InterBase olé!

Vier native OLE DB-Provider im Vergleich

von Karsten Strobel

Auch in der IT-Branche ist es derzeit nicht ganz ungewöhnlich, dass sich auf eine zu besetzende Stelle vier Bewerber mit angeblich identischer Qualifikation melden. Wenn die eingereichten Unterlagen auf den ersten Blick vergleichbar erscheinen, die Spanne zwischen der niedrigsten und der höchsten Gehaltsvorstellung mehrere hundert Prozent beträgt und einer sogar zusagt, den Dienst ganz ohne Bezahlung anzutreten, ist die Wahl ganz einfach. Oder vielleicht doch nicht?

Microsofts OLE DB-Architektur und der dazu passende Überbau ADO werden im Allgemeinen als Nachfolgetechnologie der inzwischen etwas angegrauten ODBC-Treiber betrachtet. Sie sind ein Kernstück der Universal Data Access Strategie (UDA) des Windows-Monopolisten, wurden maßgeblich durch die Einführung beim Microsofts SQL Server gefördert und werden inzwischen auch von etlichen anderen Datenbankherstellern unterstützt. Bei ODBC spricht man von Treibern, bei OLE DB bezeichnet man die Funktion einer DLL, die den Zugriff zu einer speziellen Datenbank ermöglicht, als Provider. Gemeint ist eigentlich das gleiche, doch bekanntlich ist eine neue Technologie in Marketingkreisen undenkbar, führt man nicht gleichzeitig ein ganzes Kompendium neuer Begriffe und Abkürzungen ein.

OLE DB hat sich inzwischen erheblich verbreitet und wird nicht nur von Microsofts Programmiersprachen, sondern immer mehr auch von Anwendungssoftware, Serverprodukten und Scriptsprachen als produktneutrale Middleware zum Zugriff auf relationale und dateibasierte Datenbanken verwendet. Auch Borlands Delphi und C++Builder unterstützen ADO und damit den Weg über OLE DB zur Datenbank. Für das hauseigene InterBase gibt es von Borland allerdings keinen eigenen OLE DB-Provider. Zwar kann man mit Microsofts „OLE DB-Provider für ODBC-Treiber" den von Borland vertriebenen InterBase-ODBC-Treiber einbinden, muss dabei aber einige unschöne Einschränkungen in Kauf nehmen.

Als Purist wünscht man sich natürlich einen nativen, funktionsreichen und vor allem stabilen Provider für InterBase. Bei meiner Suche im Internet habe ich immerhin vier verschiedene Angebote dieser Art gefunden, die ich „angetestet" habe und hier vorstellen möchte. Bezugsadressen und Preise finden Sie im Kasten „OLE DB-Provider". Die Präsentation der vier Produkte im Web rangiert von professionell bis hobbyistisch. Zwei der Anbieter verzichten leider ganz darauf, ein ordentliches Impressum in ihre Webseiten aufzunehmen, weswegen ich nähere Angaben über Autoren bzw. Hersteller ausgelassen habe. Die Spanne der Lizenzkosten reicht von Free- über Shareware-Niveau bis zu jährlichen Laufzeitlizenzen mit ausgewachsenen Preisen.

Einfache Selbstmontage

Die Installation erfolgt immer auf die gleiche Weise und ist auch „von Hand" herzlich unkompliziert, weshalb alle Anbieter zurecht auf die Lieferung eines Installationsprogramms verzichten: Das jeweilige Downloadarchiv enthält eine oder zwei DLL-Dateien, die man in ein im Suchpfad aufgeführtes Verzeichnis (im Allgemeinen *\Windows\System32*) kopiert und dann vom Kommando prompt aus *regsvr32 dateiname.dll* aufruft, womit der Provider in der Registry registriert und damit für das System bekannt gemacht wird. Damit ist die Installation bereits abgeschlossen. Die einfachste Art, das grundsätzliche Funktionieren des Providers zu überprüfen, ist nun, eine *.udl*-Datei anzulegen, zu parametrieren und einen Verbindungstest zu machen. Voraussetzung – neben einer InterBase Client-Installation – ist, dass die Microsoft Data Access Components (MDAC) installiert sind, was bei aktuellen Betriebssystemversionen regelmäßig schon der Fall ist. Aktuell steht MDAC in der Version 2.7 zur Verfügung und kann kostenlos von der Microsoft Website [1] heruntergeladen werden.

