***Date de dernière modification** : 15/6/2015*

Annexe **28**

Introduction à SQLfast

Remarque
Cette annexe décrit la version *beta* de l'interface graphique de SQLfast disponible en juin 2015. Cette interface a été considérablement remaniée depuis cette date, présentant par exemple deux niveaux d'interaction avec les bases de données : *Learning SQL* et *Learning SQLfast*. En outre, les matériaux de cette section ont été complétés et traduits sous la forme de documents d'aide et de tutoriels intégrés à l'interface. Ce sont ces documents, ainsi que les informations disponibles sur le site web de SQLfast qui constituent la description la plus récente de l'environnement et du langage.
Le texte ci-après est cependant conservé dans sa version d'origine.

A28.1L'ENVIRONNEMENT SQLfast

<à rédiger>

A28.2L'INTERFACE GRAPHIQUE SQLfast

Une version intégrée de ce texte est également disponible dans le menu Help>SQLfast reference>SQLfast environment.

Le nom SQLfast désigne à la fois un langage et un logiciel. Le ***langage SQLfast*** est une extension du langage SQL qui permet de développer simplement et rapidement des scripts interactifs utilisant une base de données. Le ***logiciel SQLfast*** comporte principalement le *moteur SQLfast*, qui exécute les scripts qui lui sont soumis, et une *interface graphique* qui permet d'écrire, manipuler, gérer et exécuter des scripts SQLfast ainsi que de visualiser et traiter les résultats produits par ces scripts.

A28.2.1 L'interface SQLfast

Le premier composant de l'interface graphique est la *fenêtre de commande*. Cette fenêtre comporte quatre parties superposées (figure A28.1). Dans l'ordre, la barre de menu, les répertoires et fichiers courants, la barre des boutons de gestion et la fenêtre de script.

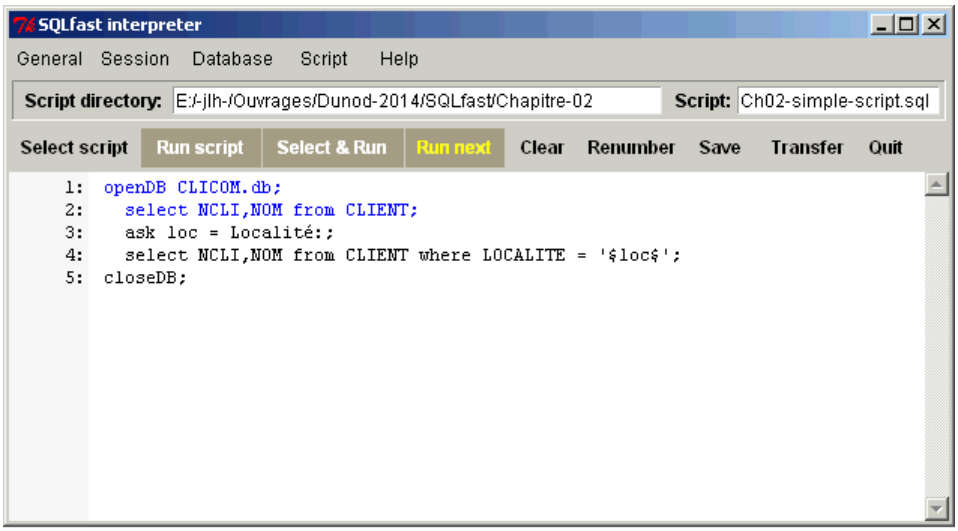


Figure A28.1 - La fenêtre de commande de l'interface SQLfast

a) Barre de menu

General

– **Open file** : ouverture d'un fichier texte de nature quelconque et chargement dans la fenêtre de script. Ce fichier ne contient pas nécessairement un script.

– **Save file** : copie du contenu de la fenêtre de texte dans un fichier.

– **Open SQLfast.ini** : ouverture du fichier de paramètres **SQLfast.ini** et chargement dans la fenêtre de script. Version spécialisée de l'action "**General>Open file**". Après modification, le texte peut être sauvé par l'action "**General>Save file**".

– **Options** : modification des paramètres de l'interface de SQLfast

– **Quit** : quitter SQLfast

Session

– **Start session** : Démarrer une nouvelle session. L'état de SQLfast est celui qu'il avait lors de son démarrage. Toute exécution d'un script en dehors d'une session ouvre automatiquement une nouvelle session.

– **Close session** : Clôture de la session courante. Toute base de données encore ouverte est fermée. Les variables SQLfast sont supprimées. Une nouvelle session peut être démarrée.

Database

– **Show DB schema** : afficher dans une fenêtre de schéma le schéma d'une base de données.

– **Show DB data** : afficher le contenu d'une base de données dans une série de fenêtres (une fenêtre par table).

– **Close database** : fermeture de la base de données actuellement ouverte. Equivalent à l'exécution de l'instruction SQLfast **`closeDB`**.

– **Delete database** : supprimer une base de données. Equivalent à l'exécution de l'instruction SQLfast **`deleteDB`**.

– **Copy database** : effectuer une copie d'une base de données.

Script

– **Select script** : sélection d'un script à partir du répertoire des scripts et chargement dans la fenêtre de script.

– **Run local script** : exécuter le script local, c'est-à-dire le contenu complet de la fenêtre de script, quelle soit la couleur de ces instructions. Si le script n'inclut pas l'instruction **`closeDB`** de fermeture de la base de données, celle-ci sera automatiquement fermée si la valeur du paramètre **autoCloseDB** est **`'script'`** ou **`'mainscript'`** (valeur par défaut) mais elle sera laissée ouverte si cette valeur est **`'sqlfast'`**. Elle sera fermée soit par l'action **Database>Close database**, soit par l'exécution d'un script qui inclut cette instruction de fermeture, soit par la sortie de SQLfact (bouton **Quit**). Si une des instructions a provoqué une erreur, elle apparaît en **ROUGE** dans le fenêtre de script.

– **Select & run script** : sélectionner un script à partir du répertoire des scripts et l'exécuter. Le contenu de la fenêtre de script n'est pas modifié.

