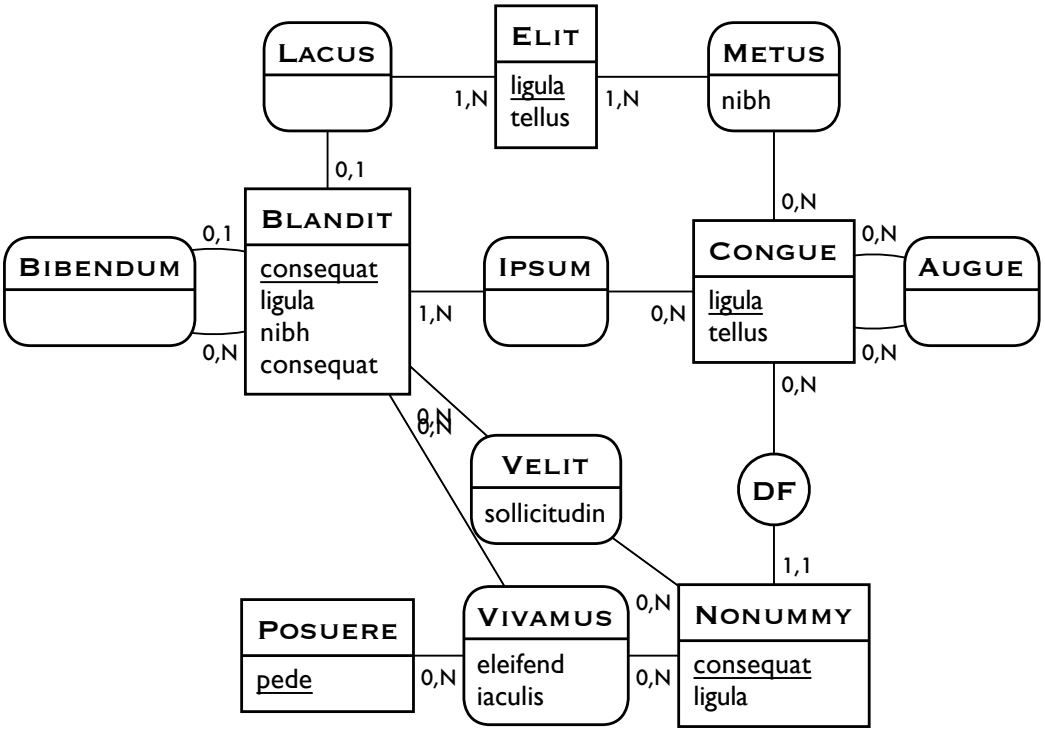
```
Metus, 1N Elit, ON Congue: nibh

Bibendum, O1 Blandit, ON Blandit
Blandit: consequat, ligula, nibh, consequat
Ipsum, 1N Blandit, ON Congue
Congue: ligula, tellus
Augue, ON Congue, ON Congue

Velit, ON Blandit, ON Nonummy: sollicitudin
DF, 11 Nonummy, ON Congue

Posuere: pede
Vivamus, ON Posuere, ON Nonummy, ON Blandit: eleifend, iaculis
Nonummy: consequat, ligula
```

Sortie brute. *Cf. diagramme ci-dessous.*



Depuis la version 1.3, Mocodo est à parité sur toutes les plateformes. Cela signifie qu'au lieu de générer un dessin statique, il génère systématiquement un script, qui lui-même générera le dessin. Quel est l'avantage de cette couche supplémentaire ? Eh bien, le script intermédiaire exprime la plupart des positions non en absolu, mais en relation à d'autres positions, peu nombreuses et qui constituent de fait les véritables paramètres. Sa capacité à évaluer des formules le rend beaucoup plus souple et plus puissant que si les valeurs résultantes étaient stockées en dur.

4.1 Retouches manuelles du script intermédiaire

Par défaut, Mocodo génère un fichier appelé `sandbox-nodebox.py` (sur Mac OS X avec NodeBox installé) ou `sandbox-svg.py` (dans tous les autres cas). Ce fichier est un script qui devra être exécuté respectivement dans l'environnement NodeBox, ou en tapant :

```
python sandbox-svg.py
```

pour générer effectivement le dessin. Le début de ce script intermédiaire ressemble à ceci :

