Microsoft®
SQL Server™ 2005

ADD.NET 2.0 기능 매트릭스

Microsoft®

요약

ADD.NET 2.0 에는 새로운 기본 클래스 기반의 공급자 모델, 모든 공급자를 위한 기능, System.Data.SqlClient 에 대한 변경 사항 등이 포함되어 있습니다. 새로운 기능에 대한 개요, 활용 사례, 공급자 독립적인 기능 그리고 SqlClient 고유 기능을 일목 요연하게 설명한 표도 제공됩니다.(14페이지)

ADD.NET 2.0 에는 기본 클래스 기반의 새로운 공급자 모델과 모든 공급자가 활용할 수 있는 기능은 물론, System.Data.SqlClient 에 대한 변경 사항 등을 비롯한 많은 새로운 기능이 포함되어 있습니다. .NET Framework 2.0 은 SQL Server 2005 와 함께 출시되기 때문에 이들 기능 중 일부는 SQL Server 2005 가 지원되는 경우에만 사용할 수 있습니다. 본 기술 문서는 새로운 기능에 대한 개요, 로드맵 및 활용 사례를 제공하기 위한 것으로 공급자 독립적인 기능과 SqlClient 고유 기능에 대한 표를 포함하고 있습니다. 시리즈 형식으로 제공될 다음 기술 문서에서는 일부 기능에 대해 보다 자세하게 다루는 한편, DataSet 및 자매품의 새로운 기능에 대해 설명할 것입니다.

Contents

기본 클래스 기반의 공급자 모델

ADO.NET 1.0 및 1.1에서는 공급자의 개발자들이 일련의 공급자별 클래스를 구현했으며 각 클래스가 일반 인터페이스를 구현한 경우에만 일반 코딩을 수행할 수 있었습니다. 예를 들어 System.Data.SqlClient에는 SqlConnection 클래스가 포함되어 있으며 이 클래스가 DbConnection을 구현합니다. System.Data.OracleClient에는 IdbConnection을 구현하는 OracleConnection 클래스가 포함되어 있습니다. 공급자별 클래스는 데이터-소스별 속성과 메소드를 구현합니다. 예를 들어 SqlConnection은 Database 속성과 ChangeDatabase 메소드를 구현합니다. Oracle 데이터베이스는 데이터베이스 인스턴스별 다수의 "데이터베이스" (ANSI SQL의 경우 카탈로그)라는 개념을 채용하지 않기 때문에 OracleConnection에서는 이를 구현하지 않습니다. ADO.NET 2.0의 새로운 공급자 모델은 System.Data.Common에 있는 일련의 기본 클래스를 기반으로 설계됩니다. 이는 공통의 기능으로 이루어진 기본 구현을 제공하고 각 기본 클래스는 백워드 호환을 위해 여전히 필요한 일반 인터페이스를 구현합니다. 공급자의 개발자는 필요에 따라 기본 클래스를 사용하거나 일반 인터페이스를 지원할 수 있습니다.

이전 버전의 경우 이러한 인터페이스 모델에는 DataAdapter/DbDataAdapter와 CommandBuilder라는 2가지 예외가 있었습니다. CommandBuilder 클래스는 간단한 SELECT 명령을 위해 동일한 열(column) 세트를 사용하는 INSERT, UPDATE 및 DELETE 명령을 자동으로 구현합니다. SqlCommandBuilder는 봉인 클래스이기 때문에 실행 명령문을 작성하는 데 사용되는 기본 알고리즘을 유지하는 동시에 CommandBuilder를 확장하는 것이 불가능했습니다. 여전히 SqlCommandBuilder 매개변수 파서를 재사용할 수는 없지만 System.Data.Common에는 DbCommandBuilder 기본 클래스가 존재합니다. 또한, 이들 클래스의 기본 클래스 수준에서 제시되는 새로운 기능을 포함하고 있습니다. DataAdapter/DbDataAdapter 기본 클래스는 SQL Server SqlTypes 같은 공급자 전용 유형을 DataSet(ReturnProviderSpecificTypes 속성)에 지정하기 위한 메커니즘과 배치(batch) 업데이트(StatementType.Batch 열거 값과 UpdateBatchSize 속성)를 위한 메커니즘을 제시합니다. DbCommandBuilder 공통 기본 클래스에는 동시 운영성 정책 옵션을 지시하기 위한 속성(ConflictDetection 속성)이 포함되어 있습니다.

공급자 팩토리

ADO.NET 1.0 및 1.1의 인터페이스 기반 접근법이 복잡한 이유 가운데 하나로 인터페이스 상에서 생성자를 호출할 수 없다는 점을 꼽을 수 있습니다. 사용자는 특정 클래스의 구체적 인스턴스를 생성해야 합니다. OLE DB나 ADO와 같은 이전의 API는 연결 문자열을 오버로드하는 방식으로 이 문제를 해결했습니다. 연결 문자열에는 공급자의 COM PROGID가 포함되어 있기 때문에 이 PROGID를 기초로 올바른 DataSource 클래스를 생성할 수 있었습니다. 이는 레지스트리에 OLE DB DataSource PROGID가 저장되어 있었기 때문에 가능했습니다.

```
' VB6 ADO code, Connection is an interface (actually it' s _Connection)
Dim conn as Connection
' note that the default provider is MSDASQL, the OLE DB provider for ODBC
' this uses the OLE DB provider for SQL Server
conn.ConnectionString = "provider=sqloledb;.." ' other parameters deleted
conn.Open();
```

