Mémo mSQL

version 0.9

Patrick Jermann

TECFA,
Faculté de Psychologie et des Sciences de l'Education,
Université de Genève

9 route de Drize, CH-1227 Carouge
Tél.: +41 22 70 9694

courrier électronique: Patrick Jermann@tecfa.unige.ch
MOO: tecfamoo.unige.ch 7777 (aka Colin)
WWW: http://tecfa.unige.ch/tecfa-people/jermann.html

1. Opérations de base

Mini-SQL est un **serveur** de bases de données, c'est à dire que plusieurs bases de données sont gérées par un seul et même programme. Toute machine UNIX peut héberger un serveur mSQL. Ainsi, à TECFA nous avons deux serveurs. Le permier réside sur *tecfasun1.unige.ch* et sert de base de données principale. Le second réside sur *tecfasun2.unige.ch* et sert de base de données tampon. Les nouvelles soumissions des utilisateurs y sont stockées en attendant l'approbation d'un administrateur pour figurer dans la base de données principale.

Une base de donnée est composée d'une ou de plusieurs tables qui contiennent des enregistrements. Ces enregistrements sont composés de champs.

1-1 Lister les bases de données, les tables et les champs

La commande relshow permet lister les bases de données qui résident sur une machine. Le paramètre -h permet de spécifier le nom de la machine qui nous intéresse. Dans l'exemple ci-dessous, l'utilisateur est connecté à la machine *tecfasun2* mais liste les bases de données sur la machine *tecfasun1*.

```
tecfasun2:~ 46) relshow -h tecfasun1

  +-----------------+
  |    Databases    |
  +-----------------+
  | pnr33           |
  | people          |
  | tecfa           |
  | csre-info       |
  | ctie            |
  | mediothek       |
  +-----------------+
```

Lorsqu'on ajoute en argument le nom d'une base de données, relshow affiche le nom des tables qui composent celle-ci. Dans l'exemple ci-dessous, la base de données *csre-info* contient une seule table nommée *info_ed*.

```
tecfasun2:~ 47) relshow -h tecfasun1 csre-info

Database = csre-info

  +---------------------+
  |       Table         |
  +---------------------+
  | info_ed             |
  +---------------------+
```

De même, en ajoutant le nom d'une table à la commande, relshow liste la définition des champs de la table. Chaque champ possède un nom (Field), un type (Type), une longueur (Length). En plus, un marqueur permet de spécifier si la valeur d'un champ peut être vide (Not Null). Finalement, il est possible de définir des index sur certains champs.

Dans l'exemple ci-dessous, nous voyons la liste des champs pour la table *info_ed*. Trois types de champs apparaissent. Les types *char* et *text* définissent des valeurs textuelles. Il n'est pas possible d'effectuer des recherches par mot-clé sur un champ de type *text*. Par contre, il est possible d'y stocker un nombre illimité de caractères.

Inversément, il est possible d'effectuer une recherche par mots-clé sur le contenu d'un champ *char*. Par contre, il n'est pas possible d'y stocker un nombre de caractères supérieur au nombre défini dans Lenght.

Ainsi, dans notre exemple, les recherches par mot-clé ne pourront porter que sur les champs suivants: `Nr`, `Titel`, `Titel_ubersetzt`, `Deskriptoren`, `Desctipteurs`, `von` et `bis`.

Le troisième type de champ est *index*. C'est un champ dont le contenu est géré par le système et qui permet d'accélérer la recherche sur certains champs.

```
tecfasun2:~ 48) relshow -h tecfasun1 csre-info info_ed

Database = csre-info
Table    = info_ed

 +-----------------+----------+--------+----------+--------------+
 |     Field       |   Type   | Length | Not Null | Unique Index |
 +-----------------+----------+--------+----------+--------------+
 | Nr              | char     | 25     | N        | N/A          |
 | Titel           | char     | 400    | N        | N/A          |
 | Titel_ubersetzt | char     | 400    | N        | N/A          |
 | Institution     | text     | 500    | N        | N/A          |
 | Forscher        | text     | 500    | N        | N/A          |
 | Abstract_dt     | text     | 700    | N        | N/A          |
 | Abstract_frz    | text     | 700    | N        | N/A          |
 | Deskriptoren    | char     | 400    | N        | N/A          |
 | Descripteurs    | char     | 400    | N        | N/A          |
 | Publikationen   | text     | 700    | N        | N/A          |
 | Meth            | text     | 300    | N        | N/A          |
 | Geo             | text     | 120    | N        | N/A          |
 | Typ             | text     | 200    | N        | N/A          |
 | Auftrag         | text     | 200    | N        | N/A          |
 | Finanzierung    | text     | 150    | N        | N/A          |
 | von             | char     | 10     | N        | N/A          |
 | bis             | char     | 10     | N        | N/A          |
 | search_idx      | index    | N/A    | N/A      | N            |
 +-----------------+----------+--------+----------+--------------+
```

1-2 L'interface en-ligne à mSQL

L'interface en ligne permet de soumettre des requêtes SQL au serveur qui nous renvoie alors le résultat de la requête. La commande msql sert à ouvrir une connexion vers le serveur mSQL. Les arguments de la commande sont le nom de la machine sur laquelle le serveur mSQL est installé et le nom de la base de données à laquelle on veut se connecter.

```
Usage : msql [-h host] database
```

Exemple:

```
tecfasun2:~ 66) msql -h tecfasun1 csre-info

Welcome to the miniSQL monitor.  Type \h for help.

mSQL >
```

On peut dès lors utiliser des commandes SQL pour effectuer des requêtes et utiliser les codes suivants:

\q ferme la connexion
\g envoie la requête
\p affiche la dernière commande

1-3 Exemple de requête SELECT avec l'interface en ligne

La syntaxe des commandes SQL implémentées dans mSQL se trouve en annexe (page 26). Une description complète est accessible à l'URL: http://tecfa.unige.ch/guides/msql/msql2/spec.htm.

Dans le premier exemple, nous sélectionnons les champs `Nr`, `von` et `bis` de la table `info_ed`. Sur la deuxième ligne, `\g` indique que la spécification de la requête est terminée. Nous n'avons pas spécifié de contraintes sur le type d'enregistrement que nous souhaitons afficher, c'est pourquoi tous les 806 enregistrements sont retournés par mSQL.

```
mSQL > select Nr,von,bis from info_ed
    -> \g

Query OK.  806 row(s) modified or retrieved.
 +----------------------------+-------------+------------+
 | Nr                         | von         | bis        |
 +----------------------------+-------------+------------+
 | 87:001                     | 1984        | NULL       |
 | 87:002                     | 1978        | NN         |
 | 87:003                     | 1983        | 1986       |
 | 87:004                     | 1979        | 1985       |
 | 87:005                     | 1986        | 1987       |

etc ... 806 lignes de résultat impossibles à lire

 | 96:076                     | 1988        | 1995       |
 | 96:077                     | 1994        | 1996       |
 | 96:078                     | 1996        | 1997       |
 | 96:079                     | 1993        | 1995       |
 | 96:080                     | 1990        | 1992       |
 +----------------------------+-------------+------------+
```

Si nous voulons connaître les numéros des recherches qui ont commencé en 1994 et se sont terminées en 1996, les mots-clé WHERE et AND permettent de spécifier cette contrainte.

```
mSQL > select Nr,von,bis from info_ed where von='1994' and bis='1996'
    -> \g

Query OK.  12 row(s) modified or retrieved.

 +----------------------------+-------------+------------+
 | Nr                         | von         | bis        |
 +----------------------------+-------------+------------+
 | 93:068                     | 1994        | 1996       |
 | 93:078                     | 1994        | 1996       |
 | 93:084                     | 1994        | 1996       |
 | 94:079                     | 1994        | 1996       |
 | 94:081                     | 1994        | 1996       |
 | 95:042                     | 1994        | 1996       |
 | 95:075                     | 1994        | 1996       |
 | 96:033                     | 1994        | 1996       |
 | 96:046                     | 1994        | 1996       |
 | 96:067                     | 1994        | 1996       |
 | 96:068                     | 1994        | 1996       |
 | 96:077                     | 1994        | 1996       |
 +----------------------------+-------------+------------+
```

