latexbd

Etienne Marache

27 décembre 2010 – v 0.13

Résumé

Ce programme nécessite python+SQLalchemy et en outre un serveur de bases de données avec un module python pour la communication avec les bases.

Il permet de réaliser des requêtes et d'incorporer des données dans un fichier latex. Les fichiers source fonctionnant avec latexdb de Hans-Georg Heßer sont utilisables avec un minimum de changements.

Table des matières

1 Introduction

LaTeXBD est capable de
- vous connecter à une base de données,
- lancer des instruction SQL (création de table, introduction-suppression de colonnes ou de lignes etc...),
- lancer des requêtes pour récupérer les données que vous souhaitez incorporer à votre document,
- introduire ces données dans un tableau ou toute autre présentation,
- lancer du code python, par exemple pour utiliser SQLAlchemy.

Le principe est que tout se fait depuis un fichier source LaTeX dans lequel on rajoute les instructions nécessaires.

Depuis la version 0.4, LaTeXBD utilise SQLalchemy.

La version 0.9 apporte deux nouveautés, sous forme de deux environnements. Le premier `lbdpour` permet d'itérer un bloc et le second `lbdpython` permet d'écrire du code python et en particulier des instructions de SQLAlchemy.

La version 0.13 utilise le codage `utf-8` et est compatible avec la version 0.6 de SQLAlchemy.

Toutes les ressources du programme sont illustrées par 4 exemples. Les bases de données utilisées sont fournies dans le fichier exemples. Pour récupérer la base sous PostgreSQL :

1. créer un utilisateur (`createuser exemples`)
2. créer la base (`createdb -U exemples -E UTF8 exemples`)
3. psql exemples < exemples

2 Mode d'emploi

2.1 Installation

Pour utiliser LaTeXBD, vous devez au préalable

1. pouvoir vous connecter à un serveur de bases de données, par exemple PostgreSQL `http://www.postgresql.org/` ou le site francophone `http://www.postgresqlfr.org/`,
2. posséder un module de communication python avec le système de bases de données, par exemple psycopg2 pour PostgreSQL `http://www.initd.org/tracker/psycopg/wiki/PsycopgTwo`,
3. avoir installé SQLalchemy, un programme python qui facilite la communication avec les bases. La version 0.4 est indispensable à partir de LaTeXBD0.11. LaTeXBD0.13 est compatible avec la version 0.6. `http://www.sqlalchemy.org/`.

Ces programmes font partie, je pense, de toutes les distributions unix, comme c'est le cas pour gentoo, ubuntu et macport. SQLAlchemy et psycopg peuvent être installés à l'aide de `easy_install`.

```
easy_install psycopg2
easy_install SQLAlchemy == 0.5.8
```

LaTeXBD a été testé sous linux pour postgreSQL (+psycopg2) et MySQL. Voir la section **??** pour les systèmes possibles.

Il vous suffit alors d'installer les fichiers `latexbd`, et `latexbd.py` ou `lbda.py` à un endroit convenable (répertoire figurant dans la variable PATH) et de les rendre exécutables (chmod +x).

Si vous travaillez avec postgres, vous pouvez utiliser la base de données.

2.2 Utilisation minimale

`latexbd [-options] <fichier>`

où `fichier` est le nom du fichier à traiter. Il n'est pas conseillé d'écrire le suffixe `.tex`. Les options `l` et `p` lancent dans la foulée la compilation du fichier `<fichier>-but.tex`, le résultat étant disponible respectivement en `<fichier>.dvi` et `<fichier>.pdf`. Les options `d` et `f` lancent l'affichage avec xdvi et xpdf.

LaTeXBD reprend en les modifiant les 3 commandes de latexdb :

– `\texdbconnection`, où l'on donne les paramètres de connexion,
– `\texdbdef`, où l'on formule une requête,
– `\texdbfor`, qui récupère et exploite les résultats d'une requête.