ADO.NET 2.0은 이에 대한 해결책을 제시합니다. 각 데이터 공급자는 ProviderFactory 클래스와 공급자 문자열을 .NET machine.config에 등록합니다. 기본 클래스인 ProviderFactory(DbProviderFactory)와 System.Data.Common.ProviderFactories 클래스가 있기 때문에 machine.config에 등록된 서로 다른 데이터 공급자에 대한 정보의 DataTable을 반환하고 Data Table에서 공급자 문자열 (ProviderInvariantName)에 해당하는 올바른 ProviderFactory나 DataRow를 검색할 수 있습니다. 작성에 사용되는 조건부 코드는 다음과 같습니다.

```
enum provider {sqlserver, oracle, oledb, odbc};
// determine provider from configuration
provider prov = GetProviderFromConfigFile();
IDbConnection conn = null;
switch (prov) {
  case provider.sqlserver:
    conn = new SqlConnection(); break;
  case provider.oracle:
    conn = new OracleConnection(); break;
  case provider.oledb:
    conn = new OleDbConnection(); break;
  case provider.odbc:
    conn = new OdbcConnection(); break;
  // add new providers as the application supports them
}
...can now be written like this:
  // get ProviderInvariantString from configuration
  string provstring = GetProviderInvariantString();
  DbProviderFactory fact = DbProviderFactories.GetFactory(provstring);
  IDbConnection = fact.CreateConnection();
```

시스템상에 설치된 데이터 공급자를 검색하기 위한 표준과 각각에 대한 ProviderFactory가 등장하면서 새로운 가능성이 열리고 있습니다.

서버 열거

machine.config의 공급자 구성 엔트리는 공급자가 지원하는 기본 클래스 또는 기본 인터페이스를 나타내는 비트마스크(bitmask)를 지정합니다. 모든 데이터 공급자가 System.Data.Common의 모든 기능을 지원할 필요는 없기 때문입니다. 예를 들어, CommandBuilder는 "있으면 좋은" 클래스이지만 없어도 무방합니다.

DbEnumerator는 ADO.NET 2.0의 혼합 클래스에 추가된 새로운 기본 클래스입니다. 이 클래스는 데이터 공급자가 데이터 소스 목록을 입수하기 위해 이를 지원하도록 허용합니다. 예를 들어, SqlClient는 이 클래스를 지원하며 네트워크상에서 제공되는 SQL Server 인스턴스 목록을 반환합니다. 이에 따라 프로그램과 도구는 데이터 소스를 선택하여 사용자를 표시할 수 있습니다. 이러한 클래스를 사용하는 도구 중 하나로 Visual Studio 2005를 들 수 있습니다.

연결 문자열 빌더 및 메타데이터 스키마

지금까지도 Visual Studio .NET은 데이터 소스를 표시할 연결 문자열을 구성하는 데 OLE DB 컴포넌트를 사용하고 있습니다. Visual Studio 2005의 Server Explorer를 사용하여 Visual Studio .NET 2003에 새로운 Data Connection을 추가하는 경우, .NET 데이터 공급자가 아니라 시스템상에 설치된 OLE DB 공급자를 나열하는 OLE DB 연결 문자열 빌더가 표시됩니다. 따라서, 공급자(비록 OLE DB 공급자이더라도)를 선택해서 해당 공급자에 대한 ADO.NET 연결 문자열을 구성할 수 있습니다. Visual Studio 2005에서는 위에서 언급한 바와 같이 DbProviderFactories가 .NET 데이터 공급자 목록을 표시하고 DbConnectionStringBuilder 클래스가 GUI 컴포넌트에 의해 사용되어 프로그래머가 연결 문자열을 그래픽으로 구성하고 구성 파일에서 연결 문자열을 로드, 저장할 수 있도록 지원합니다.

또한, Visual Studio 2005 Server Explorer는 표시할 테이블, 열, 뷰, 저장 프로시저 등의 목록을 비롯하여 데이터 소스 메타데이터를 입수합니다. ANSI SQL 규격에는 INFORMATION_SCHEMA 뷰라는 이 메타데이터에 대한 기본 규격도 포함되어 있습니다. 이들 일반 뷰는 처음 구축 시에는 충분하지만 데이터베이스별 뷰 또는 정보가 추가되면서 확장이 요구되는 경우가 종종 있습니다. ADO.NET 2.0에서는 모든 데이터베이스가 INFORMATION_SCHEMA 뷰를 지원하지는 않기 때문에 데이터 공급자는 사용할 수 있는 메타데이터와 데이터베이스에서 이를 입수할 수 있는 방법이 설명되어 있는 XML 형식의 구성 파일을 제공할 수 있습니다. 이 파일은 도구 프로그래머들이 공급자가 정의한 확장 정보 세트를 입수하는 데 상당한 도움이 될 것입니다. 향후 기술 문서에서는 이 공급자 모델에 추가된 새로운 기능 향상에 대해 자세하게 다룰 것입니다.

추적

이 기능은 프로그래머와 지원 스태프가 데이터베이스 API 호출을 추적하여 사용자의 설명이나 프로그램의 오류 메시지를 토대로 데이터 액세스 스택에서 문제가 있는 부분을 찾아내는 데 매우 유용합니다. 일반적인 문제의 원인은 다음과 같습니다.

