Date de dernière modification *: 27/9/2015*

Annexe **6**

Le langage SQL-DDL

Cette annexe s'attarde d'abord sur la prononciation du term SQL puis traduit le code de création et de chargement de la base de données CLICOM dans l'environnement SQLfast.

A6.1 COMMENT PRONONCER "SQL" ?

Il existe actuellement deux prononciations du nom SQL : soit comme la suite des trois lettres S, Q et L, soit sous l'énoncé du nom *Sequel*. Quelle est la prononciation correcte ? On peut répondre en deux points :

– les deux prononciations sont équivalentes et n'entraîneront aucune ambiguïté; tout différent à ce sujet est donc sans réel intérêt. On n'a d'ailleurs constaté aucun décès inexpliqué parmi des tenants de l'une ou de l'autre prononciation.

– néammoins,du point de vue historique, l'une est correcte et l'autre est fausse !

En effet, ces prononciations désignent deux produits différents. *Sequel* est le nom donné au langage de base de données développé pour System/R, le premier SGBD expérimental d'IBM, qui en a tiré au début des années 80 les SGBD commerciaux SQL/DS et DB2. Ce nom est apparu publiquement pour la première fois dans un article de Chamberlin et Boyce, *SEQUEL, a structured English query language*.[1] Pour des raisons de conflit avec une marque commerciale, ce nom a dû être abandonné. Il a été remplacé par celui de *Structured Query Language*, dont l'abréviation naturelle était S.Q.L., ou plus simplement SQL. SQL n'est donc en réalité pas un nom mais l'abréviation d'un nom. Le terme SQL a été repris par les SGBD relationnels développés plus tard, dont Oracle en 1979, ainsi que, dès 1986, par l'ANSI, qui préconise la prononciation S.Q.L. On lira d'ailleurs sur le site de MySQL (http://dev.mysql.com/doc/refman/5.1/en/what-is-mysql.html) :

> *The official way to pronounce "MySQL" is "My Ess Que Ell" (not "my sequel"), but we do not mind if you pronounce it as "my sequel" or in some other localized way.*

Les auteurs classiques recommandent et/ou utilisent en général la prononciation en trois lettres :

– Celko, *Trees and hierarchies in SQL*, Morgan Kaufmann, 2004 : "an SQL product"

– Melton, Simon, *Understanding the new SQL: a complete guide*, Morgan Kaufmann, 1993: "correctly pronounced "ess cue ell""

– Kreibich, *Using SQLite*, O'Reilly, 2010 : "the official pronunciation is to name each letter as "ess-cue-ell."".

– C. J. Date, *Database Systems*, Addison Wesley, 2004: "an SQL definition".

– Garcia-Molina, Ullman, Widom, *Database Systems, The complete book*, Prentice-Hall, 2002 : "an SQL query".

– Connolly, Begg, *Database Systems*, Addison Wesley, 2005 : "an SQL statement"

1. Chamberlin, D. and Boyce, R. "SEQUEL: A Structured English Query Language". Proceedings of the 1974 ACM SIGFIDET Workshop on Data Description, Access and Control (ACM): pp. 249–264 [http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf]

– Elmasri, Navathe, *Fundamentals of Database Systems*, Addison Wesley, 2000 : "an SQL schema".
– Korth, Silberschatz, *Database Systems Concepts*, McGraw-Hill, 1991 : "an SQL expression".

Quelques références à consulter [août 2015] :
– https://en.wikipedia.org/wiki/SQL
– http://patorjk.com/blog/2012/01/26/pronouncing-sql-s-q-l-or-sequel/
– http://www.vertabelo.com/blog/notes-from-the-lab/sql-or-sequel
– http://ocelot.ca/blog/blog/2013/09/23/how-to-pronounce-sql/

A6.2 UTILISATION DE SQLFAST

On trouvera à l'annexe A28 une introduction à l'installation et à l'utilisation de l'environnement SQLfast.

A6.3 CRÉATION D'UNE BASE DE DONNÉES

Le script A6.1 crée les structures de la base de données CLICOM.db soit dans le répertoire dédié aux bases de données si son nom est relatif (ce qui est le cas dans ce script), soit à l'adresse spécifiée si son nom inclut le chemin complet (par exemple D:/SQLfast/Databases/CLICOM.db). Ce répertoire est spécifié par le paramètre **`dbDirectory`** du fichier d'initialisation **SQLfast.ini**. Pour rappel, ce fichier permet notamment de préciser l'emplacement des principaux répertoires de l'environnement SQLfast :

```
-- Default directories
-- -------------------
generalDirectory = D:/SQLfast
scriptDirectory  = Scripts
outputDirectory  = Texts
fileDirectory    = Files
dbDirectory      = Databases
helpDirectory    = SQLfastHelp_FR
```

