Microsoft

Microsoft SQL Server 2005

SQL Server 2005 Integration Services (데이터 통합) 포켓북

저자 약력

한대성 에이디 컨설팅 책임 컨설턴트
- KTF dosirak 서비스, 영동세브란스병원, 삼성생명, 한국석유화학 등 튜닝 컨설팅
- SK 커뮤니케이션즈 싸이월드, 두산산업개발 2005 POC 수행
- KTF dosirak 통계 DB 구축, 에이스 아메리칸 화재해상보험 SSIS 프로젝트 수행
- MCPWorld, Technet 세미나 강의, 교육
- 옥션 DBA, 데이터 웨어하우스 및 웹로그 시스템, OLAP 시스템 구축 및 운영
- Microsoft SQL 커뮤니티 사이트 SQLLeader.com(www.sqlleader.com) 운영자

감수자 약력

하성희 에이디 컨설팅 대표 컨설턴트 겸 SQL Server MVP
- SQL Server 컨설팅, 세미나 강의, 번역
- 옥션 DBA팀장
- 마이크로소프트 SQL Server 기술지원 엔지니어 겸 Data Access 팀장
- 포스데이타 SYBASE 기술 지원 엔지니어
- POSCO DBA

서문

5년이라는 기다림 끝에 출시된 SQL Server 2005는 그 기대만큼이나 놀라운 기능과 다양한 편의성을 제공하고 있습니다. 그 중에서도 비즈니스 인텔리전스와 관련된 서비스인 Integration Services, Analysis Services, Reporting Services는 단순한 기능의 업그레이드 수준이 아닌 완전히 새로운 서비스로 간주될 만큼 많은 부분들이 향상되었습니다.
처음 Integration Services를 접했을 때 "우수하다" 라는 생각 외에도 "복잡하다", "어렵다" 라는 느낌을 많이 받았습니다. 하지만, 조금씩 익숙해지면서 어려움 보다는 여러 기능의 우수함과 편의성에 더욱 더 놀라고 만족해하고 있습니다.
나름대로 업무에서 익힌 경험과 Integration Services를 사용하면서 알게 된 사항들을 바탕으로 ETL 툴에 생소하거나 Visual Studio환경에서 익숙하지 않은 분들께 조금이나마 이러한 어려움을 덜어드리고자 본 가이드 북을 집필하게 되었습니다. 개념적인 설명보다는 구체적인 예나 [따라하기]와 같은 실습을 통한 방식으로 설명하였기에 군데군데 시족들이 많이 있지만 가능한 한 쉽게 설명하려고 한 의도로 생각해 주시기 바랍니다.
부족한 부분이 눈에 많이 띄지만 SQL Server 2005 Integration Services라는 아주 우수한 서비스를 이용하는데 미약하나마 도움이 되었으면 하는 바램으로 본 가이드북을 마칩니다.

(2007년 1월)

컨텐츠 관련 문의처

dshan@adconsulting.co.kr

목차

Integration Services 소개

Integration Services 소개

SQL Server Integration Services(SSIS)는 SQL Server 2005에 포함된 서비스 중 하나입니다. Integration Services는 Analysis Services(SSAS), Reporting Services(SSRS)와 더불어 비즈니스 인텔리전스(Business Intelligence) 영역의 서비스이며 SQL Server 7.0과 SQL Server 2000의 **데이터 변환 서비스(DTS)**와 유사한 기능을 제공합니다.

SSIS에는 서로 다른 서버나 데이터베이스 간의 데이터 전송 작업이나 데이터 변환 및 가공 작업 등을 효과적으로 수행할 수 있는 데이터 처리 작업 개체들 외에도 파일 시스템 작업이나 FTP 작업, 웹 서비스 작업, XML 작업 등과 같이 데이터 처리 과정에서 필요한 여러 부수적인 기능들을 수행할 수 있는 다양한 작업 개체들도 제공됩니다.

Integration Services의 사용 범위

SSIS는 『데이터 가져오기 / 내보내기』와 같이 단순히 테이블의 데이터를 전송하거나 변환하는 작업뿐만 아니라 데이터와 관련된 여러 분야에서 사용할 수 있는 서비스입니다.

대표적으로 다음과 같은 분야에서 사용할 수 있습니다.

- **원격 데이터 저장소에 존재하는 데이터 통합**
  다른 서버에 존재하는 SQL 데이터뿐만 아니라 이기종 DBMS의 데이터들을 통합하거나 이용하는 작업에서 효과적으로 사용할 수 있습니다. Integration Services에서는 MSSQL 데이터 외에도 텍스트 파일, 엑셀 파일 또는 Oracle, SYBASE, DB2 등과 같은

다른 DBMS의 데이터들도 모두 사용할 수 있습니다. 다양한 형태의 데이터 원본에 대해서 동일한 방식으로 데이터를 처리할 수 있으며 처리된 데이터 역시 다양한 형태의 데이터 대상으로 출력할 수 있습니다. OLE DB 연결, ODBC 연결, .NET 연결 외에도 XML 연결, 웹 서비스 연결 등도 기본적으로 제공되며 필요한 경우 데이터 공급자를 쉽게 등록하여 이용할 수 있기 때문에 어떠한 형태의 데이터라도 모두 사용할 수 있습니다.

- **데이터 정재 작업 및 표준화 작업**

  조직에서 발생하는 데이터의 종류가 많아지고 사용 범위가 넓어지면서 각기 다른 방식이나 기준에 따라 발생된 데이터들을 정리하거나 분류하는 작업을 수행해야 할 경우가 자주 발생합니다. 데이터 웨어하우스(DW)나 경영 정보 시스템(MIS) 등과 같은 대규모 데이터 시스템 구축 작업 또는 기존 데이터베이스의 통합(Consolidation), 이전(Migration)등과 같은 작업등을 수행해야 할 경우, 잘못된 데이터나 기준에 맞지 않는 데이터를 정재하고 표준화 하는 작업은 반드시 수행해야 하는 과정입니다.

  Integration Services에는 데이터 정재 및 표준화 작업을 위한 유사항목 조회, 유사항목 그룹화, 문자표 등과 같은 다양한 변환 개체들이 있습니다. 이러한 개체들을 이용하여 정해진 기준에 따라 데이터를 분류하고 처리하는 정재 작업을 간단히 수행할 수 있습니다.

- **관리 기능 및 데이터 로드 자동화**

  데이터를 이용하거나 처리하는 작업뿐만 아니라 데이터베이스의 관리나 시스템 모니터링 등과 같은 관리 및 자동화 작업 등에서도 Integration Services를 효과적으로 이용할 수 있습니다. Integration Services에는 데이터베이스 복사 또는 이전 작업, 데이터베이스 개체 복사, 인덱스 관리, 통계 관리 등을 수행할 수 있는 작업 개체들이 제공되기 때문에 복잡한 명령어를 이용하여 수행하는 대신 쉽고 간결하게 관리 작업을 설정하고 수행할 수 있습니다.

- **데이터 웨어하우스 및 데이터 마트 적재**

  Integration Services는 다양한 형태의 데이터 원본으로부터 읽어온 데이터를 정해진 기준에 따라 변환하고 처리하여 데이터 웨어하우스 또는 데이터 마트와 같은 대상에 적재하는 방식의 작업에서 우수한 성능과 사용 편의성을 제공합니다.

  입력 데이터에 대해 임시 저장소(Staging Area)의 사용 없이 직접 메모리 버퍼에서 변환과 연산을 수행하기 때문에 불필요한 디스크 I/O가 많이 감소되어 기존 방식에 비해 처리 성능이 우수합니다. 또한 느린 변경 차원(Slowly Changing Dimension)에 대한 처리를 수행할 수 있는 마법사 형태의 변환 작업이나 캐싱을 사용한 조회 변환, 조건부 분할, 멀티캐스트 등과 같이 데이터 웨어하우스의 처리 프로세스에서 자주 사용되는 여러 변환 개체들이 제공되기 때문에 사용자는 복잡한 처리 프로세스를 쉽게 구축할 수 있습니다.

  이외에도 원본 데이터 파일을 복사해 오거나 관리하는 작업이나 데이터 적재 후 Analysis Services의 디멘젼과 큐브를 처리하는 작업, 관리자 또는 업무 담당자에게 메일을 발송하는 작업, 메시지 큐에 결과를 저장시키는 작업 등도 모두 하나의 패키지 내에서 수행할 수 있습니다.

Integration Services의 향상된 기능

SQL Server 7.0의 DTS에서 SQL 2000 DTS를 거쳐 SQL 2005 SSIS로 발전되면서 상당히 많은 기능들이 향상되었으며, ETL(Extract Transformation Loading) 작업에 필요한 부가적인 기능들이 추가되었습니다.

- **새로운 아키텍처**

  이전의 DTS 패키지는 데이터의 변환이나 각 작업들이 단일 작업 영역에서 수행되었기 때문에 복잡한 변환이나 작업 단계를 구현하는데 어려움이 있었습니다. SQL 2005 SSIS에서는 제어 흐름 엔진과 데이터 흐름 엔진이라는 별도로 분리된 엔진들을 이용하여 패키지를 수행합니다. 제어 흐름은 파일 복사나 FTP 작업, SQL 명령 실행 작업

등과 같은 데이터 처리를 위한 준비 작업이나 메일 보내기 작업, 메시지 큐 작업 등과 같은 데이터 처리 후의 작업이 수행되는 일련의 프로세스이며, 데이터 흐름은 데이터를 원본으로부터 읽어온 후, 정렬, 집계, 형 변환, 연산 등과 같은 변환을 거쳐 대상으로 저장하는 프로세스입니다. 이와 같이 성격에 따른 작업 영역의 분리로 인해 패키지를 보다 더 정확히 제어할 수 있게 되었으며 명확한 데이터 처리 프로세스를 작성할 수 있게 되었습니다.

- **유연한 확장성**

  SQL 2005 SSIS는 Microsoft .NET Framework을 기반으로 수행되기 때문에 다양한 데이터 공급자나 작업 기능들을 사용할 수 있으며 다양한 사용자 API(Application Programming Interface)를 제공하기 때문에 쉽게 사용자가 원하는 작업들을 개발할 수 있게 되었습니다. 필요한 경우, 기본 작업 개체들을 이용하여 복잡한 프로세스를 구현하는 대신 VB.NET이나 C#을 이용하여 필요한 작업 개체를 만들어 이용하는 과정이 매우 용이해졌습니다.

- **패키지의 개발 환경과 관리의 분리**

  SQL 2000 DTS에서는 패키지의 개발 및 관리가 모두 엔터프라이즈 관리자에서 이루어졌지만 SQL 2005 SSIS에서는 패키지의 개발과 관리 환경이 완전히 분리되었습니다. 패키지의 개발은 전문 개발 환경인 Visual Studio에서 이루어지며, 패키지의 실행 및 모니터링과 같은 관리 작업은 SQL Server Management Studio에서 담당하게 됩니다. Visual Studio 중 Integration Services의 패키지 및 Analysis Services의 큐브와 디멘젼, Reporting Services의 리포트를 개발할 수 있는 환경을 Business Intelligence Development Studio(BIDS)라 하며, SQL Server 2005에 포함되어 있습니다. Visual Studio가 설치되어 있지 않은 환경이라도 SQL Server 2005만 설치하면 SQL Server와 관련된 BI 개발 작업을 수행할 수 있습니다.

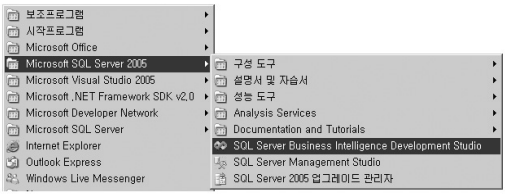
  BIDS에는 스크립트를 작성할 수 있는 Visual Studio for Application(VSA)이 포함되어 있으며 중단점이나 배포, 프로젝트 관리 기능 등과 같은 다양한 개발 기능을 지원하기 때문에 이전 버전에 비해 보다 쉽고 전문적으로 패키지를 작성할 수 있게 되었습니다.

- **모니터링 및 오류 처리 도구**

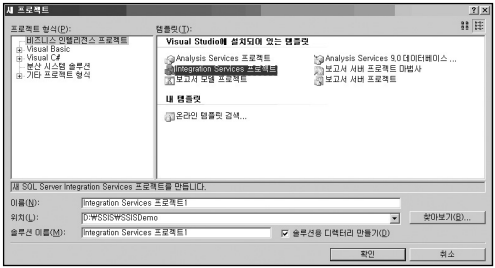
  Integration Services에는 로깅 기능 및 이벤트 처리기 등이 포함되어 있습니다. 이러한 기능들을 이용하여 SSIS 패키지 수행 시 발생하는 여러 상황에 대한 모니터링이나 오류 처리를 쉽게 수행할 수 있습니다. 뿐만 아니라 시스템의 성능 카운터에는 현재 수행되고 있는 데이터 흐름 엔진의 수, 프로세스의 수, 버퍼 사용량 등과 같은 성능 정보를 모니터링 할 수 있는 여러 카운터들이 추가되어 있기 때문에 패키지의 성능을 모니터링하고 관리하는 작업이 용이합니다. 또한 패키지를 개발하는 단계에서 중단점이나 검사점 등과 같은 기능을 이용하면 쉽게 패키지를 디버깅할 수 있습니다.

SSIS 패키지 개발 환경

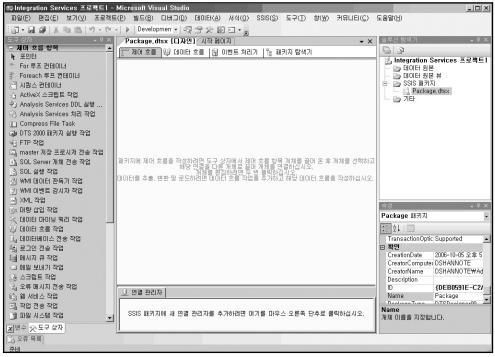
SSIS 패키지를 개발하기 위해서는 아래와 같이 SQL Server Business Intelligence Development Studio를 이용해야 합니다.



BIDS를 실행하면 프로젝트 선택 화면이 나타나며, 오른쪽의 템플릿 창에서 Integration Services 프로젝트를 선택한 후 확인을 누르면 SSIS 프로젝트가 생성됩니다. 하나의 프로젝트에는 여러 개의 SSIS 패키지가 존재할 수 있습니다. 또한 SSIS 패키지뿐만 아니라, 데이터 원본이나 데이터 원본 뷰 등과 같은 구성 요소들도 포함되어 다른 형태의 솔루션과 연동할 수 있는 기능도 제공합니다.



SSIS 패키지의 개발 환경은 다음 그림과 같이 구성되어 있습니다. SSIS 패키지를 개발하는 단계에서는 SQL Server에 연결되어 있지 않아도 됩니다. 이전 버전의 DTS 패키지는 SQL Server 2000이 실행 중인 서버에 연결한 후 엔터프라이즈 관리자의 데이터 변환 서비스에서 개발하였지만 SQL 2005 SSIS 패키지는 SQL Server의 실행 여부와는 상관없이 패키지를 개발할 수 있습니다.



제어 흐름 요소

컨테이너

컨테이너는 반복적인 작업을 수행하도록 구현하거나 서로 관련 있는 작업들을 그룹화해서 관리할 때 이용하는 작업 개체 입니다.
다음과 같은 경우에 컨테이너를 이용할 수 있습니다.

- 동일한 저장 프로시저를 100회 반복해서 수행하고자 할 경우 → **For 루프 컨테이너 이용**
- D:₩TextFile₩ 폴더에 있는 텍스트 데이터 파일들을 DB에 입력해야 할 경우 → **Foreach 루프 컨테이너 이용**
- 수행되어야 할 배치 쿼리가 많은 경우 → **Foreach 루프 컨테이너 이용**
- 데이터 처리 과정에서 일부 작업에 대해서 트랜잭션 관리가 필요한 경우 → **시퀀스 컨테이너 이용**

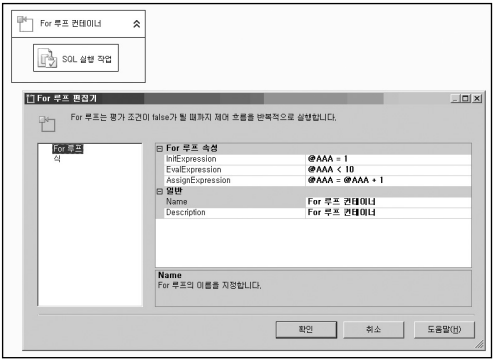
SQL 2005 SSIS에는 총 4개의 컨테이너가 있습니다.

For 루프 컨테이너

For 루프 컨테이너는 동일한 작업을 반복적으로 수행해야 하는 경우 사용하는 작업 개체 입니다.
For 루프 컨테이너는 Visual Basic 등과 같은 프로그래밍에서의 For-Next 구문과 유사합니다.

for I = 1 to 10 step 1
… 수행할 명령들 …
next

SSIS 패키지 내에서 사용자 변수를 사용하여 정해진 횟수만큼 반복적으로 작업을 수행하도록 지정할 수 있습니다.



- InitExpression - 변수의 초기값을 지정합니다.
- EvalExpression - 루핑을 계속 수행할지 평가하는 계산식이며 이 조건이 참일 때까지 반복해서 수행됩니다.
- AssignExpression - 값을 증가 또는 감소시키는 작업을 반복적으로 실행하는 식입니다.

Foreach 루프 컨테이너

Foreach 루프 컨테이너는 **For 루프 컨테이너**와 유사합니다. 하지만 **For 루프 컨테이너**는 「몇 번 반복 수행」 이라는 한정 조건을 사용하는 반면, **Foreach 루프 컨테이너**는 **열거자**(Enumerator)라는 반복적인 작업을 지정해주는 개체를 이용하여 「조건을 충족시키는 동안 반복해서 수행」하게 됩니다. **열거자**는 반복 작업 대상 집합체로서, SSIS의 변수 또는 특정 폴더 내의 파일들, 데이터베이스 개체 등과 같은 여러 유형이 있습니다.

SSIS에서의 열거자 유형은 다음과 같습니다.

- **File 열거자**
  특정 폴더 내에 있는 조건에 맞는 모든 파일을 대상으로 원하는 작업을 수행할 수 있습니다. 예를 들어 C:₩SQLScript₩ 라는 폴더 내의 여러 개의 *.sql 파일을 실행해야 할 경우, **File 열거자**에서 이 폴더를 지정한 후, **SQL 명령 실행** 작업을 이용하여 처리할 수 있습니다. 또한 여러 개의 텍스트 파일에 대한 입력 작업이나 처리 작업 등에서도 유용하게 활용할 수 있습니다.

- **Item 열거자**
  열의 유형을 설정하고, 직접 입력한 데이터를 이용하는 열거자 유형입니다.
  아래에 있는 **열(C)** 버튼을 클릭하여 입력할 데이터의 열을 미리 정한 후, 직접 데이터를 입력합니다.

- **ADO 열거자**
  Object형의 변수에 들어있는 데이터를 이용하는 열거자 유형입니다. 예를 들어 ResultSet 이라는 Object 형 변수에

```
SELECT AddressID FROM AdventureWorks.Person.Address
WHERE AddressID <= 10
```

  라는 SQL 쿼리의 결과 집합이 저장되어 있을 때, 이 값들을 차례로 읽어와서 작업하도록 설정할 수 있습니다.

- **ADO.NET 스키마 행 집합 열거자**
  .NET 공급자로 만들어진 연결에 대한 스키마 정보를 이용하는 열거자 유형입니다. 예를 들어 .NET 공급자를 이용하여 특정 DB를 지정해 놓은 경우, 해당 DB에 존재하는 모든 테이블들의 이름을 출력하는 기능을 구현할 수 있습니다. 이 때 사용되는 연결은

연결 관리자에서 **새 ADO.NET 연결**을 선택한 후, **OleDb용 .NET 공급자**를 선택해야 합니다.

- **From Variable 열거자**

  변수를 사용하는 열거자 유형입니다. 예를 들어 **InputStr** 이라는 String 형 변수에 「abcdefg」라는 값이 저장되어 있을 때, a, b, c, d, e, f, g로 값을 나누어서 작업을 수행해야 할 경우에 이용할 수 있습니다. 이 때 **변수 매핑** 탭에서 결과로 저장되는 변수의 유형은 반드시 **Object 형 변수**이어야 합니다.

- **Nodelist 열거자**

  XML 형태의 데이터 집합을 이용할 때 사용되는 열거자 유형입니다. XPath(XML Path Language)를 이용하여 전체 XML 데이터 집합에서 해당 조건에 맞는 데이터를 읽어오도록 지정합니다. 예를 들어 **[/authors/author[@period=' classical']**과 같은 형태의 **XPathString**을 지정한 경우에는 전체 XML 데이터 집합 중 classical 기간의 저자 목록만을 읽어오게 됩니다.

- **SMO 열거자**

  SMO는 SQL Server Management Object를 말하며 SQL Server 2000의 DMO와 같은 역할을 수행하는 데이터베이스 관리 개체입니다. SQL Server 2005의 스키마나 테이블 목록, 뷰 목록 등과 같이 데이터베이스의 여러 형태의 정보를 읽어올 수 있는 기능을 제공합니다. 예를 들어, 특정 데이터베이스의 모든 테이블에 대해 작업을 수행해야 하는 경우, 이 열거자를 이용하여 구현할 수 있습니다. SMO는 SQL Server 2005 뿐만 아니라 이전 버전의 SQL Server을 대상으로도 사용할 수 있습니다.

[따라하기] Foreach 루프 컨테이너

Foreach 루프 컨테이너를 이용하여 C:₩Windows₩ 폴더 아래에 있는 모든 bmp 파일을 C:₩로 복사하는 작업을 구현합니다.

1. 빈 패키지 파일을 하나 추가합니다.
2. 도구 상자에서 **Foreach 루프 컨테이너**를 선택한 후 **제어 흐름** 영역에 추가하고, 변수 창에서 ImagePath라는 String 형 변수를 추가합니다. 이 때, 이 변수의 값은 임의로 지정합니다. (예 : C:₩Windows₩aaa.bmp)

3. **Foreach 루프 컨테이너**를 더블 클릭하여 **Foreach 루프 편집기**를 연 후, **컬렉션** 탭에서 다음과 같이 지정합니다.

   Enumerator - Foreach File 열거자
   폴더 - C:₩Windows
   파일 - *.bmp

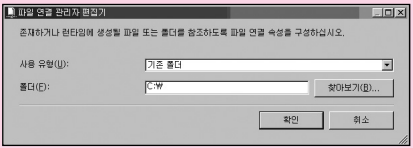
4. 현재 **Foreach 루프 컨테이너**가 수행하는 작업은 파일의 이름 정보만 읽어오는 것이며 직접 파일을 이용하여 어떠한 작업을 수행하거나 파일 내용을 조회하는 것은 아닙니다.

5. 읽어온 **파일 이름**을 앞 단계에서 정의한 ImagePath 변수에 대입하도록 지정합니다. **변수 매핑** 탭에서 **변수** 열의 아래 부분을 클릭하면 변수를 선택할 수 있습니다. 만약 앞 단계에서 변수를 추가하지 않았다면 이 부분에서 직접 추가할 수도 있습니다.

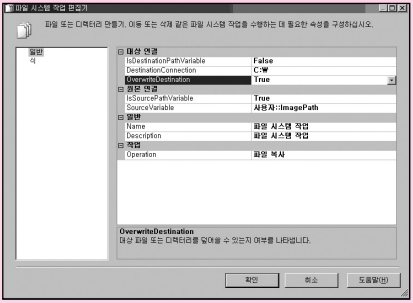
6. 도구 상자의 **파일 시스템 작업**을 선택한 후 **Foreach 루프 컨테이너** 안에 추가합니다.

7. **파일 시스템 작업**을 더블 클릭하여 나타나는 편집기에서 다음과 같이 설정합니다.

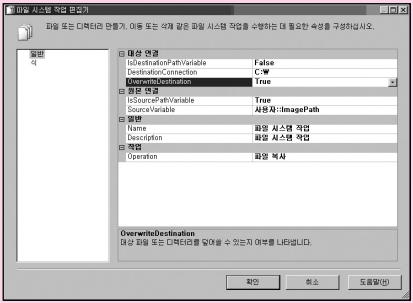
   a. DestinationConnection 부분에서 〈**새 연결...**〉을 선택하여 대상 작업에 대한 **파일 연결**을 추가합니다.



   b. IsSourcePathVariable을 True로 지정한 후, SourceConnection을 **사용자 ::ImagePath** 로 지정합니다.



c. Operation을 **파일 복사**로 지정한 후 **확인**을 눌러 편집기를 닫습니다.



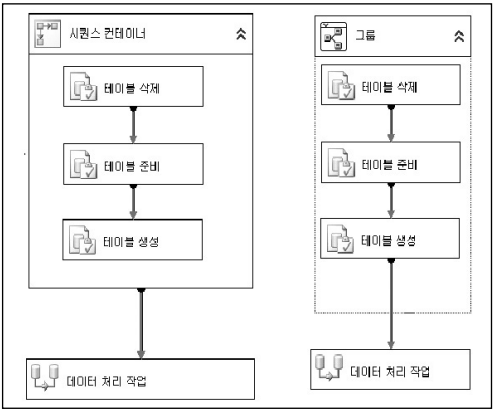
8. 패키지를 실행하면, C:₩Windows₩에 있는 bmp 파일들이 모두 C:₩로 복사됩니다.

시퀀스 컨테이너

시퀀스 컨테이너는 관련이 있는 여러 작업들을 하나로 묶어주는 그룹핑(Grouping)이나 여러 작업을 대상으로 트랜잭션을 관리하거나 일괄적으로 속성을 지정해야 할 경우에 이용하는 컨테이너입니다. 또한 변수를 설정할 때, 변수의 범위가 됩니다.

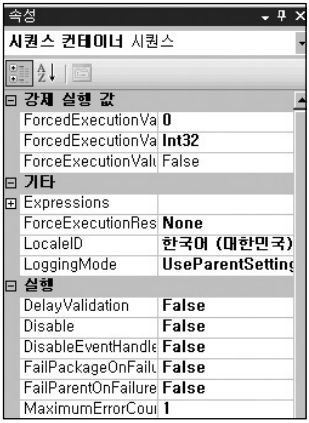
예를 들어, 패키지에서 저장 테이블 또는 원본 데이터를 준비하는 작업들이나 데이터를 변환하거나 연산을 수행하는 작업들, 처리된 데이터를 발송하거나 관리하는 작업들로 나뉠 경우 서로 관련이 있는 각 작업들을 그룹화하여 관리할 수 있습니다.
참고로, SSIS에서는 시퀀스 컨테이너와 유사한 **그룹** 기능이 있습니다. 컨트롤 키를 누른 상태에서 그룹으로 지정할 작업들을 선택한 후, 마우스 오른쪽 버튼을 클릭하여 나타나는 메뉴 중 **그룹(G)**을 선택하면 해당 작업들이 하나의 그룹으로 설정됩니다.

시퀀스 컨테이너와 **그룹**의 차이점은 다음과 같습니다.



작업을 그룹화하는 기능은 서로 비슷하지만 왼쪽의 **시퀀스 컨테이너**는 컨테이너 자체가 하나의 작업 개체로 간주됩니다. 즉, **시퀀스 컨테이너**에 포함된 작업 개체들의 수행 결과(성공 또는 실패)가 컨테이너의 최종 수행 결과로 반영되며, 외부의 다른 작업 개체들과 연결할 때에도 컨테이너와 직접 연결합니다. 하지만, **그룹**은 단순히 시각적으로 묶어주는 기능만 제공하며 패키지의 실행에는 아무런 영향을 주지 않습니다.

또한, **시퀀스 컨테이너**에는 다음과 같은 여러 속성들이 있지만 **그룹**에는 이러한 속성들이 없습니다.



작업 호스트

지금까지 설명한 세 가지 컨테이너는 도구 상자에 있는 개체들입니다. 이 외에도 SSIS에는 **작업 호스트**라는 컨테이너가 있습니다. **작업 호스트**는 별도로 존재하는 개체는 아니며, FTP **작업**이나 **스크립트 작업, 데이터 처리 작업** 등과 같이 **제어 흐름**의 작업 개체 자체를 말합니다.

SQL 실행 작업

SQL 실행 작업은 OLE DB 또는 ODBC, ADO, Excel 연결 등을 이용하여 SQL 쿼리를 실행할 수 있는 작업 개체입니다. 데이터베이스와 관련된 작업을 수행할 경우 가장 많이 사용되는 작업 개체입니다. 단순한 형태의 쿼리뿐만 아니라 입력 매개 변수가 포함된 쿼리도 사용할 수 있으며, 수행된 결과를 변수에 출력하도록 설정할 수도 있습니다.
SQL 실행 작업 편집기는 4개의 탭으로 구성되어 있습니다.

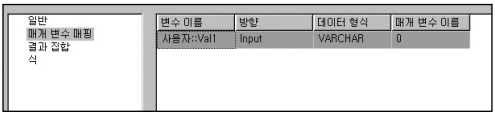
일반

일반 탭은 Connection Type이나, **옵션** 등을 설정하는 부분입니다.

- ConnectionType - EXCEL이나, OLE DB, ODBC, ADO 등 쿼리를 실행할 연결 유형을 지정합니다.
- Connection - ConnectionType에 해당하는 연결을 지정합니다.
- SQLSourceType
  - 직접 입력 - 실행할 쿼리를 직접 입력합니다.
  - 파일 연결 - 실행할 쿼리를 별도의 파일에 저장한 후 이 파일을 사용합니다. 파일 연결을 선택하면 아래의 SQLStatement 부분이 FileConnection으로 변경되며, 쿼리가 저장된 파일에 대한 **파일 연결**을 지정합니다.
  - 변수 - 실행 할 SQL 쿼리를 String형 변수에 저장한 후 이를 사용합니다.
- IsQueryStoreProcedure - 실행할 쿼리가 저장 프로시저인지를 설정합니다.
- BypassPrepare - 쿼리를 실행하기 전에 쿼리에서 사용되는 열 정보에 대한 조사 작업을 건너뛸지를 설정합니다. 만약 **OLE DB 연결**을 이용하며, 입력 매개 변수가 포함된 쿼리를 실행할 경우에는 이 값을 True로 설정해야 합니다.
- ResultSet - 쿼리 결과 유형을 지정합니다. 결과 출력 없이 단순히 실행되는 쿼리인 경우에는 **없음**으로 지정합니다. 출력 결과가 존재할 경우에는 **결과 집합** 탭에서 결과를 저장할 변수를 지정해야 합니다.
- TimeOut - 쿼리가 실행할 수 있는 최대 시간을 설정합니다. 0으로 설정한 경우, 시간 제한 없이 계속 수행할 수 있습니다.
- CodePage - 실행될 쿼리의 코드 페이지 입니다.

이 외에 쿼리 작성기 등과 같은 기능들이 추가로 포함되어 있습니다.

매개 변수 매핑



매개 변수 매핑 탭은 앞의 **일반** 탭에서 실행될 쿼리에 매개 변수가 포함되어 있는 경우, 이를 지정해 주는 부분입니다.

다음과 같은 쿼리를 살펴 봅시다.

```
SELECT * FROM TestTable  WHERE Vals = ?
```

SSIS의 변수를 WHERE 절의 매개 변수에 대입하여 실행하는 쿼리입니다. ? 부분이 매개 변수 값이 입력되는 부분이며 매개 **변수 매핑** 탭에서 이를 지정하게 됩니다.

아래 부분에 있는 **추가(A)** 버튼을 누르면 자동으로 매핑할 항목이 추가됩니다.

- **변수 이름** - 대입할 변수를 지정합니다. 미리 정해놓은 변수가 없는 경우 〈**새 변수...**〉를 이용해서 변수를 추가하면 됩니다.
- **방향** - Input, Output, ReturnValue를 지정합니다. 저장 프로시저인 경우, 출력되는 값이 있으면 Output 변수로 지정하면 됩니다.
- **데이터 형식** - 매개 변수의 데이터 형식을 지정합니다.
- **매개 변수 이름** - 기본 값으로 NewParameterName이 입력되어 있지만 사용되는 연결 방식에 맞게 변경해야 합니다.

연결 방식	입력 쿼리 (저장 프로시저가 아닌 경우)	매개 변수 이름
EXCEL, OLE DB	SELECT * FROM TestTable WHERE Vals1 = ? AND Vals2 = ?	0, 1, 2, …
ODBC	SELECT * FROM TestTable WHERE Vals1 = ? AND Vals2 = ?	1, 2, 3, …
ADO	SELECT * FROM TestTable WHERE Vals1 = ? AND Vals2 = ?	임의의 이름 지정 가능 순서대로 매핑됨
ADO.NET	SELECT * FROM TestTable WHERE Vals1 = @param1 AND Vals2 = @param2	쿼리에서 사용한 매개변수 이름 (예 : @param1, @param2)

연결 방식	입력 쿼리 (저장 프로시저인 경우)	매개 변수 이름
EXCEL, OLE DB	EXEC uspTestProc ?, ?	0, 1, 2, …
ODBC	EXEC uspTestProc ?, ?	1, 2, 3, …
ADO	IsStoredProcedure = false인 경우, EXEC uspTestProc ?, ? IsStoredProcedure = true인 경우, uspTestProc	임의의 이름 지정 가능 순서대로 매핑됨
ADO.NET	IsStoredProcedure = false인 경우, EXEC uspTestProc @param1, @param2 IsStoredProcedure = true인 경우, uspTestProc	IsStoredProcedure = false인 경우, 쿼리에서 사용한 매개변수 이름 IsStoredProcedure = true인 경우, 프로시저에서 정의된 매개변수 이름

결과 집합

결과 집합 탭은 쿼리가 실행된 후 출력되는 결과가 있을 경우 결과값을 저장할 변수를 지정하는 부분입니다.

일반 탭에서 ResultSet 속성을 **없음**으로 지정한 경우, **결과 집합** 탭의 버튼들은 비활성화되며, **단일 행** 또는 **전체 결과 집합**, XML로 지정된 경우에만 활성화됩니다.
ResultSet 속성을 **단일 행**으로 지정한 경우, 결과 이름에는 출력되는 결과의 열 이름을 지정하고 변수 이름에는 값을 저장할 변수를 지정합니다.
ResultSet 속성을 **전체 결과 집합**이나 XML로 지정한 경우, 반드시 결과 이름은 0으로 지정해야 합니다. 결과를 저장할 변수는 **전체 결과 집합**인 경우에는 Object 형, **XML형**인 경우에는 Object 형 또는 String형 이어야 합니다.

파일 관리 작업

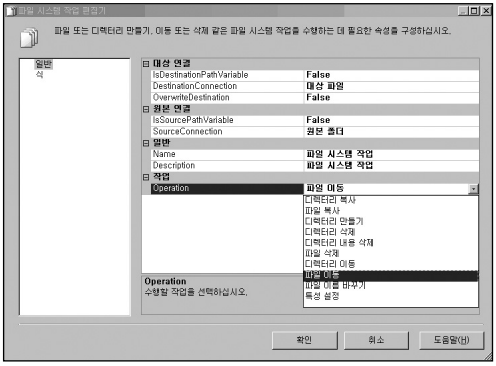
SSIS에서는 원본 데이터 파일의 복사 또는 삭제, 디렉터리 관리 작업 등과 같은 파일 시스템 작업이나 FTP 송수신 작업 등과 같은 FTP 작업을 수행하기 위한 작업 개체를 제공합니다.

파일 시스템 작업

데이터를 처리하는 과정에서 원본 파일의 복사, 디렉터리 생성, 기존 파일 삭제, 파일의 속성 변경 등 파일과 관련된 작업들이 종종 필요합니다.

SQL 2000 DTS 에서는 이러한 작업들을 수행하기 위해서는 **ActiveX 스크립트 작업**에서 FSO (File Script Object)를 이용하여 스크립트를 이용하거나 **SQL 실행 작업**에서 xp_cmdshell 명령을 이용하였습니다.

파일 시스템 작업은 파일과 관련된 다양한 작업을 수행할 수 있는 작업 개체입니다.



1. 디렉터리 복사

SourceConnection으로 지정된 원본의 디렉터리를 DestinationConnection으로 지정된 위치로 복사합니다.

OverwriteDestination 속성을 이용하여 이미 대상 위치에 디렉터리가 있을 경우 덮어쓸지 설정할 수 있습니다.

연결 관리자의 사용 유형	SourceConnection : **기존 폴더** DestinationConnection : **기존 폴더**

2. 파일 복사

디렉터리 복사와 동일한 형태이며, 디렉터리 대신 파일을 복사하는 작업입니다.

연결 관리자의 사용 유형	SourceConnection : **기존 파일** DestinationConnection : **기존 파일**

3. 디렉터리 만들기

SourceConnection으로 지정된 위치에 디렉터리를 생성합니다.

UseDirectoryIfExists 속성을 이용하여 이미 디렉터리가 있을 경우 기존 디렉터리를 이용할지 또는 작업이 실패하도록 할지를 설정할 수 있습니다.

연결 관리자의 사용 유형	SourceConnection : **폴더 만들기**

4. 디렉터리 삭제

SourceConnection으로 지정된 위치에 있는 디렉터리를 삭제합니다. 만약 삭제할 디렉터리가 없는 않는 경우에는 작업이 실패하게 됩니다.

연결 관리자의 사용 유형	SourceConnection : **기존 폴더**

5. 디렉터리 내용 삭제

SourceConnection으로 지정된 디렉터리에 존재하는 파일 및 디렉터리를 삭제합니다. 만약 삭제할 디렉터리가 없는 경우에는 작업이 실패하게 됩니다.

연결 관리자의 사용 유형	SourceConnection : **기존 폴더**

6. 파일 삭제

SourceConnection에 지정된 파일을 삭제합니다. 만약 삭제할 파일이 없는 경우에도 이 작업은 실패로 처리되지 않습니다.

연결 관리자의 사용 유형	SourceConnection : **기존 파일**

7. 디렉터리 이동

SourceConnection으로 지정된 원본의 디렉터리를 DestinationConnection으로 지정된 위치로 이동합니다.

연결 관리자의 사용 유형	SourceConnection : **기존 폴더** DestinationConnection : **기존 폴더**

8. 파일 이동

디렉터리 이동과 동일한 형태이며, 디렉터리 대신 파일을 이동시키는 작업입니다.

연결 관리자의 사용 유형	SourceConnection : **기존 파일** DestinationConnection : **기존 폴더**

9. 파일 이름 바꾸기

파일 이동과 유사한 작업이지만, DestinationConnection을 기존 폴더 대신 기존 파일로 지정해야 합니다.

연결 관리자의 사용 유형	SourceConnection : **기존 파일** DestinationConnection : **기존 파일**

10. 특성 설정

SourceConnection으로 지정된 파일의 Hidden, ReadOnly, Archive, System 속성을 설정할 수 있습니다.

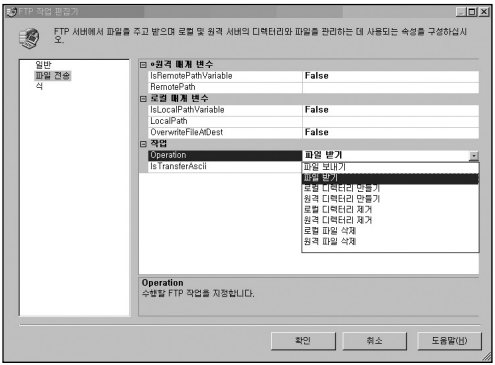
연결 관리자의 사용 유형	SourceConnection : **기존 파일**

각 작업의 공통된 특성으로 IsSourcePathVariable 및 IsDestinationPathVariable이 있습니다. 만약, 원본 파일 또는 디렉터리나 대상 파일 또는 디렉터리의 경로 정보를 변수에 저장시킨 후, 이를 이용하고자 할 경우 이 속성값을 True로 변경하고 SourceVariable 또는 DestinationVariable에 변수를 지정하면 됩니다. 이러한 방법을 사용할 경우에는 별도로 파일 연결 관리자를 지정할 필요가 없습니다.

FTP 작업

파일을 가져오거나 보낼 환경이 **파일 시스템 작업**을 이용할 수 없는 경우에는 **FTP 작업**을 이용할 수 있습니다.

SQL 2005 SSIS의 **FTP 작업**은 파일을 보내거나 받는 작업 외에도 디렉터리를 생성하거나 삭제하는 등 여러 유형의 작업을 구현할 수 있습니다.



FTP 작업을 수행하기 위해서는 **연결 관리자**에서 **FTP 연결**을 미리 생성해야 합니다. **FTP 연결**은 **연결 관리자**의 **새 연결(W)**을 선택한 후, **FTP 연결** 항목을 선택하면 됩니다.

- **파일 보내기**

  LocalPath에 지정된 연결에 해당되는 파일을 RemotePath에 지정된 FTP 연결에 해당되는 사이트로 파일을 보내는 작업입니다.

  OverwriteFileAtDest 속성을 이용하여 파일이 있는 경우 덮어 쓸지를 지정할 수 있으며, IsTransferAscii 속성을 이용하여 ASCII 모드로 전송할지를 지정할 수 있습니다.

- **파일 받기**

  원격지 사이트의 파일을 지정된 로컬 디렉터리로 가져오는 작업입니다.

- **로컬 디렉터리 만들기**
  로컬 서버에 디렉터리를 만드는 작업입니다.
  OverwriteFileAtDest 속성을 이용하여 이미 생성할 디렉터리가 있는 경우 덮어쓸지를 지정할 수 있습니다. 만약 해당 디렉터리가 있는 경우에 이 값을 False로 설정한 상태에서 작업을 수행하면 실패로 처리됩니다.

- **원격 디렉터리 만들기**
  원격 서버에 디렉터리를 만드는 작업입니다.
  로컬 디렉터리 만들기 작업의 OverwriteFileAtDest 속성과 동일한 기능을 합니다.

- **로컬 디렉터리 제거**
  LocalPath에 지정한 폴더를 삭제합니다. 삭제할 디렉터리가 없는 경우 작업은 실패하게 됩니다.

- **원격 디렉터리 제거**
  RemotePath에 지정한 폴더를 삭제합니다. 해당 디렉터리가 없는 경우 작업은 실패하게 됩니다.

- **로컬 파일 삭제**
  LocalPath에 지정된 파일을 삭제합니다. 이 작업은 **파일 시스템 작업**의 파일 삭제 작업과는 달리 삭제할 파일이 없는 경우 작업이 실패하게 됩니다.

- **원격 파일 삭제**
  RemotePath에 지정된 파일을 삭제합니다.

전송 작업

데이터베이스의 규모가 커지고 데이터베이스 시스템을 이용하는 영역이 넓어지면서 단순한 데이터의 전송이나 변환 작업뿐만 아니라 데이터베이스 서버의 이전, 통합, 업그레이드 및 마이그레이션 등과 같은 데이터베이스 개체나 데이터베이스 전체를 이전하는 작업들을 수행할 경우가 있습니다. 이럴 경우 SSIS에서 제공되는 여러 형태의 전송 작업을 이용하면 쉽고 편리하게 수행할 수 있습니다.

SSIS에는 총 6개의 전송 작업이 있습니다.

- **master 저장 프로시저 전송 작업** - 원본 서버의 master DB에 있는 사용자 저장 프로시저를 다른 서버의 master DB로 전송하는 작업입니다.
- **SQL Server 개체 전송 작업** - 원본 서버에서 데이터베이스 개체를 대상 서버로 전송하는 작업입니다. 데이터베이스 개체에는 테이블, 저장 프로시저, 뷰, 사용자 정의 함수 등 모든 개체가 포함됩니다.
- **데이터베이스 전송 작업** - 원본 서버의 데이터베이스 전체를 대상 서버로 전송하는 작업입니다.
- **로그인 전송 작업** - 원본 서버의 로그인을 대상 서버로 전송하는 작업입니다. 전송된 로그인은 기본적으로 사용 안 함 상태이며 원본 암호와는 다른 임의의 암호가 할당됩니다.
- **작업 전송 작업** - 원본 서버의 SQL Server 에이전트에 있는 작업(Job)을 대상 서버로 전송하는 작업입니다.
- **오류 메시지 전송 작업** - 원본 서버에 있는 사용자 정의 오류 메시지를 대상 서버로 전송하는 작업입니다.

전송 작업에서 이용되는 원본 및 대상 서버의 연결은 SMOSever 연결을 이용합니다.

SMO(SQL Management Objects) 연결은 서로 다른 인스턴스 간 데이터베이스 개체를 전송하는데 이용되는 연결이며 OLEDB 연결이나 ADO.NET 연결과는 달리 데이터베이스를 지정하지는 않습니다.

SMOServer는 연결 관리자에서 마우스 오른쪽 버튼을 클릭한 후 나오는 메뉴에서 **새 연결(W)**을 선택한 후 SMOServer 항목을 선택하여 추가하면 됩니다.

> ***[참고]***
> *정확한 표현은 **원본 서버**가 아닌 **원본 인스턴스**, **대상 서버**가 아닌 **대상 인스턴스**입니다. 동일 서버에 두 개 이상의 인스턴스가 존재할 수 있으며 이 인스턴스들 간의 전송 작업을 수행하는 경우도 있습니다. 하지만, 편의상 **원본 인스턴스** 대신 **원본 서버**, **대상 인스턴스** 대신 **대상 서버**로 사용하겠습니다.*

master 저장 프로시저 전송 작업

원본 서버의 master DB에 있는 사용자 저장 프로시저를 대상 서버로 복사하는 작업 개체입니다. 오직 master DB에 있는 사용자 저장 프로시저만 대상이 되며 해당 프로시저의 소유자가 dbo 인 프로시저만 옮길 수 있습니다.
이 작업을 수행하기 위해서는 원본 서버의 master DB에 있는 사용자 저장 프로시저를 볼 수 있는 권한이 있어야 하며, 또한 대상 서버의 sysadmin 서버 역할의 구성원이거나 master DB에 저장 프로시저를 만들 수 있는 권한이 있어야 합니다.

연결

- SourceConnection - 원본 서버에 대한 연결을 지정합니다.
- DestinationConnection - 대상 서버에 대한 연결을 지정합니다.

저장 프로시저

- IfObjectExists - 대상 서버의 master DB에 이미 동일한 저장 프로시저가 있는 경우 처리할 방법을 지정합니다. 실패하도록 하거나(FailTask) 덮어 쓰기(Overwrite), 건너뛰기(Skip)로 지정할 수 있습니다.
- TransferAllStoredProcedures - 저장 프로시저 모두를 전송할 것인지, 일부만 전송할 것인지를 지정합니다.
- StoredProceduresList - 저장 프로시저를 모두 전송하지 않는 경우, 전송할 저장 프로시저들을 지정합니다.

master 저장 프로시저 전송 작업은 master에 있는 사용자 저장 프로시저만을 전송하기 위한 특화된 작업이며 아래 부분에서 설명할 **SQL Server 개체 전송 작업**에서 master DB를 지정하고 저장 프로시저를 전송하는 방식과 동일한 기능을 수행합니다. **master 저장 프로시저**는 SQL 2000 DTS에서도 있었던 기능이지만 **SQL Server 개체 전송 작업**은 SQL 2005 SSIS에서 새로 추가된 기능입니다.

SQL Server 개체 전송 작업

SQL Server 개체 전송 작업은 다른 전송 작업들과는 달리 SQL 2005 SSIS에서 새롭게 추가된 전송 작업 개체입니다. 원본 서버에 있는 데이터베이스 개체를 대상 서버로 전송할 때 사용하는 작업 개체입니다. 데이터베이스 개체란, 테이블이나 뷰, 저장 프로시저, 사용자 정의 함수, 기본값 등 데이터베이스에 포함된 모든 개체를 말합니다.

SQL Server 2005에는 파티션 함수나 스키마, 어셈블리 등과 같이 새로운 데이터베이스 개체가 추가되어 있습니다. 따라서 원본 서버 혹은 대상 서버의 버전에 따라 전송할 수 있는 개체들이 다를 수 있습니다. 각 버전 별 전송 가능한 개체는 아래 표를 참고하시기 바랍니다.