1. 클라이언트 프로그램과 실제 데이터베이스 간의 스키마 불일치
2. 데이터베이스 사용 불능 또는 네트워크 라이브러리 문제
3. 하드 코딩되었거나 응용 프로그램에 의해 생성된 부정확한 SQL
4. 부정확한 프로그래밍 로직

ODBC와 같은 일부 API에는 몇 가지 사실 상의 표준이 있음에도 불구하고 지금까지 추적을 허용하는 계측(instrumenting) 코드는 개별 공급자의 작성자에게 맡겨졌습니다. 표준 OLE DB 추적 정보의 부족으로 OLE DB 및 ADO 문제의 해결이 더욱 어려워지고 있습니다. 이는 ADO.NET 전용 아키텍처는 아니지만 ADO.NET 2.0의 Microsoft 공급자는 일반화된 추적 및 계측 API를 활용하고 있습니다. 이 새로운 기능을 활용하여 응용 프로그램 스택의 모든 수준에서 문제를 추적할 수 있습니다. Microsoft ADO.NET 공급자가 계측될 뿐만 아니라 데이터 액세스 스택의 다른 부분도 이 기능을 사용할 수 있으며 공급자의 작성자가 구현하는 데에도 사용할 수도 있습니다. ADO.NET 2.0 DataSet 및 관련 클래스에는 진단 기능이 내장되어 있습니다. 추적 기능에 대한 자세한 내용은 향후 기술 문서에서 다룰 것입니다.

SqlClient 기능 향상

SqlClient은 Microsoft의 대표적 데이터베이스인 SQL Server 전용 공급자입니다. ADO.NET 2.0은 4가지 Microsoft 공급자와 함께 제공됩니다.

- SqlClient-SQL Server를 위한 Microsoft 공급자
- OracleClient-Oracle 데이터베이스를 위한 Microsoft 공급자
- OleDb- ADO.NET에서 OLE DB 공급자를 사용하기 위한 브리지 공급자
- Odbc-ADO.NET에서 ODBC 드라이버를 사용하기 위한 브리지 공급자

ADO.NET 2.0에서는 이들 4개 공급자 모두를 개선하여 부분적으로 신뢰할 수 있는 트러스트 환경에서도 사용할 수 있도록 했습니다. .NET CAS(Code Access Security)를 올바르게 구성하면 보다 데이터 중심적인 모바일 코드 시나리오를 수행할 수 있습니다. ADO.NET 1.1에서는 SqlClient 공급자만이 이 기능을 지원했습니다.

또한 데이터 공급자는 데이터베이스 업체(Oracle의 ODP.NET 및 DB2를 위한 IBM의 데이터 공급자), 공급자 전문업체(DataDirect Technologies), 오픈 소스 프로젝트 및 개인이 작성할 수도 있습니다. Microsoft는 Host Integration Server 2004 제품을 통해 DB2 데이터 공급자를 선보일 예정입니다.

SQL Server는 이러한 소프트웨어 구성에서 매우 중요한 부분을 차지하기 때문에 ADO.NET 2.0에서는 SqlClient 개선에 힘을 기울였을 뿐만 아니라 모든 Microsoft 지원 공급자에 대한 기능 개선이 이루어졌습니다. 이들 기능 중 일부는 SQL Server의 모든 버전을 지원하지만 상당수의 새로운 기능들은 SQL Server 2005에서 선보인 많은 새로운 기능을 지원하기 위한 것으로 "Yukon" 이라는 코드명으로 쉽게 식별할 수 있습니다. SQL Server 2005는 서버 내에서 작동하는 .NET 코드를 지원하며 공급자 모델을 사용하는 서버 내 데이터 액세스를 위한 최적화 기능을 제공합니다. 직접 드러나지는 않지만 중요한 내부적 변화는 ADO.NET 2.0의 SqlClient 데이터 공급자가 MDAC(Microsoft Data Access Components)를 사용하지 않는다는 사실입니다. 또한 SqlClient 데이터 공급자는 네트워크 오류에 대한 보다 명확한 오류 메시지와 전반적으로 보다 세분화된 오류 메시지를 이용하여 보다 효과적으로 오류를 처리합니다. 다음은 프로그래머가 볼 수 있는 SqlClient 고유 기능에 대한 개요를 설명하고 있습니다.

연결 풀링 기능 향상

ADO.NET 1.0은 데이터베이스 연결 풀링을 위해 새로운 인프라를 구축했습니다. Microsoft SqlClient 및OracleClient 데이터 공급자는 이러한 인프라를 사용하지만, OleDb와 Odbc 데이터 공급자는 사용하지 않습니다. 새로운 풀링 메커니즘은 최소/최대 풀 크기, 풀에서 연결을 설정하기 위해 사용자가 정의한 시간 동안 기다릴 수 있는 풀 매니저의 기능 등 세분화된 연결 풀링 매개변수를 지원합니다. ADO.NET은 프로그래밍을 통해 연결 풀을 "드레이닝(drain)" 할 수 있도록 새로운 연결 풀링 기능을 추가했습니다. 즉, 풀러(pooler)에 의해 설정된 현재의 모든 연결을 종료할 수 있습니다. 정적(Visual Basic .NET에서 공유) 메소드인 SqlConnection.ClearPool을 사용하여 특정 연결 풀을 해제하거나 SqlConnection.ClearPools 메소드를 사용하여 appdomain의 모든 연결 풀을 해제할 수 있습니다. SqlClient와 OracleClient 모두 이 기능을 구현하고 있습니다.