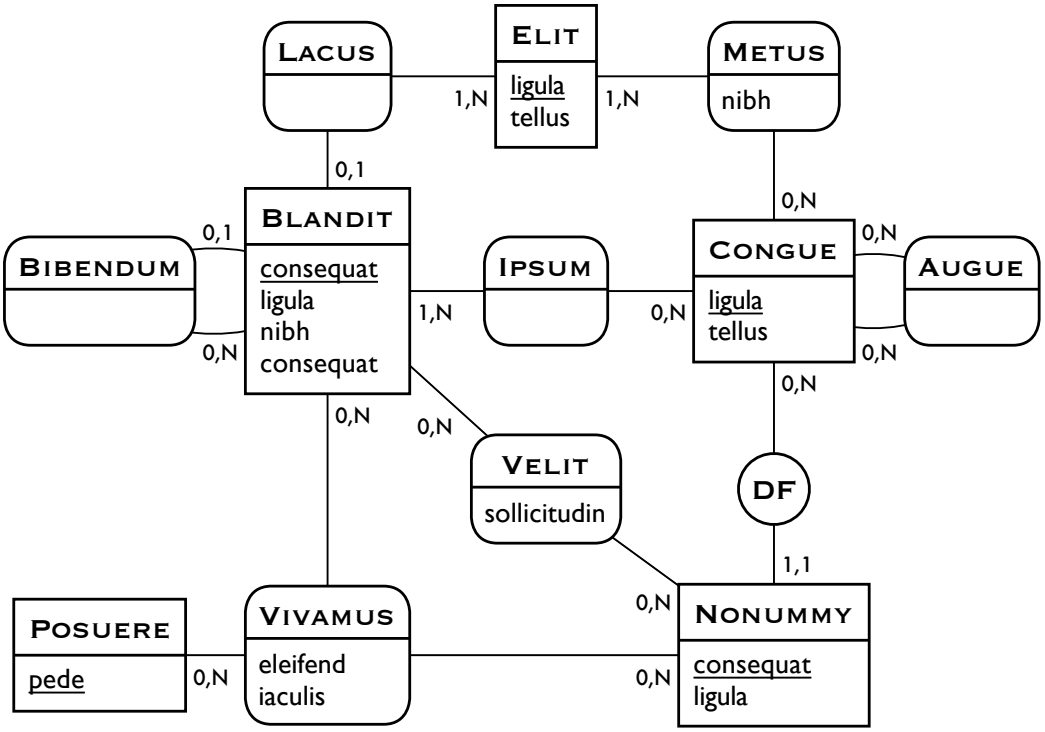
```
cx = {
  u"Lacus": 182,
  u"Elit": 299,
  u"Metus": 421,
  u"Bibendum": 60,
  u"Blandit": 182,
  u"Ipsum": 299,
  u"Congue": 421,
  u"Augue": 527,
  u"Velit": 299,
  u"DF": 421,
  u"Posuere": 182,
  u"Vivamus": 299,
  u"Nonummy": 421,
}
cy = {
  u"Lacus": 47,
  u"Elit": 47,
  u"Metus": 47,
  u"Bibendum": 162,
  u"Blandit": 162,
  u"Ipsum": 162,
  u"Congue": 162,
  u"Augue": 162,
  u"Velit": 268,
  u"DF": 268,
  u"Posuere": 357,
  u"Vivamus": 357,
  u"Nonummy": 357,
}
```

Ces lignes exposent les principaux paramètres de position : le dictionnaire `cx` (resp. `cy`) correspond aux abscisses (resp. ordonnées) des centres des entités et associations. Supposons que l'on souhaite aligner verticalement les centres de POSUERE et VIVAMUS avec ceux, respectivement, de BIBENDUM et BLANDIT. Dans `cx`, il suffit de copier la valeur de celles-ci dans celles-là :

```
cx = {
  ...
  u"Bibendum": 60,       # abscisse de référence
```

```
  u"Blandit": 182,       # abscisse de référence
  ...
  u"Posuere": 60,        # abscisse modifiée
  u"Vivamus": 182,       # abscisse modifiée
  ...
}
```

Sortie après retouche. *Cf. diagramme ci-dessous.*



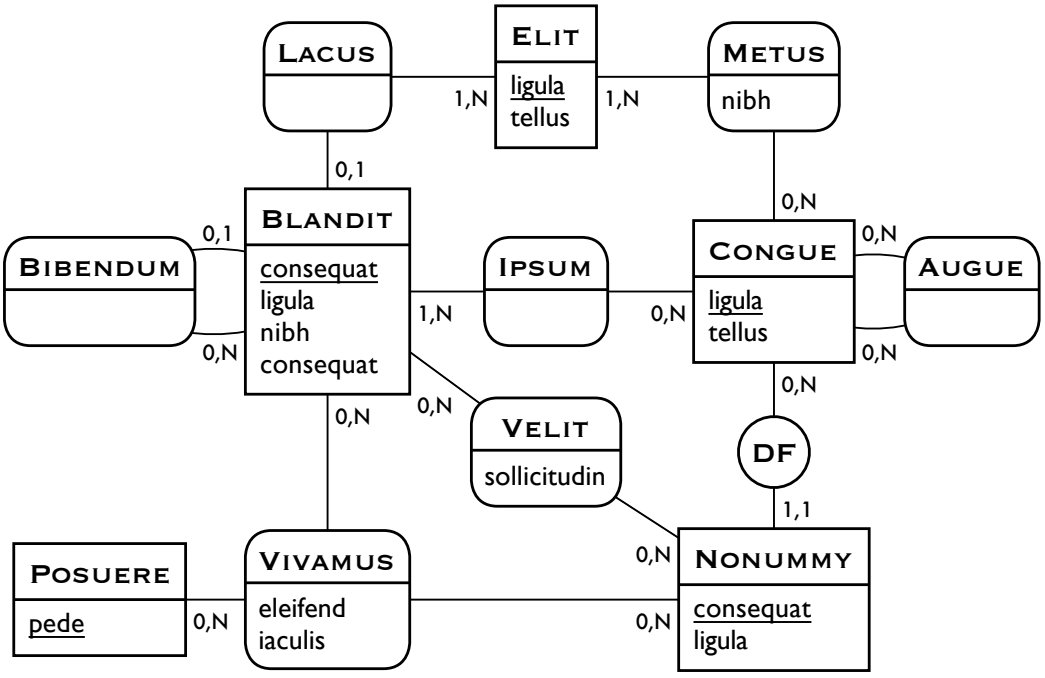
Si l'on juge que la partie inférieure de la figure occupe trop de place, on remontera les boîtes concernées en intervenant sur les ordonnées.

```
cy = {
  ...
  u"Velit": 248,         # ordonnée diminuée de 20 pixels
  u"DF": 248,            # ordonnée diminuée de 20 pixels
  u"Posuere": 327,       # ordonnée diminuée de 30 pixels
  u"Vivamus": 327,       # ordonnée diminuée de 30 pixels
  u"Nonummy": 327,       # ordonnée diminuée de 30 pixels
  ...
}
```