Voici la requête pour connaître le titre et les descripteurs de la recherche numéro 93:068:

```
mSQL > select Titel, Descripteurs from info_ed where Nr='93:068'
    -> \g
```

1-4 Créer des tables

Mécanisme de redirection

Pour des opérations comme la création d'une nouvelle table, une requête SQL peut occuper plusieurs lignes. Il est malaisé de les saisir une à une à l'invite `mSQL>`. C'est pourquoi il est nécessaire d'utiliser un **fichier de commande**, qui consiste en une ou plusieurs requêtes SQL. Celui-ci est ensuite soumis en une fois au serveur en utilisant le mécanisme de **redirection**. Le signe **<** est utilisé pour 'donner à manger' à une commande. Dans l'exemple fictif qui suit, la commande `mouline` va prendre comme entrée le fichier de commande `mes_commandes`:

```
mouline < mes_commandes
```

Le mécanisme de redirection est également utilisé pour récupérer le résultat d'une commande dans un fichier. La redirection s'exprime avec le signe **>**. L'exemple fictif ci-dessous sauve le résultat de la commande `mouline` dans un fichier `resultats.txt`. Voir "Backup des bases de données" à la page 11 pour un exemple concret.

```
mouline > resultats.txt
```

On crée des tables en écrivant une requête CREATE TABLE dans un fichier de commande et en appelant msql avec ce fichier en argument. La base de données pour laquelle on veut créer une table doit exister! Si ce n'est pas le cas, voir "Créer une base de données" à la page 12.

1-4.1 Fichier de commande

Voici un exemple de fichier de commandes qui sert à créer une table. (http://agora.unige.ch/msql/table-exemple.sql). Un éditeur de texte minimal suffit pour composer un tel fichier (Notepad, Simple Text). Les lignes précédées de # sont des commentaires et ne produisent aucun effet.

```
# Host: tecfasun1
Database: test
#--------------------------------------------------------
# Structure pour la table 'info_ed
#

DROP TABLE info_ed \g
REATE TABLE info_ed (
  Nr CHAR(25),
  Titel CHAR(400),
  Titel_ubersetzt CHAR(400),
  Institution TEXT(500),
  Forscher TEXT(500),
  Abstract_dt TEXT(700),
  Abstract_frz TEXT(700),
  Deskriptoren CHAR(400),
  Descripteurs CHAR(400),
  Publikationen TEXT(700),
  Meth TEXT(300),
  Geo TEXT(120),
  Typ TEXT(200),
  Auftrag TEXT(200),
  Finanzierung TEXT(150),
  von CHAR(10),
  bis CHAR(10)
) \g\p
```

Jette la table au cas où elle exitait déjà

Crée la table 'info_ed'

Définition des champs de la table.

\g exécute la commande

\p imprime le feed-back

Les types de champ possibles sont décrits ci-dessous. Notez qu'il n'est pas impératif d'utiliser le type MONEY pour stocker des valeurs monétaires. Une chaîne de caractères CHAR ou un entier INT peuvent faire l'affaire. Néanmoins, la recherche de valeurs précises peut être facilitée par l'utilisation de ces types. Avec le type MONEY par exemple, on sait à priori que les chiffres n'auront jamais plus de deux décimales.

- CHAR(len): Une chaîne de caractères de longueur 'len'. Ce type de champ peut faire l'objet d'une recherche par mots-clé.
- TEXT(len): Une chaîne de caractères de longueur variable. Le paramètre 'len' permet de définir la longueur attendue de la chaîne et réserve de l'espace disque en fonction. Il est néanmoins possible de stocker un nombre de caractères supérieur à 'len' dans ce type de champ. Il n'est pas possible d'effectuer une recherche par mots-clé sur un champ de type TEXT.
- INT: valeurs entières (..., -1,-3, ..., 5,..., 777, ...).
- REAL: valeurs réelles (..., -3.456, ..., 34.567, ...).
- UINT: (0, 1, 2, ...).
- DATE: Dates dans le format, DD-Mon-YYYY, comme par exemple 1-Jan-1987.
- TIME: Heures en format 24 heures, par exemple 14:56:34
- MONEY: Argent sous la forme de nombres à deux décimales, par exemple, 34.05

1-4.2 Soumettre le fichier de commande à mSQL

Pour ne pas abîmer les bases de données installées sur nos machines, une base de donnée nommée 'test' a été créée sur la machine tecfasun1. Utilisez la pour faire vos essais.

Une fois le fichier de commande terminé, lancer la commande msql avec une redirection. On donne en argument, la machine sur laquelle se trouve le serveur mSQL (-h tecfasun2), le nom de la base de données (test) et le fichier contenant les commandes SQL:

```
tecfasun2:bin 338) msql -h tecfasun1 test < /comm/tecfa/www/swiss-
edu/msql/table-exemple.sql


Welcome to the miniSQL monitor.  Type \h for help.
mSQL >     ->     ->     ->     ->     ->     ->     ->     ->     -
>     ->    ->     ->     ->     ->     ->     ->     ->     ->
Query OK.  1 row(s) modified or retrieved.
```

Remarque:

Si vous obtenez des messages, d'erreur c'est que votre fichier de commande comporte une faute de syntaxe, ou que la base de données n'existe pas, ou encore que vous n'avez pas la permission de modifier cette base de données.

1-5 Effacer des tables

L'effacement d'une table est plus simple que sa création et consiste en une requête DROP. Attention, avant d'effacer les informations contenues dans la table il est conseillé de faire un backup des données sensibles. (Voir "Backup des bases de données" à la page 11).

Voici un exemple dans lequel nous effaçons la table ‘info_ed’ dans la base de données ‘test’ sur la machine ‘tecfasun1’.

```
tecfasun2:/ 28) msql -h tecfasun1 test

Welcome to the miniSQL monitor.  Type \h for help.

mSQL > drop table info_ed
    -> \g
Query OK.  1 row(s) modified or retrieved.
```

2. Importation et mise à jour

2-1 Préparation d'un fichier plat

Nous partons d'un fichier exporté dans le format tab-separated à partir d'une base de données standard (Access, Filemaker sur Mac ou PC). Le fichier ressemble à ceci:

```
Frau    Regula  Schräder-Naef   Schräder        Erwachsenenbildung -
Bildungsbiographien      Pädagogische Abteilung
                              Erziehungsdirektion Kanton Zürich
Walchestrasse 21 8090    Zürich          01 259 53 82 direkt
                                                   01 259 53 50 sekr
                                                                  01
259 51 30    Regula.Schraeder@ed.zh.admin.ch Jacob Burckhardt-Strasse
19  8049     Zürich          01/342 02 51            21.10.97
```

La commande mouline-tab permet de changer automatiquement les tabulateurs en accolades ({) et de remplacer les accents ISO en accents HTML. On donne le nom du fichier à convertir en argument. Voici comment on l'utilise:

```
tecfasun2: 104) mouline-tab info-ed.tab
Cleaning info-ed.tab

File contains 430 records
Replaced ISO accents by HTML accents
Replaced TAB by {
Result stored in info-ed.tab.converted
```

Désormais, le fichier `info-ed.tab.converted` est prêt à être importé dans mSQL. Il contient des lignes qui ressemblent à ceci:

```
Frau{Regula{Schr&auml;der-Naef{Schr&auml;der{Erwachsenenbildung -
Bildungsbiographien{P&auml;dagogische
Abteilung<br>Erziehungsdirektion Kanton Z&uuml;rich{{Walchestrasse
21{8090{Z&uuml;rich{{01 259 53 82 direkt<br>01 259 53 50 sekr<br>{01
259 51 30{Regula.Schraeder@ed.zh.admin.ch{Jacob Burckhardt-Strasse
19{8049{Z&uuml;rich{{01/342 02 51{{21.10.97
```

2-2 Préparation de la table

Deux situations peuvent se présenter. Dans la première nous venons de créer une nouvelle table (qui n'existait pas dans la base de données), nous avons préparé un fichier de données à importer. Dans ce cas nous passons directement au point 2-3.

