

# Wie ein .NET-Datenprovider Entwickler unterstützt

## Ritmo for iSeries: .NET-Zugriff auf die iSeries/AS400 von IBM

von Helmut Knappe

CRM, Data Warehouse, e-Commerce – moderne Anwendungen erobern den Mittelstand. Dort stehen oft Midrange-Systeme wie die IBM iSeries (früher AS400) im Zentrum der IT-Landschaft. Für altentümliche „grüne Bildschirme“ sind sie berüchtigt – geschätzt werden sie wegen ihrer Stabilität und Zuverlässigkeit, die jeden PC oder Unix-Server übertreffen und fast an die der Mainframes heranreicht. Nicht wenige Entwickler stehen deshalb vor der Aufgabe, aus modernen Anwendungen heraus auf iSeries-Daten zuzugreifen. Aber keine Angst, spezielle Datenprovider stellen diese Daten dem .NET-Entwickler zur Verfügung. Wie sie darüber hinaus die Entwicklerarbeit erleichtern, zeigt dieser Artikel an Ritmo for iSeries vom kalifornischen Hersteller HiT Software.

Die Grundaufgabe eines Datenproviders lässt sich in einem Satz zusammenfassen: Er stellt in einer Anwendung externe Daten für den lesenden und schreibenden Zugriff über eine Standardschnittstelle zur Verfügung. Dadurch spart er dem Entwickler von Haus aus viel Aufwand für das Programmieren und vor allem das Testen spezieller Schnittstellen zu der jeweils eingesetzten Datenbank – und damit natürlich viel Zeit und Ärger. Doch wer denkt heute noch daran? Wie vor Jahren die ersten ODBC-Treiber den Markt erobert hatten, hat sich heute jeder daran gewöhnt, dass Datenbankzugriffe einfach funktionieren.

### kurz & bündig

#### Inhalt

Zugriff auf DB2-Datenbanken unter iSeries/AS400 – Integration mit Visual Studio und Reporting Services

#### Zusammenfassung

Auch ein Datenprovider kann die Arbeit des Entwicklers erleichtern, wenn er die Phasen des Entwicklungsprozesses mit praktischen Hilfsmitteln unterstützt – ein Beispiel liefert Ritmo for iSeries

### Datenzugriff zuverlässig und schnell

Der Datenprovider von *Ritmo for iSeries* nimmt die Anforderung der Anwendung entgegen und übergibt sie via TCP/IP und die originalen IBM-Protokolle an ein von IBM entwickeltes Systemprogramm auf der *iSeries*, den *Optimized Database Server* (ODBS). Dieses Vorgehen erlaubt performante Zugriffe auf iSeries-Daten, ohne ein systemfremdes Programm auf der iSeries zu installieren. Das kommt den weit verbreiteten Bedenken der Systemverantwortlichen entgegen, „Fremdsoftware“ auf ihre Maschine zu lassen. Bei großen iSeries-Systemen, die Millionen von Euro kosten und in international tätigen Unternehmen oft europaweit zentral eingesetzt werden, mit entsprechend hohen Sicherheitsanforderungen, ist das sehr verständlich – trifft aber auch für kleinere Systeme zu.

Der ODBS steht seit der Betriebssystemversion *V3R1* auf den damals wie heute unter dem Namen *AS/400* bekannten Systemen zur Verfügung. Das bedeutet, dass *Ritmo for iSeries* den Zugriff auch auf über zehn Jahre alte Systeme gestattet.

Zugleich sorgt die Nutzung der IBM-Protokolle für Zukunftssicherheit bei künftigen Updates des Betriebssystems.

### 100 Prozent Managed .NET-Datenprovider

Als echter 100 Prozent Managed .NET-Datenprovider bringt *Ritmo for iSeries* schon die Performance-Möglichkeiten des .NET-Systems zur Geltung. Er ist beim Datenzugriff aus .NET-Anwendungen im Vergleich mit anderen Technologien, beispielsweise dem *HiT-OLEDB/400*-Treiber, um durchschnittlich 10 bis zu 250 Prozent (bei großen Datenmengen) schneller – weitere Möglichkeiten zur Optimierung der Datenzugriffe finden sich im Kasten „Verbindungen auf Performance optimieren“. .NET-Anwendungen können natürlich auch über OLEDB- oder ODBC-Treiber auf Datenbanken zugreifen. Dieser Zugriff benötigt aber die von Microsoft gelieferten ADO.NET-to-OLEDB- und -ODBC-Brückentreiber. Dieser Umweg kostet Performance und verhindert, dass darunter liegende Treiber die Vorteile der CLR nutzen können. Ein 100 Prozent Managed .NET-Daten-

provider macht Umwege überflüssig, ermöglicht dadurch höhere Performance, volle CLR-Funktionalität und direkt vom Betriebssystem verwaltete Skalierbarkeit für Anwendungen.