Les dimensions calculées à l'origine pour l'image ne tiennent compte ni des cardinalités, ni bien sûr des retouches ultérieures. Il peut donc arriver que son cadre soit trop petit ou trop grand pour le MCD. C'est le cas maintenant. Il faut ajuster sa taille, qui se trouve au tout début du fichier :

```
(width,height) = (572,374)            # ordonnée diminuée de 30 pixels
```

Sortie après retouche. *Cf. diagramme ci-dessous.*



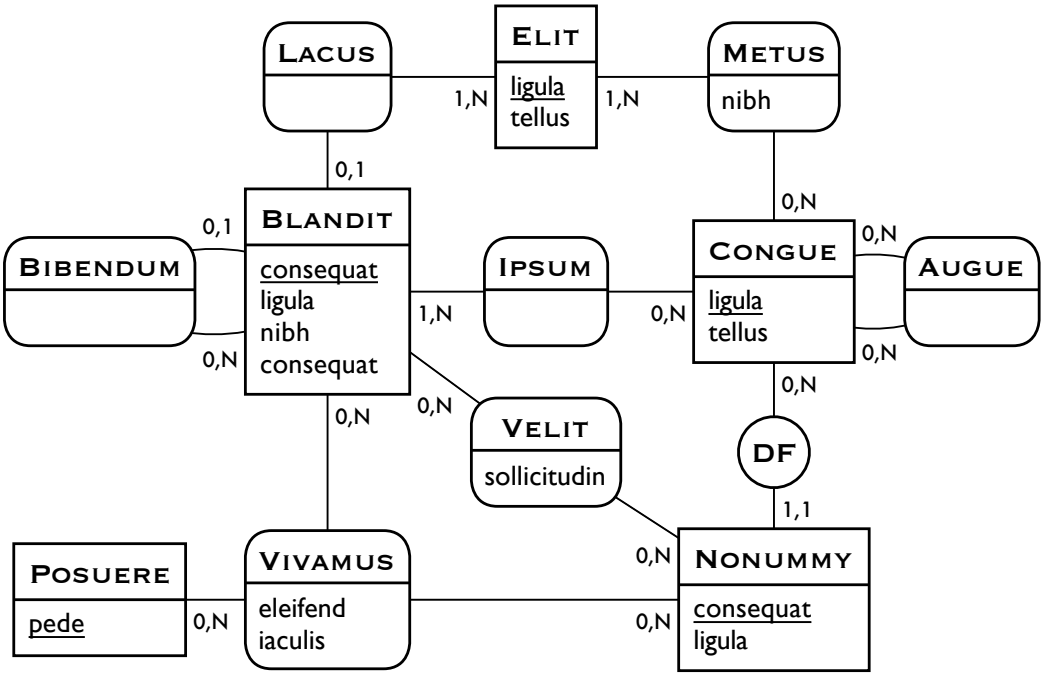
On souhaite maintenant corriger la position de certaines cardinalités. Ces positions se trouvent dans le dictionnaire `k` :

```
k = {
  u"Lacus,Blandit": 1,
  u"Lacus,Elit": 1,
  u"Metus,Elit": 1,
  u"Metus,Congue": 1,
  u"Bibendum,Blandit,-1.0": -4.0,
  u"Bibendum,Blandit,1.0": 4.0,
  u"Ipsum,Blandit": 1,
  u"Ipsum,Congue": 1,
  u"Augue,Congue,-1.0": -4.0,
  u"Augue,Congue,1.0": 4.0,
  u"Velit,Blandit": 1,
  u"Velit,Nonummy": 1,
  u"DF,Nonummy": 1,
  u"DF,Congue": 1,
  u"Vivamus,Posuere": 1,
  u"Vivamus,Nonummy": 1,
  u"Vivamus,Blandit": 1,
}
```

La valeur `1` est un facteur multiplicatif qui signifie : placer la cardinalité au-dessous de la patte et aussi haut que possible pour laisser une marge suffisante entre les deux. Le facteur `-1` signifierait la même chose, en inversant le haut et le bas. Logiquement, un facteur de `0` placerait la cardinalité sur la patte elle-même. Toutes les valeurs sont possibles, mais en pratique, quelques changements de signe sont en général suffisants pour éviter les collisions. Ici, aucune collision ne se produit, mais on peut souhaiter détacher légèrement les cardinalités de l'association réflexive BLANDIT :

```
k = {
  ...
  u"Bibendum,Blandit,-1.0": -4.5,
  u"Bibendum,Blandit,1.0": 4.5,
  ...
}
```

Sortie après retouche. *Cf. diagramme ci-dessous.*



La même technique s'applique pour faire glisser les flèches le long des arcs : le dictionnaire à modifier s'appelle `t`.
Un plus ou moins grand nombre de retouches peuvent être nécessaires, mais vous avez l'idée. D'aucuns y verront peut-être une dangereuse confusion entre données, programme et résultats ; d'autres, une application naturelle du dynamisme des langages du même nom ; au total, chacun décidera s'il est plus efficace et plus précis en faisant glisser des objets dans son cliquodrome favori, ou en ajustant des valeurs numériques.