Pour la syntaxe exacte de ces instruction, voir la section **??**. L'exemple *amis*, donne une utilisation minimale de LaTeXBD.

Premier exemple : *amis*

Commençons par un exemple très simple, une simple requête et un tableau de résultats.

Supposons que vous ayez une base de donnée *adresses* contenant une table *amis* (id,nom,prenom,tel) et que vous vouliez construire le tableau **??**.

Vous allez créer le fichier `amis.tex` suivant :

Nom	Prénom	Téléphone
Ami	Alfred	0000000001
Amie	Julie	9999999998
Zamis	Leo,Lea	5555555555

Table 1 – Exemple *amis*

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage{array}

% Les paramètres de connexion:
%\texdbconnection{postgres,localhost,<moi>,<mot de passe>,exemples-
  lbd}

% La requête:
%\texdbdef{##q}{SELECT %
%   nom,
%   prenom,
%   tel%
% FROM amis ORDER by nom,prenom}

\begin{document}
\begin{tabular}{|>{\scshape}l|l|l|}
\hline
Nom&Prénom&Téléphone\\
\hline
% On rentre ici les données.
\texdbfor{##q}{row.nom&row.prenom&row.tel\\\hline}
\end{tabular}
\end{document}
```

Quelques remarques :

– les instructions spéciales peuvent être ou non commentées en début et en fin de ligne.

– On peut placer tout code latex valide à l'intérieur de la commande lbd `\latexdbfor`, par exemple un formatage des numéros de téléphone.

Ensuite, vous lancez en ligne de commande ou avec votre éditeur favori

`latexbd amis`

ou

`latexbd -l amis`

ou encore

`latexbd -ld amis`

Un fichier amis-but.tex est créé contenant les données et prêt à être compilé. Cette compilation peut-être lancée automatiquement avec les options `-l` et `-p` qui créent directement respectivement un fichier `amis.dvi` et `amis.pdf`. Pour avoir l'affichage en plus : options `-ld` ou `-pf` (dvi ou pdf).

2.3 Introduction d'instructions SQL

Remarque importante : on évitera d'avoir les premières lignes identiques pour deux instructions texinstruction. Si cela se produit, latexbd reprend la première instruction. Ce problème sera réglé dans une version future.

Deuxième exemple : *notes*

Création d'un bilan trimestriel. Le professeur réunit dans un tableau les notes des trois devoirs, la moyenne de chaque élève et son rang. Sur une dernière ligne figurent les moyennes générales. Dans cet exemple, on va tout faire depuis le fichier latex :

- Création de la table notes, contenant les colonnes id,nom,dev1,dev2,dev3,moy,rang.
- Remplissage des notes depuis le fichier annexe releve-notes.
- Création du tableau de notes.

On va utiliser une quatrième instruction : `\texdbinstruction`, qui permet de transmettre n'importe quelle instruction SQL à la base de donnée.

Sa syntaxe est

```
\texdbinstruction{<instruction SQL>}
```

```
\documentclass{article}
% Les paramètres de connexion
\texdbconnection{postgres,localhost,<nom>,<mot de passe>,exemples-
  lbd}

% Création de la table notes, si elle existe déjà, on ajoute # ou
  un
% autre symbole devant la première ligne et on commente si ce n'est
% déjà fait.
\texdbinstruction{
  CREATE TABLE notes (
    id  integer primary key,
    nom varchar(30),
    dev1 numeric(3,1),
    dev2 numeric(3,1),
    dev3 numeric(3,1),
    moy numeric(4,2),
    rang integer)}

% On remplit la table, calcule la moyenne et le rang.
\texdbinstruction{BEGIN;
  COPY notes(id,nom,dev1,dev2,dev3)
     FROM <chemin>/releve-notes' DELIMITERS '!';
  UPDATE notes SET moy =  (dev1+dev2+dev3)/3;
  UPDATE notes SET rang =
     (SELECT count(w2.moy)+1 FROM notes w2 WHERE w2.moy>notes.moy);
  COMMIT}