비동기 명령

때때로 클라이언트 또는 미들웨어 코드에서 한번에 1개 이상의 작업을 실행해야 하는 경우가 있습니다. 기본적으로 멀티스레드된 미들웨어 코드에서 이는 처리량을 향상시키는 중요한 요인입니다. ADO.NET 2.0의 경우, SqlClient가 비동기 명령 실행을 지원합니다.
비동기 실행을 위한.NET 패러다임은 동기 연산을 위한 메소드는 물론, 연산을 위한 Begin 및 End 메소드 세트를 제공하는 것입니다. 데이터베이스 명령 실행에는 많은 시간이 소요되기 때문에 SqlClient는 비동기 실행을 수행하는 SqlCommand 메소드를 내장하고 있습니다.
다음 표는 비동기 실행과 그에 해당하는 동기 실행을 지원하는 메소드를 나타낸 것입니다.

동기 메소드	비동기 메소드
ExecuteNonQuery	BeginExecuteNonQuery, EndExecuteNonQuery
ExecuteReader	BeginExecuteReader, EndExecuteReader
ExecuteXmlReader	BeginExecuteXmlReader, EndExecuteXmlReader

비동기 실행은 훌륭한 기능이기는 하지만 불필요하게 사용해서는 안됩니다. 명령이 장시간 실행되며 동시에 실행하는 것이 유용한 작업이 있는 경우에만 사용하십시오. Windows NT 운영 체제 제품군의 Windows 스레드 스케줄러(Windows 9x 및 Me 클라이언트에서는 지원되지 않는 기능)는 스레드 간을 전환할 때 자체 오버헤드가 발생합니다. 또한 일부 .NET 라이브러리는 스레드에 민감하다는 점을 유념해야 합니다. 비동기 실행의 경우 연산을 시작하는 데 사용한 스레드가 종료하는 데 사용한 스레드와 반드시 같을 필요는 없습니다. 지만, SQL Server 네트워크 라이브러리 스택은 I/O 완료 포트를 통해 비동기 실행을 지원하도록 개선되어 왔으며 이를 통해 비동기 SQL Server 연산을 위한 처리량도 높아졌습니다. 비동기 연산은 여러 실행 문 및 저장 프로시저 실행에 효과적일 뿐만 아니라 SQL Server 2005의 여러 활성 resultset 기능과 함께 사용하면 단일 데이터베이스 연결을 통해 비동기 SELECT 문을 멀티플렉싱할 수 있습니다.

일괄 가져오기

많은 데이터베이스 응용 프로그램이 대규모 배치(batch)에서 SQL Server에 행을 신속하게 삽입(INSERT)할 수 있습니다. 대표적 예로 전화 스위치나 병원 환자 모니터와 같은 하드웨어 장치에서 판독한 정보에 따라 행을 SQL Server에 삽입하는 응용 프로그램을 들 수 있습니다. SQL Server는 유틸리티(bcp 같은)와 함께 제공되지만 일반적으로는 입력을 위해 파일을 사용합니다.

SqlClient에는 SqlBulkCopy라는 새로운 클래스가 포함되어 있습니다. 이 클래스는 파일에서 입력값을 직접 사용해 BCP 같은 파일 출력을 생성하는 대신, 클라이언트에서 데이터베이스에 많은 행을 신속하고 효율적으로 삽입하는 방법을 채택하고 있습니다. SqlBulkCopy는 DataReaders 및 DataSets에서 입력값을 입수할 수 있습니다. 따라서, 공급자(DataReader)에서 일련의 행을 직접 스트리밍할 수 있을 뿐만 아니라, 공급자가 아닌 하드웨어 장치에서 입수한 외부 데이터를 DataSets에 입력하고 이를 직접 업데이트할 수 있습니다. 이 경우, 그 어떤 공급자도 소스로서 요구되지 않습니다.

```
// Fill up a DataSet
DataSet ds = new DataSet();
FillDataSetFromHardwareDevice(ds);
// Copy the Data to SqlServer
string connect_string = GetConnectStringFromConfigFile();
SqlBulkCopy bcp = new SqlBulkCopy(connect_string);
bcp.DestinationTableName = "hardware_readings" ;
bcp.WriteToServer(ds);
```

공급자 통계

일부 응용 프로그램 작성자들은 응용 프로그램에서 "실시간" 모니터링이 유용하다는 사실을 인식하게 되었습니다. 물론 Windows Performance Monitor를 사용하거나, 자체 성능 클래스를 정의하거나 아니면 내부(시간이 지남에 따라 약해질 수도 있지만) SQL Server 메타데이터 호출을 사용하여 이러한 정보를 입수할 수도 있지만 SqlClient는 현재 기본 지원되는 방식을 통해 이러한 정보를 제공하고 있습니다. SqlConnection 클래스의 인스턴스 메소드를 통해 ODBC API에서 제공되는 것과 유사한 연결당 통계를 수집할 수 있습니다. 이러한 통계의 저장 및 수집에는 자체 오버헤드가 발생하기 때문에 통계 수집 기능을 토글링(toggling)하는 데 사용할 수 있는 속성이 제공됩니다. 또한, 카운터 리셋을 위한 메소드도 있습니다. 통계값 수집 기능은 오프 상태로 기본 설정되어 있지만 풀링 시나리오에서 Dispose나 Close를 호출하여 연결 풀에 연결을 재설정하면 활성 상태가 됩니다. 다음은 생성된 통계값의 예제를 보여주고 있습니다.

```
string connect_string = GetConnectStringFromConfigFile();
SqlConnection conn = new SqlConnection(connect_string);
conn.Open();
// Enable
conn.StatisticsEnabled = true;
// do some operations
//
SqlCommand cmd = new SqlCommand("select * from authors" , conn);
SqlDataReader rdr = cmd.ExecuteReader();
Hashtable stats = (Hashtable)conn.RetrieveStatistics();
// process stats
IDictionaryEnumerator e = stats.GetEnumerator();
while (e.MoveNext())
Console.WriteLine("{0} : {1}" , e.Key, e.Value);
conn.ResetStatistics();
Connection-specific statistics
BuffersReceived   : 1
BuffersSent       : 1
BytesReceived     : 220
BytesSent         : 72
ConnectionTime    : 149
CursorFetchCount  : 0
CursorFetchTime   : 0
CursorOpens       : 0
```

```
CursorUsed       : 0
ExecutionTime    : 138
IduCount         : 0
IduRows          : 0
NetworkServerTime : 79
PreparedExecs    : 0
Prepares         : 0
SelectCount      : 0
SelectRows       : 0
ServerRoundtrips : 1
SumResultSets    : 0
Transactions     : 0
UnpreparedExecs  : 1
```