4.1.1 Astuce

Si vous voulez tester différentes palettes de couleurs, tout en conservant les retouches effectuées, copiez les lignes correspondantes, lancez Mocodo avec une nouvelle palette, puis recollez les lignes au même endroit.

4.2 Retouches manuelles d'une sortie au format SVG

Un fichier en SVG peut être visualisé dans tout navigateur respectueux des standards du web : Firefox, Chrome, Safari, Opera, Internet Explorer 9 ou 10, etc.
Pour aller au-delà de la simple visualisation, il faudra faire appel à un logiciel de dessin vectoriel dédié, comme Inkscape (libre) ou Illustrator, Freehand, CorelDRAW®, etc. Les éléments du fichier SVG produit pourront alors être repositionnés à la souris. Certains sont associés, pour permettre leur déplacement en bloc. Dans la version actuelle, les liens ne suivent pas ces déplacements, ce qui oblige à des manipulations supplémentaires. De façon générale, les retouches courantes sont plus faciles à réaliser dans le script intermédiaire généré par Mocodo.

4.3 Exportation

L'exportation est déléguée à votre éditeur SVG ou à Nodebox. Le format PDF assure la meilleure qualité, aussi bien pour la visualisation sur écran, que pour la projection ou l'impression. Si vous devez absolument utiliser un format bitmap, préférez PNG ou GIF.

5 Passage aux modèles logique et physique

À partir de la version 1.1, en plus d'une vue du schéma entité-association (MCD), Mocodo génère une ou plusieurs descriptions du schéma relationnel (MLD). La version 1.4 étend ce formalisme à la génération d'un modèle physique de données (en MySQL).

5.1 Cas réguliers

5.1.1 Algorithme

La séquence d'opérations suivante est réalisée :

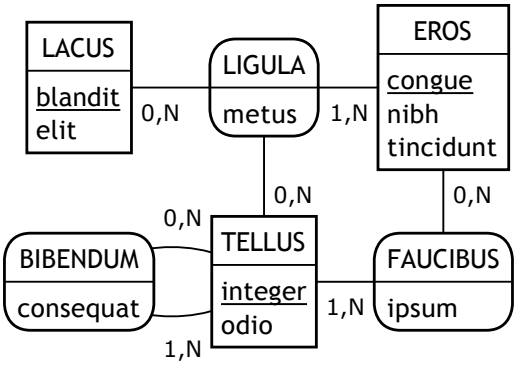
1. Pour chaque entité, une table de même nom et de mêmes attributs est créée. Le ou les identifiants de l'entité constituent la clef primaire de la table.
2. Toute table issue d'une entité faible est renforcée, c'est-à-dire que la clef primaire de l'entité qu'elle détermine fonctionnellement vient s'adjoindre à sa clef primaire.
3. Les associations sont traitées ainsi :
   – si toutes les pattes de l'association portent la cardinalité maximale N, une table de même nom et de mêmes attributs est créée. Sa clef primaire est constituée de l'ensemble des clefs primaires des tables issues des entités mises en jeu ;

- dans le cas contraire, c'est-à-dire si l'une des pattes de l'association porte la cardinalité (1,1), ou à défaut (0,1), l'entité distinguée se voit adjoindre :
  - en tant que clefs étrangères, l'ensemble des clefs primaires des autres entités mises en jeu ;
  - en tant qu'attributs, l'ensemble des attributs de l'association.

4. Les noms des attributs migrants sont éventuellement décorés du nom de leur entité d'origine et/ou de l'association de transit.
5. Les attributs homonymes pouvant subsister à l'intérieur de chaque table sont différenciés par un suffixe numérique.
6. L'ensemble de tables résultant est mis en forme en fonction du format de sortie souhaité (html, LaTeX, txt2tags, MySQL, etc.).

5.1.2 Exemple du traitement des associations non DF

Le premier cas de l'opération 3 est illustré sur un MCD comportant des associations triple, double et réflexive dont toutes les cardinalités maximales sont à N.
Cf. diagramme ci-dessous.