개체	버전
테이블	SQL Server 2000 또는 SQL Server 2005
뷰	SQL Server 2000 또는 SQL Server 2005
저장 프로시저	SQL Server 2000 또는 SQL Server 2005
사용자 정의 함수	SQL Server 2000 또는 SQL Server 2005
기본값	SQL Server 2000 또는 SQL Server 2005
사용자 정의 데이터 형식	SQL Server 2000 또는 SQL Server 2005
파티션 함수	SQL Server 2005만 해당
파티션 구성표	SQL Server 2005만 해당
스키마	SQL Server 2005만 해당
어셈블리	SQL Server 2005만 해당
사용자 정의 집계	SQL Server 2005만 해당
사용자 정의 유형	SQL Server 2005만 해당
XML 스키마 컬렉션	SQL Server 2005만 해당

대상

- DropObjectsFirst - 개체를 복사하기 전에 대상 서버에 있는 개체를 삭제할지를 설정합니다.
- IncludeExtendedProperties - 개체를 전송할 때 확장 속성을 포함할지를 설정합니다.
- CopyData - 개체를 전송할 때 데이터도 함께 전송할지를 설정합니다.
- ExistingData - 데이터를 전송하는 방법을 설정합니다. Replace로 지정하면 대상 테이블의 데이터를 덮어 쓰며 Append로 지정하면 기존 데이터에 추가됩니다.

- CopySchema - 개체를 전송할 때 스키마도 포함시킬지를 설정합니다.
- UseCollation - 개체를 전송할 때 원본에서 지정한 정렬 방식을 그대로 유지할지를 설정합니다.
- IncludeDependentObjects - 전송할 개체에 종속된 다른 개체들도 함께 전송할지를 설정합니다.

대상 복사 옵션

- CopyAllObjects - 지정한 원본 서버의 데이터베이스에 있는 모든 개체를 전송할지를 설정합니다.
- ObjectsToCopy - CopyAllObjects가 false인 경우 전송할 개체를 선택합니다.

보안

- CopyDatabaseUsers - 데이터베이스 사용자 정보를 전송할지를 설정합니다.
- CopyDatabaseRoles - 데이터베이스 역할 정보를 전송할지를 설정합니다.
- CopySqlServerLogins - 데이터베이스의 로그인 정보를 전송할지를 설정합니다.
- CopyObjectLevelPermissions - 개체 수준 사용자 권한 정보를 전송할지를 설정합니다.

연결

- SourceConnection - 원본 서버에 대한 연결을 지정합니다.
- SourceDatabase - 전송할 개체가 포함된 원본 데이터베이스를 지정합니다.
- DestinationConnection - 대상 서버에 대한 연결을 지정합니다.
- DestinationDatabase - 대상 데이터베이스를 지정합니다.

테이블 옵션

- CopyIndexes - 인덱스를 전송에 포함시킬지를 설정합니다.
- CopyTriggers - 트리거를 전송에 포함시킬지를 설정합니다.
- CopyFullTextIndexes - 전체 텍스트 인덱스를 전송에 포함시킬지를 설정합니다.
- CopyPrimaryKeys - 기본 키를 전송에 포함시킬지를 설정합니다.

- CopyForeignKeys - 외래 키를 전송에 포함시킬지를 설정합니다.
- GenerateScriptsInUnicode - 유니코드 형식으로 개체 스크립트를 생성할지를 설정합니다.

데이터베이스 전송 작업

데이터베이스 전송 작업은 SQL Server **개체 전송 작업**과는 달리 데이터베이스 전체를 대상으로 복사 또는 이전하는 작업입니다. 이 작업은 동일 인스턴스 내에서 데이터베이스를 복사하는 경우에도 이용할 수 있으며, SQL Server 2000 에서 2005로의 전송도 수행 가능합니다.

데이터베이스 전송 작업은 온라인 또는 오프라인으로 수행할 수 있습니다. 온라인으로 설정하면 원본 데이터베이스가 온라인 상태에서 전송되며 SMO(SQL Management Object)를 이용하여 데이터베이스 개체 및 데이터가 전송됩니다. 오프라인으로 설정하면 데이터베이스를 분리(Detach) 한 후 데이터 파일 및 로그 파일을 대상 위치로 복사하고 다시 연결(Attach)하는 방식으로 전송 작업을 수행합니다.

오프라인으로 수행하기 위해서는 원본 서버 및 대상 서버에 네트워크 파일 공유를 지정해야 합니다.

대상 데이터베이스

- DestinationDatabaseName - 대상 서버에 전송될 데이터베이스의 이름을 지정합니다.
- DestinationDatabaseFiles - 대상 서버에 전송될 데이터베이스의 파일을 지정합니다.
- DestinationOverwrite - 대상 서버에 이미 동일한 데이터베이스가 있는 경우 덮어쓸지를 지정합니다.

연결

- SourceConnection - 원본 서버에 대한 연결을 지정합니다.
- DestinationConnection - 대상 서버에 대한 연결을 지정합니다.

원본 데이터베이스

- Action - 원본 서버의 데이터베이스를 대상 서버로 복사(Copy) 또는 이동(Move)할지를 설정합니다.
- Method - 데이터베이스를 온라인 또는 오프라인으로 전송할지를 설정합니다. 온라인 상태로 전송하기 위해서는 패키지를 실행시키는 사용자가 sysadmin 고정 서버 역할의 멤버이거나 전송할 데이터베이스의 소유자(dbo)이어야 합니다. 오프라인 상태로 전송하기 위해서는 sysadmin 고정 서버 역할의 멤버이면 됩니다.
- SourceDatabaseName - 원본 서버에서 전송할 데이터베이스의 이름을 지정합니다.
- SourceDatabaseFiles - 원본 서버에서 전송할 데이터베이스의 파일을 지정합니다.
- ReattachSourceDatabase - 전송 작업 수행 중 오류가 발생하였을 때 원본 데이터베이스를 자동으로 연결(Attach)시킬지를 설정합니다.

로그인 전송 작업

로그인 전송 작업은 원본 서버의 로그인 정보를 대상 서버로 전송하는 작업 개체입니다. 전송 작업 수행 시 모든 로그인을 전송하거나 특정 로그인 만을 전송하도록 지정할 수 있습니다. 또한 로그인에 연결된 sid(보안 ID)를 같이 복사할 수도 있습니다. 이러한 기능은 원본 데이터베이스를 다른 서버에서 복원하여 사용할 경우, 로그인 정보를 복사하고 로그인의 sid를 원본 서버와 동일하게 변경해야 하는 작업을 대체할 수 있습니다.

SQL 2005 SSIS의 **로그인 전송 작업**은 SQL 2000 DTS의 작업에 비해 몇 가지 제약 사항이 있습니다. **로그인 전송 작업**을 이용하여 전송된 로그인은 기본적으로 **비활성화**되며, **임의의 암호가 설정**됩니다. 로그인을 전송한 후 이를 사용하기 위해서는 Management Studio나 쿼리 분석기를 이용하여 해당 로그인을 활성화하고 암호를 재설정해야 합니다.

로그인

- LoginsToTransfer - 원본 서버에서 대상 서버로 전송할 작업 유형을 지정합니다.
  - AllLogins - 원본 서버의 모든 로그인 정보를 대상 서버로 전송합니다.

- SelectedLogins - 원본 서버에서 선택한 로그인 정보만 대상 서버로 전송합니다.
- AllLoginsFromSelectedDatabases - DatabaseList에 지정된 데이터베이스의 모든 로그인 정보를 대상 서버로 전송합니다.

- LoginsList - SelectedLogins로 선택한 경우 전송할 로그인을 선택합니다.
- DatabaseList - AllLoginsFromSelectedDatabases로 선택한 경우, 전송할 로그인의 데이터베이스를 선택합니다.

연결

- SourceConnection - 원본 서버에 대한 연결을 지정합니다.
- DestinationConnection - 대상 서버에 대한 연결을 지정합니다.

옵션

- IfObjectExists - 대상 서버에 전송할 로그인이 있는 경우, 처리 방법을 설정합니다.
  - FailTask - 동일한 로그인이 있는 경우, 로그인 전송 작업이 실패합니다.
  - Overwrite - 대상 서버의 로그인을 덮어씁니다.
  - Skip - 대상 서버의 로그인을 덮어 쓰지 않고 그냥 건너뜁니다.
- CopySids - 로그인에 연결된 sid(보안 ID)를 함께 전송할지를 설정합니다.

작업 전송 작업

원본 서버의 SQL Server 에이전트에 등록된 예약 작업(Job)을 대상 서버로 전송하는 작업 개체입니다. **개체 전송 작업**이나 **로그인 전송 작업**과 마찬가지로 전체 작업을 전송하도록 하거나 특정 작업만 전송하도록 설정할 수 있으며, 대상 서버에 동일한 이름의 작업이 있는 경우 덮어쓸지를 설정할 수 있습니다.

연결

- SourceConnection - 원본 서버에 대한 연결을 지정합니다.
- DestinationConnection - 대상 서버에 대한 연결을 지정합니다.

옵션

- IfObjectExists - 대상 서버에 전송할 작업이 있는 경우, 이에 대한 처리 방법을 설정합니다.
  - FailTask - 동일한 작업이 있는 경우, 작업 전송 작업이 실패합니다.
  - Overwrite - 대상 서버의 작업을 덮어씁니다.
  - Skip - 대상 서버의 작업을 덮어 쓰지 않고, 그냥 건너뜁니다.
- EnableJobsAtDestination - 대상 서버로 작업을 전송한 후, 해당 작업의 활성화 여부를 설정합니다.

작업

- TransferAllJobs - 모든 작업을 전송할지 설정합니다.
- JobsList - 모든 작업을 전송하지 않는 경우, 전송할 작업들을 선택합니다.

오류 메시지 전송 작업

원본 서버에 있는 사용자 정의 오류 메시지를 대상 서버로 전송하는 작업 개체입니다. 사용자 정의 오류 메시지는 message_id가 50000 이상인 메시지이며 SQL Server 2000에서는 master DB의 **sysmessages** 테이블에서, SQL Server 2005에서는 **sys.messages** 관리 뷰에서 확인하실 수 있습니다.

이 작업 역시 **로그인 전송 작업**이나 **작업 전송 작업**과 유사한 옵션들로 구성되어 있으며 특정 언어의 메시지만을 전송할 수 있는 옵션이 있습니다.

옵션

- IfObjectExists - 대상 서버에 전송할 오류 메시지가 있는 경우, 이에 대한 처리 방법을 설정합니다.
  - FailTask - 동일한 메시지가 있는 경우, 오류 메시지 전송 작업이 실패합니다.
  - Overwrite - 대상 서버의 메시지를 덮어씁니다.
  - Skip - 대상 서버의 메시지를 덮어쓰지 않고 건너뜁니다.

- TransferAllErrorMessages - 대상 서버로 모든 메시지를 전송할지 설정합니다
- ErrorMessageList - 모든 메시지를 전송하지 않는 경우, 전송할 메시지들을 선택합니다.
- ErrorMessageLanguagesList - 대상 서버로 전송할 다른 언어 버전의 메시지를 선택합니다. 다른 언어의 메시지를 전송하기 위해서는 대상 서버에 us_english(1033)인 메시지가 있어야 합니다.

연결

- SourceConnection - 원본 서버에 대한 연결을 지정합니다.
- DestinationConnection - 대상 서버에 대한 연결을 지정합니다.

WMI 작업

SQL 2005 SSIS에는 OS의 WMI 정보를 이용할 수 있는 작업 개체들이 있습니다.

데이터베이스 관리자의 여러 업무들 중에서 데이터베이스 및 SQL Server가 수행되는 서버의 리소스를 관리하는 작업은 중요한 업무 중 하나입니다. 관리자는 서버의 디스크 여유 공간이나 메모리 사용량, CPU 사용량 등 다양한 리소스 정보를 정기적으로 모니터링 하고 관리합니다.

또한 서버의 CPU의 사용량이나 메모리 사용량 등의 정보는 관리 도구에 있는 성능 모니터를 이용하여 수집하고 관리합니다.

하지만 관리해야 할 서버가 많거나 서버의 디스크 드라이브가 많을 경우, 모니터링 해야 할 항목들이 많을 경우, 또는 수집한 데이터를 테이블이나 별도의 파일 형태로 관리해야 할 경우에는 위와 같은 방법들을 이용하여 수행하는 것이 복잡하고 어려운 작업일 수 있습니다. 이와 같은 경우 WMI 개체를 이용하면 간단하고 효율적으로 업무를 수행할 수 있습니다.

WMI 작업은 이러한 시스템 관리 측면 외에도 파일 생성 또는 삭제, 응용 프로그램 설치 등과 같이 시스템에서 발생하는 여러 이벤트 정보를 읽어오는 등 다양한 영역에서 사용할 수 있습니다.

WMI란?

원래 1998년 Windows NT 4.0 서비스 팩 4의 추가 구성 요소로 릴리스된 WMI는 Windows 2000, Windows XP 및 Windows Server 2003 운영 체제 제품군에 구축된 핵심 관리 기술입니다. DMTF(Distributed Management Task Force)에 의해 발견된 업계 표준을 기반으로 한 WMI는 거의 모든 Windows 리소스를 액세스하고 구성하고 관리하고 모니터링할 수 있는 수단이자 통로입니다.

WMI의 기능을 이해하려면 이전에 Windows 워크스테이션과 서버를 관리하고 모니터링 했던 방법을 생각해 보십시오. 디스크, 이벤트 로그, 파일, 폴더, 파일 시스템, 네트워크 구성 요소, 운영 체제 설정, 성능 데이터, 프린터, 프로세스, 레지스트리 설정, 보안, 서비스, 공유, 사용자, 그룹 등과 같은 Windows 리소스를 관리하는 수많은 그래픽 관리 도구를 사용해 봤거나 현재 사용하고 있을 것입니다.

그래픽 관리 도구가 기능적인 관리 솔루션을 제공하긴 했지만 그들의 공통점은 무엇일까요? 한 가지 대답은 WMI 이전에는 모든 Windows 그래픽 관리 도구가 Windows 리소스를 액세스하고 관리하는 데 Win32 API(Application Programming Interface)에 의존했다는 것입니다. 그 이유는 무엇일까요? WMI 이전에는 Win32 API를 통해서만 프로그래밍 방식으로 Windows 리소스에 액세스할 수 있었기 때문입니다. 대부분의 스크립팅 언어에서 Win32 API를 직접 호출할 수 없기 때문에 널리 사용되고 있는 스크립팅 언어를 사용하여 일반 시스템 관리 작업을 자동화하는 쉬운 방법이 없는 이러한 상황이 Windows 시스템 관리자에게 남겨진 것입니다. WMI는 모든 Windows 리소스를 외부 세계에 설명하고 드러내어 일관된 모델과 프레임워크를 제공함으로써 이러한 문제를 변화시켰습니다. 그리고 무엇보다도 시스템 관리자는 WMI 스크립팅 라이브러리를 사용하여 WMI를 통해 게시된 Windows 리소스를 관리할 시스템 관리 스크립트를 만들 수 있습니다.

Windows 스크립트 호스트와 Microsoft Visual Basic Scripting Edition(VBScript) 또는 COM 자동화를 지원하는 모든 스크립트 언어(예: ActiveState Corporation의 ActivePerl)를 사용하여 다음과 같은 기업용 시스템, 응용 프로그램 및 네트워크를 관리하고 자동화하는 스크립트를 작성할 수 있습니다.

Windows Server 2003, Windows XP Professional 및 Windows 2000 시스템 관리 스크립트를 작성하여 성능 데이터를 검색하고 이벤트 로그, 파일 시스템, 프린터, 프로세스, 레지스트리 설정, 스케줄러, 보안, 서비스, 공유 및 여러 가지 기타 운영 체제 구성 요소와 구성 설정을 관리할 수 있습니다.
네트워크 관리 WMI 기반 스크립트를 만들어 DNS, DHCP 및 SNMP 사용 장치와 같은 네트워크 서비스를 관리할 수 있습니다.
실시간 상태 모니터링 WMI 이벤트 가입을 사용하여 발생할 때마다 이벤트 로그 항목, 파일 시스템과 레지스트리 수정 및 기타 실시간 운영 체제 변경 사항을 모니터링하고 응답할 스크립트를 작성할 수 있습니다. 개념적으로 WMI 이벤트 가입 및 알림이 WMI에 대해 갖는 의미는 SNMP 트랩이 SNMP 세계에 대해 갖는 의미와 같습니다.
Windows .NET Enterprise Server 관리 Microsoft Application Center, Operations Manager, Systems Management Server, Internet Information Server, Exchange Server 및 SQL Server를 관리하는 스크립트를 작성할 수 있습니다.

원본
http://www.microsoft.com/korea/msdn/columns/contents/scripting/scripting06112002/default.aspx

WMI 데이터 판독기 작업

WMI 정보는 기본적으로 OS에서 Windows Management Instrumentation 서비스를 통해 OS에서 관리됩니다.

WMI 정보를 사용하기 위해서는 WMI 정보를 읽어올 서버에 대한 연결과 WMI 정보를 읽어올 명령어가 필요합니다. WMI 연결은 SSIS의 **연결 관리자**에서 지정할 수 있으며, WMI 정보는 SQL 쿼리와 유사한 형태인 WQL(WMI Query Language)이라는 스크립트 언어를 이용하여 읽어올 수 있습니다.

WMI 데이터 판독기 작업을 수행하기 위해서는 **연결 관리자**에 **WMI 연결**이 추가되어 있어야 하며, **WMI 데이터 판독기 작업** 내의 WMIConnection 속성에서 〈**새 WMI 연결**..〉을 선택하여 직접 등록할 수도 있습니다.

연결 관리자에서 마우스 오른쪽 버튼을 클릭하여 나타나는 메뉴 중 **새 연결(W)**을 선택한 후, WMI 연결 항목을 선택하고 연결할 서버를 지정합니다.

로컬 서버 외에도 원격 서버에 연결할 수도 있습니다.

WMI 데이터 판독기 작업의 속성은 다음과 같습니다.

- WmiConnection - WMI 연결을 지정합니다.
- WqlQuerySourceType - 데이터를 판독하기 위해 수행하는 쿼리인 WQL 쿼리의 입력 형태를 지정합니다. 직접 입력하도록 설정하거나, 변수에 WQL 쿼리를 저장한 후 이 변수를 사용하도록 지정할 수 있으며, 별도의 파일로 만든 후 이 파일을 사용하도록 지정할 수도 있습니다.
- WqlQuerySource - 직접 입력으로 지정한 경우 WQL 쿼리를 입력합니다.
- OutputType - 결과 형태를 설정합니다. **데이터 테이블**인 경우, 결과가 테이블 형태로 출력됩니다. 이 외에도, **속성 이름 및 값** 또는 **속성 값**만 출력되도록 지정할 수 있습니다.

결과가 데이터 테이블인 경우, 대상은 파일 형태이거나 Object형 변수이어야 하며, 속성 이름 및 값 또는 값인 경우에는 파일 형태이거나 String형 또는 Object형 변수이어야 합니다.

- OverwriteDestination - 기존의 파일 또는 변수의 값이 있는 경우 이에 대한 처리 유형을 지정합니다. **대상 덮어쓰기, 대상에 추가, 기존 형태 유지**로 설정할 수 있습니다.
- DestinationType - 결과를 저장할 유형을 지정합니다. 파일 또는 변수에 저장할 수 있습니다.
- Destination - 결과를 파일에 저장하도록 설정한 경우에는 **파일 연결**을 지정하며, 변수로 저장하도록 설정한 경우에는 저장할 변수 이름를 지정합니다.

[따라하기] WMI 데이터 판독기 작업

본 예제에서는 서버의 CPU 사용률(%Processor Time)을 확인하여 사용률이 30% 미만인 경우, 현재 실행 중인 모든 프로세스의 이름과 경과된 시간, 스레드 수 등을 읽어서 텍스트 파일로 저장하는 작업을 구현합니다. 이 예제를 응용하면 서버의 CPU 사용률이 높을 때 실행중인 프로세스들의 정보를 남기는 모니터링 작업이나 특정 디스크의 여유 공간이 부족할 때 지정한 파일들을 삭제하는 시스템 관리 작업을 구현할 수 있습니다.

1. 빈 패키지 파일을 하나 추가합니다.

2. 도구 상자에서 **WMI 데이터 판독기 작업**을 선택하여 **제어 흐름** 영역에 추가한 후, 이름을 **[CPU 사용량 확인]**으로 변경합니다.

3. WMI 연결을 **연결 관리자**에서 따로 추가하는 대신 직접 **WMI 데이터 판독기 작업** 내에서 추가하도록 하겠습니다. 2에서 추가한 **WMI 데이터 판독기 작업**을 더블 클릭하여 속성 창을 엽니다. 속성 창에서 WMIConnection 항목의 오른쪽에 있는 ... 버튼을 클릭한 후, 〈**새 WMI 연결...**〉을 선택합니다.

4. WMI 연결에 대한 속성 창이 나타납니다. 본 예제에서는 Windows **인증**을 사용하도록 설정합니다. 로컬 서버가 아닌 원격 서버에 대해 WMI 정보를 읽어오는 작업을 만드는 경우, 서버 이름과 사용자 자격 증명 정보를 변경하여 사용합니다.

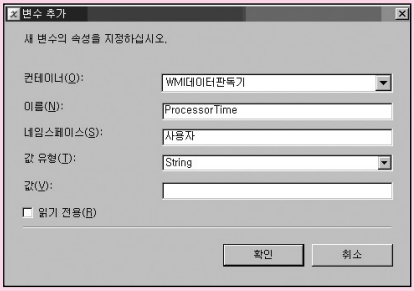
5. **WMI 데이터 판독기 작업 편집기** 창에서 WqlQuerySourceType의 값이 **직접 입력**으로 선택되었는지 확인합니다. 변수에 저장된 WQL 쿼리를 이용하거나 파일에 저장된 쿼리를 이용할 경우에는 **SQL 실행 작업**에서와 동일한 방법으로 이 속성값을 변경하면 됩니다.

6. WqlQuerySource에서 ... 버튼을 클릭하여 직접 WQL 쿼리를 입력합니다.
다음과 같은 WQL 쿼리를 사용하여 %Processor Time 의 정보를 읽어옵니다. CPU가 여러 개인 환경에서는 개별 CPU 별로도 정보를 읽어올 수 있지만 본 예제에서는 _Total 에 대한 값만 읽어오도록 설정합니다.

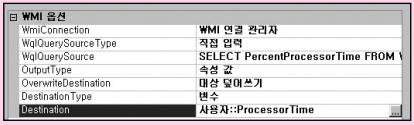
```
SELECT PercentProcessorTime FROM Win32_PerfFormattedData_
PerfOS_ Processor WHERE Name = "_Total"
```

7. OutputType는 **속성 값**, OverwriteDestination은 **대상 덮어쓰기**로 지정합니다.

8. 결과 값을 변수에 저장하는 부분을 구현합니다. 앞 단계에서 저장할 변수를 추가하지 않았기 때문에 먼저 변수를 추가합니다. DestionationType을 **변수**로 지정한 후 Destionation 속성에서 ... 버튼을 눌러 나타나는 항목 중 〈**새 변수**...〉를 선택합니다.
변수를 추가할 때, 변수의 사용 영역을 나타내는 컨테이너와 이름, 유형 등을 지정합니다.



WQL 쿼리를 이용하여 현재 서버의 %Processor Time 정보를 읽어 들인 후 이 값을 ProcessorTime이라는 변수에 저장하는 단계가 완성되었습니다.



이후 단계에서는 이 값이 30 미만인지를 판단한 후, 현재 수행되는 프로세스 정보를 수집하는 작업을 구성합니다.

9. **제어 흐름 영역**에 별도의 **WMI 데이터 판독기 작업**을 추가한 후, 이름을 **[프로세스 정보 수집]**으로 변경합니다.

10. 추가한 작업을 더블 클릭하여 편집기 창을 연 후, WmiConnection 속성에 앞 부분에서 생성한 **WMI 연결 관리자**를 지정합니다.

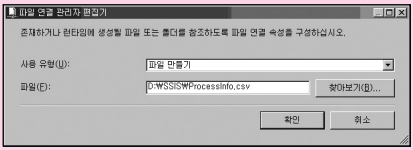
11. WqlQuerySourceType을 **직접 연결**로 설정하고 WqlQuerySource 창에 아래의 WQL 쿼리를 입력합니다.

```
SELECT Name, IDProcess, PageFileBytes, VirtualBytes, ElapsedTime,
ThreadCountFROM Win32_PerfFormattedData_PerfProc_ Process
```

12. OutputType를 **데이터 테이블**로, OverwriteDestination을 **대상 덮어쓰기**, DestinationType을 파일 연결로 지정합니다.

13. Destionation 속성에서 ... 버튼을 클릭한 후 〈**새 연결**..〉을 선택합니다. 프로세스 정보가 저장될 결과 파일을 지정하는 작업입니다.

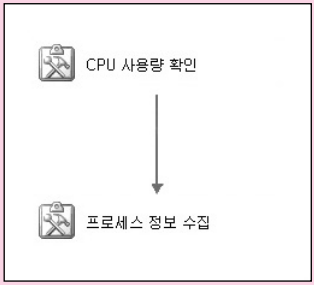
파일 연결 관리자 편집기에서 사용 유형을 **파일 만들기**로 지정한 후 새로운 파일 이름을 입력하거나, **기존 파일**로 지정한 후 해당하는 파일을 선택합니다.



WQL 쿼리를 이용하여 서버의 모든 프로세스 정보를 읽어와서 대상 위치인 텍스트 파일에 저장하도록 설정하는 단계가 완료되었습니다.

이제 두 개의 WMI 작업을 연결합니다.

14. 처음에 추가한 **[CPU 사용량 확인]** 작업의 녹색 선을 **[프로세스 정보 수집]** 작업으로 연결합니다.



CPU 사용량 확인이라는 작업이 성공적으로 수행되면 다음 작업인 프로세스 정보 수집 작업이 수행됩니다. 하지만, 본 예제의 시나리오는 CPU 사용량이 30% 미만인 경우에만 수행하도록 해야 하기 때문에 선행 제약 조건에서 추가 조건을 설정해야 합니다.

15. 녹색 연결선을 더블 클릭하여 **선행 제약 조건 편집기**를 엽니다.

16. **선행 제약 조건 편집기**에서 **평가 작업(E)**을 **식 및 제약 조건**으로 변경합니다. **식(X)** 부분에 다음과 같은 조건식을 입력합니다.

```
@ProcessorTime <" 30"
```

17. 패키지를 실행한 후 출력되는 결과를 확인합니다.

다음은 WMI 데이터 판독기에서 사용할 수 있는 WQL 쿼리 예제입니다.

```
--논리적 디스크 정보를 반환하는 쿼리
SELECT Name, FileSystem, FreeSpace, Size FROM Win32_LogicalDisk
--시스템의 페이지 파일(Paging File) 정보를 반환하는 쿼리
SELECT Description, FileSize FROM Win32_PageFile
--성능 모니터의 Logical Disk에 포함된 카운터의 값들을 출력하는 쿼리
SELECT * FROM Win32_PerfRawData_PerfDisk_LogicalDisk
--CPU 정보를 출력하는 쿼리
SELECT Caption, CpuStatus, DeviceID FROM Win32_Processor
```

[참고]
WQL 쿼리는 다음 링크의 Scriptomatic 이라는 프로그램을 이용하여 쉽게 작성하실 수 있습니다.
http://www.microsoft.com/technet/scriptcenter/tools/scripto2.mspx

WMI 이벤트 감시자 작업

WMI 이벤트 감시자 작업은 시스템에서 발생하는 이벤트 정보를 인식하는 작업입니다. WMI 이벤트란 파일의 생성이나 삭제, 응용 프로그램 실행, 프로그램 설치, 서비스 종료 등과 같은 행위를 말합니다.

WMI 이벤트 감시자를 이용하면 다음과 같은 형태의 작업을 수행할 수 있습니다.

- 지정된 폴더에 원하는 파일이 전송되는 경우, 파일을 읽어서 테이블에 로딩하는 작업을 수행합니다.
- 특정 폴더로 파일이 생성되면 자동으로 여러 대상으로 배포하는 작업을 수행합니다.
- 특정 서비스 또는 응용 프로그램이 종료될 때 사용자가 지정한 작업을 수행하도록 합니다.

WMI 데이터 판독기 작업과 마찬가지로 **WMI 이벤트 감시자 작업** 역시 **WMI 연결**을 사용합니다.

WMI 이벤트 감시자 작업의 속성은 다음과 같습니다.

- WmiConnection - WMI 연결을 지정합니다.
- WqlQuerySourceType - WMI 이벤트를 인식하기 위한 WQL 쿼리의 입력 형태를 지정합니다. 직접 입력하도록 설정하거나, 변수에 WQL 쿼리를 저장한 후 이 변수를 사용하도록 지정할 수 있으며, 별도의 파일로 만든 후 이 파일을 사용하도록 지정할 수도 있습니다.
- WqlQuerySource - **직접 입력**으로 지정한 경우 WQL 쿼리를 입력합니다.

- ActionAtEvent - 이벤트가 발생되었을 때 행할 수행 동작을 설정합니다.
  - 이벤트 기록 - 단순히 이벤트가 발생된 것을 감지하고 AfterEvent 속성에 지정된 대로 진행합니다.
  - 이벤트를 기록하고 SSIS 이벤트를 실행합니다. - AfterEvent 속성에 지정된 대로 진행할 뿐만 아니라 이벤트 처리기에서 WMI 이벤트 발생 시 수행되도록 설정한 이벤트 처리 작업을 실행합니다. 즉, 이벤트 처리기의 WMIEventWatcherEvent Occurred 부분에 구성된 작업들이 수행됩니다.

- AfterEvent - 이벤트가 발생될 경우 작업 결과를 성공 또는 실패로 반환하고 작업을 종료하거나 다시 감시하도록 설정합니다. 이벤트를 다시 감시하도록 설정할 경우, 아래에 있는 NumberOfEvents의 횟수만큼 반복하게 됩니다.

- ActionAtTimeout - Timeout 속성에서 지정한 시간(초)이 지날 경우 수행할 작업 형태를 지정합니다. 이 속성 역시 ActionAtEvent의 속성과 마찬가지로 단순히 AfterTimeout에 지정된 대로 진행할지 또는 이벤트 처리기의 WMIEventWatcherEventTimeout 이벤트 처리 작업까지 수행할지를 설정합니다.

- AfterTimeout - Timeout 속성에서 지정된 시간(초)이 지난 경우, 작업 결과를 성공 또는 실패로 반환하고 작업을 종료하거나 다시 감시하도록 설정합니다. 이벤트를 다시 감시하도록 설정할 경우에는 NumberOfEvents의 횟수만큼 반복하게 됩니다.

- NumberOfEvents - 이벤트가 발생되거나 지정된 시간을 초과한 경우, 이벤트를 다시 감시할 횟수를 설정합니다. 만약 이 값을 0으로 설정하면 계속해서 감시하게 됩니다.

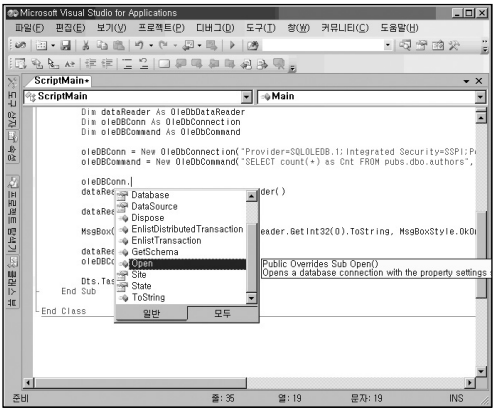
- Timeout - 이벤트가 발생할 때까지 대기할 시간을 지정합니다. 0으로 지정할 경우에는 이벤트가 발생될 때까지 무한 대기하게 됩니다.

스크립트 작업 및 ActiveX 스크립트 작업

ActiveX 스크립트 작업 및 **스크립트 작업**은 기본적으로 제공되는 작업 개체 이외의 기능이 필요하거나 SQL 쿼리로 처리하기 어려운 작업에서 이용할 수 있는 유용한 작업 개체입니다. 예를 들어 문자열 데이터를 구분 기호로 분리하는 작업을 수행해야 할 경우, SQL에서는 문자열을 분리하는 기본 함수가 없기 때문에 사용자 정의 함수(UDF)나 저장 프로시저를 이용하여 처리해야 합니다. 하지만 **ActiveX 스크립트 작업** 또는 **스크립트 작업**을 이용할 경우, *Split()* 함수나 문자열의 *split* 속성을 이용하면 간단히 처리할 수 있습니다. 이 외에도 ADO(Active Data Object)나 OWC(Office Web Components) 등과 같이 시스템에 등록된 개체를 사용하여 작업을 수행할 수도 있으며 이미 만들어진 개발 컴포넌트 들을 이용하는 프로그래밍 작업도 구현 할 수 있습니다.

SQL 2005 SSIS에서는 스크립트와 관련된 여러 사항들이 향상되었습니다.
우선 기본 스크립트 언어로 VB.NET 스크립트를 사용합니다. .NET 환경에서 개발되고 수행되기 때문에 다양한 여러 기능들을 쉽고 간결하게 구현할 수 있습니다. 이전 버전에서는 VBScript나 JavaScript를 사용할 수 있었지만, SQL 2005 SSIS에서는 VB.NET 스크립트만 사용할 수 있습니다. 하지만 이전 버전과의 호환성을 유지하기 위한 목적으로 **ActiveX 스크립트 작업** 개체가 존재하기 때문에 필요한 경우 VBScript나 JavaScript를 사용할 수도 있습니다.

스크립트를 작성하기 위한 개발 환경 또한 많이 향상되었습니다. SQL 2005 SSIS의 **스크립트 작업**에서는 **VSA**(Visual Studio for Application)라는 전용 개발 환경이 제공됩니다. VSA는 일반적인 Visual 개발 환경과 매우 유사한 환경을 제공합니다.



스크립트 작업

스크립트 작업은 VB.NET 스크립트를 이용하여 사용자가 원하는 작업을 수행할 수 있는 개체입니다. VSA(Visual Studio for Application) 개발 환경에서 작업에 필요한 스크립트를 작성할 수 있습니다. **스크립트 작업** 내에서 패키지의 변수를 사용하기 위해서는 속성 창에서 미리 사용할 변수를 지정해야 합니다.

- ScriptLanguage - 사용할 스크립트 언어를 설정합니다. Microsoft Visual Basic .NET 언어만 있습니다.

- PrecompileScriptIntoBinaryCode - 작성된 스크립트를 실행하기 전에 미리 컴파일할지를 설정합니다. 미리 컴파일하도록 설정을 할 경우 패키지가 실행되기 전에 미리 컴파일 된 상태로 저장되기 때문에 실제 수행될 때에는 수행 시간이 단축됩니다. 하지만 미리 컴파일하도록 설정을 하면 스크립트를 컴파일한 코드(Binary Code)가 패키지에 포함되기 때문에 패키지 파일의 크기는 약간 커질 수 있습니다. 하지만, 패키지의

크기가 큰 문제가 되지 않는다면 가급적 이 옵션을 True로 해서 사용하도록 합니다. 참고로, 이 속성 값을 변경하였다면 반드시 스크립트 디자인 창을 다시 열었다가 닫아야 반영됩니다. 64bit 환경에서 패키지를 수행할 경우에는 반드시 이 값을 True로 설정해야 합니다.

- EntryPoint - 스크립트 내에서 시작할 클래스 위치를 설정합니다. 기본 위치는 ScriptMain 입니다.

- ReadOnlyVariables - 스크립트 내에서 패키지의 변수를 사용하는 경우에 이용됩니다. 예를 들어, 패키지에서 미리 만들어진 Val1 이라는 변수와 Val2 라는 변수를 스크립트 내에서 읽기 전용으로 이용하려면 다음과 같이 설정해야 합니다.
  사용자::Val1, **사용자**::Val2 또는 간단히 Val1, Val2로 지정합니다.

- ReadWriteVariables - 스크립트 내에서 읽기 및 쓰기로 변수를 사용할 경우 지정합니다.

ActiveX 스크립트 작업

ActiveX **스크립트 작업**은 이전 버전인 SQL 2000 DTS 패키지를 SQL 2005 SSIS로 업그레이드 시 호환성을 유지하기 위한 작업 개체입니다.

> ***[참고]***
> *이 작업 개체는 다음 버전에서는 제거될 기능입니다. 패키지에서 스크립트 작업을 이용해야 할 경우 ActiveX 스크립트 작업 대신 스크립트 작업을 이용하시기 바랍니다. 또한, 기존 DTS 패키지를 업그레이드 한 경우에도 ActiveX 스크립트 작업을 스크립트 작업으로 수정하기 바랍니다.*

- Language - 스크립트 언어를 설정합니다.
- Script - 작업 스크립트를 지정합니다.
- EntryMethod - 시작할 함수를 지정합니다.

ActiveX **스크립트 작업**에서도 패키지의 변수를 사용할 수 있지만 **스크립트 작업**에서와 같이 별도로 설정할 필요는 없습니다. 패키지 내에 변수를 사용하는 방법은 다음의 **따라하기**를 참고하기 바랍니다.

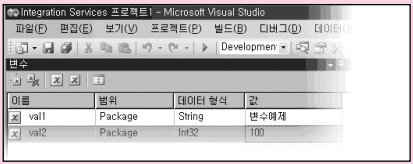
[따라하기] 스크립트 작업

스크립트 작업 및 ActiveX **스크립트 작업**에서 패키지에 지정된 사용자 변수를 사용하는 방법을 간단히 살펴보겠습니다.

1. 작업 창의 제어 흐름 영역의 빈 곳에서 마우스 오른쪽 버튼을 클릭하여 나타나는 메뉴 중, **변수(S)**를 선택합니다.

2. 변수 창에서 가장 왼쪽에 있는 변수 추가 버튼을 사용하여 변수를 추가합니다.

이름 : val1	데이터 형식 : String	값 : 변수예제
이름 : val2	데이터 형식 : Int32	값 : 100

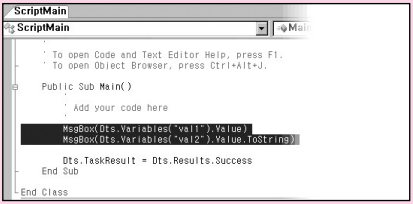


3. 도구 상자에서 **스크립트 작업**을 선택한 후, **제어 흐름** 영역에 추가합니다.

4. 스크립트 작업 편집기에서 ReadOnlyVariables 속성 값에 val1, val2를 추가한 후, 아래에 있는 **스크립트 디자인(S)**를 클릭하여 스크립트 편집기를 엽니다.

5. Sub Main() 내에 다음 스크립트를 추가한 후, 편집기를 닫고 제어 흐름 영역에서 **스크립트 작업**을 실행합니다.

```
MsgBox(Dts.Variables("val1").Value)
MsgBox(Dts.Variables("val2").Value.ToString)
```

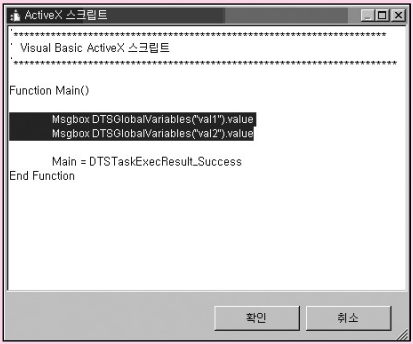


6. 도구상자에서 ActiveX **스크립트 작업**을 선택하여 **제어 흐름** 영역에 추가합니다.

7. ActiveX **스크립트 작업 편집기**에서 스크립트 속성 부분에 있는 ... 버튼을 클릭하여 스크립트 편집 창을 엽니다.

8. ActiveX **스크립트** 창에 다음 코드를 추가한 후, 편집 창을 닫고 ActiveX **스크립트 작업**을 실행합니다.

```
Msgbox DTSGlobalVariables("val1").value
Msgbox DTSGlobalVariables("val2").value
```



패키지 실행

SSIS에서는 **패키지 실행 작업**을 이용하여 다른 SSIS 패키지를 호출하여 실행할 수 있습니다. 또한 **DTS 2000 패키지 실행 작업**을 이용하여 SQL 2000 DTS 패키지를 사용할 수 있습니다.

패키지 호출 기능은 다음과 같이 이용할 수 있습니다.

1. 복잡한 데이터 추출 프로세스 또는 대규모의 데이터 처리 프로세스의 각 작업들을 작은 단위로 분리하여 관리할 수 있습니다. 예를 들어 데이터 웨어하우스의 데이터를 적재하기 위한 전체 추출 단계를 하나의 패키지로 만들 수도 있습니다. 하지만 하나의 패키지로 만들 경우, 유지 관리나 추가 개발 작업을 수행하는 것이 상당히 어려울 수 있습니다. 전체 프로세스를 하나의 패키지로 만드는 대신 주제별로 패키지를 만든 후, 마스터 패키지(=부모

패키지)에서 각각의 개별 패키지(=자식 패키지)를 호출하는 방식으로 구성할 경우 보다 효율적으로 개발 및 운영할 수 있습니다.

2. 패키지 보안을 이용할 수 있습니다. 예를 들어 데이터 추출 프로세스 중 중요한 부분이 공개되면 안 되는 경우, 해당 부분만을 별도의 패키지로 만들고 패키지 호출 작업으로 해당 부분의 패키지를 호출하도록 할 수 있습니다. 어플리케이션에서 중요한 기능을 DLL 이나 별도의 외부 컴포넌트 등으로 만들어 소스는 볼 수 없는 대신, 기능만을 사용하도록 하는 방식과 유사합니다. 별도로 만든 패키지에는 SSIS에서 기본적으로 제공하는 암호 설정 기능을 이용하여 조회나 수정이 불가능하도록 설정할 수 있습니다.

3. 작업 수행 및 관리가 용이합니다. 예를 들어 매일 수행해야 할 패키지가 10개가 있다고 가정합니다. 각 패키지를 수행하도록 하기 위해서는 10개의 SQL Server 에이전트 작업이나 윈도우 예약 작업을 만들어야 합니다. 만약 각 패키지 간에 수행되는 순서가 정해져 있거나 서버의 부하를 주지 않기 위해 동시에 수행해야 할 패키지의 수를 제한해야 할 경우에는 예약 작업 내에 별도의 관리 부분을 추가해야 합니다. 이런 방법 대신 하나의 마스터 패키지(=부모 패키지)를 만든 후, 패키지 실행 작업을 이용하여 10개의 패키지(=자식 패키지)를 호출하도록 만들 수 있습니다. 패키지가 수행되는 순서나 동시에 수행할 패키지 수를 쉽게 조절할 수도 있습니다.

패키지를 호출하는 패키지를 **부모 패키지**라 하고, 호출 당하는 패키지를 **자식 패키지**라 합니다. 부모 패키지는 자식 패키지에게 값을 전달할 수 있으며 자식 패키지에서는 이 값을 넘겨받아 사용할 수 있습니다. 예를 들어 부모 패키지에서 자식 패키지를 호출할 때 자식 패키지 내에 정의된 rundate 라는 변수에 20060807 이라는 값을 지정하여 호출할 수 있습니다. 부모 패키지가 자식 패키지의 변수에 값을 설정하여 호출하는 방법은 **따라하기**를 참고하기 바랍니다.

패키지 실행 작업

패키지 실행 작업은 DTS 2000 **패키지 실행 작업**에 비해 단순합니다. 이는 자식 패키지의 변수에 값을 지정하는 방식이 달라졌으며, 패키지를 호출할 때 필요한 연결 또한 **연결 관리자**의 OLE DB **연결**이 담당하기 때문입니다.

- Location - 호출할 패키지가 저장된 형태를 지정합니다.
  - SQL Server - SQL Server의 msdb 데이터베이스에 저장되어 있는 경우, Connection 속성에서 OLE DB 연결을 지정합니다. 이 때, 사용되는 OLE DB 연결에서 데이터베이스가 반드시 msdb를 지정할 필요는 없습니다.
  - 파일 시스템 - 파일 형태(*.dtsx)로 저장된 패키지를 호출하는 경우, Connection 속성에서 파일 연결을 지정합니다.
- Connection - 패키지를 호출할 연결을 설정합니다.
- PackageName - Location의 값이 SQL Server인 경우, 호출할 패키지를 지정합니다.
- Password - 호출할 패키지에 암호가 설정되어 있는 경우, 암호를 지정합니다.
- ExecuteOutOfProcess - 패키지를 호출하여 실행할 때, 현재의 프로세스 내에서 실행할지, 또는 별도의 프로세스를 생성하여 실행할지를 설정합니다. 만약 이 속성값을 true로 설정하면 패키지 호출 시 별도의 dtshost.exe 프로세스가 생성되어 패키지가 실행되며, 부모 패키지와는 별도의 메모리 공간 및 작업 스레드를 사용하게 됩니다.

DTS 2000 패키지 실행 작업

DTS 2000 패키지 실행 작업은 SSIS 패키지 내에서 SQL 2000 DTS 패키지를 호출하는 작업 개체입니다. SQL 2000 DTS 패키지가 업그레이드 하기에 복잡하거나 **동적 속성 작업**, 호환되지 않는 ActiveX **스크립트 작업** 등이 포함되어 있어서 기존 DTS 패키지를 그대로 사용해야 하는 경우 이 작업 개체를 이용할 수 있습니다.

msdb에 저장되어 있는 패키지를 호출하거나 확장자가 .dts 인 파일 형태의 패키지를 호출할 수 있습니다. SQL 2000 DTS 패키지를 사용하기 위해서는 **DTS 2000 런타임 엔진**이 설치되어 있어야 하며, 다음 링크(http://msdn2.microsoft.com/ko-kr/library/ms143706.aspx)에서 다운로드 받을 수 있습니다.

연결 - StorageLocation 이 SQL Server 인 경우 지정합니다. DTS 2000 **패키지 실행 작업**에서는 연결 관리자의 연결을 사용하지 않고 직접 Server 및 패키지를 지정합니다.

- SQLServer - 패키지가 저장된 서버를 지정합니다.
- AuthentificationMode - 서버의 인증 방식을 설정합니다.
- UserName - 인증 방식이 SQL Server 인 경우, 로그온 아이디를 입력합니다.
- Password - 인증 방식이 SQL Server인 경우, 로그온 패스워드를 입력합니다.

위치

- StorageLocation - 호출할 패키지의 형태를 지정합니다. **구조적 저장소**로 선택한 경우에는 확장자가 .dts 인 패키지 파일을 설정해야 합니다. **작업에 포함**은 아래에 있는 **내부에서 DTS2000 패키지 로드(L)**를 이용하여 DTS 패키지를 현재의 SSIS 패키지에 포함시킨 경우 선택됩니다.

일반

- Name - 패키지 실행 작업의 이름을 설정합니다.
- Description - 패키지 실행 작업의 설명을 설정합니다.

패키지

- PackageName - 호출할 패키지를 지정합니다.
- PackagePassword - 패키지가 암호화되어 있는 경우, 암호를 입력합니다.
- PackageID - 패키지 계보를 나타냅니다. DTS 패키지는 패키지 자체에서 버전 관리가 되기 때문에 동일한 이름의 패키지이더라도 서로 다른 계보를 가질 수 있습니다.

패키지 편집 - 지정한 DTS 패키지를 열어서 편집할 수 있습니다.
내부에서 DTS2000 패키지 로드 - 지정한 DTS 패키지를 현재의 SSIS 패키지 내에 포함시킵니다. 이 경우, 위치의 StorateLocation 속성값은 **작업에 포함**으로 변경되며, DTS 패키지가 저장된 SQL Server가 중지되거나 .dts 파일이 없더라도 DTS 패키지의 기능을 수행할 수 있습니다.

내부 변수와 외부 변수는 SQL 2000 DTS와 동일합니다. 내부 변수와 외부 변수는 자식 패키지의 속성 값을 설정하는데 이용한다는 공통점이 있습니다. 하지만, 다음과 같은 차이점이 있습니다.

- **내부 변수** - 부모 패키지에서 자식 패키지에 대한 변수 값을 지정할 수 있으며, 유형을 변경할 수 있습니다. 하지만, 이러한 값 지정 작업은 정적인 작업으로, 패키지 개발 시점에 고정적으로 지정하는 것이며, 런타임 시 변경되는 값을 할당하기 위해서는 외부 변수를 사용해야 합니다.

