Una Comparación de Interfaces de Programación Microsoft para el Acceso a Bases de Datos

Gustavo Larriera

Julio, 1998.

Resumen. Los desarrolladores de aplicaciones Windows disponen de varias tecnologías para acceder a diferentes fuentes de datos. Tales tecnologías presentan diferentes funcionalidades, eficiencia, arquitectura y se adaptan de mejor o peor manera según el tipo de aplicación que se esté desarrollando. En este artículo se describe al conjunto de tecnologías Microsoft disponibles, se detallan sus principales características, se las compara entre sí y se brindan sugerencias de aplicación para cada una de ellas.

Palabras clave: *Acceso a bases de datos; arquitectura cliente/servidor; conectividad; ODBC; API; bases de datos; DBMS; SQL Server; Visual Basic.*

Contenido

1 Introducción

Los desarrolladores de aplicaciones de bases de datos para Microsoft Windows disponen de un conjunto variado de tecnologías que permiten a una aplicación acceder a los datos almacenados. Los datos pueden estar en diversos formatos, incluyendo a datos SQL, en forma local y/o remota. Estas tecnologías han ido evolucionando con el tiempo, algunas sustituyendo a otras, aunque actualmente coexisten una media docena. Cada una de tales tecnologías tienen su propia idiosincracia y posibilidades, según sea el tipo de aplicación que se desee desarrollar y el tipo de datos a los que se desee acceder.

En las siguientes secciones se describirá cada una de las tecnologías de acceso a bases de datos disponibles en el entorno de Microsoft Windows, y se detallarán sus principales características, ventajas y desventajas de uso. Luego se realizará una comparación entre ellas. Finalmente se sugieren algunas recomendaciones para los desarrolladores que deban elegir una o varias tecnologías de acceso a bases de datos.

2 Interfaces de Programación

Las interfaces de programación que permiten a una aplicación Windows conectarse a datos externos pueden categorizarse en tres: (a) Interfaces a bases de datos basadas en archivos de datos; (b) Interfaces a sistemas gerenciadores de bases de datos; y (c) Interfaces a otros formatos de datos no almacenados en servidores de bases de datos.

Las interfaces a bases de datos basadas en archivos permiten a una aplicación acceder a los datos mediante operaciones sencillas de E/S o a través de métodos indexados (ISAM). La JET (Joint Engine Technology) es un intento de estandarización de acceso a diferentes ISAMs, utilizada por Access y Visual Basic.

Las interfaces a DBMSs pueden implementarse mediante APIs propietarias que permitan acceder a las funcionalidades del servidor de bases de datos. La tecnología ODBC se propone como una solución

abierta para conectar una aplicación a bases de datos SQL, en forma interoperable. La solución ODBC se considera industrialmente madura y estable.

Para un nuevo tipo de aplicaciones que requieren formas de acceso homogéneas para datos no necesariamente residentes en bases de datos y distribuídos en una red, Microsoft está trabajando en la especificación de OLE DB. OLE DB es una extensión de OLE (Object Linking and Embedding) que permite acceder a datos de un servidor OLE así como de servidores de bases de datos, en forma indistinta.

3 Tecnologías de Conectividad a Datos

En esta sección se describen las principales tecnologías mediante los cuales una aplicación puede conectarse a los datos externos almacenados en una base de datos, ya sea basada en ISAM o en un motor SQL.