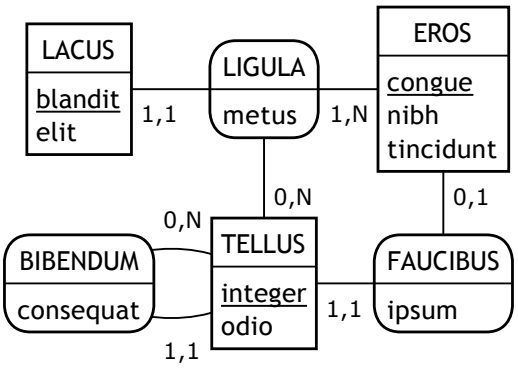


- **LACUS** (blandit, elit)
- **TELLUS** (integer, odio)
- **BIBENDUM** (#integer.1, #integer.2, consequat)
- **FAUCIBUS** (#congue, #integer, ipsum)
- **LIGULA** (#blandit, #congue, #integer, metus)
- **EROS** (congue, nibh, tincidunt)

Notez la numérotation automatique des deux attributs homonymes de BIBENDUM.

5.1.3 Exemple du traitement des associations non DF

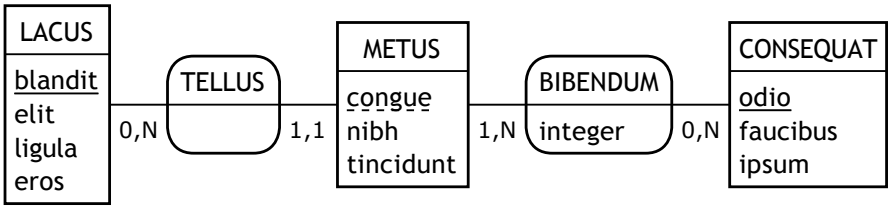
On illustrera le deuxième cas de l'opération 3 par un MCD quasiment identique, à ceci près que certaines cardinalités maximales ont été ramenées à 1. Toutes les associations vont alors disparaître.
Cf. diagramme page suivante.



– **EROS** (congue, nibh, tincidunt)
– **LACUS** (blandit, elit, #congue, #integer, metus)
– **TELLUS** (integer.1, odio, #integer.2, consequat, #congue, ipsum)
Notez la priorité de (1,1) sur (0,1) lors du traitement de l'association FAUCIBUS.

5.1.4 Exemple de traitement des entités faibles

Cf. diagramme ci-dessous.



– **BIBENDUM** (#blandit, #congue, #odio, integer)
– **CONSEQUAT** (odio, faucibus, ipsum)
– **LACUS** (blandit, elit, ligula, eros)
– **METUS** (#blandit, congue, nibh, tincidunt)
Notez le renforcement de la clef primaire des tables METUS et BIBENDUM. Tout se passe comme si l'entité METUS possédait à l'origine un identifiant double.

5.2 Cas irréguliers

Les règles de gestion peuvent parfois conduire à remettre en cause l'application mécanique de l'algorithme expliqué dans la sous-section précédente. Mocodo est capable de traiter certaines exceptions, pourvu qu'elles lui soient indiquées.

5.2.1 Réduire une clef primaire

Il arrive qu'un sous-ensemble strict de l'ensemble des identifiants des entités mises en jeu dans une association dont toutes les pattes portent la cardinalité N, suffise à constituer la

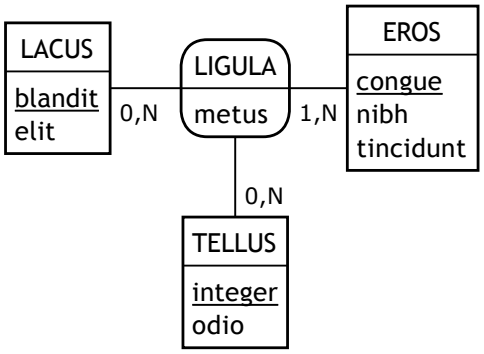
clef primaire de la table issue de cette association : cela se traduit par un dessoulignement des identifiants n'appartenant pas à ce sous-ensemble. Pour obtenir le même résultat avec Mocodo, il suffit de préfixer les entités concernées par une barre oblique.

Fichier d'entrée.

```
LACUS: blandit, elit
LIGULA, 0N LACUS, 1N /EROS, 0N TELLUS: metus
EROS: congue, nibh, tincidunt

TELLUS: integer, odio
```

Sortie brute. *Cf. diagramme ci-dessous.*



– **EROS** (congue, nibh, tincidunt)
– **LACUS** (blandit, elit)
– **LIGULA** (#blandit, #congue, #integer, metus)
– **TELLUS** (integer, odio)

Notez que la barre oblique de EROS n'apparaît pas dans le MCD, mais qu'elle conduit à la réduction de la clef primaire de LIGULA aux seuls identifiants des deux autres entités mises en jeu.

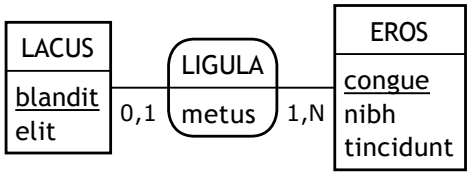
5.2.2 Éviter les champs vides

En l'absence d'une cardinalité (1,1), Mocodo traite par défaut la cardinalité (0,1) comme une dépendance fonctionnelle. Or, lorsque le 0 est grand devant le 1 en termes de fréquence d'apparition, la plupart des cellules de la clef étrangère ainsi constituée restent vides. On forcera la conversion de l'association en table en préfixant d'une barre oblique l'une au moins des entités non distinguées par le (0,1).

