TP n° 3 de Bases de Données / ORACLE

Pro*C

1) Travail préliminaire

Copier le fichier `/tmp/proc16.mk` dans votre répertoire courant.

Édition d'un programme Pro*C :

- Écrire le programme dans un fichier de votre répertoire courant et dont le nom doit se terminer par l'extension `.pc`.
- Précompilation, compilation et édition des liens sont réalisées à l'aide de la commande suivante : `make –f proc16.mk` *`nomfichier`* (sans l'extension `.pc`).

La commande `make` réalise une précompilation qui donne un programme source en C nommé *`nomfichier`*`.c`. La compilation produit un fichier objet *`nomfichier`*`.o` et l'édition des liens permet finalement d'obtenir un programme exécutable nommé *`nomfichier`*. *Les programmes exécutables sont volumineux. Évitez de les laisser sur le disque lorsque vous ne vous en servez plus.*

2) Programmation en Pro*C

1) Écrire une routine de connexion à SQLPLUS. En cas d'erreur de connexion, afficher le message d'erreur d'ORACLE.

2) Afficher les renseignements concernant un service dont le numéro est saisi au clavier. Prévoir le cas où le numéro de service est incorrect.

3) Traiter en plus de la question précédente la gestion de données incomplètes pour le service, grâce à des indicateurs.

4) Afficher une liste de toutes les pièces en commande en indiquant la désignation de chaque pièce et le service qui a passé l'ordre. Prévoir tous les cas d'erreur.

5) Changer la couleur des pièces de couleur X en Y. Valider l'opération avec l'instruction COMMIT. Prévoir tous les cas d'erreur, y compris l'annulation de la transaction en cas d'échec du COMMIT.

6) Programmer l'insertion correcte d'une nouvelle pièce (s'assurer que le numéro de pièce n'est pas déjà utilisé dans la table PIECE). Prévoir tous les cas d'erreur.

7) Programmer l'insertion correcte d'un nouvel ordre (s'assurer que la commande n'est pas déjà présente dans la table ORDRE et vérifier la validité du numéro de pièce et du numéro de service). Prévoir tous les cas d'erreur.

8) Présenter le résultat de la requête SQL26 (affichage hiérarchique de la nomenclature) sous forme indentée. (Insérer des tuples supplémentaires dans la table NOMENCLATURE pour obtenir une hiérarchie plus étoffée).

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

/* Declarations des variables globales de communication avec SQLPLUS */

EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR uid[20], pwd[20], ns[3], inti[20], loc[20], np[3], desi[20],
          coulx[15], couly[15];
  int qte, lvl;
  float pds;
  short iinti, iloc;
EXEC SQL END DECLARE SECTION;

/* Inclusion de la zone de communication SQLCA */

EXEC SQL INCLUDE SQLCA.H;

/* Prototype de fonctions */

void connect();
void service();
void commandes();
void couleur();
void ins_piece();
void ins_ordre();
void hierarchie();

void std_err();
void not_found();
void rollbck();

/* Programme principal */

main() {
  connect();
  service();
  commandes();
  couleur();
  ins_piece();
  ins_ordre();
  hierarchie();
  exit(0);
}

/* Connexion */

void connect() {
  char *buff;
  printf("\nConnexion");
  printf("\nLogin : ");
  scanf("%s",uid.arr);
  uid.len=strlen(uid.arr);
  buff=(char *)getpass("Mot de passe : ");
  strcpy(pwd.arr,buff);
  pwd.len=strlen(pwd.arr);
  EXEC SQL WHENEVER SQLERROR DO std_err();
  EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
  printf("\nConnexion..... OK");
}
```

```
/* Renseignements sur un service */

void service() {
  printf("\nRenseignements service");
  printf("\nNumero service : ");
  scanf("%s",ns.arr);
  ns.len=strlen(ns.arr);
  EXEC SQL WHENEVER SQLERROR DO std_err();
  EXEC SQL WHENEVER NOT FOUND DO not_found();
  EXEC SQL SELECT intitule, localisation
           INTO :inti:iinti, :loc:iloc
           FROM service
           WHERE nos=:ns;
  if (iinti!=0 || iloc!=0)
    printf("\nDonnees incompletes...");
  else
    printf("\nIntitule : %s, Localisation : %s",inti.arr,loc.arr);
}

/* Liste des commandes */

void commandes() {
  printf("\nListe des commandes");
  EXEC SQL WHENEVER SQLERROR DO std_err();
  EXEC SQL DECLARE curseur_com CURSOR FOR
    SELECT p.nop, designation, s.nos, intitule, quantite
    FROM ordre o, piece p, service s
    WHERE o.nop = p.nop AND o.nos = p.nos;
  EXEC SQL WHENEVER NOT FOUND DO not_found();
  EXEC SQL OPEN curseur_com;
  printf("\n\n\tnop\tdesignation\tnos\tintitule\tquantite\n");
  EXEC SQL WHENEVER NOT FOUND CONTINUE;
  EXEC SQL FETCH curseur_com INTO :np, :desi, :ns, :inti, :qte;
  while (sqlca.sqlcode!=1403) {
    printf("\n\t%s\t%s\t%s\t%s\t%d",np.arr,desi.arr,ns.arr,inti.arr,qte);
    EXEC SQL FETCH curseur_com INTO :np, :desi, :ns, :inti, :qte;
  }
EXEC SQL CLOSE curseur_com
}