Legen Sie einfach mit *Notepad.exe* eine neue, völlig leere Datei an und benennen Sie diese dann in *name.udl* um. Per Doppelklick öffnen Sie dann das von MDAC bereitgestellten Dialogfenster, wählen den Provider aus, stellen die Verbindungsparameter zu einer beliebigen InterBase-Datenbank ein und betätigen die Schaltfläche VERBINDUNG TESTEN (siehe Abb. 1). Alle vier Kandidaten nehmen diese erste Hürde mühelos. Wenn Sie alle vier Provider installiert haben und die im *.udl*-Dialog jeweils angebotenen Parameter miteinander vergleichen, stellen Sie die ersten Unterschiede fest, die schon erwarten lassen, dass

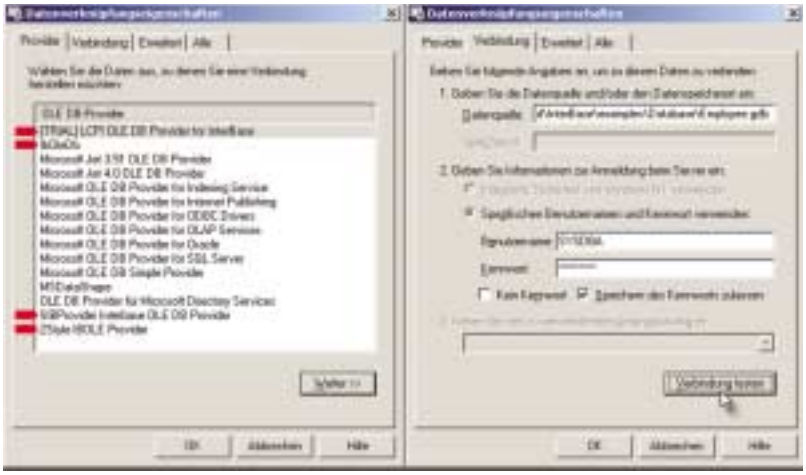

Abb. 1: UDL-Datei Editor

die Provider trotz der gattungsmäßigen Gleichheit doch verschiedene Fähigkeiten und Eigenschaften zeigen werden.

Für meine Versuche mit den OLE DB-Providern habe ich drei verschiedene Ansätze verfolgt: Erstens habe ich die Managementoberfläche ADOAnywhere [2] verwendet – ein Shareware-Tool, mit dem über ADO/OLE DB auf beliebige Datenbanken zugegriffen werden kann und das ungefähr die Funktionalität von Borlands BDE-Datenbankoberfläche besitzt. Damit lassen sich einige grundlegende Versuche machen und wenn Sie mit ADO entwickeln, kann ich dieses Werkzeug, das mit einem Lizenzpreis von $10 sehr günstig ist, empfehlen. Um etwas mehr praktische Erfahrung zu sammeln, habe ich im zweiten Schritt mit Delphi ein Testprogramm geschrieben, um bestimmte Funktionalitäten (Transaktionen, Blob-Handling, Cursors etc.) und auch die Performance gezielter zu untersuchen. Dabei habe ich auf etliche Beispiele aus Andreas Koschs Buch *ADO und Delphi* [3] zurückgegriffen – ebenfalls eine Empfehlung, wenn Sie dieses Gespann bei Ihre Entwicklung einsetzen. Schließlich habe ich drittens ein kleines, auf IBX basierendes Delphi-Projekt (nämlich das mit Delphi gelieferte MASTAPP-Beispielprogramm) auf ADO umgestellt und mit den verschiedenen Providern getestet.