– **Run next statements** : exécuter les *nouvelles instructions* de la fenêtre de script. Ces nouvelles instructions sont écrites en **NOIR** alors que celles qui ont déjà été exécutées sont écrites en **BLEU**. La base de données reste ouverte à la terminaison de l'exécution des nouvelles instructions. On peut donc exécuter pas à pas des fragments de script successifs sur une base de données ouverte. Ce mode est particulièrement dédié à l'initiation à SQL et à l'expérimentation sans passer par l'usage de scripts enregistrés. La mise en bleu des instructions anciennes permet de garder la trace les opérations déjà exécutées avec succès. Si une des instructions a provoqué une erreur, elle apparaît en **ROUGE**.

– **Clear script** : efface le contenu de la fenêtre de script.

– **Renumber lines** : assigne des numéros de séquence aux lignes de la fenêtre de script. Utile pour repérer les instructions ayant provoqué une erreur. Cette numérotation est automatique lors d'un **Select script**, d'un **General>Open file**, d'un

General>Open SQLfast.ini ou lors du transfert d'un texte de la fenêtre de sortie vers la fenêtre de script. La numérotation doit être demandée lors d'une saisie manuelle.

– **Save script** : sauvegarder le contenu de la fenêtre de script dans un fichier de script.

– **Transfer script to output** : copier le contenu de la fenêtre de script dans la fenêtre de sortie.

Help

– **About SQLfast** : informations générales sur SQLfast.

– **Getting started** : bref tutoriel introductif.

– **Survival guide** : description des informations les plus utiles pour rédiger et exécuter un script SQLfast.

– **SQLfast reference** : série de documents de référence relatifs à SQLfast.
  – **SQLfast environment** : description de l'interface interactive de SQLfast.
  – **SQLfast commands** : description de l'interface interactive de SQLfast.
  – **Help mini-language** : brève description du langage de rédaction d'un document Help.

– **SQLfast tutorial** : série de tutoriels relatifs au langage SQLfast.

Remarque : chaque item de menu peut être détaché pour former une barre d'outils flottante (figure A28.2).

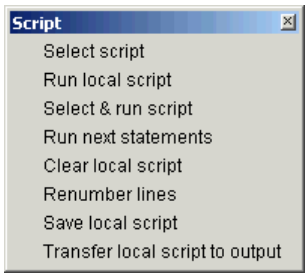


Figure A28.2 - Menu détachable

b) Répertoires et fichiers

Cette partie indique les répertoires et fichiers courants:

– **DB directory** : répertoire courant des des bases de données. Par défaut, ce répertoire est celui que spécifie le paramètre **dbDirectory** du fichier **SQLfast.ini**. Peut être modifié suite à la sélection d'une base de données.

– **Database** : nom local de la base de données sélectionnée.

– **Script directory** : répertoire courant des scripts. Par défaut, ce répertoire est celui que spécifie le paramètre **scriptDirectory** du fichier **SQLfast.ini**. Peut être modifié suite à la sélection d'un script (par **Script>Select script**).

– **Script** : nom local du script sélectionné.

c) Boutons de gestion des scripts

Ces boutons reprennent certaines fonctions importantes du menu. Les boutons de couleur foncée commandent l'**exécution** des instructions SQLfast, le dernier (**Run next**, texte en jaune) étant spécialement dédié à l'initiation et à l'expérimentation.

– **Select script** : sélection d'un script à partir du répertoire des scripts et chargement dans la fenêtre de script. Equivalent à **Script>Select script**.

– **Run script** : exécuter le script local, c'est-à-dire le contenu complet de la fenêtre de script, quelle soit la couleur de ces instructions. Equivalent à l'action **Script>Run local script**.

– **Select & Run** : sélectionner un script à partir du répertoire des scripts et l'exécuter. Le contenu de la fenêtre de script n'est pas modifié. Equivalent à **Script>Select & run script**.

– **Run next** : exécuter les nouvelles instructions de la fenêtre de script. Equivalent à **Script>Run next statements**

– **Clear** : effacer le contenu de la fenêtre de script. Equivalent à **Script>Clear local script**.

– **Renumber** : assigner des numéros de séquence aux lignes de la fenêtre de script. Equivalent à **Script>Renumber lines**.

– **Save** : sauvegarder le contenu complet de la fenêtre de script dans un fichier de script. Equivalent à **Script>Save local script**.

– **Transfer** : copier le contenu complet de la fenêtre de script dans la fenêtre de sortie. Equivalent à **Script>Transfer script to output**.

– **Quit** : quitter SQLfast. Equivalent à **General>Quit**.

d) Fenêtre de script local

Cette fenêtre peut accueillir une séquence d'instructions SQLfast, mais plus généralement un texte quelconque. Ce texte peut être introduit par l'utilisateur via le clavier, par copier/coller, par le bouton **Select script** (ou l'action **Script>Select script**), par l'action **General>Open file** ou **General>Open file**, ou encore par l'action **Transfer** à partir de la fenêtre de sortie. Il peut être modifié et sauvé dans un fichier. Il peut être exécuté par les boutons **Run script** et **Run next** et les actions équivalentes des menus.

A28.2.2 Le support de sortie

Le canal de sortie est le support sur lequel les scripts écrivent leurs résultats et sur lequel les messages de SQLfast apparaissent (ces messages constituent le journal des opérations, ou "log"). Les deux principaux supports sont la fenêtre de sortie SQLfast (figure A28.3) et la console Python. Cette dernière sera plus rarement utilisée, sauf éventuellement pour accueillir les messages SQLfast.

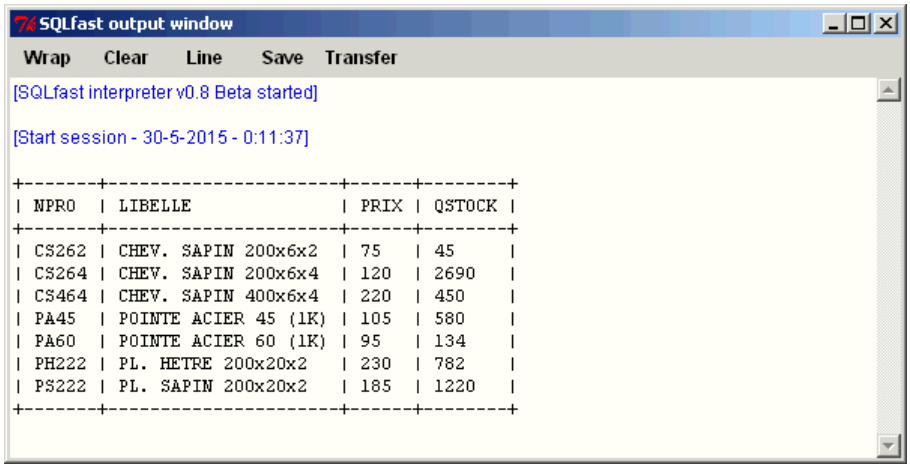


Figure A28.3 - La fenêtre de sortie SQLfast

La fenêtre de sortie comporte deux parties superposées. Dans l'ordre :

a) Boutons de gestion du contenu de la fenêtre

– **Wrap** : une ligne trop longue pour la largeur de la fenêtre peut être soit cachée (utiliser alors la barre de déroulement pour la visualiser), soit coupée pour faire apparaître la suite à la ligne suivante. Ce bouton change le mode d'affichage de ces lignes.