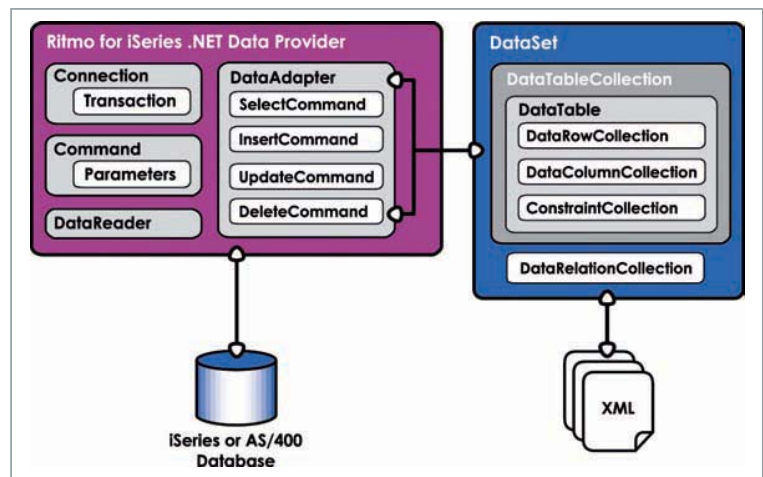
### Zugriff auf iSeries-Daten und -Befehle

In .NET-Anwendungen stellt *Ritmo for iSeries* den Datenzugriff nicht nur über einfache oder parametrisierte SQL-Anweisungen zur Verfügung. Auch Stored Procedures werden unterstützt, wie das Beispiel in Listing 1 zeigt. Data Adapter (Abbildung 1) erlauben die Anbindung von Komponenten an Datenobjekte. *Ritmo for iSeries* gestattet auch das Steuern von Transaktionen auf der iSeries/AS400 mit *Commit* und *Rollback*. Die Ausführung von Befehlen auf der iSeries erlaubt die Integration mit dort bereits vorhandenen Programmen. Für flexiblen Einsatz des Datenprovider und optimale Performance steht außerdem Unterstützung für multiple *ResultSets*, *Bulk Inserts* und Datenkompression zur Verfügung. *Connection Pooling* und Multithreading-Unterstützung sichern die Skalierbarkeit der entstehenden Anwendungen.

### Mit eigener Werkzeugkiste

Ein Datenprovider ist von Haus aus wenig sichtbar und soll es auch sein. Nur wenn einmal nicht alles glatt läuft, kann die Unterstützung für den Entwickler nicht groß genug sein. *Ritmo for iSeries* bringt dafür eine eigene Toolbox mit. Diese Werkzeugkiste hilft nicht nur bei der Problemlösung, sondern auch bei Routinearbeiten, wie dem Erstellen von Datenquellen und

Abb. 1: Ritmo for iSeries im Überblick



Connection Strings. Die fünf Seiten des *Data-Source-Configuration*-Dialogs (Abbildung 2) erfassen alle möglichen Einstellungen – von der Auswahl der Library auf der iSeries (entspricht dem Verzeichnis bei PCs) bis zum Einstellen der *Fetch Block Size* oder der Sortierung. Jetzt noch ein Klick auf GET CONNECTION STRING FOR DATA SOURCE und der Connection String ist fertig zum Kopieren in die eigene Anwendung.

### Funktioniert die Verbindung zur iSeries?

Ein Rechtsklick und die Auswahl von *Test* aus dem Kontextmenü liefert die Antwort. Der dreistufige Test (Abbildung 3) lädt die Ritmo-Assembly, öffnet eine TCP/IP-Verbindung zur iSeries und führt einen grundlegenden SQL-Test durch. Läuft der Test durch alle drei Stufen ohne Probleme, so kann man sicher sein, dass die Verbindung grundsätzlich funktio-

niert, und mit dem Test der eigenen Module beginnen.