이러한 통계값이 의미하는 바를 정확하게 알고 싶다면 ADO.NET 2.0이나 ODBC 문서를 참조하십시오.

AttachDbFileName

SqlClient 데이터 공급자는 데스크탑 응용 프로그램(데이터베이스가 사용자의 데스크탑에 저장)을 비롯하여 클라이언트-서버 및 미들웨어 기반 응용 프로그램을 지원합니다. MSDE라는 SQL Server의 특별 버전도 있습니다. SQL Server 2005 버전에서 이 제품의 명칭은 SQL Server 2005 Express Edition입니다. 데스크탑 응용 프로그램에서 데이터베이스 자체는 응용 프로그램 전용이며 응용 프로그램과 함께 번들로 제공됩니다. 응용 프로그램 셋업 프로그램은 SQL Server Express 시스템에서 실행되기 때문에 사용자는 데이터 리포지토리로 SQL Server가 사용되고 있다는 사실 조차 인식하지 못할 수 있습니다.

ADO.NET 1.0은 응용 프로그램 내부의 SQL Server Express 인스턴스에 데이터베이스 파일을 손쉽게 연결할 수 있도록 AttachDbFileName이라는 연결 문자열 매개변수를 제공했습니다. 하지만 이 매개변수는 하드 코딩된 pathname으로 지정되어야 하기 때문에 사용자들은 기본 지정된 위치가 아닌 곳에 응용 프로그램을 설치하기 어렵습니다. ADO.NET 2.0에서는 AttachDbFileName 매개변수가 상대적 경로가 될 수 있으며 응용 프로그램 구성 설정값과 함께 사용됩니다. 따라서, SQL Server Express에서 데스크탑 응용 프로그램을 셋업하는 것은 Microsoft Access 파일 기반 데이터 스토어로 연결하는 것 만큼 쉽습니다.

SQL Server 2005- SqlClient 고유 기능

MARS

독립적으로, 또는 저장 프로시저 내부에서 SQL SELECT 문을 사용하여 행 세트를 선택하면, SQL Server는 일부 데이터베이스와 같이 행 세트 위에 자동으로 커서를 생성하지 않습니다. 대신, 최적화된 메소드를 사용하여 네트워크 전반에서 resultset을 스트리밍하기 때문에 경우에 따라 네트워크 라이브러리가 패킷 크기의 청크로 데이터를 추출할 때 데이터베이스 버퍼에서 직접 읽기를 수행합니다. SQL Server Boks Oline에서는 이를 "SQL Sever의 기본 결과 집합" 또는 "커서없는 결과 집합" 이라고 합니다. SQL Server 2005 이전 버전에서는 단일 연결에서 한번에 하나의 커서없는 결과 집합만이 활성화될 수 있었습니다.

서로 다른 데이터베이스 API 및 라이브러리는 단일 연결/단일 커서없는 결과 집합의 작업을 각기 다른 방식으로 처리합니다. 두 번째 커서없는 결과 집합을 열려고 시도하면 ADO.NET 1.0 및 1.1은 오류를 발생시킵니다. 실제로 ADO "클래식" 은 이면에서 새로운 데이터베이스 연결을 설정했습니다. 오류를 발생시키는 것 보다는 새로운 데이터베이스 연결을 설정하는 것이 "정답" 은 아니지만 훨씬 편했습니다. 이러한 편리한 기능은 의도하지 않게 일부 프로그래머에 의해 남용되었으며 예상보다 많은 데이터베이스 연결을 초래했습니다.

SQL Server 2005에서는 단일 연결에서 한번에 여러 개의 커서없는 결과 집합이 활성화될 수 있도록 데이터베이스가 개선되었습니다. "MARS(Multiple Active Resultsets)" 라는 기능은 바로 이러한 노력의 결실입니다. 이 기능을 지원하기 위해 네트워크 라이브러리에 대한 변경이 이루어졌으며 MARS를 실행하기 위해서는 새로운 네트워크 라이브러리와 새로운 데이터베이스가 모두 필요합니다.