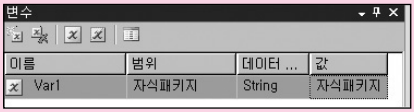
- **외부 변수** - 부모 패키지의 변수 값을 자식 패키지로 전송할 때 설정합니다. 자식 패키지의 변수와 동일한 이름으로 부모 패키지에서 변수를 만든 후(대소문자 구분), 외부 변수 부분에서 이 변수를 지정하면 자식 패키지를 호출할 때 자동으로 부모 패키지의 변수 값이 자식 패키지로 지정됩니다.

[따라하기] 패키지 실행

본 예제에서는 SQL 2005 SSIS 패키지(부모 패키지)에서 또 다른 SQL 2005 SSIS 패키지(자식 패키지)를 호출하는 것과, 부모 패키지에서 자식 패키지에게 변수를 넘기는 작업을 구현합니다.

1. 빈 패키지 파일을 추가한 후, 이름을 **자식패키지.dtsx** 라는 이름으로 변경합니다. 오른 쪽의 속성 창에서 해당 패키지 파일이 저장된 위치를 확인합니다.
   (본 예제에서는 D:₩SSIS₩Integration Services 프로젝트1₩자식패키지.dtsx 에 저장하였습니다.)

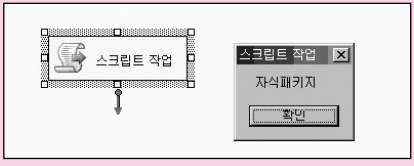
2. 제어 흐름 영역에서 마우스 오른쪽 버튼을 클릭하여 나타나는 메뉴 중 **변수(V)**를 선택한 후, 변수 창에서 Var1 이라는 String형 변수를 추가하고, 값을 "**자식 패키지**" 라고 지정합니다.



3. 도구 상자에서 스크립트 작업을 추가한 후, 스크립트 작업 편집기에서 ReadOnly Variables 속성에 Var1을 추가합니다. 스크립트 디자인(S)을 클릭하여 VSA를 연 후 다음과 같은 스크립트를 추가합니다.

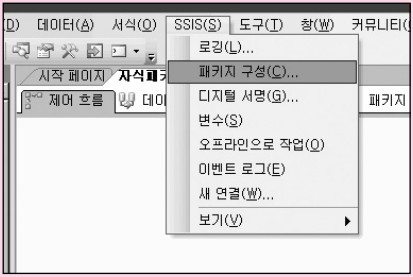
```
    MsgBox(Dts.Variables("Var1").Value)
Public Sub Main()
        Add your code here
    MsgBox(Dts.Variables("Var1").Value)

    Dts.TaskResult = Dts.Results.Success
End Sub
```

4. 패키지를 저장한 후, 실행합니다.



이제 부모 패키지의 변수 값을 받아오는 부분을 설정합니다. 이 부분에서는 SSIS의 구성 기능을 이용하게 됩니다.

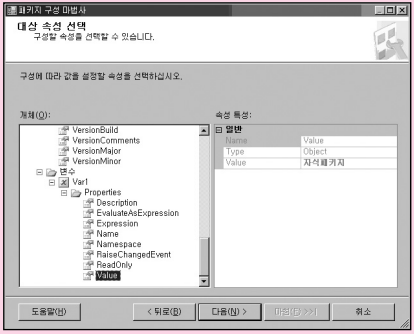
5. 상단의 메뉴 중, SSIS(S) → **패키지 구성**(C)을 선택합니다.



6. **패키지 구성 도우미** 창에서 **패키지 구성 설정**(E)을 선택한 후, 아래에 있는 **추가**(A) 버튼을 눌러 **패키지 구성 마법사**를 시작합니다.

7. **패키지 구성 마법사**에서 **구성 유형**(T)을 **부모 패키지 변수**로 지정하고, **구성 설정을 직접 지정**, 부모 변수를 ParentVar1로 지정합니다.

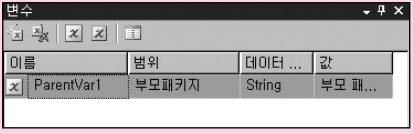
8. **다음**을 눌러 **대상 속성 선택** 창이 나타나면, 개체 부분에서 **변수** → Var1 → Properties → Value를 선택합니다.



9. **다음(N)**을 눌러 구성 이름을 지정한 후, 구성 설정을 마칩니다.

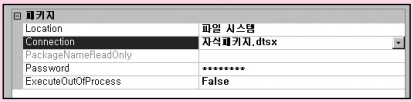
10. 작업 창의 오른쪽에 있는 솔루션 탐색기 중 **SSIS 패키지** 부분에서 **부모패키지.dtsx** 라는 이름으로 패키지 파일을 하나 추가합니다.

11. 변수 창에서 **ParentVar1** 이라는 String형 변수를 하나 추가하고, 값을 "부모 패키지"로 지정합니다.

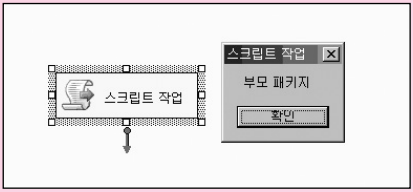


12. 도구 상자에서 **패키지 실행 작업**을 선택하여 제어 흐름에 추가한 후, **패키지 실행 작업 편집기**에서 수행할 자식 패키지를 설정합니다.

Location 은 **파일 시스템**이며, Connection 속성을 클릭한 후, 〈**새 연결..**〉을 선택하여 나타나는 **파일 연결 관리자 편집기**에서 **사용 유형**을 **기존 파일**로 지정하고, 1단계에서 확인한 경로로 자식 패키지 파일을 지정합니다.



13. 패키지를 수행하여 나타나는 메시지를 확인합니다.



부모 패키지에서 자식 패키지를 호출하는 방식은 복잡한 처리 프로세스를 개발하거나 관리할 때 유용하게 사용할 수 있습니다. 서로 연관 있는 작업들을 각각의 개별 패키지로 분리한 후, 각 패키지들을 관리하는 부모 패키지를 만들어 운용하는 것이 효과적일 수 있습니다.

XML 작업

SQL 2005 SSIS에 포함되어 있는 **XML 작업**은 XML 데이터를 이용하는 다양한 작업을 수행할 수 있는 작업 개체입니다.
XML 데이터를 조회하거나, 변경, 병합 또는 새로운 XML 결과 파일을 생성할 수 있습니다.

XML 작업 개체를 이용하여 수행할 수 있는 작업은 다음과 같습니다.

- 여러 XML 데이터를 병합하여 하나의 XML 파일로 병합하는 작업
- XSLT Style Sheet을 이용하여 XML 파일의 결과값 출력 작업
- XPATH를 이용한 XML 파일 내의 데이터 조회 작업
- 서로 다른 두 개의 XML 파일들에 대한 비교 및 XML Diffgram 파일에 해당 차이점 기록 작업
- XML 파일에 대한 유효성 검사 작업

XML 작업 개체의 속성 편집기 창에는 다른 작업들에 비해 많은 속성들이 있으며, 작업 유형(OperationType)에 따라 각 작업 속성의 의미가 달라집니다.

공통 속성

입력

- OperationType - XML 작업의 유형을 설정합니다. 작업 속성은 다음과 같습니다.
  - Validate - DTD(문서 유형 정의) 또는 XSD(XML 스키마 정의) 스키마와 비교하여 XML 문서의 유효성을 검사합니다.
  - XSLT - 원본 XML 문서를 XSL Style Sheet에 지정된 형태로 출력합니다.
  - XPATH - XPATH라는 XML 데이터 조회 문을 이용하여 원본 XML 문서에 있는 데이터를 조회합니다.

- Merge - 두 개의 XML 문서를 병합합니다.
- Diff - 두 개의 XML 문서를 비교합니다. 다양한 수준으로 비교 작업을 설정할 수 있습니다.
- Patch - 비교(Diff) 작업에서 두 문서간의 비교 결과인 XDL Diffgram 출력을 생성한 경우, 이를 이용하여 원본 문서에 변경 작업을 수행합니다.
- SourceType - 원본 XML의 입력 형태를 지정합니다. 변수, 직접 입력, 파일 연결로 설정할 수 있습니다.

출력

- SaveOperationResult - 결과를 저장할 것인지를 설정합니다. True로 설정할 경우 OperationResult 속성에서 결과 파일 형태를 지정할 수 있습니다.

OperationType = Validate

원본 XML 문서에 대한 유효성을 점검합니다.

유효성 검사 옵션

- ValidationType - 유효성 검사 작업 유형을 설정합니다.
  - DTD - DTD(Document Type Definition)를 사용합니다.
  - XSD - XSD(XML Schema Definition)를 사용합니다. 이 경우, XSD 스키마 파일을 지정해야 합니다
- FailOnValidationFail - 유효성 검사 결과가 실패(=유효하지 않음)인 경우, XML 작업을 실패로 처리할지를 설정합니다.

OperationType = XSLT

XSLT는 XML 문서를 다른 형태의 XML 형태 또는 문서 형태로 출력하기 위한 변환 언어로 작성된 문서입니다. OperationType 속성 값을 XSLT로 지정한 후, XSLT 문서를 설정하여, 원본 XML 문서를 사용자가 원하는 다른 형태로 출력할 수 있습니다.

OperationType = XPATH

XPATH 쿼리를 이용하여 원본의 XML 데이터의 값을 조회합니다.

XPath 옵션

- PutResultInOneNode - 결과를 단일 노드에 저장할지를 설정합니다.

- XPathOperation - 결과 형태를 설정합니다.
  - 계산 - XPath 쿼리의 결과를 반환합니다.
  - 노드 목록 - XPath에서 지정한 노드를 XML 형태로 반환합니다.
  - 값 - XPath에서 지정한 노드의 내부 텍스트 값을 연결 문자열로 반환합니다.

두 번째 피연산자

XQuery를 지정합니다. **변수**, **직접 입력**, **파일 연결**로 지정할 수 있습니다.

OperationType = Merge

두 개의 서로 다른 XML 문서 또는 데이터를 하나로 병합하는 작업을 수행합니다.

두 번째 피연산자

입력 부분에 지정된 첫 번째 XML 문서와 병합할 두 번째 XML 문서를 지정합니다.

병합 옵션

병합 작업을 수행할 때 사용할 XPath 쿼리 유형과 쿼리를 지정합니다. 만약 단순히 두 문서를 병합하고자 하는 경우에는 XPathStringSource를 공백으로 설정하면 됩니다. 이 경우, 두 문서 간의 구조는 동일해야 합니다.

OperationType = Diff

두 개의 서로 다른 XML 문서 또는 데이터에 대해 비교 작업을 수행합니다.

두 번째 피연산자

입력 부분에 지정된 첫 번째 XML 문서와 비교할 두 번째 XML 문서를 지정합니다.

비교 옵션

- DiffAlgorithm - 문서를 비교할 때 사용할 알고리즘을 지정합니다.
  - 자동 - XML 작업에서 처리 속도가 빠른 알고리즘을 사용할 것인지 아니면 정확도가 높은 알고리즘을 사용할 것인지 결정합니다.
  - 빠름 - 빠르지만 정확도가 낮은 비교 알고리즘을 사용합니다.
  - 정확 - 정확한 비교 알고리즘을 사용합니다.

- DiffOptions - 비교 작업에서 적용할 비교 옵션을 설정합니다.
  - IgnoreXMLDeclaration - XML 선언을 비교할지를 설정합니다.
  - IgnoreDTD - DTD(문서 유형 정의)를 무시할지를 설정합니다.
  - IgnoreWhiteSpaces - 공백을 비교할지를 설정합니다.
  - IgnoreNameSpaces - 각 요소의 네임스페이스 URI(Uniform Resource Identifier)와 해당 요소의 특성 이름을 비교할지를 설정합니다.
  - IgnoreProcessingInstructions - 처리 명령을 비교할지를 설정합니다.
  - IgnoreOrderOfChildElements - 자식 요소의 순서를 비교할지를 설정합니다.
  - IgnoreComments - 주석 노드를 비교할지를 설정합니다.
  - IgnorePrefixes - 요소와 특성 이름의 접두사를 비교할지를 설정합니다.

- FailOnDifference - 두 문서 간 차이점이 존재하는 경우 작업을 실패로 처리할지를 설정합니다.
- SaveDiffGram - 두 문서 간의 비교 결과인 DiffGram 문서를 출력할지를 설정합니다.
- DiffGramSave - SaveDiffGram 속성이 True인 경우, DiffGram 문서를 저장할 위치를 지정합니다.

OperationType = Patch

서로 다른 두 개의 XML 문서를 비교한 후 생성되는 비교 결과 파일인 DiffGram 문서를 이용하여 원본 XML 문서의 차이점을 보정하여 비교 파일과 동일한 형태로 만드는 작업입니다. 예를 들어 동일한 구조의 XML 문서이며 원본 파일에는 〈ID〉497〈/ID〉, 비교 대상 파일에는 〈ID〉499〈/ID〉 인 데이터가 있는 경우, Diff 연산을 수행하면 원본의 497 값이 대상의 499이 다르다는 정보를 포함하는 DiffGram 문서가 생성됩니다. 이 DiffGram 문서와 원본 XML 문서를 이용하여 ID를 499로 변경하는 작업이 Patch 작업입니다.

대량 삽입 작업

대량 삽입 작업은 SQL Server의 BULK INSERT 명령이나 bcp.exe 유틸리티와 같이 텍스트 형식의 데이터 파일을 SQL Server의 테이블로 입력할 때 이용되는 작업 개체입니다. 대량 삽입 작업은 데이터 흐름 엔진을 사용하지 않기 때문에 단순히 원본 텍스트 파일에서 대상 테이블로 로딩하는 작업만 가능하며, 데이터의 가공이나 집계, 정렬 등과 같은 변환 기능은 구현할 수 없습니다.

데이터 처리를 위한 준비 작업이나 단순 데이터 로딩 작업과 같이 변환 과정이 필요하지 않는 로딩 작업을 수행해야 할 경우 대량 삽입 작업을 이용하는 것이 효과적이며 다른 방식에 비해 성능도 우수합니다.

대량 삽입 작업 편집기 내에 있는 **연결** 탭과 **옵션** 탭에서 **대량 삽입 작업**에 대한 속성을 설정합니다.

연결

연결 탭에서는 원본 및 대상, 원본의 서식을 설정합니다.

대상 연결

- Connection - 원본 텍스트 파일의 데이터를 저장할 대상 테이블을 지정합니다. **연결 관리자**의 OLE DB 연결을 사용합니다.
- DestinationTable - 대상 테이블을 지정합니다.

서식

- Format - 대량 삽입 작업을 위한 서식의 형태를 선택합니다.
  - 파일 사용 - 서식이 지정된 파일을 사용합니다. BULK INSERT 명령에서 WITH (FORMATFILE = …) 옵션과 동일합니다.
  - 지정 - 직접 RowDelimeter 및 ColumnDelimerer의 속성을 지정합니다.
- RowDelimeter - 행 구분자를 설정합니다.
- ColumnDelimeter - 열 구분자를 설정합니다.

원본 연결

- File - **연결 관리자**의 **파일 연결**을 사용하여 원본으로 사용될 텍스트 파일을 지정합니다.

옵션

옵션 탭에서는 CodePage나 FileType 등과 같이 고급 옵션을 설정할 수 있습니다.

고급 옵션

- CodePage
  - ACP - char, varchar 또는 text 데이터 형식의 열은 ANSI/Microsoft Windows 코드 페이지(ISO 1252)에서 SQL Server 코드 페이지로 변환됩니다.
  - OEM - char, varchar 또는 text 데이터 형식의 열은 시스템 OEM 코드 페이지에서 SQL Server 코드 페이지로 변환됩니다.

- RAW - 다른 코드 페이지로의 변환이 이루어지지 않는 가장 빠른 옵션입니다.
- 기타 코드 페이지 - 특정 코드(예 : 949, 1252)로 변환됩니다.

- DataFileType
  - char - 일반 문자 형식입니다.
  - native - 네이티브(데이터베이스) 데이터 형식입니다. bcp 유틸리티로 SQL Server 에서 데이터를 대량 로드하여 네이티브 데이터 파일을 만들 수 있습니다. 네이티브 형식은 char 형식보다 더욱 성능이 뛰어납니다.
  - widechar - 유니코드 문자 형식입니다.
  - widenative - 데이터가 유니코드로 저장되는 네이티브(데이터베이스) 데이터 형식입니다. char, varchar 및 text 열은 제외됩니다. bcp 유틸리티로 SQL Server 에서 데이터를 대량 로드하여 widenative 데이터 파일을 만들 수 있습니다. widenative 값은 widechar 값을 대체하며 보다 뛰어난 성능을 제공합니다. 데이터 파일에 ANSI 확장 문자가 포함되어 있으면 'widenative' 로 지정해야 합니다.

- BatchSize - 일괄 처리의 행 수를 지정합니다. BatchSize를 0으로 설정하면 데이터가 단일 일괄 처리로 로드됩니다.
- LastRow - 삽입할 마지막 행의 번호를 지정합니다. 기본값은 0이며 이는 지정한 데이터 파일의 마지막 행을 가리킵니다.
- FirstRow - 삽입할 첫 번째 행의 번호를 지정합니다. 기본값은 1이며 이는 지정한 데이터 파일의 첫 번째 행입니다.

옵션

- Options
  - CHECK 제약 조건 - 대량 삽입 작업 중에 대상 테이블 또는 뷰의 모든 제약 조건을 확인하도록 지정합니다.
    CHECK 제약 조건 옵션을 지정하지 않으면 모든 CHECK 제약 조건이 무시됩니다. 하지만 Unique, Primary Key, Foreign Key 또는 Not Null 제약 조건은 항상 적용됩니다.

- Null 유지 - 삽입된 열에 기본값이 지정되지 않도록 하며 빈 열인 경우 Null 값을 유지하도록 지정합니다.
- ID 삽입 가능 - Identity 열에 데이터를 입력할 경우, 데이터 파일의 ID값이 대상 테이블의 ID열에 그대로 입력되도록 지정합니다. ID 삽입 가능을 선택하지 않는 경우, 이 열의 ID값은 확인하지만 가져오지는 않습니다.
- 테이블 잠금 - 대량 삽입 작업이 진행되는 동안 테이블 수준 잠금을 보유하도록 지정합니다. 테이블에 인덱스가 없고 TABLOCK이 지정되어 있으면 여러 클라이언트가 동시에 테이블을 로드할 수 있습니다. 기본적으로 잠금 동작은 대상 테이블의 table lock on bulk load 테이블 옵션에 의해 결정됩니다. 대량 로드 작업이 진행되는 동안에만 잠금을 보유하면 테이블에 대한 잠금 경합이 줄어들고 성능이 크게 향상됩니다.
- 트리거 실행 - 대량 삽입 작업 중에 대상 테이블에 정의된 삽입 트리거가 실행되도록 지정합니다. 트리거가 대상 테이블의 INSERT 작업에 대해 정의되어 있는 경우에는 모든 입력 데이터에 대해 발생합니다.

- SortedData - 데이터 파일의 데이터 정렬 방법을 지정합니다. 로드된 데이터가 테이블의 클러스터형 인덱스와 동일한 순서로 정렬되어 있으면 대량 삽입 작업의 성능이 향상됩니다. 데이터 파일을 다른 순서로 정렬하거나 테이블에 클러스터형 인덱스가 없으면 ORDER 절이 무시됩니다. SortedData 속성에서 지정된 열 이름은 대상 테이블의 열이어야 합니다. 예를 들어 대상 테이블에 Seq라는 열에 클러스터형 인덱스가 설정되어 있으며 데이터 파일이 Seq 순서대로 정렬되어 있는 경우, SortedData의 값을 Seq로 지정하면 처리 성능이 향상될 수 있습니다.

- MaxErrors - 대량 삽입 작업 수행 시 허용되는 최대 오류 수를 지정합니다. 대량 삽입 작업으로 가져올 수 없는 각 행은 무시되고 하나의 오류로 계산됩니다. max_errors를 지정하지 않으면 기본값은 0입니다.

프로세스 실행 작업

프로세스 실행 작업은 데이터 처리 프로세스 내에서 윈도우 어플리케이션 또는 콘솔 어플리케이션을 호출하는 작업 개체입니다. 프로그램에 따라서는 매개변수를 지정해야 할 경우도 있으며 실행 결과 값이나 오류 정보를 변수에 저장해야 할 경우도 있습니다.

프로세스

- RequireFullFileName - 프로그램을 실행할 때 전체 경로가 필요한지를 설정합니다. 만약 이 값을 True로 설정한 후, 해당 경로에 해당 파일이 없으면 작업이 실패합니다. OS의 PATH 속성에 포함된 경로에 있는 프로그램들 (notepad.exe 또는 mspaint.exe)은 전체 경로를 지정하지 않아도 수행이 됩니다.

- Executable - 실행할 프로그램 입니다. 프로그램이 일반적으로는 전체 경로가 포함된 프로그램 이름을 지정하며, 윈도우의 PATH 속성에 포함되어 있는 프로그램들을 실행시키는 경우에는 단순히 프로그램 이름 및 확장자만 입력해도 됩니다.
  (예 : notepad.exe)

- Arguments - 프로그램 실행 시 필요한 입력 변수를 지정합니다.

- WorkingDirectory - 프로그램이 실행될 작업 폴더를 지정합니다. 특정 폴더에서 해당 프로그램이 실행하도록 할 경우, 이 속성값을 지정합니다.

- StandardInputVariable - 프로그램에 입력 변수로 전달할 값이 저장된 SSIS 변수를 지정합니다. Arguments에 직접 입력 변수를 쓰는 대신 SSIS의 문자형 변수에 입력 변수로 사용할 값을 저장한 후 이 변수를 지정합니다.

- StandardOutputVariable - 프로그램이 수행된 후 반환되는 결과값을 저장할 변수를 지정합니다.

- StandardErrorVariable - 프로그램 수행 중 발생된 오류 정보를 변수에 저장하도록 설정합니다.

- FailTaskIfReturnCodeIsNotSuccessValue - 프로그램이 종료된 후 출력되는 결과값이 SuccessValue에서 정한 값과 다를 경우 해당 작업을 실패로 처리할지를 설정합니다. 일반적으로 프로그램이 정상적으로 종료한 경우에는 0이 출력됩니다.

- SuccessValue - 성공으로 판단할 값을 지정합니다. 기본값으로는 0입니다. 만약 프로그램에서 성공을 나타내는 결과값이 0이 아닌 다른 값인 경우, 이 속성을 변경하면 됩니다.

- TimeOut - 프로그램이 실행될 수 있는 시간(초)을 지정합니다. 0으로 지정하면 시간 제한 없이 프로그램이 완료되거나 오류가 발생할 때까지 계속 수행하게 됩니다.

- TerminateProcessAfterTimeOut - TimeOut 속성에 지정한 제한 시간이 지난 경우 프로그램을 강제로 종료할지를 설정합니다. 이 옵션은 TimeOut이 0이 아닌 경우에만 사용할 수 있습니다.

- WindowStyle - 프로그램이 실행될 때의 창의 모습을 지정합니다.

메일 보내기 작업

메일 보내기 작업은 SMTP를 이용하여 메일을 보낼 수 있는 작업 개체입니다. SQL 2000 DTS에서 **메일 보내기 작업**은 MAPI(Messaging Application Program Interface)를 이용하기 때문에 메일을 보내기 위해서는 서버에 MAPI 설정이 되어 있어야 합니다. SQL 2005 SSIS의 **메일 보내기 작업**은 SMTP를 이용하기 때문에 별도로 메일과 관련된 설정이나 아웃룩과 같은 프로그램의 설치가 필요 없으며 다른 서버의 SMTP 서비스를 이용할 수도 있습니다.

메일 보내기 작업에서는 **연결 관리자** 내의 **SMTP 연결**을 사용합니다. **SMTP 연결**은 SMTP 서비스가 운영되는 서버의 이름과 인증 방법만 지정해 주면 됩니다.

메일

- SmtpConnection - SMTP 연결을 지정합니다.
- From - 메일을 보내는 사람의 메일 주소
- To - 메일을 받는 사람의 메일 주소. 받는 사람이 여러 명일 경우, 세미콜론(;)으로 구분하여 입력합니다.
- Cc - 참조 메일 주소.
- BCc - 숨은 참조 메일 주소.
- Subject - 메일 제목
- MessageSourceType - 메일 본문 저장 형태를 지정합니다.
  - 직접 입력 - 직접 텍스트 형식으로 입력합니다.
  - 파일 연결 - 텍스트 형식으로 저장된 파일을 이용합니다.
  - 변수 - 본문 내용이 저장된 변수를 지정합니다.
- MessageSource - 메일 본문 내용
- Priority - 메시지의 우선 순위
- Attachments - 첨부 파일

메일 보내기 작업은 데이터 처리 작업 후 수행된 결과를 통보하거나 생성된 파일이나 보고서 등을 보내는 작업 등을 수행할 수 있습니다.

메일 내용(MessageSource)은 직접 변수의 내용을 사용하도록 설정하여 동적으로 지정할 수 있지만, 메일 제목(Subject)이나 첨부 파일(Attachments)은 **메일** 탭에서 동적으로 설정할 수 없습니다. 예를 들어, 메일의 제목을 "결과 보고서 2007년 1월 10일"로 지정하거나 첨부 파일을 "D:₩Reports₩Report_20070110.xls" 과 같은 형태로 지정하고자 할 경우, **메일 보내기 작업** 내에 있는 **식** 탭에서 **Expressions**에서 Subject나 Attachments에 대해 식을 정의하여 구현할 수 있습니다.

메시지 큐 작업

메시지 큐 작업은 MSMQ(Microsoft Message Queuing) 서비스를 이용하여 메시지를 주고 받을 수 있는 기능입니다. 메시지 큐를 이용하면 비 동기적으로 서로 다른 작업 간에 메시지 또는 파일 등을 주고 받을 수 있습니다. 예를 들어 새벽에 서버에서 처리된 전일 매출 보고서 파일을 마케팅 부서의 담당자들에게 전달해야 할 경우, 담당자의 PC가 꺼져 있어서 보내는 시점에 바로 수신을 못하는 경우라도 서버의 데이터 처리 프로세스에서는 MSMQ에 해당 파일을 보내어 저장시켜 놓을 수 있습니다. 아침에 담당자들이 PC를 켜서 파일을 수신할 수 있는 상황이 되면 MSMQ에 저장되어 있는 파일을 받아볼 수 있습니다. 다른 예로, 한 시스템에서 여러 패키지들이 수행될 경우 먼저 수행되어야 할 패키지가 작업이 완료되면 MSMQ에 작업이 완료되었다라는 메시지를 보냅니다. 다른 패키지에서는 MSMQ에 완료 메시지가 생성된 것을 확인한 후 작업을 진행하도록 할 수 있습니다.

메시지를 보내거나 받을 때, 메시지 큐 작업은 **데이터 파일**, **문자열**, **변수에 대한 문자열 메시지**, 변수 중 하나의 유형을 사용합니다. **변수에 대한 문자열 메시지**는 메시지를 받을 때만 사용할 수 있습니다. 메시지 큐 작업을 수행하기 위해서는 Integration Services 서비스가 설치되어 있어야 합니다.

메시지 큐 작업은 **연결 관리자**의 MSMQ **연결**을 사용합니다. MSMQ **연결**에서는 MSMQ의 **경로**를 지정합니다.

일반

- Name - 작업의 이름을 지정합니다.
- Description - 작업의 설명을 지정합니다.
- Use2000Format - MSMQ 2000 형식을 사용할지를 설정합니다.
- MSMQConnection - **메시지 큐 작업**을 위한 MSMQ **연결**을 지정합니다.
- Message - **메시지 큐 작업**에서 메시지를 보내거나 받을지를 지정합니다.

보내기

- UseEncryption - 메시지를 암호화하여 보낼지를 설정합니다.
- EncryptionAlgorithm - 암호화하여 보낼 경우, 암호화 할 알고리즘을 지정합니다.
- MessageType - 보낼 메시지의 유형을 설정합니다.
  - **데이터 파일 메시지** - 파일 형태의 메시지를 보냅니다.
  - **변수 메시지** - 변수에 저장된 내용을 메시지로 보냅니다.
  - **문자열 메시지** - 사용자가 입력한 문자열을 메시지로 보냅니다.

받기

- RemoveFromMessageQueue - 메시지를 받은 후 큐에서 제거할지를 설정합니다.
- ErrorIfMessageTimeOut - 메시지 제한시간이 초과할 경우, 작업을 실패로 처리할지를 설정합니다.
- TimeoutAfter - ErrorIfMessageTimeOut이 True인 경우, Timeout 시간(초)을 지정합니다.
- MessageType - 받을 메시지의 유형을 설정합니다.
  - 데이터 파일 메시지 - 메시지가 파일 형태로 저장됩니다.
  - 변수 메시지 - 메시지가 변수에 저장됩니다.
  - 문자열 메시지 - 메시지 큐에서 받은 메시지가 StringMessage에 지정한 문자열과 동일한지 비교합니다.
  - 변수에 대한 문자열 메시지 - 문자열 메시지로 전송되는 내용을 변수에 저장합니다.

- SaveFileAs - MessageType이 **데이터 파일 메시지**인 경우 나타나며, 메시지를 저장할 파일 위치를 지정합니다.

- Overwrite - MessageType이 **데이터 파일 메시지**인 경우 나타나며, 저장할 위치에 동일한 파일이 있는 경우 덮어쓸지를 설정합니다.

- Filter - 메시지에 대한 필터를 사용할지를 설정합니다. 특정 패키지로부터 온 메시지만 받도록 할 경우, 이 속성값을 True로 지정하고 IdentifierReadOnly의 값을 지정합니다.

- Compare - MessageType이 **문자열 메시지** 또는 **변수에 대한 문자열 메시지** 인 경우 나타나며, **없음** 외의 경우 CompareString에 지정된 값과 메시지의 값에 대한 비교 작업을 수행합니다.

웹 서비스 작업

웹 서비스 작업은 SQL 2005 SSIS에 새롭게 추가된 작업 개체이며, 웹 메서드를 이용하여 웹 서비스에 있는 정보를 읽어오는 기능을 합니다.

예를 들어 다음과 같은 경우에 이용할 수 있습니다.

- 주식 시세를 제공하는 사이트로부터 그날의 주가 정보를 읽어와서 테이블 또는 변수에 저장한 후 다른 작업에서 이를 이용합니다.
- 은행 사이트로부터 환율 정보를 읽어와서 현재 보고서의 단위를 원화에서 달러로 변환합니다.
- 아마존(Amazon.com)과 같은 사이트로부터 새로 업데이트된 도서 목록을 읽어와서 테이블에 저장합니다.

웹 서비스 작업을 수행하기 위해서는 **연결 관리자**의 HTTP **연결**이 필요합니다. HTTP **연결**에서는 서버 URL 정보나 프록시 설정 정보, 웹 서버 액세스를 위한 자격 증명 및 제한 시간 등을 설정합니다.

대부분의 경우에는 자격 증명을 요구하지 않지만, 인트라넷 환경 또는 기업 내에서 보안이 요구되는 환경에서는 자격 증명을 이용하거나 클라이언트 인증서를 이용하여 사용자 인증을 하도록 설정합니다.

연결

- HttpConnection - 웹 서비스 작업을 수행하기 위한 HTTP 연결을 지정합니다.
- WSDLFile - WSDL(Web Services Description Language)는 웹 서비스에서 제공하는 메서드, 메서드에 필요한 입력 매개 변수, 메서드가 반환하는 응답 및 웹 서비스와 통신하는 방법이 나열되어 있는 문서입니다. WSDL이 있는 경우에는 해당 WSDL 파일을 지정할 수 있으며, 파일이 저장될 경로만 지정한 후 아래에 있는 **WSDL 다운로드(D)** 버튼을 눌러 다운로드 받을 수도 있습니다.
- OverwriteWSDLFile - WSDL 파일을 다운로드 할 때 기존 파일이 있으면 겹쳐 쓸지를 설정합니다.

일반

- Name - 작업 이름을 지정합니다.
- Description - 작업에 대한 설명을 지정합니다.

입력

- Service - 목록에서 웹 메서드를 실행하는 데 사용할 웹 서비스를 지정합니다.
- Method - 목록에서 실행할 작업에 사용할 웹 메서드를 지정합니다.
- WebMethodDocumentation - 웹 메서드에 대한 설명을 입력하거나 찾아보기(...) 버튼을 클릭하여 웹 메서드 설명서 대화 상자에 설명을 입력합니다.
  - 이름 - 웹 메서드에 대한 입력의 이름을 나타냅니다.
  - 유형 - 입력 데이터의 유형을 나타냅니다.
  - 값 - 입력되는 값을 설정합니다.
  - 변수 - 입력되는 값이 변수에 저장된 경우, 변수 체크박스를 선택한 후 값에서 해당 변수를 지정합니다.

출력

- OutputType - 출력 결과를 저장할 형태를 지정합니다. 파일 연결을 사용할 경우에는 File 속성에서 파일 연결을 지정하며, 변수를 사용할 경우에는 Variable 속성에서 저장할 변수를 지정합니다.

[따라하기] 웹 서비스 작업

본 예제에서는 **웹 서비스 작업**을 이용하여 실시간 환율 정보를 웹 사이트에서 읽어온 후 이를 결과 파일에 저장하는 예제를 구현합니다.

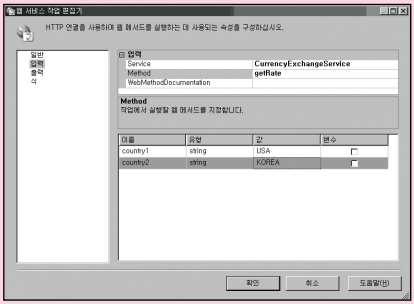
1. 빈 패키지 파일을 하나 추가합니다.
2. 도구 상자에서 **웹 서비스 작업**을 선택한 후 **제어 흐름** 영역에 추가하고, 이름을 **[환율 출력]**으로 변경합니다.

3. **웹 서비스 작업 편집기**의 HttpConnection 속성에서 〈**새 연결**..〉을 선택한 후, HTTP 연결을 추가합니다.
   HTTP 연결 관리자 편집기에서 **서버 URL**(U)의 값에 다음과 같은 주소를 입력합니다.

   http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl

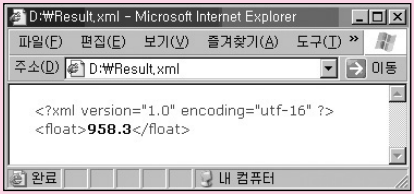
4. WSDLFile 속성에 D:₩Currency.wsdl과 같이 입력한 후, 아래에 있는 **WSDL 다운로드**(D) 버튼을 클릭하여 WSDL 파일을 다운로드 합니다.

5. **입력** 탭에서, Service는 CurrencyExchangeService, Method는 getRate로 설정한 후, 아래의 Method 부분에서 country1의 값은 USA, country2의 값은 KOREA 로 지정합니다.
   (필요에 따라 이 값을 변경하셔도 됩니다.)



6. **출력** 탭에서 OutputType을 **파일 연결**로 설정하고, File 속성에서 〈**새 연결...**〉을 선택하여 결과를 저장할 파일 연결을 지정합니다. **파일 연결 관리자 편집기**에서 **사용 유형(U)**을 **파일 만들기**로 지정하고, **파일(F)**을 D:₩Result.xml로 설정합니다.

7. **확인**을 누른 후, 작업을 실행합니다. 다음과 같은 형태로 출력됩니다.



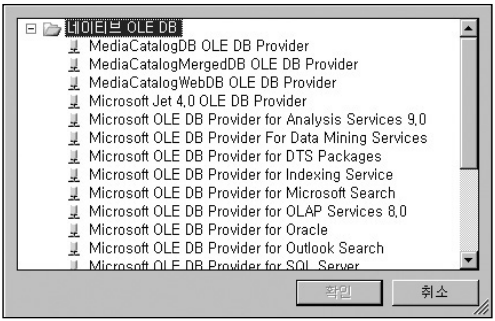
데이터 흐름 요소

데이터 흐름 원본

데이터 흐름 원본은 변환 작업에서 사용할 데이터 원본을 지정하는 부분입니다. 대부분의 원본은 SSIS의 **연결 관리자**에 정의된 연결을 사용합니다. 연결과 관련된 정보를 **연결 관리자**에서 관리하도록 분리한 점은 각각의 데이터 흐름마다 연결을 따로 만들어 주는 대신 하나의 데이터 원본을 공유해서 사용할 수 있는 장점이 있으며, 또한 원본 소스가 변경될 때 **연결 관리자**에 있는 연결 의 정보만 변경해 주면 되는 장점이 있습니다. 데이터 흐름 영역에서는 6개의 데이터 원본 유형과 **스크립트 구성 요소**를 이용한 원본이 있습니다.

OLE DB 원본

가장 일반적인 데이터 원본 형태이며 OLE DB 연결을 이용하는 다양한 형태의 데이터를 사용할 수 있습니다.



연결 관리자에서 **OLE DB 연결**을 선택하여 추가한 후 이를 사용하거나 **OLE DB 원본**에서 직접 추가할 수 있습니다.

사용할 **OLE DB 연결**을 지정한 후, **데이터 액세스 모드(A)** 부분에서 데이터의 형태를 지정합니다.

데이터 액세스 모드는 다음과 같습니다.

- **테이블 또는 뷰** - 데이터를 읽어올 테이블이나 뷰의 이름을 지정합니다.
- **테이블 이름 또는 뷰 이름 변수** - 테이블이나 뷰의 이름이 저장된 SSIS 변수를 지정하는 방식입니다.
- **SQL 명령** - 테이블이나 뷰를 지정하는 대신 SQL 쿼리를 입력합니다. 예를 들어,

```
SELECT AddressID, City, PostalCode FROM [Person].[Address]
WHERE StateProvinceID = '79'
```

  형태와 같은 SQL 쿼리를 사용할 수 있습니다.

- **변수를 사용한 SQL 명령** - SSIS 변수에 데이터 원본으로 사용할 SQL 쿼리를 저장한 후 이 변수를 지정하는 방식입니다.

열 탭에서는 원본에서 사용할 열을 지정하며, 필요한 경우 출력되는 열의 이름을 변경할 수도 있습니다. 예를 들어, 테이블에 있는 TransactionID 라는 열을 TID로 변경해서 사용할 수 있습니다.

데이터 원본을 사용할 때에는 반드시 변환 단계에서 사용할 열만 지정하도록 합니다. 모든 열을 지정하는 것이 개발 과정에서는 간단할 수 있지만, 데이터 처리 성능 면에서는 매우 좋지 않습니다. 만약 데이터 처리 과정에서 TransactionID와 ProductID 열만 사용한다면 **사용 가능한 외부 열**에서 해당 열만 선택하도록 합니다.

오류 출력 탭에서는 데이터를 읽어오는 과정에서 오류가 발생할 때 처리할 방법에 대한 설정을 할 수 있습니다. 각 열 단위로 설정이 가능하며, **오류**와 **잘림**에 대한 처리 방법을 설정할 수 있습니다. **오류**는 지정한 데이터의 범위를 벗어나는 큰 값이 입력되거나, 숫자형으로 설정된 열에 문자형 데이터가 입력되는 것과 같은 경우이며, **잘림**은 열의 크기보다 큰 문자열 데이터가 입력되어 데이터의 일부가 잘리는 경우를 말합니다. 오류를 처리하는 방법으로는 오류가 발생한 행 데이터를 무시하는 **오류 무시**가 있으며, 다른 경로로 오류 데이터를 전달하는 **행 리디렉션**, 또는 해당 작업을 실패로 처리하도록 하는 **구성 요소 실패**가 있습니다. 한 번에 여러 행에 대해 동일한 처리방법을 지정하려면 컨트롤 키를 누른 상태로 각 오류 요소를 선택한 후 아래에 있는 **이 값을 선택한 셀에 설정(S)**을 이용하여 설정합니다.

Excel 원본

Excel 원본은 엑셀 파일의 데이터 시트 또는 이름으로 정의된 특정 영역에 있는 테이블 형태의 데이터를 원본으로 사용하는 원본 개체입니다. 연결 관리자에 있는 **Excel 연결**을 이용하며 테이블 또는 뷰를 지정하는 것과 같이 데이터가 있는 시트를 지정할 수도 있으며, 쿼리를 이용하여 데이터를 읽어오도록 지정할 수 있습니다.

플랫 파일 원본

플랫 파일 원본은 행 구분자 및 열 구분자로 구분된 텍스트 파일을 데이터 원본으로 지정하는 원본 개체입니다. 각 열은 쉼표(,)나 탭(Tab)으로 구분되는 경우도 있으며, 고정 폭으로 저장되어 있을 수도 있습니다. 고정 폭 형태의 데이터는 메인프레임 등에서 주로 사용되는 데이터 형태이며 **연결 관리자**의 **플랫 파일 연결**에서 각 열의 폭을 설정할 수 있습니다. **플랫 파일 원본 편집기**에서 연결 관리자에 정의된 **플랫 파일 연결**을 지정한 후, **열** 탭에서 읽어올 열을 지정할 수 있습니다. 열 구분자나 행 구분자, 첫 번째 행 열 머리글 설정 등과 같은 세부적인 사항은 모두 연결 관리자에서 설정하기 때문에 **플랫 파일 원본**에서는 사용할 열을 지정하는 것 외에는 특별히 설정할 사항은 없습니다.

플랫 파일 원본에는 FileNameColumnName이라는 속성이 있습니다. 이 속성에 String형 사용자 변수를 지정하면 플랫 파일 원본에서 사용하는 파일 이름이 변수에 저장됩니다.

원시 파일 원본

원시 파일 원본은 SSIS에서 신속한 데이터 처리를 수행할 수 있도록 최적화된 형태의 텍스트 파일 원본입니다. 원시 파일은 SSIS의 데이터 흐름 대상 중 **원시 파일 대상**을 이용하여 만들 수 있으며 일반적인 형태의 텍스트 파일을 사용할 수는 없습니다. 저장된 데이터에 대한 메타 정보가 파일에 포함된 형태이기 때문에 SSIS에서는 데이터를 읽기 위한 해독 단계를 생략할 수 있기 때문에 **플랫 파일 원본**에 비해 읽는 속도가 빠릅니다. 하지만, SSIS 패키지 외에는 이 데이터 형태를 사용할 수 없다는 단점이 있습니다. 만약, 모든 데이터 처리 프로세스가 SSIS로 수행되며 텍스트 형태로 데이터를 주고 받는 경우이거나 서로 다른 데이터 흐름 작업 간에 데이터를 주고 받는 작업이 필요할 경우 일반 플랫 파일 대신 원시 파일을 이용하는 것이 효과적입니다.

원시 파일 원본은 연결 관리자에서 관리하는 연결을 사용하지 않고 속성에서 직접 해당 파일을 지정합니다. FileName 속성에 파일 이름을 직접 입력하거나 변수에 파일 이름을 저장한 후 이를 사용하도록 지정합니다.

XML 원본

XML 원본은 다양한 형태의 데이터를 읽어올 수 있는 원본 개체입니다. 로컬 서버에 존재하는 파일 형태의 XML 데이터 또는 HTTP나 UNC를 이용한 원격 서버의 XML 데이터를 이용하는 데이터 원본 유형입니다. XML 데이터를 지정한 후에는 XSD(XML Schema Definition) 파일을 지정해야 합니다. XSD 파일은 **XSD 생성** 버튼을 이용하여 생성할 수 있으며 **인라인 스키마**를 사용하는 경우에는 별도로 지정하지 않아도 됩니다.

DataReader 원본

DataReader 원본은 DataReader 객체를 이용하는 원본 유형입니다. DataReader는 데이터베이스로부터 데이터를 스트림 형태로 처리하는데 사용되는 데이터 오브젝트입니다. 메모리 내에서 한번에 하나의 행만 접근하므로 시스템의 부하를 적게 발생시키며 어플리케이션의 성능 또한 높일 수 있습니다. DataReader 오브젝트는 버퍼를 사용하지 않으므로 메모리에 데이터를 캐싱하지 않습니다. 따라서 많은 양의 데이터를 사용해야 할 때 우수한 성능을 나타냅니다.

연결 관리자에서 .NET 공급자 유형의 데이터 연결을 지정한 후, **고급 DataReader 원본 편집기**의 **구성 요소 속성** 탭에 있는 SqlCommand 부분에서 원본으로 사용할 SQL 쿼리를 지정합니다.

데이터 흐름 대상

DataReader 대상

DataReader 대상은 DataReader 오브젝트를 이용하는 대상입니다. 이 대상 개체는 다음과 같은 경우에 유용하게 사용할 수 있습니다.

- SSIS 패키지 수행 후의 결과를 Reporting Services에서 출력하는 경우
- 외부 응용 프로그램에서 SSIS 패키지에 매개 변수를 지정하여 호출한 후, 결과값을 받아오는 경우

DataReader 대상을 이용하면 SSIS 패키지에서 출력되는 결과를 임시 테이블이나 텍스트 파일 등과 같은 중간 저장소를 거치지 않고도 ADO.NET을 사용하는 환경에서 직접 이용할 수 있습니다.

Excel 대상

Excel 대상은 Excel 파일에 결과를 저장할 때 사용되는 대상 개체입니다. 원본에서와 동일한 방식으로 저장할 Excel 시트 또는 **이름**으로 정의된 테이블을 지정하고 매핑 탭에서 저장할 열을 매핑합니다.

OLE DB 대상

OLE DB 원본과 마찬가지로 일반적으로 가장 많이 사용되는 대상 유형입니다. MSSQL 뿐만 아니라 Oracle이나, DB2, SYBASE 등과 같은 이기종 DBMS로 데이터를 저장할 수도 있습니다. **OLE DB 원본**에서와는 달리 여러 속성들을 지정할 수 있습니다.

- **데이터 액세스 모드**
  - [테이블 또는 뷰] - 대상으로 사용할 테이블이나 뷰를 직접 지정합니다.
  - [테이블 또는 뷰 - 빠른 로드] - 대상이 SQL Server 인 경우 나타나는 옵션이며 ID 유지, Null 유지, 테이블 잠금, Check 제약 조건, 일괄 처리당 행 수, 최대 삽입 커밋 크기 등과 같은 추가적인 속성들을 설정할 수 있습니다.
  - [테이블 이름 또는 뷰 이름 변수] - 대상으로 사용할 테이블이나 뷰의 이름을 SSIS의 변수에 저장한 후 이 변수를 지정합니다.
  - [테이블 이름 또는 뷰 이름 변수 - 빠른 로드] - [테이블 이름 또는 뷰 이름 변수]와 동일하며 추가적인 속성들을 설정할 수 있습니다.
  - [SQL 명령] - 대상을 저장할 SQL 쿼리를 입력합니다. 이 쿼리는 SELECT 형태로 저장될 열을 지정하면 됩니다.

- **ID 유지** - Identity 열에 데이터를 저장할 경우, 원본의 ID값이 ID열에 동일하게 저장되도록 지정합니다.

- **Null 유지** - 저장될 열에 기본값이 지정되지 않도록 하며, 빈 열인 경우에는 Null 값을 유지하도록 지정합니다.

- **테이블 잠금** - 테이블에 데이터를 저장하는 동안 테이블 수준 잠금을 보유하도록 지정합니다. 기본적으로 잠금 동작은 해당 테이블의 table lock on bulk load 테이블 옵션에 의해 결정됩니다. 저장 작업이 진행되는 동안에만 잠금을 보유하면 테이블에 대한 잠금 경합이 줄어들고 성능이 향상됩니다.

- **CHECK 제약 조건** - 데이터를 저장할 때 대상 테이블 또는 뷰의 모든 제약 조건을 확인하도록 지정합니다. CHECK 제약 조건 옵션을 지정하지 않으면 모든 CHECK 제약 조건이 무시됩니다. Unique, Primary Key, Foreign Key 또는 Not Null 제약 조건은 항상 적용됩니다.

- **일괄 처리당 행 수** - 일괄 처리의 행 수를 지정합니다.

- **최대 삽입 커밋 크기** - 빠른 로드 작업을 수행하는 동안 커밋을 시도하는 일괄 처리 크기를 지정합니다. 기본값은 0이며 이 경우 모든 행이 처리된 다음 모든 데이터가 단일 일괄 처리로 커밋됩니다.

SQL Server Mobile 대상

Pocket PC와 같은 Mobile 장치에 데이터를 저장하도록 지정하며, DataReader 대상과 설정 방식이 유사합니다. 이 대상을 사용하기 위해서는 **연결 관리자**에서 **SQL Server Mobile** 연결을 추가해야 합니다.

