Premiers pas avec SQLfast

Niveau Basic

1 décembre 2020

Objectif

Ce premier tutoriel montre comment utiliser l'outil SQLfast pour la manipulation élémentaire des bases de données et du langage SQL. Il décrit également certaines fonctions de base de l'interface SQLfast.

SQLfast offre deux niveaux d'utilisation :

- **Basic** - Apprentissage et utilisation des bases de données et du langage SQL
- **Expert** - Apprentissage et utilisation du langage SQLfast, programmation et exploitation d'applications de bases de données.

Le présent document est consacré à l'utilisation de SQLfast au niveau **Basic**. Il résulte de la conversion du guide ***Getting started*** du niveau **Basic**, accessible via l'écran d'accueil et le bouton **Help** de la fenêtre principale

Avertissement

Tant le logiciel que le(s) langage(s) SQLfast sont en évolution constante. Il est donc possible que certaines descriptions et certains documents soient relatifs à des versions plus anciennes. Ils seront actualisés dès que possible.

Le logiciel référencé dans ce tutoriel est la version 64 bits sous Windows, exécutable sous toutes les versions standard de Windows disponibles depuis janvier 2015. Elle fonctionne également sous Linux via Wine. Dans ce dernier cas, quelques ajustements dans le fichier d'initialisation "SQLfast.ini" peuvent s'avérer nécessaires, dont certains "paths" par défaut et le codage des fichiers de texte (par exemple 'UTF-8' au lien du standard Windows 'cp1252').

D'autres fonctions sont en préparation mais n'ont pas encore fait l'objet d'une description. Il est recommandé de vérifier régulièrement l'état d'avancement de ces fonctions.

Il va sans dire que les conséquences de l'utilisation des outils, des langages et des documents relatifs à SQLfast sont de la totale responsabilité des utilisateurs. L'utilisation de ces produits n'engage en aucune manière la responsabilité des auteurs de ces derniers ni celle de l'université de Namur.

Contact : jean-luc.hainaut@unamur.be

Table des matières

1. Installation

Le logiciel SQLfast aura été installé selon les instructions du document SQLfast_Installation.pdf disponible à l'adresse https://projects.info.unamur.be/~dbm/mediawiki/index.php/DUNOD2015_SQLfast.

La figure 1 montre l'état du répertoire SQLfast, installé à la racine du disque **D:**. Il contient divers répertoires et fichiers dont seuls nous intéressent pour débuter :

- **SQLfast.exe** : le logiciel SQLfast. Pour simplifier son utilisation, il peut être utile d'en créer un raccourci sur le bureau (voir Figure 2).
- **Databases** : le répertoire par défaut dans lequel seront stockées les bases de données que nous créerons et utiliserons. Plusieurs bases de données sont déjà disponibles et prêtes à l'emploi, dont CLICOM.db et ORDERS.db (sa traduction en anglais).

Les autres fichiers et répertoires sont utiles ou nécessaires au fonctionnement de SQLfast mais ne nous intéressent pas pour l'instant.

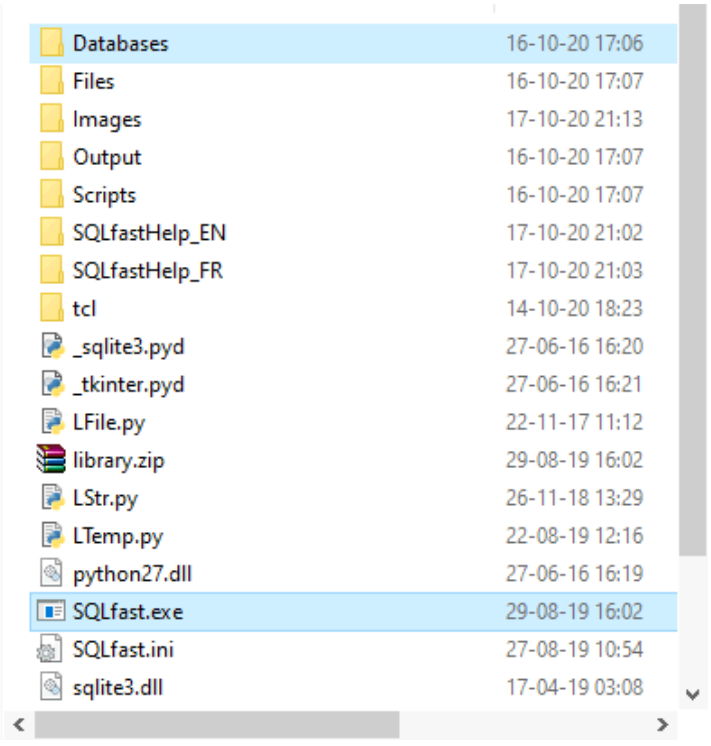


Figure 1 - Le contenu du répertoire SQLfast

2. Démarrage de SQLfast

Double-cliquer sur **SQLfast.exe** dans le répertoire **SQLfast** ou sur son raccourci. Les deux fenêtres principales de l'interface graphique de SQLfast apparaissent (figure 2) :

– **fenêtre principale** : C'est à partir de cette fenêtre que nous allons effectuer les principales manipulations. Elle comporte en particulier un espace - dit *zone de script* - dans lequel nous allons écrire les requêtes SQL à exécuter.

– **fenêtre de sortie** : (ou SQLfast output Window) C'est dans cette fenêtre que vont s'afficher les résultats de l'exécution des requêtes SQL. On y trouvera aussi divers messages d'information sur les conditions d'exécution des requêtes.

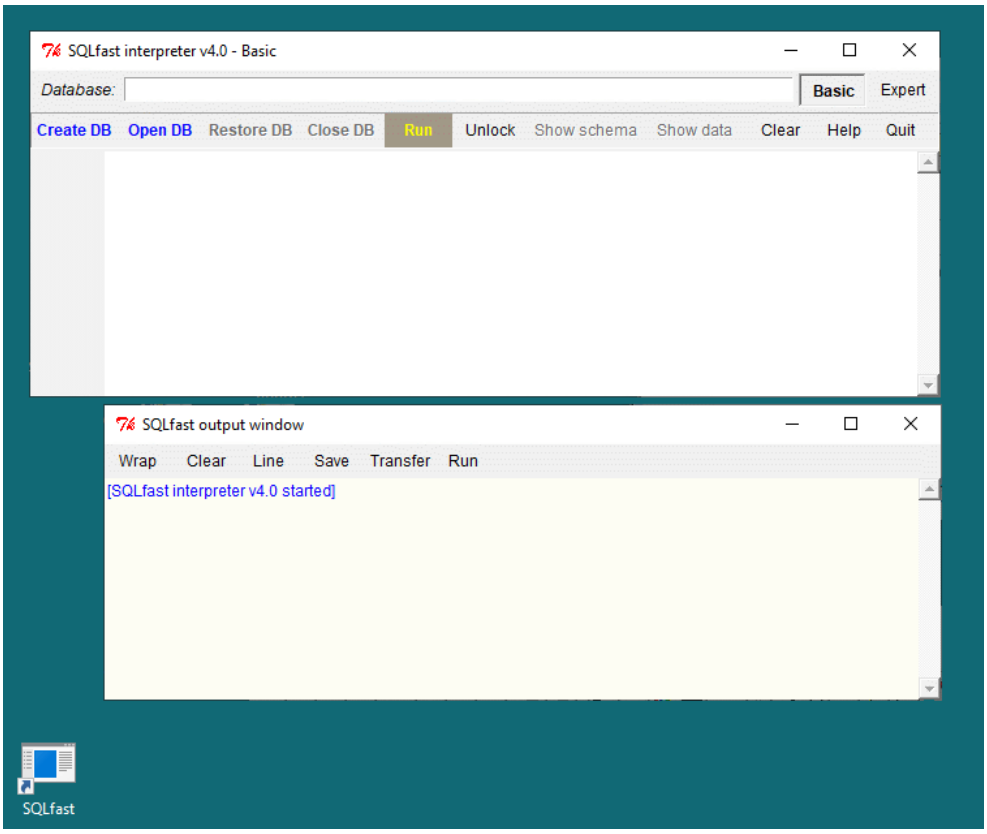


Figure 2 - Les fenêtres de l'interface SQLfast

Remarque

La fenêtre comporte, en haut à droite, deux boutons libellés "**Basic**" et "**Expert**". Le bouton "**Basic**" doit être enfoncé (Figure 3).

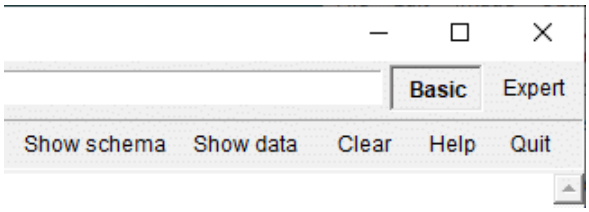


Figure 3 - Sélection du mode de l'interface

Lorsqu'aucune base de données n'est ouverte, les boutons **Create DB** et **Open DB** sont actifs alors que les boutons **Restore DB**, **Close DB**, **Show schema** et **Show data** sont désactivés.

En revanche, dès qu'une base de données est ouverte, tous les boutons sont actifs, sauf **Create DB** et **Open DB** (Figure 5).

3. Quitter SQLfast (bouton *Quit*)

On quitte SQLfast en fermant la fenêtre principale : bouton **Quit** (le dernier de la barre des boutons) ou bouton standard **X** à droite de la barre de titre. Toutes les autres fenêtres se ferment automatiquement. Si une base de données est encore ouverte, elle est également fermée.

4. Ouvrir une base de données (bouton *Open DB*)

Pour examiner une base de données, il est nécessaire au préalable de l'ouvrir via le bouton bleu **Open DB**, qui présente une liste des bases de données du répertoire Database (Figure 4).

Nous sélectionnons la base de données CLICOM.db. Son nom apparaît dans le champ *Database* en haut de la fenêtre principale (figure 5). Cette base de données devient notre *base de données courante*.

Remarque

SQLfast crée, lors de l'ouverture de la base de données, une copie de sauvegarde (qui reçoit ici le nom CLICOM.db.bak).

