

Gupta-Datenbankumstieg mit sqlPorter und sqlTranslator

# Im Falle eines Falles

Ein manueller Datenbankumstieg ist zeitaufwendig und fehleranfällig. Die Tools sqlPorter und sqlTranslator sollen einen Plattformwechsel (fast) ohne Anwendungsänderung erlauben. **Andreas Maslo**

## Auf einen Blick

### Inhalt

Der Umstieg von einem Datenbankmanagementsystem auf ein anderes ist in der Praxis ein steiniger Weg. Das Beratungs- und Softwarehaus fecher unterstützt Endkunden und Softwarehäuser bei der Migration von Anwendungen auf Microsoft SQL Server und die Oracle-Datenbank. Das Tool sqlPorter sorgt für die Übertragung aller Datenbankobjekte und sqlTranslator für die dynamische Umsetzung proprietärer SQL-Dialekte zur Laufzeit.

### Autor

Dipl.-Ing. Andreas Maslo leitet das Ingenieurbüro IngES, das sich mit der Erstellung von EDV-Publikationen und der Software-Entwicklung befasst. Er ist als freier Journalist, EDV-Berater, Fachbuchautor, Software-Entwickler und Redakteur der database pro tätig.

Der Wechsel von einer Datenbank zu einer anderen ist in der Theorie ganz einfach: Dank der Standardisierung von SQL sollten vorhandene Anwendungen grundsätzlich mit einer beliebigen neuen Datenbankplattform zusammenarbeiten. Die Praxis allerdings sieht anders aus. Da stehen einem problemlosen Umstieg inkompatible Herstellerstandards zur Definition von Tabellen und Views, proprietäre Programmiersprachen für Trigger und Stored Procedures und nicht zuletzt erhebliche Abweichungen der einzelnen SQL-Dialekte im Wege. Mit seiner toolgestützten Dienstleistung sqlPorter und der Middleware-Komponente sqlTranslator ist das Software- und Consultinghaus fecher [1] getreten, den Datenbankumstieg so einfach zu machen, wie er von den Erfindern von SQL eigentlich einmal gedacht war. Derzeit stehen sqlPorter und sqlTranslator für die Quelldatenbank Gupta SQL-Base [2] und die Zieldatenbanken Microsoft SQL Server [3] und Oracle [4] zur Verfügung. Die Unterstützung für PostgreSQL und weitere Datenbanken ist nach Aussagen des Herstellers derzeit in Vorbereitung. Auf die Weiterentwicklungen darf man gespannt sein.

## Automatisierung und Co.

Die Aufgaben, die mit der Umstellung auf ein neues Datenbanksystem verbunden sind, lassen sich nur teilweise automatisieren. Entsprechend

bedient sich das Migrationsangebot sqlPorter intern zwar verschiedener Werkzeuge, die vom Hersteller für diesen Zweck eigens entwickelt wurden, stellt sich aber dem Kunden gegenüber als Dienstleistung zum Festpreis dar. Dabei greift fecher auf die Erfahrung aus gut hundert Migrationsprojekten von Gupta-Datenbankanwendungen auf die .NET-Umgebung zurück, die das Unternehmen in den vergangenen zwei Jahren unter dem Namen „The Porting Project“ durchgeführt hat.

Im Gegensatz zur Anwendungsportierung lautet die wichtigste Vorgabe für sqlPorter-Projekte allerdings, dass die vorhandenen Applikationen mit möglichst wenigen Eingriffen in den Code mit der neuen Datenbank weiterarbeiten müssen. Damit die neue Datenbank sich aus Anwendungssicht exakt wie die alte verhält, geht es in der ersten Phase zunächst einmal darum, die Tabellenstruktur möglichst eins zu eins auf die Zieldatenbank abzubilden.

## Fallstricke durch vielfältige Standards

Schon beim Anlegen der erforderlichen Tabellen lauern die ersten Fallstricke. Neben den ANSI-standardisierten Datentypen bieten die Datenbankhersteller in der Regel zahlreiche Erweiterungen an, für die erst einmal eine passende Entsprechung auf der neuen Plattform gefunden sein will. Hier ist zu berücksichtigen, ob für die später zu speichernden Daten der ASCII-Zeichensatz ausreicht oder Unicode-Unterstützung benötigt wird. Unterschiedlich gehandhabt wird auch die Verwendung von Primärschlüsseln als *ROWID*, *PRIMARY KEY* oder *TIME-STAMP* und die Frage, ob die zugehörigen Indizes automatisch erzeugt werden. Schnell kann ein Index fehlen oder doppelt angelegt sein, so dass äußerste Vorsicht geboten ist, sollen die Datenkonsistenz und eine gute Performance nicht gefährdet werden.