- **JET.** El motor JET es el manejador de datos ISAM disponible en Access y Visual Basic. Incluye un procesador de consultas, que bajo ciertas circunstancias puede ser salteado si se desea utilizar el procesador nativo de otro DBMS. La manipulación del motor JET se realiza mediante dos interfaces de programación: DAO (Data Access Objects) y RDO (Remote Data Objects) que se describirán más adelante.
- **DBLIB y SQL-DMO.** DBLIB es el protocolo nativo de SQL Server, implementado bajo la forma de una API con funciones para acceder a los datos almacenados en SQL Server. En Visual Basic, el control VBSQL es la implementación de DBLIB. SQL-DMO (Data Maintenance Objects) actúa como una interfaz sobre las funciones de administración disponibles en SQL Server. SQL-DMO dispone de controles para manipular a las tablas, procedimientos almacenados, vistas, triggers y propiedades de configuración de SQL Server.
- **ODBC.** ODBC es un estándar abierto que permite conectividad y uso de un SQL estándar (SQL-ODBC) de forma tal que una aplicación se comunique con un ambiente heterogéneo de DBMSs. Si bien una aplicación se puede comunicar directamente con las funciones de la ODBC API, ODBC puede verse como una base de bajo nivel con la cual se comunican objetos definidos en la interfaz DAO y RDO.
- **OLE DB.** OLE DB es una de las tecnologías más nuevas de Microsoft, surgida como una mezcla de OLE y ODBC. La utilización de OLE DB permite a una aplicación comunicarse con datos ODBC (residentes por ejemplo en DBMSs) y datos OLE (planillas, documentos, etc.) en forma uniforme.

4 Acceso Programático mediante DAO, RDO y ADO

Los desarrolladores de aplicaciones Windows disponen de varias alternativas que les permiten acceder a fuentes de datos. En esta sección se describen: DAO, RDO y ADO.

4.1 DAO (Direct Access Objects)

DAO es la interfaz de programación disponible para comunicarse con JET. DAO se utiliza en contextos donde el motor reside en forma local y cuando se basan en ISAM. Sin embargo, DAO puede abrir datos remotos ODBC pasando a través de JET, aunque con grandes limitaciones de performance. Estas limitaciones se deben fundamentalmente a que JET es ineficiente en la negociación de conexiones y a que no realiza caché de los datos ODBC. Una actualización posterior, denominada ODBCDirect, permite acceder a funcionalidades de ODBC, salteando la capa del JET. Este enfoque es útil para aquellas aplicaciones que usen indistintamente datos ISAM y datos SQL. Cuando solamente se va a acceder a datos SQL, RDO es una mejor alternativa. Como dato importante, DAO está siendo discontinuado por Microsoft.

4.2 RDO (Remote Data Objects)

RDO es una capa que abstrae los detalles de la ODBC API, utilizada únicamente en aplicaciones de 32-bit. RDO es una alternativa eficiente para trabajar sobre ODBC (es decri, sobre datos remotos), y con varias

ventajas funcionales frente a DAO: ejecución de stored procedures del servidor de base de datos, ejecución asincrónica de consultas y procesamiento de múltiples tipos de cursores. La principal ventaja de RDO es que fue diseñado para el uso eficiente de las funcionalidades de ODBC. Como principales desventajas frente a DAO, RDO sólo dispone de conexión a datos ODBC a través de drivers de 32-bit y no está disponible en el Visual Basic for Applications disponible en Office 97. Esto último es una limitación fuerte para los desarrolladores de aplicaciones Office.

4.3 ADO (ActiveX Data Objects)

ADO es la interfaz a OLE DB y debe verse como un modelo de programación, más que como una implementación específica. La aparición de ADO surge de la necesidad de acceder a datos en otros formatos (no necesariamente datos en bases de datos) en forma remota y con cierto énfasis en el entorno de Intranet/Internet. Esto último se logra a través de una interfaz al modelo de objetos distribuídos (DCOM, Distributed Common Object Model, una extensión de OLE a ambiente distribuído). Si bien ADO aún no es una tecnología estable, Microsoft apunta a consolidarla como "su" tecnología futura de acceso a datos.

4.4 Ejemplos

En esta sección se brindan algunos ejemplos de programación usando los distintos métodos de acceso, a los efectos de mostrar el uso en un lenguaje de desarrollo, como Visual Basic.