% Requête
\texdbdef{##q}{SELECT %
   nom,
   dev1,
   dev2,
```

```
   dev3,
   moy,
   rang
FROM notes ORDER by rang}

% Requête pour les moyennes de classe
\texdbdef{##m}{SELECT
  round(avg(dev1),1) as moydev1,
  round(avg(dev2),1) as moydev2,
  round(avg(dev3),1) as moydev3,
  round(avg(moy),1) as moyg
FROM notes}

\begin{document}
\begin{tabular}{|c|l|c|*{3}{c|}}
\hline
 Rang&Nom&moy&Dev1&Dev2&Dev3\\
\hline
% LES NOTES
\texdbfor{##q}{row.rang&row.nom&row.moy&row.dev1&row.dev2&row.dev
  3\\}
\hline
% LES MOYENNES
\texdbfor{##m}{&&row.moyg&row.moydev1&row.moydev2&row.moydev3\\}
\hline
\end{tabular}
\end{document}
```

2.4 Familles de requêtes

Les commandes lbd `\latexdbdef` et `\latexdbfor` possèdent un argument optionnel qui permet de définir une famille de requêtes au lieu d'une seule.

Troisième exemple : *temperatures*

Vous relevez consciencieusement tous les jours la température à la même heure et vous voulez avoir pour chaque mois le minimum, le maximum et la moyenne. Les résultats sont présentés sous forme de 12 tableaux, un par mois.

Vos relevés sont dans le fichier releves.txt et ici vous vous contentez d'ajouter les nouveaux relevés dans la table temperatures de la base, comportant 2 colonnes date et temp.

Le fichier temperatures.tex sera le suivant :

```
\documentclass[12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage{numprint}
\newcommand*{\mois}[1]{
 \ifcase#1
    \or janvier\or f\'evrier\or mars\or avril\or mai\or juin\or
    juillet\or ao\^ut\or septembre\or octobre\or novembre\or
    d\'ecembre\fi}

% Les paramètres de connexion (locale):
```

```
\texdbconnection{postgres,localhost,<nom>,<mot de passe>,exemples-
  lbd}

% Création de la table
\texdbinstruction{
  CREATE TABLE temperatures (
    date date UNIQUE,
    temp numeric(3,1))}

% On complète la table:
\texdbinstruction{COPY temperatures FROM
  '<chemin>/rel-temp'}

% On crée la famille de requêtes ##q1, ... ,##q12,
% ##1 est remplacé par les entiers de 1 à 12.
\texdbdef{##q}[range(1,13,1)]{SELECT %
  date_part('month'::text,date_trunc('month'::text, date::timestamp
  with time zone)) AS mois,
  date_part('year'::text,date_trunc('month'::text, date::timestamp
  with time zone)) AS an,
   min(mini) as mini,
   max(mini) as maxi,
   round(avg(mini),1) as moy
  FROM mini
  WHERE  date_part('month'::text,date_trunc('month'::text, date::
    timestamp
  with time zone)) = ##1
  GROUP BY date_trunc('month'::text,date::timestamp with time zone)
  ORDER BY an}

\begin{document}
% ##q et ##1 sont remplacés par ##i et i, où i parcourt les entiers
   de
% 1 à 12. ## texdbfor est remplacé
\texdbfor{##q}[range(1,13,1)]{
\numprint{row.an}&row.mini&row.maxi&row.moy\\}%
\begin{center}
{\large \textbf{\mois{##1}}}