OLE DB-Provider

IBProvider

Internet: www.lcpi.lipetsk.ru/prog/eng/start/ibprov_ib.html
Dateiname: *_IBProvider_trial_d.dll* und *cc3250mt.dll*
Aktuelle Version: 1.7.6.350 von August 2002
Lizenzkosten zwischen einmalig $150 („Company“) und jährlich $600 („Corporate“).

zstyle

Internet: www.zstyle.dp.ua/eng/index.htm
Dateiname: *ibole.dll*
Aktuelle Version: 4.25 von August 2002
Lizenzkosten abhängig von Benutzeranzahl, ab $35.

IbOleDb

Internet: www.oledb.net/
Dateiname: *IbOleDb.dll*
Aktuelle Version: 1.1 von April 2002
Lizenzkosten: keine, Freeware

SIBProvider

Internet: www.sibprovider.com/en_us/default.asp
Dateiname: *SIBPRO2.dll*
Aktuelle Version: 2.2 von August 2002
Lizenzkosten abhängig von Benutzeranzahl, ab $18.

IBProvider

Das erste Produkt trägt den schlichten aber zutreffenden Namen „InterBase OLE DB Provider“ oder kurz „IBProvider“ und wird kommerziell vermarktet. Es gibt auch eine ältere, kostenlose Version, die ich aber hier nicht berücksichtige. Die (leider anonyme) Webseite in schlichtem Layout hebt sich ab durch eine brauchbare Online-Dokumentation, Anwendungsbeispiele in diversen Programmiersprachen und eine umfangreiche Linksammlung zum Thema, bei der sogar die Konkurrenz nicht außen vor bleibt. Die Selbstdarstellung weist besonders auf die Features Unicode-Unterstützung, umfangreiche Typen- und Schemaimplementierung sowie Blob- und Array-Verarbeitung hin.

Falls Sie mit diesem Provider anfangs bei jedem Datenzugriff die Fehlermeldung *Automatic transaction is disabled* erhalten, brauchen Sie die Flinte nicht gleich ins Korn zu werfen. Sobald Sie die Eigenschaft *auto_commit* auf *True* einstellen, wird (wie sonst auch üblich) bei Bedarf automatisch eine Transaktion gestartet. Nur wenn Sie den Default-Wert *False* belassen, muss für jeden Zugriff eine explizite Transaktion begonnen und beendet werden.

Ansonsten gibt es mit diesem Provider kaum Schwierigkeiten. Die typischen Schwachstellen anderer Provider wie der Blob-Zugriff oder das Ausführen Gespeicherter Prozeduren mit Return-Parametern sind für diesen Kandidaten kein Problem. IBProvider implementiert als einziger die Schema-Informationen so vollständig, dass es ADOAnywhere möglich ist, die Struktur der Beispieldatenbank *Employee.gdb* einschließlich der durch Linien symbolisierten Foreign-Key-Beziehungen darzustellen (siehe Abb. 2).

Vorsicht: Nach Ablauf der Trial-Periode verweigert dieser Provider den Dienst und verursacht die schwer verständliche Meldung *Unbekannter Fehler*, anstatt konkret auf die abgelaufene Probezeit hinzuweisen.

zstyle

Ein ebenfalls kommerziell ausgerichtetes Produkt wird von der Firma zstylegroup angeboten. Sympathisch war mir auf Anhieb das Incentive-Angebot an passionierte Kammerjäger: Wer drei Bugs in der Software findet, bekommt eine zeitlich befristete Lizenz kostenlos. Um diese Offerte und andere Informationen im Internet lesen zu können, muss man etwas Geduld aufbringen, wenn die Anbindung der zstyle-Website ist offenbar sehr langsam.

