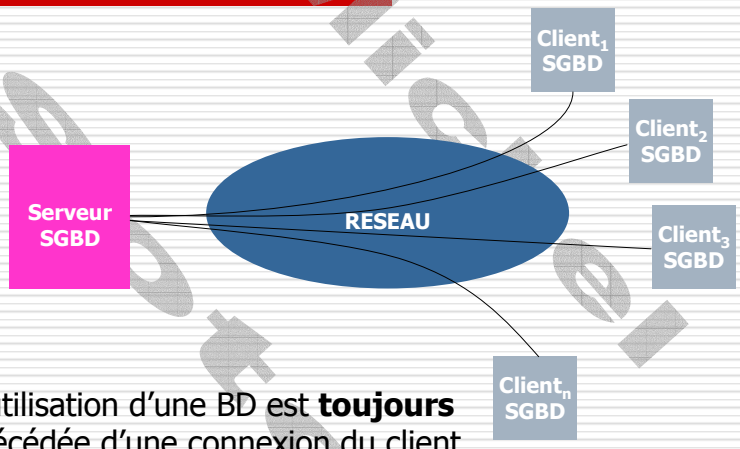
Le langage SQL

Intégration dans un langage hôte de programmation (Embeded SQL)

Déploiement des SGBD

- **Architecture Client-Serveur**
  - **SERVEUR**
    - Il réalise les tâches exigeant beaucoup de mémoire, de puissance CPU et d'espace disque.
      - Ces tâches sont réalisées sur une machine dotée des ressources en conséquence.
  - **CLIENTS**
    - Ils réalisent les tâches peu exigeantes en mémoire, en puissance CPU et en espace disque.
      - Ces tâches sont réalisées sur une machine dotée de ressources standards.
  - Les clients et le serveur **communiquent via le réseau**.
  - L'ensemble {**Clients + Serveur**} forme **l'application complète**
  - Cette architecture est utilisée par de très nombreuses applications:
    - WEB
    - Courrier électronique
    - Transfert de fichiers
    - VOD
    - **SGBD**
    - ...

Déploiement des SGBD



L'utilisation d'une BD est **toujours** précédée d'une connexion du client au serveur

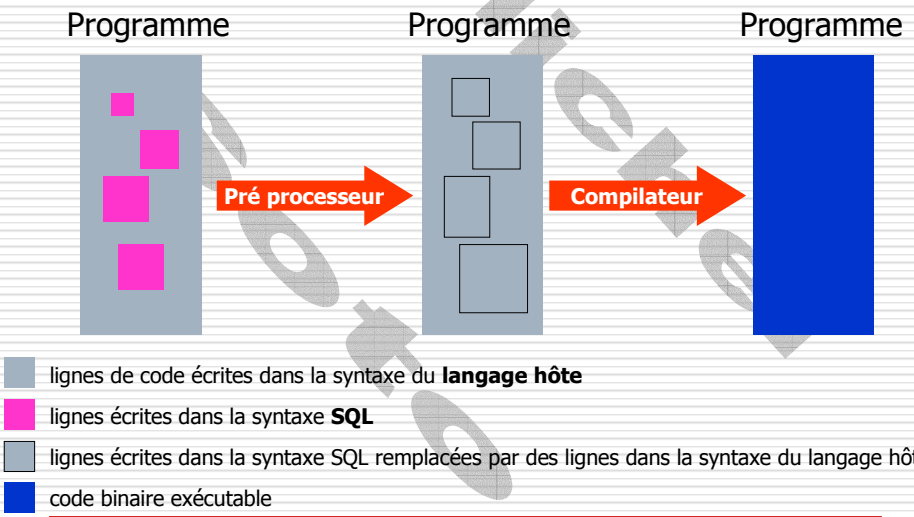
Utilité de SQL en mode intégré

- **Le mode interactif ne convient pas:**
  - Lors de tâches répétitives sur un grand volume de données
  - Lorsque les utilisateurs ne possèdent pas les connaissances SQL
  - Certaines limites SQL sont atteintes (récursivité)
- **Le *mode intégré* (embeded) consiste à utiliser SQL à partir d'un langage de programmation classique appelé *langage hôte* :**
  - Langage C
  - Java
  - Etc.