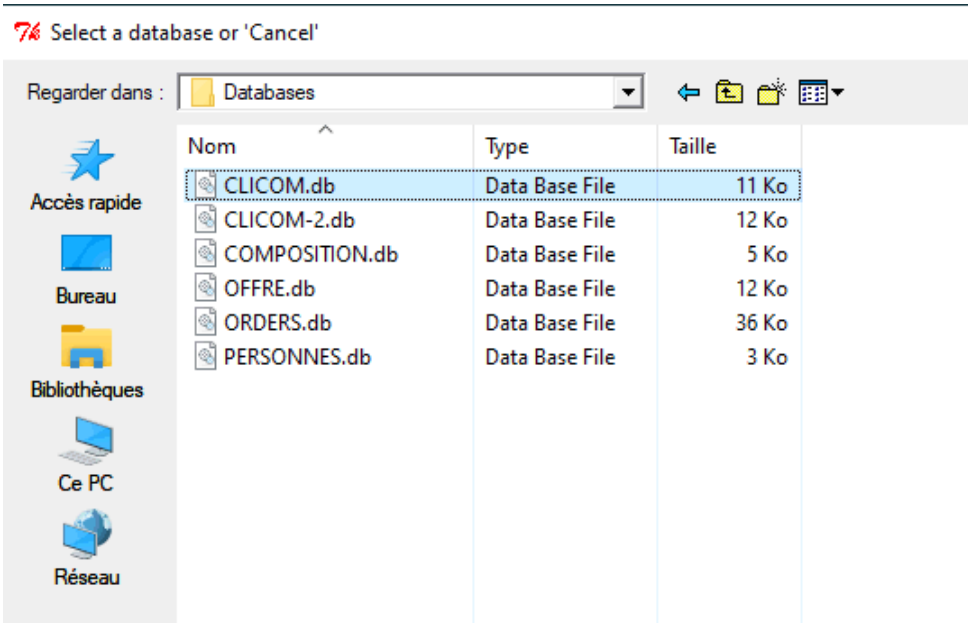


Figure 4 - Sélection d'une base de données à ouvrir

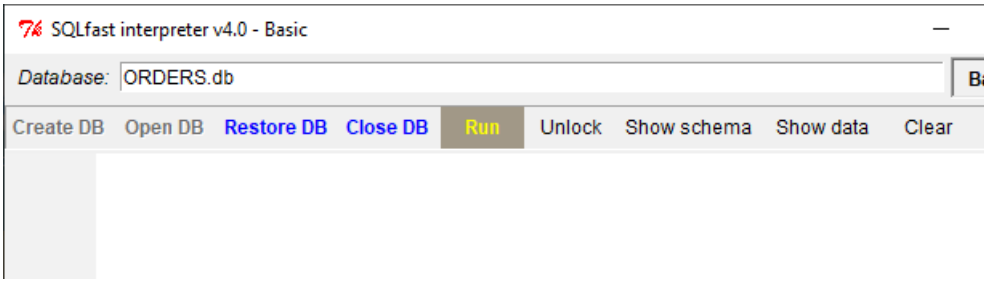


Figure 5 - La base de données CLICOM.db est ouverte

5. Examiner la base de données courante

La base de données CLICOM.db ayant été ouverte, nous allons, avant de manipuler son contenu à l'aide du langage SQL, l'examiner de plus près et effectuer quelques manipulations élémentaires.

5.1 Le schéma (bouton *Show schema*)

La première chose à connaître d'une base de données est son **schéma**, c'est-à-dire la manière dont les données sont organisées. En cliquant sur le bouton **Show schema**,

nous ouvrons une fenêtre dans laquelle une synthèse du schéma de la base de données courante apparaît (figure 6).

On y apprend que :

– la base de données comprend quatre tables

– chaque table comprend des colonnes

– chaque colonne contient des données d'un certain type; la présence d'un astérisque indique que la colonne est obligatoire ("not null")

– certaines colonnes font partie de l'*identifiant primaire* ou *clé primaire* de la table. Une marque **PK2** indique que la colonne appartient à la clé primaire, à la position 2. On précise par exemple que la table DETAIL possède une clé primaire formée des colonnes NCOM et NPRO, dans cet ordre.

Il est conseillé d'ouvrir cette fenêtre lorsque nous rédigerons des requêtes SQL.

On ferme la fenêtre du schéma par le bouton **Close**.

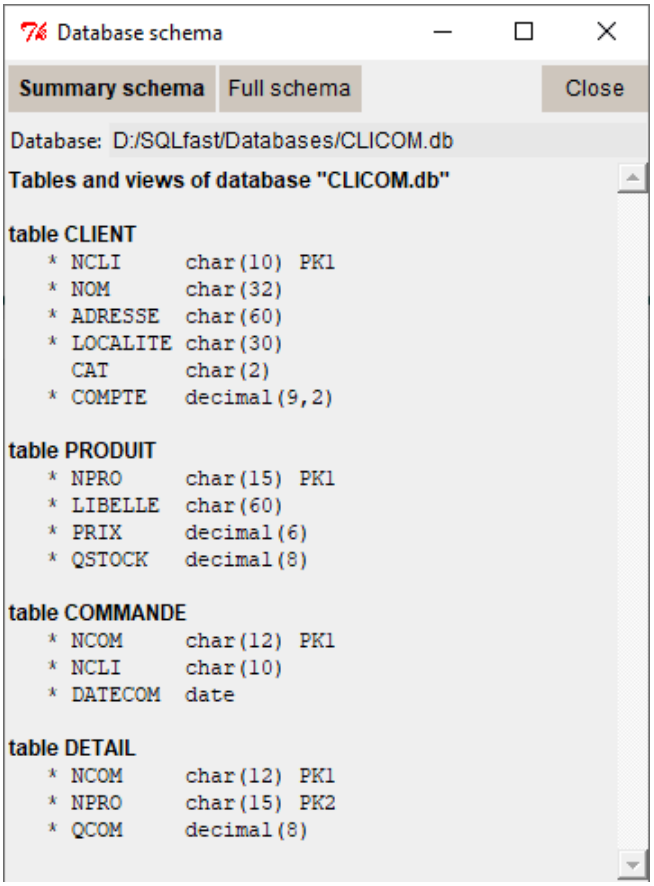


Figure 6 - Schéma synthétique de la base de données CLICOM.db

La figure 6 est un résumé (*summary* en anglais) du schéma de la base de données, ce qu'indique l'étiquette "**Summary schema**" du bouton à gauche du bandeau supérieur. Pour visualiser le schéma complet, on clique sur le bouton "**Full schema**" (figure 7). S'affiche alors le code SQL-DDL du schéma..

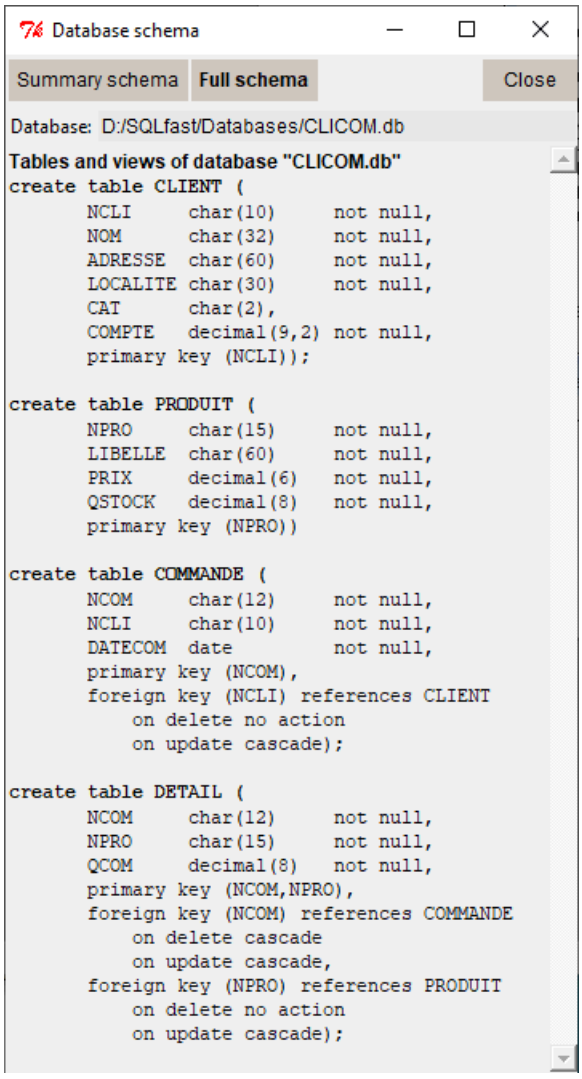
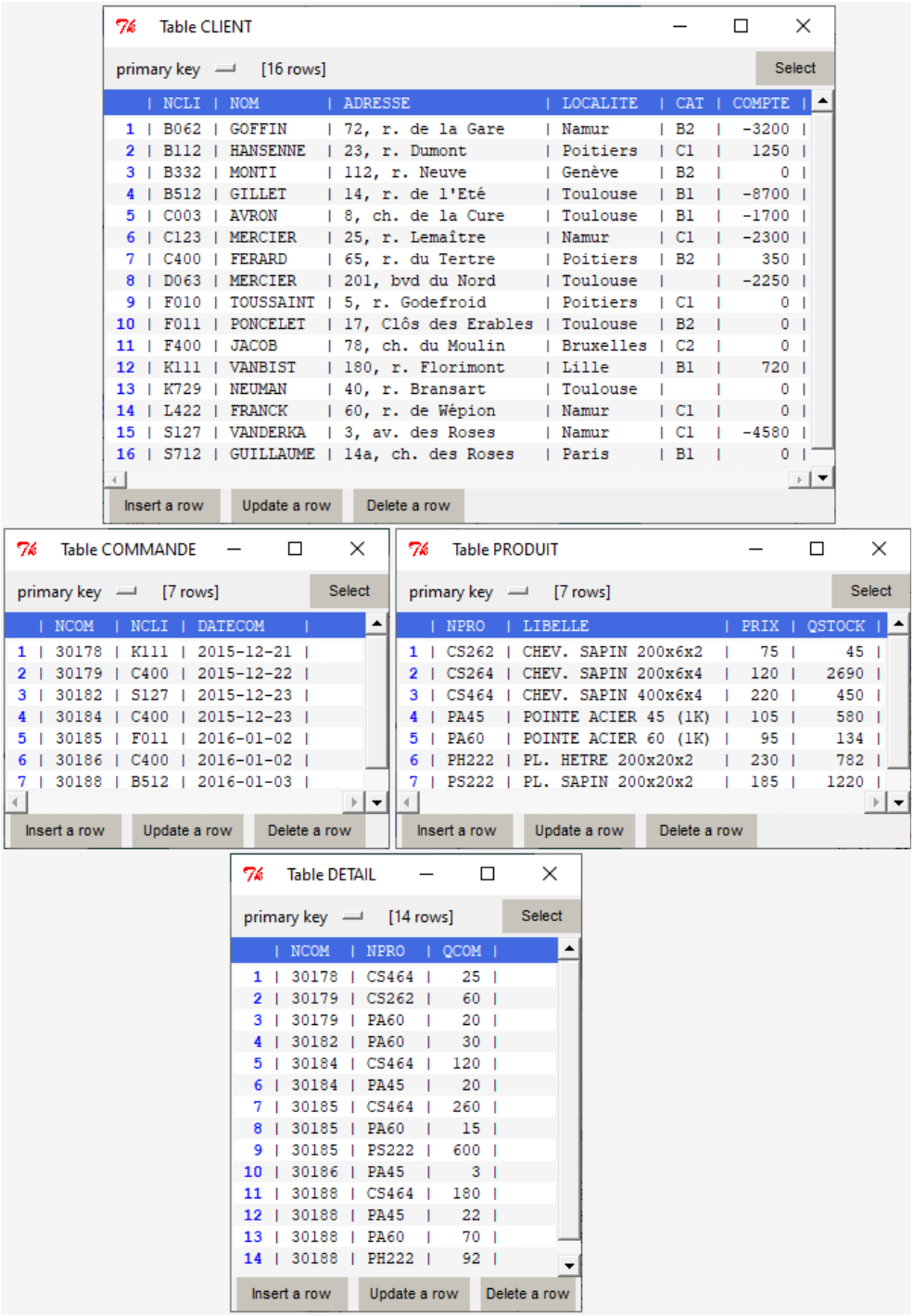


Figure 7 - Schéma SQL-DDL complet de la base de données CLICOM.db

5.2 Les données (bouton *Show data*)

Le bouton **Show data** sera utilisé pour visualiser le contenu des tables de la base de données. A chaque table correspond une *fenêtre de données* (figure 8).