SqlClient 코드에서는 여러 SqlCommand 인스턴스가 동일한 연결을 사용하기 때문에 결과 집합을 멀티플렉싱할 수 있습니다. 각 SqlCommand는 Command.ExecuteReader 호출을 통해 생성된 SqlDataReader를 수용할 수 있으며 여러 SqlDataReader를 함께 사용할 수 있습니다. ADO.NET 1.0 및 1.1의 경우, 여러 개의 SqlCommand가 사용되고 있다 하더라도 사용 중인 SqlDataReader를 먼저 종료해야만 다른 SqlDataReader를 사용할 수 있습니다. 동일한 SqlCommand 인스턴스에서 다중 ExecuteReader 호출로 인해 생성된 SqlDataReader는 멀티플렉싱할 수 없다는 점에 유의하십시오. 다음은 크게 유용하지는 않지만 간단하게 이해를 돕는 예입니다.

```
// connection strings should not be hardcoded
string connstr = GetConnStringFromConfigFile();
SqlConnection conn = new SqlConnection(connstr);
SqlCommand cmd1 = new SqlCommand(
  "select * from employees" , conn)
SqlCommand cmd2 = new SqlCommand(
  "select * from jobs" , conn)
SqlDataReader rdr1 = cmd1.ExecuteReader();
// next statement causes an error prior to SQL Server 2005
SqlDataReader rdr2 = cmd2.ExecuteReader();
// now you can reader from rdr1 and rdr2 at the same time.
```

이 기능은 오류를 줄이거나 ADO 라이브러리의 강점을 규명하기 위한 것만은 아닙니다. 위에서 설명한 비동기 실행에서 함께 사용하면 매우 유용합니다. 여러 개의 비동기 SELECT 문이나 저장 프로시저 호출을 동시에 실행하여써 데이터베이스 연결을 줄이고 처리량을 최적화할 수 있습니다. 단일 연결을 이용하여 동시에 단일 형식으로 20개의 드롭다운 목록 박스를 채운다고 생각해 보십시오. 하나의 결과 집합이 활성화된 동안 non-resultset-returning 문을 실행할 수도 있습니다.

동시에 여러 개의 실행 스트림이 활성화될 수 있지만 트랜잭션이 존재할 경우 모든 실행 스트림은 동일한 트랜잭션을 공유해야 합니다. 트랜잭션은 여전히 명령 범위가 아니라 연결 범위로 설정됩니다. SqlCommand Transaction 속성을 기존 버전의 ADO.NET과 동일하게 설정하면 SqlTransaction 인스턴스를 SqlCommand와 연결할 수 있습니다.

SqlDependency 및 SqlNotificationRequest

다른 사람이 데이터베이스의 행을 변경했다는 사실을 기초로 캐시를 새로 고칠 수 있는 미들티어 캐싱 상황에서 매우 유용한 기능입니다. 이를 위해 프로그래머는 테이블이나 뷰 변경 시 파일을 업데이트하는 트리거를 작성하거나 데이터베이스 변경 여부에 관계없이 캐시 새로 고침을 실행하는 것을 비롯하여 몇가지 기법을 활용했습니다. SqlClient SqlNotificationRequest 및SqlDependency클래스는 데이터베이스 통보에 등록하는 가장 간편한 방법입니다.

SqlDependency는 SqlNotificationRequest를 포함하고 통보 정보를 .NET 이벤트로서 표시하는 상위 클래스입니다. SqlNotificationRequest를 사용하면 이벤트가 없으며 통보를 등록하고 정보를 자체적으로 수집하는 "대대적 작업"을 수행할 필요가 없습니다. 상당 수의 프로그래머들이 SqlDependency를 사용할 것으로 보입니다. SqlDependency는 독립적으로 사용될 수 있으며 ASP.NET Cache 클래스를 사용할 경우 그 기능을 직접 사용할 수 있습니다.

이러한 SQL Server 2005 고유의 기능은 확장형 큐잉 시스템을 구현하는 새로운 기능인 SQL Server Service Broker를 활용합니다. ASP.NET Cache 클래스를 사용하는 경우, 유사한 기능을 구현하려면 Service Broker 대신 데이터베이스 폴링(polling)을 사용해야 한다는 점을 유념하십시오. Service Broker와 SQL Server 2005를 사용하면 통보를 받기 위해 데이터베이스에 대한 연결을 유지해야 할 필요가 없습니다. SqlDependency는 사용자가 선택한 HTTP나 TCP 프로토콜을 사용하여 기준 행이 변경되면 알려 줍니다. 각 행 단위의 정보는 통보에 포함되지 않기 때문에 통보를 받으면 전체 행 세트를 다시 가져와 통보를 위해 재등록해야 합니다.

이 기능은 단일 캐시나 제한된 수의 사용자에게 적합한 것이며 대규모 동시 사용자를 대상으로 하는 경우에는 각별한 주의를 기울여야 합니다. 각 사용자는 열이 변경되면 캐시의 전체 열 세트에 대한 새로 고침을 실행해야 합니다. 변경 사항이나 사용자가 많을 경우 데이터베이스에서 새로 고침에 사용된 SELECT 문의 조회 수가 증가할 수 있습니다.

암호 변경

SQL Server 2005는 로그인(SQL Server에 연결된 Windows 로그인)을 통합한 다른 암호와 마찬가지로 만료 정책을 따르는 SQL 로그인을 사용하는 메커니즘을 제공합니다. 이 기능을 사용하기 위해서는 Windows Server 2003에서 실행하는 SQL Server 2005가 필요합니다. SQL 로그인 암호('sa' 같은)가 만료되면 기존 Windows 메커니즘과 암호 변경 API를 사용해서 변경할 수 없습니다. Transact SQL ALTER LOGIN 동사의 호출을 종료하는 SQL Server 클라이언트를 사용해야만 이 암호를 변경할 수 있습니다.