SQL Server 대상

SQL Server에 최적화된 대상 개체이며 SQL Server 2005의 테이블이 대상인 경우 다른 대상 개체보다 성능이 우수합니다. **OLE DB 대상**에서 빠른 로드를 선택한 경우 나타나는 옵션들 외에 추가적으로 다양한 옵션을 설정할 수 있습니다.

- **트리거 실행** - 대상 테이블에 INSERT 트리거가 정의되어 있을 경우 데이터 입력 시점에 이 트리거를 실행할지를 지정합니다.
- **첫 번째 행** - 데이터를 저장할 첫 번째 행을 지정합니다.
- **마지막 행** - 데이터를 저장할 마지막 행을 지정합니다.
- **최대 오류 개수** - 데이터를 저장할 때 오류가 발생되더라도 허용할 개수를 지정합니다.
- **제한 시간** - 데이터를 저장할 때 수행되어야 할 제한 시간을 설정합니다. 데이터를 저장하는 중이더라도 제한 시간을 초과하면 작업이 실패합니다.
- **열 순서 지정** - 저장할 데이터의 데이터 정렬 방법을 지정합니다. 이 옵션은 저장할 데이터를 정렬해서 저장하도록 하는 설정은 아닙니다. 저장될 데이터가 대상 테이블의 클러스터형 인덱스에 따라 정렬되면 저장 작업의 성능이 향상됩니다. 데이터를 다른 순서로 정렬하거나 테이블에 클러스터형 인덱스가 없으면 ORDER 절이 무시됩니다.

열 순서 지정 속성에서 지정된 열 이름은 대상 테이블의 열이어야 합니다. 예를 들어 테이블에 Seq라는 열에 대해 클러스터형 인덱스가 설정되어 있고, 데이터 파일이 Seq 순서대로 정렬되어 있는 경우, SortedData의 속성 값을 Seq로 지정해 주면 성능이 향상될 수 있습니다.

데이터 마이닝 모델 성향 습득

데이터 마이닝 모델 성향 습득 대상은 데이터 마이닝 모델 알고리즘을 통해 대상에서 수신하는 데이터를 전달함으로써 데이터 마이닝 모델의 성향을 습득하도록 설정합니다.

레코드 집합 대상

레코드 집합 대상은 ADO 레코드 셋에 데이터를 저장하도록 설정합니다. ADO 레코드 셋을 지정하는 간단한 방법으로는 SSIS의 Object 변수를 사용하는 것입니다. Object 형 사용자 변수에 결과 집합을 저장한 후, 이를 **스크립트 작업**이나 **Foreach 루프 컨테이너** 등에서 이용할 수 있습니다.

고급 레코드 집합 대상 편집기의 VariableName 속성에 Object형 변수를 지정하고, **입력 열** 탭에서 저장할 열을 설정하면 됩니다. **레코드 집합 대상**은 메모리에 임시로 데이터가 저장되는 방식이기 때문에 패키지 디버깅 작업이나 테스트 작업 등에서도 자주 사용할 수 있습니다.

DataReader **대상**은 다른 어플리케이션이나 패키지와 데이터를 주고받는데 이용하는 대상 개체인 반면, **레코드 집합 대상**은 패키지 내의 서로 다른 데이터 흐름 작업이나 제어 흐름 작업 간에 데이터를 주고받을 때 주로 이용하는 대상 개체입니다.

원시 파일 대상

원시 파일 대상은 데이터 흐름 원본의 **원시 파일 원본**에서 이용될 수 있는 형태의 텍스트 파일로 데이터를 출력하는 대상 개체입니다. 다른 대상 개체와는 달리 연결 관리자의 연결 설정을 이용하지 않고 직접 대상 파일을 지정합니다. 파일 내에 native format 정보가 포함되기

때문에 데이터를 읽어오는 작업의 경우 성능이 우수하지만, SSIS 패키지 외에서는 이 파일 형식을 사용할 수 없다는 단점이 있습니다. 패키지 간 또는 패키지 내의 서로 다른 데이터 흐름간의 데이터를 연계해야 할 경우 유용하게 사용할 수 있는 대상 개체입니다.

차원 처리 및 파티션 처리

Analysis Service의 차원이나 큐브의 파티션을 처리하도록 지정합니다. 처리 방법을 증분, 전체, 데이터만으로 설정할 수 있으며 특정 디멘젼 또는 특정 큐브를 지정할 수 있습니다.

플랫 파일 대상

텍스트 파일에 데이터를 저장할 때 사용되는 대상 개체입니다. **플랫 파일 원본**과 마찬가지로 세부적인 설정은 **연결 관리자**의 **플랫 파일 연결**에서 지정하며 **플랫 파일 대상 편집기**의 **매핑** 탭에서 입력 데이터에 대한 열 매핑만 설정하면 됩니다.

OLE DB 명령

OLE DB 명령은 입력 데이터의 각 행 별로 SQL 명령을 수행하는 작업입니다. 각 행 별로 수행될 SQL 쿼리에는 ? 로 표시되는 입력 매개 변수가 포함되며, **열 매핑** 탭에서 매개 변수와 입력 데이터간의 매핑을 설정합니다. **OLE DB 명령**은 SQL 2000 DTS의 **데이터 기반 쿼리** 작업과 비슷하지만 DTS에 비해 처리되는 방식이 달라졌으며 다양한 옵션이 추가되었습니다.

OLE DB 명령의 편집기는 다음과 같이 4개의 탭으로 구성되어 있습니다.

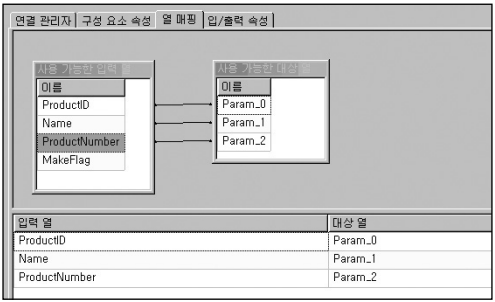
- **연결 관리자** - **OLE DB 명령**을 수행할 연결 정보를 지정하며 **연결 관리자**의 **OLE DB 연결**을 사용합니다.

- **구성 요소 속성** - 입력 데이터를 이용하여 수행할 SQL 명령을 설정합니다. SQL 명령 외에 Timeout 값, 코드 페이지 등을 설정할 수 있으며 변환 작업의 이름이나 설명을

변경할 수도 있습니다. SqlCommand에는 저장 프로시저 또는 Ad-Hoc 쿼리 형태로 설정하며, 쿼리에는 입력 매개 변수로 ? 를 지정합니다.

예) exec sp_insertdata ?,?,?

- **열 매핑** - SqlCommand에서 지정한 쿼리의 매개 변수에 대해 입력 데이터의 열을 매핑합니다. **사용 가능한 입력 열** 부분이 입력되는 데이터의 열이며, **사용 가능한 대상 열** 부분이 매개 변수를 나타냅니다. 매개 변수가 Ad-Hoc 쿼리인 경우, 차례대로 Param_0, Param_1, Param_2, … 등과 같은 형태로 나타나며, 저장 프로시저인 경우 프로시저의 입력 변수 이름이 나타납니다.



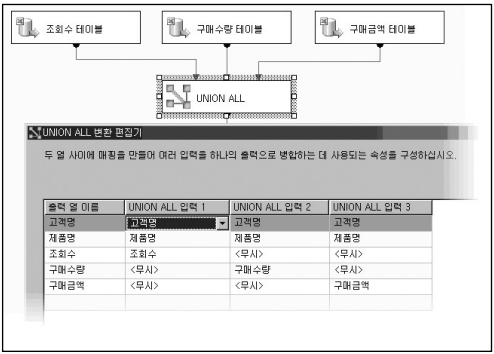
- **입/출력 속성** - 입력 열 및 출력 열에 대한 상세 속성을 조회하거나 각 열의 데이터 유형이나 길이 등의 속성을 변경할 수 있습니다.

UNION ALL, 정렬, 집계 변환

SSIS에는 SQL에서 사용되는 연산과 유사한 형태의 변환 개체들이 많이 있습니다. 대표적으로 UNION ALL, 정렬, 집계, 조인 변환 등이 있습니다. SQL 쿼리는 테이블에 있는 데이터를 대상으로 SQL Server 엔진에서 연산 작업들을 수행합니다. 하지만, SSIS에서는 테이블에 저장된 데이터뿐만 아니라 텍스트 파일이나 엑셀, 다른 DBMS의 데이터에 대해서도 수행할 수 있습니다. 예를 들어, 여러 개의 텍스트 파일 및 엑셀 파일에서 읽은 데이터를 임시 저장 단계 없이 직접 UNION ALL 변환을 이용하여 하나로 합칠 수 있으며, 특정 열로 정렬시킨 후 출력할 수도 있습니다. 또한 COUNT나 SUM등과 같은 집계 작업도 수행할 수 있습니다. 이러한 연산 작업들은 모두 파이프 라인이라는 SSIS 엔진의 내부 메모리 영역에서 수행되며, SQL Server의 엔진 대신 SSIS 자체 처리 엔진을 사용합니다.

UNION ALL 변환

UNION ALL 변환은 여러 개의 입력 데이터를 하나로 결합하는 변환입니다. 여러 개의 입력에 대해 하나의 출력을 제공합니다.



UNION ALL **입력** 1과 같이 **구매수량**이라는 출력 열에 해당하는 데이터가 없는 경우에는 〈**무시**〉로 설정을 하면 해당 열은 NULL로 출력됩니다. 고객명이나 제품명과 같이 서로 다른 입력에 대해 동일한 출력으로 지정하도록 할 경우, 각 입력 데이터의 유형은 반드시 동일해야 합니다.

정렬

입력 데이터를 오름차순이나 내림차순으로 정렬시키는 변환입니다. SQL 쿼리에서와 마찬가지로 여러 열에 대해서도 정렬이 가능하며 각 열 별로 정렬 형태를 지정할 수 있을 뿐만 아니라 각 열 별로 정렬 옵션을 지정할 수 있습니다. 오름차순이나 내림차순과 같은 설정뿐만 아니라 대/소문자 무시, 문자 너비 무시 등과 같은 **비교 플래그** 설정도 가능합니다. 또한 다른 변환에서와 마찬가지로 출력되는 열의 이름을 변경할 수도 있습니다.

- **비교 플래그**
  - 대/소문자 무시 - 대소문자 구분 여부를 설정합니다. 이 옵션을 선택하면 "ABC"와 "abc"를 동일하게 인식하여 정렬합니다.
  - 가나 형식 무시 - 두 가지 유형의 일본어 가나 문자인 히라가나와 가타가나에 대한 구분 여부를 설정합니다.
  - 비공백 문자 무시 - 공백 문자와 분음 기호의 구분 여부를 설정합니다. 이 옵션을 선택하면 "ｱ"와 "ａ"를 동일하게 인식합니다.
  - 문자 너비 무시 - 싱글바이트 문자와 더블바이트 문자의 구분 여부를 설정합니다. 이 옵션을 선택하면 "A"와 "Ａ"를 동일하게 인식합니다.
  - 기호 무시 - 글자 문자와 공백 문자, 문장 부호, 통화 및 수학 기호와 같은 기호 문자에 대한 구분 여부를 설정합니다. 이 옵션을 선택하면 "ABC"나 "*ABC", "ABC"를 모두 동일하게 인식합니다.
  - 문장 부호를 기호로 정렬 - 영문자 또는 숫자 앞에 하이픈과 따옴표(')를 제외한 모든 문장 부호에 대한 구분 여부를 설정합니다. 이 옵션을 선택하면 ".ABC"가 "ABC" 앞에 정렬합니다. 단, 데이터의 코드가 949(한글)인 경우에는 특별히 이 옵션을 설정하지 않아도 문장 부호를 기호로 정렬합니다.

- **중복되는 정렬 값이 있는 행 제거**
  입력 데이터 중 정렬을 수행하는 열에 대해 중복되는 열 값이 발생할 경우, 이 값들을 제거할지를 설정합니다. 이 옵션을 선택하면 동일한 정렬 키 열에 대해서는 하나의 행만 출력합니다. 이 기능을 이용하면 SQL 쿼리의 distinct를 이용한 출력과 유사한 형태의 결과를 쉽게 얻을 수 있습니다

집계

입력 데이터에 대해 SUM이나 AVERAGE, COUNT 등과 같은 집계 연산을 수행하여 결과를 출력하는 변환입니다. SQL 쿼리에서 GROUP BY 구문과 수행 방법이 비슷하며 집계를 수행할 열과 집계 연산을 지정합니다. 다른 변환 작업과는 달리 집계 변환에는 고급 설정 기능이 많이 포함되어 있습니다. 집계 작업을 수행할 때 보다 효과적으로 수행하기 위해 배율이나 키 수 등과 같은 성능과 관련된 여러 가지 설정을 할 수 있습니다.

집계

- 연산
  - Group By - 집계 작업을 수행하기 위한 그룹 열입니다. 예를 들어 고객명과 제품명으로 집계를 수행할 경우, [고객명], [제품명]이 그룹 열이 됩니다.
  - Sum - 열의 값에 대한 합계를 계산합니다. 열의 값이 숫자형인 경우에만 설정할 수 있습니다.
  - Average - 열의 값에 대한 평균을 계산합니다. 열의 값이 숫자형인 경우에만 설정할 수 있습니다.
  - Count - 그룹 열에 대한 열의 항목 개수를 계산합니다.
  - Distinct Count - 그룹 열에 대한 열 중 Null 값을 제외한 항목의 고유한 개수를 계산합니다.
  - Minimum - 열의 값 중 최소값을 반환합니다.
  - Maximum - 열의 값 중 최대값을 반환합니다.

SQL 쿼리와는 달리 SSIS에서는 Maximum, Minimum 연산의 경우 문자형에 대해서는 수행할 수 없습니다.

- **비교 플래그** - **정렬** 변환의 비교 플래그 부분을 참고하기 바랍니다.

- **고유 수 배율** - Distinct Count 연산일 때 이 속성을 설정할 수 있으며, 고유한 키의 대략적인 수준을 지정해 줍니다. Distinct Count 연산 수행 시 고유 수의 정도에 따라 메모리를 미리 확보함으로써 연산 수행 성능을 향상시킬 수 있습니다. 기본 값은 Unspecified 입니다.

  Unspecified - 고유 수 배율 속성을 사용하지 않습니다.
  Low - 약 500,000개 정도의 고유 수 정도인 경우
  Medium - 약 5,000,000개 정도의 고유 수 정도인 경우
  High - 약 25,000,000개 정도의 고유 수 정도인 경우

- **고유 키 수** - **고유 수 배율**은 대략적인 고유 수 값을 지정하는 것입니다. 이에 비해 고유 키 수는 정확한 고유 수 값을 아는 경우 설정해주는 부분입니다. 예를 들어, Distinct Count를 수행해서 나올 결과가 대략 9,500,000 정도라는 것을 미리 알고 있다면, **고유 키 수**의 속성값에 이 추정치를 입력하여 집계 연산의 작업에서 필요한 메모리를 미리 확보할 수 있도록 설정할 수 있습니다. **고유 수 배율**과 **고유 키 수**를 모두 지정하면 **고유 키 수**의 속성이 적용됩니다.

집계 탭에서 상단의 **고급(A)** 버튼을 클릭하면 출력 경로에 대한 설정을 할 수 있는 부분이 나타납니다. 집계 연산은 하나의 입력에 대해 여러 개의 여러 개의 집계를 만들 수 있으며, 각각의 집계 작업이 출력됩니다. 예를 들어, 고객명, 제품명, 조회수, 구매수량, 구매금액이 입력 데이터에 대해, 고객명과 제품명을 기준으로 값들의 합계를 구하는 연산 작업(집계 출력 1)과, 고객명에 대한 고유한 제품명 수(Distinct Count)를 계산하는 집계 작업(집계 출력 2)을 수행해야 한다면, 별도의 집계 변환을 추가하지 않고도 하나의 집계 변환에서 두 가지 형태의 집계 작업을 수행할 수 있습니다.

- **집계 이름** - 출력의 이름을 지정합니다.
- **Group By 열** - 집계 작업의 그룹 열입니다.
- **키 소수 자릿수** -고유 수 배율과 비슷한 옵션이며, 키 열로 수행되는 Group By 연산 결과에 대한 추정치를 설정합니다. 예를 들어 Group By 연산에 의해 생성되는 결과 행 수가 대략 6,500,000개 정도인 경우 키 소수 자릿수 속성을 Medium으로 설정함으로써 좀 더 최적화된 성능을 기대할 수 있습니다.

- **키** - Group By 연산 결과 출력되는 건수를 대략 알고 있을 때 이 값을 입력합니다. 키 소수 자릿수 설정과 마찬가지로 값이 설정되면 SSIS의 처리 엔진에서는 이 값을 기준으로 집계 연산에 필요한 메모리를 미리 확보하기 때문에 처리 성능이 향상될 수 있습니다. **키 소수 자릿수**와 **키** 값이 모두 설정되어 있는 경우에는 **키** 값의 설정이 적용됩니다.

고급

고급 탭에서도 역시 키 배율, 키 수, 고유 수 배율, 고유 키 수를 지정할 수 있습니다. 하지만, 이러한 속성을 고급 탭에서 지정하면 변환 작업 전체에 대해 설정이 적용됩니다. 즉, **고급** 탭의 설정은 집계 변환에 포함된 모든 집계 작업 전체에 대한 설정이고, **집계** 탭의 고급 버튼을 눌러 나오는 화면에서 지정하는 것은 각 집계 작업에 대한 설정이며, **집계** 탭 하단에서 지정하는 것은 출력에 포함된 각 열에 대해 지정하는 것입니다.

고급 탭에는 **자동 확장 비율**이라는 속성이 있습니다. 이는 집계 작업을 위한 메모리 사용에 대한 고급 설정으로, 집계 작업 수행 중 추가로 메모리가 필요할 때 늘릴 비율을 지정하는 것입니다. 기본값은 25% 입니다.

[따라하기] UNION ALL, 정렬, 집계, 파생 열 변환

세 개의 서로 다른 입력 데이터에 대해 **UNION ALL 변환**과 **정렬**, **집계**, **파생 열** 변환을 이용하여 데이터를 병합하는 예제를 구현합니다.

입력 데이터는 고객명과 제품명이 키 열인 조회수 데이터와 구매수량, 구매금액 데이터입니다.

1. 메모장이나 텍스트 편집기를 이용하여 다음과 같은 데이터 파일을 생성합니다.

D:₩조회수.txt

고객명	제품명	조회수
DS Han	TV	2
HK Jeon	TV	3
HS Jang	냉장고	1
SH Ha	세탁기	3
SC Lee	냉장고	5
MB Lee	세탁기	3
DM Han	세탁기	4
JY Kim	TV	2
JH Choi	세탁기	2
JH Seo	세탁기	1
HC Jun	냉장고	1
SM Lee	세탁기	1
MS Kim	세탁기	1
CY Kim	TV	1
JH Choi	냉장고	1

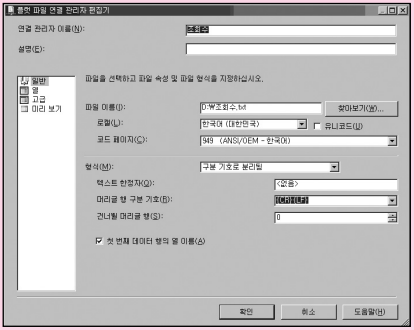
D:₩구매수량.txt

고객명	제품명	구매수량
DS Han	TV	1
HK Jeon	TV	2
HS Jang	냉장고	1
SH Ha	세탁기	1
SC Lee	냉장고	3
MB Lee	세탁기	2
DM Han	세탁기	1
JY Kim	TV	2
JH Choi	세탁기	1

D:₩구매금액.txt

고객명	제품명	구매금액
DS Han	TV	2500000
HK Jeon	TV	4800000
HS Jang	냉장고	1000000
SH Ha	세탁기	700000
SC Lee	냉장고	2750000
MB Lee	세탁기	1350000
DM Han	세탁기	700000
JY Kim	TV	2400000
JH Choi	세탁기	710000

2. 빈 패키지 파일을 추가한 후, **연결 관리자**에서 **플랫 파일 연결**을 하나 추가합니다. **플랫 파일 연결 관리자 편집기**의 **일반** 탭에서 **연결 관리자 이름(N)**을 **[조회수]**로 지정하며, **파일 이름(I)**은 위에서 저장한 파일의 경로로 지정합니다. 아래 부분에 있는 **첫 번째 데이터 행의 열 이름(A)** 옵션을 선택합니다.



3. **열** 탭을 클릭하여 입력 데이터가 정상으로 나타나는지 확인한 후, **고급** 탭을 클릭하여 열의 유형과 길이를 다음과 같이 설정합니다.

고객명 - 문자열(DT_STR), 길이 8
제품명 - 문자열(DT_STR), 길이 6
조회수 - 부호 없는 4바이트 정수(DT_I4)

4. 동일한 방식으로 구매수량.txt, 구매금액.txt에 대해서도 **플랫 파일 연결**을 추가한 후 다음과 같이 설정합니다.

구매수량.txt

고객명 - 문자열(DT_STR), 길이 8
제품명 - 문자열(DT_STR), 길이 6
구매수량 - 부호 없는 4바이트 정수(DT_I4)

구매금액.txt

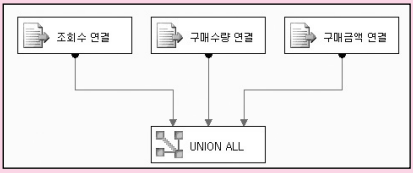
고객명 - 문자열(DT_STR), 길이 8
제품명 - 문자열(DT_STR), 길이 6
구매금액 - 10진수(DT_DECIMAL)

5. 도구 상자에서 **데이터 흐름 작업**을 선택하여 **제어 흐름** 영역에 추가합니다.
6. 이제 앞 단계에서 추가한 각 연결에 대한 원본을 지정하는 작업을 수행합니다. **데이터 흐름 작업**을 더블 클릭하여 **데이터 흐름** 영역으로 전환한 후, 도구 상자에서 **플랫 파일 원본**을 추가합니다. 추가한 **플랫 파일 원본**을 더블 클릭한 후, **플랫 파일 연결 관리자(C)**를 **조회수** 연결 설정으로 변경합니다.

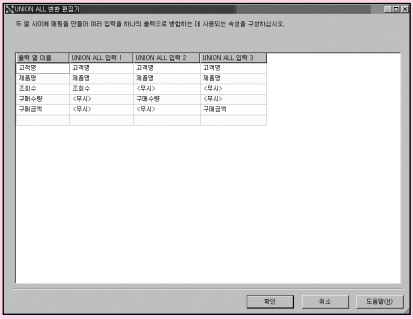
7. **열** 탭을 클릭하여 열 정보를 확인한 후, **확인** 버튼을 눌러 **플랫 파일 원본 편집기**를 닫습니다.

8. 이와 동일한 방식으로 두 개의 플랫 파일 원본을 더 추가하여 **구매수량**과 **구매금액** 연결에 대해서도 설정합니다.
(본 예제에서는 각각의 연결에 대해 "조회수 연결", "구매수량 연결", "구매금액 연결"로 이름을 변경하였습니다.)

9. 도구 상자에서 UNION ALL 변환을 선택한 후, **데이터 흐름** 영역에 추가하고 각 연결의 녹색 선을 UNION ALL 변환에 연결합니다.

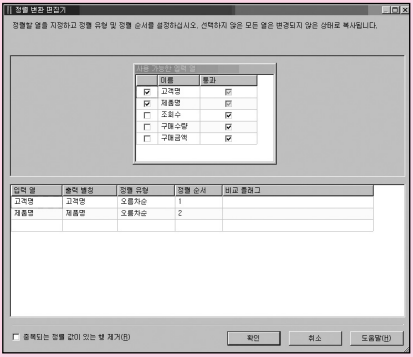


10. UNION ALL 변환을 더블 클릭한 후, 다음 그림과 같이 **구매수량**, **구매금액** 열을 추가합니다. 기본적으로 UNION ALL 변환은 첫 번째로 연결된 입력(UNION ALL 입력 1)에 대한 열 정보에 대해서만 자동 설정이 되기 때문에 본 예제에서와 같이 첫 번째 데이터인 **조회수**.txt 파일에는 없는 구매수량과 구매금액 열에 대해서는 추가 작업을 수행해야 합니다.



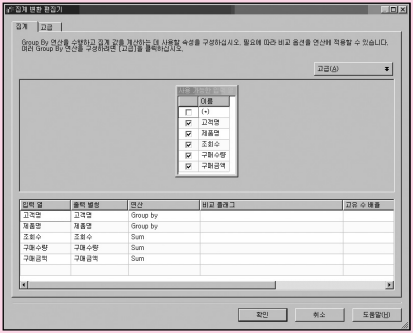
11. **확인** 버튼을 눌러 UNION ALL **변환 편집기**를 닫은 후, 도구 상자에서 **정렬** 변환을 선택하여 **데이터 흐름** 영역에 추가하고, UNION ALL 변환의 녹색 선과 연결합니다.

12. **정렬** 변환을 더블 클릭한 후, 다음 그림과 같이 **고객명**과 **제품명** 열에 대한 오름차순 정렬을 설정하고, **확인** 버튼을 눌러 **정렬 변환 편집기**를 닫습니다.



13. 도구 상자에서 **집계** 변환을 선택한 후, **데이터 흐름** 영역에 추가하고 **정렬** 변환의 녹색 선과 연결합니다.

14. **집계** 변환을 더블 클릭한 후, 편집기의 **사용 가능한 입력 열** 부분에서 (*)을 제외한 모든 열을 선택하고 아래와 같이 각 열에 대한 연산 방식을 설정합니다. 고객명과 제품명을 이용하여 조회수, 구매수량, 구매금액 데이터에 대한 GROUP BY 연산을 수행하는 단계입니다.



15. 도구 상자에서 **파생 열** 변환을 선택하여 **데이터 흐름** 영역에 추가한 후, **집계** 변환의 녹색 선과 연결합니다.

16. **파생 열** 변환을 더블 클릭하여 편집기를 연 다음, 다음과 같이 설정합니다. **집계** 변환을 수행한 결과가 NULL인 경우, 0으로 변경하는 작업을 수행하는 것입니다.

파생 열 이름	파생 열	식
조회수	바꾸기 '조회수'	ISNULL([조회수]) ? 0 : [조회수]
구매수량	바꾸기 '구매수량'	ISNULL([구매수량]) ? 0 : [구매수량]
구매금액	바꾸기 '구매금액'	ISNULL([구매금액]) ? 0 : [구매금액]

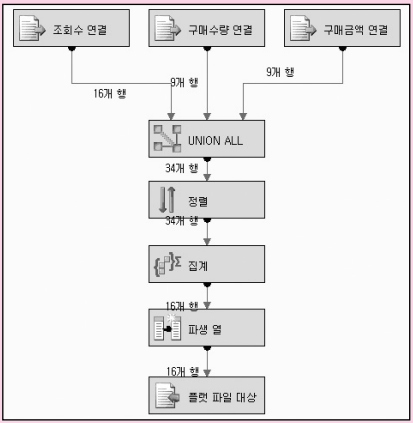
17. 도구 상자에서 **플랫 파일 대상**을 선택하여 데이터 흐름 영역에 추가한 후, **파생 열** 변환의 녹색 선과 연결합니다.

18. **플랫 파일 대상**을 더블 클릭하여 나타나는 편집기에서 플랫 파일 연결 관리자의 **새로 만들기(N)** 버튼을 클릭하여 결과를 저장할 **플랫 파일 연결**을 추가합니다. 플랫 파일 형식은 **구분 기호로 분리됨**으로 설정한 후, **플랫 파일 연결 관리자 편집기**에서 **연결 관리자 이름(N)**과 **파일 이름(I)**을 적절히 설정합니다.

19. **열** 탭과 **고급** 탭을 차례로 클릭하여 저장될 열의 정보를 확인한 후, **확인** 버튼을 눌러 **플랫 파일 연결 관리자 편집기**를 닫습니다.

20. **플랫 파일 대상 편집기**에서 **매핑** 탭을 눌러, 추가한 연결의 열 정보와 변환 결과 정보에 대한 매핑 정보를 확인합니다. 기본적으로 열 이름이 동일한 경우에는 자동으로 매핑이 이루어집니다.

21. **확인** 버튼을 눌러 **플랫 파일 대상 편집기**를 닫은 후, 패키지를 실행하여 생성되는 결과 파일을 확인합니다.



감사

감사 변환은 패키지가 실행 될 때의 환경이나 상황에 대한 정보를 데이터 흐름에 추가하는 변환 개체입니다. 입력 데이터에 데이터 처리 작업을 수행하는 시간이나, 컴퓨터의 이름, 패키지의 GUID, 버전 등과 같은 정보를 추가하여 출력할 수 있습니다. 예를 들어, A, B, C 라는 세 개의 열로 이루어진 입력 데이터에 수행 시간 정보를 나타내는 열 D를 추가하여 출력할 수 있습니다.

감사 변환은 하나의 입력과 하나의 출력을 가지며 오류 출력은 없습니다.

감사 유형

- **실행 인스턴스 GUID** - 패키지가 수행할 때의 GUID값이며, 이 값은 매번 수행될 때마다 달라집니다.

- **패키지 ID** - 수행되는 패키지의 ID 정보입니다. 패키지 ID는 제어 흐름 영역에서 속성 창의 ID 부분에서 확인할 수 있습니다.

- **패키지 이름** - 수행되는 패키지의 이름 정보입니다.

- **버전 ID** - 패키지의 버전 정보입니다. 패키지 버전은 패키지 내의 변경 사항이 발생된 후, 저장하면 변경됩니다. 만약 패키지에 변경 사항이 없는 경우에는 동일한 값이 유지됩니다.

- **실행 시작 시간** - 패키지가 수행되기 시작한 시간을 나타냅니다. 이 시작 시간은 패키지의 시작 시간이며, 감사 단계가 포함된 변환 작업의 시작 시간은 아닙니다.

- **컴퓨터 이름** - 수행되는 컴퓨터의 이름 정보입니다.

- **사용자 이름** - 패키지를 수행하는 사용자 이름 정보입니다.

- **작업 이름** - 감사 변환이 포함되어 있는 데이터 흐름 작업의 이름 정보입니다.

- **작업 ID** - 감사 변환이 포함되어 있는 데이터 흐름의 ID 정보입니다. 이 정보는 제어 흐름 영역에서 데이터 흐름 작업을 클릭한 후, 속성 창의 ID 부분에서 확인할 수 있습니다.

감사 변환은 데이터가 입력될 때의 상황에 대한 정보를 결과 데이터에 포함시킬 수 있는 기능으로 잘못된 데이터 처리나 데이터에 대한 버전 관리 등을 수행할 때 유용하게 사용할 수 있는 변환 개체입니다.

멀티캐스트, 조건부 분할 변환

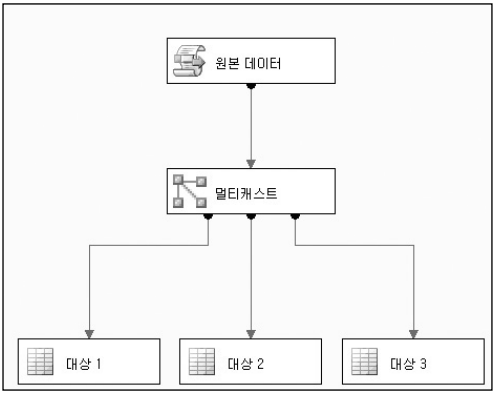
멀티캐스트 및 **조건부 분할** 변환에 대해 설명하기 전에 먼저 다음의 두 경우를 생각해 보겠습니다.

- **하나의 원본 데이터에서 읽은 데이터를 여러 대상에 저장하는 경우**
  예를 들어, Source라는 테이블의 데이터를 10개의 서로 다른 서버에 있는 대상 테이블에 전송해야 할 경우를 생각해 봅시다. SQL의 쿼리 또는 DTS에서는 원본 대 대상의 연결은 1:1입니다. 즉, 10개의 대상 테이블에 데이터를 전송하기 위해서는 원본 데이터를 10번 읽어야 합니다. 하지만, SQL 2005 SSIS의 **멀티캐스트**를 이용하면 원본 데이터를 단 한번만 읽어와도 됩니다. 원본에서 읽어온 데이터를 메모리 내에서 복사하여 여러 경로로 출력하기 때문에 동일한 원본 데이터를 반복해서 읽어오는 부하를 줄일 수 있습니다.

- **원본에서 데이터를 읽어온 후, 조건에 따라 대상을 달리하는 경우**
  원본 데이터에 따라 저장 대상을 달리 해야 하는 경우도 이와 유사한 형태입니다. 예를 들어, 전체 회원 데이터를 지역에 따라 서로 다른 테이블로 나누는 경우를 생각해봅시다. 우선, 원본 데이터 중 지역이 '서울'인 데이터를 검색한 후 결과를 '서울' 테이블에 저장합니다. 동일한 방식으로 '부산', '대구' 등 각 지역별로 검색하여 해당 지역 테이블에 저장합니다. 이 경우에도 역시 동일한 원본 데이터를 여러 번 읽어오는 작업을 수행해야 합니다. SSIS의 **조건부 분할**을 이용한다면 한 번 읽어온 원본 데이터에 대해 메모리 내에서 조건 별로 대상을 달리하여 출력할 수 있습니다.

멀티캐스트

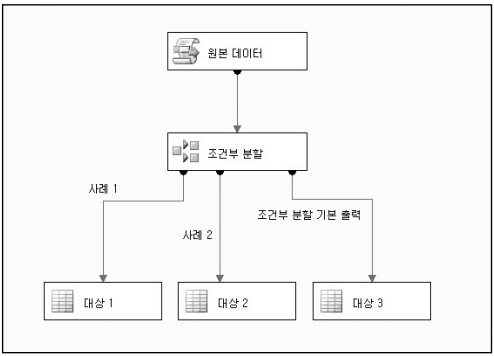
멀티캐스트 변환은 입력 데이터를 여러 개의 출력으로 내보내는 작업을 수행합니다. 데이터 가공이나 유형 변환 등의 작업은 수행하지 않으며, 단순히 데이터를 메모리 내에서 복사하여 출력하는 역할만 수행하기 때문에 원본 데이터와 출력 데이터는 동일합니다. 하나의 원본을 지정한 후 여러 경로로 출력하도록 설정할 수 있습니다.

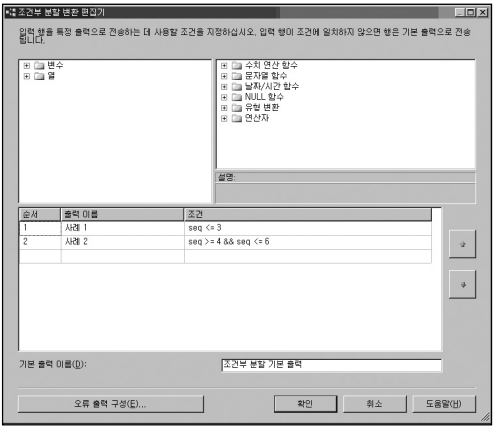


위의 그림에서와 같이, 원본 데이터의 모든 데이터를 대상 1, 대상 2, 대상 3으로 모두 동일하게 보냅니다.

조건부 분할

멀티캐스트 변환과는 달리 입력되는 데이터에 대해 조건에 따라 서로 다른 출력 경로로 데이터를 구분하여 내보내는 변환입니다.





위의 경우, 입력되는 데이터 중, seq열의 값이 3보다 작거나 같은 경우 [사례 1]이라는 출력 명으로 출력되며, 4에서 6 사이인 경우에는 [사례 2]라는 출력으로 데이터가 출력됩니다.
이 조건을 만족하지 않는 데이터들은 기본 출력인 [조건부 분할 기본 출력]으로 데이터가 출력됩니다.

SQL 쿼리의 CASE 구문이나 프로그램 언어에서 SELECT 문 등에서와 같이, 데이터가 상위 조건에 일치하면 해당 경로로 출력되며, 다른 경로로는 출력되지 않습니다. 만약 위의 경우에서 [사례 2]의 조건을 실수로 **seq〉=3** && seq〈=6으로 잘못 입력하였더라도 seq가 3 인 데이터는 [사례 1]의 경로에서 먼저 처리되기 때문에 [사례 2]의 출력으로는 출력되지 않습니다. **기본 출력 이름(D)**은 SQL 쿼리의 CASE 문에서 ELSE와 같은 부분입니다. 즉, 데이터가 위의 조건들에 모두 맞지 않을 경우의 출력을 나타냅니다. 만약 seq의 값이 7 이상인 경우, 상위의 조건들을 모두 만족하지 않기 때문에 기본 출력인 [조건부 분할 기본 출력]으로 출력됩니다.

열 복사, 데이터 변환, 파생 열, 문자표

멀티 캐스트가 입력되는 데이터 전체에 대한 복사 작업이라면, **열 복사** 변환은 입력되는 데이터의 열에 대해서 복사를 하는 변환입니다. **파생 열**이나 **데이터 변환**, **문자표** 변환 역시 입력 데이터에 대해 열 수준의 변경 작업을 수행하는 변환 개체입니다.

모두 입력 데이터의 열에 대한 변환 작업으로 비슷한 유형이지만 다음과 같은 차이가 있습니다.

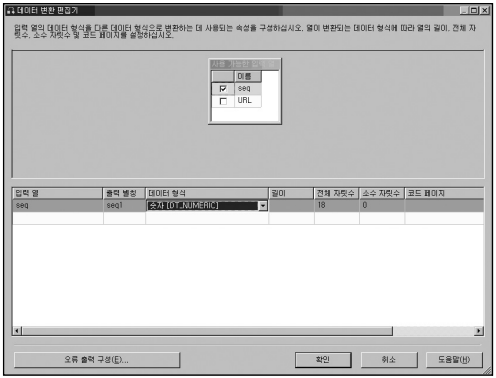
변환 명	작업 내용
열 복사	입력 데이터 중 특정 열을 동일한 형태로 복사
데이터 변환	특정 열에 대하여 데이터 형식, 길이, 자릿수, 코드 페이지 등의 속성을 변경
파생 열	특정 열 또는 여러 열을 이용한 계산된 열이나 문자열 함수 등을 이용한 계산된 열을 추가
문자표	특정 열에 대하여 대/소문자 변환, 바이트 반전 등과 같은 특수 연산을 수행

열 복사

가장 단순한 열 수준의 변환 작업입니다. 입력 데이터 중, 특정 열을 단순히 다른 이름으로 복사해서 출력하는 변환입니다. 만약 원본에서 데이터를 읽어온 후, 여러 변환 과정을 수행할 때, 원본 데이터의 복사본을 그대로 유지하고 싶은 경우에 사용할 수 있습니다.

데이터 변환

입력 데이터 중 특정 열에 대한 데이터 속성을 변경할 때 사용되는 변환입니다.



상단의 사용 가능한 입력 열에서 변경을 원하는 열을 선택한 후, 아래의 속성 변경 부분에서 설정을 변경합니다.

- **출력 별칭** - 변환을 수행한 후, 출력되는 열의 이름을 정합니다. 기존의 열을 바꾸는 기능은 없으며, 새로운 열로 추가만 할 수 있습니다.
- **데이터 형식** - 변경할 데이터의 유형을 설정합니다.
- **길이** - 데이터 형식이 문자열(DT_STR), 유니코드 문자열(DT_WSTR), 바이트 스트림(DT_BYTES)인 경우, 문자열의 길이를 설정합니다.
- **전체 자릿수, 소수 자릿수** - 데이터 형식이 10진수(DT_DECIMAL), 숫자(DT_NUMERIC)인 경우, 숫자 데이터의 전체 자릿수 및 소수 자릿수를 설정합니다.
- **코드 페이지** - 데이터 형식이 문자열(DT_STR), 유니코드 문자열(DT_WSTR), 바이트 스트림(DT_BYTES), 텍스트 스트림(DT_TEXT)인 경우, 데이터의 코드 페이지를 설정합니다.

하단의 **오류 출력** 버튼을 클릭하여, 오류 발생 시 처리할 방법을 설정할 수 있습니다. 예를 들어, 데이터 형식을 숫자로 설정하였지만, 원본 데이터가 "abc"와 같은 문자 데이터가 들어오는 경우, 데이터 변환 작업은 오류를 발생하게 됩니다. 이 경우, 오류 데이터를 다른 경로로 출력하도록 하거나 오류 데이터를 무시, 또는 변환 작업을 실패로 처리하도록 할 수 있습니다.

파생 열

SQL 쿼리에서 계산된 열(Calculated Column)과 같이 기존의 열을 이용하여 새로운 열을 정의하는 변환입니다. 예를 들어, 원본 데이터가 A, B 열로 구성이 된 경우, A+B로 정의되는 새로운 열 C를 추가할 때 이용할 수 있습니다.

파생 열 변환은 **데이터 변환**과 거의 유사합니다. 하지만, 새로운 열을 추가하는 대신 기존 열을 대체하도록 하는 **바꾸기** '**변수명**'으로 설정을 할 수 있으며, 식 부분에서 해당 열에 대한 정의를 할 수 있습니다. 식은 단순히 LEFT, RIGHT 등과 같은 문자열 함수뿐만 아니라, 편집기의

오른쪽에 있는 다양한 함수들을 이용하여 작성할 수 있으며, 열 외에도 SSIS의 사용자 변수나 시스템 변수를 식에 포함시킬 수도 있습니다.

길이, **전체 자릿수**, **소수 자릿수**, **코드 페이지** 등과 같은 속성은 데이터 변환 부분과 동일합니다.

문자표

문자표 변환은 대문자를 소문자로 변환하거나 전자 문자를 반자 문자로 변환하는 등 문자 데이터에 대해 변환 작업을 수행할 때 사용하는 개체입니다.

대상 속성을 **내부 변경**으로 설정하면 기존 열을 대체하게 되며, **새 열**로 설정하면 변환을 적용한 새로운 열을 추가하게 됩니다.

수행할 수 있는 연산 작업은 다음과 같습니다.

- 소문자 - 입력 열의 데이터를 모두 소문자로 변경합니다.
- 대문자 - 입력 열의 데이터를 모두 대문자로 변경합니다.
- 바이트 반전 - 바이트의 순서를 반대로 변경합니다.
- 히라가나 - 입력 열의 데이터가 일본어인 경우 모두 히라가나로 변경합니다.
- 가타카나 - 입력 열의 데이터가 일본어인 경우 모두 가타카나로 변경합니다.
- 반자 - 전자 문자를 반자로 변경합니다.
- 전자 - 반자 문자를 전자로 변경합니다.
- 대소문자 구분 기능 - 대/소문자의 구분 기능을 적용합니다.
- 중국어 간체 - 입력 열의 데이터가 중국어인 경우 모두 간체로 변경합니다.
- 중국어 번체 - 입력 열의 데이터가 중국어인 경우 모두 번체로 변경합니다.

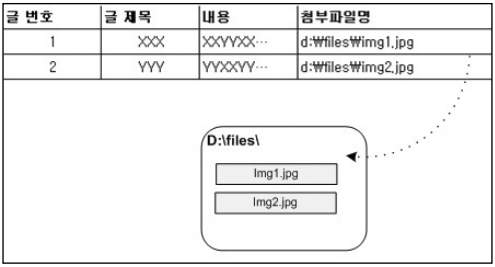
연산 작업들은 동시에 여러 개를 설정할 수 있습니다. 예를 들어 소문자와 반자, 바이트 반전 옵션을 설정할 수 있습니다. 하지만, 소문자와 대문자, 반자와 전자, 히라가나와 가타카나 등과 같이 서로 상반되는 옵션은 동시에 설정할 수 없습니다.

열 가져오기, 열 내보내기

열 가져오기 및 **열 내보내기** 변환은 BLOB(Binary Large Object) 형 데이터를 대상으로 수행하는 변환 작업입니다. 테이블에서 BLOB 데이터 형의 열을 분리하거나 또는 이와는 반대로 별도로 존재하는 텍스트 파일이나 이미지 파일들을 데이터 테이블에 저장할 경우에 사용할 수 있습니다.

열 가져오기

파일을 첨부 할 수 있는 게시판의 경우, 일반적으로 게시물에 포함된 첨부 파일이나 이미지 파일들은 글이 저장되는 테이블 내에 저장되지 않고, 특정 위치에 저장됩니다. 테이블에는 파일이 있는 물리적인 위치 정보만 저장됩니다.

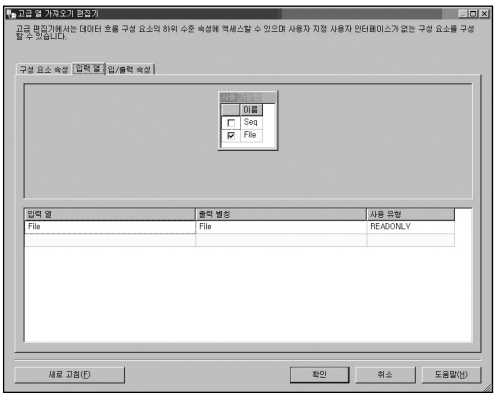


하지만, 이러한 방식과는 달리 첨부 파일이나 이미지 파일들을 SQL에서 제공하는 text, image, varchar, varbinary 등과 같은 형태의 열에 직접 저장할 수도 있습니다.

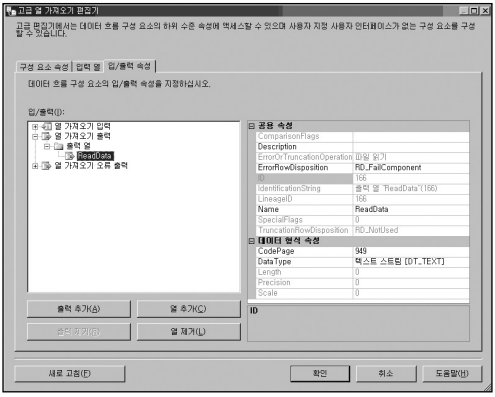
글 번호	글 제목	내용	첨부파일 데이터
1	XXX	XXYYXX…	0xFFD8FFE120EE435…
2	YYY	YYXXXYY…	0xDFE8983FEEDFC12…

테이블 내에 직접 저장할 경우에는 해당 테이블의 크기가 커지는 문제가 있지만 여러 장점들도 있습니다. 파일 정보가 데이터베이스에 저장되기 때문에 데이터베이스 수준의 보안 및 관리 기능을 이용할 수 있으며, 테이블에서 파일이 있는 위치를 읽어와서 해당 파일을 조회하는 대신 테이블에서 직접 데이터를 읽어오기 때문에 처리 성능이 우수한 경우도 있습니다.

열 가져오기 변환은 이러한 형태의 작업을 쉽게 수행할 수 있는 변환 개체입니다. 별도로 존재하는 파일을 읽어와서 텍스트 형 또는 이미지 형 데이터 열로 데이터 흐름에 추가할 수 있습니다. 데이터 흐름에 추가한 후 테이블이나 파일 등에 저장시킬 수 있으며, 다른 변환들을 이용하여 변환 작업을 수행할 수도 있습니다.



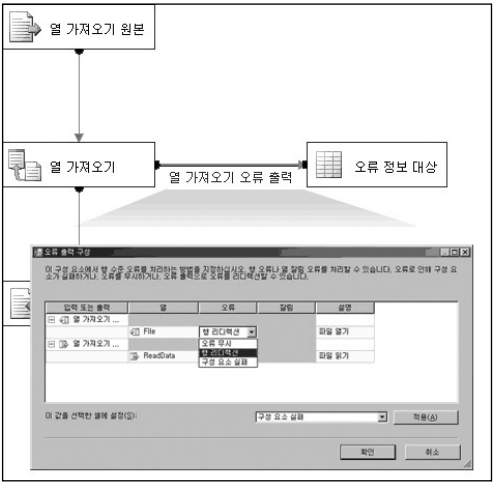
열 가져오기 변환은 **고급 속성 편집기**만을 제공합니다. **입력 열** 탭에서는 데이터 위치를 저장하고 있는 열을 지정합니다. 위의 그림에서 File이라는 열에는 열 가져오기를 수행할 대상인 파일 정보가 저장되어 있습니다.



