Prácticas de Estructuras de Datos

David Arroyo Guardeño

Escuela Politécnica Superior de la Universidad Autónoma de Madrid

1. *Introducción*
2. *Normativa*
   - *General*
   - *Requisitos de las entregas*
3. *Introducción a PostgreSQL*
4. *Introducción a ODBC*
   - *Segunda Práctica*
   - *Instalación*
   - *Etapas*
   - *DSN*
   - *Comandos*
   - *Referencias*

- *Ejemplos*
- *Práctica ODBC-II*

Información general

- ✓ Correo $\leftrightarrow$ xxx.yyy@estudiante.uam.es
- ✉ [EDAT]
- ✓ Nota: $0.7 \times N_T + 0.3 \times N_P$
- ☠ COPIAS

Requisitos de las entregas

Retraso Penalización de 1 punto por día de retraso

Entrega moodle Límite: una hora antes del comienzo de la primera sesión de la siguiente práctica

Nombre entregable Gxxx_Pyy_z.tgz
en caso contrario, ↓ 1 punto

Contenido del fichero

1. Fichero de texto: nombre alumnos, email, grupo y fecha
2. Documentación en formato pdf
   - ✗ Respuesta a preguntas breves
   - ✗ Cómo se ha realizado cada ejercicio
   - ✗ Documentación ejercicio final:
3. Listado de código fuente documentado
   - ✗ Programación modular y estructurada
   - ✗ Makefile
   - ☠ Compilación no correcta: 0!!!

Instalación en casa

```
sudo apt-get install postgresql postgresql-contrib
sudo -i -u postgres createuser edat -d -l
sudo -i -u edat createdb -U edat p1_tweet
```

Importar/Exportar en postgreSQL

Exportar: **pg_dump −U alumnodb −d p1_tweet −f p1_tweet.sql**

Importar: **psql −U alumnodb −d p1_tweet −f p1_tweet.sql**

Segunda Práctica

- ✓ Entrega: semana 26-30 octubre antes de la sesión de prácticas correspondiente
- ✓ Refinar el diseño de la base de datos
- ✓ Acceso a la base de datos mediante ODBC

ODBC

- ✓ Open DataBase Connectivity
- ✓ Estándar de acceso a bases de datos

LINUX!!!!!!!!!!!!!!!!!!!

LINUX!!!!!!!!!!!!!!!!

Entrega

- ✓ Makefile: obligatorio $\rightarrow$ condición necesaria para que la práctica sea corregida
- ✓ Código debidamente documentado $\rightarrow$ recomendable el uso de Doxygen, muy recomendable
- ✓ Depuración del código: gdb, ddd
- ✓ Dos recomendaciones
  1. `http://arantxa.ii.uam.es/~darroyo/s1.pdf`
  2. `http://arantxa.ii.uam.es/~darroyo/makefile_comentarios.pdf`

Instalar ODBC en casa

Instalar unixodbc, utilidades y las bibliotecas de desarrollo

Instalar ODBC en casa

Instalar unixodbc, utilidades y las bibliotecas de desarrollo

```
sudo apt-get install unixodbc-bin unixodbc-dev unixodbc
```

Instalar ODBC en casa

Instalar unixodbc, utilidades y las bibliotecas de desarrollo

```
sudo apt-get install unixodbc-bin unixodbc-dev unixodbc
```

Instalar el controlador ODBC para potgresql

Instalar ODBC en casa

Instalar unixodbc, utilidades y las bibliotecas de desarrollo

```
sudo apt-get install unixodbc-bin unixodbc-dev unixodbc
```

Instalar el controlador ODBC para potgresql

```
sudo apt-get install odbc-postgresql
```

Instalar ODBC en casa

Instalar unixodbc, utilidades y las bibliotecas de desarrollo

```
sudo apt-get install unixodbc-bin unixodbc-dev unixodbc
```

Instalar el controlador ODBC para potgresql

```
sudo apt-get install odbc-postgresql
```