Die Trial-Version weist bei jeder neuen Datenbanksitzung auf die Restlaufzeit der Testversion hin und fordert nervtötend dazu auf, doch endlich den Entschluss zum Lizenzieren zu fassen. Leider gibt es einige Mängel, die vor einem

schnellen Kauf zurückschrecken lassen: Zumindest in ADOAnywhere werden Text-Blobs nicht korrekt angezeigt; es erscheinen nur Fragezeichen. Der Versuche, eine Gespeicherte Prozedur asynchron auszuführen, führt nicht zu der zu erwartenden Meldung, dass dies nicht unterstützt wird, sondern zu einer Schutzverletzung.

Einschränkungen gibt es auch bei den Cursors-Typen. So wird *Forward Only* leider nicht unterstützt; der Provider ändert den angeforderten Typ gegebenenfalls selbstständig in *Keyset* um, womit ein ressourcenschonendes Vorwärts-Lesen einer großen Datenmenge leider nicht möglich ist.

IbOleDb

Der Freeware-Kandidat IbOleDb verweigert schon bei den ersten Schritten den Gehorsam. Sehr viele Operationen, die von der Konkurrenz ohne Klagen ausgeführt werden, quittiert IbOleDb nur lakonischen mit *E_FAIL*, was soviel heißt wie „ich würde ja gerne, kann aber irgendwie nicht". Auch andere unschöne Erscheinungen wie Schutzverletzungen sind eher die Regel als die Ausnahme. Ein Arbeiten mit ADOAnywhere ist praktisch unmöglich.

Es ist schon fast eine Kunst, mit diesem Provider überhaupt etwas aus einer InterBase-Datenbank zu lesen. Aus diesem Grund konnte ich mit dieser Software nur sehr eingeschränkt testen. Überraschend dagegen, dass ausgerechnet dieser Wackelkandidat als einziger das asynchrone Ausführen von Abfragen unterstützt. Der Provider ist leider insgesamt so instabil, dass man sich über diesen Vorteil nicht so richtig freuen kann.

Die Webseite des Anbieters verspricht einiges mehr, als man tatsächlich erhält. Eine Dokumentation existiert zwar in Ansätzen, sie macht aber einen sehr lückenhaften Eindruck und mündet in Notizen noch geplanter Erweiterungen, von denen eine Verbesserung der Beschreibung eine ist. Vielleicht wäre eine Öffnung des Quellcodes für dieses Projekt eine Chance, die Unterstützung weiterer Entwickler zu finden und so zu der für den professionellen Einsatz erforderlichen Stabilität zu kommen.

SIBProvider

Der bis dato wahrscheinlich bekannteste Aspirant heißt SIBProvider, denn der Autor offeriert seine Software außer auf der eigenen Homepage auch in verschiedenen einschlägigen Foren, zum Beispiel in Borlands Community Site. Leider lässt sich der Bekanntheitsgrad nicht mit Qualität gleichsetzen. So war es nicht möglich, jede mit ADOAnywhere alle Tabelle von *Employee.gdb* einzeln auszulesen. Bei einigen Tabellen mündet dieser doch bescheidene Versuch bereits in *E_FAIL*-Resultaten oder Schutzverletzungen.

SIBProvider bietet merkwürdiger Weise eine Vorbelegung für den Pfad zu einer Datenbank namens *mydb.gdb* an, vermutlich das Testprojekt des Autors, das er zu seiner Bequemlichkeit und zur Verwirrung der Anwenders als Vorgabe fest in seinen Treiber verdrahtet hat.

Beim Versuch, auf eine Tabelle mit Blob-Feldern zuzugreifen (und bei anderen Gelegenheiten), erhielt ich die Fehlermeldung *Falta implementar suporte a arrays (P:\MWB\OLE DB\sibprovider2\uSIBSQLRowSet.pas, line 2476)*. Dies gebe ich hier nur deshalb in voller Länge wieder, da sich daraus der unerfreuliche Schluss ziehen lässt, dass interne Meldetexte in Portugiesisch statt in dem allgemein zu erwartenden Englisch verfasst sind und außerdem Assertions und vermutlich auch Symboldaten mit in den ausgelieferten Code geraten sind. Das ist, für sich betrachtet, kein Hinweis auf eine hohe Qualität des Produkts. Jedesmal, wenn eine solche Meldung auftritt, resultieren weitere Operationen regelmäßig in Schutzverletzungen, so dass man die jeweilige Anwendung beenden und neu starten muss, um weiterzuarbeiten.