– **Clear** : effacer le contenu de la fenêtre de sortie.

– **Line** : ajouter une ligne blanche au contenu de la fenêtre.

– **Save** : sauvegarder le contenu de la fenêtre de sortie dans un fichier de texte.

– **Transfer** : copier le contenu de la fenêtre de sortie dans la fenêtre de script, après avoir effacé le contenu de cette dernière. Utile lorsque le script exécuté génère lui-même des scripts SQLfast.

b) Fenêtre de sortie proprement dite

C'est dans cette fenêtre que le moteur SQLfast écrit le résultat de l'exécution des scripts. Outre ces résultats, qui apparaissent dans la police Courier 9, de couleur **noire**, on y trouvera, dans la police Helvetica 9 :

– en **bleu**, les messages du journal des opérations.
– en **rouge**, les messages décrivant les erreurs survenant lors de l'exécution d'un script SQLfast.
– en **vert**, les messages d'avertissement n'interrompant pas l'exécution d'un script SQLfast.

Remarque

Le contenu de la fenêtre de sortie est éditable par l'utilisateur et sauvable, tout comme la fenêtre de script.

A28.2.3 3. La fenêtre de schéma

La fenêtre de schéma contient le schéma d'une base de données (figure A28.4). Elle constitue une documentation utile lors de la rédaction de scripts. Cette fenêtre s'obtient par la commande de menu **Database>Show DB schema**.

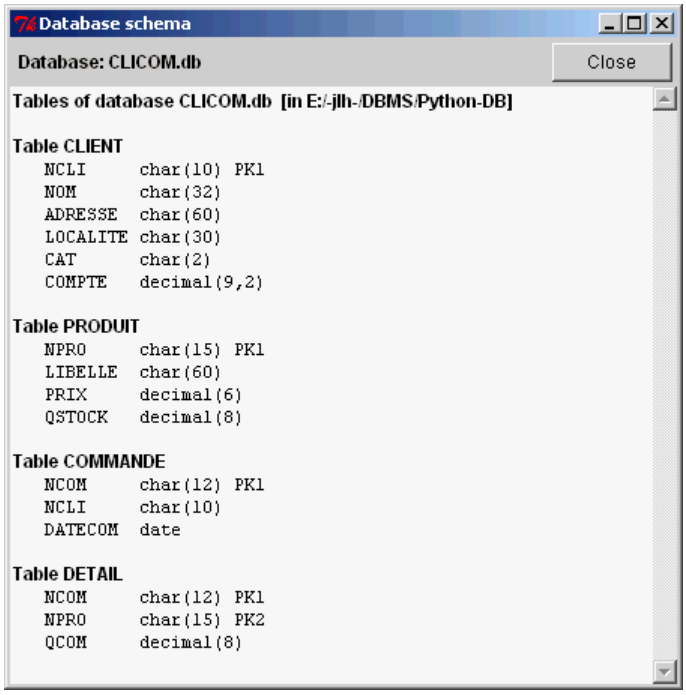
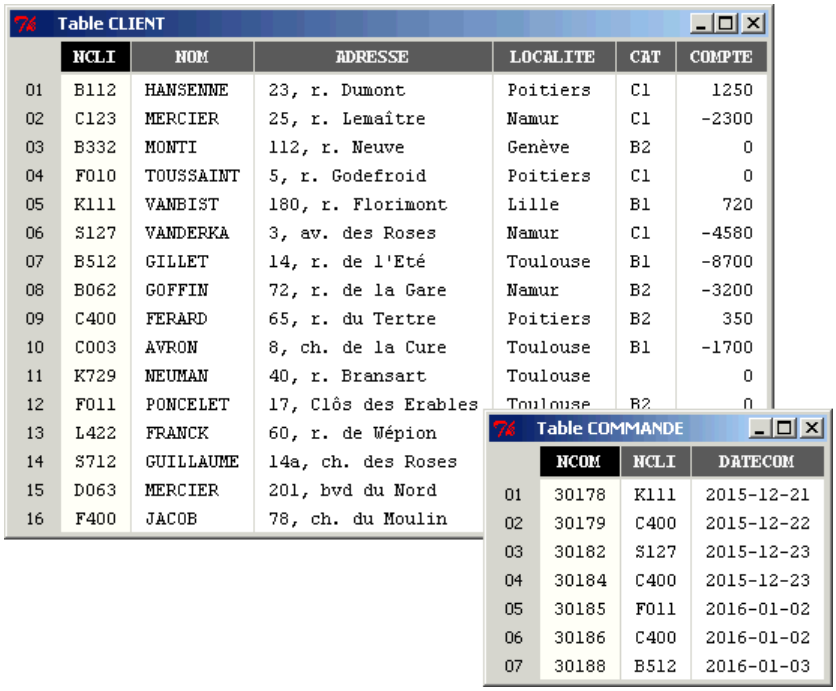


Figure A28.4 - La fenêtre du schéma CLICOM.db

A28.2.4 4. Les fenêtres de données

Les fenêtres de données visualisent chacune le contenu d'une table d'une base de données (figure A28.5). Elle permettent de vérifier le résultat de l'exécution de script ou de suivre l'évolution des données dans les opérations de modification. Ces fenêtres s'obtiennent par la commande de menu **Database>Show DB data**.