Principe d'intégration de SQL dans un langage hôte

- **Un programme sera constitué à la fois de:**
  - lignes de code écrites dans la ***syntaxe du langage hôte***
  - lignes écrites dans la ***syntaxe SQL***
- **Ce programme fera l'objet d'un pré-traitement avant sa compilation**
  - Les *lignes SQL seront remplacées* par des lignes conforme à la syntaxe du langage hôte afin de rendre la compilation possible.
- **La notion de curseur est ajoutée à SQL afin de pouvoir parcourir et traiter un ensemble de tuples résultant de l'exécution d'une requête.**
- **Une variable SQLCODE permet de récupérer dans le programme des informations sur le déroulement d'une requête SQL**

Principe d'intégration de SQL dans un langage hôte



lignes de code écrites dans la syntaxe du **langage hôte**

lignes écrites dans la syntaxe **SQL**

lignes écrites dans la syntaxe SQL remplacées par des lignes dans la syntaxe du langage hôte

code binaire exécutable

Principe d'intégration de SQL dans un langage hôte

- **Exemple en C**

```
int main(){
    exec sql insert into joueur values ('Noah', 'Yannick', 26, 'France');
    return(0);

} /* main */
```

Chaque instruction SQL est repérée dans le programme hôte par les mots EXEC SQL

Pré processeur

```
int main(){
  {ECPGdo(__LINE__, 0, 1, NULL, 0, ECPGst_normal,
   "insert into joueur values ( 'Noah' , 'Yannick' , 26 , 'France' )",
   ECPGt_EOIT, ECPGt_EORT);}
   return(0);

} /* main */
```


Variables hôtes

- **Variables du langage hôte utilisées aussi par les instructions SQL**
  - Déclarées dans le programme hôte entre les lignes:
    - **EXEC SQL BEGIN DECLARE SECTION**
    - **EXEC SQL END DECLARE SECTION**
  - Préfixées par le caractère '**:**' lors de leur utilisation dans une requête SQL
  - Les types utilisés :
    - sont ceux du langage hôte
    - le type spécial VARCHAR
    - doivent être compatibles avec les types SQL des attributs de la BD

Variables hôtes (suite)

- **Exemple 1**

```
exec sql begin declare section;
   varchar nom[20];     /* Le type varchar n'existe pas en C */
   char prenom [16];    /* 15 + 1 pour le \0 */
   int age;
   char nationalite[16]; /* 15 + 1 pour le \0 */
exec sql end declare section;

int main(){
   return(0);
} /* main */
```

Pré processeur

```
struct varchar_nom  { int len; char arr[ 20 ]; }  nom;
char  prenom [16];
int  age;
char  nationalite [16];

int main(){
   return(0);
} /* main */
```


Variables hôtes (fin)

- **Exemple 2**

```
exec sql begin declare section;
   varchar nom[20];
   char prenom [16];    /* 15 + 1 pour le \0 */
   int age;
   char nationalite[16]; /* 15 + 1 pour le \0 */
exec sql end declare section;

int main(){
  printf("Nom joueur ? "); scanf("%s", nom.arr);
  nom.len=strlen(nom.arr);

  printf("Prénom joueur ? ");      scanf("%s", prenom);
  printf("Age joueur ? ");         scanf("%d", &age);
  printf("Nationalité joueur ? "); scanf("%s", nationalite);

  exec sql insert into joueur values (:nom, :prenom, :age, :nationalite);

  return(0);

} /* main */
```

Curseur

- **Utilisé afin d'itérer un traitement sur un ensemble de tuples produit par une requête SQL**
  - Déclaré dans le programme hôte:
    - `exec sql declare Nom_Curseur cursor for requête_SQL;`
      - Associe le nom du curseur à une requête SQL
  - Ouvert par le programme hôte avant son utilisation:
    - `exec sql open Nom_Curseur;`
      - Provoque l'envoi au serveur de la requête SQL associée au curseur puis la réception par le client du résultat de la requête

Curseur (fin)