Das Ausführen einer Gespeicherten Prozedur per *EXECUTE PROCEDURE* über das ADO Connection-Objekt war nicht möglich. In erhebliche Verwirrung stürzt man diesen Provider, wenn man beim Öffnen eines Recordset mit serverseitigem Cursor eine Cachegröße über eins angibt. Der Provider liefert der Anwendung beim Navigieren in der Datenmenge immer nur den letzten gecachten Record und unterschlägt den Rest. Und auch die Fehlerbehandlung, wenn zum Beispiel ein Update wegen eines Lock-Konflikts mit einer ande-

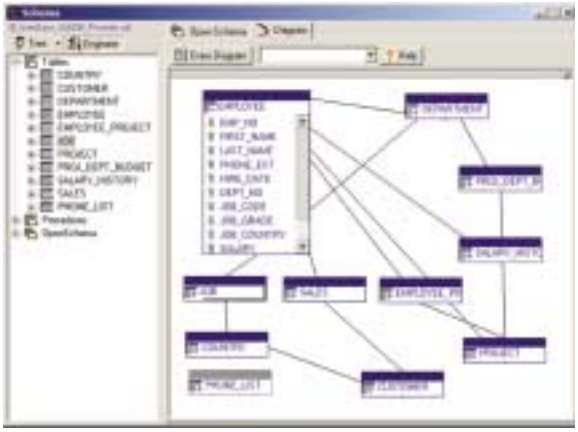

Abb. 2: Schemadiagramm mit ADOAnywhere mit IBProvider

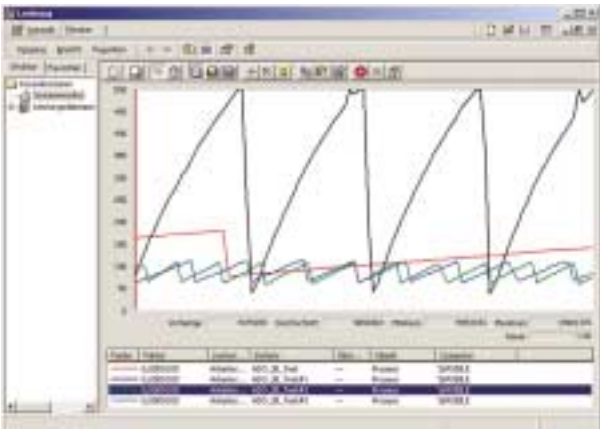

Abb.3: Speicherkonsum eines Client-Cursors (rt = IBProvider, sw = zstyle, gr = IbOleDb, bl = SIBProvider)

ren Transaktion scheitert, endet mitunter in einem Totalabsturz statt einer adäquaten Fehlermeldung.

Verbrauchswerte

Eine gesonderte Betrachtung verdient das Bereich Performance und Ressourcenbedarf. Ich habe mich auf einen simplen Testfall beschränkt, nämlich das Lesen einer Tabelle mit etwa 25.000 Adressdatensätzen à la SELECT * FROM ADRESSEN in ein Recordset mit clientseitigem Cursor. Hier überraschten mich die enormen Unterschiede bei der Speichermenge, die jeweils zum Zwischenspeichern dieses Datenvolumens benötigt wurde. Abbildung 3 zeigt das Profil des Speicherverbrauchs von vier parallel laufenden Instanzen eines Testprogramms beim wiederholten Auslesen der Adresstabelle, wobei natürlich jede Instanz einen anderen Provider verwendete. Das Sägezahnmuster veranschaulicht den Prozess des sukzessiven Lesens und Pufferns der Daten, wodurch stetig mehr dynamischer Speicher beansprucht wird. Wenn die Datenmenge geschlossen und die gepufferten Records freigegeben werden, fällt die Verbrauchskurve steil ab und beginnt mit dem nächsten Lesevorgang der selben Daten wieder zu steigen. Die horizontale Streckung der Sägezähne deutet auf die Dauer des Vorgangs hin.