Etapas transacción ODBC

1. Localización del controlador (*driver*) adecuado
2. Carga del *driver*
3. Carga del entorno y del manejador del driver
4. Conexión con la fuente de datos
5. Ejecución de la consulta

Etapas transacción ODBC

1. Indicar al gestor del controlador ODBC el entorno y el manejador de la conexión
   - **SQLAllocHandle**, dos invocaciones
   - Primera llamada: inicialización de la interfaz ODBC y reserva de memoria para información global
   - Segunda llamada: reserva de memoria para el manejador de conexión y la información asociada
2. Localización del controlador (*driver*) adecuado
3. Carga del *driver*
4. Carga del entorno y del manejador del

Etapas transacción ODBC

1. Indicar al gestor del controlador ODBC el entorno y el manejador de la conexión
2. Localización del controlador (*driver*) adecuado
   - ✗ **SQLConnect**
3. Carga del *driver*
4. Carga del entorno y del manejador del driver
5. Conexión con la fuente de datos
6. Ejecución de la consulta

Etapas transacción ODBC

1. Localización del controlador (*driver*) adecuado
2. Carga del *driver*
3. Carga del entorno y del manejador del driver
4. Conexión con la fuente de datos
5. Ejecución de la consulta
   - ✗
     ```
     SQLAllocHandle(SQL_HANDLE_STMT, dbc,
     ```
   - ✗
     ```
     SQLExecDirect(stmt, consulta, SQL_NTS);
     ```

Control de errores

SQLRETURN ret; SQL_SUCCEEDED(ret)

Control de errores

SQLRETURN ret; SQL_SUCCEEDED(ret)

```
SQLRETURN ret = SQL_SUCCESS;
cliRC = SQLAllocHandle(SQL_HANDLE_ENV,
    SQL_NULL_HANDLE,
&env);
if (ret != SQL_SUCCESS)
{
printf("\n--ERROR while allocating the
    environment handle.\n");
printf(" ret = %d\n", r);
printf(" line = %d\n", __LINE__);
printf(" file = %s\n", __FILE__);
return 1;
}
```

Control de errores

SQLRETURN ret; SQL_SUCCEEDED(ret)

```
SQLRETURN ret = SQL_SUCCESS;
cliRC = SQLAllocHandle(SQL_HANDLE_ENV,
    SQL_NULL_HANDLE,
&env);
if (ret != SQL_SUCCESS)
{
printf("\n--ERROR while allocating the
    environment handle.\n");
printf(" ret = %d\n", r);
printf(" line = %d\n", __LINE__);
printf(" file = %s\n", __FILE__);
return 1;
}
```

odbc.h en los ejemplos:

void odbc_extract_error(char *fn, SQLHANDLE handle, SQLSMALLINT type);

Liberar recursos

```
int odbc_disconnect(SQLHENV env, SQLHDBC dbc) {
    SQLRETURN ret;
    /* Disconnect from database */
    ret = SQLDisconnect(dbc);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", dbc, SQL_HANDLE_DBC);
        return ret;
    }
    /* Free connection handle */
    ret = SQLFreeHandle(SQL_HANDLE_DBC, dbc);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", dbc, SQL_HANDLE_DBC);
        return ret;
    }
    /* Free environment handle */
    ret = SQLFreeHandle(SQL_HANDLE_ENV, env);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", env, SQL_HANDLE_ENV);
        return ret;
    }
    return ret;
}
```

Liberar recursos

```
int odbc_disconnect(SQLHENV env, SQLHDBC dbc) {
    SQLRETURN ret;
    /* Disconnect from database */
    ret = SQLDisconnect(dbc);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", dbc, SQL_HANDLE_DBC);
        return ret;
    }
    /* Free connection handle */
    ret = SQLFreeHandle(SQL_HANDLE_DBC, dbc);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", dbc, SQL_HANDLE_DBC);
        return ret;
    }
    /* Free environment handle */
    ret = SQLFreeHandle(SQL_HANDLE_ENV, env);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", env, SQL_HANDLE_ENV);
        return ret;
    }
    return ret;
}
```

ret = SQLDisconnect(dbc);