- Fermé par le programme hôte après son utilisation:
  - `exec sql close Nom_Curseur;`
    - Libère la mémoire occupée par l'ensemble des tuples du résultat de la requête
- Les variables hôtes sont valuées à chaque déplacement du curseur
  - `exec sql fetch from Nom_Curseur into :V1, …,:Vn;`
- Une variable nommée **SQLCODE** permet de contrôler l'itération du traitement sur les tuples
  - De façon générale, **SQLCODE** , permet de récupérer dans le programme des informations sur le déroulement d'une requête SQL (échec, succès, etc)

Curseur – exemple

SERVEUR SGBD (MACHINE X)

BD TENNIS

JOUEUR
<Connors, Jimmy, 1952, USA>
<Fromm, Eric, 1958, USA>
<Becker, Boris, 1967, Allemagne>

```
CLIENT (MACHINE Y)
exec sql begin declare section;
    varchar nom[21];
    char prenom [16];
    int age;
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
     printf("%s %s %d\n", nom.arr, prenom, age);
     exec sql fetch from C into :nom, :prenom, :age;
  } /* for */

  exec sql close C;
}
```


Curseur – exemple

(2)

CACHE DE LA MACHINE Y
<Connors, Jimmy, 45>
<Fromm, Eric, 42>
<Becker, Boris, 49>

SERVEUR SGBD (MACHINE X)

BD TENNIS

JOUEUR
<Connors, Jimmy, 1952, USA>
<Fromm, Eric, 1958, USA>
<Becker, Boris, 1967, Allemagne>

(1) select nom, prenom from joueur;

```
CLIENT (MACHINE Y)
exec sql begin declare section;
    varchar nom[21];
    char prenom [16];
    int age;
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
     printf("%s %s %d\n", nom.arr, prenom.arr, age);
     exec sql fetch from C into :nom, :prenom, :age;
  } /* for */

  exec sql close C;
}
```

Curseur – exemple

```
SERVEUR SGBD (MACHINE X)
BD TENNIS
  JOUEUR
  <Connors, Jimmy, 1952, USA>
  <Fromm, Eric, 1958, USA>
  <Becker, Boris, 1967, Allemagne>
```

```
CACHE DE LA MACHINE Y
<Connors, Jimmy, 45>
<Fromm, Eric, 42>
<Becker, Boris, 49>
```

```
CLIENT (MACHINE Y)
exec sql begin declare secti
    varchar nom[21];
    char prenom [16]; Jimmy
    int age; 45
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
     printf("%s %s %d\n", nom.arr, prenom.arr, age);
     exec sql fetch from C into :nom, :prenom, :age;
   } /* for */

   exec sql close C;
}
```

```
struct varchar_nom  {
  int len; 7
   char arr[20]; Connors
} nom
```


Curseur – exemple

```
SERVEUR SGBD (MACHINE X)
BD TENNIS
  JOUEUR
  <Connors, Jimmy, 1952, USA>
  <Fromm, Eric, 1958, USA>
  <Becker, Boris, 1967, Allemagne>
```

```
CACHE DE LA MACHINE Y
<Connors, Jimmy, 45>
<Fromm, Eric, 42>
<Becker, Boris, 49>
```

```
CLIENT (MACHINE Y)
exec sql begin declare secti
    varchar nom[21];
    char prenom [16]; Jimmy
    int age; 45
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
     printf("%s %s %d\n", nom.arr, prenom.arr, age);
     exec sql fetch from C into :nom, :prenom, :age;
   } /* for */

   exec sql close C;
}
```

```
struct varchar_nom  {
  int len; 7
   char arr[20]; Connors
} nom
```

Connors Jimmy 45

Curseur – exemple

SERVEUR SGBD (MACHINE X)

BD TENNIS

JOUEUR
<Connors, Jimmy, 1952, USA>
<Fromm, Eric, 1958, USA>
<Becker, Boris, 1967, Allemagne>

```
CACHE DE LA MACHINE Y
<Connors, Jimmy, 45>
<Fromm, Eric, 42>
<Becker, Boris, 49>
```

```
CLIENT (MACHINE Y)
exec sql begin declare secti
    varchar nom[21];
    char prenom [16]; Eric
    int age; 42
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
      printf("%s %s %d\n", nom.arr, prenom.arr, age);
      exec sql fetch from C into :nom, :prenom, :age;
  } /* for */

  exec sql close C;
}
```

```
struct varchar_nom  {
   int len; 5
    char arr[20]; Fromm
} nom
```