Dans la seconde nous effectuons une mise à jour. Il faut alors suivre la recette suivante:

1. effacer la table (Voir "Effacer des tables" à la page 7)
2. recharger la structure de la table (Voir "Soumettre le fichier de commande à mSQL" à la page 7)
3. Importer le fichier plat (Voir section suivante)

2-3 Importation du fichier plat

La commande msqlimport permet de réaliser facilement des importations. Voici la syntaxe générique de la commande:

```
usage:  msqlimport [-h host] [-v] [-s Char] [-e Char] [-q Char]
database table

        Imports a formatted ASCII file from the standard input.

        -v              Verbose
      -s Char       Use the character Char as the separation character
                        Default is a comma.
        -q Char         Use the specifed Char as the quote character
                        Default is none
        -e Char        Use the specifed Char as the escape character
                        Default is \
```

Et voici comment importer notre fichier plat nommé `info-ed.tab.converted` dans la table ‘info_ed’ de la base de données nommée ‘test’:

```
tecfasun2: 34) msqlimport -h tecfasun1 -s \{ test info_ed < info-
ed.tab.converted
```

Si vous recevez des erreurs disant qu’un champ est trop grand pour l’importation, c’est qu’une de vos valeurs excède la taille de la colonne prévue dans le fichier de création de table. Il faut alors changer la taille du (des) champ(s) qui pose(nt) problème, recharger la table et recommencer l’importation

Attention, l’ordre des champs dans le fichier plat doit être identique à l’ordre des champs dans la base de données mSQL.

3. Administration du serveur mSQL

3-1 Backup des bases de données

Deux commandes mSQL permettent d'exporter des données.

- msqldump exporte la structure d'une ou de plusieurs tables et les données qu'elles contiennent sous la forme de requêtes SQL. Cette commande est particulièrement adaptée aux sauvegardes destinées à être réutilisées par mSQL ou par un autre système qui comprend SQL (ORACLE, etc.). En voici la syntaxe:

```
usage:  msqldump [-h host] [-f conf] [-v] [-t] [-c] [-w where_clause]
database [table]

Produce an ASCII dump of a database table or an entire database
```

Utilisée telle quelle, cette commande imprime à l'écran le contenu d'une base de données ou d'une table. Pour récupérer ce résultat nous utilisons une *redirection* qui permet d'écrire l'output d'une commande dans un fichier. La redirection s'exprime avec le signe >. Dans l'exemple qui suit nous sauvons la base de données `csre-info` et écrivons le résultat de la commande dans un fichier qui s'appelle `csre-info.backup`. Ce fichier ne doit pas forçément préexister à l'utilisation de la commande. Son nom est à discretion de l'utilisateur et ne doit pas forçément porter l'extension `.backup`.

```
tecfasun2:~ 74) msqldump -h tecfasun1 csre-info > csre-info.backup
```

- msqlexport produit un fichier plat et permet à l'utilisateur de spécifier lui-même le format d'exportation, en particulier les séparateurs de champ et d'échappement des caractères spéciaux. Cette commande est particulièrement adaptée pour exporter les données en vue de les réutiliser avec un autre système. Elle ne permet d'exporter qu'une table à la fois. Voici la syntaxe:

```
usage:  msqlexport [-h host] [-v] [-s Char] [-q Char] [-e Char]
database table

        Produce an ASCII export of the table.

        -v              Verbose
      -s Char       Use the character Char as the separation character
                        Default is a comma.
        -q Char        Quote each value with the specified character
        -e Char        Use the specifed Char as the escape character
                        Default is \
```

Comme pour la commande msqldump, il faut récupérer le résultat de la commande en effectuant une redirection à l'aide du signe >. Voici un exemple qui sauve les données de la table `edutech` de la base de données `tecfa` dans un fichier nommé `edutech.save`. Le caractère { est utilisé comme séparateur de champ.

```
tecfasun2:~ 79) msqlexport -h tecfasun1 -s \{ tecfa edutech >
edutech.save
```

3-2 Restoration d’un backup

Deux cas de figure se présentent.

Dans le premier nous avons utilisé msqlexport pour sauvegarder nos données. Pour les recharger nous procédons comme pour importer un fichier plat, voir “Importation du fichier plat” à la page 10.

Dans le second, nous avons utilisé msqldump pour sauvegarder nos données. Il suffit alors de soumettre le fichier de backup (qui est en fait un fichier de commandes SQL) à la commande msql comme suit. (Voir aussi “Soumettre le fichier de commande à mSQL” à la page 7)

```
tecfasun2 33): msql -h tecfasun1 test < test.backup
```

3-3 Créer une base de données

1. donner les permissions de lecture et d’écriture pour la base de données = éditer le fichier de configuration :
   /comm2/soft/msql/msql.acl

```
# Access control for mSQL
## Entries are of the form
#

database=mediothek
read=jermann,-root,nobody
write=jermannhost=*
access=local,remote
option=rfc931

database=csre-info
read=jermann,-root,nobody
write=jermann,meyer,bauhofer
host=*access=local,remote
option=rfc931
```

Lignes ajoutées avant la création de la base de données csre-info

L’utilisateur nobody donne des droits aux utilisateurs qui se connectent par le WWW. Il suffit de copier une section préexistante et de la modifier en fonction des besoins.

2. se connecter sur tecfasun1:
   depuis le shell UNIX, tapez ‘rsh tecfasun1’ et donnez votre mot de passe à la demande.

```
tecfasun2:~ 67) rsh tecfasun1
Password:
Last login: Sun Dec  1 19:25:12 from tecfasun2
X11 Display: tecfasun2:0
tecfasun1:~ 1)
```

3. utiliser la commande msqladmin pour créer la base de données. La même commande sert à effacer une base de données et à arrêter le serveur msql.

```
usage : msqladmin [-h host] [-q] <Command>

where command =  drop DatabaseName
                 create DatabaseName
                 shutdown
                 reload
                 version

 -q     Quiet mode.  No verification of commands.
```

La commande ci-dessous crée une base de données intitulée ‘csre-info’

```
tecfasun1:bin:~ 2) msqladmin create csre-info
Database "csre-info" created.
tecfasun1:bin:~ 3)
```

3-4 Effacer une base de données

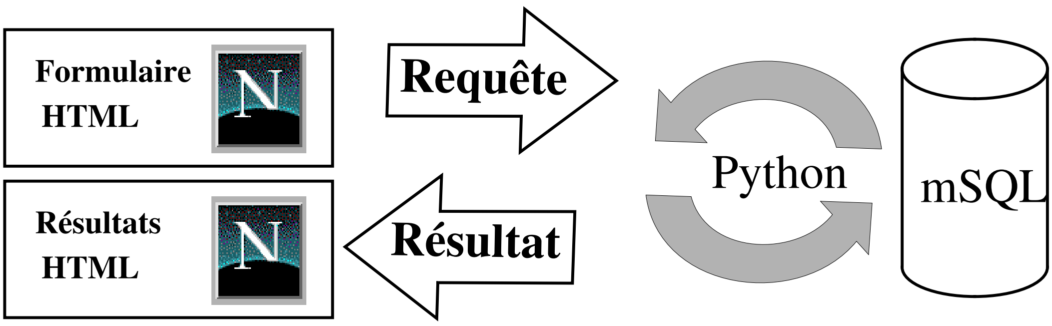
Attention, effacer une base de données efface toutes les informations que vous y aviez mises. Il n’y a aucun moyen pour récupérer les données.

```
tecfasun1: 17) msqladmin drop csre-info

Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.

Do you really want to drop the "csre-info" database?  [Y/N] y
Database "csre-info" dropped
```

4. Interface WWW

L’interface WWW à mSQL repose sur un ensemble de scripts Python qui sont capables de transformer le contenu d’un formulaire HTML en une requête SQL, de la soumettre à mSQL et de formatter le résultat en HTML.