	NCLI	NOM	ADRESSE	LOCALITE	CAT	COMPTE
1	B062	GOFFIN	72, r. de la Gare	Namur	B2	-3200
2	B112	HANSENNE	23, r. Dumont	Poitiers	C1	1250
3	B332	MONTI	112, r. Neuve	Genève	B2	0
4	B512	GILLET	14, r. de l'Eté	Toulouse	B1	-8700
5	C003	AVRON	8, ch. de la Cure	Toulouse	B1	-1700
6	C123	MERCIER	25, r. Lemaître	Namur	C1	-2300
7	C400	FERARD	65, r. du Tertre	Poitiers	B2	350
8	D063	MERCIER	201, bvd du Nord	Toulouse		-2250
9	F010	TOUSSAINT	5, r. Godefroid	Poitiers	C1	0
10	F011	PONCELET	17, Clôs des Erables	Toulouse	B2	0
11	F400	JACOB	78, ch. du Moulin	Bruxelles	C2	0
12	K111	VANBIST	180, r. Florimont	Lille	B1	720
13	K729	NEUMAN	40, r. Bransart	Toulouse		0
14	L422	FRANCK	60, r. de Wépion	Namur	C1	0
15	S127	VANDERKA	3, av. des Roses	Namur	C1	-4580
16	S712	GUILLAUME	14a, ch. des Roses	Paris	B1	0

	NCOM	NCLI	DATECOM
1	30178	K111	2015-12-21
2	30179	C400	2015-12-22
3	30182	S127	2015-12-23
4	30184	C400	2015-12-23
5	30185	F011	2016-01-02
6	30186	C400	2016-01-02
7	30188	B512	2016-01-03

	NPRO	LIBELLE	PRIX	QSTOCK
1	CS262	CHEV. SAPIN 200x6x2	75	45
2	CS264	CHEV. SAPIN 200x6x4	120	2690
3	CS464	CHEV. SAPIN 400x6x4	220	450
4	PA45	POINTE ACIER 45 (1K)	105	580
5	PA60	POINTE ACIER 60 (1K)	95	134
6	PH222	PL. HETRE 200x20x2	230	782
7	PS222	PL. SAPIN 200x20x2	185	1220

	NCOM	NPRO	QCOM
1	30178	CS464	25
2	30179	CS262	60
3	30179	PA60	20
4	30182	PA60	30
5	30184	CS464	120
6	30184	PA45	20
7	30185	CS464	260
8	30185	PA60	15
9	30185	PS222	600
10	30186	PA45	3
11	30188	CS464	180
12	30188	PA45	22
13	30188	PA60	70
14	30188	PH222	92

Figure 8 - Les fenêtres de données de la base de données CLICOM.db

Les fenêtres de données sont accompagnées d'une boîte de commande (figure 9) qui comprend deux boutons :

– **Select tables**, qui permet de choisir les tables à visualiser,

– **Close all**, qui permet de fermer toutes les fenêtres de données.

La sélection des tables à visualiser s'opère par une boîte de dialogue dans laquelle apparaît la liste des tables et des vues de la base de données courante (figure 10).

Le bouton **Select all** sélectionne toutes les tables. En cliquant sur une table de cette sélection, on la retire de celle-ci.

Par exemple, la sélection de la figure 10 pourraient être obtenue soit en cliquant sur chacune des trois tables, soit en sélectionnant toutes les tables et en en retirant DETAIL.

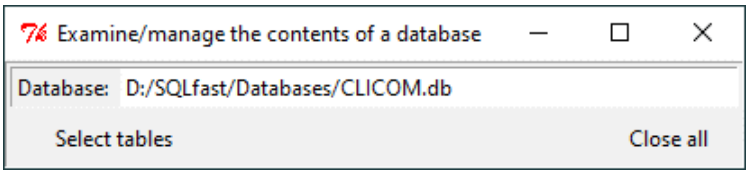


Figure 9 - Boîte de commande des fenêtres de données

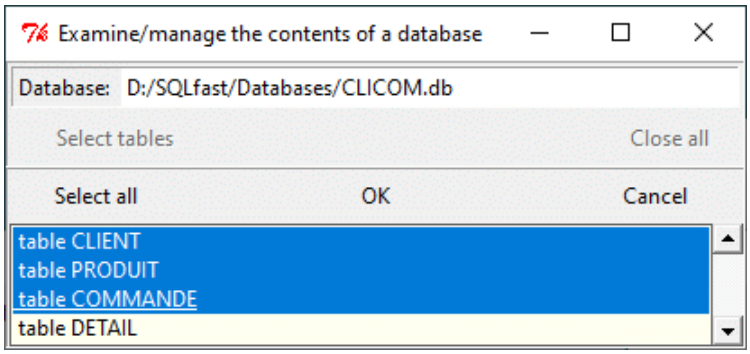


Figure 10 - Sélection des tables à visualiser

6. Trier les données

Le bouton de gauche de la fenêtre de données, généralement intitulé "primary key", du moins avant qu'on l'active, permet de spécifier l'ordre dans lequel les lignes apparaissent dans la fenêtre. On dira que les lignes sont *triées* selon le *critère* indiqué par le bouton.

L'ordre de tri par défaut est celui des valeurs de l'*identifiant primaire* ou *clé primaire* (primary key) de la table : NCLI pour la table CLIENT, NPRO pour PRODUIT, NCOM pour COMMANDE et le couple (NCOM, NPRO) pour la table DETAIL (figure 8).

Si la table ne possède pas d'identifiant primaire, les lignes apparaissent par défaut dans un ordre quelconque, appelé -natural-. Tel est également le cas des vues, qui n'ont pas de clé primaire déclarée.

D'autres critères de tri peuvent être choisis via le bouton de gauche, qui indique le *critère de tri courant*. On les fait apparaître en cliquant sur ce bouton. La Figure 11 montre qu'on va choisir de trier les lignes selon les valeurs croisantes de la colonne LOCALITE.

Le résultat est illustré à la Figure 12.

Un ordre des lignes choisi judicieusement permet de mettre en évidence certaines propriétés intéressantes des données :

sélectionnons pour la table CLIENT le critère de tri LOCALITE, pour obtenir la liste de la figure 12. Celle-ci crée les groupes de lignes de même valeur de LOCALITE. On observe, par exemple,

– que les clients de Toulouse sont les plus nombreux
– qu'aucun client n'habite à Madrid.

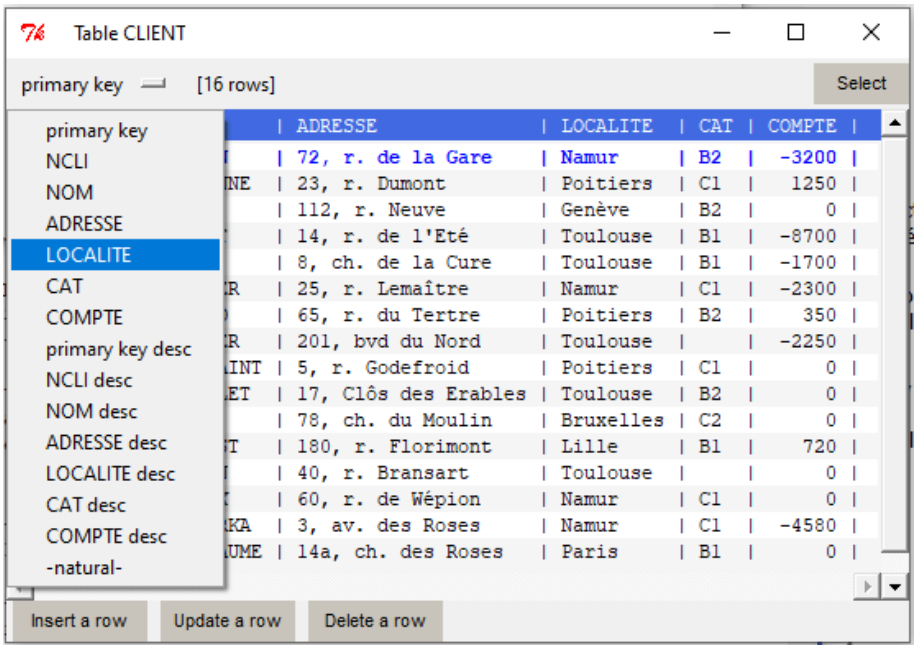
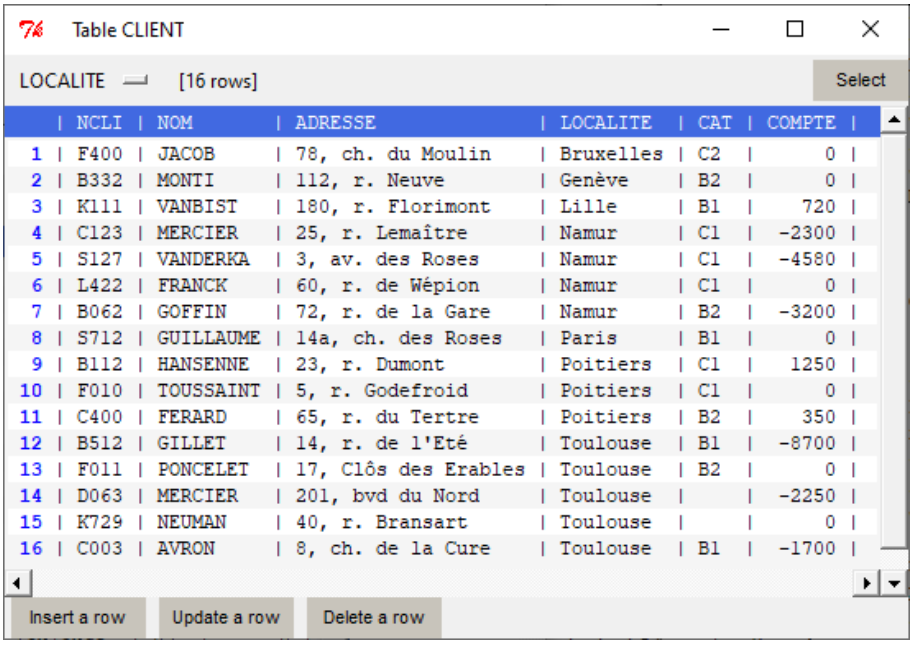


Figure 11 - Les lignes de CLIENTS sont triées par LOCALITE



	NCLI	NOM	ADRESSE	LOCALITE	CAT	COMPTE
1	F400	JACOB	78, ch. du Moulin	Bruxelles	C2	0
2	B332	MONTI	112, r. Neuve	Genève	B2	0
3	K111	VANBIST	180, r. Florimont	Lille	B1	720
4	C123	MERCIER	25, r. Lemaître	Namur	C1	-2300
5	S127	VANDERKA	3, av. des Roses	Namur	C1	-4580
6	L422	FRANCK	60, r. de Wépion	Namur	C1	0
7	B062	GOFFIN	72, r. de la Gare	Namur	B2	-3200
8	S712	GUILLAUME	14a, ch. des Roses	Paris	B1	0
9	B112	HANSENNE	23, r. Dumont	Poitiers	C1	1250
10	F010	TOUSSAINT	5, r. Godefroid	Poitiers	C1	0
11	C400	FERARD	65, r. du Tertre	Poitiers	B2	350
12	B512	GILLET	14, r. de l'Eté	Toulouse	B1	-8700
13	F011	PONCELET	17, Clôs des Erables	Toulouse	B2	0
14	D063	MERCIER	201, bvd du Nord	Toulouse		-2250
15	K729	NEUMAN	40, r. Bransart	Toulouse		0
16	C003	AVRON	8, ch. de la Cure	Toulouse	B1	-1700

Figure 12 - Les lignes de CLIENTS sont triées par LOCALITE

Critères de tri multiples

Il est facile de trier les lignes selon un critère simple : l'identifiant principal (primary key) ou une colonne, soit par valeurs croissantes, soit par valeurs décroissantes (suffixe **desc**).