Liberar recursos

```
int odbc_disconnect(SQLHENV env, SQLHDBC dbc) {
    SQLRETURN ret;
    /* Disconnect from database */
    ret = SQLDisconnect(dbc);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", dbc, SQL_HANDLE_DBC);
        return ret;
    }
    /* Free connection handle */
    ret = SQLFreeHandle(SQL_HANDLE_DBC, dbc);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", dbc, SQL_HANDLE_DBC);
        return ret;
    }
    /* Free environment handle */
    ret = SQLFreeHandle(SQL_HANDLE_ENV, env);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", env, SQL_HANDLE_ENV);
        return ret;
    }
    return ret;
}
```

ret = FreeHandle(SQL_HANDLE_DBC, dbc);

Liberar recursos

```
int odbc_disconnect(SQLHENV env, SQLHDBC dbc) {
    SQLRETURN ret;
    /* Disconnect from database */
    ret = SQLDisconnect(dbc);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", dbc, SQL_HANDLE_DBC);
        return ret;
    }
    /* Free connection handle */
    ret = SQLFreeHandle(SQL_HANDLE_DBC, dbc);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", dbc, SQL_HANDLE_DBC);
        return ret;
    }
    /* Free environment handle */
    ret = SQLFreeHandle(SQL_HANDLE_ENV, env);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", env, SQL_HANDLE_ENV);
        return ret;
    }
    return ret;
}
```

ret = SQLFreeHandle(SQL_HANDLE_ENV, env);

Liberar recursos

```
int odbc_disconnect(SQLHENV env, SQLHDBC dbc) {
    SQLRETURN ret;
    /* Disconnect from database */
    ret = SQLDisconnect(dbc);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", dbc, SQL_HANDLE_DBC);
        return ret;
    }
    /* Free connection handle */
    ret = SQLFreeHandle(SQL_HANDLE_DBC, dbc);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", dbc, SQL_HANDLE_DBC);
        return ret;
    }
    /* Free environment handle */
    ret = SQLFreeHandle(SQL_HANDLE_ENV, env);
    if (!SQL_SUCCEEDED(ret)) {
        odbc_extract_error("", env, SQL_HANDLE_ENV);
        return ret;
    }
    return ret;
}
```

```
ret = SQLFreeHandle(SQL_HANDLE_STMT, stmt);
```

Primer paso: creación de una fuente de datos

- ✓ DSN: Data Source Name
- ✓ Si está instalado unixodbc-bin: **ODBCCreateDataSourceQ4** (para ver todas las fuentes existentes, **ODBCManageDataSourceQ4**)
- ✓ Alternativa: crear un fichero .odbc.ini en el directorio *home* del usuario
  - ✗ **cd** $\rightarrow$ nos lleva a nuestro directorio *home*
  - ✗ **cd ~** $\rightarrow$ nos lleva a nuestro directorio *home*
  - ✗ **touch .odbc.ini**
  - ✗ **gnome−open .odbc.ini**

```
[p1_tweet]
Description             = PostgreSQL Unicode
Driver          = PostgreSQL Unicode
Trace           = No
TraceFile               =
Database                = p1_tweet
Servername              = localhost
Username                = alumnodb
Password                = alumnodb
Port            = 5432
Protocol                = 6.4
ReadOnly                = No
RowVersioning                   = No
ShowSystemTables                = No
ShowOidColumn                   = No
FakeOidIndex                    = No
ConnSettings                    =
```

Comandos útiles

- ✓ **odbcinst −q −d**: Lista todos los controladores ODBC instalados
- ✓ **odbcinst −q −s**: Lista todas las fuentes de datos definidas
- ✓ **isql −v p1_tweet**: Para conectarse a la fuente de datos

Enlaces útiles I

- ✓ `http://www.unixodbc.org/`
- ✓ Doxygen documentation: `https://fossies.org/dox/unixODBC-2.3.4/index.html`
- ✓ `http://www.easysoft.com/developer/languages/c/index.html`