Scheitert der Test, bietet die Erstellung von Trace-Dateien, die ebenfalls in der Toolbox eingeschaltet wird (Abbildung

### Listing 1

#### Stored-Procedure-Beispiel

```

/// <summary>
/// Anweisung
/// Stored Procedure ausführen und ResultSet anzeigen
/// </summary>

public void sample_storedProcedure()
{
//Verbindung zur DB öffnen und
//Sql400Connection-Objekt zurückgeben
Sql400Connection myConnection =
    new Sql400Connection(myConnectionString);
myConnection.Open();
//Sql400Command-Objekt erzeugen
Sql400Command myCommand = new Sql400Command();
myCommand.CommandText = "MyStoredProcedure";
myCommand.Connection = myConnection;
//Anweisung zur Ausführung einer Stored Procedure
myCommand.CommandType = CommandType.
    StoredProcedure;

//Data-Reader-Objekt erzeugen und Stored Procedure
//ausführen
Sql400DataReader myReader = myCommand.
    ExecuteReader();

//Data Reader zeilenweise durchgehen
while (myReader.Read())
{
//myReader.GetBoolean(0);
//myReader.GetInteger(1);
//...
}

//Data Reader und Verbindung schließen
myReader.Close();
myConnection.Close();
}

```

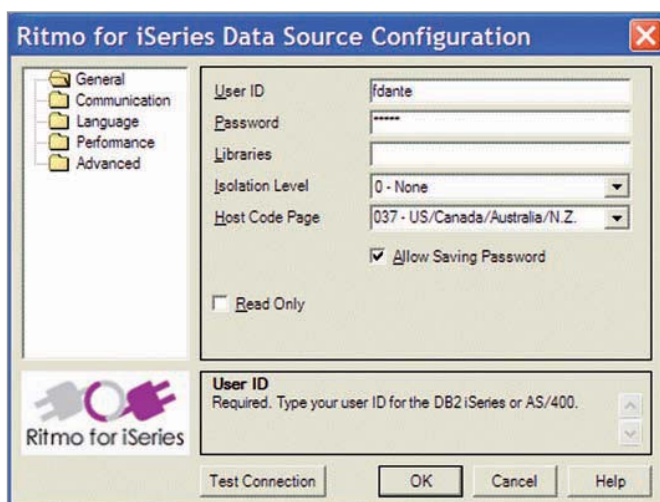


Abb. 2: Datenquellen sauber konfiguriert

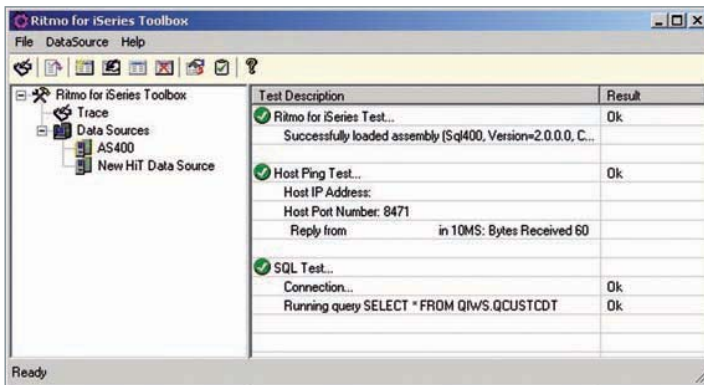


Abb. 3: Verbindungstest – Ritmo for iSeries meldet „Alles ok“

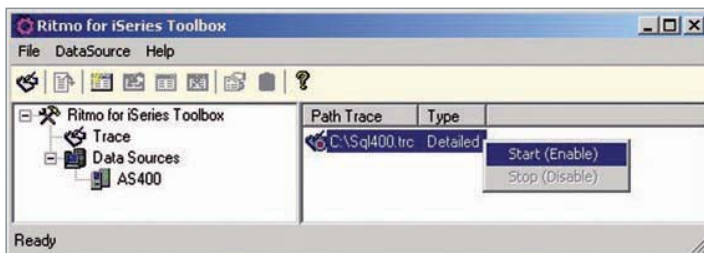


Abb. 4: Zur Lösung Traces einschalten

4), Hinweise auf mögliche Ursachen – natürlich auch in anderen Fällen. Die Trace-Funktionalität speichert akribisch die detaillierten Meldungen der beteiligten Instanzen (Provider, iSeries) in einer Datei ab. Dauerhaft lässt sich das Tracing über die Konfigurationsdatei *RitmoSeries.xml* einschalten durch Einfügen von

```
<trace>
<tracefile>C:\Program Files\Hit Software\
  Ritmo for iSeries\logs\RitmoSeries.trc</tracefile>
<traceflag>True</traceflag>
</trace>
```

Das Tracing bremst natürlich die Ausführung. Deshalb sollte man nicht ver-

gessen, es spätestens beim Abschluss der Entwicklung wieder auszuschalten – am besten schon vor den Performance-Tests beim Übergang auf das Produktivsystem. So ein Trace hilft auch bei der Optimierung des erstellten Codes, beispielsweise mithilfe der sog. *Packages* – das sind vorkompilierte SQL-Anweisungen, also die Stored Procedures der iSeries.

### Integration in Visual Studio

Microsoft propagiert Visual Studio 2005 als die Entwicklungsumgebung für alle Arten von Projekten – von der üblichen Entwicklung einer Anwendung oder DLL bis zu Business-Intelligence-Projekten. Dabei ist die Vorgehensweise weitgehend gleich:

- Projekt öffnen oder neu erzeugen,