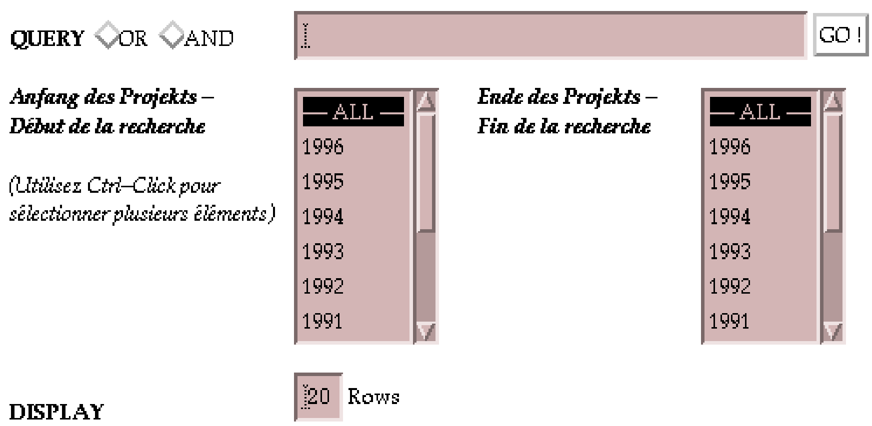
La majorité des interfaces WWW à nos bases de données repose sur deux scripts. Un premier script soumet une requête et retourne une liste d’enregistrements. Le second permet d’afficher en entier les informations contenues dans un seul enregistrement.



4-1 Formulaire HTML

Nous prenons comme exemple le formulaire qui permet d’interroger la base de données ‘Information sur la recherche éducationnelle’ qui se trouve à l’adresse suivante: http://agora.unige.ch/csre/info/msql/index.html.

Voici le formulaire tel qu’il apparaît dans Netscape. L’utilisateur peut entrer des mots-clé, choisir des items dans des listes à choix multiple et spécifier le nombre d’enregistrements qu’il souhaite voir sur l’écran.



Voici le code HTML (les tags relatifs à la mise en forme du formulaire dans un tableau ont été enlevés pour plus de lisibilité).

Les **champs obligatoires** dans les formulaires sont décrits ci-dessous. Soit l'utilisateur peut les définir lui-même, soit vous les spécifiez en les définissant avec un tag caché `<input type=hidden name=nom_du_champ value=votre_valeur>`. (Les champs `maxrow` et `output_type` sont des champs cachés dans l'exemple ci-dessous):

- `action` indique l0URL du script qui va traiter le formulaire.
- `boolean` peut prendre soit la valeur " OR " soit " AND " (espaces!). Ce paramètre spécifie le connecteur logique qui relie les mots-clé spécifiés dans le champ `query`.
- `query` contient les mots-clé entrés par l'utilisateur.
- `maxrow` défini le nombre d'enregistrements affichés sur un écran de résultats. Si le nombre d'enregistrements retournés par mSQL est supérieur à ce nombre, un bouton 'Show next records' permet de voir les `maxrow` enregistrements suivants.
- `output_type` peut prendre soit la valeur LIST soit la valeur TABULAR. Lorsque LIST est sélectionné, les champs des enregistrements sont affichés les uns après les autres. Lorsque TABULAR est sélectionné, les champs des enregistrements sont affichés dans une table HTML.

Les champs `vonChoice` et `bisChoice` sont spécifiques à notre exemple. Ils font référence à des listes à choix multiples. Voir la section "Choix multiples HTML" à la page 16.

```
<form name="queryForm" action="http://tecfa.unige.ch/cgi-bin/agora/
info-ed-96/info-ed-query-result.py" method=post>

<strong>QUERY</strong>
<input type=radio name=boolean value=" OR ">OR
<input type=radio name=boolean value=" AND ">AND

<input type=text size=40 name=query>
<input type=button value="GO !" onClick="makeQuery(this.form)">

<strong><i>Anfang des Projekts - <br>Début de la recherche </i></
strong><p><i>(Utilisez Ctrl-Click pour <br>sélectionner plusieurs
éléments)</i>

<select name=vonChoice multiple size=7>
<option selected>-- ALL --
<option>1996
<option>1995
etc..
<option>1991
<option>1990
<option>1989
</select>

<strong><i>Ende des Projekts - <br>Fin de la recherche</i></strong>

<select name=bisChoice multiple size=7>
<option selected>-- ALL --
<option>1996
<option>1995
<option>1994
etc...
<option>1990
<option>1989
</select>

<strong>DISPLAY</strong>
<input type=text name=maxrow size=2 value=20> Rows
<input type=hidden name=output_type value="LIST">
</form>
```

4-2 Choix multiples HTML

Lorsqu'un champ de la base de données contient des champs qui prennent un ensemble fini de valeurs, il est possible d'utiliser une liste à choix multiples pour effectuer des recherches.

Dans notre exemple, les champs `von` et `bis` contiennent les dates du début et de la fin d'une recherche: 1989, 1990, etc... La liste à choix multiples consiste en un tag SELECT et une série d'entrées spécifiées à l'aide du tag OPTION.

```
<select name=vonChoice multiple size=7>
<option selected>-- ALL --
<option>1996
<option>1995
<option>1994
<option>1993
<option>1992
<option>1991
<option>1990
<option>1989
</select>
```

Pour que Python puisse interpréter le choix de l'utilisateur il faut respecter d**eux contraintes**:

- le nom du champ est composé de deux parties: **`<nom_du_champ>Choice`**, dans notre exemple: **`vonChoice`**. <nom_du_champ> correspond au nom du champ en question dans la base de données mSQL.
- La première option doit s'appeller '**`-- ALL --`**' (avec un espace après -- et un espace après ALL). Elle est sélectionnée par défaut. Lorsque ALL est sélectionné le champ en question (von) est ignoré dans la requête. Ce n'est que lorsque une des options restantes est sélectionnées qu'elle est prise en compte,

4-3 Configuration de base d'un script Python

La configuration des scripts Python est simple et permet d'influer sur l'interface WWW et sur le mécanisme de recherche. ATTENTION, faites une **copie de sauvegarde** du script avant de le modifier.

Les scripts utilisés par l'interface WWW se trouvent dans les sous-répertoires de `/comm2/soft/httpd/cgi-bin/agora`. Les scripts qui servent d'interface aux différentes bases de données sont semblables et sont reconnaissables par les suffixes `query-result` et `full-display`.

Par exemple, dans le répertoire `/comm2/soft/httpd/cgi-bin/agora/info-ed-96/` se trouvent deux scripts:

- `info-ed-query-result.py*` transforme le contenu du formulaire HTML, soumet la requête à mSQL et renvoie une liste d'enregistrements.
- `info-ed-full-display.py*` affiche en entier un enregistrement

Notation

En Python, les chaînes de caractère sont entourées de guillemets : " ou '

```
dbname = "csre-info"
```

Pour écrire une chaîne de caractères sur plusieurs lignes on utilise les triple guillemets:

```
header = """ Ceci est le
titre de mon
document
"""
```

Les listes sont entourées de parenthèses carrées et leurs éléments sont séparés par des virgules:

```
multi_choice= ["SpracheChoice","vonChoice","bisChoice"]
```

4-3.1 Script 'query-result'

Cette section présente les paramètres qui règlent le fonctionnement du script. Le script qui correspond à l'exemple se trouve ici: `/comm2/soft/httpd/cgi-bin/agora/info-ed-96/info-ed-query-result.py`. La zone des paramètres se trouve au début du fichier et se termine par le signe:

```
... paramètres ....
                                        #
                                        ##
                                        # #
########################################   #
#                                           #
# No further changes below this line ...     #
#                                           #
########################################   #
                                        # #
                                        ##
                                        #
```

Paramètres de base

Au début du fichier, une série de commentaires (signalés par #) donne quelques indications que nous complétons ci-après.