Fichier d'entrée.

```
LACUS: blandit, elit
LIGULA, 01 LACUS, 1N /EROS: metus
EROS: congue, nibh, tincidunt
```

Sortie brute. *Cf. diagramme ci-dessous.*



– **LACUS** (blandit, elit)
– **LIGULA** (#blandit, #congue, metus)
– **EROS** (congue, nibh, tincidunt)

5.3 Mise en forme

Le passage au relationnel se fait en deux étapes :

1. la création d'une représentation **interne** complète du MLD ;
2. la traduction de celle-ci en une représentation **externe** dans le ou les formats de sortie souhaités.

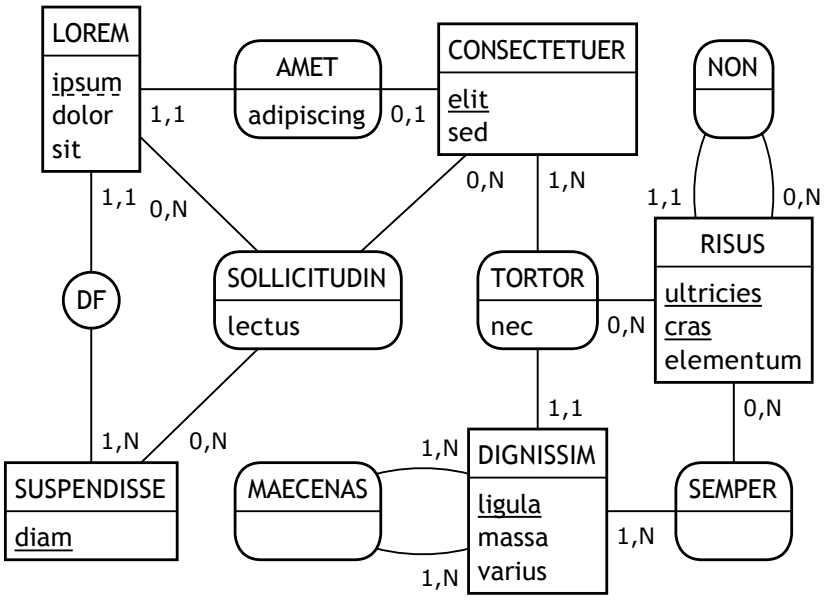
Chaque format de sortie est défini dans un fichier JSON placé dans le répertoire `tables`. La distribution inclut des fichiers prêts à l'emploi pour les formats suivants :
– **Texte brut**. À ouvrir directement avec un éditeur de texte Unicode.
– **Html.** À ouvrir directement avec un navigateur internet ou un programme de traitement de texte (dont Microsoft Word, OpenOffice, Apple Pages, etc.)
– **Txt2tags.** À compiler avec le générateur de documents Txt2tags pour une sortie dans de nombreux formats : HTML, XHTML, SGML, LaTeX, Lout, Man page, Wikipedia, Google Code Wiki, DokuWiki, MoinMoin, MagicPoint, PageMaker, texte brut.
– **LaTeX**. À compiler sous LaTeX pour une sortie de haute qualité aux formats PDF ou PostScript.
– **MySQL**. À importer avec MySQL.

5.3.1 Exemples

Cf. diagramme page ci-contre.

Sortie en texte brut.

```
SOLLICITUDIN (_#diam.1_, _#elit_, _#diam.2_, _#ipsum_, lectus)
SEMPER (_#ultricies_, _#cras_, _#ligula_)
RISUS (_ultricies.1_, _cras.1_, elementum, #ultricies.2, #cras.2)
MAECENAS (_#ligula.1_, _#ligula.2_)
DIGNISSIM (_ligula_, massa, varius, #ultricies, #cras, #elit, nec)
CONSECTETUER (_elit_, sed)
--- SUSPENDISSE (_diam_)
LOREM (_#diam_, _ipsum_, dolor, sit, #elit, adipiscing)
```



Sortie en html.

```
<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN'>
<HTML>
<HEAD>
<META HTTP-EQUIV='Content-Type' CONTENT='text/html; charset=utf8'>
</HEAD><BODY BGCOLOR='white' TEXT='black'>
<FONT SIZE='4'>
</FONT></CENTER>
<B>SOLLICITUDIN</B> (<U><I>diam.1</I></U>, <U><I>elit</I></U>, <U><I>diam.2</I></U>,
 <U><I>ipsum</I></U>, lectus)<BR>
<B>SEMPER</B> (<U><I>ultricies</I></U>, <U><I>cras</I></U>, <U><I>ligula</I></U>)<BR>
<B>RISUS</B> (<U>ultricies.1</U>, <U>cras.1</U>, elementum, <I>ultricies.2</I>,
 <I>cras.2</I>)<BR>
<B>MAECENAS</B> (<U><I>ligula.1</I></U>, <U><I>ligula.2</I></U>)<BR>
<B>DIGNISSIM</B> (<U>ligula</U>, massa, varius, <I>ultricies</I>, <I>cras</I>,
 <I>elit</I>, nec)<BR>
<B>CONSECTETUER</B> (<U>elit</U>, sed)<BR>
<!-- <B>SUSPENDISSE</B> (<U>diam</U>)<BR> -->
<B>LOREM</B> (<U><I>diam</I></U>, <U>ipsum</U>, dolor, sit, <I>elit</I>, adipiscing)<BR>
</BODY></HTML>
```