Connors Jimmy 45

Curseur – exemple

SERVEUR SGBD (MACHINE X)

BD TENNIS

JOUEUR
<Connors, Jimmy, 1952, USA>
<Fromm, Eric, 1958, USA>
<Becker, Boris, 1967, Allemagne>

```
CACHE DE LA MACHINE Y
<Connors, Jimmy, 45>
<Fromm, Eric, 42>
<Becker, Boris, 49>
```

```
CLIENT (MACHINE Y)
exec sql begin declare secti
    varchar nom[21];
    char prenom [16]; Eric
    int age; 45
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
      printf("%s %s %d\n", nom.arr, prenom.arr, age);
      exec sql fetch from C into :nom, :prenom, :age;
  } /* for */

  exec sql close C;
}
```

```
struct varchar_nom  {
   int len; 5
    char arr[20]; Fromm
} nom
```

Connors Jimmy 45
Fromm Eric 42

Curseur – exemple

SERVEUR SGBD (MACHINE X)

BD TENNIS

JOUEUR
<Connors, Jimmy, 1952, USA>
<Fromm, Eric, 1958, USA>
<Becker, Boris, 1967, Allemagne>

```
CACHE DE LA MACHINE Y
<Connors, Jimmy, 45>
<Fromm, Eric, 42>
<Becker, Boris, 49>
```

```
CLIENT (MACHINE Y)
exec sql begin declare secti
    varchar nom[21];
    char prenom [16]; Boris
    int age; 49
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
      printf("%s %s %d\n", nom.arr, prenom.arr, age);
      exec sql fetch from C into :nom, :prenom, :age;
   } /* for */

  exec sql close C;
}
```

```
struct varchar_nom  {
  int len; 6
   char arr[20]; Becker
} nom
```

Connors Jimmy 45
Fromm Eric 42

Curseur – exemple

SERVEUR SGBD (MACHINE X)

BD TENNIS

JOUEUR
<Connors, Jimmy, 1952, USA>
<Fromm, Eric, 1958, USA>
<Becker, Boris, 1967, Allemagne>

```
CACHE DE LA MACHINE Y
<Connors, Jimmy, 45>
<Fromm, Eric, 42>
<Becker, Boris, 49>
```

```
CLIENT (MACHINE Y)
exec sql begin declare secti
    varchar nom[21];
    char prenom [16]; Boris
    int age; 49
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
      printf("%s %s %d\n", nom.arr, prenom.arr, age);
      exec sql fetch from C into :nom, :prenom, :age;
   } /* for */

  exec sql close C;
}
```

```
struct varchar_nom  {
  int len; 6
   char arr[20]; Becker
} nom
```

Connors Jimmy 45
Fromm Eric 42
Becker Boris 49

Curseur – exemple

SERVEUR SGBD (MACHINE X)

BD TENNIS

JOUEUR
<Connors, Jimmy, 1952, USA>
<Fromm, Eric, 1958, USA>
<Becker, Boris, 1967, Allemagne>

```
CACHE DE LA MACHINE Y
<Connors, Jimmy, 45>
<Fromm, Eric, 42>
<Becker, Boris, 49>
```

```
CLIENT (MACHINE Y)
exec sql begin declare secti
    varchar nom[21];
    char prenom [16]; Boris
    int age; 49
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
      printf("%s %s %d\n", nom.arr, prenom.arr, age);
      exec sql fetch from C into :nom, :prenom, :age;
  } /* for */

  exec sql close C;
}
```

```
struct varchar_nom  {
   int len; 6
    char arr[20]; Becker
} nom
```

Connors Jimmy 45
Fromm Eric 42
Becker Boris 49

Curseur – exemple

SERVEUR SGBD (MACHINE X)