Mais est-il possible de trier les lignes selon **deux critères**, comme le montre la figure 13 ? On y voit que les lignes sont d'abord triées selon LOCALITE, puis, dans chaque groupe de même valeur de LOCALITE, les lignes sont triées par valeurs croissantes de NCLI. Globalement, les lignes sont triées sur le couple (LOCALITE, NCLI). LOCALITE est le *critère majeur* et NCLI le *critère mineur*.

Pratiquement, on obtient la liste de la figure 13 en deux étapes comme suit :

– on trie d'abord les lignes selon les valeurs du *critère mineur* NCLI,

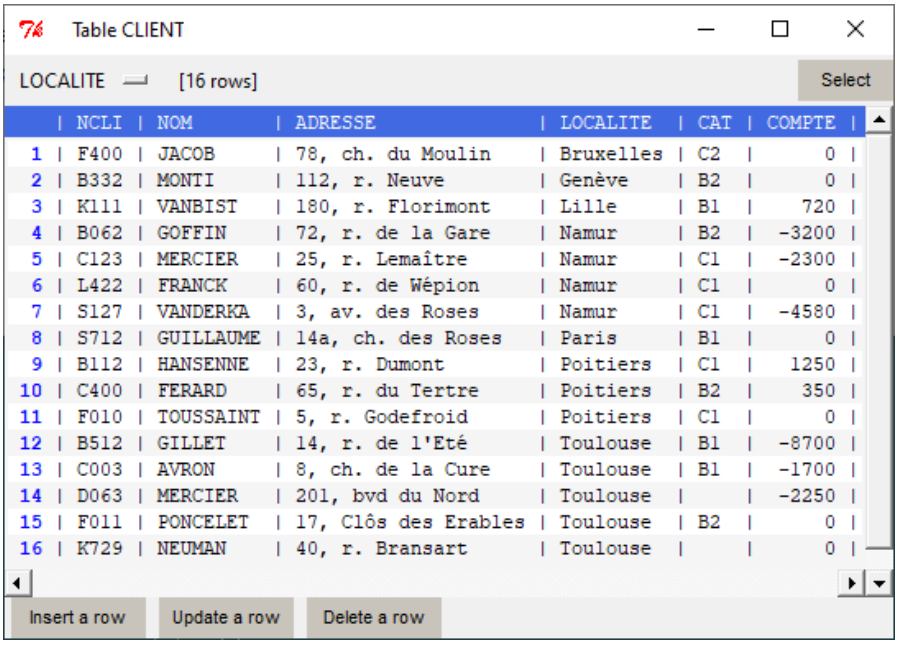
– ensuite, on trie les lignes selon les valeurs du *critère majeur* LOCALITE.

On observe que l'ordre obtenu lors du premier tri (NCLI) est *préservé* lors du second tri (LOCALITE) pour les lignes de même valeur de LOCALITE. Les spécialistes diront que l'opération de tri de SQLfast est **stable**.

Notes

– Ces opérations ne modifient pas le contenu de la base de données mais simplement l'aspect des données dans la fenêtre de données.

– La manipulation des données dans une fenêtre de données n'autorise que les tris à deux niveaux, *majeur* et *mineur*, et basés uniquement sur des colonnes. Les tris plus complexes nécessiteront l'usage de requêtes SQL.



	NCLI	NOM	ADRESSE	LOCALITE	CAT	COMPTE
1	F400	JACOB	78, ch. du Moulin	Bruxelles	C2	0
2	B332	MONTI	112, r. Neuve	Genève	B2	0
3	K111	VANBIST	180, r. Florimont	Lille	B1	720
4	B062	GOFFIN	72, r. de la Gare	Namur	B2	-3200
5	C123	MERCIER	25, r. Lemaître	Namur	C1	-2300
6	L422	FRANCK	60, r. de Wépion	Namur	C1	0
7	S127	VANDERKA	3, av. des Roses	Namur	C1	-4580
8	S712	GUILLAUME	14a, ch. des Roses	Paris	B1	0
9	B112	HANSENNE	23, r. Dumont	Poitiers	C1	1250
10	C400	FERARD	65, r. du Tertre	Poitiers	B2	350
11	F010	TOUSSAINT	5, r. Godefroid	Poitiers	C1	0
12	B512	GILLET	14, r. de l'Eté	Toulouse	B1	-8700
13	C003	AVRON	8, ch. de la Cure	Toulouse	B1	-1700
14	D063	MERCIER	201, bvd du Nord	Toulouse		-2250
15	F011	PONCELET	17, Clôs des Erables	Toulouse	B2	0
16	K729	NEUMAN	40, r. Bransart	Toulouse		0

Figure 13 - Les lignes de CLIENTS sont triées par LOCALITE puis par NCLI

7. Sélectionner les lignes d'une fenêtre de données

Il est souvent utile de n'afficher qu'un sous-ensemble des lignes d'une table ou d'une vue, particulièrement lorsque le nombre de lignes est important.

On définit ce sous-ensemble par un *filtre*, qui est une condition que les lignes doivent satisfaire pour apparaître dans la fenêtre.

7.1 Définition d'un filtre

Cherchons par exemple à visualiser les données des clients de Toulouse.

En cliquant sur le bouton **Select**, à droite du bandeau supérieur de la fenêtre de données, on fait apparaître une boîte qui nous permet de définir un filtre (figure 14).

Ce filtre est formé à partir de trois champs :

– **Column** : on y sélectionne dans la liste déroulante la colonne sur laquelle portera la condition, ici, LOCALITE;

– **Operator** : on y sélectionne dans la liste déroulante l'opérateur de comparaison, ici, **=**;

– **Simplified value** : on y introduit la valeur de référence à laquelle seront comparées les valeurs de LOCALITE des lignes, ici, Toulouse.

On a ainsi défini le filtre :

```
LOCALITE = Toulouse
```

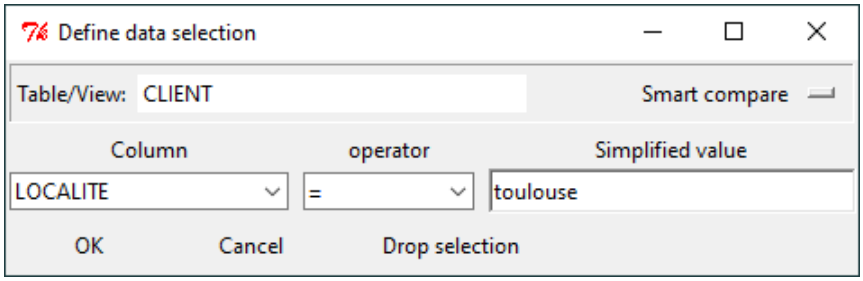
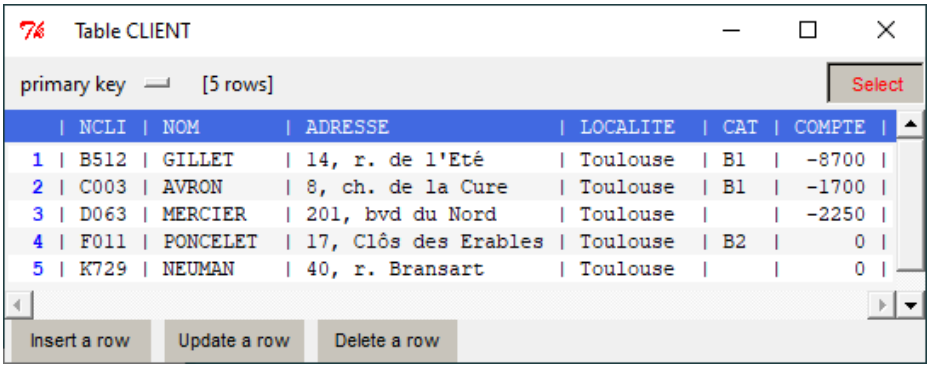


Figure 14 - Définition d'un filtre sélectionnant les clients de Toulouse

Si nous cliquons sur le bouton **OK** de la boîte, le contenu de la fenêtre de la table CLIENT est modifié comme le montre la figure 15.



	NCLI	NOM	ADRESSE	LOCALITE	CAT	COMPTE
1	B512	GILLET	14, r. de l'Eté	Toulouse	B1	-8700
2	C003	AVRON	8, ch. de la Cure	Toulouse	B1	-1700
3	D063	MERCIER	201, bvd du Nord	Toulouse		-2250
4	F011	PONCELET	17, Clôs des Erables	Toulouse	B2	0
5	K729	NEUMAN	40, r. Bransart	Toulouse		0

Figure 15 - Le contenu de la fenêtre de données est réduit aux lignes des clients de Toulouse

Comme attendu, on n'y trouve plus que les lignes des clients de Toulouse. En outre, l'aspect du bouton **Select** est modifié de manière à indiquer clairement qu'un filtre de sélection est actif.

Les lignes de la fenêtre réduite peuvent être triées, modifiées et supprimées. Il est possible d'y ajouter de nouvelles lignes, mais celle-ci, même si elles ont été introduites dans la table, n'apparaissent dans la fenêtre que si elle satisfont le filtre.

7.2 Modifier et supprimer un filtre

En cliquant à nouveau sur le bouton **Select**, on peut modifier le filtre ou le supprimer via le bouton **Delete selection**. Dans ce dernier cas, la fenêtre reprend à nouveau la totalité des lignes de la table.

7.3 Les modes de sélection

À droite du bandeau supérieur de la boîte de définition du filtre, une étiquette indique **Smart compare** (figure 16). Elle spécifie le "mode de sélection", c'est-à-dire la manière dont les trois composants du filtre doivent être interprétés.

En cliquant sur cette étiquette, on peut activer un autre mode de sélection (figure 16) :

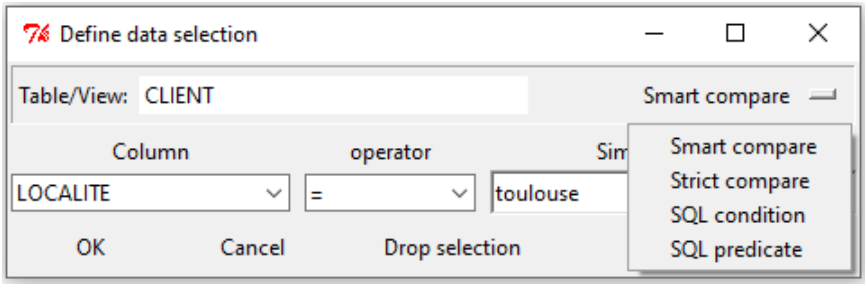


Figure 16 - Choix du mode de sélection

Il existe quatre modes de sélection : **Smart compare**, **Strict compare SQL condition** et **SQL predicate** (Figure 17).

– **Smart compare**. Les composants **Column** et **Operator** doivent être choisis dans les listes proposées. Les opérateurs reprennent ceux du langage SQL mais comprennent aussi des formulations plus naturelles : *starts with*, *contains*, *is among*, etc. Chose importante : le composant **Simplified value** peut être introduit selon une orthographe très laxiste : les accents peuvent être omis et les majuscules et minuscules sont confondues. Les apostrophes délimitant les valeurs peuvent être ignorées.