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

	NCOM	NCLI	DATECOM
01	30178	K111	2015-12-21
02	30179	C400	2015-12-22
03	30182	S127	2015-12-23
04	30184	C400	2015-12-23
05	30185	F011	2016-01-02
06	30186	C400	2016-01-02
07	30188	B512	2016-01-03

Figure A28.5 - Deux fenêtres de données de la base de données CLICOM.db

A28.3INSTALLER SQLfast

<à rédiger>

A28.4RÉDIGER ET EXÉCUTER UN SCRIPT SQLfast

<à rédiger>

A28.5INTRODUCTION AU LANGAGE SQLfast

<à rédiger>

A28.6GUIDE DE SURVIE : MODÈLES TYPIQUES DE SCRIPTS SQLFAST

22 juin 2015

DOCUMENT PROVISOIRE, EN COURS DE REDACTION

Objectif

Ce tutoriel propose une série de modèles généraux de scripts SQLfast. Ces modèles peuvent servir de base à des scripts plus spécifiques et plus complexes. Les chapitres du tutoriel en pdf disponible sur le site SQLfast fourniront des exemples et des discussions plus détaillés.

Sauf rares exceptions, les modèles sont illustrés par le traitement de la base de données CLICOM.db, dont le contenu est rappelé ci-après.

La plupart des modèles sont complets. Ils peuvent être copiés et collés directement dans la fenêtre principale pour devenir un script local prêt à être exécuté.

Pour exécuter un modèle de script, on utilisera de préférence le bouton "Run script" plutôt que le bouton "Run next". De cette manière, chaque script est autonome et ne fait pas d'hypothèse sur les résultats des exécutions précédentes (hormis bien sûr l'état de la base de données). En particulier, la base de données est fermée, explicitement ou automatiquement, à la fin de l'exécution de chaque script.

Table des matières

(double-cliquer dans un titre pour accéder à la rubrique)

14. Où sommes-nous ? Les variables de l'environnement
15. Les calculs
16. L'alternative
17. Extraction de données d'une ligne d'une table
18. La boucle d'intervalle
19. La boucle SQL
20. La boucle sur fichier de texte
21. La boucle "while"
22. Contrôle d'une boucle
23. Contrôle des dialogues
24. Procédures, fonctions, applications SQLfast. Programmes externes
25. Valeur 'null' et chaînes vides
26. Affichage du résultat d'une requête
27. Personnalisation du format d'affichage
28. Où écrire ? Les différents supports d'écriture
29. Les opérations sur les fichiers externes
30. Traitement d'images
31. Gestion des erreurs
32. Arrêt d'un script
33. Contrôle de l'opérateur de substitution
34. Contrôle de l'état d'un script
35. Exécution automatique d'un script
36. Exécution de scripts de grande taille
37. Génération-exécution de scripts
38. Les tables du catalogue de la base de données
39. Visualisation des données en HTML

0. La base de données CLICOM.db - Version Juin 2015

table CLIENT

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

```
| S712 | GUILLAUME | 14a, ch. des Roses   | Paris     | B1  | 0      |
+------+-----------+----------------------+-----------+-----+--------+
```

table PRODUIT

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

tables COMMANDE et DETAIL

```
+-------+------+------------+
| NCOM  | NCLI | DATECOM    |
+-------+------+------------+
| 30178 | K111 | 2015-12-21 |
| 30179 | C400 | 2015-12-22 |
| 30182 | S127 | 2015-12-23 |
| 30184 | C400 | 2015-12-23 |
| 30185 | F011 | 2016-01-02 |
| 30186 | C400 | 2016-01-02 |
| 30188 | B512 | 2016-01-03 |
+-------+------+------------+
```

```
+-------+-------+------+
| NCOM  | NPRO  | QCOM |
+-------+-------+------+
| 30178 | CS464 | 25   |
| 30179 | CS262 | 60   |
| 30179 | PA60  | 20   |
| 30182 | PA60  | 30   |
| 30184 | CS464 | 120  |
| 30184 | PA45  | 20   |
| 30185 | CS464 | 260  |
| 30185 | PA60  | 15   |
| 30185 | PS222 | 600  |
| 30186 | PA45  | 3    |
| 30188 | CS464 | 180  |
| 30188 | PA45  | 22   |
| 30188 | PA60  | 70   |
| 30188 | PH222 | 92   |
+-------+-------+------+
```

1. Lire les données d'une table

Examiner les données de certaines tables d'une base de données.

```
openDB CLICOM.db;
  select * from CLIENT where LOCALITE = 'Poitiers';
  select * from DETAIL where NPRO = 'PA45';
closeDB;
```

La dernière instruction est recommandée mais est facultative en mode standard, dans lequel la base de données laissée ouverte par le script est automatiquement fermée (paramètre "autoCloseDB = mainscript"). On peut donc écrire :

```
openDB CLICOM.db;
select * from CLIENT;
```

Une instruction peut s'écrire en plusieurs lignes :

```
select NOM,LOCALITE from CLIENT where CAT = 'B1' order by
NOM;
```
ou
```
select NOM,LOCALITE
from   CLIENT
where  CAT = 'B1'
order by NOM;
```

2. Créer une base de données

On crée la base de données VENTES.db, contenant la table CLIENT, dans laquelle on insère quelques lignes.

```
createOrReplaceDB VENTE.db;
create table CLIENT(NCLI char(10),NOM char(32),LOCALITE
char(30));
  insert into CLIENT values('B112','HANSENNE','Poitiers');
  insert into CLIENT values('B512','GILLET','Toulouse');
  insert into CLIENT values('C400','FERARD','Poitiers');
  insert into CLIENT values('B332','MONTI','Genève');
commitDB;
closeDB;
```

L'instruction "createOrReplaceDB" crée une base de données si elle n'existe pas encore et la remplace si elle existe. L'instruction "createDB" suppose que la base de données n'existe pas. Dans le cas contraire, une erreur est déclenchée. L'instruction "commitDB" rend les modifications persistantes. Sans elle, les modifications seront annulées à la fermeture de la base de données.