SqlClient는 SqlConnection 클래스의 ChangePassword 메소드를 통해 이를 수용합니다. SQL Server 2005 인스턴스에 대해 실행된 경우에만 이 메소드를 사용할 수 있다는 점에 유의하십시오. 기존 버전의 데이터베이스에서 SQL 로그인을 변경할 수도 있지만 이 API는 다른 버전의 SQL Server가 지원하지 않는 네트워크 패킷 유형을 사용합니다. 암호 변경에 있어 고려해야 할 또 다른 측면은 프로그램의 연결 문자열에서 더 이상 SQL Server 로그인 ID와 암호를 하드 코딩할 수 없다는 점입니다. .NET IL(Intermediate Language)를 생성하여 암호가 변경될 때마다 실행 파일을 교체(현재 지원되는 옵션이 아님)하지 않는 경우라면 구성 파일에 SQL Server 암호를 저장해야 합니다. 신중한 SQL Server 개발자들은 일시적으로 (되도록 부호화된) 암호를 저장하기 위해 구성 파일을 사용하고 있습니다. 보다 효과적인 것은 항상 SQL Server 통합 보안 기능을 사용하는 것입니다.

System. Transactions 통합

ADO.NET 2.0의 SqlClient 공급자는 프로모션 가능한 트랜잭션이라는 기능을 지원하여 새로운 System.Transactions 네임스페이스를 통합했습니다. 로컬 또는 분산 트랜잭션(BEGIN TRANSACTION, BEGIN DISTRIBUTED TRANSACTION)을 시작하기 위해 Transact SQL을 사용할 수도 있지만 특히 프로그래머가 단일 데이터베이스나 여러 데이터베이스 시나리오에서 사용할 수 있는 컴포넌트를 작성하기를 원하는 클라이언트측/미들티어 프로그래밍에 유용합니다. 이들 시나리오에는 다수의 SQL Server 인스턴스가 포함될 수 있으며, SQL Server는 다수의 인스턴스 액세스를 자동으로 탐지하여 로컬에서 다중 인스턴스(분산)로 트랜잭션을 "확장" 할 수 있습니다. 첫번째 데이터베이스(분산 트랜잭션 토폴로지에서 자원 관리자로 알려진)가 SQL Server인 경우에는 다수의 데이터베이스 제품이나 다수의 연결이 사용된 경우에도 가능합니다. 확장 가능한 트랜잭션은 ADO.NET에서 기본적으로 지원됩니다.

클라이언트 장애 조치

SQL Server 2005는 데이터베이스 미러링을 통해 "핫 스페어(hot spare)" 기능을 지원합니다. SQL Server 인스턴스에 장애가 발생한 경우 작업은 백업 서버로 자동으로 이동될 수 있습니다. 이를 위해서는 witness instance라고 하는 장애 조치를 감시하는 인스턴스가 필요합니다. 핫 스페어 시나리오에 따르면 기존 클라이언트 연결은 장애 조치(새로운 서버 인스턴스와 연결 설정)를 "인식" 해야 합니다. 다음으로 시도된 액세스에서 오류를 발생시키고 클라이언트 프로그래밍에 의해 수동으로 "장애 조치" 되어야 하는 클라이언트 연결은 차선책입니다. ADO.NET 2.0의 SqlClient는 응용 프로그램의 특별한 프로그래밍 없이도 클라이언트 장애 조치를 지원합니다.

새로운 트랜잭션 격리 수준 지원

SQL Server 2005는 잠금 기능과 버전 관리 등 2가지 메소드를 통해 트랜잭션 격리를 수행합니다. 이전 버전의 SQL Server는 잠금 기능은 지원했지만 버전 관리는 지원하지 않았습니다. 문장 수준의 버전 관리와 트랜잭션 수준의 버전 관리 등 2가지 유형의 버전 관리 기능이 제공되며 극단적인 상황에서 잠금 기능을 선택적으로 줄이고 버전 관리 데이터베이스를 위해 설계된 응용 프로그램을 손쉽게 변환할 수 있도록 지원합니다. 버전 관리 데이터베이스를 위해 설계된 응용 프로그램을 잠금 기능 데이터베이스로, 또는 그 반대로 포팅하는 경우, 대대적인 변경 작업을 수행해야 합니다. 버전 관리 데이터베이스의 기본 동작은 문장 수준의 버전 관리와 상당 부분 일치합니다. 세부적인 차이점은 Beauchemin, Berglund, Sullivan 공저의 "A First Look at SQL Server 2005 for Developers" 를 참조하십시오.

서로 다른 2가지의 버전 관리와 서로 다른 잠금 기능은 특정 수준의 트랜잭션 격리를 통해 트랜잭션을 시작할 수 있다는 것을 의미합니다. ANSI SQL 규격에서는 다음과 같이 4가지의 트랜잭션 격리 수준을 정의하고 있습니다.

- READ UNCOMMITED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE

SQL Server2005는 이전 버전과 마찬가지로 4가지 격리 수준을 모두 지원합니다. 일반적으로 버전 관리 데이터베이스는 READ COMMITTED와 SERIALIZABLE만 지원합니다. 버전 관리 데이터베이스에서 READ COMMITTED는 문장 수준의 버전 관리를, SERIALIZABLE는 트랜잭션 수준의 버전 관리를 구현합니다. 잠금 기능을 사용하든 버전 관리를 사용하든 관계없이 거의 모든 데이터베이스에서 READ COMMITTED는 기본 동작으로 설정되어 있습니다.