- Datenquellen auswählen oder definieren und
- Komponenten auf Formularen oder in Designern positionieren.

Komponenten von Drittanbietern integrieren sich bei der Installation gleich in die Toolbox von Visual Studio und stehen dort zur Auswahl bereit. Das gilt auch für *Ritmo for iSeries* (Abbildung 5). Die Vorteile liegen auf der Hand: Statt vieler verschiedener Entwicklungsumgebungen muss nur eine, wenn auch eine umfangreiche erlernt werden. Die Arbeit von Entwicklern wird dadurch effektiver. Außerdem stehen wichtige Module von Visual Studio für Projekttypen zur Verfügung, bei denen das bisher nicht selbstverständlich war. Die *Ritmo Developer Tools for Visual Studio .NET* verbessern die Produktivität in der Visual-Studio-Umgebung mit grafischen Tools für Datenbankverbindungen, SQL-Anweisungen und Datenadapter. Die folgenden drei *Ritmo-iSeries*-Komponenten *Sql400Connection*, *Sql400Command* und *Sql400DataAdapter* finden sich im Datenabschnitt der Visual Studio Toolbox:

- *Sql400Command* – SQL-Anweisungen mit dem Command-Design-Dialog oder über die Objekteigenschaften erzeugen.
- *Sql400Connection* – Verbindung zur iSeries/AS400 über den Data-Source-Configuration-Dialog oder die Einstellung der Verbindungseigenschaften in Visual Studio herstellen.
- *Sql400DataAdapter* – einen DataAdapter für eine iSeries/AS400-Verbindung mit dem Data-Adapter-Configuration-Assistenten oder über die Einstellung

### Fit für das Reporting

Die *Ritmo Data Extensions* stecken unter dem Namen *DataExtension.dll* im Unterverzeichnis *lib\Sql400ReportingServices* und müssen in die Verzeichnisse für Report Server Extensions kopiert werden:

```
C:\Programme\Microsoft SQL Server\MSSQL.3\Reporting Services\ReportServer\bin
```

Damit sie auch im Report Designer erkannt werden, kommen sie zusätzlich noch an dessen Standardort für Extensions:

```
C:\Programme\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies
```

Folgende Einträge in *RSReportServer.config* und *RSReportDesigner.config* unter dem Data-Element vollenden die Installation:

```
<Extension Name="Ritmo for iSeries" Type="HitSoftware.Sql400.ReportingServices.DataExtensions.
  Sql400ConnectionExtension,HitSoftware.Sql400.ReportingServices.DataExtensions"/>
<Extension Name="Ritmo for iSeries" Type="Microsoft.ReportingServices.QueryDesigners.VDTQueryDesigner,Microsoft.
  ReportingServices.QueryDesigners"/>
```

Weitere Einstellungen sind nötig für die Code Access Security (siehe Visual-Studio-Dokumentation) in *rssrvpolicy.config* und *rspreviewpolicy.config*.