입/출력 속성 탭에서는 읽어 들인 데이터 파일을 출력할 열을 지정합니다. **입/출력(I)** 부분에서 **열 가져오기 입력** 부분은 파일 정보를 가지고 있는 열에 대한 속성입니다. **열 가져오기 출력**의 **출력 열** 부분에서 아래의 **열 추가(C)** 버튼을 이용하여 읽어온 데이터를 출력할 열을 설정합니다. 위의 그림에서는 열 이름을 ReadData, 데이터 유형을 **텍스트 스트림**(DT_TEXT)으로 지정하였습니다. **열 가져오기 입력**의 **입력 열**을 확장한 후, 파일 이름이 저장된 열의 속성 중 FileDataColumnID의 값에 출력에서 추가한 열의 ID값(위의 그림에서는 166)을 입력 합니다. 입력 받은 열 정보에 대한 출력 열을 매핑 시켜주는 작업입니다.

열 가져오기 변환을 수행할 때, **오류 출력**을 설정하지 않은 상태에서 가져올 파일이 없는 경우에는 작업이 실패하게 됩니다. 예를 들어 파일 정보를 가지고 있는 열에서는 d:₩files ₩img1.jpg 라는 값으로 저장되어 있지만 해당 파일이 존재하지 않는 경우에는 열 가져오기

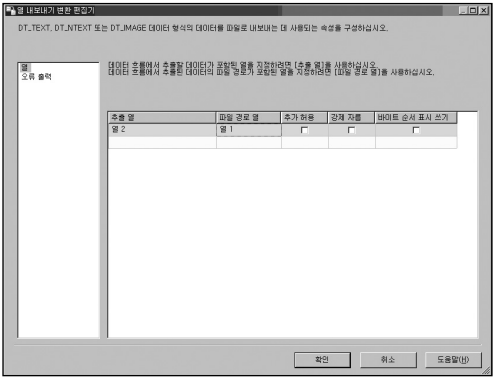
변환 작업은 실패하게 됩니다. 이 경우, 오류 출력 대상을 지정한 후 **열 가져오기** 변환의 오류 출력선(적색 선)을 연결하여 오류 처리 방법을 설정해 주면 파일이 없더라도 실패하지 않고 계속 처리할 수 있습니다. 오류 출력 대상은 단순히 오류 정보를 기록하는 대상으로써, 텍스트 파일 또는 테이블 형태가 될 수도 있으며, 다음 그림과 같이 레코드 집합 대상을 이용하여 메모리 변수에 저장하도록 할 수도 있습니다.



열 내보내기

열 내보내기 변환은 **열 가져오기** 변환과 반대되는 작업을 수행합니다. 입력 데이터 중, 특정 열의 값을 별도의 파일로 출력합니다. 다른 변환 작업과는 달리, 데이터를 저장할 대상 개체를 별도로 지정하는 대신 변환 작업 내의 속성 창에서 직접 설정합니다.

열을 내보낼 수 있는 데이터 형식은 **텍스트 스트림**(DT_TEXT), **유니코드 텍스트 스트림**(DT_NTEXT), **이미지**(DT_IMAGE)형만 가능하며, 파일에 저장할 때의 세부 옵션을 지정할 수 있습니다.



- **추출 열** - 추출할 데이터가 있는 열을 지정합니다. 입력 데이터 중, 텍스트 형이나 이미지 형의 열이 자동으로 나열됩니다.
- **파일 경로 열** - 추출 열의 데이터가 저장될 파일 경로를 포함하고 있는 열을 지정합니다.
- **추가 허용** - 대상 파일이 이미 있는 경우, 추가로 쓸 것인지를 설정합니다.
- **강제 자름** - 데이터를 쓰기 전에 기존의 파일 내용을 삭제하고 새로 쓸 것인지를 설정합니다.
- **바이트 순서 표시 쓰기** - BOM(Byte Order Mark : 바이트 순서 표시 정보)를 파일에 쓸 것인지를 설정합니다. 데이터 형식이 유니코드 텍스트 스트림(DT_NTEXT)이며, 기존 파일에 추가되지 않는 경우에만 BOM을 쓸 수 있습니다.

[참고] BOM(Byte Order Mark) - 바이트 순서 표시

유니코드 텍스트 파일이 Little-Endian인지 Big-Endian인지 아니면 UTF-8 인지를 쉽게 알 수 있도록 하기 위해, 유니코드 파일의 시작 부분에 2~3바이트 정도의 문자열을 추가합니다. 이 문자열은 보이지는 않으며, 헥사 에디터(Hex Editor)를 이용해서 확인할 수 있습니다.

Little-Endian의 BOM	*: FF FE*
Big-Endian의 BOM	*: FE FF*
UTF-8의 BOM	*: EF BB BF 또는 BOM 정보가 없음*

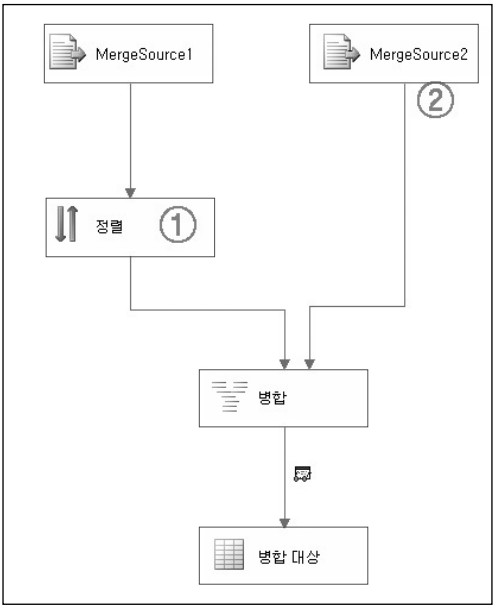
병합

병합 변환은 두 개의 입력 데이터를 병합하는 변환입니다. 입력되는 데이터는 반드시 정렬된 형태이어야 하며 정렬 키를 기준으로 데이터를 병합한 후 다시 정렬된 형태로 데이터가 출력됩니다. **병합** 변환은 UNION ALL 변환과 유사하지만, 다음과 같은 차이가 있습니다.

- **두 개의 입력만 가능합니다.** - UNION ALL 변환은 여러 개의 입력을 사용할 수 있지만, **병합** 변환은 단지 두 개의 입력만 사용할 수 있습니다. 만약, 여러 개의 입력을 사용하고자 할 경우에는 UNION ALL 변환과 **정렬** 변환을 이용해야 합니다.

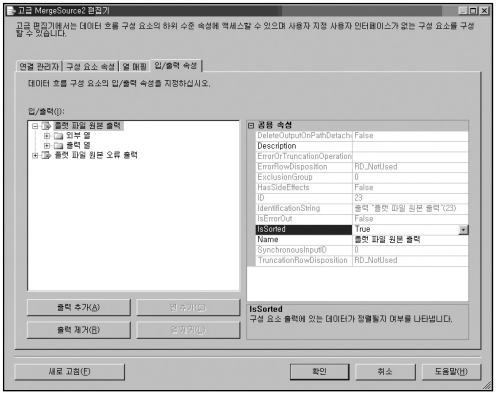
- **입력되는 데이터의 유형 및 길이, 언어 코드가 일치해야 합니다.** - 입력 데이터의 데이터 유형이 일치해야 하는 점은 UNION ALL 변환도 마찬가지이지만, 데이터의 길이 등은 서로 다를 수 있습니다. 예를 들어, 하나의 입력은 DT_STR 10자리이고, 다른 하나의 입력은 DT_STR 30자리인 경우, UNION ALL 변환은 입력의 순서와는 상관없이 두 입력 데이터를 결합할 수 있으며, 출력되는 결과는 입력 데이터 유형 중 가장 큰 유형인 DT_STR 30자리가 됩니다. 하지만, **병합** 변환의 경우에는 입력되는 데이터의 순서에 따라 변환 가능 여부가 결정됩니다. 병합의 첫 번째 입력이 두 번째 입력보다 더 큰 유형인 경우에만 변환이 가능하며, 출력 데이터는 첫 번째 입력 유형을 따릅니다. 첫 번째 입력이 DT_STR 30자리이고, 두 번째 입력이 DT_STR 10자리인 경우에는 수행 가능하지만, 반대로 첫 번째 입력이 DT_STR 10자리이고, 두 번째 입력이 DT_STR 30자리인 경우에는 수행이 안됩니다.

- **입력되는 데이터는 반드시 정렬된 형태이어야 합니다.** - 입력 데이터가 정렬되었다라는 속성을 가져야 합니다. 정렬 속성을 가진다는 것은 입력 데이터가 병합에 연결되기 전에 **정렬 변환**을 거쳐서 **병합 변환**에서 사용할 키 열을 기준으로 정렬이 이루어진 상태이거나, 입력 데이터의 고급 편집기의 출력 속성에서 IsSorted 값을 True로 설정된 상태를 말합니다.

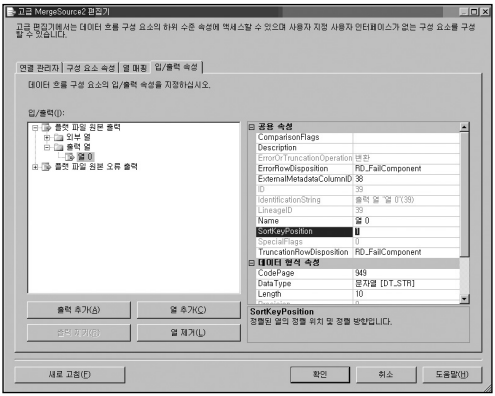


① 입력 데이터가 **정렬** 변환을 거치면 데이터는 정렬 속성을 가지게 됩니다.

② 데이터가 정렬된 것을 보장할 경우, 강제로 정렬된 속성을 지정할 수 있습니다. 위의 그림에서 MergeSource2에 대해 오른쪽 마우스를 클릭하여 나타나는 메뉴 중, **고급 편집기 표시(A)**를 선택하여 **고급 편집기**를 엽니다.

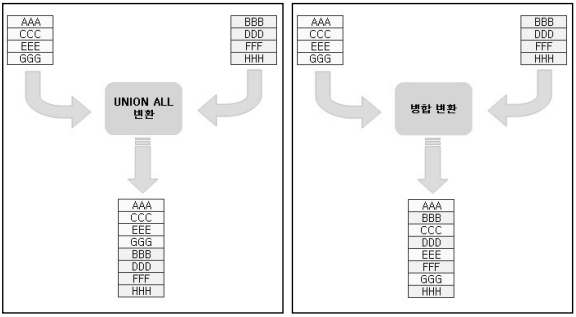


입/출력 속성 탭에서 **원본 출력** 부분을 클릭하면 오른쪽에 출력에 대한 속성 창이 나타납니다. 이 중에서, IsSorted의 속성값을 True로 변경합니다. 그런 다음 **출력 열** 부분을 확장하여 정렬로 설정할 열을 선택한 후, SortKeyPosition의 값을 0에서 1 또는 -1로 설정합니다. 1은 **오름차순**(ASC)으로 정렬되었다라는 것을 나타내며, -1은 **내림차순**(DESC)으로 정렬되었다라는 것을 나타냅니다.



위의 두 가지 설정을 적용하면 **정렬** 변환 없이도 입력 데이터는 정렬 속성을 가지게 되며, 직접 **병합 변환**에 연결할 수 있습니다. IsSorted 속성은 **플랫 파일 원본**뿐만 아니라 다른 모든 원본에 대해서도 존재하는 속성입니다.

- **데이터가 결합되는 순서에 차이가 있습니다.** - UNION ALL 변환과 **병합** 변환은 입력 데이터를 하나로 합치는 기능은 동일합니다. 하지만, UNION ALL 변환은 첫 번째 입력 데이터 뒤에 두 번째 입력 데이터를 덧붙이는 방식이지만, **병합** 변환은 두 입력에 대해 정렬 순위를 판단하여 출력하는 방식입니다.

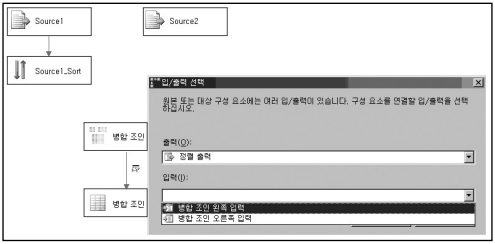


만약 위의 그림에서와 같이 UNION ALL 변환 이후에 정렬을 수행하면 **병합** 변환을 수행한 결과와 동일한 결과를 얻을 수 있습니다.

상황에 따라서 UNION ALL 변환이 유리한지 **병합** 변환이 유리한지 판단하여 사용해야 합니다. 만약 100만 건의 데이터 파일이 10개가 있다고 할 때, 100만 건의 데이터 10개를 각각 정렬한 후 병합 변환을 거쳐 1000만 건의 정렬된 데이터를 생성하는 방식과, UNION ALL을 이용하여 1000만 건의 데이터로 만든 후 이를 정렬하는 방식을 비교할 수 있습니다. 서버의 메모리가 많지 않을 경우, 후자의 방법과 같이 1000만 건의 데이터를 한 번에 정렬하는 것 보다는 10개의 100만 건 데이터로 나누어서 정렬하는 전자의 방법이 더 유리할 수 있습니다. 대신, 병합 변환을 9회 (5 + 2 + 1 + 1) 수행해야 하는 점을 고려해야 합니다.

병합 조인 변환

병합 조인 변환은 병합 변환과 마찬가지로 두 개의 정렬된 입력 데이터에 대해 조인 연산을 수행하는 변환 작업입니다. SQL 쿼리에서 JOIN 문을 이용하는 것과 같이 두 입력 데이터에 대해 조인 연산을 수행합니다. 수행할 수 있는 조인은 **내부 조인**(Inner Join), **왼쪽 우선 조인**(Left Outer Join), **완전 조인**(Full Outer Join)입니다. 입력 데이터가 정렬되어야 하며, 단지 두 개의 입력에 대해서만 조인 연산을 수행할 수 있다는 점에서 SQL 쿼리를 사용하는 것에 비해 비효율적일 수 있습니다. 하지만, 입력 데이터가 SQL 테이블 데이터가 아니거나 또는 서로 다른 유형의 데이터인 경우라도 임시 저장 테이블로 로딩한 후 SQL 쿼리를 이용하여 조인 연산을 수행하는 대신 직접 데이터를 읽어온 후 바로 조인 연산을 수행할 수 있다는 장점이 있습니다.



입력 데이터를 **병합 조인** 변환으로 연결할 경우, 위의 그림에서와 같이 입력 유형을 설정해야 합니다. **내부 조인**이나 **왼쪽 우선 조인**인 경우, **왼쪽 입력**이 조인 연산의 기준이 되는 입력이 됩니다. 입력되는 데이터는 **병합** 변환에서와 마찬가지로 정렬된 속성을 가지고 있어야 합니다.

두 입력을 지정한 후, **병합 조인 변환 편집기**에서 조인할 열과 출력할 열을 설정합니다. 조인할 열은 정렬이 된 열이어야 하며, **출력 별칭**을 이용하여 출력 열의 이름을 변경할 수 있습니다. 위의 그림에서 순서는 정렬 방식을 나타냅니다. 1은 오름차순을 나타내며, -1은 내림차순을 나타냅니다. **입력 바꾸기(S)** 버튼은 **왼쪽 우선 외부 조인**인 경우에만 활성화 됩니다.

용어 변환

SSIS에서는 입력 데이터에 대해 단어를 추출하여 사전 형식의 데이터 결과를 만들어주는 **용어 추출 변환**과 미리 설정된 용어 참조 데이터의 용어가 입력된 데이터에서 얼마나 자주 나타났는지에 대한 분석을 수행하는 **용어 조회 변환** 기능을 제공합니다.

용어 추출 변환

용어 추출 변환은 입력 데이터에서 명사 또는 명사구의 형태로 용어를 추출하는 작업입니다. 텍스트 입력 데이터에 대해 용어 형태 및 빈도 임계값, 최대 용어 길이 등의 옵션을 설정하여 용어를 추출합니다. 이러한 변환 작업은 텍스트 형태의 기사나 e메일 등의 데이터에서 주요 이슈 사항을 추출하여 분류, 분석하는 작업에서 이용할 수 있습니다

※ 이 변환 작업은 한글 데이터는 안되며, 영어 텍스트 데이터만 가능합니다. 또한, 입력 데이터는 유니코드 문자열(DT_WSTR) 또는 유니코드 텍스트 (DT_NTEXT)형만 가능합니다.

용어 추출 변환은 다음과 같은 특징이 있습니다.

1. **관사와 대명사는 추출하지 않습니다.** - 예를 들어 the bicycle, my bicycle 의 데이터는 모두 bicycle로 추출됩니다.

2. **기본적으로 대/소문자를 구분하지 않습니다.** -고급 옵션에서 대/소문자 구분을 선택하지 않는 경우, bicycle 이나 Bicycle, BICYCLE 등을 모두 bicycle로 분류합니다.

3. **단/복수를 동일하게 추출합니다.** - bicycles 와 같이 복수 형태의 단어도 단수로 처리됩니다. 예를 들어 men을 man으로, mice를 mouse로, bicycles를 bicycle로 분류합니다.

4. **변환은 사전에 캐싱된 모드에서 수행됩니다.** - 제외 탭에서 제외할 용어 데이터를 설정한 경우 이 데이터 집합은 전용 메모리 공간에 저장된 후 변환을 수행합니다.

5. **변환은 내부의 자체 알고리즘과 통계 모델을 사용합니다.** - 사용자가 원하는 형태의 결과가 나오지 않을 수 있기 때문에, 고급 탭에 있는 다양한 옵션을 이용하여 출력 형태를 설정해야 합니다.

용어 추출 변환 편집기는 용어 추출, 제외, 고급 탭으로 구성되어 있습니다.

용어 추출

입력 데이터에서 용어를 추출할 열을 선택합니다. 출력 열에서는 추출한 용어를 나타낼 열 이름과 빈도수를 나타내는 점수 열의 이름을 설정합니다.

제외

입력 데이터에서 추출하는 용어 중 제외할 용어가 있을 때 이 탭에서 설정합니다. 제외할 용어를 별도의 테이블에 저장시킨 후 이를 참조하도록 설정할 수 있습니다.

고급

- **용어 유형**
  - 명사 - 단일 명사만을 추출합니다. bicycle, landscape 등이 명사 입니다.
  - 명사구 - 명사구 형태의 데이터만 추출합니다. 명사구는 하나의 명사와 명사 또는 형용사를 포함하는 두 개 이상의 단어입니다. 예를 들어, beautiful bicycle 이 명사구가 됩니다.
  - 명사 및 명사구 - 명사와 명사구 용어 모두를 추출합니다.

- **점수 유형**
  - 빈도 - 점수 유형을 빈도로 설정합니다.
  - TFIDF - 점수 유형을 TF(용어 빈도)와 IDF(역 문서 빈도)의 곱으로 설정합니다. 계산식은 다음과 같습니다.

- **매개 변수**
  - 빈도 임계값 - 단어 또는 구로 추출되기 위한 최소의 임계치 입니다. 예를 들어 이 값을 5로 설정한 경우, 해당 용어가 5회 이상 나와야 출력에 포함될 수 있습니다.
  - 최대 용어 길이 - 추출되는 용어의 최대 길이를 설정합니다. 이 값은 명사구에만 영향을 줍니다.

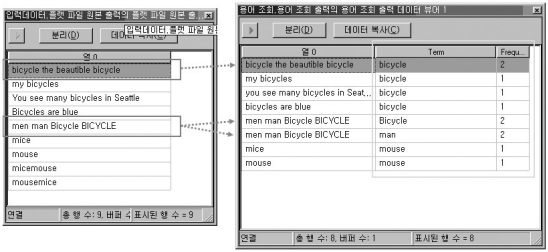
- **옵션**
  - 대/소문자 구분 용어 추출 사용 - 용어 추출 시 대/소문자 구분 여부를 설정합니다. 이 때, 단어에 첫 번째 글자가 대문자인 것은 별도로 구분되지 않습니다. 즉, Bicycle은 bicycle로 분류가 되며, Bicycle 또는 BICYCLE 등과 같은 경우는 별도의 용어로 분류됩니다.

용어 조회 변환

용어 조회 변환은 입력 데이터에서 미리 정의된 참조 데이터의 용어가 얼마나 나타나는지를 조회하는 변환입니다. 이 변환 역시 영문 데이터에 대해서만 정상적으로 수행되며, 입력되는 데이터는 **유니코드 문자형**(DT_WSTR) 또는 **유니코드 텍스트**(DT_NTEXT) 데이터입니다.

참조 데이터로 사용될 수 있는 형태는 OLE DB 연결만 가능하며, SQL Server 2000 또는 SQL Server 2005, Access 데이터베이스의 테이블만 가능합니다.
미리 정의된 용어 목록을 별도의 테이블에 저장시킨 후 변환이 시작될 때 이를 메모리에 캐싱한 후 조회 작업을 수행합니다.

용어 조회 변환 출력 결과는 입력 데이터에 **용어**(Term)와 **빈도**(Frequency)가 추가됩니다. 빈도 계산은 입력 데이터 행 단위로 수행됩니다.



피벗, 피벗 해제 변환

피벗 변환 및 **피벗 해제 변환**을 설명하기 전에 우선 피벗 연산에 대해 살펴보겠습니다.

아이디	성별	연령대	금액
AAA	남	20~24세	100
BBB	남	25~29세	50
CCC	여	35~39세	30
DDD	여	25~29세	60
EEE	여	30~34세	150

위와 같은 데이터에 대해 다음과 같은 형태로 집계할 수 있습니다.

합계 : 금액	연령대				
성별	20~24세	25~29세	30~34세	35~39세	총합계
남	100	50			150
여		60	150	30	240
총합계	100	110	150	30	390

합계 : 금액	성별		
연령대	남	여	총합계
20~24세	100		100
25~29세	50	60	110
30~34세		150	150
35~39세		30	30
총합계	150	240	390

Excel에서는 위의 데이터를 이용하여 쉽게 피벗을 구현할 수 있습니다. 하지만, SQL 쿼리를 이용하는 경우에는 다음과 같은 형태의 복잡한 쿼리를 이용해야 합니다.

```
SELECT [성별],
        SUM(CASE WHEN [연령대] = '20~24세' THEN [금액] ELSE 0 END) AS [20~24세],
        SUM (CASE WHEN [연령대] = '25~29세' THEN [금액] ELSE 0 END) AS [25~29세],
        SUM (CASE WHEN [연령대] = '30~34세' THEN [금액] ELSE 0 END) AS [30~34세],
        SUM (CASE WHEN [연령대] = '35~39세' THEN [금액] ELSE 0 END) AS [35~39세]
FROM [테이블]
GROUP BY [성별]
GO

SELECT [연령대],
        SUM(CASE WHEN [성별] = '남' THEN [금액] ELSE 0 END) AS [남],
        SUM(CASE WHEN [성별] = '여' THEN [금액] ELSE 0 END) AS [여]
FROM [테이블]
GROUP BY [연령대]
GO
```

SQL Server 2005에서는 PIVOT, UNPIVOT 연산이 추가되어 위의 경우보다는 단순하게 쿼리가 가능합니다.

피벗 변환

SSIS에서는 **피벗 변환**을 이용하여 이와 같은 기능을 구현할 수 있습니다. 테이블에 저장된 데이터뿐만 아니라 다양한 형태의 입력 데이터에 대해서도 피벗 변환을 수행할 수 있습니다.

피벗 변환은 기본적으로 **고급 피벗 편집기**에서 속성을 설정합니다. **입력 열** 탭에서는 입력 데이터의 열 중에서 피벗 연산에서 쓰일 열을 선택합니다.

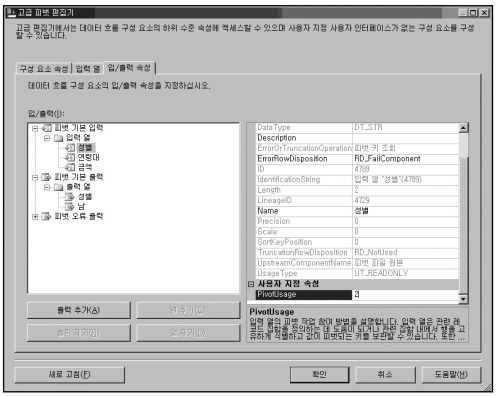
입/출력 속성 탭의 설정을 설명하기 전에 다음과 같은 사항을 참고하기 바랍니다.
피벗 변환에서는 입력 데이터의 열에 대해 피벗 결과의 용도를 설정합니다. 위의 입력 데이터에 대해 다음과 같은 형태의 피벗 결과를 생성한다고 할 때, **[연령대]** 열이 **행 속성**이 되는 열이 됩니다. 그리고, **[성별]** 열의 **[남]**, **[여]**의 값이 **열 속성**이 되며, **[금액]**이 **값 속성**이 됩니다.

합계 : 금액	성별		
연령대	남	여	총합계
20~24세	100		100
25~29세	50	60	110
30~34세		150	150
35~39세		30	30
총합계	150	240	390

열

행

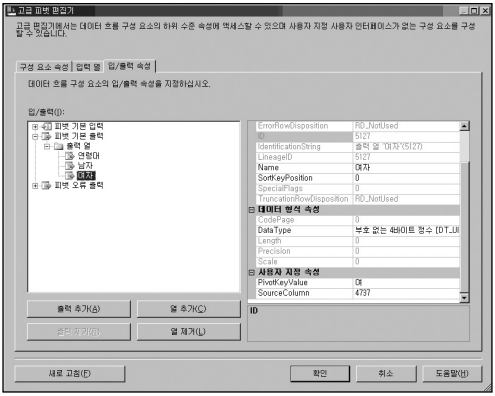
값



입력 열의 각 열에 대해 PivotUsage 값을 지정합니다.

PivotUsage 값	설명
0	열이 피벗 연산에 참여하지 않고 바로 출력됩니다.
1	행 속성을 가지는 열입니다.
2	이 열의 값이 열 속성을 가지게 됩니다.
3	값 속성을 가지는 열입니다.

피벗 기본 출력 부분에서는 행 속성을 가지는 열 및 열 속성을 가지는 열을 생성시키며, 입력 열과의 연결을 설정합니다.



- ComparisonFlags - 그룹핑 작업을 수행할 때 문자열에 대한 비교 처리 방법입니다. **대/소문자 구분**이나 **문자 너비, 기호 무시** 등의 문자열 비교 속성을 설정할 수 있습니다.

- PivotKeyValue - SQL 쿼리의 [CASE WHEN 컬럼명 = '조건값' THEN …] 형태에서 '조건값' 에 해당하는 값을 지정합니다. 위의 그림에서는 출력 열 [여자]는 PivotKeyValue에서 지정한 '여' 라는 값일 경우, 연산을 수행하는 것을 의미합니다.

- SourceColumn - 연산을 수행할 값의 LineageID(계보 ID) 값입니다. 위의 그림에서는 금액 열의 속성 중 LineageID 값을 나타냅니다. 이 값을 지정하면 자동으로 데이터 형식 속성의 DataType 속성이 변경됩니다.

[참고] Excel의 피벗 연산과 SSIS의 피벗 변환과의 차이점

Excel을 이용한 피벗 연산이나 CASE WHEN 문을 이용하는 SQL 쿼리를 이용한 결과와 SSIS의 피벗 변환 결과는 다음과 같은 차이가 있습니다.

아래와 같은 데이터를 살펴봅시다.

아이디	성별	연령	값
User1	남	30세	1
User2	여	25세	1
User3	남	25세	1

피벗 테이블에서 사용할 열과 행의 값이 고유한 경우에는 Excel이나 SSIS의 피벗 변환의 출력 결과는 동일합니다.

성별	25세	30세
남	1	1
여	1	

하지만 일반적으로 입력 데이터에는 동일한 피벗 속성을 가지는 여러 개의 값이 존재합니다.

아이디	성별	연령	값
User1	남	30세	1
User2	여	25세	1
User3	남	25세	1
User4	남	25세	1
User5	여	25세	1

이와 같은 경우, 엑셀 또는 SQL 쿼리를 이용한 피벗 결과는 다음과 같은 형태입니다.

성별	25세	30세
남	2	1
여	2	

하지만, SSIS의 피벗 변환은 이와는 다른 결과가 출력됩니다.

성별	25세	30세
남	1	1
여	1	
남	1	
여	1	

*Excel이나 SQL 쿼리와 같은 형태로 출력하도록 하기 위해서는 **피벗** 변환 후, **집계** 변환을 이용하여 피벗 열에 대해 GROUP BY 연산을 수행해야 동일한 결과를 얻을 수 있습니다. 이러한 점은 SQL 2005의 PIVOT 연산에서도 마찬가지 입니다.*

피벗 해제 변환

피벗 해제 변환은 **피벗 변환**과는 반대로 피벗 형태의 출력 결과를 일반 테이블 형태로 변경하는 변환입니다.

원 데이터 형태

아이디	성별	연령대	금액
AAA	남	20~24세	100
BBB	남	25~29세	50
CCC	여	35~39세	30
DDD	여	25~29세	60
EEE	여	30~34세	150

피벗

피벗 결과 형태

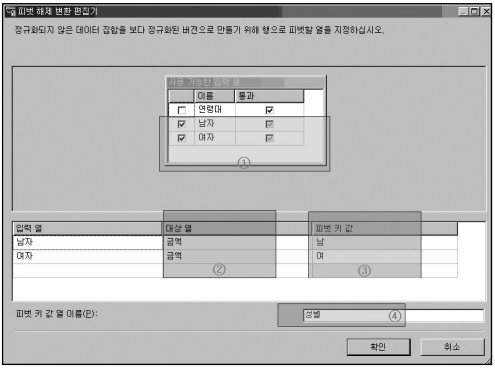
연령대	남	여
20~24세	100	
25~29세	50	60
30~34세		150
35~39세		30

피벗해제

피벗 해제 결과 형태

성별	연령대	금액
남	20~24세	100
남	25~29세	50
여	35~39세	30
여	25~29세	60
여	30~34세	150

피벗 해제 변환은 피벗 키 값에서 데이터 열로 변환될 값과 열 이름을 지정합니다.



① 입력 열 지정 - 원본 데이터에서 피벗 해제를 수행할 열을 선택합니다.

② 대상 열 - 값으로 출력될 열의 이름을 지정합니다.

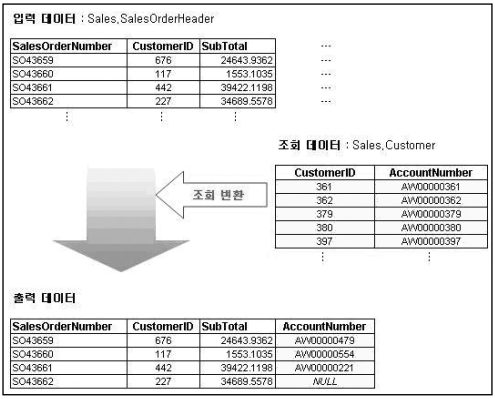
③ 피벗 키 값 - 피벗에서 열 속성의 열 이름을 지정합니다.

④ 피벗 키 값 열 이름 - ③의 값들을 나타낼 열의 이름을 지정합니다.

조회 변환

조회 변환은 입력 데이터에 대해 코드 테이블 또는 디멘젼 테이블과 같은 참조 테이블을 조회할 때 사용하는 변환입니다. **조회 변환**은 **SQL 2000 DTS**의 **데이터 변환 작업**이나 **데이터 기반 쿼리** 작업과 상당히 유사합니다. SQL 2005 SSIS에서는 조회 데이터에 대한 다양한 캐싱 옵션이 추가되었고 변환을 설정하는 방법도 이전의 방식에 비해 간단해졌습니다.

다음과 같은 작업을 생각해 봅시다.



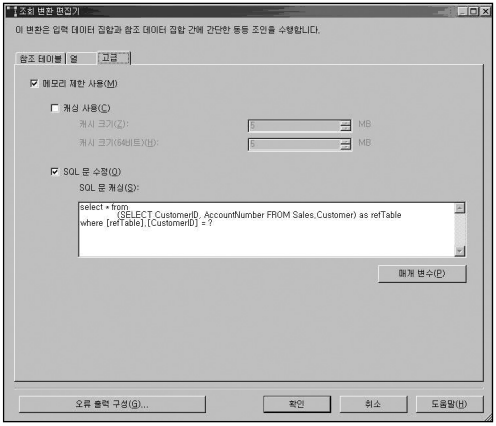
입력 데이터의 CustomerID를 이용하여 조회 테이블인 Sales.Customer 에서 Account Number 값을 찾아낸 후 이를 결합하여 출력하는 작업입니다. **조회 변환**은 조인 연산(INNER JOIN 또는 OUTER JOIN)과 유사합니다.

조회 변환 편집기는 세 개의 탭으로 구성되어 있습니다.

참조 테이블 탭에서는 조회 테이블을 설정합니다. **OLE DB 연결**을 이용하여 조회 테이블 또는 조회에 이용할 쿼리를 지정합니다. 직접 테이블을 지정할 수 있지만, 가급적이면 필요한 열만을 포함하는 SQL 쿼리를 이용하는 것이 성능상 유리합니다.

열 탭에서는 입력 데이터와 참조 데이터 간의 연결을 설정하며, 참조 데이터에서 출력할 열을 지정합니다.

고급 탭에서는 참조 테이블 또는 데이터에 대한 캐싱을 설정합니다.

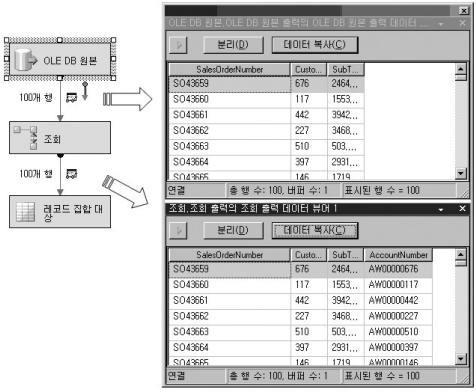


- **메모리 제한 사용(M)**을 체크하지 않으면 전체 캐싱으로 설정되어 참조 테이블 전체를 메모리에 캐싱합니다. 이 경우, 변환 작업 전에 모든 조회 데이터를 메모리로 로딩하는 작업이 수행됩니다.

- **메모리 제한 사용(M)**을 체크한 후, **캐싱 사용(C)**을 체크하지 않으면 캐싱을 하지 않습니다.

- **메모리 제한 사용(M)**을 체크한 후, **캐싱 사용(C)**을 체크하고, 캐시 크기를 설정하면 부분 캐싱을 수행합니다.

SQL 문 캐싱(S)을 설정하면 참조 테이블 전체를 캐싱하는 대신 참조 테이블에서 조회 되는 데이터에 대해서만 캐싱하도록 설정합니다. 이 때, 매개변수를 이용한 SQL 문이 사용되며 이 SQL 문 또한 캐싱이 되어 수행됩니다. 조회 테이블 또는 데이터가 상당히 큰 경우에는 이 옵션을 사용하는 것이 전체 테이블을 캐싱하는 것에 비해 훨씬 유리할 수 있습니다.



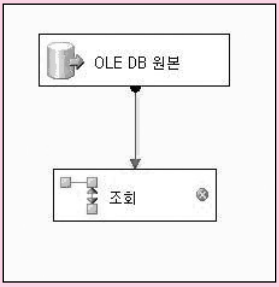
[따라하기] 조회 변환

1. **연결 관리자**에서 OLE DB **연결**을 하나 추가한 후 tempdb로 데이터베이스를 선택합니다.

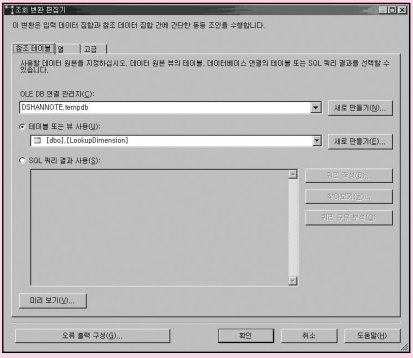
2. 도구 상자에서 **데이터 흐름 작업**을 선택한 후, 제어 흐름 영역에 추가합니다.

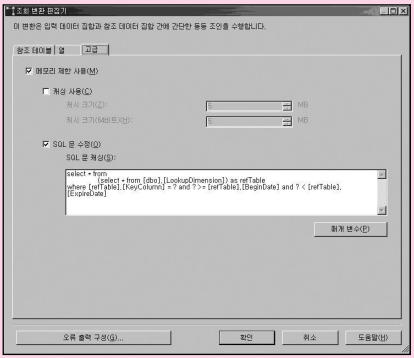
3. **데이터 흐름 영역**에서 **OLE DB 원본**을 추가한 후, 이미 만들어 놓은 데이터 원본 테이블인 SourceTable을 선택합니다.

4. 도구 상자의 **조회 작업**을 선택하여 데이터 흐름 영역에 추가한 후, **OLE DB 원본**과 연결합니다.



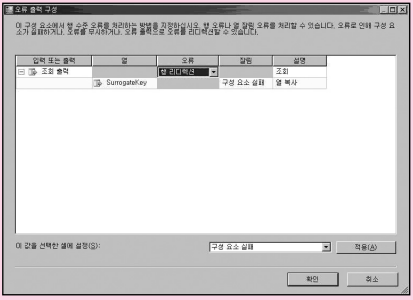
5. **조회** 변환의 속성을 다음과 같이 지정합니다.





아래에 있는 **매개 변수(P)**를 눌러 다음과 같이 매개 변수를 설정합니다.

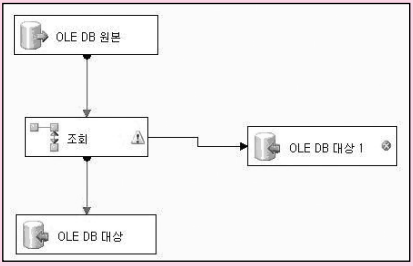
6. 아래에 있는 **오류 출력 구성**을 클릭한 후 다음과 같이 설정합니다.



7. 도구 상자에서 **OLE DB 대상**을 선택하여 추가한 후, **조회**의 녹색선과 연결합니다.

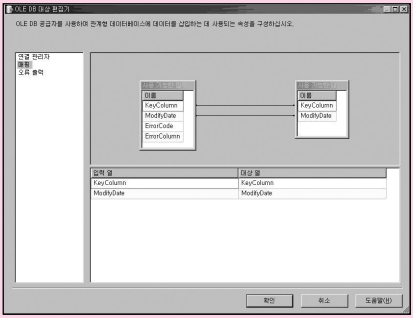
8. **OLE DB 대상**의 속성에서 **대상 테이블**을 [dbo].[TargetTable]로 지정한 후, **매핑** 탭을 눌러 정상적으로 열이 매핑 되었는지 확인합니다.

9. 도구 상자에서 **OLE DB 대상**을 선택하여 추가한 후, **조회** 변환의 적색 선과 연결시킵니다.



이 때, 자동으로 **오류 출력 구성** 창이 나타나며, 이미 6단계에서 오류 지정을 했기 때문에 그냥 **확인**을 누릅니다.

10. **OLE DB 대상1**의 속성에서 대상 테이블을 [dbo].[ErrorTable]로 지정한 후, **매핑** 탭을 클릭하여 정상적으로 열이 매핑되었는지 확인합니다.



11. **확인**을 누른 후, 실행합니다.

12. 만약 매핑이 안된 데이터를 별도로 보관할 필요가 없는 경우, 6단계에서 '**행 리디렉션**' 대신 '**오류 무시**'로 설정하고, 9, 10 단계를 생략합니다.

[따라하기를 수행하기 위한 임시 테이블 생성 스크립트]

```
/*--------------------------------------------------------------------------
        SSIS 조회(Lookup) 작업 따라하기를 위한 사전 테이블 생성 스크립트
--------------------------------------------------------------------------*/

USE TEMPDB
GO
--데이터 테이블 생성 및 데이터 입력
CREATE TABLE dbo,SourceTable (
        KeyColumn VARCHAR(32),
        ModifyDate SMALLDATETIME
)
GO
--정상적으로 매핑될 할 데이터
INSERT INTO dbo,SourceTable (KeyColumn, ModifyDate) VALUES
( 'a' ,' 2006-01-01' )
GO
--디멘젼 값의 유효 시작일 이전의 데이터 : 매핑되어서는 안됨
INSERT INTO dbo,SourceTable (KeyColumn, ModifyDate) VALUES
 ( 'a' ,' 2005-10-01' )
GO
--매핑되는 디멘젼이 없는 데이터: 매핑되어서는 안됨
INSERT INTO dbo,SourceTable (KeyColumn, ModifyDate) VALUES
( 'b' ,' 2006-01-01' )
GO
```

```
--매핑된 데이터가 저장될 테이블
CREATE TABLE dbo,TargetTable (
        KeyColumn VARCHAR(32),
        ModifyDate SMALLDATETIME,
        SurrogateKey INT
)
GO

--매핑 실패한 데이터가 저장될 테이블
CREATE TABLE dbo,ErrorTable (
        KeyColumn VARCHAR(32),
        ModifyDate SMALLDATETIME
)
GO

--디멘젼 테이블 생성 및 데이터 입력
CREATE TABLE dbo,LookupDimension (
        KeyColumn VARCHAR(32),              -- 디멘젼 값
        BeginDate SMALLDATETIME,            -- 디멘젼 값의 유효 시작일
        ExpireDate SMALLDATETIME,           -- 디멘젼 값의 종료일
        SurrogateKey INT                    -- 대리키
)
GO

INSERT INTO dbo,LookupDimension (KeyColumn, BeginDate, ExpireDate,
SurrogateKey)
VALUES ( 'a' ,  '2005-12-11' ,  '2006-02-28' , 1)
```

```
GO
--다음과 같은 결과가 출력되어야 합니다.
--Case1) 디멘젼으로 매핑이 되는 경우만 출력 (TargetData에 저장될 것입니다.)
SELECT
        S.KeyColumn,
        S.ModifyDate,
        D.SurrogateKey
FROM dbo.SourceTable S
LEFT OUTER JOIN dbo.LookupDimension D
    ON S.KeyColumn  =  D.KeyColumn
    AND S.ModifyDate >= D.BeginDate
    AND S.ModifyDate <  D.ExpireDate
WHERE D.SurrogateKey IS NOT NULL
GO

--Case2) 디멘젼으로 매핑이 안되는 데이터를 별도로 출력 (ErrorData에
저장될 것입니다.)
SELECT
        S.KeyColumn,
        S.ModifyDate
FROM dbo.SourceTable S
LEFT OUTER JOIN dbo.LookupDimension D
    ON S.KeyColumn  =  D.KeyColumn
    AND S.ModifyDate >= D.BeginDate
    AND S.ModifyDate <  D.ExpireDate
WHERE D.SurrogateKey IS NULL
GO
```

샘플링 변환

행 샘플링 변환 및 **비율 샘플링 변환**을 이용하여 전체 데이터에서 샘플 데이터를 추출하는 작업을 수행할 수 있습니다. **행 샘플링 변환**은 입력 데이터에서 지정한 행 수만큼 샘플 대상을 선정하는 작업이며, **비율 샘플링 변환**은 입력되는 데이터에 대해 정해진 비율만큼을 대상으로 선정하는 작업입니다.

다음과 같은 경우에 **샘플링 변환**을 이용할 수 있습니다.

1. 전체 회원 데이터 중 특정 수 만큼의 샘플 대상을 추출해야 할 경우
2. 웹 로그 데이터와 같이 대량의 데이터에 대해 표본조사를 수행해야 할 경우

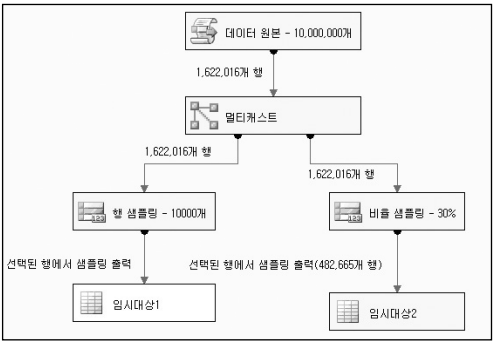
행 샘플링 변환과 **비율 샘플링 변환**은 샘플의 수를 설정하는 방법에서 차이가 있습니다. 예를 들어, 전체 데이터 중 **1000개** 샘플 데이터와 같이 정확한 수의 샘플이 필요할 때에는 **행 샘플링 변환**을 사용하며 원본 데이터의 **10%** 또는 원본 데이터의 **3%** 등과 같은 형태로 비율로 설정할 때에는 **비율 샘플링 변환**을 이용하면 됩니다.

- **샘플 출력 이름** - 입력 데이터에 대해 샘플 행 수 또는 행의 백분율만큼의 샘플 데이터를 추출한 후, 출력할 경로의 이름을 지정합니다. 예를 들어 1,000개의 데이터 중 행의 백분율로 30으로 정했다면 "**선택된 행에서 샘플링 출력**" 이라는 경로로 300개의 데이터가 출력되며 "**선택되지 않은 행에서 샘플링 출력**" 이라는 경로로 나머지 700개의 데이터가 출력됩니다.

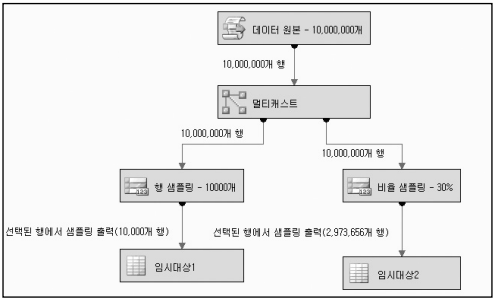
- **선택하지 않은 출력 이름** - 샘플링을 해서 출력시킨 후, 샘플링 되지 않은 데이터를 출력할 경로의 이름을 지정합니다.

- **다음과 같은 임의 초기값 사용** - 이 옵션을 체크하지 않으면 샘플링 작업이 수행될 때마다 매번 다른 샘플이 추출됩니다. 하지만, 이 옵션을 체크하고 초기값(Seed)을 지정하면 동일 데이터에 대해서는 Seed 값에 따라 샘플링 되는 값은 항상 동일합니다. 동일한 샘플 데이터를 계속해서 이용하는 추적 조사(Tracking Research)나 문제가 있는 샘플 데이터에 대한 프로그램 수정 시 이 옵션을 이용할 수 있습니다.

행 샘플링 변환은 입력 데이터를 전부 다 읽은 후 샘플을 추출하여 출력하는데 비해, **비율 샘플링 변환**은 어느 정도의 데이터가 입력되면 동시에 샘플링 한 결과 데이터를 출력합니다. 또한, **행 샘플링 변환**은 변환 편집기에서 지정한 숫자만큼 정확히 샘플을 추출하지만, **비율 샘플링 변환**은 약간 오차가 있습니다. 즉, 10,000,000개의 데이터에 대해 30%라고 지정을 하더라도 정확히 3,000,000개의 데이터가 출력되지는 않습니다.



패키지 수행 시 출력 비교



행 샘플링과 비율 샘플링의 정확도 차이

유사 항목 조회

대량의 데이터 환경에서 데이터 정재(Data Cleansing) 작업은 중요하면서도 수행하기 어려운 작업 형태 중 하나입니다.

SQL 쿼리는 기본적으로 조회 조건과 데이터가 완전히 일치하는 경우에만 결과가 출력됩니다. LIKE 연산자나 문자열 함수 등을 이용하여 어느 정도의 유사 조회를 수행할 수는 있지만 이러한 유사 데이터에 대한 조회 작업은 제한적으로만 이용할 수 있으며 유사성 계산과 같은 작업을 수행하는 데에는 한계가 있습니다.

다음과 같은 예를 살펴보겠습니다.
인터넷 초기에는 회원 가입을 받을 때 사용자들로부터 직접 전체 주소 데이터를 입력하도록 하는 경우가 대부분이었습니다. 요즘은 대부분의 사이트가 우편번호를 이용하여 규격화된 형태로 자동 입력이 되지만, 이전의 방식과 같이 직접 입력하도록 한 경우에는 동일한 주소가 다양한 형태로 입력되는 경우가 많았습니다.