\smallskip
\begin{tabular}{|>{\nprounddigits{0}}c||r|r|r|}
\hline
Année&Minimum&Maximum&Moyenne\\
\hline
% Introduction des cellules de ##qi
##texdbfor%
\hline
\end{tabular}
\end{center}
\endtexdbfor
% fin de la boucle
\end{document}
```

On a ajouté à la commande lbd `\texdbdef` l'argument optionnel (écrit entre cro-

chet) `range(1,13,1)`. Ceci est une commande python qui crée la liste [1,2,3,4,5,6,7,8,9,10,11,12]. L'argument optionel est une liste ou un tuple (éléments entre crochets ou parenthèses séparés pas des virgules). Cette liste (ce tuple) peut être donnée en extension ou créée par un code python. Pour chaque valeur `var` de l'indices on remplace `##q` par `##q<var>` et `##1` par `<var>`.. Dans notre exemple, il y a un `##1` à remplacer à la ligne 26 par le numéro du mois.

Le principe est le même pour `\texdbfor` sauf que en plus, `##texdbfor` sera remplacé par un code LaTeX où les cellules auront pris la valeur donnée par la requête `##qvar`. La zone de remplacement prend fin à la commande `\endtexdbfor`.

2.5 Itérerations

L'environnement `lbdpour` est une boucle rudimentaire, qui écrit le contenu de l'environnement pour toutes les valeurs de l'indice qui parcourt une liste ou un tuple (au sens de python).

Dans l'exemple suivant, on introduit plusieurs requêtes dans la boucle, ce que ne permet pas l'argument optionnel de `\latexdbfor`.

Quatrième exemple : relevés température (bis)

Vous souhaitez afficher dans un tableau, les relevés de températures avec les moyennes hebdomadaires pour une année donnée. Il vous faut une requête pour les relevés quotidiens et une autre pour les moyennes. On va donc faire 2 fois 52 requêtes, pas de problème avec l'argument optionnel et une boucle avec 52 passage pour les relevés de chaque semaine, ici l'argument optionnel de `\texdbfor` ne fonctionne pas car il y a 2 requête dans chaque passage, celle des relevés et celle des moyennes. C'est l'environnement `lbdpour` qui va fournir le boucle.

```
\documentclass[12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage{supertabular}
\texdbconnection{postgres,localhost,<login>,<mot de passe>,<base>}

% famille de requêtes
\texdbdef{##q}[range(1,53,1)]{SELECT %
  date,temp  FROM temperatures
  WHERE  EXTRACT(week from date) = ##1 AND
date_trunc('week',date)>'2005-12-31'
AND date_trunc('week',date)<'2007-01-01'
  ORDER BY date}

\texdbdef{##r}[range(1,53,1)]{SELECT %
  round(avg(temp),1) as moy FROM temperatures
  WHERE  EXTRACT(week from date) = ##1 AND
date_trunc('week',date)>'2005-12-31'
AND date_trunc('week',date)<'2007-01-01'
  GROUP BY date_trunc('week',date::timestamp)}

\begin{document}
\begin{supertabular}{cc}
\hline
Date&Température\\
```

```
\hline
\begin{lbdpour}{range(1,53,1)}
  \texdbfor{##q##1}{row.date&row.temp\\}
  \hline
  \texdbfor{##r##1}{semaine ##1&moyenne: row.moy\\}
  \hline
\end{lbdpour}
\end{supertabular}
\end{document}
```

2.6 Introduction de code python

Attention, l'utilisation de l'environnement `lbdpython` nécessite la connaissance de SQLAlchemy.

Le texte de l'environnement est un code source python, il doit être saisi comme tel et ne doit pas être commenté.

Je pense qu'il n'est pas souhaitable d'aller plus loin dans l'introduction de nouvelles fonctionnalité du moins avec la méthode envisagée jusqu'à présent. Si l'on veut un traitement plus sophistiqué des données, il est alors sans doute plus avantageux d'utiliser directement SQLAlchemy. C'est pour cela que j'ai créé l'environnement `lbdpython`. Il se contente d'envoyer des blocs de code à l'interpréteur python qui les traite avec `exec`. Dans l'exemple précédent, la boucle va être créée en python. Pour cela on utilise l'environnement `lbdpython` qui va envoyer des instructions SQLAlchemy à l'interpréteur `python`.

Si tout est fait à partir du seul environnement `lbdpython`, on peut utiliser l'option -a de latexbd pour lancer lbda.py, version simplifiée de latexbd.

Quatrième exemple avec lbdpython