3. Créer une table temporaire dans une base de données existante

On ajoute à la base de données CLICOM.db une table nommée BONS_CLIENT qui disparaîtra automatiquement lors de la fermeture de cette base de données. L'instruction "commitDB" est donc inutile.

```
openDB CLICOM.db;
create temporary table BONS_CLIENT(NCLI char(10),NOM
char(32));
  insert into BONS_CLIENTS select NCLI,NOM from CLIENT where
COMPTE > 0;
  select * from BONS_CLIENT;
closeDB;
```

4. Ecrire une ligne de texte

L'instruction "write" écrit dans la fenêtre de sortie une chaîne de caractère.

```
write Données de la table CLIENT;
write -------------------------;
```

```
write Note : ces données sont présentées sous une forme
tabulaire.;
```

5. Utilisation des variables

La chaîne 'Poitiers' (sans les apostrophes) est assignée à la variable de nom "ville" (sans les guillemets)

```
openDB CLICOM.db;
  set ville = Poitiers;
  write Clients habitant à $ville$ :;
  select NCLI,NOM from CLIENT where LOCALITE = '$ville$';
```

Ce script produit le résultat suivant :

Clients habitant à Poitiers :

```
+------+-----------+
| NCLI | NOM       |
+------+-----------+
| B112 | HANSENNE  |
| F010 | TOUSSAINT |
| C400 | FERARD    |
+------+-----------+
```

Les délimiteurs $ et $ entourant le nom de la variable constituent l'opérateur de "substitution variable/valeur". L'expression $ville$ est remplacée avant exécution par la valeur de ville, soit Poitiers.

Ainsi, l'instruction :

```
    select NCLI,NOM from CLIENT where LOCALITE = '$ville$'
```

est transformée en :

```
    select NCLI,NOM from CLIENT where LOCALITE = 'Poitiers'
```

puis est exécutée.

Pour économiser les instructions "write", on peut demander que la ligne écrite soit précédée d'une ligne blanche (-b = "before" :

```
write-b Clients habitant à $ville$ :;
```

ou suivie d'une ligne blanche (-a = "after") :

```
write-a Clients habitant à $ville$ :;
```

ou les deux :

```
write-ab Clients habitant à $ville$ :;
```

Les symboles $ et $ sont les délimiteurs par défaut. Ceux-ci peuvent être modifiés, notamment via l'instruction "parameter", qui permet d'écrire :

```
parameter delimiters = [ ];
write Clients habitant à [ville] :;
```

En fait, seul le délimiteur de droite est obligatoire :

```
parameter delimiters = : ;
write Clients habitant à :ville :;
```

Ou, pour les plus littéraires :

```
parameter delimiters = valeur( ) ;
write Clients habitant à valeur(ville) :;
```

Une instruction SQLfast peut être partiellement ou complètement définie à partir de variables. Quelques exemples :

```
set select = NCLI,NOM;
set table = CLIENT;
select $select$ from $table$;

set open = openDB CLICOM.db;
set sfw = select * from CLIENT;
set close = closeDB;
$open$;
$sfw$;
$close$;
```

6. Valeur fournie par l'utilisateur

Il est possible de demander à l'utilisateur d'un script de fournir la valeur d'une variable par l'instruction "ask":

```
ask ville = Localité;
write Clients habitant à $ville$ :;
```

Une boite de saisie de donnée s'ouvre et invite l'utilisateur à introduire le nom d'une localité. L'étiquete 'Localité' est assignée au champ de saisie. La valeur obtenue est assignée à la variable "ville" :

7. Saisie de valeurs multiples

Une boite de saisie à champs multiples permet de collecter plusieurs valeurs en une seule opérations :

```
ask nom,ville = Nom|Localité;
write Clients de nom $nom$ habitant à $ville$ :;
```

Ce qui donne :

Le symbole '|' sépare les étiquettes des différents champs et indique que ceux-ci sont disposés verticalement. Le séparateur '||' les disposeraient horizontalement, côte à côte.

8. Modification de valeurs par l'utilisateur

L'instruction "ask-u" ("-u" pour "update") propose à l'utilisateur de modifier une valeur :

```
set ville = Poitiers;
ask-u ville = [/bChoisir une nouvelle valeur] Localité;
write Clients de nom $nom$ habitant à $ville$ :;
```

Ce qui donne :

Le bloc [/bChoisir une nouvelle valeur] affiche un commentaire destiné, par exemple, à informer l'utilisateur. Le paramètre '/b' indique que ce commentaire doit apparaître en gras ("b" = "boldface"). Le commentaire peut occuper un nombre quelconque de lignes. Un passage à la ligne s'indique par l'insertion de la commande '@n' dans le commentaire. De même, la commande '@t' introduit une tabulation.

9. Saisie de valeurs par sélection dans une liste

Il s'agit d'inviter l'utilisateur à sélectionner une valeur dans une liste de propositions prédéfinies plutôt qu'introduire une valeur au clavier :

```
ask nom,ville =
Nom|Localité[(Paris,Londres,Bruxelles,Genève)];
write Clients de nom $nom$ habitant à $ville$ :;
```

Le bloc [(Paris,Londres,Bruxelles,Genève)] spécifie la liste (entre parenthèses) de valeurs de référence. A l'exécution, une liste déroulante apparaît:

Les valeurs peuvent être extraites de la base de données :

```
ask ville = Localité[select distinct LOCALITE from CLIENT
order by LOCALITE];
write Clients habitant à $ville$ :;
```

Le bloc [select distinct LOCALITE from CLIENT] spécifie la liste de valeurs sous la forme de requête SQL. Si la liste est longue, il est prudent de la trier par la clause "order by".

Dans les deux formats de liste, on peut forcer l'utilisateur à sélectionner une valeur en lui interdisant de l'introduire au clavier. On utilisera pour ce faire le symbole '!' :

```
ask ville = Localité[!select distinct LOCALITE from CLIENT];
```

10. Affichage de valeurs

(à rédiger)

```
showData cli,nom,loc = [Données à vérifier (OK ou Cancel)]
Numéro|Nom|Localité;
```

11. Sélection d'items (unique, multiples)

(à rédiger)

```
selectOne table = CLIENT|PRODUIT|COMMANDE|DETAIL;
selectMany cli,pro,com,det = CLIENT|PRODUIT|COMMANDE|DETAIL;
```

12. Saisie et affichage d'un texte. Affichage d'un message

(à rédiger)

```
askText   pv = [Introduire le PV de la réunion];
askText-u pv = [Introduire le PV de la réunion];
showText  pv = [Validez le PV de la réunion (OK ou Cancel)];
```

showMessage La base de données db n'existe pas.;
pause La base de données db n'existe pas.;