BD TENNIS

JOUEUR
<Connors, Jimmy, 1952, USA>
<Fromm, Eric, 1958, USA>
<Becker, Boris, 1967, Allemagne>

```
CACHE DE LA MACHINE Y
<Connors, Jimmy, 45>
<Fromm, Eric, 42>
<Becker, Boris, 49>
```

```
CLIENT (MACHINE Y)
exec sql begin declare secti
    varchar nom[21];
    char prenom [16]; Boris
    int age; 49
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
      printf("%s %s %d\n", nom.arr, prenom.arr, age);
      exec sql fetch from C into :nom, :prenom, :age;
  } /* for */

  exec sql close C;
}
```

```
struct varchar_nom  {
   int len; 6
    char arr[20]; Becker
} nom
```

Connors Jimmy 45
Fromm Eric 42
Becker Boris 49

Curseur – exemple

SERVEUR SGBD (MACHINE X)

BD TENNIS

JOUEUR
<Connors, Jimmy, 1952, USA>
<Fromm, Eric, 1958, USA>
<Becker, Boris, 1967, Allemagne>

CACHE DE LA MACHINE Y

```
CLIENT (MACHINE Y)
exec sql begin declare secti
    varchar nom[21];
    char prenom [16]; Boris
    int age; 49
exec sql end declare section;

int main {
   exec sql declare C cursor
       for select nom, prenom, age from joueur;

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
     printf("%s %s %d\n", nom.arr, prenom.arr, age);
     exec sql fetch from C into :nom, :prenom, :age;
   } /* for */

   exec sql close C;
}
```

```
struct varchar_nom  {
   int len; 6
    char arr[20]; Becker
} nom
```

```
Connors Jimmy 45
Fromm Eric 42
Becker Boris 49
```


Connexion – déconnexion

- Etablir la connexion entre le client et le serveur:
  - **exec sql connect to *target* [*user_name*];**
    - ***target*** peut prendre plusieurs forme (cf. documentation), la plus fréquente étant:
      - **dbname[@hostname][:port]**
    - ***user_name*** peut prendre plusieurs forme (cf. documentation), les plus fréquentes étant:
      - **username | username/password | username IDENTIFIED BY password | username USING password**
- Fermer la connexion avec le serveur:
  - **exec sql disconnect [connection name | DEFAULT | CURRENT | ALL]**
  - Libère les ressources chez le client et le serveur

Exemple complet

```
#include <stdio.h>
exec sql begin declare section;
    varchar nom[21];
    char prenom [16];
    int age;
exec sql end declare section;

exec sql include sqlca;
int main {
   exec sql connect to MaBD@Machine_Serveur
                    user “login" using “Mot de passe”;

   exec sql declare C cursor
       for select nom, prenom from joueur;

   printf("Nom joueur ? "); scanf("%s", &nom.arr);
   nom.len=strlen(nom.arr);

   printf("Prénom joueur ? ");      scanf("%s", &prenom);
   printf("Age joueur ? ");         scanf("%d", &age);
   printf("Nationalité joueur ? "); scanf("%s", &nationalite);

   exec sql insert into joueur values (:nom, :prenom, :age, :nationalite);

   exec sql open C;
   exec sql fetch from C into :nom, :prenom, :age;

   for (;sqlca.sqlcode == 0;){
     printf("%s %s %d\n", nom.arr, prenom.arr, age);
     exec sql fetch from C into :nom, :prenom, :age;
  } /* for */

  exec sql close C;
  exec sql disconnect;
}
```


Programme type

```
/* ==== déclaration éventuelle de variables hôtes ====== */
    exec sql begin declare section;
        V1,
        …..
        VN
    exec sql end declare section;

/* ===== include propre à SQL ===== */
    exec sql include nom_include

/* ===== connexion à la BD ===== */
     exec sql connect to "nom BD";

/* ===== déclaration éventuelle des curseurs ===== */
    exec sql declare nom_curseur cursor  for requête;
      …

/* ===== ouverture éventuelle des curseurs ===== */
    exec sql open nom_curseur ;
```