### Was tun, wenn die DB2/400 Verbindung nicht funktioniert

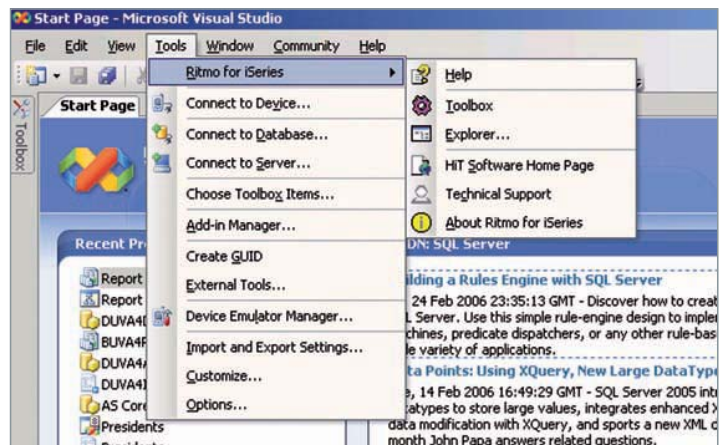
1. User ID und Password prüfen – reichen die Zugriffsrechte für die gewünschte Operation aus?
2. Sind Host-IP-Adresse und Portnummer korrekt?
3. Laufen auf der iSeries/AS400 die nötigen Subsysteme QZDASOINIT und QZDASRVSD?

Dies lässt sich mit *WRKSBSJOB QSERVER* prüfen, gegebenenfalls *QServer* mit *STRSBS QSERVER* starten oder die Subsysteme mit *STRHOSTSVR \*ALL* – danach erneut prüfen.

der Eigenschaften in Visual Studio erzeugen.

Eine SQL-Abfrage für die iSeries in Visual Studio zu erzeugen, geht deshalb genau wie bei anderen Datenbanken ganz einfach: Komponente auf das Formular ziehen, Doppelklick und die Verbindungsparameter einer *sql400Connection*-Komponente lassen sich direkt über ihre Eigenschaften in Visual Studio definieren. Dann zieht man ein *sql400Command*-Objekt auf das Formular und stellt dessen Eigenschaften ein. Nach Angabe der Verbindung (*sql400Connection*-Objekt) folgt dann der Entwurf der Anweisung im Command-Design-Dialog. Der Database Tree zeigt die Tabellen, Views und Stored Procedures an, die für die DB2/400-Verbindung zur Verfügung stehen. In der erweiterten Liste finden sich die Spalten- und Stored-Procedure-Informationen. Dazu werden jeweils die Eigenschaften des gewählten Objekts angezeigt. Auch den Aufbau der SQL-Anweisung unterstützt Ritmo durch das übliche Ziehen der Objekte in den Textbereich und durch die zusätzliche Definition von Parametern

Abb. 5: Ritmo for iSeries ist vollständig in Visual Studio integriert



für parametrisierte Abfragen und Stored Procedures.

### Eine Developer Edition für das Extras-Menü

Die Entwicklerversion von *Ritmo for iSeries* macht deutlich, dass sie von Entwicklern geplant und gebaut wurde. Sie enthält neben der allgemeinen Ritmo-Toolbox eine spezielle C#-Toolbox und einen Database Explorer – integriert in das EXTRAS-Menü von Visual Studio. Dieser verschafft bessere Orientierung in der iSeries-Um-

gebung und erlaubt interaktiv die iSeries-Daten zu betrachten – zum komfortablen Überblick für Entwickler. Zusätzlich erlaubt die Developer Edition in Verbindung mit den *SQL Server Reporting Services* das Erzeugen von Berichten auf DB2-Daten über den .NET-Provider von Ritmo.