Aus den unterschiedlichen Herstellerstandards bei der Einrichtung von zusätzlichen Indizes, Beziehungen zur referenziellen Integrität, Views, Synonymen und Benutzerkonten ergeben sich weitere Klippen. Sind diese alle umschifft, beginnt mit der Phase 2 die Umsetzung der Datenbanklogik. Für die Programmierung von Stored Procedures oder Functions sowie von Triggern enthält der SQL-Standard keiner-

Datenübernahme mit sqlPorter (Bild 1)



## SQLBase – von Gupta zu Unify

Die ehemals von Gupta entwickelte Datenbank SQLBase wird mittlerweile über Unify vertrieben und weiterentwickelt. In den mehr als zwanzig Jahren seit der Gründung wurde Gupta [1] mehrfach veräußert und durch andere Firmen übernommen:

- 1984: Firmengründung durch Umang Gupta
- 1997: Umbenennung in Centura
- 2001: Anfang des Jahres Übernahme durch Platinum Equity
- 2001: Ende des Jahres Umbenennung in Gupta
- 2005: Übernahme durch Warp Technology Holdings, Inc.
- 2005: Umbenennung der Warp Technology Holdings, Inc. in Halo Technology Holdings, Inc.
- 2006: Übernahme durch Unify Corporation

lei Vorgaben. Hier hat jeder Datenbankanbieter seine eigene Sprache implementiert. Von Team-Developer-Dialekten über PL/SQL oder Transact-SQL bis zu Java gibt es zahlreiche Varianten, die sich nicht ohne eingehende Analyse des Quellcodes ineinander überführen lassen. Besonders bei Triggern ist zudem zu beachten, wann diese jeweils ausgelöst werden: Der SQL Server etwa feuert Trigger nur einmal pro Statement, während andere Datenbanken pro Datensatz feuern. Hier ist entsprechende manuelle Codierarbeit gefragt, um das Verhalten der ursprünglichen Datenbank korrekt nachzubilden.

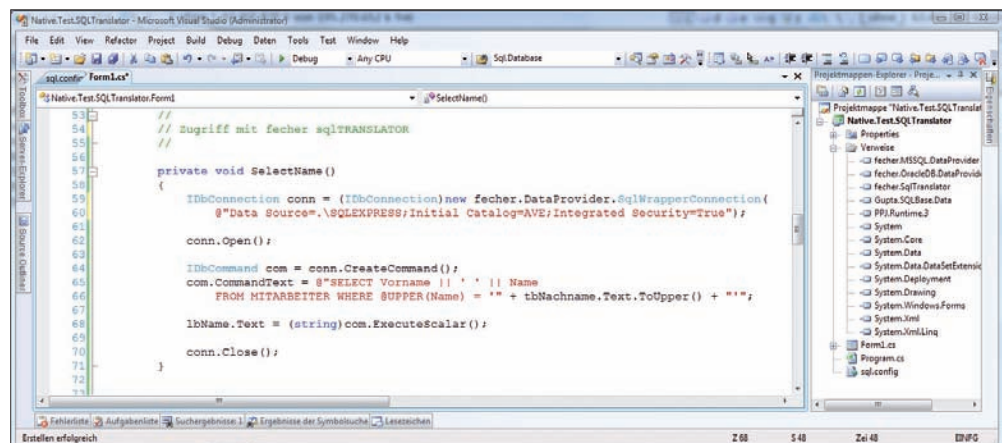
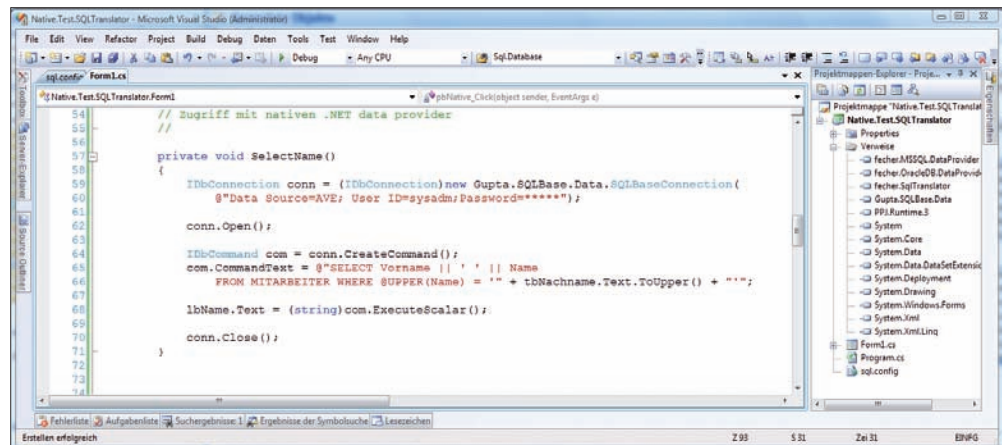
Sind Datenstrukturen und Datenbanklogik auf der neuen Plattform vollständig realisiert, können die vorhandenen Daten von der Quelldatenbank übernommen werden. Zu diesem Zweck hält der sqlPorter (Bild 1) eine Datenpumpe bereit, die sich als SQL-Anwendung mit beiden Datenbanken verbindet und die auf der einen Seite ausgelesenen Daten direkt auf der anderen Seite wieder einfügt. Den abschließenden Schritt bildet eine Datenverifikation, bei der nicht nur die Struktur und die Anzahl, sondern auch die genauen Inhalte der Datensätze einschließlich sämtlicher Indizes in Quell- und Zieldatenbank verglichen werden. Erst wenn diese Überprüfung fehlerfrei durchgelaufen ist, ist die Datenportierung erfolgreich abgeschlossen.