Ainsi, les filtres suivants sont équivalents :

```
LOCALITE  =  Genève
LOCALITE  =  'geneve'
LOCALITE  =  GENEVE
```

Les nombres peuvent inclure une virgule décimale ou un point décimal. Les séparateurs apparaissant dans une date peuvent être "-" ou "/". On pourra aussi écrire :

```
COMPTE  >  10,5
LOCALITE  =  Saint-Martin-d'Hères
```

```
LOCALITE  is among  parîs,Geneve,'POITIERS'
DATE  is between  2020-01-01,'2020/03/31'
```

Lorsque le filtre est défini, on clique sur le bouton **OK** pour l'activer. La fenêtre de données ne montre plus que les lignes qui satisfont le filtre.

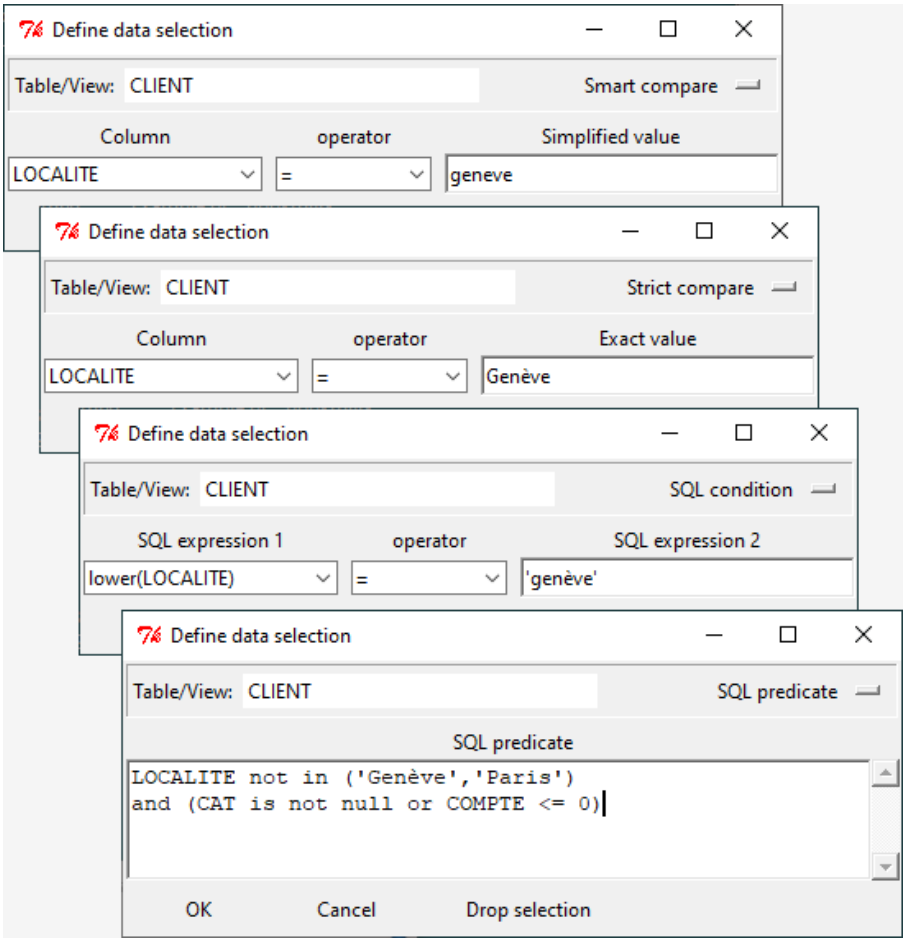


Figure 17 - Boîtes de construction d'un filtre dans les quatre modes

– **Strict compare**. Ce mode est similaire au précédent pour les composants **Column** et **Operator**, mais exige une orthographe exacte des chaînes des valeurs de référence. Les majuscules, minuscules et lettres accentuées doivent être introduites comme elles sont enregistrées dans la base de données. Les apostrophes délimitant les valeurs peuvent être ignorées.
Ainsi, pour la sélection des clients de Genève, seul le filtre suivant est valide :

```
LOCALITE  =  Genève
```

On devra aussi écrire :

```
LOCALITE  is among  Paris,Genève,Poitiers
```

– **SQL condition**. Les composants **Column** et **Value** s'intitulent désormais **Expression 1** et **Expression 2**. **Expression 1** et **Operator** peuvent être choisis dans les listes proposées ou introduits manuellement. La caractéristique essentielle de ce mode est que l'assemblage des trois composants doit produire une condition élémentaire **strictement conforme** au langage SQL :

```
LOCALITE  =  'Genève'
LOCALITE  =  'Saint-Martin-d''Hères'
LOCALITE  like  '%Saint-Martin%'
LOCALITE  in  ('Paris','Genève','Poitiers')
DATE  between  '2016-01-01' and '2016-03-31'
lower(NOM)  like  '%'||lower(LOCALITE)||'%'
COMPTE  >=  (select avg(COMPTE) from CLIENT
```

– **SQL predicate**. La boîte de saisie ne propose plus la saisie de trois composants, mais un champ de texte dans lequel on introduira un prédicat SQL de complexité quelconque, pouvant combiner des conditions élémentaires à l'aide des opérateurs logiques **and**, **or** et **not**, éventuellement formulé en plusieurs lignes.
Exemple de prédicat écrit en deux lignes :

```
LOCALITE not in ('Genève','Paris')
and (CAT is not null or COMPTE <= 0)
```

Le bouton **Drop selection** supprime le filtre. Toutes les lignes de la table apparaissent à nouveau dans la fenêtre de données.

8. Modifier les données à partir des fenêtres de données

Comme nous le verrons plus loin, la modification des données s'effectue normalement par les requêtes SQL **`insert`** (insérer des lignes dans une table), **`update`** (modifier des lignes d'une table) et **`delete`** (supprimer des lignes d'une table).

Il est néanmoins possible de modifier les données directement via les fenêtres de données, sans passer par les requêtes SQL de modification.

– Pour **insérer une nouvelle ligne** : cliquer sur le bouton **Insert a row** de la fenêtre de données. Remplir le formulaire de saisie de données (figure 18) et cliquer sur **OK**. Il est en outre possible d'insérer une série de lignes dont certaines colonnes peuvent contenir les mêmes valeurs. Dans l'exemple illustré, on indique notre intention d'introduire plusieurs lignes et que celles-ci auront, par défaut, les mêmes valeurs de LOCALITE, CAT et COMPTE.

– Pour **modifier ou supprimer une ligne existante** : un double clic sur la ligne concernée ouvre une boîte de dialogue qui permet de modifier OU de supprimer cette ligne (figure 19). Dans le cas d'une vue, la boîte de dialogue présente simple-

ment les données mais ne permet pas de les modifier. *Remarque* : les boutons Update a row et Delete a row sont actuellement inactifs.

Les modifications directes des données sont également soumises aux contraintes d'intégrité définies dans le schéma de la base de données (identifiants, clés étrangères, colonnes "not null", types de données). Si la modification demandée est acceptée, le nouveau contenu des tables est automatiquement répercuté dans les fenêtres de données.

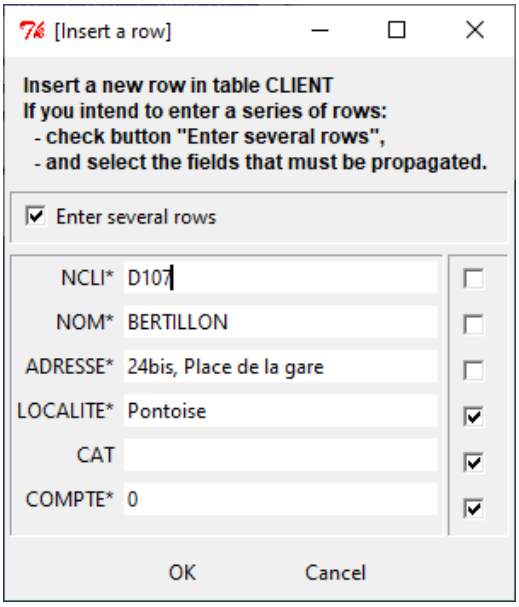


Figure 18 - Formulaire d'insertion d'une ligne de CLIENT

Si nous cochons le bouton **Delete the current row** dans la boite de la Figure 19 et si nous validons par le bouton "OK", l'opération de suppression sera refusée (Figure 20). En effet, plusieurs lignes de DETAIL dépendent de la ligne que nous voudrions supprimer.

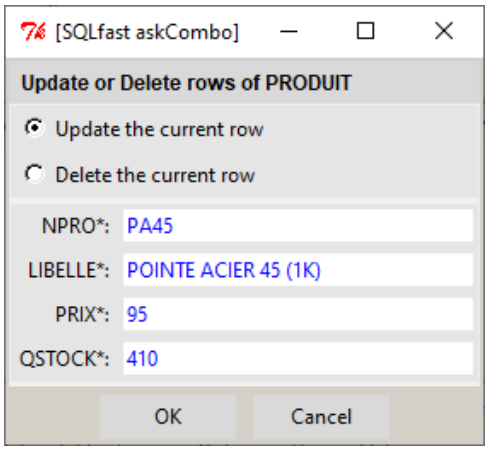


Figure 19 - Une boîte unique pour modifier ou supprimer une ligne

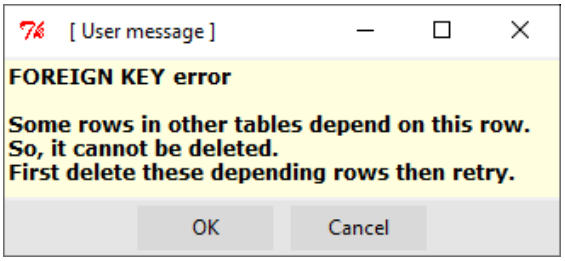


Figure 20 - Le suppression de la ligne de PRODUIT est refusée

Tentons une petite manipulation : sélectionnons la ligne 30185 de la table COMMANDE puis supprimons-la via le bouton **Delete the current row**. Qu'observe-t-on dans les fenêtres de données ?

– la ligne 30185 a disparu de la table COMMANDE,

– ... mais également les trois lignes de la table DETAIL dont NCOM = 30185.

Nous aurons dans les tutoriels ultérieurs de SQLfast l'occasion d'étudier plus avant le comportement des SGBD relationnels lors des modifications de données.

9. Interroger une base de données par une requête SQL

Jusqu'à présent, nous avons manipulé les données sans faire appel au langage SQL. Ces manipulations sont simples à réaliser mais sont relativement limitées.

Nous allons maintenant écrire nos premières requêtes SQL d'extraction de données. Nous supposons bien sûr que la base de données (en l'occurrence

CLICOM.db) est ouverte. Il peut être utile d'afficher la fenêtre du schéma pour éviter les erreurs dans les noms des tables et des colonnes.

Recherchons dans la table CLIENT les lignes dont la colonne LOCALITE a la valeur 'Poitiers'. La formulation en SQL est simple et intuitive :

```
select * from CLIENT where LOCALITE = 'Poitiers';
```

On pourrait écrire cette requête de manière plus claire, en trois lignes (attention, le signe "**;**" n'apparaît qu'à la fin de la dernière ligne) :

```
select *
from   CLIENT
where  LOCALITE = 'Poitiers';
```

Introduisons cette requête, en une ou plusieurs lignes, dans la zone de texte, dite *zone de script*, de la fenêtre principale (Figure 21).