### Die lieben Berichte

Hand aufs Herz: Entwickler schreiben gern Code, entwerfen Objekte und Strukturen, aber Berichte? Das klingt für viele wie Strafarbeiten. Da fehlt der Freiraum,

## Impressum



**Verlag:**  
Software & Support Verlag GmbH  
Geleitsstraße 14  
D-60599 Frankfurt am Main  
Tel. +49 (0) 69 6300890  
www.software-support.biz

**Anschrift der Redaktion:**  
dot.net magazin  
Software & Support Verlag GmbH  
Hauptstr. 8b  
D-82008 Unterhaching  
Tel.: +49 (0) 89 665576-21  
Fax: +49 (0) 89 665576-11  
E-Mail: red@dotnet-magazin.de  
www.dotnet-magazin.de

**Chefredaktion:** Peter Monadjemi  
**Redaktion:** Felix Schrader, Markus Zeischke  
**Redaktionsassistentz:** Sebastian Brück, Alexander Schmidt  
**Leiter Grafik/Produktion:** Jens Mainz  
**Layout, Titel:** Dominique Bergmann, Michel Michiels-Corsten, Jessica Demirkaya, Melanie Hahn, Daniel Hartung, Jens Mainz, Sissy Mertens, Maria Rudi

**Autoren dieser Ausgabe:**  
Thomas Becker, Jörg Behrendt, Marcellus Buchheit, Marcel Franke, Jörg Freiberger, Dirk Frischalowski, Joachim H. Fröhlich, Christian Gross, Ralf Hüttel, Immo Landwerth, Yasmin Limberger, Helmut Knappe, Andreas Kosch, Markus Palme, Peter Pohmann, Mathias Raacke, Jochen Reinelt, Raphael Romeikat, Ralph Schuster, Martin Szugat, Jörg Wegener, Reinhard Wolfinger

**Fachübersetzungen:** Frank Langenau  
**Pressevertrieb:**  
DPV Network  
Tel. +49 (0) 37845626, www.dpv-network.de

**Anzeigenverkauf:**  
MARKETING PROJEKT 2000 GmbH  
Hochstr. 3  
86453 Dasing  
Tel. +49 (0) 8205 96233  
Fax. +49 (0) 8205 962345  
E-Mail: mediainfo@dotnet-magazin.de

Es gilt die Anzeigenpreisliste Nummer 4

**Druck:** PVA Landau  
**ISSN:** 1619-7933

**Abo-Service:**  
Software & Support Verlag GmbH  
Tel. +49 (0) 69 6300890  
www.dotnet-magazin.de/service/

**Abonnementpreise der Zeitschrift:**

Inland:	10 Ausgaben	€ 76,50
Europ. Ausland:	10 Ausgaben	€ 86,50
Studentenpreis (Inland):	10 Ausgaben	€ 63,00
Studentenpreis (Ausland):	10 Ausgaben	€ 73,00

**Einzelverkaufspreis:**

Bundesrepublik Deutschland:	€ 8,50
Schweiz:	sFr 16,50
Österreich:	€ 9,60

**Erscheinungsweise:** 10 Ausgaben pro Jahr  
© 2006 für alle Beiträge

**Ständige Beilage:** Best of MSDN Magazine  
Alle Rechte, auch für Übersetzungen, sind vorbehalten. Reproduktionen jeglicher Art (Fotokopie, Nachdruck, Mikrofilm oder Erfassung auf elektronischen Datenträgern) nur mit schriftlicher Genehmigung des Verlages. Jegliche Software auf der Begleit-CD zum dot.net magazin unterliegt den Bestimmungen des jeweiligen Herstellers. Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden. Honorierte Artikel gehen in das Verfügungsrecht des Verlags über. Mit der Übergabe der Manuskripte und Abbildungen an den Verlag erteilt der Verfasser dem Herausgeber das Exklusivitätsrecht zur Veröffentlichung. Für unverlangt eingesandte Manuskripte, Fotos und Abbildungen keine Gewähr.

Microsoft .NET ist eingetragenes Warenzeichen der Microsoft Corp. Das dot.net magazin sowie die Software & Support Verlag GmbH sind unabhängig von Microsoft Corp. Alle weiteren Markennamen sind eingetragene Warenzeichen der entsprechenden Unternehmen.