Die gute Nachricht zuerst: Ressourcenlecks konnte ich mit dieser Methode keinem der vier Provider nachweisen. Der Speicherhunger fällt bei IBProvider allerdings mit in der Spitze knapp 18 MB ziemlich hoch, beim zstyle-Produkt mit bis zu 58 MB geradezu maßlos aus. Dagegen treten die anderen beiden Produkte – vielleicht aufgrund intern einfacherer Funktionsprinzipien – mit circa 10 MB deutlich bescheidener auf. Der Zeitbedarf für diese Operationen differiert ebenfalls erheblich und zwar (isoliert betrachtet) von etwa drei Sekunden bei IbOleDb und SIBProvider über sieben Sekunden bei IBProvider bis hin zu zwölf Sekunden bei zstyle. Diese Werte sind natürlich sehr von der Hardwareausstattung abhängig, müssen also relativ zueinander bewertet werden. Der in Abbildung 3 zu erkennende Zeitverlauf (die Länge der Sägezähne) zeigt noch größere Unterschiede an, die aber nur im konkurrierenden Betrieb der vier Programminstanzen so extrem ausfallen. Vermutlich gelingt es den schneller operierenden Provider-Varianten, durch geringere Kooperativität relativ mehr CPU-Zeit an sich zu reißen. Damit lassen sie in dieser Konstellation die Konkurrenz noch schlechter aussehen, was aber nicht unbedingt als realistisches Szenario zu verstehen ist.

Fazit

Eine eindeutige Empfehlung kann ich leider auch nach meinen Versuchen noch nicht aussprechen, nur eine Tendenz zeichnet sich ab. Erst ein längerfristiger Einsatz in einem realen Projekt brächte die nötige Erfahrung und Gewissheit. Als Gesamteindruck ergab sich für mich, dass ein höherer Preis in diesem Fall für mehr Funktionalität und Qualität steht. Der Freeware-Provider IbOleDb hat erhebliche Mängel und auch bei dem preisgünstigen SIBProvider muss man bei der Qualität Abstriche machen. Diese beiden Kandidaten würde ich nur für kleinere Aufgaben (z.B. sporadischer Datenimport/-export) einsetzen, für eine komplexe und dauernd laufende Anwendung aber nicht.

Wer höhere Ansprüche stellt, sollte IBProvider und das Produkt von zstyle in die engere Wahl ziehen und am besten in der Entwicklungsphase beide Provider verwenden, um sich dann endgültig zu entscheiden. Bei zstyle sollte man den Speicherbedarf im Auge behalten. Das von mir exemplarisch nach ADO portierte Delphi-IBX-Projekt *MASTAPP* ließ sich mit diesen beiden Providern ohne größere Probleme zum Laufen bringen. Die No- bzw. Low-Cost Varianten IbOleDb und SIBProvider sorgten leider eher für ein Feuerwerk von Fehlermeldungen als für eine brauchbare Datenbankanbindung dieser Test-Applikation.

Für klassische, mit Delphi oder C++ Builder programmierte Client/Server-Lösungen, die nicht den sowieso eher hinderlichen und oft übertriebenen Anspruch der Datenbankneutralität erfüllen müssen, werde ich weiterhin auf die nativen Lösungen IBObjects [4] oder IBX [5] setzen. ■

Links & Literatur

[1] www.microsoft.com/data
[2] www.adoanywhere.com/
[3] Andreas Kosch, „ADO und Delphi“, Software und Support Verlag
[4] www.ibobjects.de/
[5] codecentral.borland.com/codecentral/ccweb.exe/author?authorid=102