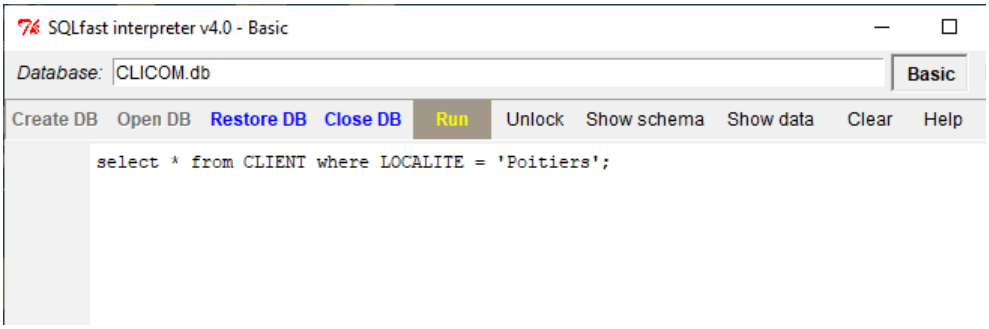


Figure 21 - Une requête SQL est introduite dans la zone de script de la fenêtre principale

Pour exécuter cette requête, nous cliquons sur le bouton **Run**. Le résultat s'affiche dans le fenêtre de sortie (figure 22). Nous savons désormais qui sont les trois clients habitant à Poitiers !

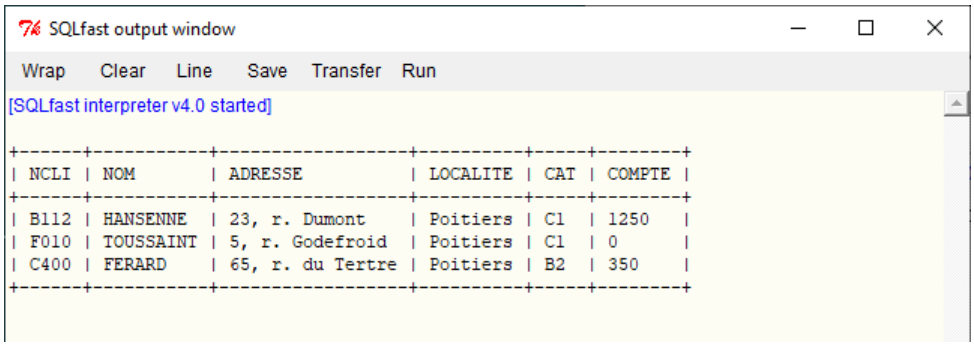


Figure 22 - Le résultat de l'exécution de la requête apparaît dans la fenêtre de sortie

La requête s'est exécutée sans erreur, ce qui se remarque notamment par la couleur bleue dans laquelle elle apparaît (figure 23). Ainsi marquée, cette requête sera ignorée lors des demandes d'exécution suivantes.

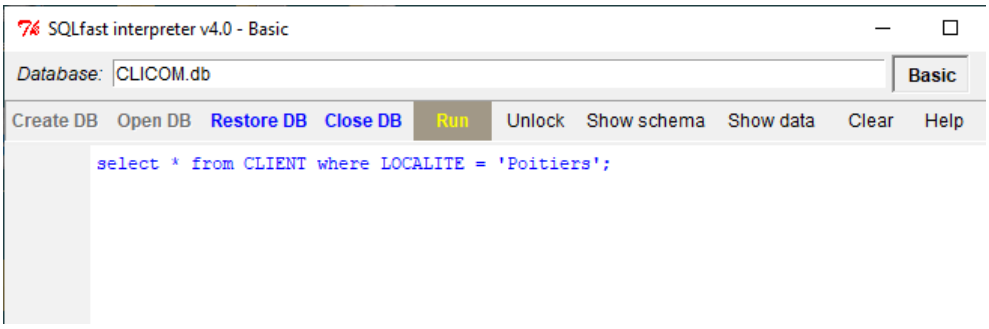


Figure 23 - L'exécution ayant réussi, la requête est colorée en bleu

10. Interroger une base de données. Détection des erreurs

A présent, introduisons une requête comportant une erreur manifeste (figure 24) : une faute de frappe conduit à une erreur dans le nom de la table COMMANDE.

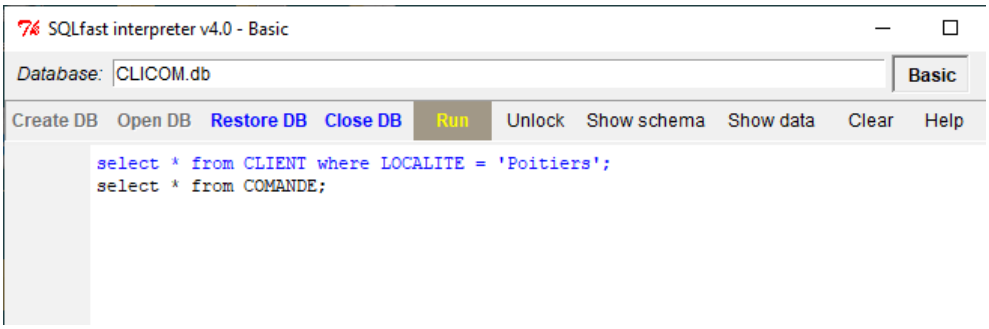


Figure 24 - Une deuxième requête, erronée, est introduite dans la fenêtre de script

Sa tentative d'exécution via le bouton **Run** provoque plusieurs réactions (figure 25) :

– un message apparaît dans une petite boîte d'alerte, de titre SQLfast ERROR. Ce message indique la détection d'une erreur, sa description (no such table: COMANDE), la ligne ayant provoqué l'erreur (select * from COMANDE) et l'endroit où se trouve cette ligne (local script, suivi du numéro de ligne [1]). Ces détails peuvent sans doute paraître excessif pour une erreur évidente dans une simple requête mais ils prendront tout leur sens dans des cas plus complexes. On clique sur l'un de ses boutons pour fermer cette boîte de message.

– le message d'erreur est également copié dans la fenêtre de sortie.

– la requête fautive apparaît en rouge dans la fenêtre du script.

Nous corrigeons l'erreur en rectifiant le nom de la table et nous en demandons l'exécution. Cette fois, l'exécution réussit. La deuxière requête est colorée en bleu et le résultat apparaît dans la zone de sortie (figure 26).

Les requêtes qui apparaissent en bleu ne sont plus exécutables. Elles sont conservées à titre d'historique jusqu'au moment où on les efface.

Il est cependant possible de les *ramener à la vie* via le bouton **Unlock**. Celui-ci rend toutes les requêtes de la zone de script à nouveau exécutable. Elles apparaissent à nouveau en noir.

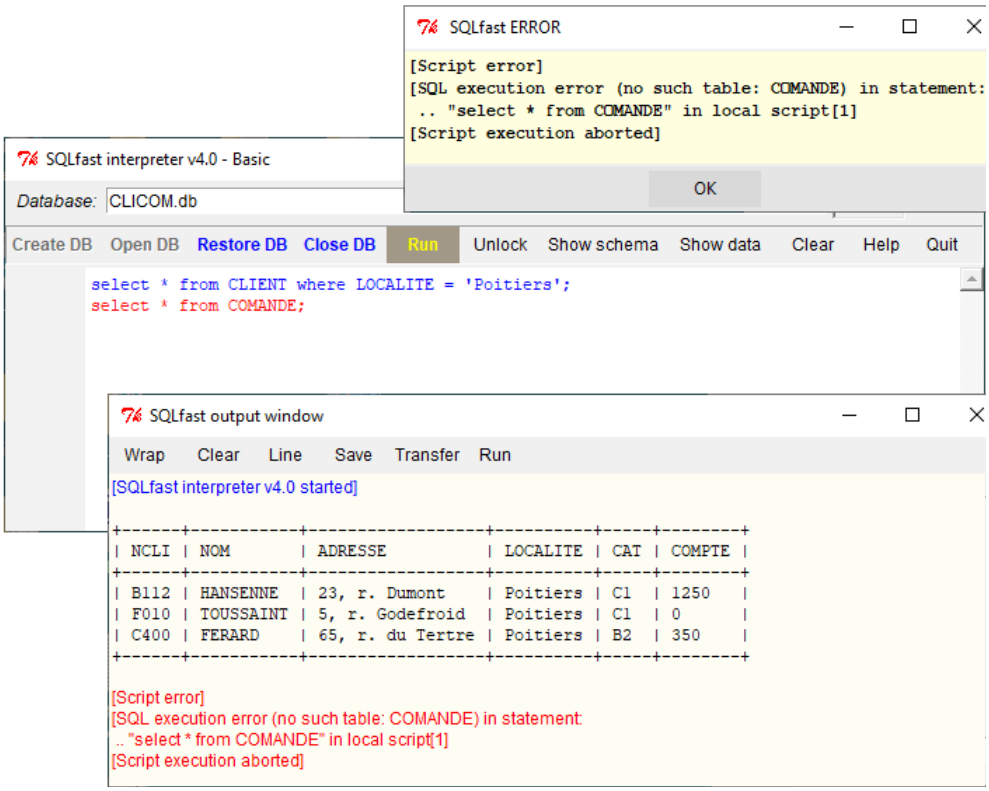


Figure 25 - L'exécution de la requête erronée provoque de vives réactions !

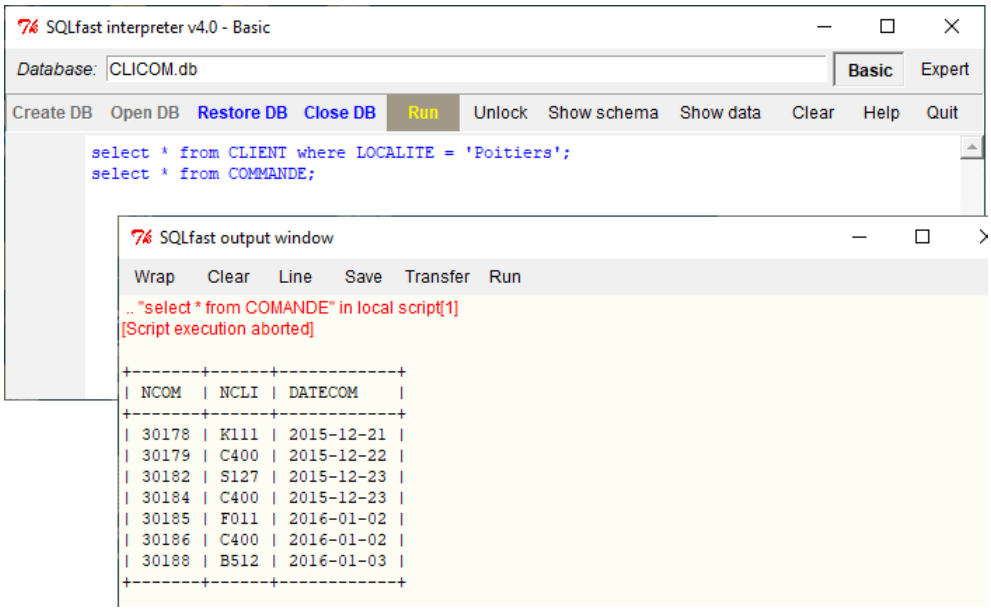


Figure 26 - La requête est corrigée et son exécution réussit

11. Fermer la base de données courante (bouton *Close DB*)

Lorsque le travail sur la base de données courante est terminé, nous la fermons en cliquant sur le bouton bleu **Close DB**. Son nom disparaît du champ *Database* de la fenêtre principale et les boutons agissant sur la base de données courante sont désactivés.

12. Modifier le contenu de la base de données courante via SQL

Ouvrons à nouveau la base de données "CLICOM.db" et affichons le contenu de ses quatre tables (bouton **Show data**). Nous allons effectuer quelques modifications élémentaires des données mais en utilisant cette fois les requêtes SQL.