/* Edition de la couleur des pieces ivoires */

void couleur() {
  printf("\nChangement de couleur");
  printf("\nCouleur X : ");
  scanf("%s",coulx.arr);
  coulx.len=strlen(coulx.arr);
  printf("\nCouleur Y : ");
  scanf("%s",couly.arr);
  couly.len=strlen(couly.arr);
  EXEC SQL WHENEVER NOT FOUND DO not_found();
  EXEC SQL UPDATE piece SET couleur=:couly WHERE couleur=:coulx;
  EXEC SQL WHENEVER SQLERROR DO rollbck();
  EXEC SQL COMMIT WORK;
  printf("\nMofification effectuee");
}
```

```
/* Insertion d'une piece */

void ins_piece() {
  printf("\nAjout d'une piece");
  printf("\nNumero : ");
  scanf("%s",np.arr);
  np.len=strlen(np.arr);
  printf("\nDesignation : ");
  scanf("%s",desi.arr);
  desi.len=strlen(desi.arr);
  printf("\nCouleur : ");
  scanf("%s",coulx.arr);
  coulx.len=strlen(coulx.arr);
  printf("\nPoids : ");
  scanf("%f",&pds);
  EXEC SQL WHENEVER NOT FOUND CONTINUE;
  EXEC SQL SELECT * FROM piece WHERE nop=:np;
  if (sqlca.sqlcode!=1403) {
    printf("\nErreur - cle existante !\n");
    return;
  }
  EXEC SQL WHENEVER SQLERROR DO rollbck();
  EXEC SQL INSERT INTO piece VALUES (:np,:desi,:coul,:pds);
  EXEC SQL COMMIT WORK;
  printf("\nPiece ajoutee");
}

/* Insertion d'un ordre */

void ins_ordre() {
  printf("\nAjout d'un ordre");
  printf("\nNumero de piece : ");
  scanf("%s",np.arr);
  np.len=strlen(np.arr);
  printf("\nNumero de service : ");
  scanf("%d",ns.arr);
  ns.len=strlen(ns.arr);
  printf("\nQuantite : ");
  scanf("%s",&qte);
  /* Ctrl d'existance du couple */
  EXEC SQL WHENEVER NOT FOUND CONTINUE;
  EXEC SQL SELECT * FROM ordre WHERE nop=:np AND nos=:ns;
  if (sqlca.sqlcode!=1403) {
    printf("\nErreur - Ce couple de numeros existe deja !\n");
    return;
  }
  /* Ctrl d'existance de la piece */
  EXEC SQL WHENEVER NOT FOUND DO not_found;
  EXEC SQL SELECT * FROM piece WHERE nop=:np;
  /* Ctrl d'existance du service */
  EXEC SQL WHENEVER NOT FOUND DO not_found;
  EXEC SQL SELECT * FROM service WHERE nos=:ns;
  /* Insertion */
  EXEC SQL WHENEVER SQLERROR DO rollbck();
  EXEC SQL INSERT INTO ordre VALUES (:np,:ns,:qte);
  EXEC SQL COMMIT WORK;
  printf("\nOrdre ajoute");
}
```

```
/* Nomenclature hierarchique indentee */

void hierarchie() {
  int i;
  printf("\nNomenclature hierarchique");
  EXEC SQL WHENEVER SQLERROR DO std_err();
  EXEC SQL DECLARE curseur_nom CURSOR FOR
    SELECT level, nopc, quantite FROM nomenclature
    CONNECT BY PRIOR nopc=nopa
    START WITH nopc='P7';
  EXEC SQL WHENEVER NOT FOUND DO not_found();
  EXEC SQL OPEN curseur_nom;
  printf("\n\n\tnop\tquantite\n");
  EXEC SQL WHENEVER NOT FOUND CONTINUE;
  EXEC SQL FETCH curseur_nom INTO :lvl, :np, :qte;
  while (sqlca.sqlcode!=1403) {
    printf("\n\t");
    for (i=0; i<lvl; i++) printf("\t");
    printf("%s\t%d",np.arr,qte);
    EXEC SQL FETCH curseur_nom INTO :lvl, :np, :qte;
  }
  EXEC SQL CLOSE curseur_nom;
}

/* Exception : erreur standard */

void std_err() {
  printf("\nErreur SQLPLUS - %s\n",sqlca.sqlerrm.sqlerrmc);
  exit(1);
}

/* Exception : enregistrement non trouve */

void not_found() {
  printf("\nErreur SQLPLUS – Enregistrement non trouve\n");
  exit(2);
}

/* Exception : annulation */

void rollbck() {
  EXEC SQL WHENEVER SQLERROR CONTINUE;
  EXEC SQL ROLLBACK WORK RELEASE;
  printf("\nErreur - Transaction annulee\n");
  exit(3);
}
```