13. Le temps

Le temps est disponible via trois variables systèmes : "date" (date courante), "time" (temps dans la journée) et "timer" (temps écoulé, en secondes et millisecondes, depuis remise à zéro du chrono). L'instruction "start-timer" remet le chrono ("timer") à zéro. L'instruction "wait" suspend l'exécution soit un nombre fixe de millisecondes, soit un temps aléatoire entre deux bornes.

```
write Date  :  $date$;
write Date1 :  $date1$;
write Temps : $time$;
start-timer;
wait 100,900;
write Durée : $timer$;
```

Résultat :

```
Date  :  2015-6-18
Date1 :  18-6-2015
Temps : 22:58:54.703
Durée : 0.453
```

Utile pour dater un résultat et pour mesurer le temps d'exécution d'un traitement :

```
openDB CLICOM.db;
```

```
  start-timer;
  select count(*) from CLIENT,COMMANDE,DETAIL,PRODUIT;
  write Temps d'exécution : $timer$;
closeDB;
```

14. Où sommes-nous ? Les variables de l'environnement

SQLfast dispose d'une série de variables systèmes renseignant sur le contexte d'exécution d'un script : les répertoires courants, la plateforme, le SGBD, la base de données ouverte, le nom du script en cours d'exécution, le niveau d'appel du script courant, etc. :

```
openDB CLICOM.db;
  write scriptDirectory :    $scriptDirectory$;
  write dbDirectory :        $dbDirectory$;
  write platform :           $platform$;
  write DBMS :               $DBMS$;
  write dbName :             $dbName$;
  write scriptName :         $scriptName$;
closeDB;
```

Ce qui nous donnera un aperçu de l'environnement courant:

```
scriptDirectory :    D:/SQLfast-2015/SQLfast/Chapitre-18
dbDirectory :        D:/SQLfast-2015/Databases/
platform :           win32
DBMS :               SQLite 3.8.10.2
dbName :             CLICOM.db
scriptName :         _SQLfast_local.sql
```

15. Les calculs

L'instruction "compute" assigne à une variable le résultat de l'évaluation d'une expression SQL.

```
compute N = $N$ + 1;
compute dist = sqrt(sq($X$) + sq($Y$);
```

Remarque importante

L'expression à évaluer obéit strictement à la syntaxe SQL. Néanmoins, contrairement aux requêtes SQL proprement dites, les fonctions admises sont indépendantes du SGBD utilisé. Les assignations suivantes sont valides :

```
OK = $adresse$ like '%Neuve%'
année = year(current_date)
bonus = case when $M$ < 1000 then 20 else $M$/50 end;
traduc = replace(trim($adresse$),'Neuve','New');
```

16. L'alternative

L'alternative, ou instruction "if", permet d'exécuter une ou plusieurs instructions sous condition. La suppression de toutes les lignes de la table DETAIL sera effectuée si l'utilisateur le demande explicitement (réponse '1'); :

```
openDB CLICOM.db;
  ask del = Supprimer lignes de DETAIL (0/1) ?;
  if ($del$ = 1) delete from DETAIL;
  commitDB;
closeDB;
```

L'exécution conditionnelle peut concerner une séquence de plusieurs instructions :

```
openDB CLICOM.db;
  ask del = Supprimer lignes de DETAIL (0/1) ?;
  if ($del$ = 1);
     delete from DETAIL;
     commitDB;
     showMessage La table DETAIL est vide;
  endif;
closeDB;
```

On peut aussi spécifier les opérations à exécuter si la condition est fausse :

```
openDB CLICOM.db;
  ask del = Supprimer lignes de DETAIL (0/1) ?;
  if ($del$ = 1);
     delete from DETAIL;
     commitDB;
  else;
     showMessage La table DETAIL n'a pas été modifiée;
  endif;
closeDB;
```

Remarque importante
La condition obéit strictement à la syntaxe SQL. On écrira donc :

```
if ('$ville$' = 'Poitiers')
if (upper('$libelle$') like '%SAPIN%')
if ('$nombre$' = '0')
if (cast('$nombre$' as integer) = 0)
```

17. Extraction de données d'une ligne d'une table

(à rédiger)

```
extract nom,loc = select NOM,LOCALITE from CLIENT where NCLI
= 'C400';
extract nom,loc = select NOM,LOCALITE from CLIENT where CAT
is not null;
```

```
extract nom,loc = select NOM,LOCALITE from CLIENT where CAT
is not null #2;
```

18. La boucle d'intervalle

Permet d'exécuter une ou plusieurs instructions pour les valeurs successives d'in intervalle (fermé ou ouvert à droite) de nombres entiers ou réels.

```
for I = [1,5] write I = $I$;
for I = [1,5,2] write I = $I$;
for I = [1,2,0.1] write I = $I$;
for I = [1,2,0.1[ write I = $I$;
```

La forme "for-endfor" permet un corps de boucle de plusieurs instructions.

```
for I = [1,5];
   compute résultat = cast($I$*($I$-1)/2 as real);
   write Résultat pour I = $résultat$;
endfor;
```

19. La boucle SQL

Permet d'exécuter une ou plusieurs instructions pour les lignes successives du résultat d'une requête d'extraction SQL.

```
openDB CLICOM.db;
  for nom,loc = [select NOM,LOCALITE from CLIENT] write
$nom$, $loc$;
closeDB;
```

La forme "for-endfor" permet un corps de boucle de plusieurs instructions.

```
openDB CLICOM.db;
  ask cat = Catégorie[select distinct CAT from CLIENT where
CAT is not null];
  set I = 0;
  for nom,loc = [select NOM,LOCALITE from CLIENT where CAT =
'$cat$'];
     compute I = $I$ + 1;
     write Client $I$: $nom$, $loc$;
  endfor;
closeDB;
```

Il est possible de limiter le résultat aux lignes dont le rang tombe dans un intervalle :

```
for nom,loc = [select NOM,LOCALITE from CLIENT order by NCLI
#[1,5]];
for nom,loc = [select NOM,LOCALITE from CLIENT order by NCLI
#[$de$,$à$]];
```

```
for nom,loc = [select NOM,LOCALITE from CLIENT order by NCLI
#[1,5,2]];
```