- *host*: indique sur quelle machine réside la base de données qui est intérrogée par le script.
- *dbname*: indique le nom de la base de données que le script interroge.
- *table*: indique le nom de la table qui est concernée par le script
- *start_page*: l'URL du formulaire qui appelle ce script
- *addition_page*: l'URL du formulaire qui permet d'ajouter un enregistrement
- *full_script*: l'URL du script qui affiche un enregistrement au complet
- *query_script*: l'URL de ce script
- *temp_path*: le répertoire qui héberge les fichiers temporaires produits par le script
- *logfile*: le chemin et le nom du fichier qui contient une trace des requêtes soumises à ce script

host='tecfa.unige.ch'
dbname = "csre-info"
table = "info_ed"
start_page = "http://agora.unige.ch/csre/info/msql/index.html"
addition_page="http://agora.unige.ch/csre/info/msql/info_ed-addition-form.html"
full_script = "http://tecfa.unige.ch/cgi-bin/agora/info-ed-96/info-ed-full-display.py"
query_script = "http://tecfa.unige.ch/cgi-bin/agora/info-ed-96/info-ed-query-result.py"

temp_path= '/temp/msql.temp/'
logfile='/temp/msql.temp/info-ed-log'

Paramètres pour la requête

Ces paramètres permettent de régler l’étendue de la recherche et les champs à afficher en résultat.

- *select_list*: donne la liste des champs qui seront présents dans les résultats du script. Le premier champ doit toujours être `_rowid`.
- *where_list*: donne la liste des champs qui sont examinés en cas de recherche par mots-clé
- *multi_choice*: énumère les listes à choix multiple (cf. 4-2 “Choix multiples HTML”)
- *unique_list*: liste des champs dont la valeur ne doit apparaître qu’une seule fois dans les résultats. (Utile lorsqu’une information est redondante à l’intérieur des données).

select_list = [“_rowid”,”Titel”,”Titel_ubersetzt”,”Deskriptoren”,”Descripteurs”]
where_fields = [“Titel”,”Titel_ubersetzt”, “Deskriptoren”,”Descripteurs”]
multi_choice= [”vonChoice”,”bisChoice”]
unique_list=[]

Choix multiples

Ces paramètres sont les plus difficiles à fixer. Nous avons vus le principe de la liste à choix multiples dans la section 4-2 “Choix multiples HTML”.

Imaginons que nous avons un champ ‘langue’ dans notre base de données. Lors de la saisie, nous avons donné des codes aux langues. Ainsi, une valeur 1 signifie ‘allemand’, une valeur 2 signifie ‘français’ et une valeur 3 signifie ‘italien’. Nous ne pouvons pas demander à l’utilisateur de choisir entre trois numéros (il ne sait pas quelle est leur signification) !

Pour remédier à ce problème nous définissons des dictionnaires qui font correspondre une valeur entrée par l’utilisateur à une valeur à utiliser pour la recherche dans la base de données. Le script va faire correspondre la valeur choisie par l’utilisateur (français) dans le formulaire et la valeur qu’il doit rechercher dans la base de données (2).

Le nom du dictionnaire est composé de deux parties: `<nom_du_champ>Dict`. Dans notre exemple : `langueDict`. Voici le dictionnaire:

```
langueDict = {‘allemand’:’1’,
             ‘français’:’2’,
             ‘italien’:’3’
         }
```

- Les dictionnaires `vonDict` et `bisDict` correspondent aux éléments de formulaire `vonChoice` et `bisChoice` (cf. 4-2 “Choix multiples HTML”). Or, les valeurs des dates sont stockées sous une forme humainement compréhensible. Ainsi, dans cet exemple les dictionnaires ne servent pas à faire une transformation mais mettent en correspondance des valeurs identiques.
- *multi_boolean*: règle le connecteur logique lorsque plus d’une valeur est sélectionnée dans une liste à choix multiples. Les valeurs possibles de ce paramètre sont “ AND “ et “ OR “. (Il faut respecter les espaces avant et après).

multi_boolean = “ AND “

vonDict = {‘1996’:’1996’,
‘1995’:’1995’,
‘1994’:’1994’,
‘1993’:’1993’,
‘1992’:’1992’,
‘1991’:’1991’,
‘1990’:’1990’,
‘1989’:’1989’,
‘1988’:’1988’,
‘1987’:’1987’,
}

bisDict = {‘1996’:’1996’,
‘1995’:’1995’,
‘1994’:’1994’,
‘1993’:’1993’,
‘1992’:’1992’,
‘1991’:’1991’,
‘1990’:’1990’,
‘1989’:’1989’,
‘1988’:’1988’,
‘1987’:’1987’,
}

Contacts

- *bug_report*: l’adresse E-mail de la personne à conteacter en cas de problèmes.
- *contact_address:* l’adresse E-mail de la personne à contacter pour des questions d’ordre général.

bug_report = “Patrick.Jermann@tecfa.unige.ch”
contact_address = “skbf-csre@ping.ch”

Booléens

- *hide_rowid*: peut prendre les valeurs 0 ou 1. Lorsque hide_rowid est égal à 0 le numéro interne des enregistrements est affiché dans les résultats. (En général on laisse hide_rowid à 1).
- *debug*: peut prendre les valeurs 0 ou 1. Lorsque debug est égal à 1 des résultats intermédiaires sont ajoutés aux enregistrements affichés par le script. Ce mode permet de chercher une erreur éventuelle dans le programme en inspectant les variables.
- *allow_empty_query*: peut prendre les valeurs 0 ou 1. Si la valeur est 1, les utilisateurs peuvent lancer une recherche sans avoir spécifié de mot clé. Tous les enregistrements de la table sont alors affichés. Si la valeur est 0, un message d’erreur est présenté à l’utilisateur qui ne spécifie pas de mots-clé dans son formulaire.
- *enable_add*: ajoute un bouton ‘Add a record’ qui affiche un formulaire de saisie où les utilisateurs peuvent eux-même entrer des enregistrements.