Sortie en txt2tags.

```
%!encoding: utf8
- **SOLLICITUDIN** (__#diam.1__, __#elit__, __#diam.2__, __#ipsum__, lectus)
- **SEMPER** (__#ultricies__, __#cras__, __#ligula__)
- **RISUS** (__ultricies.1__, __cras.1__, elementum, #ultricies.2, #cras.2)
- **MAECENAS** (__#ligula.1__, __#ligula.2__)
- **DIGNISSIM** (__ligula__, massa, varius, #ultricies, #cras, #elit, nec)
```

```
- **CONSECTETUER** (__elit__, sed)
% - **SUSPENDISSE** (__diam__)
- **LOREM** (__#diam__, __ipsum__, dolor, sit, #elit, adipiscing)
```

Le résultat de la compilation de ce texte source apparaît ci-dessous, transformé en LaTeX ou en Google Code Wiki selon que vous lisez cette documentation (elle-même écrite en txt2tags) en PDF ou sur le site du programme :

- **SOLLICITUDIN** (#diam.1, #elit, #diam.2, #ipsum, lectus)
- **SEMPER** (#ultricies, #cras, #ligula)
- **RISUS** (ultricies.1, cras.1, elementum, #ultricies.2, #cras.2)
- **MAECENAS** (#ligula.1, #ligula.2)
- **DIGNISSIM** (ligula, massa, varius, #ultricies, #cras, #elit, nec)
- **CONSECTETUER** (elit, sed)
- **LOREM** (#diam, ipsum, dolor, sit, #elit, adipiscing)

Sortie en LaTeX. Pour une mise en forme LaTeX plus souple, on choisira la sortie directe :

```
\begin{mld}
  \relat{Sollicitudin} & (\prim{\foreign{diam.1}}, \prim{\foreign{elit}},
 \prim{\foreign{diam.2}}, \prim{\foreign{ipsum}}, \attr{lectus})\\
  \relat{Semper} & (\prim{\foreign{ultricies}}, \prim{\foreign{cras}},
 \prim{\foreign{ligula}})\\
  \relat{Risus} & (\prim{ultricies.1}, \prim{cras.1}, \attr{elementum},
 \foreign{ultricies.2}, \foreign{cras.2})\\
  \relat{Maecenas} & (\prim{\foreign{ligula.1}}, \prim{\foreign{ligula.2}})\\
  \relat{Dignissim} & (\prim{ligula}, \attr{massa}, \attr{varius},
 \foreign{ultricies}, \foreign{cras}, \foreign{elit}, \attr{nec})\\
  \relat{Consectetuer} & (\prim{elit}, \attr{sed})\\
%   \relat{Suspendisse} & (\prim{diam})\\
  \relat{Lorem} & (\prim{\foreign{diam}}, \prim{ipsum}, \attr{dolor},
 \attr{sit}, \foreign{elit}, \attr{adipiscing})\\
\end{mld}
```

Les différentes commandes devront bien sûr être définies dans le préambule de votre document. À titre d'exemple, voici celles que j'utilise dans mes propres supports :

```
\usepackage[normalem]{ulem}

\newenvironment{mld}
{\par\begin{minipage}{\linewidth}\begin{tabular}{rp{0.7\linewidth}}}
{\end{tabularx}\end{minipage}\par}
\newcommand{\relat}[1]{\textsc{#1}}
\newcommand{\attr}[1]{\emph{#1}}
\newcommand{\prim}[1]{\uline{#1}}
\newcommand{\foreign}[1]{\#\textsl{#1}}
```

Cf. diagramme page suivante.

Sollicitudin	(*#diam.1*, *#elit*, *#diam.2*, *#ipsum*, *lectus*)
Semper	(*#ultricies*, *#cras*, *#ligula*)
Risus	(ultricies.1, cras.1, *elementum*, *#ultricies.2*, *#cras.2*)
Maecenas	(*#ligula.1*, *#ligula.2*)
Dignissim	(ligula, *massa*, *varius*, *#ultricies*, *#cras*, *#elit*, *nec*)
Consectetuer	(elit, *sed*)
Lorem	(*#diam*, ipsum, *dolor*, *sit*, *#elit*, *adipiscing*)

Sortie en MySQL. Pour qu'elle soit exploitable directement, il convient d'ajouter un type entre crochets après chaque attribut. Voir la section *Syntaxe de description du MCD*, sous-section *Besoin plus avancés* pour un exemple complet.

5.3.2 Configuration

Vous pouvez configurer selon vos propres besoins la sortie tabulaire de Mocodo, soit en modifiant les fichiers de configuration existants dans le répertoire `tables`, soit en en créant de nouveaux (conseillé). Les paramètres sont les suivants :

`addForeignKey` **(facultatif).** Au niveau physique, la prise en compte de clefs étrangères se fait *a posteriori* par altération des tables créées précédemment. Vous devez combiner `%(table)s` (la table à altérer), `%(foreignKey)s` (la clef étrangère) et `%(foreignTable)s` (la table de référence).