Expérience n° 1

Introduisons dans la zone de script la requête suivante :

```
delete from CLIENT where NCLI = 'C123';
```

Nous demandons de la sorte la suppression des données du client 'MERCIER' de 'Namur'. Lorsque nous exécutons cette requête, nous observons la disparition immédiate de cette ligne dans la fenêtre de données de la table CLIENT.

Expérience n° 2

Introduisons à la suite de la première la requête suivante :

```
delete from CLIENT where NCLI = 'C400';
```

Cette fois, SQLfast refuse d'obéir, arguant de la violation d'une contrainte référentielle ('FOREIGN KEY constraint'). Lors de la création de la table COMMANDE, la clé étrangère NCLI de la table COMMANDE a effectivement été déclarée "on delete no action", de sorte qu'il est interdit de supprimer une ligne de CLIENT qui est encore référencée par des lignes de COMMANDE. Avant de supprimer une telle ligne de CLIENT, nous devons supprimer toutes les lignes de COMMANDE qui la référencent.

Expérience n° 3

Effaçons le contenu de la zone de script (bouton **Clear**) et introduisons la requête suivante :

```
delete from COMMANDE where NCOM = '30185';
```

En examinant attentivement le contenu de la table COMMANDE, nous observons, non seulement que la ligne '30185' a effectivement disparu, mais aussi que les trois lignes de DETAIL dépendant de cette dernière ont aussi disparu.

Cette suppression en cascade est en accord avec la déclaration de la clé étrangère NCOM de la table DETAIL : on delete cascade : lorsqu'une ligne de COMMANDE est supprimée, toutes les lignes de DETAIL qui la référençaient sont également supprimées "en *cascade*".

Expérience n° 4

Introduisons enfin la requête suivante, qui demande de modifier le numéro du client 'C400' en 'C999' :

```
update CLIENT set NCLI = 'C999' where NCLI = 'C400';
```

Le changement a été effectué mais nous observons aussi que les trois lignes de COMMANDE dépendant de ce client ont été ajustées en conséquence. Cette modification *en cascade* est conforme à la déclaration de la clé étrangère NCLI de la table COMMANDE : on update cascade.

13. IMPORTANT : restaurer la base de données courante (bouton *Restore DB*)

Après avoir exécuté ces requêtes de modification, nous avons laissé la base de données dans un état, disons, ... *dégradé* (dans la *vraie vie*, on ne joue pas avec les données !)

En cliquant sur le bouton bleu **Restore DB**, nous annulons l'effet de ces modifications. L'état des données est ainsi celui qui existait au moment de l'ouverture de la base de données, comme nous pouvons l'observer dans les fenêtre de données.

Cette manoeuvre utilise la copie de sauvegarde que SQLfast avait créée au moment de l'ouverture de la base de données.

Si, par mégarde, nous oublions de restaurer la base de données CLICOM.db au moment de sa fermeture, pas de panique :

– nous supprimons manuellement la base de données CLICOM.db du répertoire *Databases* et nous la remplaçons par la copie de sauvegarde CLICOM.db.bak dont nous effaçons le suffixe '**.bak'**.
– en dernier recours, nous pouvons retirer manuellement la version d'origine de l'archive zip SourceDB.zip, présente dans le répertoire Databases.

14. Effacer le contenu d'une fenêtre

La fenêtre principale et la fenêtre de sortie disposent chacune d'un bouton **Clear**, dont l'effet est d'effacer le contenu des zones de texte.

Il est à noter que ces deux zones de texte sont éditables, de sorte qu'il est également possible de supprimer, modifier, extraire, introduire une partie de leur contenu de la manière habituelle : sélectionner une partie du texte puis **^C**, **^X** et **^V**.

15. Les boutons de la fenêtre de sortie

La fenêtre de sortie dispose d'une série de boutons dont l'utilité apparaîtra surtout lors de l'exécution de requêtes plus complexes. Nous en noterons trois :

– **bouton Wrap** : définit le comportement de la fenêtre lorsque les lignes affichées sont plus longues que la largeur de la fenêtre. Une ligne trop longue peut être soit cachée (on utilise alors la barre de défilement pour la visualiser), soit coupée pour faire apparaître la suite à la ligne suivante. Ce bouton change le mode d'affichage de ces lignes.
– **bouton Clear** : efface le contenu de la fenêtre.
– **bouton Line** : ajoute une ligne blanche au texte de la fenêtre.

16. Les documents d'aide et les tutoriels

SQLfast comporte un gestionnaire d'aide permettant d'utiliser et de développer des documents d'aide et des tutoriels. Ces documents sont rédigés dans un langage de mise en page permettant de spécifier les titres, les styles, les figures et la navigation au sein du document et entre documents.

Un document peut comporter des requêtes SQL et des fragments de script SQLfast, lesquels vont jouer un rôle important. Un exemple :

```
select NCLI,NOM from CLIENT where LOCALITE = 'Toulouse';
```

Les documents d'aide et les tutoriels ont la même structure. Ils ne se distinguent que par l'extension de leurs fichiers sources : '**.help**' pour l'aide et '**.tuto**' pour les tutoriels, extension qui permet simplement de les classer.

16.1 Comment accéder aux tutoriels ?

L'accès aux documents d'aide et tutoriels s'effectue via le bouton **Help**. Un petit menu s'ouvre, constitué de quatre items (figure 27) :

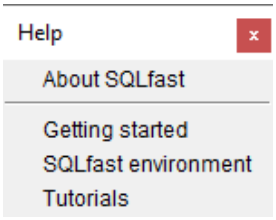


Figure 27 - Le menu **Help** du niveau Basic

– **About SQLfast** : la *carte d'identité* de SQLfast. Cette boîte permet entre autres choses de passer d'une langue à l'autre pour les tutoriels et documents d'aide.
– **Getting started** : le présent document d'aide au démarrage au niveau **Basic**
– **SQLfast environment** : description détaillée de l'interface et des fonctions du niveau **Basic**.
– **Tutorials** : accès à la liste des tutoriels spécifiques au niveau **Basic**.

16.2 Navigation dans un tutoriel

Ouvrons le tutoriel de nom **1._Requetes_elementaires.tuto**. A la suite de l'introduction et des recommandations d'utilisation, le tutorial comprend une première partie, intitulée **Première partie : les énoncés** (figure 28).
La première section de cette partie, intitulée **1. Requêtes élémentaires**, propose l'exercice **E1** dont l'énoncé est formulé ainsi :

- Afficher le numéro, le nom et la localité des clients

Le lecteur est invité à rédiger, dans la zone de script, une requête répondant à cet énoncé, à en demander l'exécution (bouton **Run**), puis à vérifier que le résultat obtenu dans la fenêtre de sortie est bien équivalent au tableau de données qui suit cet énoncé.

En cas de doute, ou s'il ne trouve pas de solution satisfaisante, le lecteur peut consulter la solution proposée dans la seconde partie du tutoriel.

Pour ce faire, il n'est pas nécessaire de parcourir laborieusement le document, mais il suffit d'utiliser la fonction d'*hypertexte* du tutoriel : en double-cliquant sur une partie de l'énoncé, la rubrique *solution* de l'exercice apparaît instantanément dans la fenêtre (figure 29).

Le lecteur peut ainsi comparer sa solution avec celle du tutoriel, par exemple pour vérifier que cette dernière est bien correcte !

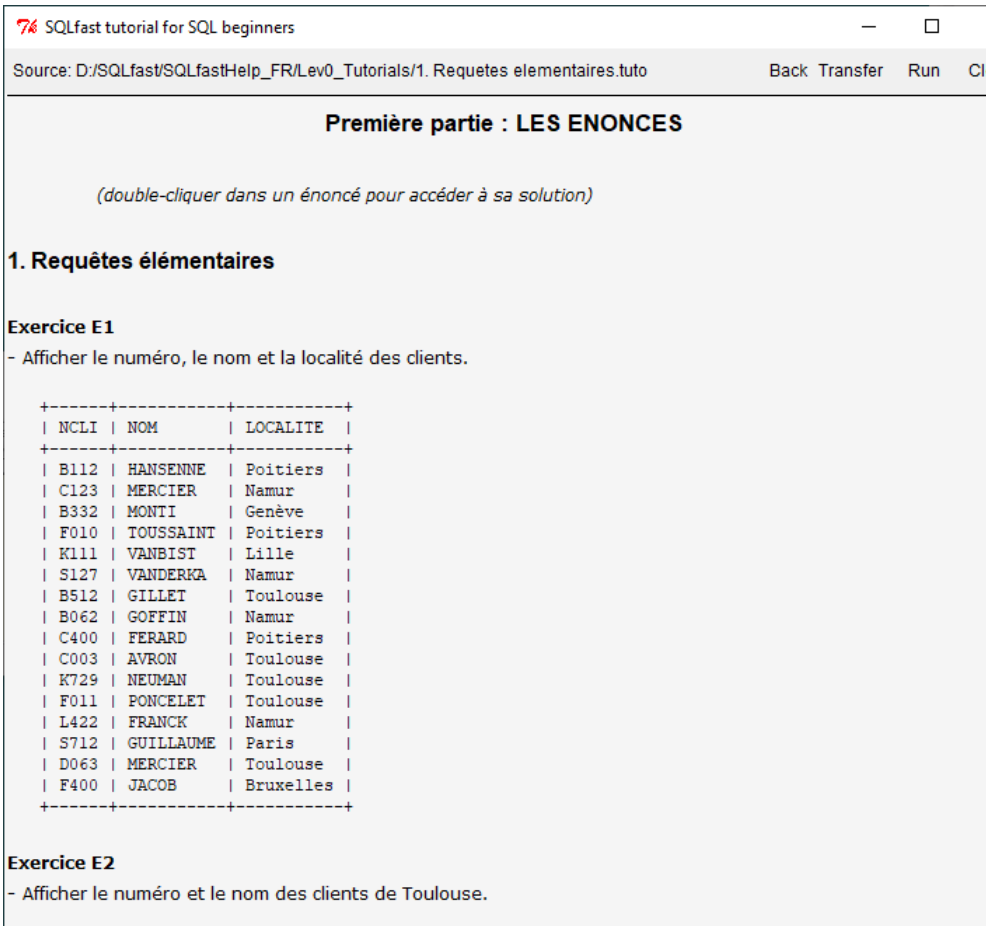


Figure 28 - Extrait de la première partie du tutoriel "1._Requetes_elementaires.tuto" : premier exercice de la section 7.1 du chapitre 7

Le mécanisme d'hypertexte est particulièrement simple : double-cliquer dans un paragraphe amène au centre de la fenêtre le prochain fragment de texte qui lui est identique.