```
\documentclass[12pt]{article}
\usepackage[latin1]{inputenc}

\usepackage{supertabular}

% \begin{lbdpython}%
db = create_engine('postgres://<login>@localhost/<base>')
metadata = BoundMetaData(db)
% \end{lbdpython}%

% Les requêtes,
\begin{lbdpython}
temp_table = Table('temperatures',metadata,autoload=True)
q = {}
m = {}
plage = range(1,53,1)
t = temp_table.alias('t')

for sem in plage:
   q[sem] = select([t.c.date,t.c.temp],
          and_(func.date_trunc('week',t.c.date) > '2005-12-31',func
            .date_trunc('week',t.c.date) <'2007-01-01',
```

```
            extract('week',t.c.date) == sem),order_by=[t.c.date])
    m[sem] = select([func.round(func.avg(t.c.temp),1).label('moy')],
            and_(func.date_trunc('week',t.c.date) > '2005-12-31',func
              .date_trunc('week',t.c.date) <'2007-01-01',
            extract('week',t.c.date) == sem),group_by=[func.date_
              trunc('week',t.c.date)])
\end{lbdpython}


\begin{document}
\begin{supertabular}{cc}
\hline
Date&Température\\
\hline

\begin{lbdpython}
for sem in plage:
      qq = q[sem].execute()
      for row in qq:
            chaine = r"%s&%s\\"%(row.date,row.temp)
            but.write(chaine)
      but.write(r"\hline ")
      mm = m[sem].execute()
      ligne = mm.fetchone()
      chaine = r"semaine %s&moyenne %s\\" %(sem,ligne.moy)
      but.write(chaine)
      but.write(r"\hline ")
\end{lbdpython}
\end{supertabular}
\end{document}
```

La commande `\texdbinstruction` qui permet d'exécuter une instruction SQL, utilise une fonction `traite`. Cette fonction peut rendre service dans l'environnement `lbdpython`.

Voici un exemple d'utilisation : vous disposez d'une table users, qui est remplie à l'aide d'un fichier de données `copy.dat`. Ici, on suppose que le fichier de données a été modifié. On vide la table, puis on la remplit à nouveau.

```
\documentclass{article}
\usepackage[latin1]{inputenc}

\begin{lbdpython}
db = create_engine('postgres://<moi>@localhost/test')
metadata = BoundMetaData(db)
users = Table('users', metadata,
   Column('user_id',Integer,primary_key=True),
   Column('user_name',String(40)),
   Column('password',String(80))
)
t = "BEGIN;\
    DELETE FROM users;\
    COPY users FROM\
    '<chemin>/copy.dat';\
    COMMIT"
traite(db,t)
```

```
s = select([users])
result= s.execute()
rows = result.fetchall()
\end{lbdpython}

\begin{document}
\begin{tabular}{ll}
\begin{lbdpython}
for row in rows:
   chaine = r"%s&%s\\"%(row.user_name,row.password)
   but.write(chaine)
\end{lbdpython}
\end{tabular}
\end{document}
```

3 Instructions lbd

Examinons maintenant la syntaxe précise des 4 instructions spéciales du programme appelées *commandes lbd*.

Les commandes lbd utilisent la syntaxe LaTeX bien qu'elles n'aient rien à voir avec ce programme, mais cela est hérité de latexdb et ma fois, c'est assez pratique à utiliser.

`\texdbconnexion` prend 1 argument, les paramètres de connexion.

`\texdbconnexion{dbType,dbHost,dbUser,password,base}`

- `dbType` est le nom du système : postgres, mysql, oracle, ...
- `dbHost` est le nom de l'hôte : `localhost` pour une connexion locale.
- `dbUser` est le nom de l'utilisateur
- `password` est son mot de passe
- `base` est le nom de la base de données utilisée.

`\texdbinstruction` prend un argument optionnel, et un argument obligatoire, la commande SQL à exécuter.

`\texdbinstruction[liste]{Commande SQL}`

Si l'argument optionnel est présent, on exécute une famille de commandes SQL indicée par les éléments de la liste.

Exemple *notes*

Dans l'exemple *notes*, supposons que les notes soient non plus dans un seul fichier mais dans 3 fichiers `devi` avec $i \in \{1, 2, 3\}$, chacun de la forme