## Middleware als Dolmetscher zwischen den SQL-Dialekten

Damit auch die Anwendung mit der neuen Datenbankumgebung zurechtkommt, ist eine weitere Hürde zu überwinden: Unterschiedlich bezeichnete Funktionen zur String-Manipulation, Mathematik- und Datumsfunktionalität, Abweichungen im Verhalten der Datentypen für Datum und Uhrzeit oder die Syntax von SQL-Befehlen und Join-Definitionen führen dazu, dass die erhaltenen SQL-Statements stets nur auf der Datenbank funktionieren, für die sie zunächst entwickelt wurden. Hier sorgt der sqlTranslator für die erforderliche Umsetzung der Syntax. Die Anwendung sieht diesen als Data Provider und spricht mit ihm die Sprache der „alten“ Datenbank. Der Data Provider selbst verbindet sich mit der neuen Datenbank und übersetzt die SQL-Statements für diese passend.

Um eine Anwendung, die bislang mittels ADO.NET auf Gupta SQLBase zugreift, auf den SQL Server von Microsoft umzustellen, muss der zur neuen Zieldatenbank passende Data Provider *fecher.MSSQL.DataProvider* in die Projektverweise aufgenommen werden. Außerdem wird der sqlTranslator als neuer Datenbankprovider in der Datei *sql.config* eingerichtet, siehe Beispiel in Listing 1.

► Beispiel für den Datenzugriff mit dem nativen .NET Data Provider (Bild 2)



Beispiel für den Datenzugriff mit sqlTranslator (Bild 3)

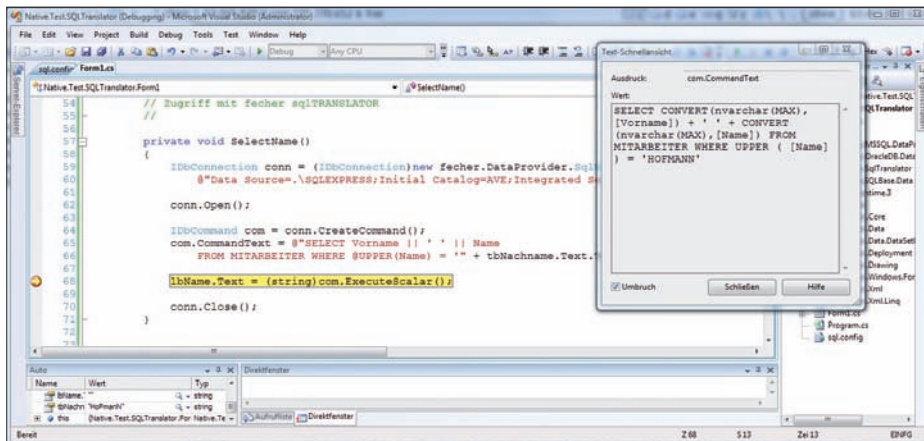
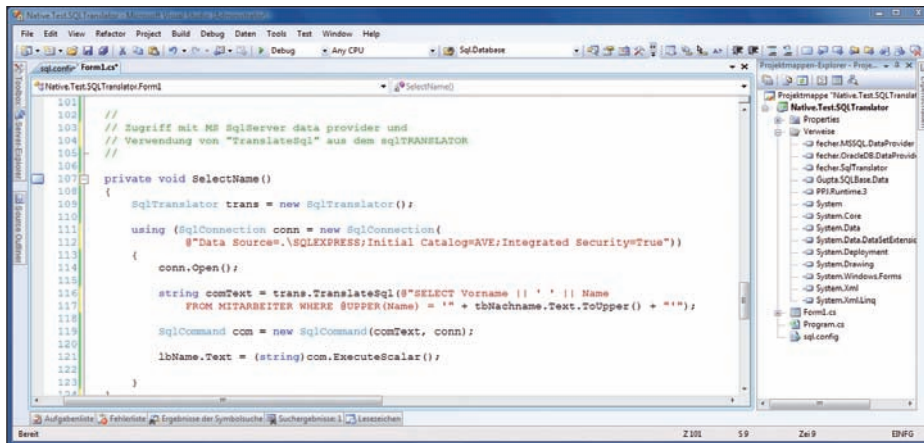
Listing 1: Konfiguration des sqlTranslator-Datenbankproviders

```
<database name="AVE">
  <provider name="SqlWrapper" namespace="fecher.DataProvider"
    assembly="fecher.MSSQL.DataProvider" brand="31"
    driver_type="MicrosoftSqlServer"/>
  <date_format value="YMD" />
  <sqlbase_version value="9.0" />
  <concat_null_yields_null value="on" />
  <read_structure value="false" />
  <rowid_type value="timestamp" />
  <unicode value="false" />
  <on_parse_error value="returnsource" />
  <tracing value="true" />
  <connection_string value="Data Source=.\SQLEXPRESS;Initial
    Catalog=AVE;Integrated Security=True" />
</database>
```