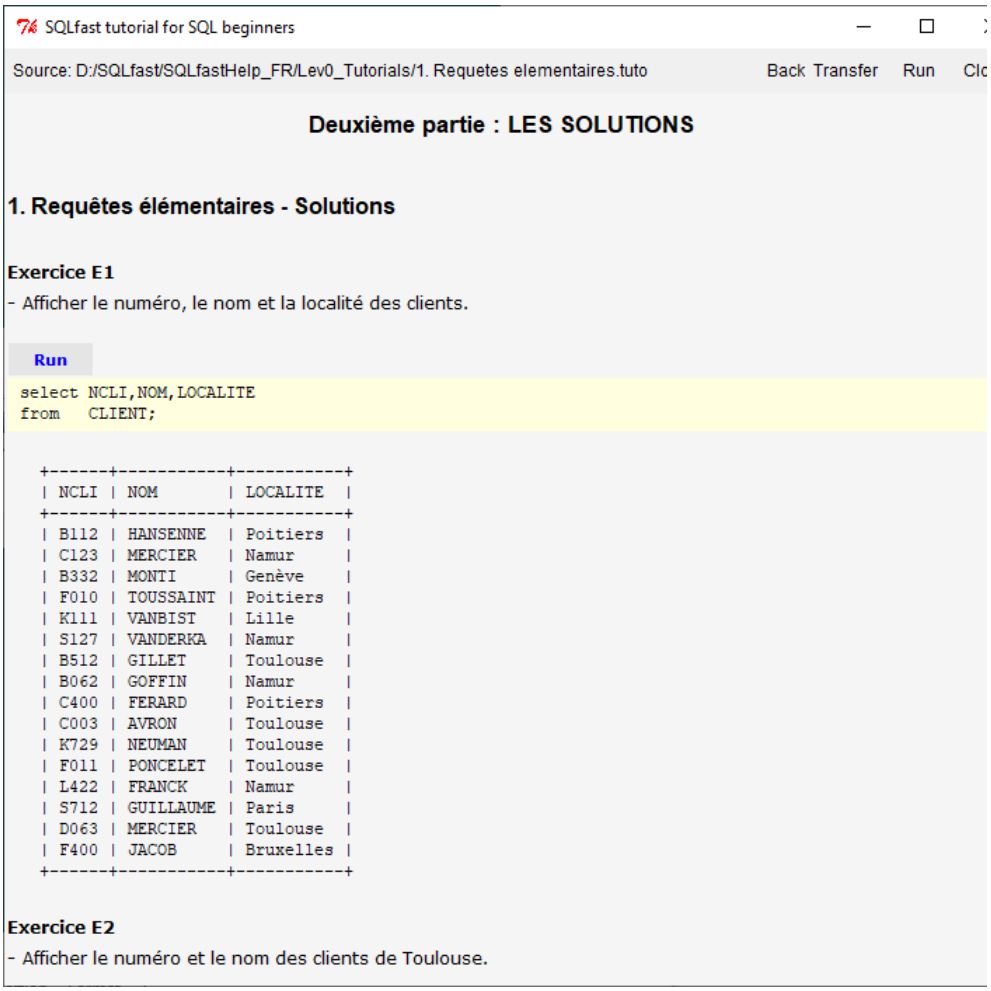


Figure 29 - ... plus loin dans le tutoriel : la solution de l'exercice

Une autre application simple très pratique : double-cliquer dans une ligne de la table des matières située au début de ce document renvoie instantanément à la rubrique de même nom.

La structure d'un tutoriel en deux partie n'est évidemment pas obligatoire, le rédacteur étant libre de l'organiser selon son idée. On pourraît d'ailleurs, pour des exercices ou des applications plus complexes, adopter une structure en quatre parties : Rappel des concepts, Enoncés, Premiers indices et Solutions.

16.3 Navigation entre tutoriels

Un tutoriel peut renvoyer vers un autre tutoriel qui le suit, le complète ou l'approfondit. Ce *renvoi* apparaît sous la forme d'une ligne en bleu comme celle-ci :

> Les requêtes SQL élémentaires

Cette ligne est un lien qui permet d'accéder à un autre document tout comme les liens d'une page web. En cliquant sur ce lien, le document référencé apparaît dans la fenêtre.

Pour revenir au document précédent, il suffit de cliquer sur le bouton **Back** dans l'entête de la fenêtre courante.

16.4 Exécuter un fragment d'un tutoriel

Un tutoriel SQLfast (ou un document d'aide, tel que le présent document) n'est pas qu'un simple texte passif que l'utilisateur est invité à lire comme le serait un document pdf.

Montrons-le par un premier exemple élémentaire. On vérifie que la base de données CLICOM.db est ouverte ou on l'ouvre si nécessaire. Considérons ensuite la ligne suivante :

```
select * from CLIENT;
```

Nous pourrions l'exécuter simplement en la copiant (sélection puis ^c), en la collant dans la zone de script (bouton **Clear** puis ^v) puis en l'exécutant (bouton **Run**).

Essayons autre chose : sélectionnons tout ou une partie de cette ligne (pour surligner le mot **from** par exemple) puis cliquons sur le bouton **Transfer** situé en haut à droite de la présente fenêtre.

Résultat : la ligne complète a été copiée dans la zone de script. Il ne reste plus qu'à l'exécuter via le bouton **Run** de la fenêtre principale.

Il est cependant possible de faire mieux ! Sélectionnons la ligne de la requête comme nous l'avons fait ci-dessus puis cliquons sur le bouton **Run** en haut à droite de la présente fenêtre.

Résultat : la requête a tout simplement été exécutée, non pas à partir de la zone de script de la fenêtre principale mais directement à partir du tutoriel !

Essayons un exemple plus complet. La suite de requêtes ci-dessous crée une table (temporaire) de nom PRODUIT_ETHIQUE puis y insère trois lignes de données :

```
create temporary table PRODUIT_ETHIQUE(
            NPRO char(15),
            LIBELLE char(60),
            PRIX decimal(6));
insert into PRODUIT_ETHIQUE values('BC99',
                                   'Barre chocolatée',1.2);
insert into PRODUIT_ETHIQUE values('S099','Soda',1.5);
```

```
insert into PRODUIT_ETHIQUE values('AD99',
                                   'Assurance décès',125);
```

Plutôt que de les recopier dans la zone de script, sélectionnons ces sept lignes comme illustré à la figure 30. Il n'est pas nécessaire que la première et la dernière lignes soient entièrement sélectionnées.

Nous cliquons sur le bouton **RUN** de la présente fenêtre. Rien ne se passe en apparence bien sûr, mais si nous introduisons dans la zone de script la requête suivante (ou si nous l'exécutons à partir de ce document) :

```
select * from PRODUIT_ETHIQUE;
```

... nous constatons que la table a effectivement été créée et qu'elle contient bien les trois lignes de données prévues (figure 31).

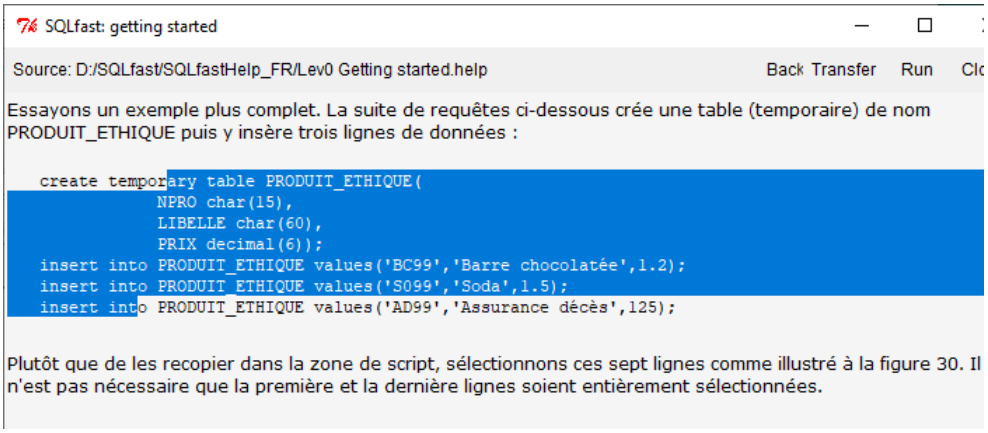


Figure 30 - Suite de requêtes SQL sélectionnée dans la présente fenêtre. En tête de la fenêtre, les boutons **Transfer** et **Run**.

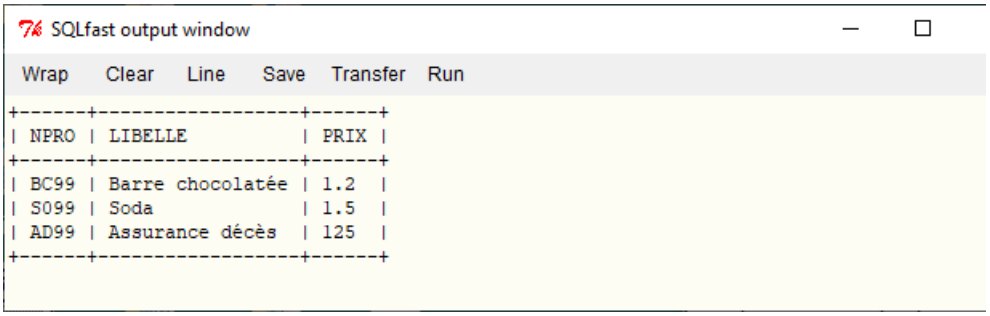


Figure 31 - La preuve que la table PRODUIT_ETHIQUE a bien été créée !

Si la base de données n'a pas subi d'autres outrages que la création de cette table, nous pouvons la fermer sans la restaurer. Le propre d'une table temporaire est en effet de disparaître lors de la fermeture de la base de données.

16.5 Exécuter un script à partir d'un tutoriel

Il existe cependant une manière beaucoup plus élégante d'insérer un petit script exécutable dans un tutoriel, via les **embedded scripts**. Ainsi, la création de la table PRODUIT_ETHIQUE peut être présentée comme ceci :

Run

```
create temporary table PRODUIT_ETHIQUE(
            NPRO char(15),
            LIBELLE char(60),
            PRIX decimal(6));
insert into PRODUIT_ETHIQUE values('BC99','Barre chocolatée',1.2);
insert into PRODUIT_ETHIQUE values('S099','Soda',1.5);
insert into PRODUIT_ETHIQUE values('AD99','Assurance décès',125);
```

Pour exécuter cette requête, il suffit de cliquer sur le bouton **Run** qui apparaît au dessus du script. Auparavant nous aurons restauré, via le bouton **Restore DB**, la base de données dans son état initial: à ce stade en effet, elle contient déjà une table nommée PRODUIT_ETHIQUE.

Si la base de données n'a pas subi d'autres outrages que la création de cette table, nous pouvons la fermer sans la restaurer. Le propre d'une table temporaire est en effet de disparaître lors de la fermeture de la base de données.

17. Comment continuer ?

L'un des premiers objectifs du niveau **BASIC** est de permettre à l'utilisateur de se familiariser avec les concepts des bases de données et d'apprendre les bases du langage SQL.

Une série de tutoriels associés aux chapitres 6, 7, 8 et 9 de l'ouvrage de référence sont mis à disposition pour démarrer cet apprentissage. Ils sont accessibles à partir du bouton **Help**, sous l'intitulé **Tutorials**. La liste des tutoriels est évolutive. Ceux-ci peuvent être modifiés et enrichis selon les besoins. Le code SQL de ces chapitres et de leurs annexes est également disponible dans le répertoire **Scripts/BD-2018** sous la forme de scripts SQLfast prêts à exécuter. Pour exécuter un fichier de script, cependant, SQLfast doit être en mode **Expert**.

Pour commencer en douceur, on suggère de travailler la première leçon : ***Les requêtes SQL élémentaires***.

La plupart des tutoriels exploitent la base de données CLICOM.db, dont le schéma et le contenu sont ceux de l'ouvrage de référence. En cas d'*accident,* ou oubli de restauration, on pourra restaurer la base de données à partir de l'archive SourceDB.zip du répertoire Databases. Cette archive contient aussi diverses bases de

données plus spécialisées utilisées par les tutoriels. Ces bases de données peuvent être recréées via les tutoriels ou simplement extraites de leur archive.

Pour plus de précisions concernant l'interface graphique SQLfast, on consultera de document d'aide ***SQLfast environment*** (en français malgré son titre !) disponible via le bouton **Help**.