Une structure de boucles emboîtées permet la prpduction de rapports personnalisés (la commande "@S3" crée 3 espaces) :

```
openDB CLICOM.db;
  for cli,nom = [select NCLI,NOM from CLIENT
                 where  LOCALITE = 'Poitiers'
                 and    NCLI in (select NCLI from COMMANDE)];
     write CLIENT : $cli$ $nom$;
     for com,dat = [select NCOM,DATECOM from COMMANDE where
NCLI = '$cli$'];
        write @S3COMMANDE $com$ du $dat$ :;
        for pro,qte,pri,mon = [select
D.NPRO,QCOM,PRIX,QCOM*PRIX
                               from   DETAIL D,PRODUIT P
                               where  NCOM = '$com$' and
D.NPRO = P.NPRO];
           write @S6Détail [$pro$] : $qte$ x $pri$ = $mon$;
        endfor;
     endfor;
  endfor;
closeDB;
```

Résultat :

```
CLIENT : C400 FERARD
   COMMANDE 30179 du 2015-12-22 :
      Détail [CS262] : 60 x 75 = 4500
      Détail [PA60] : 20 x 95 = 1900
   COMMANDE 30184 du 2015-12-23 :
      Détail [CS464] : 120 x 220 = 26400
      Détail [PA45] : 20 x 105 = 2100
   COMMANDE 30186 du 2016-01-02 :
      Détail [PA45] : 3 x 105 = 315
```

Exemple de lecture par pages de 5 lignes:

```
openDB CLICOM.db;
  extract Ncli = select count(*) from CLIENT;
  for de = [1,$Ncli$,5];
     compute à = $de$ + 4;
     for nom,loc = [select NOM,LOCALITE from CLIENT order by
NOM #[$de$,$à$]];
        write $nom$, $loc$;
     endfor;
     pause;
  endfor;
closeDB;
```

20. La boucle sur fichier de texte

Permet d'extraire les lignes successives d'un fichier de texte.

(à rédiger)

```
for ligne = [file commandes.txt];
   write $ligne$;
endfor;

openTextFile  donnees.txt;
nextTextLine  donnees.txt;
closeTextLine donnees.txt;
```

21. La boucle "while"

La boucle "while" permet d'exécuter une séquence d'une ou plusieurs instructions tant qu'une condition est satisfaite.

(à rédiger)

```
set I = 0;
while (not $I$ between 1 and 10) ask I = Nombre:;

while ($I$ <> 0);
   extract A = select City from CUSTOMER where CustID =
'$ID$';
   execSQL Process.sql;
endwhile;
```

22. Contrôle d'une boucle

On peut modifier le déroulement naturel d'une boucle par deux instructions :
- next : pour forcer l'itération suivante de la boucle,
- exit : pour sortir prématurément de la boucle.

Une technique (un peu barbare) pour ignorer les clients en négatif :

```
openDB CLICOM.db;
  for nom,loc,cpt = [select NOM,LOCALITE,COMPTE from CLIENT];
     if ($cpt$ < 0) next;
     write $nom$, $loc$;
  endfor;
closeDB;
```

Une boucle "while" peut être complètement contrôlée par les instructions "next" et "exit" :

```
while (True);
   execSQL Process.sql;
   if ($I$ = 0) exit;
   if ($I$ > 10) next;
   execSQL Print.sql;
endwhile;
```

23. Contrôle des dialogues

(à rédiger)

```
if ('$DIALOGbutton$' = 'OK') insert into DETAIL values
('$com$','$pro$',$qte$);
if ('$DIALOGbutton$' = 'Cancel' or '$loc$' = '') exit;
```

24. Procédures, fonctions, applications SQLfast. Programmes externes

(à rédiger)

```
execSQL lireParametres.sql;

execApp v1,v2 = saisieParametres $profil$;
function long,statut = LGeometrie:calculPerimetre $poly$;

execProg "Foxit Reader.exe" Manuel.pdf;
```

dans SQLfast.ini :

```
PDFprocessor = "D:/Program Files/Foxit Software/Foxit Reader/
Foxit Reader.exe"
execProg $PDFprocessor$ Manuel.pdf;
```

25. Valeur 'null' et chaînes vides

(à rédiger)

26. Affichage du résultat d'une requête

(à rédiger)

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

```
+--------+-----------+
| NCLI   | NOM       |
+--------+-----------+
| B112   | HANSENNE  |
| F010   | TOUSSAINT |
| C400   | FERARD    |
+--------+-----------+
```

27. Personnalisation du format d'affichage

(à rédiger)

```
                          selectIntro
1----2---3-----2-----4    bar1 = (bar11,bar12,bar13,bar14)
1 222222 3 2222      4    head = (head1,head2,head3,head4)
1----2---3-----2-----4    bar2 = (bar21,bar22,bar23,bar24)
```

```
1 2222   3 22222222  4  data = (data1,data2,data3,data4)
| F010   | TOUSSAINT |
| C400   | FERARD    |
1----2---3-----2-----4  bar3 = (bar31,bar32,bar33,bar34)
                        selectClose

set bar11 = +-;      set bar21 = +-;      set bar31 = +-;
set bar12 = -;       set bar22 = -;       set bar32 = -;
set bar13 = -+-;     set bar23 = -+-;     set bar33 = -+-;
set bar14 = -+@n;    set bar24 = -+@n;    set bar34 = -+@n;

set head1 = | ;      set data1 = | ;
set head2 = @h;      set data2 = @v;
set head3 = @s| ;    set data3 = @s| ;
set head4 = @s|@n;   set data4 = @s|@n;

selectIntro
selectClose
```

Formats prédéfinis:

```
UTIL-SELECT-parameters-standard-1.sql
UTIL-SELECT-parameters-standard-2.sql
UTIL-SELECT-parameters-for-CVS.sql
UTIL-SELECT-parameters-for-HTML.sql
UTIL-SELECT-parameters-for-KEY-VALUE.sql
UTIL-SELECT-parameters-for-LATEX.sql
UTIL-SELECT-parameters-for-RTF.sql
UTIL-SELECT-parameters-for-SQL.sql
UTIL-SELECT-parameters-for-TUPLE.sql
UTIL-SELECT-parameters-for-XML.sql
UTIL-SELECT-parameters-for-JSON.sql
```

Exemple : génération de données au format Key-value :

```
execSQL UTIL-SELECT-parameters-for-KEY-VALUE.sql;
select NCLI,NOM,LOCALITE from CLIENT;
```