`attributeWithType` **(facultatif).** La manière de combiner un nom d'attribut avec son type lorsque celui-ci est donné dans le MCD d'entrée (entre crochets). La plupart des formats de sortie n'en tiennent pas compte : `"%(attribute)s"`, mais si l'on souhaite inclure cette information du modèle physique, on écrira par exemple pour MySQL : `"%(attribute)s %(attributeType)s"`.

`closing`**.** Le texte à composer à la fin, typiquement des balises fermantes.

`columnSep`**.** Une chaîne à insérer entre deux noms de colonnes consécutifs, typiquement une virgule, un retour-chariot ou une tabulation.

`comment`**.** Si vous souhaitez que les tables réduites à un unique attribut soit commentées dans le résultat, et donc disparaissent à la compilation, précisez la syntaxe de commentaire du langage cible. Dans le cas contraire, mettez juste `" %s"` pour composer la table telle quelle.

`df`**.** Pour limiter l'appauvrissement sémantique dû à la suppression des dépendances fonctionnelles, il est possible de préciser l'origine d'une clef étrangère en faisant référence, non seulement à son nom (`%(attribute)s`), mais aussi à celui de l'entité dont elle issue (`%(entity)s`) et/ou de l'association par laquelle elle a transité (`%(association)s`). On écrira par exemple : `"%(attribute)s de %(entity)s par %(association)s"`.

`distinguish`**.** Les attributs résultants homonymes peuvent être numérotés automatiquement, par exemple en texte brut : `"%(label)s.%(count)s"`. Si l'on ne veut pas de cette différenciation, on écrira simplement : `"%(label)s"`.

`extension`. L'extension du nom de fichier de sortie, sa base étant identique à celle du fichier de sortie graphique spécifié avec l'argument `output` du programme (par défaut, `sandbox`).

`foreignPrimary`. La mise en forme à appliquer aux clefs simultanément primaires et étrangères, typiquement un soulignement et un dièse en préfixe. Mêmes possibilités que le paramètre `table`.

`foreign`. La mise en forme à appliquer aux clefs étrangères, typiquement un dièse en préfixe. Mêmes possibilités que le paramètre `table`.

`lineSep`. Une chaîne à insérer éventuellement entre deux `line`s consécutives.

`line`. La manière de composer une table complète, typiquement le nom de la table (`%(table)s`) suivi du nom des colonnes (`%(columns)s`). Comme son nom ne l'indique pas, il est tout à fait possible que le résultat de la composition occupe plusieurs lignes.

`nonDf`. Les clefs étrangères des autres types d'association pourront de la même manière se voir qualifier par le nom de leur entité d'origine : `"%(attribute)s de %(entity)s"`. Le paramètre `%(association)s` est disponible, mais inutile ici.

`opening`. Le texte à composer au tout début, typiquement un préambule et des balises ouvrantes.

`primary`. La mise en forme à appliquer aux attributs faisant partie de la clef primaire, typiquement un soulignement. Mêmes possibilités que le paramètre `table`.

`primarySep`. Le séparateur de la liste d'attributs constituant la clef primaire, disponible dans le paramètre `%(primaryList)s` et utile pour la sortie en SQL.

`replace`. Une liste de couples de motifs de recherche et de remplacement. Par exemple, en LaTeX, `[["_","\\_"],["\$","\\$"]]` permettra d'échapper tout les symboles de soulignement et tous les dollars. Notez que les motifs de recherche et de remplacement suivent la syntaxe des expressions régulières. Si vous rencontrez des erreurs en utilisant ce paramètre, reportez-vous au chapitre correspondant de la documentation de Python.

`simple`. La mise en forme à appliquer aux attributs normaux, typiquement rien du tout. Mêmes possibilités que le paramètre `table`.

`strengthen`. La conversion en relationnel d'une entité faible étant assez délicate, on pourra de la même manière vouloir l'expliciter : `"renforcement par %(attribute)s de %(entity)s transitant par %(association)s"`.

`table`. La mise en forme à appliquer au nom de chaque table. Pour html, par exemple, `"<B>%s</B>"` signifie que le nom (représenté par `%s`) doit apparaître en gras. Un changement de casse peut être appliqué en remplaçant `%s` par `%(lower)s` (minuscules), `%(upper)s` (capitales), `%(capitalize)s` (capitale initiale, minuscules pour le reste) ou `%(title)s` (capitale à l'initiale de chaque mot, minuscules pour le reste).

6 Un mot personnel pour conclure

En l'état, Mocodo pour Nodebox suffit à mes modestes besoins d'enseignant de Bases de données. Sa façade SVG est un ajout destiné à le rendre multiplateforme. Son placement sous GPL devrait en garantir la pérennité, même si la poursuite de son développement ne fait pas partie de mes priorités actuelles.

Mes quatre étudiants de projet de synthèse, Abou-Sophiane Belhadj, Benjamin Gannat, Dorian Mahut et Cédric Poinsaint, ont travaillé au printemps 2009 sur une version préliminaire du code. Michel Grandmougin a essuyé les plâtres de la version bêta. Zsuzsanna Róka m'a fait profiter à maintes reprises de son expérience en bases de données. Bryan Oakley m'a indiqué comment résoudre un problème Tkinter particulièrement retors. Qu'ils en soient chaleureusement remerciés, ainsi que tous les collègues, étudiants ou amis qui ont suivi ou accompagné le développement.