hide_rowid = 1
debug = 0
allow_empty_query = 1
enable_add = 0

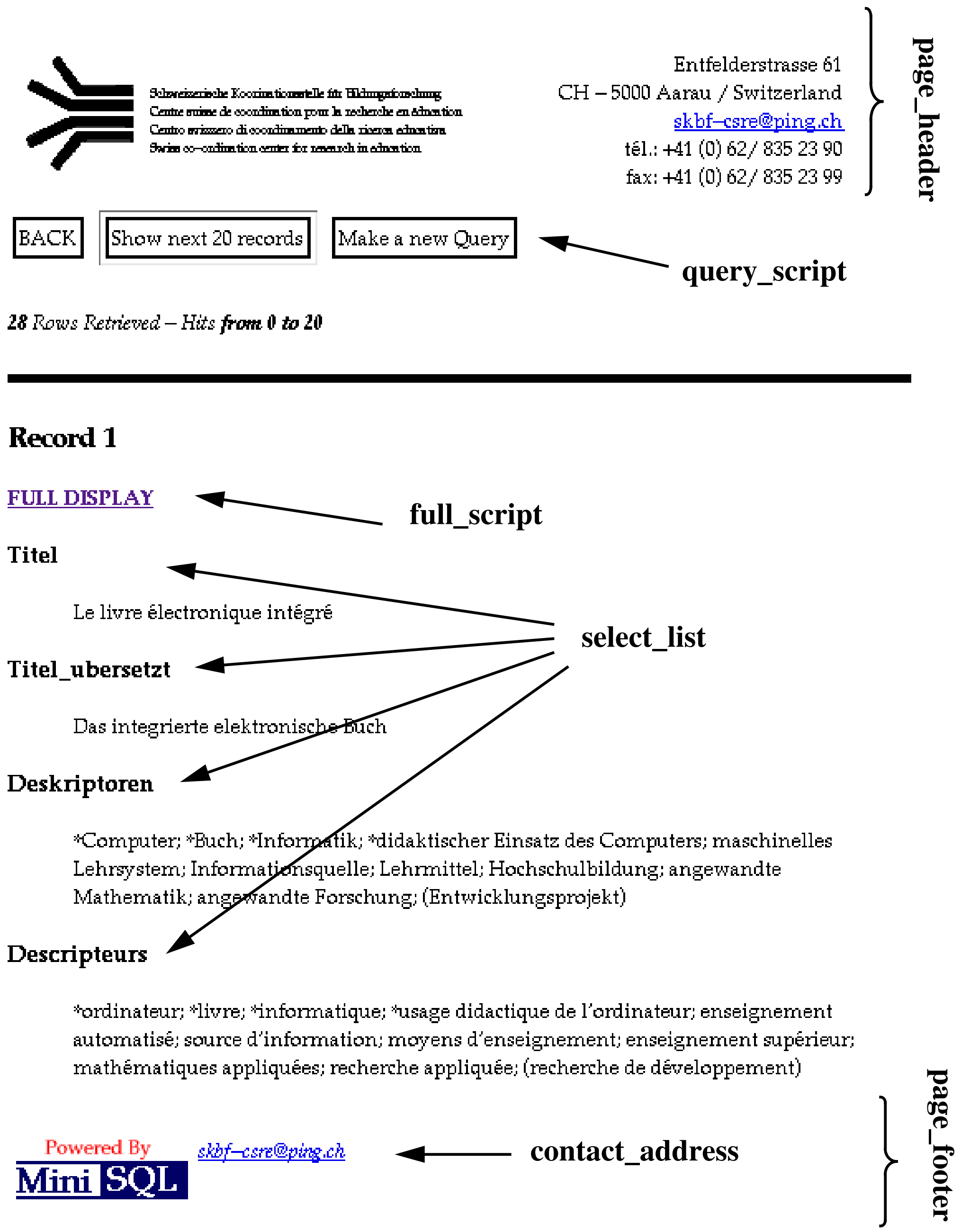
Header et Footer HTML

- *page_header*: du code HTML affiché en haut de la page produite par le script
- *page_footer*: du code HTML affiché en bas de la page produite par le script

```
page_header =  ““““
<table border=0>
<tr>
<td><a href=\”http://agora.unige.ch/csre/\”><img alt=\”CSRE logo\” width=76 height=80 border=0 src=\”http://
agora.unige.ch/csre/images/t_CSRE_logo.gif\”></a>
<td>
<font size=-2>
Schweizerische Koorinationsstelle für Bildungsforschung<br>
Centre suisse de coordination pour la recherche en éducation<br>
Centro svizzero di coordinamento della ricerca educativa<br>
Swiss co-ordination center for research in education
</font>
<td width=50>
<br>
<td align=right>
Entfelderstrasse 61<br>
CH - 5000 Aarau / Switzerland<br>
<a href=\”mailto:skbf-csre@ping.ch\”>skbf-csre@ping.ch</a><br>
t&eacute;l.: +41 (0) 62/ 835 23 90<br>
fax: +41 (0) 62/ 835 23 99<br>
</table>
““““
```

```
page_footer= """
<p><a href=\"http://Hughes.com.au/\"><img src=\"http://agora.unige.ch/images/msql-pwr.gif\" width=110 height=40
border=0 align=left></a>
<address>      <a href=\"mailto:%s\">%s</a></address>
</body></html>
""" % (contact_address,contact_address)
```

4-3.2 Output du script query-result



4-3.3 Script full-display

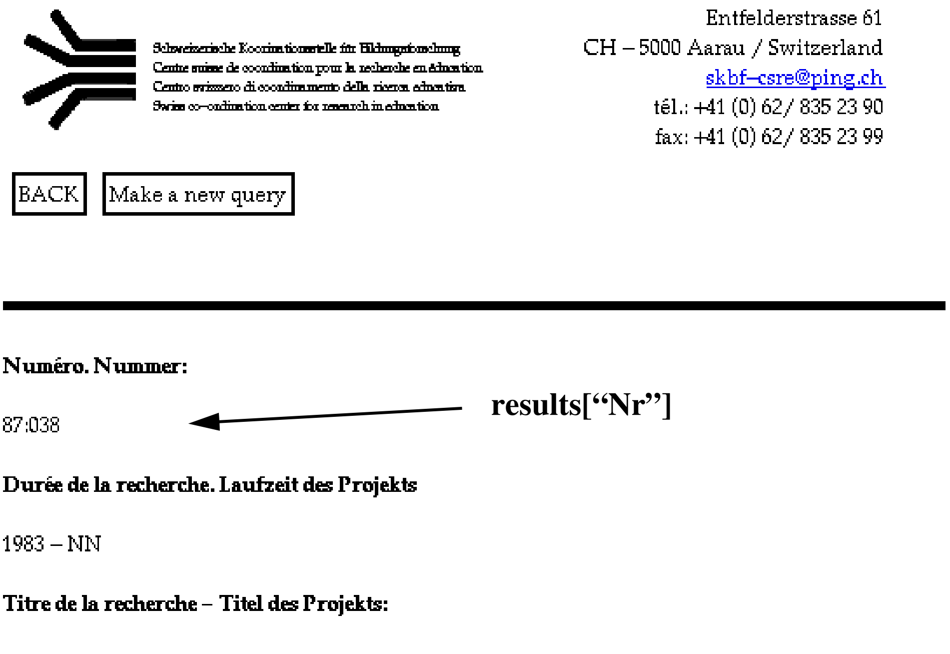
Le script full-display affiche en entier un seul enregistrement de la base de données. Le script qui correspond à l'exemple se trouve ici: `/comm2/soft/httpd/cgi-bin/agora/info-ed-96/info-ed-full-display.py`. La modification de ce script permet de changer l'apparence de la page HTML qu'il produit.

Au milieu du script se trouve une commande print qui produit le corps de la page.

Les %s sont remplacés dans l’ordre par les valeurs spécifiées à la fin de la commande print après le signe “““ %. Ainsi, le premier %s qui apparaît est remplacé par la valeur de `results["Nr"]`, le numéro de la recherche. Pour ajouter un hypothétique champ ‘langue’ il suffirait d’ajouter un %s dans le print et d’ajouter `results["langue"]` à la fin du print (attention à le mettre à la bonne place).

```
print “““
<H4>Numéro. Nummer:</H4>
%s
<h4>Durée de la recherche. Laufzeit des Projekts</h4>
%s - %s
<h4>Titre de la recherche - Titel des Projekts:</h4>
<h2>%s</h2>
<h2>%s</h2>
<H4>Institution:</H4>
%s
<H4>Chercheurs. Bearbeiter des Projekts</H4>
%s
<H4>Brève description de la recherche:</H4>
%s
<H4>Kurzbeschreibung des Projekts:</H4>
%s
<H4>Descripteurs (EUDISED):</H4>
%s
<H4>Deskriptoren (EUDISED):</H4>
%s
<H4>Publications. Veröffentlichungen:</H4>
%s
<H4>Méthodes de recherche. Methodische Anlage des projekts:</H4>
%s
<H4>Délimitation géographique. Geographischer Raum:</H4>
%s
<H4>Type de recherche. Art des projekts:</H4>
%s
<H4>Mandataire de la recherche. Auftragsgeber des Projekts:</H4>
%s
<H4>Financement de la recherche. Finanzierung:</H4>
%s
“““ %(results[“Nr”],results[“von”],results[“bis”],results[“Titel”],results[“Titel_ubersetzt”],
results[“Institution”],results[“Forscher”],results[“Abstract_frz”],results[“Abstract_dt”],results[“Descripteurs”],results
[“Deskriptoren”],results[“Publikationen”],results[“Meth”],results[“Geo”],results[“Typ”],results[“Auftrag”],results[“F
inanzierung”])
```