Par défaut, les répertoires de travail, dont celui qui contient les bases de données, sont localisés dans le répertoire général (`generalDirectory`) qui contient le logiciel SQLfast.

```
createOrReplaceDB CLICOM.db;

create table CLIENT (  NCLI     char(10) not null,
                       NOM      char(32) not null,
                       ADRESSE  char(60) not null,
                       LOCALITE char(30) not null,
                       CAT      char(2),
                       COMPTE   decimal(9,2) not null,
                    primary key (NCLI) );

create table PRODUIT ( NPRO     char(15) not null,
                       LIBELLE  char(60) not null,
                       PRIX     decimal(6) not null,
                       QSTOCK   decimal(8) not null,
                    primary key (NPRO) );

create table COMMANDE (NCOM     char(12) not null,
                       NCLI     char(10) not null,
                       DATECOM  date not null,
                    primary key (NCOM),
                    foreign key (NCLI) references CLIENT
                    on delete no action on update cascade);

create table DETAIL (  NCOM     char(12) not null,
                       NPRO     char(15) not null,
                       QCOM    decimal(8) not null,
                    primary key (NCOM,NPRO),
                    foreign key (NCOM) references COMMANDE
                    on delete cascade on update cascade,
                    foreign key (NPRO) references PRODUIT
                    on delete no action on update cascade);

commitDB;
closeDB;
```

Figure A6.1 - Creation de la base de données CLICOM.db

La base de données est créée par une instruction **`createDB`** ou **`createOrReplaceDB`**. Dans le premier cas, l'existence préalable d'une base de données portant ce nom provoque une erreur alors que dans le second cas, cette base de données sera automatiquement remplacée par la nouvelle.

La suite des instructions **`create table`** crée les quatre tables de la base de données. L'instruction **`commitDB`** confirme la modification de la base de données[2] et l'instruction **`closeDB`** ferme la base de données.

On note l'absence des objets de *connexion* et des *curseurs* des langages de bases de données traditionnels.

2. Il faut noter que certains SGBD, dont SQLite, exécutent automatiquement un `commit` avant et après l'exécution de chaque instruction DDL. Dans ce cas, le **`commitDB`** est inutile.

En cas d'erreur d'exécution, le script s'arrête prématurément et un message est affiché. Pour gérer les erreurs, il faut inhiber cet arrêt par l'instruction **`onError continue`**. Dans ce cas, l'erreur est spécifiée par le contenu des variables système **`SQLdiag`** et **`EXTENDEDdiag`**. Le comportement par défaut est **`onError stop`**.

A6.4 CHARGEMENT D'UNE BASE DE DONNÉES

Pour charger les données, on rédige un second script (A6.2) dans lequel on ouvre la base de données (**`openDB`**), on insère les lignes de données (**`insert`**, qui sera étudiée au chapitre 8), on clôture la transaction (**`commitDB`**) puis on ferme la base de données (**`closeDB`**).

A6.5 ÉTATS D'UNE BASE DE DONNÉES EN SQLfast

Quelques remarques concernant la gestion d'une base de données en SQLfast:

1. Normalement, si un script se termine sans fermer la base de données, celle-ci reste ouverte. Le script suivant peut alors travailler sur cette base de données sans devoir l'ouvrir.
2. Une seule base de données peut être ouverte à la fois. Pour ouvrir une autre base de données la première doit d'abord être fermée.
3. Par défaut, lors de la clôture de la *session courante* de SQLfast, toute base de données qui serait encore ouverte est fermée. Une opération **`abortDB`** est exécutée au préalable, de manière à annuler les modifications des données qui n'auraient pas encore été confirmées. *Conclusion* : **toujours** terminer une série de modifications par un **`commit`**.
4. Cette fermeture automatique par défaut peut être modifiée par le paramètre **`autoCloseDB`** du fichier SQLfast.ini :

   `autoCloseDB = session` : la base de données est automatiquement fermée à la fin de la session. Il s'agit de l'option par défaut.

   `autoCloseDB = mainscript` : la base de données est automatiquement fermée à la terminaison du *script principal* (celui qui est spécifié à partir de la boite de sélection de script).

