SQL – API

Le but, pour un analyste-développeur, est de pouvoir se connecter à une base de données au travers d'une application développée spécifiquement.

Pour ce faire, les SGBDR mettent à disposition une API (Application Programming Interface).

MySQL n'échappe pas à la règle.

SQL – API

Les fichiers d'entête (header files) et les bibliothèques (code de l'API) sont installés :

- avec le serveur MySQL
- avec le connecteur C de MySQL (ce qui évite d'installer un serveur sur un poste de développement qui se connecterait à un serveur tiers)

API – Documentation

La documentation complète de l'API est disponible sur le site officiel.

API – C

Même s'il est possible de se connecter à MySQL au travers de dizaines de langages, ce document décrit l'utilisation de l'API en langage C.

API – Header

La première instruction nécessaire à l'utilisation de MySQL dans votre code C est l'inclusion du fichier header central :

```
#include <mysql.h>
```

Rem : il peut être nécessaire d'ajouter le chemin d'accès au dossier des fichiers d'entête mysql dans les paramètres de votre projet

API – Bibliothèques

Vous devez également ajouter les fichiers bibliothèques au paramètres de « linkage » du projet.

Ces fichiers peuvent être (selon l'OS utilisé) :

- libmysqlclient.dll
- libmysqlclient.a
- libmysqlclient.so
- libmysqlclient.lib

Vous trouverez plus de détails sur cette page

API – Configuration

Il est parfois nécessaire de signaler à Mysql que les connexions distantes sont acceptées.

Il faut pour cela modifier le fichier de config (my.ini ou my.cnf) comme ceci :

```
[mysqld]
bind-address  = 0.0.0.0
```

API – Variables

Plusieurs types de variables sont nécessaires à l'exécution d'une requête de sélection :

- `MYSQL` : structure de gestion des connexions
- `MYSQL_RES` : structure de gestion des résultats
- `MYSQL_ROW` : structure de gestion d'une ligne de résultat
- `MYSQL_FIELD` : structure de gestion des colonnes

API – Initialisation

La fonction permettant d'initialiser la structure globale :

```
MYSQL mysql;
mysql_init(&mysql);
```

API – Connexion

La fonction de connexion au serveur prend plusieurs paramètres :

- Structure globale (de type `MYSQL`)
- Adresse du serveur (de type `const char*`)
- Nom d'utilisateur (de type `const char*`)
- Mot de passe de l'utilisateur (de type `const char*`)
- Nom de la DB (de type `const char*`)
- Numéro du port (de type `unsigned int`)
  - Si 0, port par défaut est utilisé
- Socket UNIX (de type `const char*`) : `NULL` par défaut
- Configuration client (de type `unsigned long`) : 0 par défaut

API – Connexion

La fonction de connexion :

```
MYSQL mysql;
…
mysql_real_connect(&mysql, serveur, utilisateur,
mot_passe, nom_db, num_port, NULL, 0);

// Ou
MYSQL mysql;
MYSQL *connexion = NULL;
…
connexion = mysql_real_connect(&mysql, serveur,
utilisateur, mot_passe, nom_db, num_port, NULL,
0);
```

API – Requête

La fonction d'envoi de la requête :

```
mysql_query(connexion, "SELECT * FROM client");
```

API – Résultats

La fonction de sauvegarde des résultats :

```
MYSQL_RES *resultats = NULL;
…
resultats = mysql_use_result(connexion);
```

API – Résultats

La fonction permettant de connaître le nombre de colonnes retournées :

```
unsigned int n_colonnes;
…
n_colonnes = mysql_num_fields(resultats);
```

API – Résultats

La fonction permettant de connaître le nombre de lignes retournées :

```
unsigned int n_lignes;
…
// Pour un SELECT
n_lignes = mysql_num_rows(resultats);

// Pour un INSERT, UPDATE ou DELETE
n_lignes = mysql_affected_rows(resultats);
```

API – Résultats

La fonction permettant de lire la ligne suivante dans les résultats :

```
MYSQL_ROW ligne;
…
ligne = mysql_fetch_row(resultats)
```

API – Résultats

Boucle de lecture et d'affichage de toutes les lignes retournées :

```
// Pour toutes les lignes retournées
while ((ligne = mysql_fetch_row(resultats))) {
    // On affiche toutes les colonnes
    for (i = 0; i < n_colonnes; i++) {
        printf("%s ",ligne[i]);
    }
    putchar('\n');
}
```

API – Libération

Libération des résultats :

```
mysql_free_result(resultats);
```

API – Fermeture

Fermeture de la connexion :

```
mysql_close(connexion);
```

API – Erreur

Message d'erreur :

```
mysql_error(&mysql);
// ou
mysql_error(connexion);
```

Numéro d'erreur :

```
mysql_errno(&mysql);
// ou
mysql_error(connexion);
```

API – Exemple complet

Voir fichier :

- mysql_api_select.c sur moodle pour un exemple complet de sélection
- mysql_api_update.c sur moodle pour un exemple complet de mise à jour

À partir de ces exemples, il est très aisé d'écrire un code d'insertion et de suppression.

API – Exercices

Exercices 1 :

A partir des exemples précédents, veuillez écrire :

- un code d'insertion
- Un code de suppression.

API – Exercices

Exercice 2 :

- Créez un formulaire d'ajout d'un client

API – Exercices

Exercice 3 :

- Poursuivez l'exercice 2 en le complétant avec un formulaire d'ajout d'une commande

API – Exercices

Exercice 4 :

- Réalisez un formulaire de mise à jour d'un client qui propose par défaut, pour un idclient donné, les valeurs actuelles pour chacune des colonnes pouvant être modifiées