4-3.4 Output du script full-display



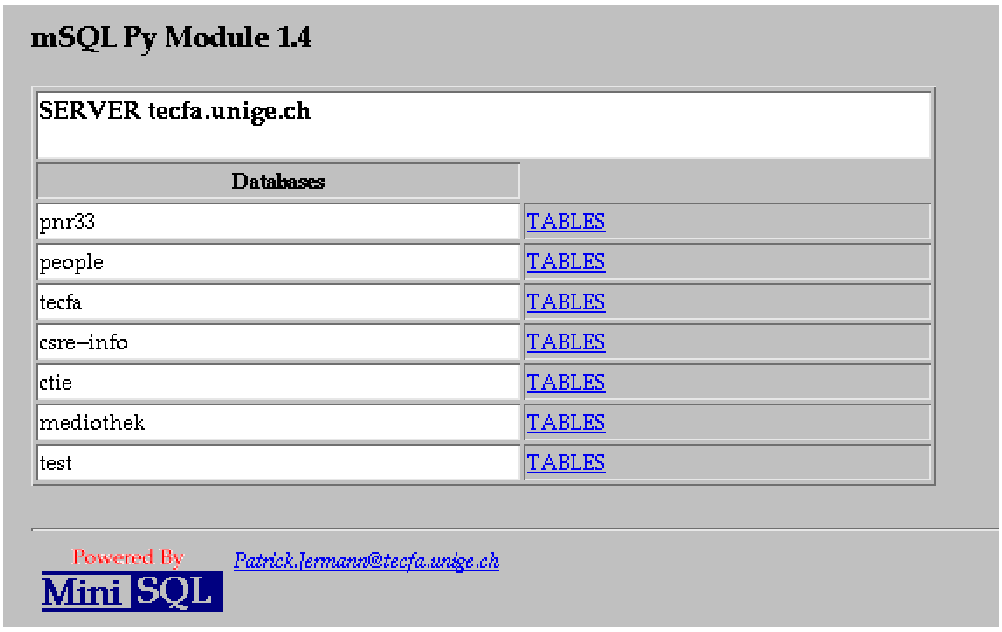
4-4 Interface d'Administration

Il existe une interface 'administrateur' qui permet de manipuler toutes les bases de données installées sur tecfasun1. Il est possible d'ajouter de nouveaux enregistrements, d'en modifier et d'en effacer. Cette page est protégée par un mot de passe.

L'URL pour y accéder: http://tecfa.unige.ch/cgi-bin/msql/main.py

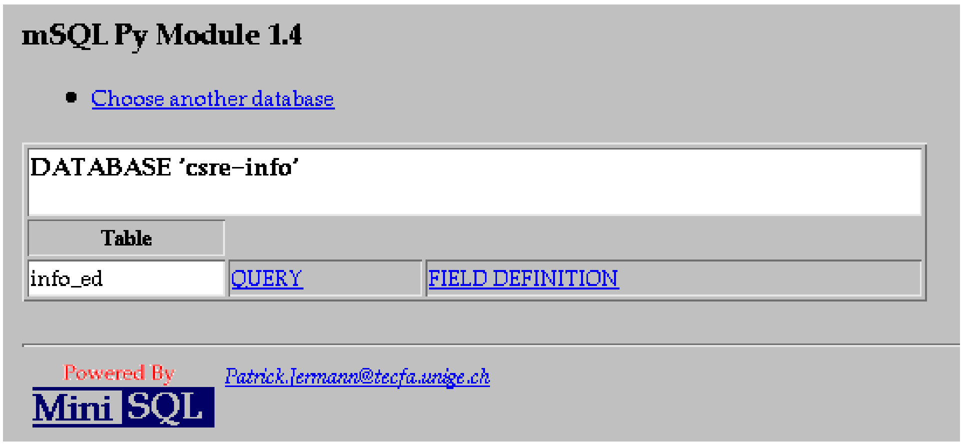
Page d'acceuil

Cette page permet de choisir la base de données qui nous intéresse. Cliquons sur csre-info à titre d'exemple.



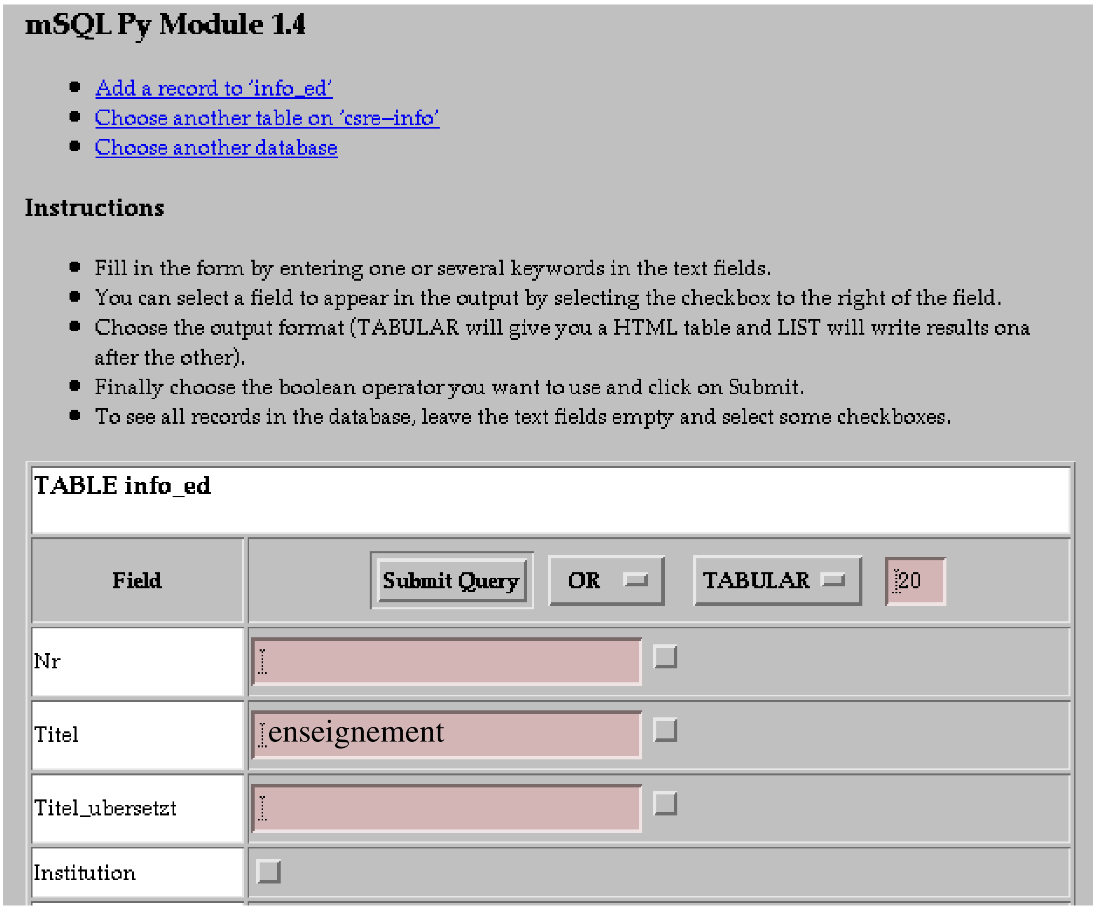
Page 'Table'

Cette page offre deux choix. En cliquant sur FIELD DEFINITION, les définitions des champs de la table sont affichés. (Cf. 1-1 "Lister les bases de données, les tables et les champs"). En cliquant sur QUERY un formulaire de recherche est affiché.