문장 수준의 버전 관리가 활성화되어 있으면 각 데이터베이스 별로 데이터베이스 옵션을 설정하여 이를 기본 동작으로 설정할 수 있습니다. 또한, IsolationLevel.ReadCommitted나 IsolationLevel.ReadUncommitted를 지정하여 이러한 동작을 설정할 수 있습니다. 트랜잭션 수준의 격리를 지원하기 위해서 SQL Server 2005는 새로운 격리 수준인 IsolationLevel.Snapshot을 정의합니다. SqlClient만이 유일하게 이러한 격리 수준을 지원하고 있습니다. 문장 수준의 버전 관리와 트랜잭션 수준의 버전 관리를 각기 따로 활성화할 수 있기 때문에 이러한 격리 수준이 필요했던 것이며, IsolationLevel.Serializable은 SQL Server가 잠금 동작에 대응하는 데 이미 사용되고 있습니다.

데이터 유형 - UDT, XML 데이터 유형, "MAX" BLOB 및 CLOB

SQL Server 2005는 사용자 정의 유형의 지원, 기본 XML 데이터 유형 및 대규모 데이터 지원을 추가했습니다. Transact-SQL 유형인 VARCHAR(MAX), NVARCHAR(MAX) 및 VARBINARY(MAX)를 사용함으로써 대용량 데이터 지원 기능을 향상시킬 수 있었습니다. 사용자 정의 유형 및 기본 XML 유형은 SQL:1999 및 SQL:2003 규격에 정의되어 있습니다. SqlClient에서 이러한 데이터 유형을 사용하기 위해 System.Data.SqlTypes 네임스페이스의 새로운 클래스가 정의되고(SqlUdt 및 SqlXml), SqlDbTypes 열거에 대한 추가 지원이 이루어졌으며 .NET Object 유형으로 UDT를 반환하고 .NET String 유형으로 XML을 반환하도록 IDataReader.GetValue가 개선되었습니다.

이러한 새로운 SQL Server 2005 유형은 SQL SELECT 문에서 반환된 DataReaders와 SqlParameter를 사용하는 매개변수로서 지원됩니다. 특별 클래스인 SqlMetaData는 강력한 형식의 XML 열이 고수하는 XML 스키마 집합이나 UDT의 데이터베이스 이름 같은 새로운 데이터 유형의 확장 속성에 대한 정보를 반환할 수 있습니다. 일반 코드의 경우 클라이언트에서 직접 이들 유형을 사용할 수 있으며 DataSet에서도 마찬가지입니다. 마지막으로 클라이언트에서 "MAX" 데이터 유형에 대한 부분 업데이트를 수행할 수 있습니다. 이를 위해서는 ADO.NET 2.0 이전의 특별 SQL 함수를 사용해야 합니다. 향후 보다 자세하게 다른 기술 문서를 이 사이트에서 제공할 것입니다

결론

여러분들은 속속 발표되는 많은 새로운 기능들을 끊임없이 살펴 보고 익혀야 합니다. 이와 같이 많은 새 기능을 이해하는 데 어려움을 겪지 않도록 돕기 위해 각각의 새 기능과 이를 사용하는 데 필요한 데이터베이스, 공급자 및 버전을 나타내는 표를 제공하는 것으로 이 글을 마무리하겠습니다. 현재까지 필자는 ADO.NET의 일부인 4가지 공급자에 관한 정보만을 가지고 있지만 머지 않아 여타 다른 공급자 벤더들이 조만간 이 대열에 합류하게 될 것입니다. 향후 기술 문서에서는 이 도표가 더욱 확장되기를 기대합니다.

새로운 기능 지원 여부

	모든 공급자	SQL Server 2000	SQL Server 2005
공급자 팩토리	×	×	×
부분적으로 신뢰할 수 있는 환경에서 작동	×	×	×
서버 열기	×	×	×
연결 문자열 빌더	×	×	×
메타데이터 스키마	×	×	×
배치 업데이트 지원	×	×	×
공급자별 유형	×	×	×
충돌 탐지	×	×	×
추적 지원	×	×	×
풀링 기능 향상	SqlClient 및 OracleClient	×	×
MARS			×
SqlNotificationRequest			×
SqlDependency			×
비동기 명령		×	×
클라이언트 장애 조치			×
일괄 가져오기		×	×
Password Change API			×
통계		×	×
새로운 데이터 유형			×
Promotable Tx		×	×
AttachDbFileName		×	×

Bob Beauchemin은 DevelopMentor의 강사이며 교육 과정 저작자이며 데이터베이스 커리큘럼 과정 담당자입니다. 25년 이상 데이터 기반 분산 시스템의 설계자, 프로그래머 및 관리자로서 활동했습니다. Microsoft Systems Journal과 SQL Server Magazine 등에서 ADO.NET, OLE DB 및 SQL Server에 관한 기술 문서를 집필하고 있으며, "A First Look at SQL Server 2005 for Developers and Essential ADO.NET"의 저자이기도 합니다.

문서 번호. SQL-200509-19

Microsoft®

한국마이크로소프트(유)

서울특별시 강남구 대치동 892번지 포스코센터 ▌전화 : 080-985-2000 ▌인터넷 : www.microsoft.com/korea