Die Website des dot.net magazin wird gehostet von Host Europe (www.hosteurope.de).

dot.net magazin contains articles under license from Microsoft Corporation and CMP Media LLC, copyright © 2006  
Microsoft Corporation and CMP Media LLC. All rights reserved. The following are trademarks of Microsoft Corporation: MSDN, the MSDN logo, Microsoft. Das dot.net magazin enthält Artikel unter Lizenz der Microsoft Corporation und von CMP Media LLC, Copyright © 2006 Microsoft Corporation und CMP Media LLC. Alle Rechte vorbehalten. Die folgenden eingetragenen Warenzei-

es geht darum, dass Werte an bestimmten Stellen auftauchen, Summen, Gruppen- und Seitenwechsel stimmen – das bedeutet viele Stunden und jede Menge Detailarbeit. Software, die diese mühevollen Tätigkeit erleichtert, erntet den Beifall der Entwickler. Auch ein Treiber kann dazu beitragen, beispielsweise indem er, wie *Ritmo for iSeries*, die Microsoft SQL Server Reporting Services unterstützt. Dazu dienen die Extensions der *Ritmo for iSeries Developer Edition*. Mehr Informationen zur *Reporting Services Data Processing Extension* finden sich unter [3] in der Dokumentation der Microsoft Reporting Services. Um die Ritmo Extension einzusetzen, sind allerdings noch einige manu-

elle Vorbereitungen erforderlich (siehe Kasten „Fit für das Reporting“).

### Unterm Strich

Auch ein einfacher Datenprovider kann einen Unterschied in der Unterstützung machen, die er für Software-Entwicklung bietet. Dabei kommt es weniger auf die Anzahl der Merkmale an, als auf ein auf den praktischen Entwicklungsprozess abgestimmtes Konzept. Was auf diesem Gebiet möglich ist, zeigt sich am Beispiel von *Ritmo for iSeries*. Ein weiterer wichtiger Gesichtspunkt gehört unbedingt noch dazu: Der schnelle und kompetente Service, den ein Hersteller von Zusatz-Software seinen Entwicklerkunden bieten kann,

entscheidet oft über das Gelingen des Projekts – doch da muss wohl jeder seine Erfahrungen selbst machen.

**Dr.-Ing. Helmut G. Knappe** engagiert sich seit mehr als 20 Jahren in der IT-Entwicklung und Projektleitung bei Datenbankanwendungen. Zu seinen Schwerpunkten zählt die Integration heterogener Datenbanken. Seit 2002 ist er als Country Manager Deutschland und Österreich für den amerikanischen Software-Hersteller Hit Software Inc. tätig. Er hält Workshops und Vorträge auf nationalen und internationalen Konferenzen und schreibt Artikel für Fachzeitschriften. E-Mail: [helmut.knappe@hitsw.com](mailto:helmut.knappe@hitsw.com).

### ● Links & Literatur

- [1] [www.developer.com/net/cplus/print.php/2197621](http://www.developer.com/net/cplus/print.php/2197621)  
 [2] [www.ondotnet.com/lpt/a/3226](http://www.ondotnet.com/lpt/a/3226)

## Verbindungen auf Performance optimieren

Durch passende Einstellungen auf der Performance-Seite des Data-Source-Configuration-Dialogs lassen sich Geschwindigkeit und Effizienz des Datenzugriffs erheblich steigern. Dazu hier einige hilfreiche Hinweise:

### APPC-Verbindung mit großen Datenmengen

Das Feld *APPC Buffer Size* stellt die Größe des beim Informationsaustausch mit dem Server verwendeten Pufferspeichers ein. Je größer dieser Puffer, desto geringer die Anzahl der Netzwerkanforderungen. Bei großen Datenmengen muss der Treiber viele Sendevorgänge durchführen und synchron auf den Empfang warten. Ein höherer Wert für die *APPC Buffer Size* ist daher vorteilhaft beim Senden großer Mengen von Daten der Typen *binary/varchar/long* und *varbinary/long varchar* in parametrischen Abfragen.

### Begrenzung der abgeholten ResultSets

Liefert eine Abfrage viele *ResultSets*, begrenzt ein Wert größer als 0 für *Max Rows in ResultSet* in der Datenverbindung die Anzahl der gleichzeitig abgeholten Zeilen auf diesen Wert. Bei Erreichen dieses Werts liefern nachfolgende *Fetch*-Aufrufe *NO DATA FOUND* zurück.