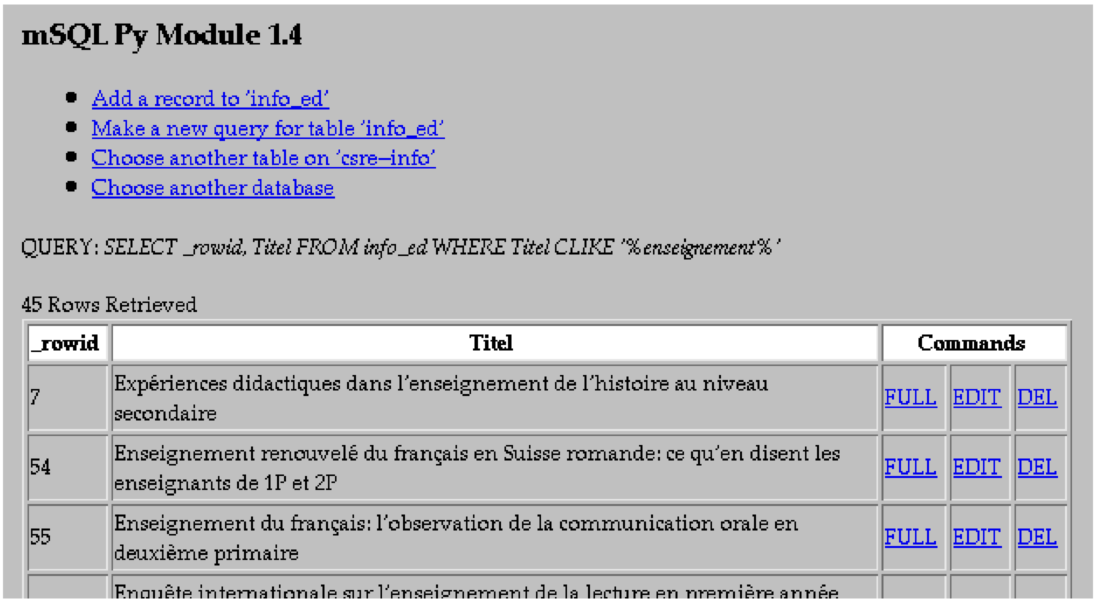
Page 'Query'

Outre un pointeur vers la pages 'Addition' d'un enregistrement présentée plus bas, cette page permet de composer de requêtes précises sur un ou plusieurs champs et de choisir précisément quels champs seront affichés en résultat. En entrant un mot-clé dans une zone de texte, on demande à mSQL de le rechercher dans un seul champ. Les champs sans zone de texte sont de type TEXT (Cf. 1-4 "Créer des tables"). Lorsque les checkbox à droite des zones de texte sont sélectionnés, les champs correspondants figurent dans les résultats. On peut en plus choisir le nombre d'enregistrements à afficher et le connecteur logique à utiliser.



Page 'Query result'

Voici la page qui suit la soumission du formulaire 'Query'. Pour chaque enregistrement qui correspond à la requête, trois commandes sont à disposition: FULL affiche l'enregistrement au comple, EDIT permet de le modifier et DEL l'efface (attention...). L'utilisation de ces pages ne devrait pas poser de problèmes et ne sont par conséquent pas décrites ici. (...probieren geht über studieren ;)



mSQL Py Module 1.4

- Add a record to 'info_ed'
- Make a new query for table 'info_ed'
- Choose another table on 'csre–info'
- Choose another database

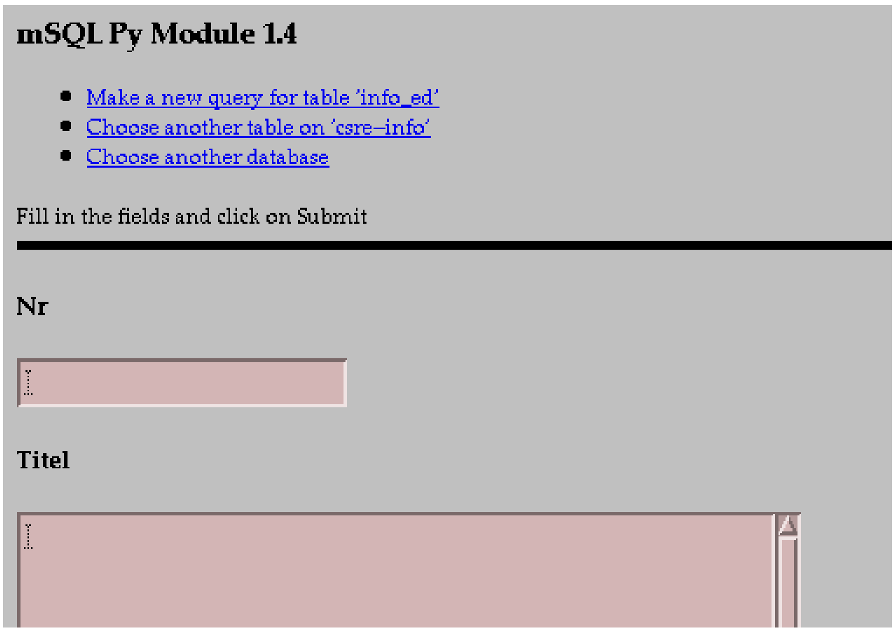
QUERY: *SELECT _rowid, Titel FROM info_ed WHERE Titel CLIKE '%enseignement%'*

45 Rows Retrieved

_rowid	Titel	Commands		
7	Expériences didactiques dans l'enseignement de l'histoire au niveau secondaire	FULL	EDIT	DEL
54	Enseignement renouvelé du français en Suisse romande: ce qu'en disent les enseignants de 1P et 2P	FULL	EDIT	DEL
55	Enseignement du français: l'observation de la communication orale en deuxième primaire	FULL	EDIT	DEL
	Enquête internationale sur l'enseignement de la lecture en première année			

Page 'Addition'

La page d'addition est un formulaire qui permet de saisir une nouvelle entrée. Veillez à respecter les types de champs en entrant des données (ne mettez pas de texte dans un champ INT).



5. Références

URLS

Référence mSQL : http://tecfa.unige.ch/guides/msql/msql2/welcome.htm

Version Postscript: http://tecfa.unige.ch/guides/msql/mSQL.ps

mSQL FAQ: http://tecfa.unige.ch/guides/msql/msql2/faq.html

Home Page Python: http://www.python.org/

Référence Python: http://tecfa.unige.ch/guides/python/python.html

Interface Python: http://www.ollie.clive.ia.us/jeff/python/msql/

Livres

Livre mSQL: à paraître.

Python: Mark Lutz. (1996) Programming Python. O'Reilly & Associates.

6. Annexe

6-1 Syntaxe SQL

Pour une description détaillée, se référer à l'URL suivant: http://tecfa.unige.ch/guides/msql/msql2/spec.htm.

CREATE

```
CREATE TABLE table_name (
                  col_name col_type [ not null ]
                  [ , col_name col_type [ not null ] ]**
            )

CREATE [ UNIQUE ] INDEX index_name ON table_name (
                  field_name
                  [ , field_name ] **
            )

CREATE SEQUENCE ON table_name [ STEP step_val ] [ VALUE initial_val ]
```

DROP

```
DROP TABLE table_name
DROP INDEX index_name FROM table_name
DROP SEQUENCE FROM table_name
```

INSERT

```
INSERT INTO table_name [ ( column [ , column ]** ) ]
```

SELECT

```
SELECT [table.]column [ , [table.]column ]**
        FROM table [ = alias] [ , table [ = alias] ]**
        [ WHERE [table.] column OPERATOR VALUE
        [ AND | OR [table.]column OPERATOR VALUE]** ]
        [ ORDER BY [table.]column [DESC] [, [table.]column [DESC] ]


OPERATOR can be <, >, =, <=, >=, <>, LIKE, RLIKE or CLIKE
VALUE can be a literal value or a column name

Where clauses may contain '(' ')' to nest conditions e.g. "where (age
< 20 or age > 30) and sex = 'male'" .
```

DELETE

```
DELETE FROM table_name
                  WHERE column OPERATOR value
                  [ AND | OR column OPERATOR value ]**

OPERATOR can be <, >, =, <=, >=, <>, LIKE, RLIKE, or CLIKE
```

UPDATE

```
UPDATE table_name SET column=value [ , column=value ]**
                  WHERE column OPERATOR value
                       [ AND | OR column OPERATOR value ]**

OPERATOR can be <, >, =, <=, >=, <>, LIKE, RLIKE or CLIKE
```