```
1   Jean   10
2   Pierre   15
3   Jean-Pierre   18
```

On utilise une table annexe contenant 3 colonnes pour rentrer successivement les 3 séries de notes : Les lignes 19-25 devenant

```
% Création de la table annexe
\texdbinstruction{CREATE TABLE temp (id integer,nom varchar(30)
  , note numeric(3,1))}
% On remplit la table,
```

```
\texdbinstruction[(1,2,3)]{BEGIN;
  DELETE FROM temp;
  COPY temp FROM <chemin>/dev##1;
  UPDATE notes SET dev##1 = temp.note FROM temp WHERE
         notes.num=temp.num;
  COMMIT}
% on calcule de la moyenne et le rang.
\texdbinstruction{BEGIN;
  UPDATE notes SET moy =  (dev1+dev2+dev3)/3;
  UPDATE notes SET rang =
     (SELECT count(w2.moy)+1 FROM notes w2 WHERE w2.moy>notes.
       moy);
  COMMIT}
```

\texdbdef prend 2 arguments obligatoires, le nom de la requête et la requête elle même et un argument optionnel (voir section **??**)

`\texdbdef{##<nom>}[liste]{SELECT ...}`

\texdbfor prend 2 arguments obligatoires, le premier est le nom de la requête utilisée et le deuxième le code LaTeX contenant les noms des cellules qui seront remplacées.

```
\texdbfor{##<nom>}[liste]{<source à itérer>}
... code latex ........
\endtexdbfor
```

Le code latex et la commande lbd `\endlatexdbfor` ne sont présents qu'avec l'argument optionnel. Voir exemple *temperatures*.

Le symbole de commentaire % et les espaces peuvent être utilisés librement en début et en fin de ligne. Pour rendre inactive une instruction, il suffit de la faire précéder du symbole % suivi de n'importe quel autre caractère.

4 Utilisation de lbda.py

Ce programme est pour l'instant une simple restriction de latexbd.py à l'environnement `lbdpython`. Il s'adresse donc aux utilisateurs avertis qui sont capable de tout faire à partir de SQLAlchemy, le programme se contente alors d'envoyer les blocs de commandes pour exécution.

Le travail revient à l'utilisateur, l'utilisation de cette méthode est donc beaucoup plus contraignante que l'utilisation de latexbd.py. En contrepartie, les possibilités sont illimitées puisque l'on peut créer ses propres blocs en python.

Il sera utilisé avantageusement pour la deuxième méthode de l'exemple 4.

L'option -a de `latexbd` lance `lbda.py` à la place de `latexbd.py`.

5 Modifications pour autres systèmes

N'ayant pas la possibilité de tester, je ne sais pas avec quels système autre que PostgreSQL le programme fonctionne tel quel. Merci de me contacter si vous avez réussi à le faire fonctionner ou si vous souhaitez le faire.

Depuis la version 0.4, LaTeXBD utilise SQLalchemy, il suffit donc de définir un « engin »adapté à votre système. Voici les systèmes possibles et le nom (dbBase) à déclarer dans `\texdbconnexion` :

- Postgres : psycopg2 (postgres)
- SQLite : pysqlite (sqlite)
- MySQL : MySQLDB (mysql)
- Oracle : `cx_Oracle` (oracle)
- MS-SQL : adodbapi pymssql
- Firebird : kinterbasdb

Pour plus de détails voir `http://www.sqlalchemy.org/docs/dbengine.myt`

Une adaptation éventuelle se ferait à la ligne 149 de `latexdb.py`