### Parameterabfragen

Anwendungen wie Access verwenden oft Parameterabfragen. Der Wert *Use Packages* sorgt dafür, dass Verbindungen vorübersetzte Parameteranweisungen in *Packages* speichern.

### Langsame Verbindung zur AS400

Die *Fetch Block Size* reduziert die Anzahl der Datensätze, die ein *Fetch* holt, und kann damit die Übertragung bei geringer Übertragungsrates verbessern. Dieser Wert (in Kilobytes) sollte aber immer größer als der größte einzelne Datensatz im *ResultSet* sein. Diese Einstellung gilt auch für komplexe SQL-Abfragen und Abfragen auf großen/verbundenen Tabellen.

### Datenkompression

Die optionale Datenkompression erlaubt dem AS/400 die Daten für die Übertragung zum Client zu komprimieren. Je nach Datentyp bringt das mehr oder weniger. Die Kompressionsalgorithmen können einige Daten leicht komprimieren, andere dagegen überhaupt nicht. Datenkompression erhöht die Belastung des Prozessors. Falls die Datenquellen nicht erheblich mehr Nutzen aus der Kompression ziehen, könnte sich die Performance sogar ein wenig verschlechtern. Ein gezielter Test empfiehlt sich, um den Nutzen oder die Konsequenzen in einer bestimmten Umgebung festzustellen.

## Fünf Fragen an Andrea Laspertini, Software Engineer, und Giacomo Lorenzin, CEO von Hit Software

### Wann und warum fiel die Entscheidung für das .NET Framework? Standen Alternativen zur Diskussion?

*Hit Software* hat sich von Anfang auf Middleware-Treiber für IBM DB2 spezialisiert. Wir begannen mit ODBC und als Microsoft mit OLEDB kam, entwickelten wir die *HIT-OLEDB*-Treiber für DB2/400 und für DB2. Als Microsoft das .NET Framework vorstellte war es unausweichlich, dass wir auch für diese Plattform Treiber entwickeln würden.

### Wie hat das .NET Framework die Entwicklung positiv beeinflusst?

Wir waren von der Qualität des .NET Framework sehr beeindruckt. Da wir umfangreiche Erfahrung mit ODBC, OLEDB und JDBC hatten, lernten wir den neuen Komfort zu schätzen. Microsoft hat die alten Standards ODBC und JDBC bei der Entwicklung der .NET-Provider berücksichtigt, sodass das Resultat ein sauberer und abgerundeter Konnektivitätsstandard ist.

### Was waren die größten Hindernisse, die überwunden werden mussten?

Es gab eine Limitierung bei ADO.NET, die wir in unseren Treibern ausbügeln mussten. Wir mussten z.B. unseren eigenen Pooling-Mechanismus einführen, eine Unterstützung für Schema-Kataloge, die es erst bei .NET 2.0 gibt, wir führten parametrisierte Abfragen auch bei SQL-Abfragen ein und ermöglichten es, dass beim Verbindungsobjekt zusätzliche Pa-

rameter eingestellt werden können. Der *Sql400DataReader* ermöglicht es, dass mehr als ein *ResultSet* offen sein darf, was ADO.NET nicht gestattet.

### Wurden bei der Umsetzung Schwachstellen im .NET Framework deutlich oder gibt es Dinge, die Microsoft hätte anders oder besser lösen können?

Ich glaube nicht, dass es echte Schwachstellen gibt, aber ein paar Dinge, die Microsoft hätte besser machen können. So ist die Integration in Visual Studio recht komplex, es sollte für 3rd-Party-Entwickler einfacher sein, ihre Produkte in die IDE zu integrieren. Auch bei VS 2005 ist die Integration mit dem Server Explorer recht schwierig. Auch die Installation in den GAC könnte einfacher sein. Und zwischen den einzelnen Framework-Versionen gibt es Inkompatibilitäten bei den ADO.NET-Treibern. Auch ein 2-Phase-Commit stand bei .NET 1.1 nicht zur Verfügung. Aber ich glaube, daran wird gearbeitet.

### Wie sieht die Zukunft des Projekts aus? Gibt es bereits Überlegungen das .NET Framework 2.0 zu nutzen?

*Ritmo for iSeries 3* unterstützt bereits das .NET Framework 2.0 – beim nächsten größeren Release wird das .NET 2.0 API vollständig unterstützt.