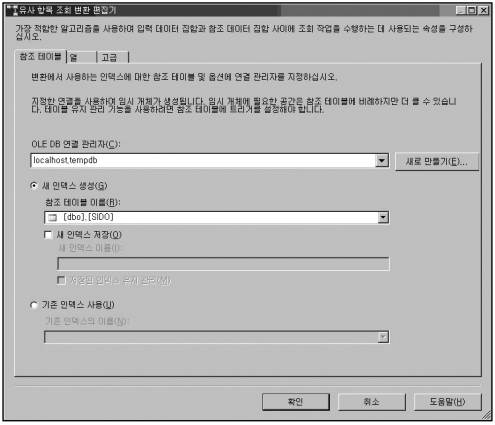
- 서울특별시 강남구 삼성동
- 서울시 강남구 삼성동
- 서울 강남구 삼성동
- 서울 강남 삼성

사이트의 규모가 커지고 데이터의 사용 영역이 많아지면서 정제되지 않은 데이터를 표준화 된 데이터로 변경하는 작업에 대한 요구가 많이 발생되고 있습니다. 이러한 표준화 작업은 데이터 정재 작업의 여러 형태 중 하나입니다.

SQL 2005 SSIS에는 표준화 되지 않은 데이터에 대해 기준 데이터와 가장 유사한 값을 조회하고 조회된 데이터와의 유사도, 신뢰도를 판단해 주는 작업을 수행할 수 있는 **유사 항목 조회** 변환이 있습니다.

유사 항목 조회 변환은 **조회** 변환과 비슷합니다. 조회 변환은 참조 테이블의 비교 항목과 정확히 일치하는 경우에만 출력하지만 **유사 항목 조회** 변환은 정확히 일치하지는 않더라도 유사하다고 판단될 경우 해당 데이터를 출력합니다.

참조 테이블



참조 테이블 탭에서는 조회 대상이 되는 참조 테이블을 지정하며 **유사 항목 조회 변환** 수행 시 사용할 인덱스에 대한 속성을 지정합니다. 이 인덱스는 유사 비교를 수행할 때 이용하는 메타 데이터 테이블을 말합니다.

새 인덱스 저장(O) 옵션을 선택한 후 **새 인덱스 이름(I)**을 지정하면 참조 테이블이 존재하는 데이터베이스에 지정한 이름의 인덱스가 생성됩니다. 여기서 말하는 인덱스란 유사 항목 조회 작업을 수행할 때 필요한 메타 정보를 담고 있는 테이블을 말합니다.
저장된 인덱스 유지 관리 옵션을 선택할 경우 참조 테이블에 인덱스의 정보를 관리할 트리거가 생성됩니다. 이 트리거는 참조 테이블에 insert, update, delete 작업이 수행될 때 인덱스에 자동으로 반영하는 트리거입니다. 입력 데이터가 자주 변경되며 **유사 항목 조회** 작업을 반복해서 수행할 경우 이 옵션을 이용하여 인덱스 테이블을 효과적으로 관리할 수 있습니다.

데이터베이스에는

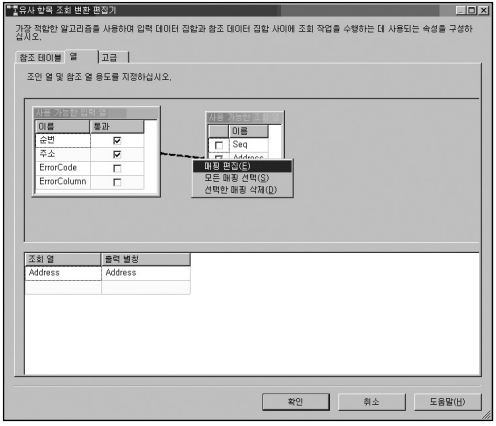
```
tg_DataCleaningMaintenance_PendingDelete__20060908_000000_fabb4233-
bde1-4515-8d09-a89f35023b4e
tg_DataCleaningMaintenance_PendingInsert__20060908_000000_87baf700-
b004-440e-b169-0659e166380d
```

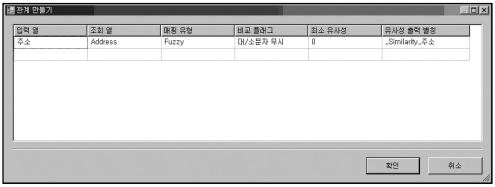
등과 같은 형태의 인덱스가 생성됩니다.

기존 인덱스 사용을 선택하면 이미 생성되어 있는 인덱스를 사용하게 됩니다.

열

열 탭에서는 입력 데이터와 조회 데이터 간의 매핑을 설정합니다. 또한 각 열의 연결에 대해 상세한 매핑 설정을 할 수 있습니다.



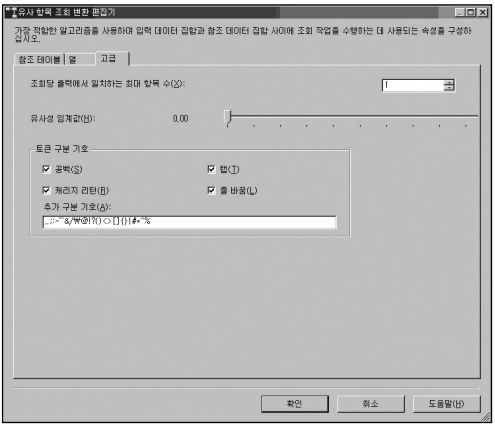


매핑 유형을 Fuzzy 또는 Exact로 설정할 수 있습니다. Fuzzy는 유사 조회를 수행하는 것이며 Exact는 정확히 일치하는 조회를 수행하도록 하는 것입니다. 모든 비교 열이 Exact일 때에는 **조회 변환**과 동일하게 수행됩니다. 예를 들어, A, B, C라는 원본 데이터의 열에 대해 A' , B' , C' 라는 참조 테이블의 열과 **유사 항목 조회**를 수행할 때, A 열과 A' 열은 반드시 일치를 해야 하며, B와 B' , C와 C' 는 어느 정도의 유사성만 있으면 매핑이 되도록 설정할 수 있습니다. 이 때, A와 A' 의 매핑 유형은 Exact로 지정하고 나머지는 Fuzzy로 지정합니다.

비교 플래그에서는 **문자표** 변환 또는 **집계 변환** 등에서와 같이 문자열 비교 작업에 대한 속성을 설정합니다.

최소 유사성에는 해당 열에 대한 매핑 작업 수행 시 매핑을 수행할 최소의 유사성을 지정합니다. 이 값은 0에서 1 사이의 값입니다. 예를 들어, B와 B' 열 간의 매핑에서 최소 유사성을 0.80으로 설정한다면, 유사성이 0.80이상인 경우에만 매핑이 이루어집니다.
유사성 출력 별칭은 해당 열의 유사성을 출력할 열의 이름을 설정합니다.

고급



조회당 출력에서 일치하는 최대 항목 수는 입력 데이터에 대해 유사 조회 연산을 수행한 후 출력할 수 있는 최대 항목 수를 지정합니다. 이 값을 1로 설정할 경우 유사한 항목으로 판단되는 데이터들 중에서 가장 유사성이 높은 항목 1개만 출력을 합니다. 예를 들어 "서울 강동구"라는 데이터에 대해 "서울시 강동구"가 유사도가 가장 높은 조회 데이터입니다. 하지만 "서울시 강서구"도 유사하다고 판단될 수 있습니다. 만약 이 값이 1인 경우에는 가장 유사하다고 판단되는 "서울시 강동구"가 출력되지만, 2로 설정한 경우, 두 개의 유사 데이터 모두가 출력됩니다.

유사성 임계값은 변환 전체의 유사성에 대한 임계값을 설정합니다. 이는 **열 탭**에서의 각 열의 **최소 유사성**과 비슷한 기능이지만 전체 열의 평균값을 이용하여 판단하는 점이 다릅니다. 예를 들어 B와 B' 열의 유사성은 0.90이며, C와 C' 열의 유사성은 0.5 인 경우, 전체 유사성은 0.70이 됩니다. 만약 **유사성 임계값**을 0.8로 설정했다면 이 경우에는 유사 조회가 수행되지 않습니다.

토큰 구분 기호는 입력 데이터를 비교 가능한 개별 토큰으로 구분할 때 사용되는 구분기호(Delimiter)를 지정하는 부분입니다.

[참고] 유사성(Similarity)과 신뢰성(Confidence)
유사 항목 조회 변환 및 유사 항목 그룹화 변환에서는 유사성과 신뢰성이라는 두 개의 수치가 출력됩니다.

***유사성**은 입력 데이터와 참조 데이터 사이의 문자적 유사성에 대한 측정치 입니다. 예를 들어 "AAA AA"라는 참조 데이터에 대해 "AAA AB"가 "AAA BB" 보다 유사성이 높습니다. 입력 데이터와 참조 데이터가 완전히 동일한 경우에는 유사성은 1이며, 완전히 다른 경우에는 0입니다.*

***신뢰성**은 출력되는 데이터가 참조 테이블의 다른 일치 데이터들과 비교했을 때 가장 유사한 일치 항목이 될 수 있는지의 가능성을 나타내는 수치입니다. 입력 데이터에 대해 선택된 조회 데이터가 다른 유사한 항목 없이 가장 확실한 대상이 된다고 판단되는 경우 1이며, 다른 비슷한 형태의 조회 데이터가 존재하는 경우에는 0에 가깝습니다.*

*"Chapter 4"라는 입력 데이터에 대해 "Chapter 1", "Chapter 2", "Chapter 3"이라는 조회 데이터가 있을 때, 세 개의 데이터 모두가 유사 항목의 후보가 되며, 전체 8자리 중 7자리가 동일하기 때문에 이 입력 값에 대한 각 참조 데이터의 **유사성**은 높습니다. 하지만, 세 개의 참조 데이터 중 어떤 것이 가장 일치하는지는 알 수 없기 때문에 각 참조 데이터에 대한 **신뢰성**은 낮습니다. 만약 입력 데이터가 "3"인 경우, 유사 항목의 후보가 될 수 있는 것은 "Chapter 3"만 가능합니다. 이 경우, 8자리 중 한 자리만 동일하기 때문에 **유사성**은 매우 낮지만, 유사 항목은 하나만 존재하기 때문에, **신뢰성**은 높습니다.*

유사 항목 그룹화

유사 항목 조회 변환은 입력 데이터에 대해 참조 테이블의 데이터와 비교하여 유사하다고 판단되는 항목을 출력하는 작업인 반면, **유사 항목 그룹화** 변환은 입력 데이터들 중 서로 유사하다고 판단되는 항목들로 그룹화 시키는 작업입니다.

다음과 같은 입력 데이터에 대해 설명하겠습니다.

FIELDS OPERATION MGR
FLDS OPS MGR
FIELDS ORS MGR
FIELDS OPERATIONS MANAG

유사 항목 조회 변환의 경우, 참조 테이블에 기준이 되는 항목 값인 FIELDS OPERATION MGR 라는 값이 저장되어 있다고 할 때, 각 입력 데이터는 이 기준 값에 대한 유사성을 판단하여 유사성 및 신뢰도 점수와 함께 데이터를 출력하게 됩니다.
하지만, **유사 항목 그룹화 변환**은 4개의 입력 데이터를 이용하여 유사성을 분석한 후 그룹화 작업을 수행합니다. 예를 들어, 유사성 임계값을 0.70으로 설정한 경우 위의 데이터는 3개의 그룹으로 그룹화를 수행합니다. 아래 표와 같이 유사성 임계값에 따라 분류되는 그룹의 수는 달라집니다.

입력 데이터	유사성 임계값 = 0.70	유사성 임계값 = 0.50	유사성 임계값 = 0.25
FIELDS OPERATION MGR	FIELDS OPERATION MGR	FLDS OPS MGR	FIELDS OPERATION MGR
FLDS OPS MGR	FLDS OPS MGR	FLDS OPS MGR	FIELDS OPERATION MGR
FIELDS ORS MGR	FLDS OPS MGR	FIELDS OPERATIONS MANAG	FIELDS OPERATION MGR
FIELDS OPERATIONS MANAG	FIELDS OPERATIONS MANAG	FIELDS OPERATIONS MANAG	FIELDS OPERATION MGR
그룹 수	3개	2개	1개

유사 항목 그룹화를 수행할 수 있는 데이터 형태는 DT_WSTR만 가능합니다.

연결 관리자

유사 항목 그룹화 변환 편집기의 **연결 관리자** 탭에서는 유사 항목 그룹화 작업을 수행할 때 필요한 임시 저장 테이블의 연결을 지정합니다. 이 때 생성되는 임시 테이블은 입력 데이터를 토큰화 하여 저장하는 인덱스 입니다. 변환 과정에서 자동으로 임시 테이블을 생성하고 필요한 처리 데이터를 입력한 후 쿼리를 이용하여 그룹화 작업을 수행합니다. 입력되는 데이터에 따라 임시 테이블의 크기가 매우 커질 수 있기 때문에 운영 DB 대신 별도의 임시 DB를 사용하는 것이 좋습니다.

열

열 탭에서는 **유사 항목 그룹화**를 수행할 열을 지정하고 그룹화 작업의 세부 설정을 지정합니다.

- **입력 열** - 입력 데이터에서 입력 열을 선택합니다.
- **출력 별칭** - 입력 열을 다른 이름으로 출력할 경우, 이 값을 변경하면 됩니다. 기본적으로 입력 열의 이름과 동일하게 설정됩니다.
- **그룹 출력 별칭** - 입력 데이터에 대해 그룹화 작업을 수행한 후, 가장 유사하다고 판단되는 그룹의 열 데이터를 출력할 때 사용할 열의 이름을 설정합니다. 기본적으로 출력 별칭 뒤에 '_clean' 이라는 접미사가 붙는 형태입니다.

- **일치 유형** - Fuzzy로 설정하면 유사 비교를 수행하며, Exact로 설정하면 완전히 일치하는 경우의 비교만 수행합니다.
- **최소 유사성** - 유사하다고 판단할 최소의 유사성을 설정합니다.
- **유사성 출력 별칭** - 입력 데이터와 그룹핑 한 데이터 간의 유사성을 출력할 열의 이름을 설정합니다. 기본적으로 출력 별칭 앞에 '_Similarity_' 접두사가 붙는 형태입니다.
  - 숫자 - 입력 데이터에서 숫자에 대한 처리 방식을 지정합니다.
  - Neither - 입력 데이터의 앞부분 및 뒷부분의 숫자 모두 분류 작업에 특별한 의미가 없습니다.
  - Leading - 입력 데이터의 앞부분에 나타나는 숫자만 의미가 있습니다.
  - Trailing - 입력 데이터의 뒷부분에 나타나는 숫자만 의미가 있습니다.
  - LeadingAndTrailing - 입력 데이터의 앞부분 및 뒷부분의 숫자 모두 의미가 있습니다.
- **비교 플래그** - 문자열 데이터를 비교할 때 비교 옵션을 지정합니다.

고급

- **입력 키 열 이름** - 입력 데이터 각 행의 고유한 값을 나타내는 열의 이름을 지정합니다. 일반적으로 이 값은 1,2,3,… 과 같이 순차적인 값으로 설정됩니다.

- **출력 키 열 이름** - 입력 데이터가 그룹화 된 행의 고유 값을 나타냅니다.

- **유사성 점수 열 이름** - 입력 데이터와 이 데이터가 그룹화 되는 기준 데이터 간의 유사성 점수를 나타냅니다. 이 값은 0에서 1 사이이며 1에 가까울수록 더 유사하다는 것을 나타냅니다. 이 값은 열 탭에서의 각 열에 대한 유사성에 대한 평균값이며 변환 작업 전체에 대한 평균 유사성 점수를 나타냅니다.

- **유사성 임계값** - 그룹화를 수행할 최소 유사성을 설정합니다. 임계값이 높으면 그룹화 시키는 조건이 더 엄격해지므로 더 많은 그룹으로 분류가 되며, 반대로 임계값이 낮으면 더 적은 수의 그룹으로 분류됩니다.

- **토큰 구분 기호** - 입력 데이터를 그룹화 하는 최소 단위인 토큰으로 구분할 때 사용되는 토큰의 구분 기호를 설정합니다.

출력 형태 분석

위의 그림에서 붉은 색으로 테두리 한 부분을 살펴보겠습니다.

- _key_in 열은 입력 데이터의 각 행에 대한 고유 키 값이며 자동으로 생성됩니다.
- _key_in의 값은 2이지만, _key_out의 값은 1입니다. 이는 두 번째의 입력 데이터가 1번 키 값을 가지는 데이터로 그룹핑 되었다라는 것을 의미합니다.
- _score는 행 전체에 대한 유사성 점수를 나타내며, 현재 유사 비교를 수행하는 열이 InputData라는 열 하나만 존재하기 때문에 _Similarity_InputData 와 동일한 0,5802794를 나타내고 있습니다.
- InputData의 값은 입력된 데이터를 나타내며, InputData_clean은 그룹화 되어 분류되는 값을 나타냅니다. 즉, "EXEC VICE PRES"는 전체 데이터에 대해 유사 항목 그룹화 변환을 수행할 경우, "EXECUTIVE VICE PRESIDENT" 이라는 값으로 처리됩니다.
- _Similarity_InputData는 InputData열에 대한 유사성을 나타냅니다.

행 개수

행 개수는 SSIS의 작업 개체 중 가장 간단하면서도 유용한 개체입니다. 입력 데이터를 직접 변경하지 않기 때문에 정확히 말해서 변환 개체는 아니며, 단순히 데이터 흐름 경로 상에서 통과하는(=처리되는) 행 수를 사용자가 정한 변수에 저장하는 역할을 수행합니다. 입력 데이터에 대해 아무런 변환 작업을 수행하지 않기 때문에 입력 데이터와 출력 데이터가 동일합니다.

일반적으로 데이터 변환 작업 후에 정상적으로 데이터가 처리되었는지를 확인하는 단계가 필요합니다. 예를 들어, 입력 데이터가 100,000개이며 1:1로 처리되어야 하는 조인 변환을 수행한 후 출력 데이터도 100,000개 인지를 확인해야 할 필요가 있습니다. 만약 출력 데이터의 개수가 입력 데이터의 개수와 다른 경우, 매핑 오류로 누락되거나 또는 중복 매핑으로 데이터가 중복이 발생되었다라고 판단하게 됩니다.
이 외에도 **집계 변환**이나 **조건부 분할** 등과 같이 입력되는 데이터에 대해 조건에 맞는 데이터의 수를 별도로 관리해야 할 경우도 있습니다.

만약 저장되는 데이터가 테이블 형태라면 단순히

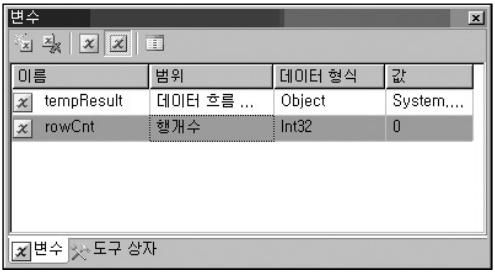
```
SELECT COUNT(*) FROM 〈결과 테이블〉
```

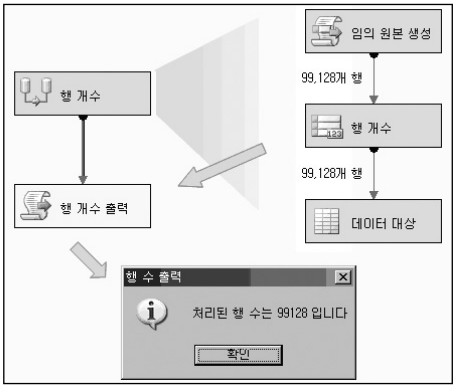
과 같은 쿼리를 이용하여 대상 테이블의 개수를 카운트해서 확인할 수 있습니다. 하지만 결과 테이블이 매우 큰 경우에는 이 작업 또한 많은 부하를 발생시킬 수 있습니다. 또한 데이터의 대상이 텍스트 파일이나 엑셀 파일, ADO 개체 등과 같이 쉽게 카운트 할 수 없는 경우에는 복잡한 작업이 필요할 수도 있습니다.

이러한 경우, 단순히 데이터 대상의 앞부분 또는 변환 작업의 뒷부분에 **행 개수** 개체를 설정하여 처리되는 행 수를 별도의 사용자 변수에 저장하도록 할 수 있습니다.

이전 변환 작업과 다음 변환 작업의 사이 또는 변환 작업과 데이터 흐름 대상 사이에 **행 개수** 개체를 삽입한 후, **고급 행 개수 편집기**의 **구성 요소 속성** 탭에 있는 VariableName 속성에 행 개수를 저장할 변수 명을 지정합니다. **입력 열** 탭에서 나열되는 열은 별도로 선택할 필요는 없습니다.

일반적으로 **데이터 흐름** 작업 내에서 행 개수를 변수에 저장한 후, **제어 흐름** 영역이나 다른 데이터 흐름 작업에서 이용하게 됩니다. 따라서 행 개수를 저장할 변수의 **범위**는 데이터 흐름 작업 상위의 영역에서 정의되는 변수이어야 합니다.





스크립트 구성 요소

스크립트 구성 요소는 다양한 기능을 수행할 수 있는 개체입니다. 스크립트를 이용한 작업은 제어 흐름 영역에 **스크립트 작업**도 있습니다. 하지만 **스크립트 구성 요소**는 원본 데이터를 변형하는 작업과 같은 데이터 흐름과 관련된 작업을 전문적으로 수행하는 개체입니다.

DTS에서의 ActiveX **스크립트 작업**이 SSIS의 **스크립트 작업**이라면, DTS의 Data Pump **작업** 내의 ActiveX **스크립트 작업**이 SSIS의 **스크립트 구성 요소 작업**이라 할 수 있습니다.

스크립트 구성 요소는 데이터 **원본**도 될 수 있으며, 입력 데이터를 처리하는 **변환**도 될 수 있고, 또한 변환 작업을 통해 처리된 데이터에 대한 데이터 **대상**도 될 수 있습니다.

스크립트 구성 요소가 **데이터 원본**으로 사용할 수 있는 경우는 다음과 같습니다.

- **입력 데이터가 플랫 파일 연결이나 OLE DB 연결 등과 같이 간단히 설정할 수 없는 경우** - 원본으로 사용할 데이터가 데이터 연결이나 SQL 쿼리로는 사용할 수 없는 경우에 스크립트 작업을 이용할 수 있습니다. 예를 들어 암호화 된 텍스트 데이터 파일을 읽어와야 할 경우, 암호 해제 모듈을 이용하여 이를 먼저 해독하는 작업이 필요합니다. 이 때 별도의 해독 단계를 구성하지 않고 직접 스크립트 원본 내에서 암호 해제 모듈을 사용하여 읽어오도록 구현할 수 있습니다.

- **기본적으로 제공하는 데이터 원본 유형 이외의 형태** - SQL 2005 SSIS에서는 ODBC 연결을 이용한 데이터 원본을 지원하지 않습니다. 하지만 스크립트 구성 요소를 이용하면 ODBC를 이용한 데이터 원본도 사용할 수 있습니다. 또한 ADSI(Active Directory Service Interface)를 사용하여 Active Directory의 사용자 정보를 원본 데이터로 이용하는 작업도 가능합니다. 기본적으로 제공되는 데이터 원본 형식 외에도 프로그래밍으로 처리 가능한 다양한 형태의 데이터 형태에 대한 접근이 가능합니다.

- **임의의 테스트 데이터를 발생시키는 경우** - 개발 환경이나 테스트 환경에서 자주 사용되는 형태로써 필요한 형태의 원본 데이터를 임의로 생성시킬 수도 있습니다.

스크립트 구성 요소가 **데이터 변환**으로 사용할 수 있는 경우는 다음과 같습니다.

- **입력 데이터에 대해 복잡한 변환 수행** - 예를 들어 '56', '32' 등과 같은 두 자리의 코드 데이터를 이용하여 상태를 판단한 후, 세부적인 상태 정보를 출력하는 작업을 수행할 수 있습니다. 또한 단순히 형 변환 작업이나 자릿수 변환 작업 등도 스크립트 변환을 이용하여 구현 가능합니다.

- **입력 데이터를 이용한 연산 수행** - 입력 데이터를 이용하여 복잡한 산술 연산을 수행할 때 이용할 수 있습니다.

- **입력 데이터에 열 추가** - 입력 데이터의 각 행에 행 번호를 추가하거나 누적 합계 열을 추가하는 작업을 수행할 수 있습니다.

- **입력 데이터를 이용한 유효성 판단** - 입력 데이터의 유효성을 판단한 후 유효하지 않은 경우에는 처리되지 않도록 건너뛰는 작업을 수행할 수 있습니다. 예를 들어, 입력되는 데이터의 값이 Null인 경우 해당 행의 데이터를 처리하지 않고 다음 행으로 건너뛰도록 설정할 수 있습니다.

스크립트 구성 요소가 **데이터 대상**으로 사용할 수 있는 경우는 다음과 같습니다.

- **기본적으로 제공하는 데이터 대상 유형 이외의 형태** - ODBC 연결 대상과 같이 기본적으로 SQL 2005 SSIS에서 제공하지 않는 데이터 대상 유형을 구현할 수 있습니다.

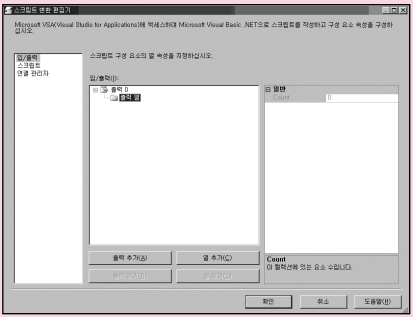
- **처리되는 데이터를 이용하여 다양한 작업을 수행** - 처리된 데이터를 입력 받은 후, 이를 이용하여 SQL 쿼리 파일을 만들거나 웹 보고서를 생성하는 등 결과 데이터를 이용한 다양한 작업을 수행할 수 있습니다.

[따라하기] 스크립트 구성요소 - 데이터 원본으로 이용하기

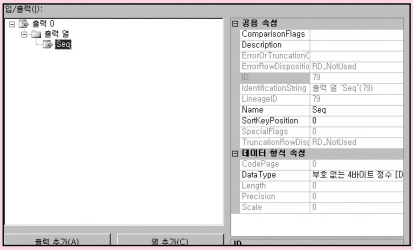
스크립트 구성 요소를 데이터 원본으로 이용하는 예제를 구현합니다. 간단히 **스크립트 구성 요소**를 이용하여 다음 그림과 같은 10,000개의 순차적인 열과 난수 데이터를 발생시키는 예제 입니다.

Seq	Col1	Col2
1	4,190	2,805
2	7,705	2,637
3	5,118	1,857
4	6,984	6,619
⋮	⋮	⋮

1. 빈 패키지 파일을 하나 추가한 후, **데이터 흐름 작업**을 추가합니다.
2. **데이터 흐름 영역** 내에 **스크립트 구성 요소**를 추가한 후, **원본**으로 유형을 설정합니다.
3. 추가한 **스크립트 구성 요소**를 더블 클릭하여 **스크립트 변환 편집기**를 연 후, **입/출력** 탭의 중간 부분에 있는 **출력 0** 부분을 확장한 후, 출력 열을 선택합니다.



4. 아래의 **열 추가(C)**를 클릭하여 열을 추가합니다. 열의 이름을 **Seq**, 데이터 타입을 **부호 없는 4바이트 정수(DT_I4)**로 설정합니다.



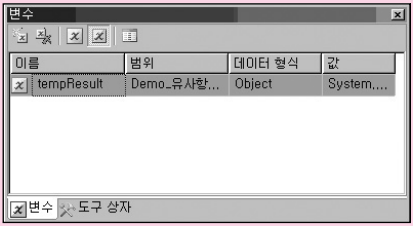
5. 동일한 방법으로 Col1, Col2라는 **부호 없는 4바이트 정수(DT_I4)** 유형의 열을 추가합니다.

6. **스크립트** 탭의 **스크립트 디자인(S)** 버튼을 클릭하여 스크립트 개발 환경인 VSA(Visual Studio for Applications)를 띄운 후 다음 스크립트를 입력합니다.

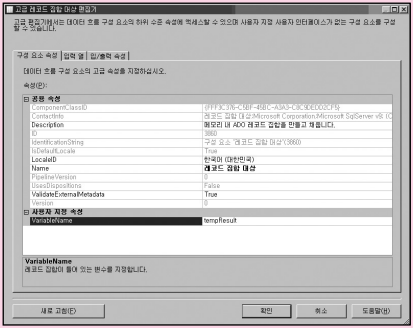
```
Imports System
Imports System.Data
Imports System.Math
Imports Microsoft.SqlServer.Dts.Pipeline.Wrapper
Imports Microsoft.SqlServer.Dts.Runtime.Wrapper

Public Class ScriptMain
    Inherits UserComponent

    Public Overrides Sub CreateNewOutputRows()
        '
        Dim i As Integer
        Dim randVal As New Random

        For i = 1 To 10000
            With 출력Buffer
                .AddRow()                                   ' 출력 행을 추가
                .Seq = i
                .Col1 = randVal.Next(1, 9999)  ' 1에서 9999 사이의 난수발생
                .Col2 = randVal.Next(1, 99999)  ' 1에서 99999 사이의 난수발생
            End With
        Next

    End Sub

End Class
```

7. VSA를 닫고 확인 버튼을 눌러 **스크립트 변환 편집기**를 닫습니다.

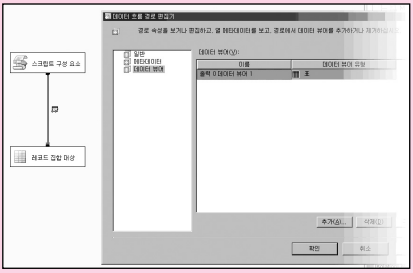
8. **데이터 흐름 영역**의 빈 곳에서 마우스 오른쪽 버튼을 클릭하여 나타나는 메뉴 중 **변수(S)**를 선택하여 변수 창을 띄운 후, tempResult라는 이름의 Object 형 변수를 하나 추가합니다. 이는 데이터 대상인 **레코드 집합 대상**에서 이용할 변수입니다.



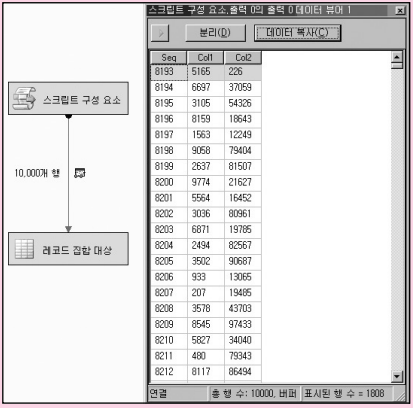
9. **도구 상자**에서 **레코드 집합 대상**을 추가한 후, **스크립트 구성 요소**의 녹색 선과 연결합니다. **레코드 집합 대상**을 더블 클릭하여 **레코드 집합 대상 편집기**를 띄운 후, **구성 요소 속성** 탭에서 VariableName에 tempResult를 입력합니다. 입력 열 탭에 있는 세 개의 열을 모두 선택합니다.



10. 데이터가 처리되는 과정을 확인하기 위해 중간 경로에 **데이터 뷰어**를 추가합니다. **스크립트 구성 요소**와 **레코드 집합 대상** 간의 녹색 경로를 더블 클릭한 후 나타나는 **데이터 흐름 경로 편집기**의 **데이터 뷰어** 탭에서 **추가(A)**를 누른 후 표 형태의 뷰어를 추가합니다.



11. 패키지를 실행하여 생성되는 원본 데이터를 확인합니다.



[따라하기] 스크립트 구성요소 - 데이터 변환 개체로 이용하기

SQL 쿼리를 이용한 입력 데이터에 순번 열과 누적 수량 열을 추가하는 작업을 구현합니다.

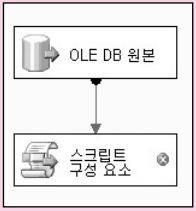
1. 빈 패키지 파일을 하나 추가한 후, 제어 흐름 영역에 **데이터 흐름 작업**을 추가합니다.

2. 아래 부분에 있는 **연결 관리자**에서 AdventureWorks DB를 지정하는 **OLE DB 연결**을 생성합니다. 이 연결은 **스크립트 구성 요소**를 이용한 변환 작업을 수행하기 위한 **OLE DB 원본**에서 사용합니다.

3. **데이터 흐름 영역**에 **OLE DB 원본**을 추가한 후, 2에서 추가한 연결을 지정하고 **데이터 액세스 모드**를 **SQL 명령**으로 변경합니다. 그런 다음, **SQL 명령 텍스트**에 다음과 같은 쿼리를 입력합니다.

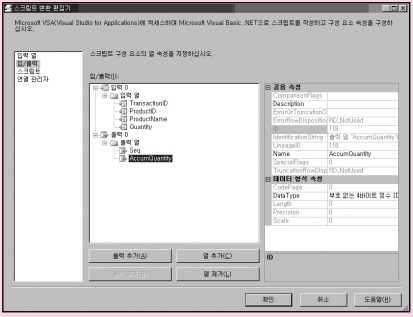
```
SELECT A.TransactionID, A.ProductID, B.Name AS ProductName, Quantity
FROM Production.TransactionHistory A
        JOIN Production.Product B ON A.ProductID = B.ProductID
```

4. **도구 상자**에서 **스크립트 구성 요소**를 추가한 후, 유형을 **변환**으로 선택하고 **OLE DB 원본**의 녹색 선을 연결합니다.



5. **스크립트 구성 요소**를 더블 클릭하여 **스크립트 변환 편집기**를 연 후, **입력 열** 탭에서 네 개의 열을 모두 선택합니다.

6. **입/출력** 탭에서 **출력 0**을 확장한 후, **출력 열**을 선택한 상태에서 **열 추가(C)**를 클릭하여 열을 추가합니다. 추가한 열의 이름을 **Seq**, 유형을 **부호 없는 4바이트 정수**(DT_I4)로 설정합니다. 동일한 방법으로 열을 하나 더 추가한 후, 이름을 **AccumQuantity**, 유형을 **부호 없는 4바이트 정수**(DT_I4)로 설정합니다.



7. 왼쪽의 **스크립트** 탭에서 **스크립트 디자인(S)** 버튼을 클릭하여 스크립트 개발 환경인 VSA(Visual Studio for Applications)를 띄운 후 다음 스크립트를 입력합니다.

```
Imports System
Imports System.Data
Imports System.Math
Imports Microsoft.SqlServer.Dts.Pipeline.Wrapper
Imports Microsoft.SqlServer.Dts.Runtime.Wrapper
```

```
Public Class ScriptMain
    Inherits UserComponent

    Dim seq As Integer
    Dim AccumQty As Integer

    Public Sub New()
        seq = 0
        AccumQty = 0
    End Sub

    Public Overrides Sub 입력_ProcessInputRow(ByVal Row As 입력Buffer)

        seq += 1

        Row.Seq = seq
        AccumQty = AccumQty + Row.Quantity
        Row.AccumQuantity = AccumQty

    End Sub

End Class
```

8. VSA를 닫은 후, 확인 버튼을 눌러 **스크립트 변환 편집기**도 닫습니다.

9. **데이터 흐름 영역**의 빈 곳에서 마우스 오른쪽 버튼을 클릭하여 나타나는 메뉴 중 **변수(S)**를 선택하여 변수 창을 띄운 후, tempResult라는 이름의 Object 형 변수를 하나 추가합니다

10. 도구 상자에서 **레코드 집합 대상**을 추가한 후, **스크립트 구성 요소**의 녹색 선과 연결합니다. **레코드 집합 대상**을 더블 클릭하여 **레코드 집합 대상 편집기**를 띄운 후, **구성 요소 속성** 탭의 VariableName에 tempResult를 입력합니다. **입력 열** 탭에서 모든 열을 선택합니다.

11. 처리되는 데이터를 확인하기 위해 **데이터 뷰어**를 추가합니다. **스크립트 구성 요소**와 **레코드 집합 대상** 사이의 데이터 흐름 경로를 더블 클릭한 후, **데이터 뷰어** 탭에서 **표** 형태의 **데이터 뷰어**를 추가합니다.

12. 패키지를 실행하여 처리되는 데이터를 확인합니다.

[따라하기] 스크립트 구성요소 - 데이터 대상으로 이용하기

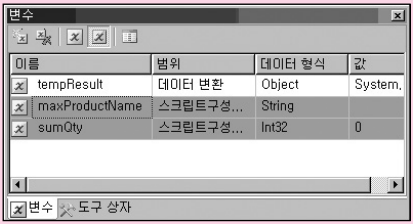
[따라하기 - 데이터 변환 작업으로 이용하기]의 작업에서 **레코드 집합 대상** 대신 **스크립트 구성 요소**를 이용한 대상을 만드는 예제를 구현합니다. 스크립트를 이용한 대상에서는 변환 단계를 거쳐 대상으로 입력되는 데이터 중에서 ProductName열의 길이가 가장 긴 데이터와 Quantity 열의 합계를 메시지 박스로 출력하는 예제입니다.

[따라하기 - 데이터 변환 작업으로 이용하기]의 8단계 까지는 동일합니다.

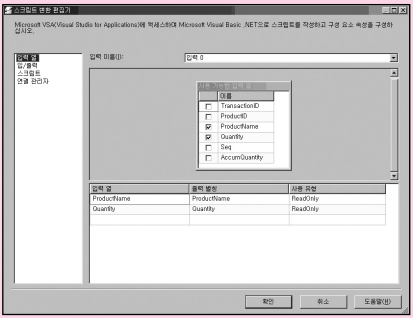
9. **도구 상자**에서 **스크립트 구성 요소**를 하나 더 추가한 후, **유형**을 **대상**으로 선택하고 이름을 **스크립트 대상**으로 변경합니다. 이 후, 변환 작업을 수행하는 **스크립트 구성 요소**의 녹색 선과 연결합니다.

10. 패키지 수준의 변수를 추가해야 합니다. **데이터 흐름 영역**의 빈 곳에서 마우스 오른쪽 버튼을 클릭하여 나타나는 메뉴 중 **변수(S)**를 선택하여 변수 창을 띄웁니다.

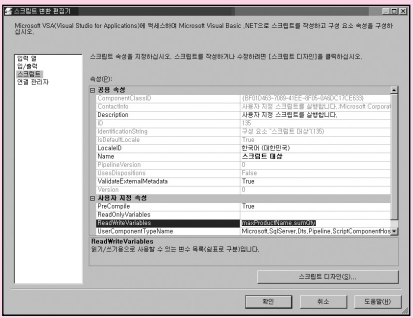
11. 변수 창에서 maxProductName이라는 String형 변수와 sumQty라는 Int32형 변수를 추가합니다.



12. **데이터 흐름** 영역에서 **스크립트 대상**을 더블 클릭한 후, **입력 열** 탭에서 ProductName 열과 Quantity 열을 선택합니다.



13. **스크립트** 탭에서 ReadWriteVariables 속성에 maxProductName, sumQty를 입력한 후, **스크립트 디자인(S)**을 클릭하여 VSA를 띄웁니다.



14. 다음과 같은 코드를 입력한 후, VSA를 닫고 확인 버튼을 눌러 편집기 창을 닫습니다.

```
Imports System
Imports System.Data
Imports System.Math
Imports Microsoft.SqlServer.Dts.Pipeline.Wrapper
Imports Microsoft.SqlServer.Dts.Runtime.Wrapper

Public Class ScriptMain
    Inherits UserComponent

    Dim tmpStr As String
    Dim sumQty As Integer
    Public Overrides Sub PreExecute()

        tmpStr = ""
        sumQty = 0

    End Sub

    Public Overrides Sub 입력_ProcessInputRow(ByVal Row As 입력Buffer)

        If Row.ProductName.Length > tmpStr.Length Then
            tmpStr = Row.ProductName.ToString
        End If
```

```
        sumQty = sumQty + Row.Quantity

    End Sub

    Public Overrides Sub PostExecute()

        Variables.maxProductName = tmpStr
        Variables.sumQty = sumQty

    End Sub

End Class
```

15. 이제 최종 결과가 maxProductName, sumQty 변수에 저장됩니다. 이 저장된 값을 확인하기 위해 **제어 흐름** 영역에서 **스크립트 작업**을 추가한 후, **데이터 흐름 작업**의 녹색 선과 연결합니다.

16. **스크립트 작업**을 더블 클릭한 후 스크립트 탭의 ReadOnlyVariables 속성에 maxProductName, sumQty을 입력합니다.

17. **스크립트 디자인(S)**를 클릭하여 VSA를 실행시킨 후, 다음 스크립트를 입력합니다.

```
Imports System
Imports System.Data
Imports System.Math
Imports Microsoft.SqlServer.Dts.Runtime
```

```
Public Class ScriptMain

    Public Sub Main()
    MsgBox("Max Length ProductName : " &
Dts.Variables("maxProductName").Value.ToString, MsgBoxStyle.Information)

    MsgBox("Sum Quantity : " & Dts.Variables("sumQty").Value.ToString,
MsgBoxStyle.Information)

                    Dts.TaskResult = Dts.Results.Success
    End Sub

End Class
```

18. 패키지를 실행하여 수행되는 결과를 확인합니다.

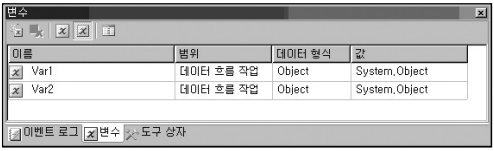
기타 구성 요소

변수

SSIS에는 크게 두 가지 유형의 변수가 있습니다. 하나는 **시스템 변수**이며, 다른 하나는 **사용자 변수**입니다. 이러한 구분은 변수 정의 시 **네임 스페이스**로 구분이 됩니다.

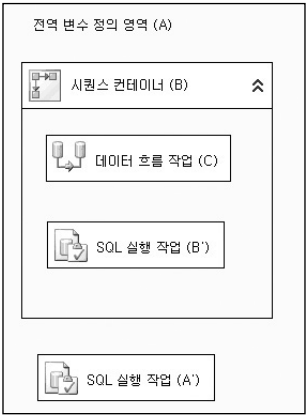
시스템 변수는 패키지 생성 일자, 패키지 GUID, Version GUID, Machine Name 등 패키지 수행 환경에 대한 정보가 저장되는 변수이며 사용자가 해당 값을 변경할 수 없습니다.

사용자 변수는 패키지 내에서 작업을 수행할 때 사용되는 변수로 사용자가 필요에 따라 추가하며, 값을 변경할 수 있습니다.



SQL 2000 DTS에서의 변수와 가장 큰 차이점은 변수 정의 시 범위가 설정되는 점입니다.

DTS에서의 변수는 범위의 개념 없이 모두 동일한 수준에서 처리되었습니다. 하지만, SSIS에서는 컨테이너 개체를 기준으로 범위가 설정됩니다. 작업 개체 역시 하나의 **작업 호스트** 컨테이너이기 때문에 변수의 범위로 정의될 수 있습니다.



변수

이름	범위	데이터 형식	값
변수1	변수예제	Int32	0
변수2	시퀀스 컨테이너 (B)	Int32	0
변수3	데이터 흐름 작업 (C)	Int32	0

위의 그림에서, [변수1]은 예제 패키지 파일의 이름과 동일한 **변수예제**라는 최상위 범위에서 정의된 변수입니다. 따라서 [변수1]은 패키지의 모든 영역에서 이용할 수 있습니다.

[변수2]는 하위 컨테이너인 **시퀀스 컨테이너**(B)의 범위에서 정의된 변수이며 **시퀀스 컨테이너**(B) 내에 포함된 **SQL 실행 작업**(B')이나 **데이터 흐름 작업**(C) 등과 같은 하위 작업에서는 사용할 수 있지만 **SQL 실행 작업**(A')에서는 사용할 수 없습니다.

[변수3]은 **데이터 흐름 작업**(C)의 범위에서 정의된 변수이기 때문에, **데이터 흐름 작업**(C) 외에는 사용할 수 없습니다.

변수의 범위는 변수를 추가할 때 지정할 수 있는 아니며, 변수를 추가할 때 선택되어 있는 컨테이너가 자동으로 변수의 범위로 설정됩니다. 즉, **데이터 흐름 작업**(C) 내의 여러 변환 개체에서 변수를 정의할 경우에는 자동으로 해당 변수의 범위가 현재 작업 중인 컨테이너인 **데이터 흐름 작업**(C)으로 설정됩니다.
만약 **데이터 흐름 작업**(C) 에서 상위 범위의 변수를 추가하기 위해서는 상위의 범위를 선택한 상태에서 변수를 추가해야 합니다.

변수 창의 **회색 X** 버튼을 누르면 시스템 변수 목록이 나타나며, **파란색 X** 버튼을 누르면 현재 패키지에서 정의된 모든 사용자 변수가 나타납니다. 이 버튼이 선택되지 않은 상태에서는 현재 컨테이너에서 사용 가능한 변수의 목록만 나타납니다.

제어 흐름 영역 및 **데이터 흐름 영역**, **이벤트 처리기 작업 영역**에서의 시스템 변수는 조금 차이가 있습니다. 각 영역별 시스템 변수는 다음 표를 참고하기 바랍니다.

- **패키지 수준의 시스템 변수**

시스템 변수	데이터 형식	설명
CancelEvent	Int32	0이 아닌 값으로 설정되는 경우 작업 실행이 중지됨을 나타내는 이벤트 핸들입니다.
CreationDate	DateTime	패키지를 만든 날짜입니다.
CreatorComputerName	String	패키지를 만든 컴퓨터입니다.
CreatorName	String	패키지를 만든 사용자의 이름입니다.
ExecutionInstanceGUID	String	실행 중인 패키지의 고유 식별자입니다.

시스템 변수	데이터 형식	설명
InteractiveMode	Boolean	패키지가 대화형 모드에서 실행 중인지 여부를 나타냅니다. SSIS 디자이너에서 패키지를 실행 중인 경우 이 속성은 True로 설정됩니다. DTExec 명령 프롬프트 유틸리티를 사용하여 패키지를 실행 중인 경우 이 속성은 False로 설정됩니다.
LocaleId	Int32	패키지에서 사용되는 로캘입니다.
MachineName	String	패키지가 실행 중인 컴퓨터 이름입니다.
OfflineMode	Boolean	패키지가 오프라인 모드인지 여부를 나타냅니다. 오프라인 모드에서는 데이터 원본에 연결하지 않습니다.
PackageID	String	패키지의 고유 식별자입니다.
PackageName	String	패키지의 이름입니다.
StartTime	DateTime	패키지 실행을 시작한 시간입니다.
UserName	String	패키지를 시작한 사용자의 계정입니다. 사용자 이름은 도메인 이름에 의해 한정됩니다.
VersionBuild	Int32	패키지 버전입니다.
VersionComment	String	패키지 버전에 대한 설명입니다.
VersionGUID	String	버전의 고유 식별자입니다.
VersionMajor	Int32	패키지의 주 버전입니다.
VersionMinor	Int32	패키지의 부 버전입니다.

• 컨테이너의 시스템 변수

시스템 변수	데이터 형식	설명
LocaleId	Int32	컨테이너에서 사용되는 로캘입니다.

• 작업 수준의 시스템 변수

시스템 변수	데이터 형식	설명
CreationName	String	작업 이름입니다.
LocaleId	Int32	작업에서 사용되는 로캘입니다.
TaskID	String	작업의 고유 식별자입니다.
TaskName	String	작업 이름입니다.
TaskTransactionOption	Int32	작업에서 사용되는 트랜잭션 옵션입니다.