   `autoCloseDB = script` : la base de données est automatiquement fermée lorsque le script courant (principal ou secondaire) se termine.
5. **Rappel** : la *session* est l'ensemble des activités effectuées durant une exécution ininterrompue de SQLfast. Un *script principal* est un script lancé manuellement durant une session par l'utilisateur (ou exécuté automatiquement au démarrage de la session selon le paramètre **`autoExec`**). Un *script secondaire* est exécuté à partir d'un autre script, principal ou secondaire, par l'instruction **`execSQL`**.

```
openDB CLICOM.db;

insert into CLIENT values
    ('B112','HANSENNE','23, r. Dumont','Poitiers','C1',1250);
insert into CLIENT values
    ('C123','MERCIER','25, r. Lemaître','Namur','C1',-2300);
...
insert into CLIENT values
    ('D063','MERCIER','201, bvd du Nord','Toulouse',null,-
2250);
insert into CLIENT values
    ('F400','JACOB','78, ch. du Moulin','Bruxelles','C2',0);

insert into PRODUIT values
    ('CS262','CHEV. SAPIN 200x6x2',75,45);
insert into PRODUIT values
    ('CS264','CHEV. SAPIN 200x6x4',120,2690);
...
insert into PRODUIT values
    ('PH222','PL. HETRE 200x20x2',230,782);
insert into PRODUIT values
    ('PS222','PL. SAPIN 200x20x2',185,1220);

insert into COMMANDE values ('30178','K111','2015-12-21');
insert into COMMANDE values ('30179','C400','2015-12-22');
...
insert into COMMANDE values ('30186','C400','2016-1-2');
insert into COMMANDE values ('30188','B512','2016-1-3');

insert into DETAIL values ('30178','CS464',25);
insert into DETAIL values ('30179','CS262',60);
...
insert into DETAIL values ('30188','PA60',70);
insert into DETAIL values ('30188','PH222',92);

commitDB
closeDB;
```

Figure A6.2 - Extraits du script de chargement des données de CLICOM.db

A6.6 LES SCRIPTS SQLFAST DU CHAPITRE 6

Les deux scripts illustrés ci-dessus ont été regroupés dans le fichier **CLICOM-creer-charger.sql**. Le script **CLICOM-extraire-tout.sql** permet de visualiser le contenu de la base de données (figure A6.3).

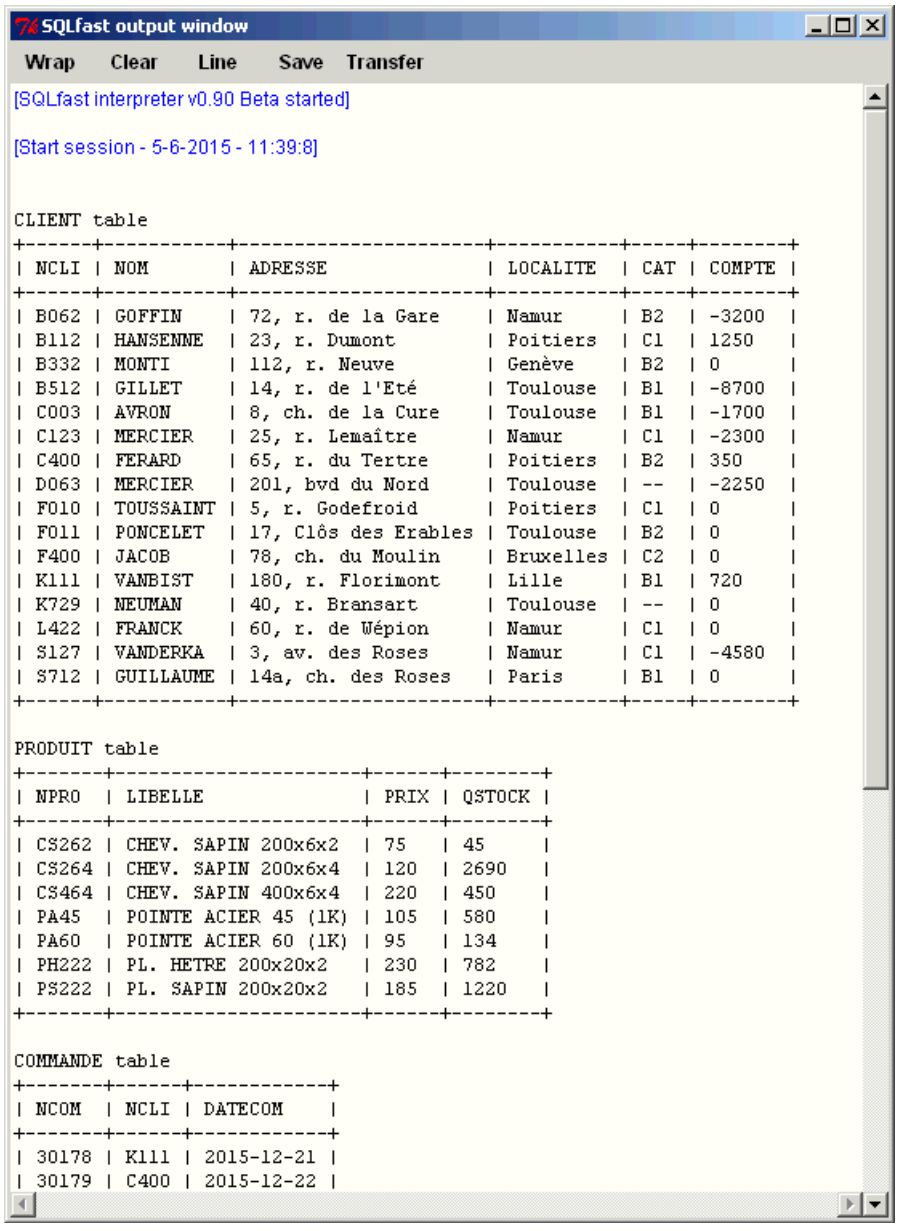
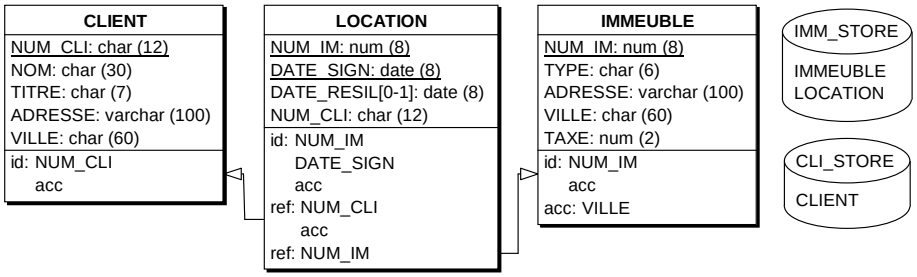