28. Où écrire ? Les différents supports d'écriture

(à rédiger)

Les canaux de sortie :

- La fenêtre de sortie
```
outputOpen window;
```

- Un fichier texte
```
outputOpen depenses.csv;
```

- Un fichier texte de même nom que le script principal
```
outputOpen standard;
```

- Une variable interne

```
outputOpen resultat.var;
```

Modes d'écriture des canaux :

- Remplacement du contenu existant

```
outputOpen depenses.csv;
```

- Ajout au contenu existant

```
outputAppend depenses.csv;
```

29. Les opérations sur les fichiers externes

(à rédiger)

```
selectDirectory source = D:/SQLfast;
selectFile code = D:/SQLfast/Scripts, SQL scripts, sql;

copyFile CLICOM.db, CLICOM.bak;

deleteFile CLICOM.db;
deleteFileIfExists CLICOM.db;

fileNames nomsF.txt = D:/SQLfast/Images, jpg;
fileNamesAll nomsF.txt = D:/SQLfast/Images, jpg;

dirNames nomsD.txt = D:/SQLfast;
dirNamesAll nomsD.txt = D:/SQLfast;

dirFileNames nomsFD.txt = D:/SQLfast;
dirFileNamesAll nomsFD.txt = D:/SQLfast;

variableToFile param = parametres.txt;
fileToVariable parametres.txt = param;

for ligne = [file commandes.txt];
   write $ligne$;
endfor;
```

30. Traitement d'images

(à rédiger)

```
insertLOB var = IMAGE from ALBUM where ImID = 123;
extract var = select IMAGE from ALBUM where ImID = 123;
variableToFile var = image-06.gif;
showPicture image-06.gif = [Photo de l'arrivée];
```

31. Gestion des erreurs

(à rédiger)

```
SQLdiag
FILEdiag
EXTENDEDdiag
onError stop;
onError continue;
error ER0503 Violation de la contrainte d'unicité. Opération
annulée.;
```

32. Arrêt d'un script

(à rédiger)

```
return;
stop;
shutdown;
```

33. Contrôle de l'opérateur de substitution

(à rédiger)

```
set-n A = $B$;
substitution off;
substitution on;
```

34. Contrôle de l'état d'un script

(à rédiger)

```
procLevel
scriptName
mainScriptName
scriptNameFull
mainScriptNameFull

SQLdiag
FILEdiag
EXTENDEDdiag
DIALOGbutton

writeVariables;
writeSystemVariables;
writeParameters;

dropVariable A;
dropAllVariables;
```

+ variables d'environnement :
  `generalDirectory,scriptDirectory,outputDirectory,`

```
fileDirectory,dbDirectory,helpDirectory,
platform,sysPath
```

35. Exécution automatique d'un script

(à rédiger)

```
autoexec
shutDown;
```

36. Exécution de scripts de grande taille

Il s'agit d'une alternative rapide à "execSQL". Les instructions sont directement soumises au moteur SQL sans validation ni substitution de variables. Le script n'étant pas chargé en mémoire, il n'est pas soumis à une limitation de taille.

Le fichier source ne contient que des instructions SQL "create", "insert", "update" et "delete". Les lignes blanches et les commentaires (--) sont admis.

L'instruction spécifie le fichier source, et trois entiers : "first", "last" et "step". SQLfast execute les instructions dont le rang dans le fichier source est compris dans l'intervalle ["first","last"]. Le paramètre "step" indique l'endroit dans l'intervalle auquel un checkpoint est exécuté. A chaque checkpoint, un "commitDB" est exécuté si le suffixe '-c' est spécifié. En outre, si le suffixe '-t' est spécifié, un message précisant le temps d'exécution est affichée.

```
fastExec    source.sql, 1000000, 2000000, 50000;
fastExec-c  source.sql, 1000000, 2000000, 50000;
fastExec-t  source.sql, 1000000, 2000000, 50000;
fastExec-ct source.sql, 1000000, 2000000, 50000;
```

37. Génération-exécution de scripts

Un script peut générer un fichier contenant un script SQLfast puis en demander l'exécution. Les applications de ce modèle sont très nombreuses, en particulier pour spécialiser des scripts à la base de données courante.

```
outputOpen scriptGenere.sql;
  write openDB CLICOM.db;;
  write select * from CLIENT;;
  write closeDB;;
outputOpen window;
execSQL scriptGenere.sql;
deleteFile $scriptDirectory$/scriptGenere.sql;
```

Cette technique, très puissante, réclame un peu d'attention :

- outputOpen. L'instruction "outputOpen" dérive les écritures vers le fichier "scriptGenere.sql", qui sera placé dans le répertoire par défaut "scriptDirectory".
- write. Les trois instructions "write" créent les trois instructions du script. Attention à y inclure le signe ';' de fin d'instruction (il en faut donc deux).

- outputOpen. L'instruction "outputOpen window" est nécessaire pour reprendre la main mais aussi pour fermer le fichier et donc le libérer. Sinon, son exécution sera impossible (erreur Windows de code 32).
- execSQL. L'instruction "execSQL" exécute le nouveau script.
- deleteFile. L'instruction "deleteFile" (facultative) supprime ce script après exécution. Attention, il est nécessaire de préciser le répertoire de celui-ci ("$scriptDirectory$/scriptGenere.sql") puisque l'instruction "deleteFile" ignore qu'il s'agit d'un script.

Si le script générateur et le script généré contiennent tous les deux une expression telle que var, une deuxième exécution du générateur risque d'échouer parce que la variable var, inconnue lors de la première génération, a été créée dans le script généré et possède désormais une valeur. Il sera prudent d'insérer au début du générateur une instruction "dropVariable var" qui rendra cette variable inconnue.

38. Les tables du catalogue de la base de données

(à rédiger)

```
openDB CLICOM.db;
  createDictionary;
  select TableID,TableName from SYS_TABLE;
closeDB;

deleteDictionary;
```

39. Visualisation des données en HTML

(à rédiger)

```
askData loc = Nom de localité:;
outputOpen $loc$.html;
execSQL UTIL-SELECT-parameters-for-HTML.sql
select * from CLIENT where LOCALITE = '$loc$';
outputOpen window;
execProg $HTMLprocessor$ $loc$.html;
```