Enlaces útiles II

- http://www.easysoft.com/developer/languages/c/examples/index.html
- https://msdn.microsoft.com/en-us/library/ms716298(v=vs.85).aspx
- Conversión de tipos de datos desde C a SQL: https://msdn.microsoft.com/es-es/library/ms716298(v=vs.85).aspx

Makefile ejemplo

```
CC = gcc
CFLAGS = -Wall -Wextra -pedantic -ansi
LDLIBS = -lodbc

EXE = odbc-test odbc-example1 odbc-example2 odbc-
    example3 odbc-example4

all : $(EXE)

clean :
    rm -f *.o core $(EXE)

$(EXE) : % : %.o odbc.o
```

Muestra las fuentes de datos instaladas

```
#include <stdio.h>
#include <sql.h>
#include <sqlext.h>

main() {
  SQLHENV env;
  char dsn[256];
  char desc[256];
  SQLSMALLINT dsn_ret;
  SQLSMALLINT desc_ret;
  SQLUSMALLINT direction;
  SQLRETURN ret;

  SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &env);
  SQLSetEnvAttr(env, SQL_ATTR_ODBC_VERSION, (void *) SQL_OV_ODBC3, 0);

  direction = SQL_FETCH_FIRST;
  while(SQL_SUCCEEDED(ret = SQLDataSources(env, direction,
                        dsn, sizeof(dsn), &dsn_ret,
                        desc, sizeof(desc), &desc_ret))) {
    direction = SQL_FETCH_NEXT;
    printf("%s - %s\n", dsn, desc);
    if (ret == SQL_SUCCESS_WITH_INFO) printf("\tdata truncation\n");
  }
}
```

Muestra los controladores ODBC instalados

```
#include <stdio.h>
#include <sql.h>
#include <sqlext.h>

main() {
  SQLHENV env;
  char driver[256];
  char attr[256];
  SQLSMALLINT driver_ret;
  SQLSMALLINT attr_ret;
  SQLUSMALLINT direction;
  SQLRETURN ret;

  SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &env);
  SQLSetEnvAttr(env, SQL_ATTR_ODBC_VERSION, (void *) SQL_OV_ODBC3, 0);

  direction = SQL_FETCH_FIRST;
  while(SQL_SUCCEEDED(ret = SQLDrivers(env, direction,
                       driver, sizeof(driver), &driver_ret,
                       attr, sizeof(attr), &attr_ret))) {
    direction = SQL_FETCH_NEXT;
    printf("%s - %s\n", driver, attr);
    if (ret == SQL_SUCCESS_WITH_INFO) printf("\tdata truncation\n");
  }
}
```

¿Qué debo hacer para la segunda parte de la práctica 2? I

- ✓ Cuatro programas (+ optativo)
  1. usuario

     inserción de un nuevo usuario

     usuario + <screen_name> “<full

     eliminación de un usuario

     usuario − <screen_name>
  2. tweet →

     tweet <screen_name> “<tweet>”
  3. retweet →

     retweet <screen_name> <tweet_id>
  4. reply → **reply <screen_name> “<text>”**

¿Qué debo hacer para la segunda parte de la práctica 2? II

- ✓ Crear sentencias SQL de inserción (**INSERT INTO ... VALUES ...**)
- ✓ Crear sentencias SQL de elimación de registros (**DELETE FROM ... WHERE...**)
- ✓ Leer/interpretar con detenimiento la biblioteca odbc que se da como ejemplo
- ✓ Sustituid la constante **CONNECTION_PARS** en odbc.h
- ✓ Buscad las líneas que determinan la consulta SQL en los ejemplos
- ✓ En odbc-example2.c

```
SQLExecDirect(stmt, (SQLCHAR*)"select x, y from a", SQL_NTS);
```

- ✓ En odbc-example3.c

```
sprintf(query, "select y from a where x = %s;", x);
```

- ✓ En odbc-example4.c

```
SQLPrepare(stmt, (SQLCHAR*)"select y from a where x = ?", SQL_N
```

```
SQLBindParameter(stmt, 1, SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0, &x, 0,
```