Programme type (fin)

```
/* ===== positionnement du curseur sur le 1er tuple du resultat = */
     exec sql fetch from nom_curseur into :V1, :V2, …, :VN;

/* == itération sur les tuples du resultat tant que SQLCODE vaut 0 == */

              /* ===== traitement d'un tuple ===== */
                 ……..

                /* === positionnement du curseur sur tuple suivant === */
                 exec sql fetch from nom_curseur into :V1, :V2, …, :VN;

/* ===== fin de l'itération sur les tuples du resultat ============== */

/* ===== fermeture du curseur ===== */
      exec sql close nom_curseur ;

/* ===== déconnexion ===== */
      exec sql disconnect;
```


SQL Dynamique

- **Utilisé lorsque tous les paramètres d'une requête SQL ne sont pas connus au moment de la compilation**
  - Les paramètres manquant seront connus au moment de l'exécution
  - Surtout utile pour les SELECT qui retournent un résultat composé de plusieurs lignes
    - Utilisable néanmoins pour INSERT, UPDATE et DELETE
  - Utilisation de la clause PREPARE

SQL Dynamique - exemple

```
exec sql begin declare section;
    varchar nom[21];
    int age, age_min, age_max;
    char *select1 = "select nom, anneenaissance "
                    " from joueur "
                    " where age >= ? and age <= ?";
exec sql end declare section;

int main {
...
   exec sql prepare st_select1 from :select1;
   exec sql declare C cursor for st_select1;

   printf("Age min du joueur ? "); scanf("%d", &age_min);
   printf("Age max du joueur ? "); scanf("%d", &age_max);

   exec sql open C using :age_min, :age_max;
   exec sql fetch from C into :nom, :age;

   for (;sqlca.sqlcode == 0;){
     printf("%s %d\n", nom.arr, age);
     exec sql fetch from C into :nom, :age;
  } /* for */
  exec sql close C;
...
}
```

Valeurs inconnues au moment de l'écriture de la requête

SQL Dynamique - exemple

```
$ ./consult_bdtennis_dyn
Annee de naissance min du joueur ? 1970
Annee de naissance max du joueur ? 1990

Chang                    | 1972
Bruguera                 | 1971
Courier                  | 1970
(3 ligne(s))

$ ./consult_bdtennis_dyn
Annee de naissance min du joueur ? 2000
Annee de naissance max du joueur ? 2015
(0 ligne(s))
```

Gestion des erreurs

- **Principe**
  - L'envoi d'une requête SQL au serveur provoque en retour la valuation de la variable **`sqlca.sqlcode`**
    - si `sqlca.sqlcode < 0`
      - L'exécution de la requête a échoué
      - La variable `sqlca.sqlerrm.sqlerrmc` contient un message sur la nature du problème ayant causé l'échec de la requête
    - si `sqlca.sqlcode = 0`
      - La requête s'est exécutée sans problème

Gestion des erreurs - exemple

```
// ------------------------------------------------
    void printSqlError (char *userMessage){
// ------------------------------------------------
//  Affiche un message en cas d erreur d exécution
//  d une requête SQL
// ------------------------------------------------
    printf ("%d: %s, %s\n", sqlca.sqlcode, userMessage, sqlca.sqlerrm.sqlerrmc);
} // printSqlError

int main {
...
   exec sql connect to MaBD@Machine_Serveur
                    user "login" using "Mot de passe";
   if (sqlca.sqlcode < 0) {printSqlError("Erreur connect"); exit(EXIT_FAILURE);}
...
   exec sql open C using :age_min, :age_max;
   exec sql fetch from C into :nom, :age;
   if (sqlca.sqlcode < 0) {printSqlError("Erreur 1 exec sql fetch from C"); exit(EXIT_FAILURE);}

   for (;sqlca.sqlcode == 0;){
     printf("%s %d\n", nom.arr, age);
     exec sql fetch from C into :nom, :age;
     if (sqlca.sqlcode < 0) {printSqlError("Erreur 2 exec sql fetch from C"); exit(EXIT_FAILURE);}
  } /* for */
...
}
```

Gestion des erreurs - exemple

```
struct {
    char sqlcaid[8];
    long sqlabc;
    long sqlcode;
    struct { int sqlerrml;
             char sqlerrmc[SQLERRMC_LEN];
    } sqlerrm;
    char sqlerrp[8];
    long sqlerrd[6];
    char sqlwarn[8];
    char sqlstate[5]; } sqlca;
```