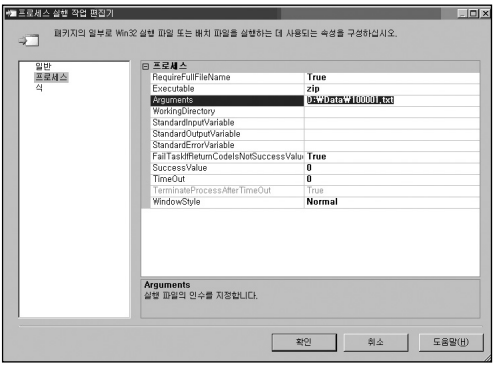
• 이벤트 처리기의 시스템 변수

시스템 변수	데이터 형식	이벤트 처리기
Cancel	오류, 경고 또는 쿼리 취소가 발생할 때 이벤트 처리기 실행이 중지되는지 여부를 나타냅니다.	OnError OnWarning OnQueryCancel
ErrorCode	오류 식별자입니다.	OnError OnInformation OnWarning
ErrorDescription	오류에 대한 설명입니다.	OnError OnInformation OnWarning

시스템 변수	데이터 형식	이벤트 처리기
ExecutionStatus	현재 실행 상태입니다.	OnExecStatusChanged
ExecutionValue	실행 값입니다.	OnTaskFailed
LocaleId	이벤트 처리기에서 사용되는 로캘입니다.	All
PercentComplete	완료된 작업의 백분율입니다.	OnProgress
ProgressCountHigh	OnProgress 이벤트에 의해 처리된 전체 작업 개수를 나타내는 64비트 값의 상위 부분입니다.	OnProgress
ProgressCountLow	OnProgress 이벤트에 의해 처리된 전체 작업 개수를 나타내는 64비트 값의 하위 부분입니다.	OnProgress
ProgressDescription	진행률에 대한 설명입니다.	OnProgress
Propagate	이벤트가 상위 수준의 이벤트 처리기로 전달되는지 여부를 나타냅니다.	All
SourceDescription	이벤트 처리기에서 이벤트를 발생시킨 실행 개체에 대한 설명입니다.	All
SourceID	이벤트 처리기에서 이벤트를 발생시킨 실행 개체의 고유 식별자입니다.	All
SourceName	이벤트 처리기에서 이벤트를 발생시킨 실행 개체의 이름입니다.	All
VariableDescription	변수 설명입니다.	OnVariableValueChanged
VariableID	변수의 고유 식별자입니다.	OnVariableValueChanged

식

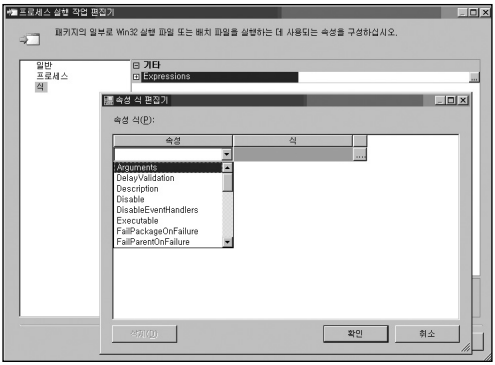
식(Expressions) 기능은 SSIS를 동적으로 운영할 수 있도록 하는 기능입니다.



예를 들어 **프로세스 실행 작업**을 이용하여 zip.exe라는 압축 프로그램을 매일 실행하며, Arguments에는 *20060915_Data.txt* 와 같이 수행될 때마다 매번 파일 이름을 변경해야 할 경우를 생각해 봅시다.
이 경우에는 간단히 StandardInputVariable 속성을 이용하여 구현할 수 있습니다. 패키지에 String형 변수를 생성한 후, **스크립트 작업**이나 **SQL 실행 작업**, 또는 **식**(Expression)을 이용하여 파일 이름을 변수에 저장합니다. 그런 다음, StandardInputVariable 속성에 해당 변수를 지정하면 됩니다.

하지만 Arguments 대신 실행하는 프로그램 이름(Executable)이나 작업 디렉터리(WorkingDirectory)등과 같은 다른 속성들을 지정해야 하는 경우도 있습니다. SQL 2000 DTS에서는 **동적 속성 작업**이나 **ActiveX 스크립트 작업**을 이용하여 작업의 속성들을 변경하였습니다. SQL 2005 SSIS에도 동적 속성 작업을 대체하는 구성(Configurations) 기능이 있지만 이는 DTS에서와는 차이가 있습니다.

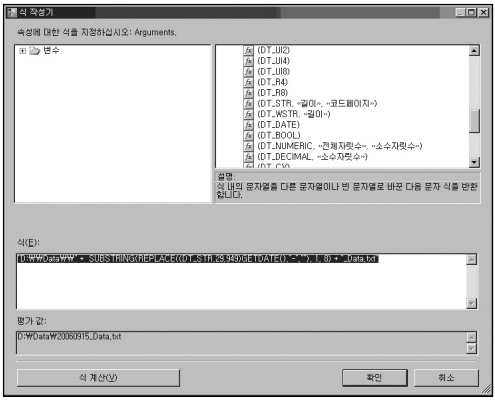
SQL 2005 SSIS의 대부분의 작업 개체 및 작업 영역에서는 해당 개체의 속성을 설정할 수 있는 식(Expressions) 기능이 있습니다. 앞의 예에서 Arguments에 파일 이름을 지정할 때 변수를 사용하는 대신 직접 식을 이용하여 정의할 수도 있습니다. 작업 편집기의 **식** 탭에서 Expressions 오른쪽에 있는 ... 버튼을 누르면 **속성 식 편집기**가 나타납니다.



이 편집기에서 해당 개체의 다양한 속성값을 설정할 수 있습니다.

속성에서 Arguments를 선택한 후 오른쪽의 식 부분 옆에 있는 ... 버튼을 클릭하여 **식 작성기**를 띄운 후, 다음과 같은 형식으로 식을 작성하면 패키지가 실행될 때 Arguments 속성 부분에 해당 날짜의 파일 이름이 지정됩니다.

```
"D:₩₩Data₩₩" + SUBSTRING(REPLACE((DTSTR,29,494)GETDATE(), "-","" ),
1, 8) + "_Data.txt"
```



이와 같은 방식으로 식을 작성하여 사용할 수 있습니다.
변수에는 단 하나의 식만 정의할 수 있지만, 작업 개체나 컨테이너, 연결 등과 같이 여러 속성을 가지는 개체에는 여러 개의 식을 정의하여 사용할 수도 있습니다.
식에서 정의할 수 있는 속성은 일부 공통 항목 외에는 각 개체마다 조금씩 차이가 있습니다. 위의 예에서와 같이 Arguments 속성은 **프로세스 실행 작업**에만 있는 속성 항목이며, Name, TransactionOption 등과 같은 속성은 대부분의 작업에 포함되어 있는 속성 항목입니다.

식 작성기에서 식을 작성할 때에는 **시스템 변수** 및 **사용자 변수**를 사용할 수도 있으며, **수치 연산 함수, 문자열 함수, 날짜/시간 함수, NULL 함수, 유형 변환, 연산자** 등을 이용하여 작성할 수 있습니다.

• 수치 연산 함수

함수	설명
ABS	숫자 식의 절대값을 양수로 반환합니다.
EXP	밑이 e인 지정한 식의 지수를 반환합니다.
CEILING	숫자 식보다 크거나 같은 최소 정수를 반환합니다.
FLOOR	숫자 식보다 작거나 같은 최대 정수를 반환합니다.
LN	숫자 식의 자연 로그를 반환합니다.
LOG	숫자 식의 상용 로그를 반환합니다.
POWER	숫자 식의 거듭제곱을 반환합니다.
ROUND	특정 길이나 전체 자릿수로 반올림한 숫자 식을 반환합니다.
SIGN	숫자 식의 양수(+), 음수(-) 또는 영(0) 부호를 반환합니다.
SQUARE	숫자 식의 제곱을 반환합니다.
SQRT	숫자 식의 제곱근을 반환합니다.

• 문자열 함수

함수	설명
CODEPOINT	문자 식에서 가장 왼쪽 문자의 유니코드 코드 값을 반환합니다.
FINDSTRING	식에서 지정한 문자열 항목의 인덱스(1부터 시작)를 반환합니다.
HEX	정수의 16진수 값을 나타내는 문자열을 반환합니다.
LEN	문자 식에 포함된 문자의 수를 반환합니다.
LOWER	대문자를 소문자로 변환한 후에 문자 식을 반환합니다.

함수	설명
LTRIM	선행 공백을 제거하고 문자 식을 반환합니다.
REPLACE	식 내의 문자열을 다른 문자열이나 빈 문자열로 바꾼 후 문자 식을 반환합니다.
REPLICATE	지정한 횟수만큼 복제된 문자 식을 반환합니다.
REVERSE	문자 식을 역 순서로 반환합니다.
RIGHT	오른쪽의 지정한 문자 수에서 시작하여 문자열의 일부를 반환합니다.
RTRIM	후행 공백을 제거하고 문자 식을 반환합니다.
SUBSTRING	문자 식의 일부를 반환합니다.
TRIM	선행 및 후행 공백을 제거하고 문자 식을 반환합니다.
UPPER	소문자를 대문자로 변환한 후에 문자 식을 반환합니다.

• 날짜/시간 함수

함수	설명
DATEADD	지정한 날짜에 날짜 또는 시간 간격을 더하여 새로운 DT_DBTIMESTAMP 값을 반환합니다.
DATEDIFF	지정한 두 날짜 간에 교차되는 날짜와 시간 경계값을 반환합니다.
DATEPART	날짜의 특정 부분을 나타내는 정수를 반환합니다.
DAY	지정한 날짜의 일을 나타내는 정수를 반환합니다.
GETDATE	시스템의 현재 날짜를 반환합니다.

함수	설명
GETUTCDATE	시스템의 현재 날짜를 UTC 시간(국제 표준시 또는 그리니치 표준시)으로 반환합니다.
MONTH	지정한 날짜의 월을 나타내는 정수를 반환합니다.
YEAR	지정한 날짜의 연도를 나타내는 정수를 반환합니다.

- NULL 함수

함수	설명
ISNULL	식이 Null인지 여부에 따라 부울 결과를 반환합니다.
NULL	요청한 데이터 형식의 Null 값을 반환합니다.

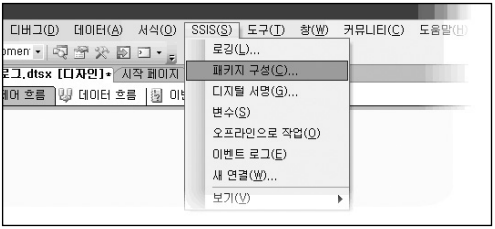
구성

SQL 2005 SSIS에서는 패키지의 속성값을 설정할 수 있는 **구성**(Configurations)이라는 기능을 제공합니다. 패키지에 포함되어 있는 컨테이너나 작업, 연결, 변환 등에 대한 속성뿐만 아니라 패키지의 격리 수준이나 검사점 파일 이름 등과 같은 패키지의 전반적인 속성에 대해서도 값을 설정할 수 있습니다.

구성 기능은 SQL 2000 DTS에서의 **동적 속성 작업**과 상당히 유사하지만, 부모 패키지 변수 설정이나 SQL Server의 테이블에서 정보 가져오기, XML 형태의 설정 파일 저장 등과 같은 기능들이 추가되어 훨씬 다양한 기능을 제공합니다. 패키지의 구성은 **패키지 구성 마법사**를 통해 쉽게 수행할 수 있게 되었습니다.

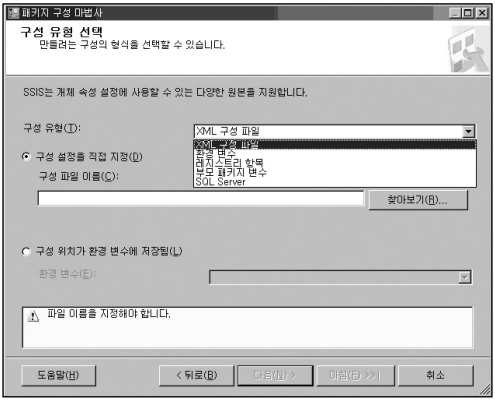
동일한 패키지 작업을 여러 서버에서 수행해야 하거나 다수의 패키지에서 사용하는 연결 속성 등을 일괄 관리하고자 할 때 구성을 이용할 수 있습니다. 또한, 부모 패키지에서 자식 패키지로 값을 지정해 주는 기능을 구현할 때에도 이용할 수 있으며, 시스템 레지스트리의 항목을 읽어오거나 시스템 변수의 값을 사용해야 하는 경우에도 이용할 수 있습니다.

구성에서 설정된 값들은 패키지가 실제 실행이 되는 시점(런타임)에 적용됩니다. 그리고 구성이 설정되었더라도 해당 구성 파일 또는 테이블 정보가 없는 경우에도 패키지는 정상적으로 수행됩니다. 예를 들어 D:₩package.config 라는 XML 형태의 구성 파일을 지정하였더라도, 패키지를 실행할 때 해당 파일이 없는 경우에도 오류 없이 정상적으로 수행됩니다. 이러한 경우에는 구성 정보를 이용하여 속성값을 바꾸는 대신 기존의 속성값을 그대로 이용하게 됩니다.



패키지 개발 화면의 상단에 있는 SSIS(S) → **패키지 구성(C)**을 선택하여 **패키지 구성 마법사**를 수행할 수 있습니다.

구성은 다음과 같은 유형으로 지정할 수 있습니다.



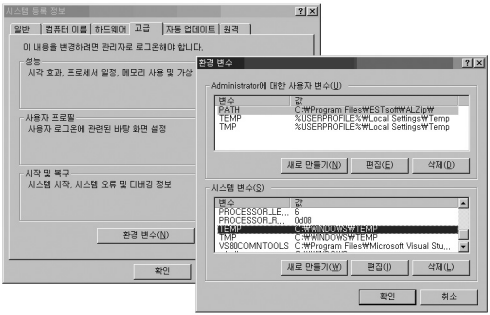
- **XML 구성 파일**

  XML 파일 형태로 구성의 설정 값이 저장됩니다. 하나의 XML 구성 파일에는 여러 개의 설정 값이 저장될 수 있습니다.

  XML 구성 파일은 크게 두 개의 부분으로 나눠집니다. 상단의 제목 부분은 구성 파일 자체에 대한 정보를 포함하는 부분이며, 파일을 만든 시간, 패키지 명, 패키지 ID 등과 같은 값 등이 저장됩니다. 하단의 내용은 구성으로 저장된 속성과 속성 값이 저장되는 부분입니다.

- **환경 변수**

  시스템의 환경 변수에 등록된 값을 패키지에서 이용하도록 설정할 수 있습니다.



위의 그림에서와 같이 시스템의 환경 변수 설정 부분에서 변수를 추가한 후 이를 SSIS의 패키지에서 사용하도록 지정할 수 있으며, 시스템에서 설정된 환경 변수들을 패키지의 작업 폴더의 경로나 다른 속성의 값으로 이용하도록 기본적으로 제공되는 환경 변수를 지정할 수 있습니다.

- **레지스트리 항목**

  구성에서 사용할 항목의 값을 시스템의 레지스트리에 저장한 후 이를 사용할 수 있습니다. 또한 환경 변수와 마찬가지로 시스템에 저장되어 있는 레지스트리 항목을 SSIS 패키지에서 사용할 수 있습니다. 이 때, 읽어오거나 지정할 수 있는 레지스트리는 HKEY_CURRENT_USER 하위에 있는 키 값이어야 합니다.

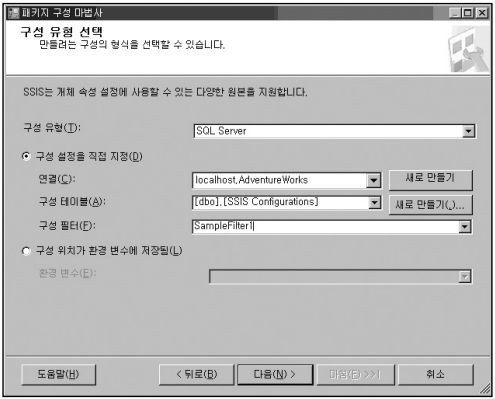
- **부모 패키지 변수**

  부모 패키지에서 자식 패키지를 호출할 때, 자식 패키지에 있는 변수의 값 또는 개체의 속성 값을 지정할 수 있습니다. 자식 패키지에서는 값을 가져올 부모 패키지의 변수 이름을 지정해 줍니다. 부모 패키지 변수가 설정된 자식 패키지를 부모 패키지에서 호출할 때 특별한 설정 없이도 부모 패키지에 해당 변수가 있을 때 이 변수에 저장된 값이 자식 패키지로 전달되어 실행됩니다. (제어 흐름 요소의 **패키지 실행 작업 따라하기** 참고)

- SQL Server

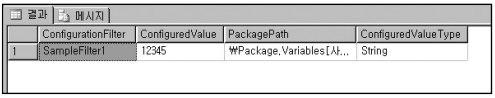
  SQL Server에 테이블 형태로 구성 정보를 저장하여 이용할 수 있습니다. 구성 정보를 저장하는 테이블은 다음과 같은 형태입니다.

```
CREATE TABLE [dbo].[SSIS Configurations]
(
        ConfigurationFilter NVARCHAR(255) NOT NULL,
        ConfiguredValue NVARCHAR(255) NULL,
        PackagePath NVARCHAR(255) NOT NULL,
        ConfiguredValueType NVARCHAR(20) NOT NULL
)
```



구성 테이블이 저장되어 있는 DB에 대한 연결을 지정하고, **구성 테이블(A)**에서 구성 테이블로 사용할 테이블의 이름을 선택합니다.

구성 필터(F)는 구성 테이블 중, 현재 사용할 속성의 ConfigurationFilter 열 값입니다.


결과 | 메시지

	ConfigurationFilter	ConfiguredValue	PackagePath	ConfiguredValueType
1	SampleFilter1	12345	\Package.Variables[사...	String

검사점

수행하는 패키지가 복잡하거나 많은 패키지가 반복적으로 수행되는 환경에서는 여러 가지 이유로 패키지의 수행이 실패할 수 있습니다. 데이터가 저장된 DB에서 문제가 발생하거나 잘못된 데이터가 입력되어 오류가 발생할 수도 있으며, 데이터를 저장할 디스크의 공간이 부족하여 작업이 실패할 수도 있습니다.

이러한 오류 상황이 발생되었을 때에는 일반적으로 다음과 같은 과정을 거쳐 처리하게 됩니다.

1. 패키지에서 오류가 발생한 작업 개체를 정확히 아는 경우에는, 오류가 발생한 단계의 앞 부분에 있는 작업들을 삭제하거나 비활성화 한 후, 오류가 발생한 부분부터 재실행 합니다.

2. 패키지가 반복해서 수행되는 경우라면, 오류가 발생한 작업의 앞 부분에 있는 작업들이 여러 번 수행되었을 수 있습니다. 중복으로 데이터가 적재된 경우에는 해당 데이터를 삭제하고 다시 처리해야 합니다.

3. 일 배치 작업과 같이 수행 시간이 짧을 경우는 쉽게 재실행 할 수 있지만, 월 배치 또는 년 배치 작업과 같이 수행 시간이 매우 길거나 개별 작업들의 수행 시간이 긴 경우에는 다시 실행하는 것이 어려울 수 있습니다.

SQL 2005 SSIS의 **검사점**(Checkpoint) 기능을 이용할 경우 위와 같은 사항들을 쉽게 처리할 수 있습니다.

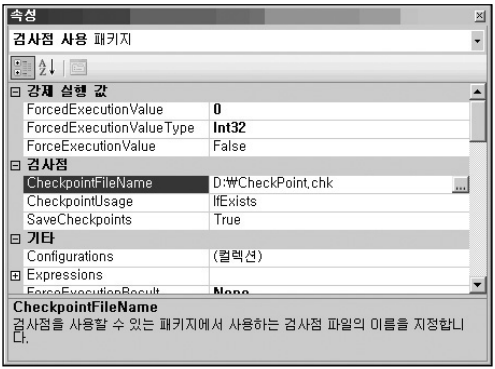
1. 패키지에 **검사점**을 설정하면 패키지의 수행이 실패했을 때 오류가 발생한 작업 개체에 대한 정보와 정상적으로 처리된 작업들의 정보가 별도의 파일에 기록됩니다. 이 기록 파일에는 수행 시점에서 적용된 변수의 값도 포함됩니다.

2. **검사점** 파일이 존재하는 경우에는 패키지가 다시 수행되더라도 이미 수행한 작업들에 대해서는 다시 수행하지 않으며, 오류가 발생한 작업부터 수행을 시작합니다.

3. 패키지의 모든 작업들이 오류 없이 정상적으로 수행되는 경우에는 검사점 파일이 생성되지 않습니다. 따라서, 관리자는 작업이 실패가 발생되었는지를 확인하기 위한 방법으로 **검사점** 파일의 생성 여부만 확인하면 됩니다.

검사점은 **제어 흐름** 영역에서 설정합니다. 검사점에서 관리하는 최소 작업 단위는 개별 작업(작업 호스트) 단위입니다. 트랜잭션이 설정된 컨테이너인 경우, 해당 컨테이너가 최소 작업 개체가 됩니다.

검사점은 패키지의 **제어 흐름 영역**의 속성에서 설정합니다.



- CheckpointFileName - 검사점 파일을 지정합니다.
- CheckpointUsage - 검사점 사용 여부를 설정합니다.
  - Never - 검사점을 사용하지 않습니다. 기본값이며, 검사점을 사용하기 위해서는 이 값 대신 Always 또는 IfExists로 변경해야 합니다.
  - Always - 검사점 파일을 항상 사용합니다. 오류가 발생하지 않더라도 검사점 파일이 있어야 합니다.
  - IfExists - 검사점 파일이 있는 경우, 해당 파일을 사용합니다.
- SaveCheckpoints - 오류 발생시 검사점 파일에 오류 정보를 저장할지를 설정합니다. 검사점 기능을 사용하기 위해서는 이 옵션을 True로 설정해야 합니다.

이 외에도 **검사점** 기능을 구현하기 위해서는 제어 흐름에 있는 각 작업의 FailPackage OnFailure 속성이 True로 설정되어 있어야 합니다.

패키지 내에 작업이 많은 경우, 각 작업들을 모두 선택한 후에 속성 창에서 해당 속성값을 변경하면 일괄적으로 변경됩니다.

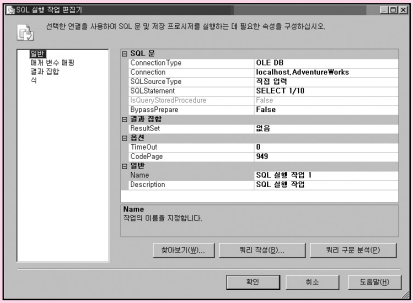
[따라하기] 검사점

검사점 기능을 이용하여 패키지 운영 중 오류가 발생했을 때의 처리 상황에 대한 예를 구현해 봅니다.

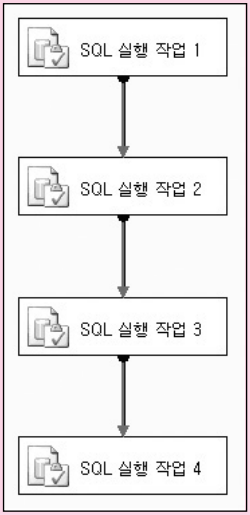
1. 빈 패키지 파일을 하나 추가한 후, **연결 관리자**에서 **OLE DB** 연결을 하나 추가합니다. 이 때, 서버 및 DB는 임의로 지정합니다.

2. 도구 상자에서 네 개의 **SQL 실행 작업**을 추가한 후, 이름을 "**SQL 실행 작업 1**", …, "**SQL 실행 작업 4**" 로 변경한 후, 각각의 **SQL 실행 작업 편집기**에서 다음과 같이 설정합니다.

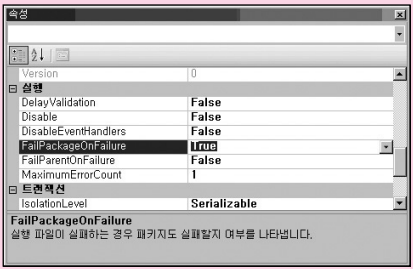
Connection	- *〈1에서 추가한 OLE DB 연결〉*
SQLStatement	- SELECT 1/10



3. 네 개의 작업들을 차례대로 연결합니다.

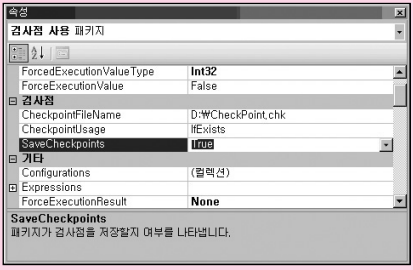


4. 네 개의 작업들을 모두 선택한 후, 속성에서 FailPackageOnFailure 의 값을 True로 설정합니다. 각 작업들을 선택해서 하나씩 변경해도 되지만, 작업의 수가 많은 경우, 이와 같이 일괄적으로 변경할 수도 있습니다.



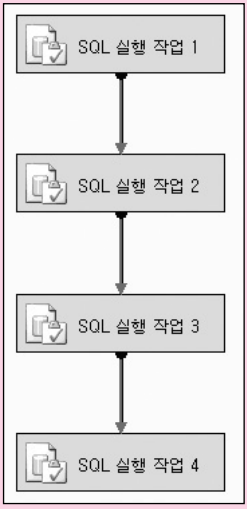
5. **제어 흐름 영역**의 빈 곳을 선택한 후, 속성 창에서 다음과 같이 설정합니다.

CheckpointFileName - D:₩CheckPoint.chk
CheckpointUsage - IFExists
SaveCheckpoints - True



6. **[테스트 시나리오 1] 정상인 경우**

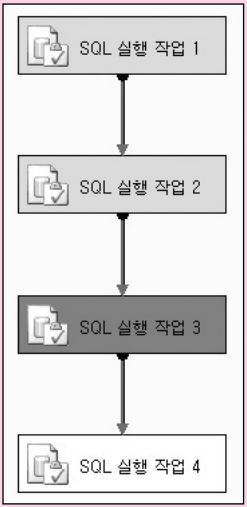
패키지를 수행한 후, 정상적으로 수행되는지 확인합니다.



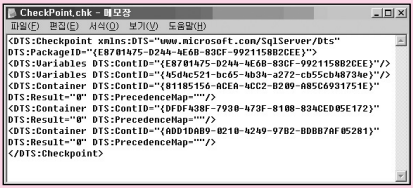
7. **[테스트 시나리오 2] 임의 오류 발생**

SQL 실행 작업 3의 SQLStatement를 다음과 같이 변경한 후, 패키지를 실행합니다.

SQLStatement	- SELECT 1/0

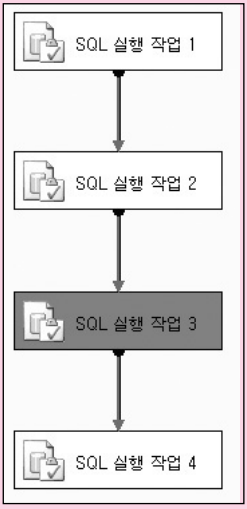


SQL 실행 작업 3에서 작업이 실패가 되었으며, D:₩CheckPoint.chk 라는 이름의 Checkpoint 파일이 생성되었습니다.



8. **[테스트 시나리오 2'] 오류 발생 후, 재 수행**

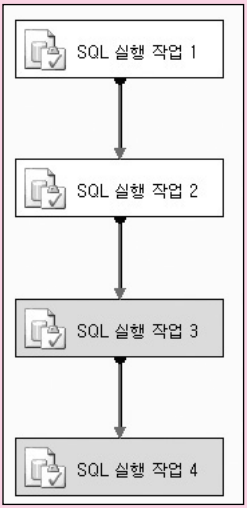
오류가 발생한 부분을 수정하지 않고 패키지를 다시 수행합니다.



SQL 실행 작업 3에서 작업이 실패한 후, 이 정보가 CheckPoint.chk 파일에 저장이 되어 있으며, 이전 작업들은 처리되었기 때문에 다시 수행하더라도 **SQL 실행 작업** 1 및 **SQL 실행 작업** 2는 다시 실행되지 않고 **SQL 실행 작업** 3부터 실행됩니다.

9. **[테스트 시나리오 3] 오류 수정 후, 재 수행**

SQL 실행 작업 3의 SQLStatement를 다시 SELECT 1/10 으로 수정한 후, 패키지를 실행합니다.

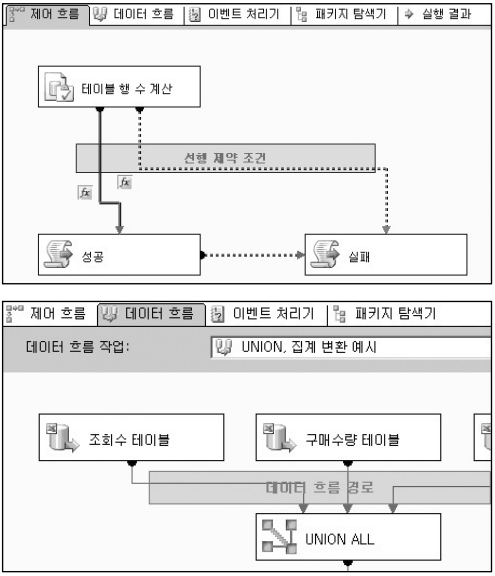


작업이 실패한 **SQL 실행 작업** 3부터 정상적으로 나머지 작업들이 수행됩니다. 패키지가 정상적으로 수행되면 자동으로 Checkpoint 파일은 삭제됩니다. 이제 다시 패키지를 수행하면 정상적으로 **SQL 실행 작업** 1부터 수행됩니다.

선행 제약 조건

선행 제약 조건은 이전 작업의 수행 결과를 이용하여 다음 작업의 수행을 제어하는 기능을 하는 연결 경로입니다.

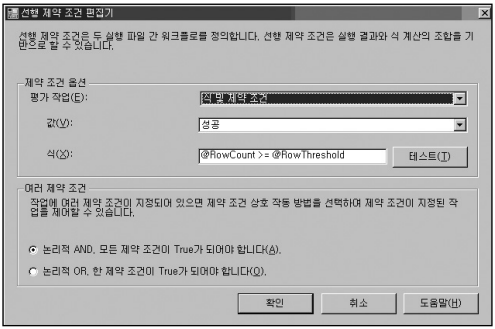
SSIS에는 **제어 흐름**과 **데이터 흐름**에 유사한 형태의 경로가 있습니다.



제어 흐름 영역에서의 경로를 **선행 제약 조건**이라 하며, **데이터 흐름** 영역에서의 경로는 **데이터 흐름 경로**라 합니다.

SSIS의 **선행 제약 조건**은 DTS에서와 마찬가지로 이전 작업이 성공일 때 수행하도록 할 경우에는 녹색 선, 실패일 때 수행하도록 할 경우에는 적색 선, 성공 이나 실패의 여부에 상관없이 완료되었을 때 수행하도록 할 경우에는 파란색 선으로 표시됩니다. 하지만, 이러한 **제약 조건**과 더불어 식을 이용하여 판단하는 기능이나 **여러 제약 조건**에 대한 처리 기능이 추가되었습니다.

만약 선행 작업이 데이터를 변환하는 작업일 경우, 데이터 변환 작업의 성공 여부뿐만 아니라, 변환된 행 수에 따라 이 후 작업의 실행 여부를 설정해야 할 경우가 있습니다. 즉, 변환된 행 수가 최소한 100건 이상인 경우에만 다음 작업을 수행하도록 할 경우, **[성공]**이라는 **제약 조건**과 **[행 수가 100보다 커야 한다]**라는 식을 동시에 설정하여 다음 작업 실행 여부를 판단하도록 설정할 수 있습니다.



제어 흐름 영역에서의 연결 선을 더블 클릭하면 위와 같은 **선행 제약 조건 편집기**가 나타납니다.

평가 작업(E)은 **식**과 **제약 조건**에 대한 유형을 설정합니다.

- **제약 조건** - 단순히 이전 작업의 수행 결과가 성공, 실패, 완료인지에 따라 다음 작업의 실행 여부가 결정되도록 설정하는 유형입니다. 이 옵션을 선택할 경우, 자동으로 아래의 **식(X)**부분은 비활성화 됩니다.

- **식** - 이전 작업의 성공, 실패, 완료 여부와는 상관없이 **식(X)** 부분에서 정의된 조건식의 여부에 따라 다음 작업의 실행 여부가 결정되도록 설정하는 유형입니다.

- **식 및 제약 조건** - 이전 작업의 수행 결과와 조건식이 모두 만족하는 경우에만 다음 작업이 실행되도록 설정하는 유형입니다. 예를 들어, 위의 그림에서 앞 부분의 작업이 수행되면 행 수가 **[사용자::RowCount]**라는 SSIS 내의 사용자 변수에 저장됩니다. 이 경우, 앞 부분의 작업 수행 여부도 성공이어야 하며, RowCount에 저장된 값이 RowThreshold라는 변수에 저장된 값보다 크거나 같은 경우에만 다음 단계를 실행하도록 설정한 것입니다.

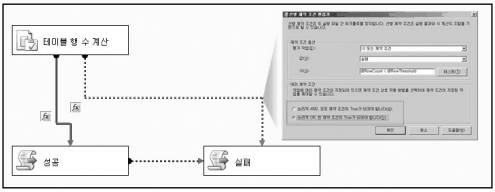
- **식 또는 제약 조건** - 이전 작업의 수행 결과 또는 조건식 중 하나라도 만족하는 경우, 다음 작업이 실행되도록 설정하는 유형입니다.

식(X)에서 사용자 변수는 위에서와 같이 **@변수명** 형태로 설정해야 합니다. 조건식을 작성한 후, **테스트(T)**를 이용하여 작성한 조건식이 유효한지를 확인할 수 있습니다.

여러 제약 조건 또한 SSIS에서 새롭게 추가된 기능입니다. 예를 들어 하나의 작업에 여러 선행 작업이 설정되어 있을 경우,

모든 선행 작업의 제약 조건이 모두 True가 되어야 다음 작업을 수행하도록 할지,
(=논리적 AND)
또는
여러 작업들 중 하나라도 만족하면 다음 작업을 수행하도록 할지
(=논리적 OR)
를 설정합니다.

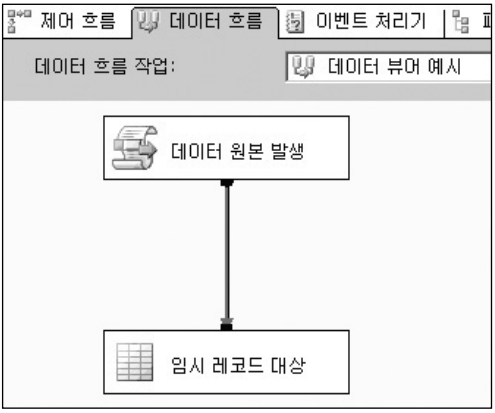
다음 그림에서 **[실패]**라는 스크립트 작업에 대해 **[테이블 행 수 계산]**과 **[성공]** 스크립트 작업이 선행 작업으로 설정되어 있습니다. 이 때 **[테이블 행 수 계산]** 작업이 실패하거나(OR), **[성공]** 스크립트 작업이 성공적으로 수행되는 경우 **[실패]** 스크립트 작업이 수행되도록 설정한 형태입니다.



논리적 AND인 경우에는 실선으로 표시되며, **논리적** OR인 경우 위의 그림과 같이 점선으로 표시됩니다.

데이터 흐름 경로

데이터 흐름 경로는 **데이터 흐름** 영역 내에서 변환과 변환, 또는 원본과 변환, 변환과 대상간의 연결을 시켜주는 경로입니다. 앞부분에서 소개한 **제어 흐름** 영역의 **선행 제약 조건**에서는 여러 가지 옵션을 설정할 수 있었던 것에 비해 **데이터 흐름 경로**에서는 특별한 설정은 없습니다. 단지 데이터가 변환되어 처리되는 과정에서의 열 유형이나 열 길이와 같은 메타 데이터를 확인할 수 있으며, **데이터 뷰어** 라는 기능을 이용하여 개발 또는 디버깅 단계에서 처리되는 데이터를 쉽게 확인할 수 있는 기능만을 제공합니다.

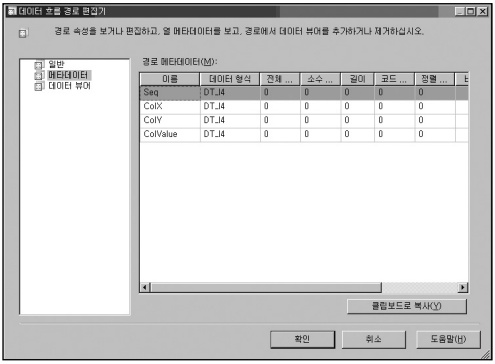


데이터 흐름 경로에는 정상적인 변환 결과를 출력하는 경로(녹색)와 오류 정보를 출력하는 경로(적색)가 있습니다. **오류 출력**을 추가한 경우, 원본 또는 변환의 **오류 구성 설정**에서 오류가 발생할 열에 대해 **행 리디렉션**을 설정해야 합니다.

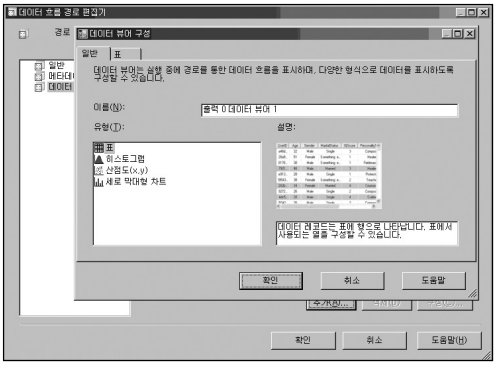
데이터 흐름 경로는 세 개의 탭으로 구성되어 있습니다.

일반 탭에서는 PathAnnotation을 이용해서 경로의 설명을 표시할 방법을 설정합니다.

메타데이터 탭에서는 현재의 데이터 흐름 경로를 지나는 데이터의 유형이나 자릿수 등과 같은 메타 데이터를 확인할 수 있습니다.



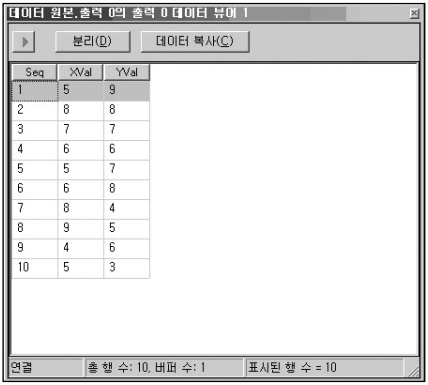
데이터 뷰어 탭에서는 **데이터 흐름 경로**를 지나는 데이터를 확인할 수 있는 뷰어(Viewer)를 설정할 수 있습니다. **데이터 뷰어**는 패키지를 개발하거나 디버깅 하는 단계에서만 사용되며 DTEXEC, DTEXECUI 또는 SQL Server 에이전트 등을 이용하는 실제 운영 환경에서 수행될 때에는 작동하지 않습니다.



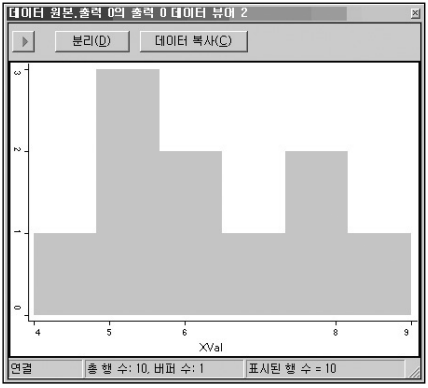
하나의 **데이터 흐름 경로**에는 여러 개의 **데이터 뷰어**를 설정할 수 있습니다.
데이터 뷰어는 **표, 히스토그램, 산점도**(x,y), **세로 막대형 차트** 등 네 가지의 형태가 있습니다. 단순히 처리되는 데이터의 형태를 표 형태로 보고자 할 때에는 **표** 유형을 사용하며, 데이터의 분포도 등을 확인하고자 할 때에는 **산점도**(x,y) 또는 **세로 막대형 차트** 유형을 사용할 수 있습니다.

다음은 각 유형별 출력 예입니다.

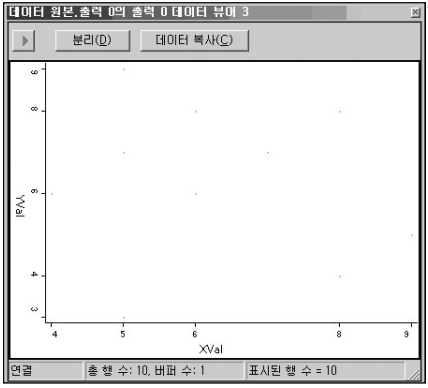
- **표** - 전체 열을 출력하도록 설정한 경우입니다. 확인이 필요한 열만 출력하도록 설정할 수도 있습니다.



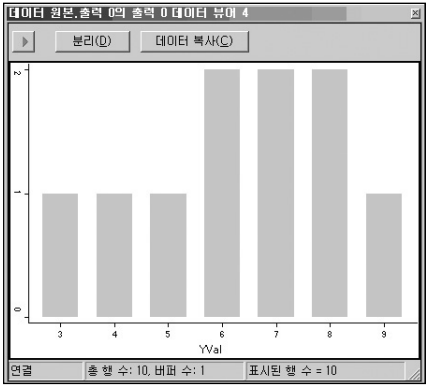
- **히스토그램** - XVal 열에 대한 히스토그램을 설정한 예입니다.



- **산점도(x,y)** - XVal, YVal열에 대해 산점도를 설정한 예입니다.



- **세로 막대형 차트** - YVal열에 대해 차트를 설정한 예입니다.



데이터 뷰어를 통해 한 번에 출력되는 데이터의 양은 데이터 흐름 작업 영역의 속성 중 DefaultBufferMaxRows 및 DefaultBufferSize의 값에 따라 달라집니다. 예를 들어, 전체 처리되는 데이터가 1,000,000건이더라도 DefaultBufferMaxRows의 값이 10,000으로 설정되어 있다면 데이터 뷰어에서 조회되는 데이터는 10,000개씩 나누어 출력됩니다. 또한, DefaultBufferMaxRows의 값이 10,000이더라도 데이터 한 행의 크기가 커서 10,000개의 행이 처리되기 이전에 전체 데이터의 크기가 DefaultBufferSize를 넘어서는 경우, 해당 크기만큼의 데이터만 출력됩니다. **데이터 뷰어** 창의 왼쪽 상단에 있는 ▶ 버튼을 클릭하면 다음 처리 버퍼의 내용이 출력되며, 어느 정도 데이터 검증이 끝난 경우, **분리(D)** 버튼을 클릭하면 데이터 확인 작업을 종료하고 데이터 처리를 계속 진행하게 됩니다. **데이터 복사(C)**를 클릭하면 현재 데이터 뷰어 창에 나타난 데이터 또는 그래프에서 이용되는 데이터를 클립보드로 복사합니다.

SSIS는 변환 작업 간의 데이터 유형을 엄격히 일치시키도록 관리합니다. 이는 개발자의 실수로 인해 원하지 않는 데이터 변형이 발생되어 데이터가 잘못되는 현상을 막기 위한 목적입니다. **데이터 흐름 경로**를 이용하여 변환과 변환, 변환과 대상 사이의 열 정보와 같은 메타 데이터를 쉽게 확인할 수 있습니다. 또한, 변환 작업이 정상적으로 처리하는지를 직접 확인할 수 있기 때문에 개발 및 디버깅의 시간도 많이 단축됩니다.

이벤트 처리 및 디버깅, 오류 처리

이벤트 및 이벤트 처리기

SSIS의 개발 환경이 Visual Studio에 통합되면서 패키지 개발 단계에서도 Visual Studio에서 제공하는 다양한 부가 기능들을 이용할 수 있게 되었습니다. 그 중에서 **이벤트** 및 **이벤트 처리기** 기능은 패키지 개발 및 운영 단계에서 유용하게 사용할 수 있는 기능입니다.

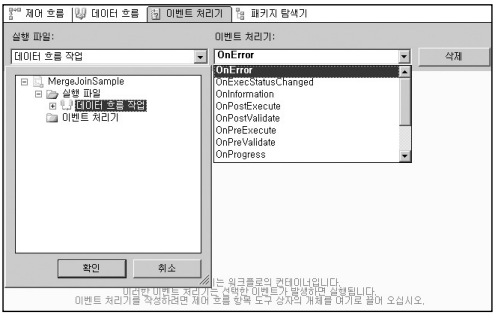
VB.NET이나 VC++ 등과 같은 어플리케이션의 경우, 소스 내의 클래스 또는 함수, 개체 등이 호출되어 실행될 때 다양한 이벤트가 발생합니다. 예를 들어 클래스가 생성되거나 실행될 때에는 OnCreate 또는 OnExecute 등과 같은 이벤트가 발생되며 오류가 발생하면 OnError 이벤트가 발생합니다.

SSIS 패키지에서도 이와 같은 형태로 패키지 및 패키지 내의 개별 작업 개체들이 실행될 때 **이벤트**가 발생합니다. 예를 들어, 패키지에서 지정한 위치에서 오류가 발생하면 OnError 이벤트가 발생되며, 오류와 관련된 시스템 변수에 저장된 정보들을 이용하여 오류 처리 작업을 수행하도록 할 수 있습니다.

다음은 SSIS 패키지에서 발생하는 이벤트의 종류입니다.

종류	이벤트
OnError	이 이벤트는 오류가 발생할 때 실행 개체에 의해 발생합니다.
OnExecStatusChanged	이 이벤트는 실행 상태가 변경될 때 실행 개체에 의해 발생합니다.
OnInformation	이 이벤트는 정보 보고를 위한 실행 개체의 유효성 검사 및 실행 중에 발생합니다. 이 이벤트에는 오류 또는 경고를 제외한 정보만 포함됩니다.

종류	이벤트
OnPostExecute	이 이벤트는 실행을 마친 바로 다음 실행 개체에 의해 발생합니다.
OnPostValidate	이 이벤트는 유효성 검사가 완료될 때 실행 개체에 의해 발생합니다.
OnPreExecute	이 이벤트는 실행되기 바로 전에 실행 개체에 의해 발생합니다.
OnPreValidate	이 이벤트는 유효성 검사가 시작될 때 실행 개체에 의해 발생합니다.
OnProgress	이 이벤트는 실행 개체의 진행 상태를 측정할 수 있는 경우 실행 개체에 의해 발생합니다.
OnQueryCancel	이 이벤트는 실행 중지 시기를 결정하기 위해 실행 개체에 의해 발생합니다.
OnTaskFailed	이 이벤트는 작업이 실패할 때 해당 작업에 의해 발생합니다.
OnVariable ValueChanged	이 이벤트는 변수 값이 변경될 때 실행 개체에 의해 발생합니다. 이 이벤트는 변수가 정의되는 실행 개체에 의해 발생합니다. 변수에 대한 RaiseChangeEvent 속성을 False로 설정한 경우에는 이 이벤트가 발생하지 않습니다.
OnWarning	이 이벤트는 경고가 발생할 때 실행 개체에 의해 발생합니다



SSIS 개발 화면의 상단에 있는 **이벤트 처리기** 부분에서 이벤트에 대한 처리 작업을 수행할 수 있습니다. 왼쪽의 **실행 파일** 부분에서는 이벤트 처리기가 작동될 범위를 지정하며, **오른쪽의 이벤트 처리기** 부분에서는 처리할 이벤트를 선택할 수 있습니다.

위의 그림의 경우, **[데이터 흐름 작업]**에 대한 OnError 이벤트를 처리하는 처리기를 선택한 것입니다.

각 이벤트 처리 수준은 계층으로 관리됩니다. 만약, 전체 패키지 수준(위의 그림에서는 MergeJoinSample)으로 OnError 이벤트 처리기를 설정하였다면, 이 이벤트 처리기는 패키지에서 발생하는 모든 오류 이벤트에 대해 수행합니다. 위의 그림과 같이 **[데이터 흐름 작업]**에 대한 OnError 이벤트 처리기를 설정한 경우에는 **[데이터 흐름 작업]**에 포함된 여러 변환 단계에서 발생하는 OnError 이벤트에 대해서만 처리하게 됩니다.

이벤트 처리기 작업 영역에는 **제어 흐름** 영역이나 **데이터 흐름** 영역보다 더 많은 더 많은 시스템 변수가 있습니다. 시스템과 관련된 기본적인 정보 외에도 해당 이벤트 정보를 저장하기 위한 변수들이 추가되어 있습니다. 예를 들어, OnError 이벤트 처리기에는 어떤 작업 개체 ID에서 오류가 발생되었으며([시스템::TaskID]), 오류 코드는 무엇이며([시스템::ErrorCode]), 오류 처리기가 시작된 시간이 언제인지([시스템::EventHandlerStartTime]) 등의 정보를 가지고 있는 변수들이 있습니다.

개발자는 시스템 변수에 저장된 값을 적절히 이용하여 이벤트 처리 프로세스를 구성할 수 있습니다.

이벤트 처리기를 이용한 처리 프로세스의 예는 다음과 같습니다.

1. 패키지 내에서 오류가 발생되었을 때 이를 처리하기 위한 용도로 사용할 수 있습니다. 예를 들어, FTP **작업**을 이용하여 파일을 가져오는 작업에서 가져올 파일이 없어서 실패할 경우, OnError 이벤트 처리기에서 파일과 관련된 여러 사항들을 확인한 후 관리자에게 통보하도록 구성할 수 있습니다.