Figure A6.3 - Résultat de l'exécution du script **CLICOM-extraire-tout.sql**

A6.7 EXERCICES DU CHAPITRE 6

Un jeu d'exercices accompagnés de leurs solutions est disponible dans le tutoriel **6._Modification_des_donnees.tuto** du niveau *Learning SQL* de l'environnement SQLfast.

A6.1 Écrire le code SQL DDL correspondant au schéma ci-dessous.



Solution
Le code ci-dessous a été généré par l'atelier DB-MAIN selon le style *Standard SQL*. A noter que ce schéma n'est pas strictement conforme au standard SQL2. Il n'est pas non plus totalement conforme à SQLfast mais peut être adapté sans difficulté.
Note : les dernières versions de DB-MAIN comprennent un générateur SQLite.

```
-- ********************************************
-- * Standard SQL generation                  *
-- *------------------------------------------*
-- * Generator date:  Apr 14 2003             *
-- * Generation date: Sun Dec 28 18:17:20 2008 *
-- ********************************************

-- Database Section
-- ________________

create database DUNOD_2009_Ch06;

-- DBSpace Section
-- _______________

create dbspace IMM_STORE;

create dbspace CLI_STORE;

-- Table Section
-- _____________

create table CLIENT (
```

```
     NUM_CLI char(12) not null,
     NOM char(30) not null,
     TITRE char(7) not null,
     ADRESSE varchar(100) not null,
     VILLE char(60) not null,
     primary key (NUM_CLI))
     in CLI_STORE;

create table IMMEUBLE (
     NUM_IM numeric(8) not null,
     TYPE char(6) not null,
     ADRESSE varchar(100) not null,
     VILLE char(60) not null,
     TAXE numeric(2) not null,
     primary key (NUM_IM))
     in IMM_STORE;

create table LOCATION (
     NUM_IM numeric(8) not null,
     DATE_SIGN date not null,
     DATE_RESIL date,
     NUM_CLI char(12) not null,
     primary key (NUM_IM, DATE_SIGN))
     in IMM_STORE;

-- Constraints Section
-- ___________________

alter table LOCATION add constraint GRLOCATION
     foreign key (NUM_CLI)
     references CLIENT;
alter table LOCATION add constraint GRLOCATION_1
     foreign key (NUM_IM)
     references IMMEUBLE;

-- Index Section
-- _____________
create unique index IDCLIENT
     on CLIENT (NUM_CLI);
create unique index IDIMMEUBLE
     on IMMEUBLE (NUM_IM);
create index GRIMMEUBLE
     on IMMEUBLE (VILLE);
create index GRLOCATION
     on LOCATION (NUM_CLI);
create unique index IDLOCATION
     on LOCATION (NUM_IM, DATE_SIGN);
```