```
chaine = '%s://%s@%s/%s'%(dbType,dbUser,dbHost,dbDB)
```

6 Historique

6.1 De latexdb à LaTeXBD

La principale force de latexdb est d'individualiser chaque cellule récupérée dans la base de donnée permettant ainsi une utilisation très fine des données. Ceci était repris dans latexbd-0.3, depuis la version 0.4 on utilise SQLalchemy qui permet de rappeler chaque cellule avec son nom (ou alias) de colonne.

Le programme latexdb est composé de 3 scripts :

- latexdb-preparse.py script python qui prépare le fichier initial pour le traitement par latexdb.py.
- latexdb.py script python qui récupère les données et les incorpore dans un nouveau fichier.
- latexdb script bash qui gouverne l'ensemble : lance les script python puis la compilation latex et restitue le fichier initial.

Je n'ai gardé qu'un seul programme python gérant le traitement des requêtes et l'incorporation des données. Seules les instructions non latex sont traitées, le reste étant laissé intact. LaTeXBD crée un seul fichier (fichier.tex est transformé en fichier-but.tex) où seule l'instruction `\texdbfor` est remplacée, les autres étant simplement commentées. Ensuite, on ne touche plus au fichier initial.

Ainsi il n'y a plus que 2 fichiers : le fichier initial et un nouveau prêt à être compilé. Les options `l` et `p` du script `latexdb` permettent de lancer la compilation si on le désire.

L'instruction `\texdbinstruction` a été ajoutée qui lance une commande SQL puis les environnements `lbdpour` et `lbdpython`.

Afin de pouvoir réutiliser les fichiers sources de latexbd, depuis la version 0.4 on reprend sa syntaxe à l'exeption des variables qui n'ont plus besoin d'être déclarées dans `\texdbdef` et qui sont utilisées sous la forme `row.<nom>` où `<nom>` est le nom ou un alias de la colonne. Le seul changement à effectuer consiste à remplacer dans les commandes `\texdbfor ##nom` par `row.nom`. Il faut également veiller à ce que toutes les colonnes aient un nom quitte à donner un alias à celles qui n'en ont pas (voir exemple 2).

6.2 Différences entre versions 0.3 et 0.4

Les différences (pour l'utilisateur) sont les suivantes :

- Retour aux accolades comme séparateurs.

– Il n'est plus nécessaire de faire suivre les commandes sur plusieurs lignes de `% FIN %`
– Les noms de systèmes de bases de données sont ceux de SQLalchemy
– La commande `\texdbdef` n'a plus que 2 arguments et les colonnes doivent avoir un nom.
– Les cellules sont appelées par `row.nom` où `nom` est le nom de colonne.

6.3 Différences entre versions 0.4 , 0.5 et 0.6

Les versions 0.5 et 0.6 ajoutent les arguments optionnels qui permettent d'utiliser des familles de requête et de les utiliser dans des boucles et de faire une famille de commandes SQL.

6.4 Différences entre versions 0.6 ,0.9 et 0.10

Création des deux environnements lbdpour et lbdpython. Le premier permet de recopier son contenu autant de fois que nécessaire seul le paramètre étant modifié. Le deuxième permet de transmettre directement à latexbd du code python. Ajout du fichier lbda.py qui n'utilise que l'environnement lbdpython, l'option -a de latexbd lance ce programme qui pour l'instant n'apporte rien de nouveau.

La version 0.10 est identique à la version 0.9 la seule différence est l'ajout de la fonction `traite` à `lbda.py`.

7 Conclusion

Le programme a l'air de bien fonctionner, les sources sont je l'espère compréhensibles, elles sont commentées au fur et à mesure.

Je ne suis absolument pas spécialiste de python et plus généralement de programmation, aussi n'hésitez pas à me transmettre remarques et critiques. `e.marache@wanadoo.fr`.