Wird jetzt in der Anwendung der Gupta Data Provider noch durch die *fecher.DataProvider.SqlWrapperConnection* ersetzt, erfolgt ohne großen Aufwand der Zugriff ab sofort auf die Microsoft-Datenbank (Bild 2 und 3).

Das Besondere hierbei ist: Proprietärer SQL-Code, in diesem Beispiel zwei String-Konkationen und eine Wandlung in Großbuchstaben, darf unverändert in der alten Form stehen bleiben. Um dessen Übersetzung kümmert sich bereits der sqlTranslator, was sich durch das

Für SQL Server übersetzter SQL-Code (Bild 4)



Beispiel für die direkte Nutzung der TranslateSql-Funktion (Bild 5)

Setzen eines Haltepunktes im Debugmodus von Visual Studio sehr leicht zeigen lässt (Bild 4).

Eine Alternative bietet sich für Anwendungen an, die bereits mit der neuen Zieldatenbank arbeiten. Hier müssen vielleicht nur an einzelnen Stellen SQL-Zugriffe gemäß der alten Syntax unterstützt werden – etwa weil in der Applikation neu geschriebene und portierte Teile aufeinander treffen. Dazu kann der Entwickler die Funktion *TranslateSql*, die vom sqlTranslator intern zur Umsetzung des SQL-Codes genutzt wird, auch direkt aufrufen. Die Funktion ist vollkommen unabhängig vom eigentlichen Datenbankzugriff, sodass sich auf diesem Wege etwa auch SQL-Statements konvertieren lassen, die ihrerseits in einer Datenbank gespeichert sind (Bild 5).

Beurteilung

Die Kombination aus der toolgestützten automatischen Portierung von Datenbankstrukturen, -inhalten und -logik und der Middleware-Komponente sqlTranslator ermöglicht den Wechsel der Datenbankplattform zwar nicht auf Knopfdruck, aber immerhin mit überschaubarem, von vornherein feststehendem Aufwand und ohne größere Anpassung der Anwendung.

Die Aufgabentrennung zwischen der Datenübernahme und der dynamischen Umsetzung der SQL-Dialekte zur Laufzeit macht Sinn und vereinfacht die Migration. Einziger Schwachpunkt: Derzeit ist die Unterstützung für die Quell- und Zieldatenbanken noch sehr stark eingeschränkt. Auf Dauer würde man sich natürlich wünschen, dass nicht nur weitere Ziel-, sondern auch Quelldatenbanksysteme – und zwar wechselseitig – für eine Migration unterstützt werden, zumal nicht jeder als Ausgangsdatenbank Gupta und als Zieldatenbanken den SQL Server oder Oracle verwendet. Administratoren und Entwickler bevorzugen zwar im Allgemeinen Tools, mit denen sie Migrationen in eigener Regie durchführen können. Auf der anderen Seite sind durch das Dienstleistungsangebot von fecher aber auch Sonderfälle, die im Rahmen der Migration anfallen können, bereits abgedeckt und müssen nicht eigenständig gelöst werden. [am]

[1] fecher e. Kfm; [www.fecher.eu](http://www.fecher.eu)  
 [2] (Gupta) SQL-Base; [www.unify.com](http://www.unify.com)  
 oder [www.guptaworldwide.com](http://www.guptaworldwide.com)  
 [3] Microsoft SQL Server;  
[www.microsoft.com/germany/sql/2008/default.aspx](http://www.microsoft.com/germany/sql/2008/default.aspx)  
 [4] Oracle; [www.oracle.com/lang/de/database/index.html](http://www.oracle.com/lang/de/database/index.html)