Ejemplo 1. DAO

```
'Creacion del espacio de trabajo de JET
Dim ws as Workspace
Set ws = CreateWorkspace("", "loginid", "passwd", dbUseJet)
'Conexion a una base de datos JET
Dim db as Database
Set db = ws.OpenDatabase("C:\data\northwind.mdb")
'Abrir un cursor
Dim rs as RecordSet
Set rs = db.OpenRecordset("Empleados", dbOpenDynaset, dbReadonly)
'Procesamiento
…
'Cierre y desconexion
rs.Close
db.Close
ws.Close
```

Ejemplo 2. DAO + ODBCDirect

```
'Creacion del espacio de trabajo
Dim ws as Workspace
Set ws = CreateWorkspace("", "loginid", "passwd", dbUseODBC)
'Conexion a una base de datos ODBC
Dim cn as Connection
Set cn = ws.OpenConnection("",,,"ODBC; DATABASE=pubs; UID=sa; PWD=; DSN=pubsdata")
'Abrir un cursor
Dim rs as RecordSet
Set rs = db.OpenRecordset("Empleados", dbOpenDynamic)
'Procesamiento
…
'Cierre y desconexion
rs.Close
db.Close
ws.Close
```

Ejemplo 3. RDO

```
'Conexion
Set cn = New rdoConnection
With cn
        .Connect = "DSN=pubsdata; UID=sa; PWD="
        .EstablishConnection
End With
'Abrir un cursor
Dim rs as rdoresultset
Set rs = cn.OpenResultSet("Empleados", rdOpenKeyset, rdConcurReadOnly)
```

Ejemplo 4. DAO

```
'Abrir un cursor
Dim rs As New ADODB.Recordset
Rs.Open "Empleados; DSN=pubsdata; UID=sa; PWD="
'Procesamiento del cursor
Do Until rs.EOF
        Print rs!apellido
        Rs.MoveNext
Loop
Es.Close
```

5 Comparación de métodos de acceso programático

En esta sección se comparan ventajas y desventajas de cada uno de los 3 métodos de acceso programático descriptos anteriormente.

METODO	VENTAJAS	DESVENTAJAS
DAO	• Interfaz consolidada y estable de JET • Eficiente en ISAM • ODBCDirect para acceso a ODBC sin usar JET	• Ineficiente en ODBC • En retroceso por parte de Microsoft
RDO	• Optimizado para ODBC • Compatibilidad con ISAM	• Disponible sólo para plataformas 32-bit • No disponible para aplicaciones de Office 97.
ADO	• Superset de RDO • Acceso a otros formatos de datos (planillas, documentos, etc.) • Interfaz a aplicaciones que usen DCOM • Apoyo muy visible de Microsoft	• Especificación aún en elaboración • Implementación aún no consolidada

6 Conclusiones

Los desarrolladores de aplicaciones Windows disponen de diversas tecnologías y métodos de acceso a bases de datos. Resulta importante que en la elección de una alternativa se consideren los siguientes aspectos:

- **Formato de los datos a los que se desea acceder.** Cada interfaz ha sido diseñada teniendo en mente un formato específico de datos, lo cual hace que una interfaz sea ineficiente cuando trata de

acceder a datos en otro formato. Por ejemplo, DAO soporta ISAM mientras que RDO está orientada a ODBC.

- **Accesos heterogéneos.** En caso de que la aplicación deba acceder en forma heterogénea usando ISAM y ODBC, se podría usar una combinación de DAO y RDO por ejemplo, o simplemente ADO con ODBCDirect.
- **Ciclo de vida de la aplicación.** Para aplicaciones de vida corta se recomienda usar el método más eficiente. Para aplicaciones pre-existentes, tal vez sea aconsejable continuar usando DAO aún cuando se le agreguen a la aplicación requerimientos ODBC.
- **Usos específicos.** Para aplicaciones diseñadas específicamente para bases de datos propietarias y donde la conexión universal no es importante, se recomienda usar APIs propietarias. En el caso de SQL Server como DBMS target, se puede usar DBLIB y SQL-DMO.

Bibliografía

- Microsoft Windows Architecture for Developers, Student Workbook. Microsoft, 1997.
- Road map of Database Technologies. Por Mark Gendron. Microsoft, 1996.
- Readings on Microsoft Windows & WOSA. Microsoft, 1995.
- ODBC 3.0 System Development Kit. Microsoft, 1997.
- Visual Basic 5.0 Books Online. Microsoft, 1997.
- Mastering Visual Basic 5 Instructor's Guide. Microsoft, 1997.