2. OnPreExecute, OnPostExecute 이벤트 처리기를 이용하여 패키지 전체 및 개별 작업의 시작 시간 및 종료 시간을 기록하도록 할 수 있습니다. 패키지 전체 수준에서만 이러한 작업을 구성해 놓으면, 패키지 내에 작업 개체들이 추가되더라도 별도의 추가 작업 없이도 시작 및 종료 시간 정보를 관리할 수 있습니다.

3. OnVariableValueChanged 이벤트 처리기를 이용하여 패키지 내의 사용되는 변수의 값이 변경되는 사항을 관리할 수 있습니다. 변수를 사용하는 패키지를 구성하였을 때, 어떠한 변수 값을 이용하여 패키지가 처리되었는지를 관리하는 데 사용할 수 있습니다.

4. OnProgress 이벤트 처리기를 이용하여 각 작업의 실행중인 상태를 모니터링 할 수 있습니다. 예를 들어 데이터 변환 작업에 대해 이벤트 처리기를 구성한 경우, 현재 처리된 데이터의 처리량을 파악할 수 있습니다.

중단점

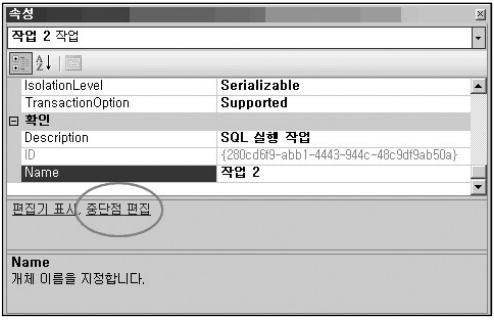
Visual Studio 환경에서 응용 프로그램을 개발할 때, 프로그램의 특정 위치까지만 수행되도록 설정할 때 **중단점**(Breakpoint)을 이용하게 됩니다. 개발중인 프로그램에서 중단점으로 지정한 부분까지 실행시킨 후, 변수의 값이 변경되는 상태나 코드의 처리 순서 등을 확인할 수 있습니다.

SSIS 패키지를 개발할 때에도 이와 동일한 방식으로 이용할 수 있습니다.
우선 다른 응용 프로그램에서와 마찬가지로 **스크립트 작업**이나 **스크립트 구성 요소** 내에서 **중단점**을 설정하여 패키지가 실행될 때 스크립트 작업의 지정한 부분에서 잠시 중지하도록 설정할 수 있습니다.

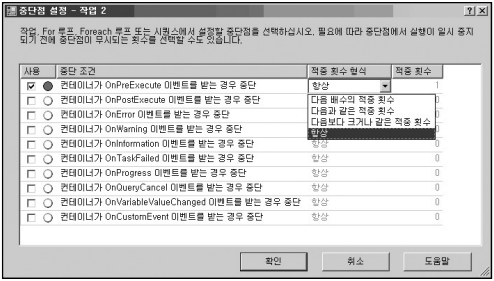
스크립트 개발 환경인 VSA(Visual Studio for Applications)의 **디버그**(D) 메뉴 내에 있는 중단점 관련 메뉴를 이용하여 원하는 부분에 중단점을 설정하고 해제할 수 있습니다.

스크립트 작업뿐만 아니라 **제어 흐름** 영역에 있는 작업 개체의 **이벤트**에 대해서도 중단점을 설정할 수 있습니다.

제어 흐름 영역에 있는 여러 작업 개체들 중 중단점을 설정할 작업을 선택하면 오른쪽의 속성 창 부분에 다음과 같은 항목이 나타납니다.



중단점 편집을 클릭하면 다음과 같은 **중단점 설정** 창이 나타나며 원하는 조건에 맞게 설정할 수 있습니다. 기본적으로 해당 작업에서 발생할 수 있는 모든 이벤트에 대해 중단점을 설정할 수 있으며, **적중 횟수 형식**과 **적중 횟수**를 이용하여 실행을 중단 시키는 세부 조건을 지정할 수도 있습니다.

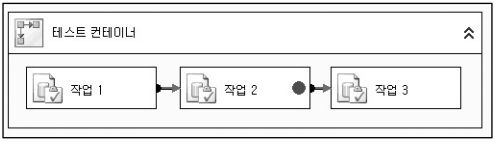


적중 횟수 형식

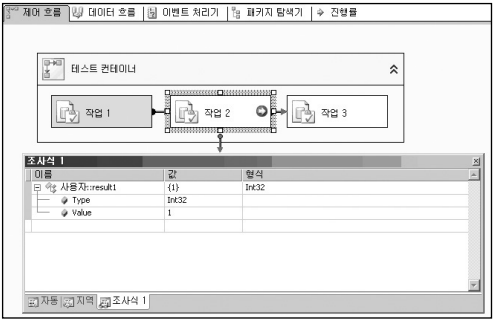
- **다음 배수의 적중 횟수** - 해당 이벤트가 **적중 횟수**의 배수가 되면 중단합니다. 예를 들어 **적중 횟수**를 3으로 설정할 경우 해당 개체의 이벤트가 3의 배수 번째일 때마다 중단하게 됩니다.
- **다음과 같은 적중 횟수** - 해당 이벤트가 **적중 횟수**가 될 때 중단합니다.
- **다음보다 크거나 같은 적중 횟수** - 해당 이벤트가 **적중 횟수**보다 크거나 같은 경우 중단합니다.
- **항상** - **적중 횟수**에 상관없이 해당 이벤트가 발생될 때마다 중단합니다.

예를 들어, 30회를 반복 수행하도록 설정된 **For 루프 컨테이너** 내에 **작업** 2라는 **SQL 실행 작업**이 포함되어 있으며, OnPreExecute 이벤트에 대해 **적중 횟수** 3회, **적중 횟수 형식**은 **다음 배수의 적중 횟수**로 설정한 경우 SQL 실행 작업이 3회째, 6회째, 9회째 등과 같이 3의 배수 번째일 때마다 중단하게 됩니다.

중단점을 설정하면 다음 그림과 같이 해당 작업에는 붉은 점이 표시됩니다.



또한 중단된 상태에서 **조사식** 창을 이용하면 현재 시점에서의 변수의 값을 확인할 수 있습니다.

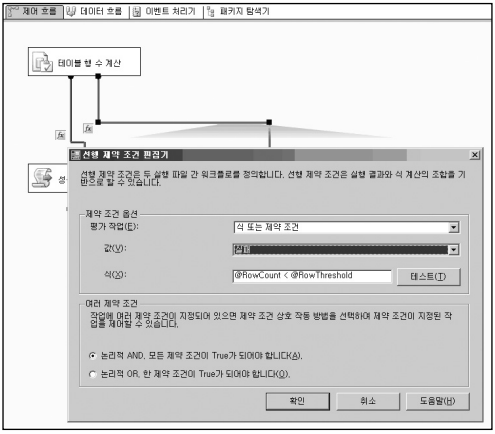


위의 그림은 현재 작업 2에서 **OnPreExecute** 이벤트가 발생하여 작업이 잠시 중단된 상태이며, 이 때 변수인 **[사용자::result1]**의 값은 1이라는 것을 나타냅니다.

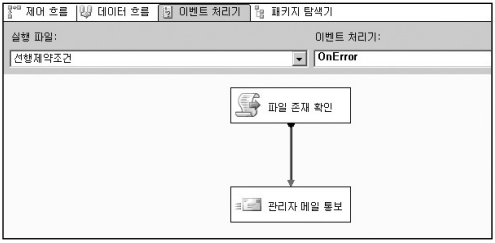
오류 출력

SSIS에서는 오류를 처리하는 방식이 크게 세가지가 있습니다.

- 첫 번째는 **제어 흐름** 영역의 각 작업들이 실패로 처리될 경우에 대한 설정으로 **선행 제약 조건**을 실패로 설정한 후 작업이 실패가 되었을 때 처리할 다음 작업과 연결시키는 방식입니다. 이 때, 두 작업을 연결하는 연결선(선행 제약 조건)은 적색으로 표시됩니다.



- 두 번째는 **이벤트 처리기**의 OnError 이벤트를 사용하는 방식입니다. 패키지 내의 작업 개체에서 오류가 발생하면 OnError 이벤트가 발생하며, 이벤트 처리기의 OnError 처리 영역에 지정한 작업들이 수행되도록 구성하는 방식입니다.



- 마지막으로 **데이터 흐름** 영역 내에서 원본, 변환, 대상 등에서 **오류 출력**을 설정하는 방식입니다.

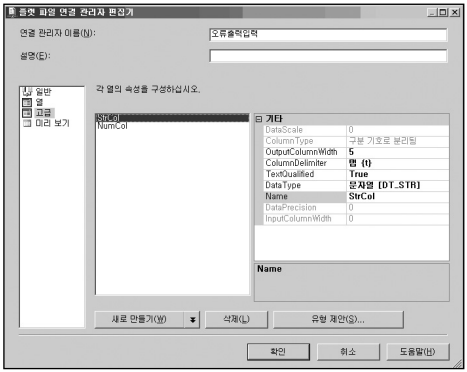
앞의 두 방식은 **선행 제약 조건** 및 **이벤트 처리** 부분을 참고하기 바랍니다.

오류 출력은 **데이터 흐름 영역**에서만 설정할 수 있습니다. 데이터를 처리하는 과정에서 발생되는 오류에 대한 처리 방식을 설정하는 것이며, 원본, 변환, 대상 등 데이터 흐름과 관련된 대부분의 구성 요소에서 설정 가능합니다. 하지만, **행 개수** 또는 **멀티 캐스트** 등과 같이 오류가 발생하지 않는 작업에서는 오류 출력 기능이 포함되어 있지 않습니다.

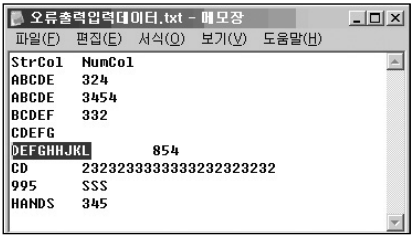
데이터 흐름 작업 수행 시 발생할 수 있는 오류는 다음과 같습니다.

- **데이터 원본 오류**

  SSIS에서는 텍스트 파일이나 테이블 형식의 데이터에 대해 **연결 관리자**에서 지정한 연결 방식을 이용하여 제어하게 됩니다. 이 데이터를 데이터 흐름 영역에서 이용할 경우, **DataReader 원본, OLE DB 원본, 플랫 파일 원본** 등과 같은 데이터 원본 개체를 이용하여 읽어오게 됩니다. 읽어오는 데이터에 대한 데이터 유형이나 길이 등은 연결 관리자에 있는 연결에서 설정이 되며, 데이터 원본에서는 이 연결 설정을 바탕으로 실제 데이터에 접근합니다. 이 경우, 연결에서 정의된 열의 정보가 실제 데이터와 다른 경우 오류가 발생할 수 있습니다. 예를 들어 다음과 같은 경우입니다.



플랫 파일 연결에서는 StrCol의 열을 **문자열**(DT_STR) 형의 **5자리**로 지정을 하였지만 실제 입력되는 데이터는 다음과 같이 5자리를 넘을 수 있습니다.



이 경우, **잘림 오류**가 발생하며 오류 처리가 설정되어 있지 않은 경우에는 데이터 흐름 작업이 중단됩니다. 원본에서 발생될 수 있는 오류 중 대부분은 이와 같이 데이터 유형이나 길이에 따른 오류입니다.

- **데이터 변환 오류**

  병합 변환이나 **데이터 변환**, **조회**, **파생 열** 등과 같은 변환 작업을 수행할 때 발생할 수 있는 오류입니다. 예를 들어 String형의 데이터를 Int형으로 변경한 후 새로운 열로 출력시키는 파생 열 변환의 경우, 입력 데이터가 "abcd"와 같이 숫자로 변환을 할 수 없을 때 발생할 수 있습니다.

- **데이터 대상 오류**

  데이터 원본 또는 변환 개체에서 대상으로 지정된 데이터 연결에 저장할 때 발생할 수 있는 오류입니다. 예를 들어 Not Null로 설정된 Int형 열을 포함하고 있는 테이블에 데이터를 저장할 경우, 저장될 데이터가 Null 이거나 Int형이 아닌 경우 데이터 대상 단계에서 오류가 발생합니다.

데이터 흐름 영역의 각 개체에서 오류 출력을 구성하는 방식은 모두 동일합니다. 원본이나 대상, 또는 변환의 편집기에 있는 **오류 출력** 탭을 이용하거나, 각 개체를 적색 선으로 연결하면 오류 출력을 구성할 수 있는 설정 창이 나타납니다.

오류 출력 구성은 다섯 개의 열로 구성되어 있습니다.
입력 또는 출력은 현재 오류 구성의 작업 이름에 대한 입력 또는 출력을 나타냅니다. 오류 출력 구성은 데이터의 각 열 별로 설정을 할 수 있습니다. 중요하지 않은 열에서 오류가 발생할 경우에는 **오류 무시**로 설정을 하여 오류가 발생하더라도 처리 작업을 계속 진행하도록 할 수 있습니다.

SSIS에서는 데이터 처리 중 발생되는 오류와 데이터 잘림 현상을 구분하여 처리합니다. 데이터 오류에 대한 처리 방식은 **오류** 열에서 설정하며, 데이터 잘림에 대한 처리 방식은 **잘림** 열에서 설정합니다.

- **오류 무시** - 오류 또는 데이터 잘림 현상이 발생되더라도 이를 무시하고 작업을 계속 진행합니다.
- **행 리디렉션** - 오류 또는 데이터 잘림 현상이 발생할 경우, 해당 오류 데이터를 별도의 경로(적색 선)로 출력합니다. **행 리디렉션**으로 설정한 경우에는 반드시 적색 선을 다른 작업 개체와 연결시켜야 합니다.
- **구성 요소 실패** - 오류 또는 데이터 잘림 현상이 발생할 경우 작업을 실패로 처리합니다.

설정해야 할 열이 여러 개이며 모두 동일한 설정을 하고자 할 경우에는 여러 열을 선택한 후 아래에 있는 **이 값을 선택한 셀에 설정**(S) 기능을 이용하여 한 번에 설정할 수 있습니다.

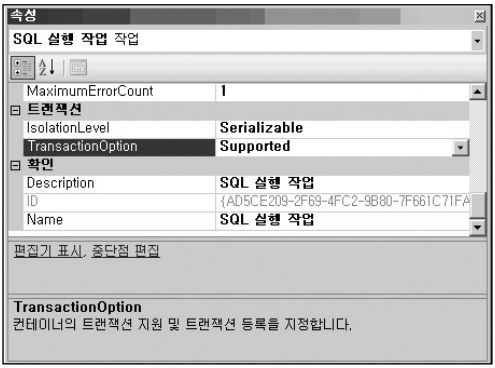
행 리디렉션 방식으로 오류가 처리될 때에는 **오류 코드**(ErrorCode)와 **오류가 발생된 열의 번호**(ErrorColumn) 정보가 추가되어 출력됩니다.

트랜잭션 및 패키지 보안

트랜잭션

SQL Server에서와 같이 SSIS에서도 데이터베이스 작업에 대해 트랜잭션을 지원합니다. 여러 작업들이 하나의 트랜잭션으로 설정된 경우, 모든 작업이 성공할 때에만 커밋되고 그렇지 않을 때에는 작업들이 모두 롤백되도록 설정할 수 있습니다. 이는 패키지의 전체 작업 또는 일부 작업들에 대해 트랜잭션을 이용하여 데이터 무결성을 관리할 수 있는 기능입니다.

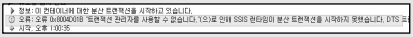
트랜잭션은 각 작업 개체 또는 컨테이너, 패키지의 속성 중 TransactionOption을 이용하여 설정할 수 있습니다.



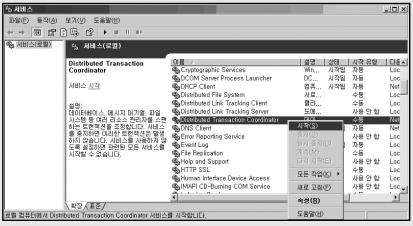
[참고] 트랜잭션을 사용하기 위한 설정 - DTC(Distributed Transaction Coordinator) 실행

SSIS에서 트랜잭션을 사용하기 위해서는 패키지가 수행되는 서버 또는 PC에 MSDTC 서비스가 실행되고 있어야 합니다.

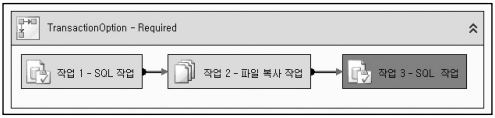
DTC 서비스가 실행되고 있지 않은 상태에서 트랜잭션 설정을 적용한 패키지를 실행할 경우 다음과 같은 오류 메시지가 발생합니다.



DTC 서비스는 [제어판] → [관리도구] → [서비스] 내에서 Distributed Transaction Coordinator 서비스를 시작하면 실행됩니다.



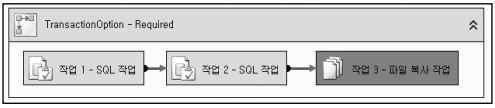
SSIS의 모든 작업 개체 및 컨테이너의 속성에서 TransactionOption을 설정할 수 있지만, 트랜잭션은 데이터베이스와 관련된 작업 개체에서만 적용됩니다. 예를 들어, 아래 그림과 같이 **작업 1**과 작업 3은 **SQL 실행 작업**이며 **작업 2**는 파일 복사를 수행하는 **파일 시스템 작업**인 경우, **작업 3**에서 오류가 발생하여 해당 컨테이너가 커밋을 하지 못할 경우 **작업 1**인 SQL **작업**은 롤백되고 데이터 처리가 되지 않지만, **작업 2** 파일 복사 작업은 롤백되지 않습니다. 즉, 이미 복사된 파일이 삭제되지는 않습니다.



(작업 1, 작업 2, 작업 3 | TransactionOption : Supported)

하지만, 데이터베이스와 관련된 작업이 아닌 경우에도 트랜잭션이 필요한 이유는 다음과 같습니다.

작업 1과 **작업** 2는 데이터베이스와 관련된 작업이며, **작업** 3은 파일을 복사하는 작업입니다. 세 개의 작업이 트랜잭션을 사용하는 하나의 컨테이너에서 수행된다고 할 때, 데이터베이스 작업이 아닌 **작업** 3에서 실패가 나더라도 **작업** 1과 **작업** 2는 롤백됩니다.



(작업 1, 작업 2, 작업 3 | TransactionOption : Supported)

이는 **작업** 3이 **작업** 1이나 **작업** 2와 같이 트랜잭션의 처리 결과에 대해 커밋이나 롤백할 작업은 아니지만 트랜잭션에 영향을 미칠 수 있는 개체이기 때문입니다.

TransactionOption에서 설정할 수 있는 옵션은 다음과 같습니다.

- Required - 선택한 개체가 트랜잭션을 생성합니다. 개체란 패키지 또는 컨테이너, 개별 작업 개체를 말합니다. 만약 선택한 개체가 부모 컨테이너에 포함되어 있으며, 부모 컨테이너에서 트랜잭션이 생성되도록 설정되어 있는 경우에는 Supported 설정과 동일한 방식으로 부모 컨테이너의 트랜잭션에 참여합니다.
  만약 패키지는 트랜잭션을 생성하지 않는 Not Required로 설정되어 있더라도, 패키지 내에 있는 **시퀀스 컨테이너**의 트랜잭션 속성이 Required로 설정된 경우, 시퀀스 컨테

이너는 트랜잭션을 생성하며 이 컨테이너 내에 포함되는 작업들(Not Supported로 설정된 작업 제외)에는 트랜잭션이 적용됩니다.

- Supported - 선택한 개체가 새로운 트랜잭션을 생성하지는 않고 단지 부모 컨테이너의 트랜잭션에 참여만 합니다. 예를 들어, 패키지는 트랜잭션을 생성하는 Required로 설정이 되어 있으며 패키지 내에 포함된 세 개의 SQL 실행 작업들은 Supported로 설정되어 있는 경우, 각각의 SQL 실행 작업은 부모 컨테이너인 패키지의 트랜잭션에 참여하게 되며 세 작업 중 하나라도 실패하면 전체 작업이 롤백됩니다.

- Not Required - 선택한 개체가 새로운 트랜잭션을 생성하지도 않으며, 부모 컨테이너의 트랜잭션에 참여하지도 않습니다.

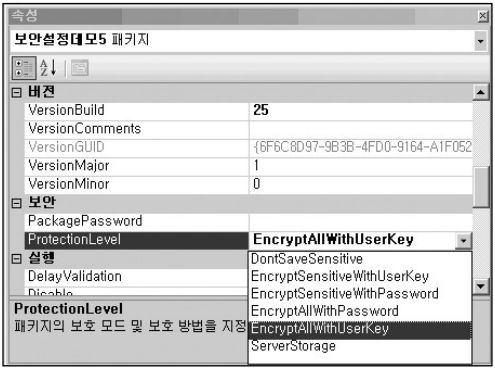
보안

SSIS에서는 강력한 패키지 보안 수준을 제공합니다. SSIS에서 제공하는 보안과 관련된 사항은 다음과 같습니다.

- 패키지의 ProtectionLevel 설정에 따라 패키지의 암호화를 설정할 수 있습니다. 데이터베이스 암호나 연결 문자열 등과 같은 중요한 데이터를 포함시키지 않도록 설정하거나 암호화 수준을 설정할 수 있습니다.

- ProtectionLevel 및 PackagePassword 속성을 이용하여 패키지의 보안을 설정할 수 있습니다. 암호를 이용하거나 개인 키를 이용하여 보안을 설정할 수 있습니다.

- 보안과 관련된 사항을 SQL Server가 담당하도록 지정할 수 있습니다. ProtectionLevel을 ServerStorage로 설정한 후, 패키지를 SQL Server의 msdb에 직접 저장하면 패키지의 보안 관련된 사항은 모두 SQL Server에서 관리하게 됩니다.



패키지의 속성 중 ProtectionLevel과 PackagePassword를 이용하여 보안 설정을 구성할 수 있습니다.

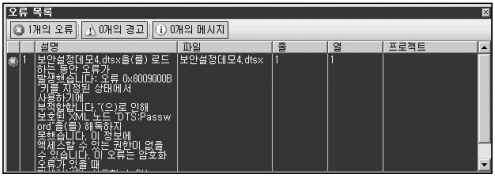
설정값	상세 설명
DontSaveSensitive (중요한 정보 저장 안 함)	패키지를 저장할 때 중요한 정보를 제외합니다. 이 보호 수준은 암호화는 사용하지 않지만 중요한 것으로 표시된 속성이 패키지로 저장되는 것을 방지하여 사용자가 중요한 데이터를 사용하지 못하도록 합니다. 이 설정은 동일한 사용자가 패키지를 저장하여 닫은 후, 다시 여는 경우라도 저장이 되지 않기 때문에 매번 패키지를 열 때마다 중요한 정보를 다시 지정해야 합니다.

설정값	상세 설명
EncryptAllWith Password (암호로 모두 암호화)	암호를 사용하여 전체 패키지를 암호화합니다. 패키지를 만들거나 내보낼 때 사용자가 입력한 암호를 사용하여 패키지를 암호화합니다. 사용자는 SSIS 디자이너에서 패키지를 열거나 dtexec 명령 프롬프트 유틸리티를 사용하여 패키지를 실행할 때 패키지 암호를 입력해야 합니다. 암호를 입력하지 않으면 패키지를 실행할 수 없습니다. 이 설정을 적용할 경우, PackagePassword 항목에 적절한 암호를 지정한 후 패키지를 저장해야 합니다.
EncryptAllWith UserKey (사용자 키로 모두 암호화)	사용자 프로필을 기반으로 하는 암호를 사용하여 전체 패키지를 암호화합니다. 동일한 프로필을 사용하는 동일한 사용자만 패키지를 로드할 수 있습니다. 패키지를 만들거나 내보낸 사용자를 기반으로 하는 키를 사용하여 패키지를 암호화합니다. 패키지를 만들거나 내보낸 사용자만 SSIS 디자이너에서 패키지를 열거나 dtexec 명령 프롬프트 유틸리티를 사용하여 패키지를 실행할 수 있습니다.
EncryptAllWith Password (암호로 중요한 정보 암호화)	암호를 사용하여 패키지 내의 중요한 정보를 암호화합니다. 이 암호화에는 DPAPI가 사용됩니다. 중요한 데이터는 패키지의 일부로 저장되지만 패키지를 만들거나 내보낼 때 사용자가 입력한 암호를 사용하여 암호화됩니다. SSIS 디자이너에서 패키지를 열려면 패키지 암호를 입력해야 합니다. 암호를 입력하지 않으면 패키지의 중요한 정보가 제외되며 패키지를 실행시키기 위해서는 사용자가 중요한 정보 부분에 값을 입력해야 합니다. 암호를 입력하지 않으면 패키지를 실행할 수 없습니다. 이 설정을 적용할 경우, PackagePassword 항목에 적절한 암호를 지정한 후 패키지를 저장해야 합니다.

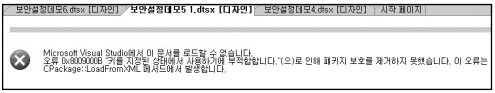
설정값	상세 설명
EncryptSensitive WithUserKey (사용자 키로 중요 정보 암호화)	현재 사용자를 기반으로 하는 키를 사용하여 패키지 내의 중요한 정보를 암호화합니다. 동일한 프로필을 사용하는 동일한 사용자만 패키지를 로드할 수 있습니다. 다른 사용자가 패키지를 여는 경우 중요한 정보는 빈칸으로 대체되므로 현재 사용자가 중요한 데이터에 새 값을 지정해야 합니다. 사용자가 패키지를 실행하려고 시도하는 경우 패키지 실행이 실패합니다. 이 암호화에는 DPAPI가 사용됩니다.
ServerStorage (암호화에 서버 저장소 사용)	SQL Server 데이터베이스 역할을 사용하여 전체 패키지를 보호합니다. 이 옵션은 패키지를 SQL Server msdb 데이터베이스에 저장할 때만 지원됩니다. Business Intelligence Development Studio에서 파일 시스템에 패키지를 저장하는 경우에는 지원되지 않습니다.

중요한 데이터란 DB 연결 정보 또는 암호, XML 노드 정보 등과 같은 연결 정보입니다.

만약 EntryptAllWithUserKey 또는 EncryptSensitiveWithUserKey로 설정된 패키지를 다른 사용자의 프로필 환경에서 사용할 경우 다음과 같은 오류가 발생합니다.



EncryptSensitiveWithUserKey 또는 EncryptSensitiveWithPassword로 설정된 패키지의 경우, 다른 사용자의 프로필 환경에서 열 경우 또는 암호가 틀릴 경우에는 중요한 데이터만 사용할 수 없으며, 나머지 패키지의 개체는 조회 가능합니다. 하지만, EncryptAllWith UserKey 또는 EncryptAllWithPassword 등으로 설정된 경우, 사용자 프로필이 다르거나 암호가 다르면 패키지의 조회도 불가능합니다.



로깅 및 배포

로깅

SSIS는 패키지가 실행될 때의 이벤트 정보를 기록할 수 있는 로깅 기능을 제공합니다. 패키지가 시작되는 시간 및 종료되는 시간, 오류가 발생했을 때의 정보, 패키지 내의 각 작업들의 수행 시간 등 패키지를 관리할 때 필요한 여러 가지의 기록을 관리할 수 있습니다. SSIS에서 발생되는 모든 종류의 이벤트에 대해 로그를 남길 수 있으며 각 이벤트에 대해 **스키마**라 불리는 열 정보에 이벤트와 관련된 사항들이 저장됩니다.

이벤트 정보에 대한 스키마 정보는 다음과 같습니다.

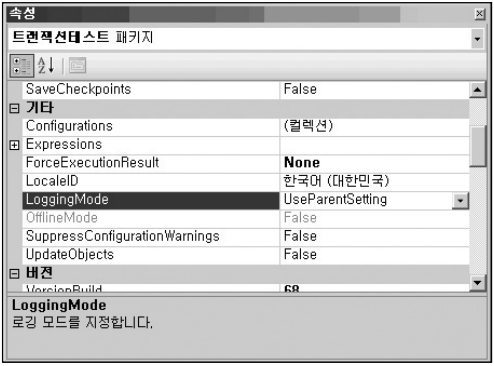
요소	설명
Computer	이벤트가 발생한 컴퓨터의 이름입니다.
Operator	패키지를 시작한 사용자의 ID입니다.
SourceName	이벤트가 발생한 컨테이너 또는 작업의 이름입니다.
SourceID	이벤트가 발생한 패키지, For Loop, Foreach Loop, 시퀀스 컨테이너 또는 작업의 고유 식별자입니다.
ExecutionID	패키지 실행 인스턴스의 GUID입니다.
MessageText	이벤트 항목과 관련된 메시지입니다.
DataBytes	이벤트 항목과 관련된 바이트 배열입니다. 이 필드의 의미는 로그 항목에 따라 다릅니다.

스키마 정보 외에도 이벤트 명, 이벤트 발생 시작 시간, 이벤트 종료 시간이 기록됩니다.

하나의 패키지 내에 있는 모든 작업에 대해 일괄적으로 로그를 남기도록 설정할 수도 있으며, 일부 작업 개체에 대해서만 로그를 남기도록 설정할 수도 있습니다.

로그 설정

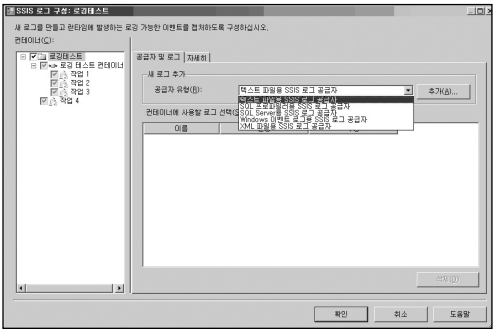
패키지 또는 각 개체의 속성에는 해당 개체에 대한 로깅 설정을 할 수 있는 LoggingMode라는 항목이 있습니다.



기본값은 UseParentSetting이며, 현재 작업 개체가 포함된 컨테이너의 로그 설정을 따른다는 의미입니다. LoggingMode 값을 Disable로 설정하면 패키지에서 로그를 남기도록 설정하더라도 로그를 발생시키지 않습니다.

로그 구성

로그를 구성하기 위해서는 BIDS 상단에 있는 SSIS(S) 메뉴에서 로깅(L)을 클릭합니다.



위의 그림과 같이 로그를 구성할 수 있는 창이 나타납니다. 왼쪽의 **컨테이너(C)** 부분에서는 패키지 내에서 로그를 남길 요소를 선택합니다. 상위 개체가 선택되면 자동으로 하위 개체는 로그를 남기게 되며 비활성화됩니다. 만약 일부 작업에 대해서만 로그를 남기고자 한다면 이 부분에서 해당 개체만 선택하면 됩니다.

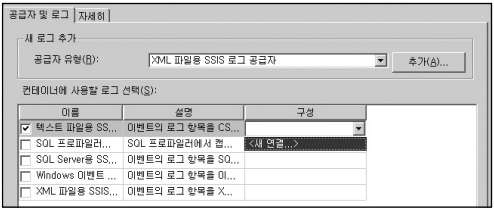
공급자 및 로그 탭에서 로그를 남길 유형을 설정할 수 있습니다. 위의 그림과 같이 텍스트 파일이나 프로파일러용 로그 파일, XML 파일 등 다양한 형태의 로그 유형을 설정할 수 있습니다.

자세히 탭에서는 로그를 남길 이벤트를 선택할 수 있으며, 아래에 있는 **고급(N)** ›› 버튼을 클릭하면 각 이벤트에 대해 저장할 열 정보(스키마 정보)를 선택할 수 있습니다.

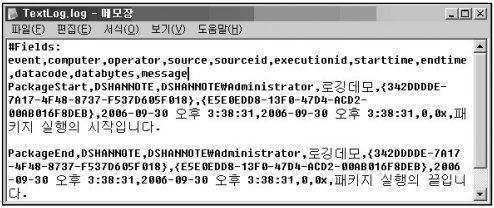
또한 **저장(S)**을 이용하여 현재 구성한 로그 설정 사항을 XML 파일로 저장할 수 있으며, 기존에 저장된 로그 설정 파일이 있을 경우 **로드(L)**를 이용하여 불러올 수 있습니다.

로그 유형

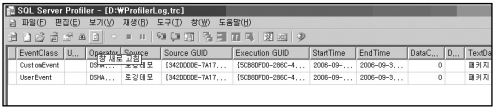
SSIS에서는 5가지의 형태의 로그 공급자를 사용할 수 있습니다. 동시에 여러 형태로 설정하여 로그를 남길 수도 있습니다. 로그 공급자 유형을 선택한 후, 해당 로그를 선택하고 구성을 눌러 로그 공급자가 사용할 로그 유형에 대한 연결을 설정해야 합니다. **텍스트 파일용 로그 공급자**나 **SQL 프로파일러용 SSIS 로그 공급자**, **XML 파일용 로그 공급자**인 경우에는 **파일 연결 관리자**를 이용하여 로그 파일을 지정하며, **SQL Server용 SSIS 로그 공급자**인 경우에는 **OLE DB 연결 관리자**를 이용해서 지정합니다.



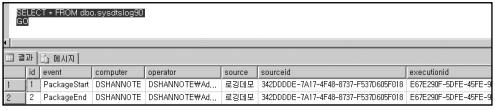
1. **텍스트 파일용 SSIS 로그 공급자** - CSV(쉼표로 구분된 형태) 파일로 저장됩니다. 일반 메모장과 같은 텍스트 편집기에서 확인할 수 있으며, 엑셀에서 해당 파일을 읽어올 수도 있습니다.



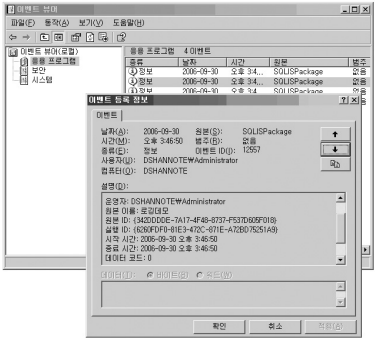
2. **SQL 프로파일러용 SSIS 로그 공급자** - SQL Server 2005 프로파일러에서 읽어올 수 있는 형태의 로그 파일을 생성합니다. 기본 확장자는 .trc 입니다.



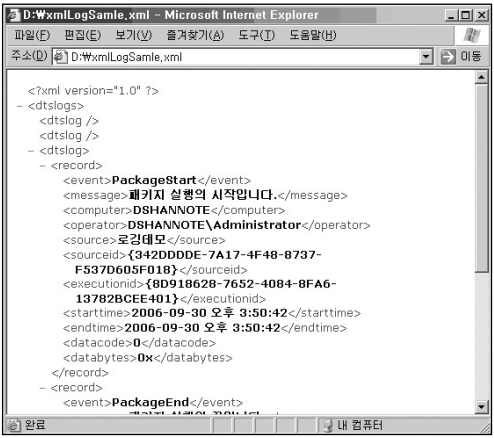
3. SQL Server용 SSIS 로그 공급자 - 로그 구성 화면에서 설정된 OLE DB 연결에 해당하는 데이터베이스에 sysdtslog90이라는 테이블이 생성되며 이 테이블에 로그 정보를 기록합니다. 이미 해당 데이터베이스에 테이블이 있는 경우에는 새로 생성하지는 않습니다. 테이블에 저장된 로그의 경우 SQL 쿼리를 이용해서 쉽게 조회하고 관리할 수 있는 장점이 있습니다.



4. Windows 이벤트 로그용 SSIS 로그 공급자 - Windows의 이벤트 로그 정보에 로그 정보가 기록됩니다. Microsoft Operations Manager나 별도의 로그 파일을 관리하는 프로그램을 사용하는 경우, 다른 어플리케이션의 로그와 같이 관리할 수 있는 장점이 있습니다.



5. **XML 파일용 SSIS 로그 공급자** - XML 형태의 파일로 로그 정보가 기록됩니다. XML 형태로 저장된 파일은 XML Viewer나 기타 관리 프로그램을 이용하여 쉽게 관리할 수 있으며, SQL 쿼리를 사용하여 쉽게 조회할 수도 있습니다.

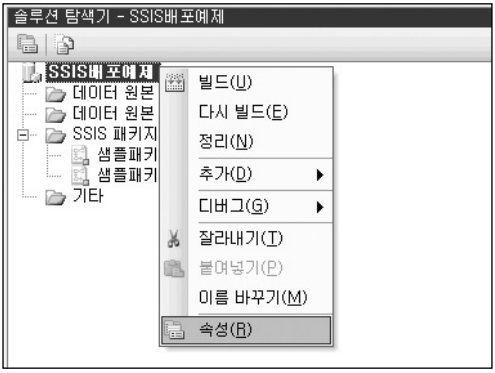


배포 및 배포 마법사

패키지를 개발한 후 서버에 등록하거나 다른 환경에 배포하기 위해서는 단순히 .dtsx 파일을 서버의 특정 위치로 복사하거나 BIDS의 **[파일]** 메뉴 중, **[다른 이름으로 저장]**, **[복사본 저장]** 등을 이용하여 수행할 수 있습니다.

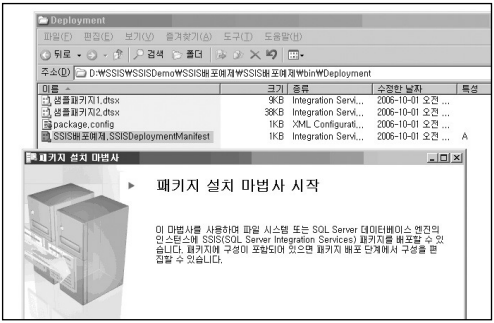
그렇지만, 프로젝트의 규모가 크거나 **구성** 정보를 포함하여 배포해야 할 경우에는 SSIS에서 제공하는 **배포 마법사**를 이용하는 것이 효과적입니다. **배포 마법사**는 어플리케이션의 설치 파일과 같은 형식으로 패키지를 설치할 수 있는 형태로 만들어주는 기능입니다.

배포 마법사는 개별 패키지 수준이 아닌 프로젝트 전체 수준으로 수행됩니다. **솔루션 탐색기**의 프로젝트 명에서 **속성(R)**을 선택한 후 나타나는 속성 창에서 **배포 유틸리티**를 선택하면 됩니다.



- AllowConfigurationChanges - 패키지를 배포할 때 구성 정보를 수정할 수 있도록 허용할지를 설정합니다. 예를 들어, **파일 연결**의 Connection 정보를 관리하는 **구성** 파일을 설정해 두었을 때, AllowConfigurationChanges의 속성을 True로 한 후 배포 파일을 만들면, 배포 과정에서 구성으로 설정된 값에 대해 수정할 수 있는 부분이 나타납니다. 개발 서버와 운영 서버의 환경이 다르거나 연결 정보를 변경해야 할 경우 이 속성을 True로 설정하여 구성에서 관리하는 속성값을 변경할 수 있도록 합니다.

- CreateDeploymentUtility - 패키지를 배포할 수 있는 유틸리티를 포함할지를 설정합니다. 배포할 수 있는 유틸리티 파일은 확장자가 .SSISDeploymentManifest 인 형태입니다. 이 유틸리티를 사용하여 실행하면 배포 위치를 지정하거나 설정 등을 쉽게 지정할 수 있는 패키지 설치 마법사가 실행됩니다.



- DeploymentOutputPath - 배포 파일과 유틸리티가 저장되는 폴더의 위치를 지정합니다. 기본적으로 프로젝트 파일이 있는 위치의 하위 폴더에 생성됩니다.

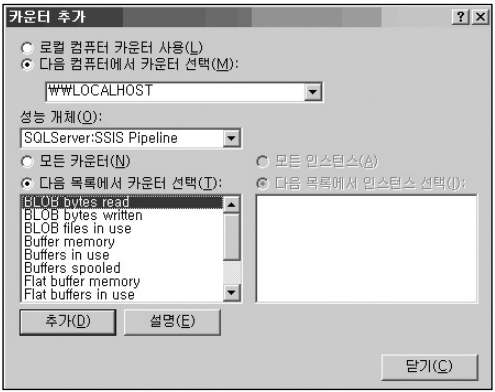
속성 패키지의 **배포 유틸리티** 부분에서 배포와 관련된 사항을 설정하였더라도 패키지를 **빌드**하기 전에는 배포 파일이 생성되지 않습니다. **솔루션 탐색기**에서 프로젝트 이름에서 마우스 오른쪽 버튼을 클릭한 후, **빌드(U)**를 선택하거나 BIDS의 상단 메뉴 중 **빌드(B)** 메뉴를 클릭하여 패키지를 빌드하면 배포 파일이 생성됩니다.

패키지 설치 마법사를 이용하여 개발된 패키지 파일을 손쉽게 **파일 시스템** 또는 SQL Server에 배포할 수 있으며, 구성 정보에 대한 세부 항목의 설정을 변경할 수도 있습니다.

SSIS 기타 사항

성능 카운터

SQL Server 2005에서는 SSIS와 관련된 성능 카운터 항목들을 제공합니다.



[제어판] → [관리도구] → [성능]을 선택하여 성능 모니터를 실행합니다. 성능 개체 중 SQLServer:SSIS Pipeline과 SQLServer:SSIS Service가 SSIS의 성능을 모니터링할 수 있는 카운터 입니다.

SQLServer:SSIS Service 개체에는 SSIS Package Instances라는 카운터만 존재하며 현재 수행되고 있는 SSIS 패키지의 개체 수를 나타냅니다.

SQLServer:SSIS Pipeline 개체의 카운터는 다음 표를 참고하기 바랍니다. 이 개체는 데이터 흐름 작업에서의 성능에 대한 정보를 제공합니다. 현재 패키지가 사용하고 있는 메모리 버퍼의 크기라든지, 원본 또는 대상에서의 행 수, BLOB(Binary Large Object) 데이터의 통계 정보 등을 제공합니다.

성능 카운터	설명
BLOB bytes read	데이터 흐름 엔진이 모든 원본에서 읽어 온 BLOB (Binary Large Object) 데이터의 바이트 수입니다.
BLOB bytes written	데이터 흐름 엔진이 모든 대상에 기록한 전체 BLOB 데이터의 바이트 수입니다.
BLOB files in use	데이터 흐름 엔진이 스풀링을 위해 사용하는 BLOB 파일 수입니다.
Buffer memory	사용 중인 모든 유형의 메모리 버퍼 양입니다. 이 값이 물리적인 메모리의 양보다 크면 Buffers Spooled는 늘어나며 이는 메모리의 스와핑이 증가함을 나타냅니다. 메모리 스와핑이 증가하면 데이터 흐름 엔진의 성능이 떨어집니다.
Buffers in use	데이터 흐름 엔진이 현재 사용 중인 모든 유형의 버퍼 개체 수입니다.
Buffers spooled	디스크에 쓰여진 버퍼 수입니다. 데이터 흐름 엔진에 물리적 메모리가 부족하면 현재 사용되지 않은 버퍼는 디스크에 쓰여지고 필요에 따라 다시 로드됩니다.
Flat buffer memory	모든 플랫 버퍼가 사용하는 전체 메모리(바이트)입니다. 플랫 버퍼는 구성 요소가 데이터 저장에 사용하는 메모리 블록입니다. 플랫 버퍼는 바이트의 큰 블록이며 바이트 단위로 액세스됩니다.
Flat buffers in use	데이터 흐름 엔진이 사용하는 플랫 버퍼 수입니다. 모든 플랫 버퍼는 전용 버퍼입니다.

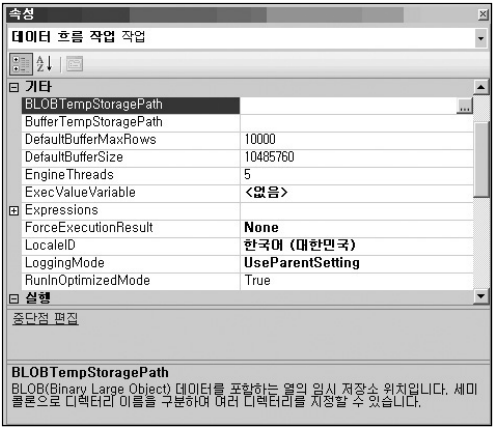
성능 카운터	설명
Private buffer memory	모든 전용 버퍼가 사용하는 전체 메모리 양입니다. 데이터 흐름 엔진이 데이터 흐름을 지원하기 위해 만드는 버퍼는 전용 버퍼가 아닙니다. 전용 버퍼는 변환 작업에서 임시 작업용으로만 사용하는 버퍼입니다. 예를 들어 집계 변환은 전용 버퍼를 사용하여 내부 계산을 수행합니다.
Private buffers in use	변환 작업에서 사용하는 버퍼 수입니다.
Rows read	원본에서 생성하는 행 수입니다. 조회 변환이 참조 테이블에서 읽은 행은 포함되지 않습니다.
Rows written	대상에 제공된 행 수입니다. 대상 데이터 저장소에 쓰여진 행은 반영되지 않습니다.

성능 카운터에서의 Private buffer와 Flat buffer는 다음과 같습니다.

- Private buffer - 정렬 변환이나 집계 변환 등과 같이 변환 작업 개체가 결과를 처리하기 위해 사용하는 임시 메모리 공간입니다.

- Flat buffer - 데이터 흐름 작업에서 데이터를 저장하는데 이용되는 메모리 공간입니다. 예를 들어, **조회** 변환에서 조회 테이블의 데이터를 임시로 저장하여 사용할 때 Flat buffer를 이용합니다.

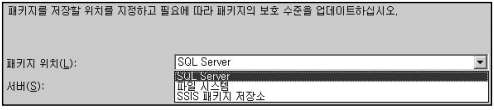
성능 카운터에서 제공하는 메모리와 관련된 정보를 이용하여 시스템 환경에 맞는 값을 설정하여 패키지의 처리 성능을 향상시킬 수 있습니다.



현재 작업량에 비해 사용 가능한 메모리가 클 경우 DefaultBufferSize의 크기를 늘릴 수도 있으며, Text형이나 Image 형과 같은 BLOB(Binary Large bject) 데이터의 처리가 많은 경우 BLOBTempStoragePath를 별도로 지정하여 처리 성능을 개선할 수도 있습니다.

패키지 저장 방식에 따른 장점

SSIS에서는 기본적으로 세 가지 방식으로 패키지를 저장할 수 있습니다.



1. SQL Server로 지정하면 SQL Server의 msdb에 저장됩니다. 해당 패키지의 정보는 **msdb.dbo.sysdtspackages90** 테이블에 저장됩니다.
2. 파일 시스템으로 저장하면 사용자가 지정한 위치에 저장합니다.
3. SSIS 패키지 저장소 역시 파일 형식으로 패키지를 관리하지만, 저장 위치는 서버의 특정 위치에 저장됩니다.
   기본적으로 %ProgramFiles%₩Microsoft SQL Server₩90₩DTS₩Packages 에 저장됩니다.

즉, 패키지는 SQL Server에 저장되거나 파일로 저장되어 관리됩니다. 각각의 방식에 따른 장점은 다음과 같습니다.

- **파일 형태로 저장하여 관리하는 방식의 장점**
  ① 연결 정보를 쉽게 공유하여 사용할 수 있습니다.
  ② 보안 설정에서 사용자 키 기반의 암호화(EncryptSensitiveWithUserKey 또는 EncryptAllWithUserKey)로 설정한 경우, 매우 강력한 패키지 암호화를 구현할 수 있습니다. 이 경우, 해당 사용자의 프로필 환경 외에서는 패키지 수정이나 조회가 어렵습니다.
  ③ 네트워크 장애 또는 DB 장애 발생시에도 안전합니다.
  ④ 구성 파일이나 기타 파일들을 함께 관리할 수 있습니다.
  ⑤ 패키지를 수정해야 할 때 개발 환경으로 로드하는 작업이 간단합니다.
  ⑥ 파일로 관리되기 때문에 파일 시스템의 계층 구조를 이용하여 관리할 수 있습니다.
  ➔ 자주 수정되거나 아직 최종적으로 개발 완료되지 않은 패키지 형태인 경우 파일 형태의 관리가 용이합니다.

- **SQL Server에 저장하여 관리하는 방식의 장점**

  ① 여러 사용자들이 공유하기가 용이합니다.

  ② 데이터베이스의 보안 기능을 사용하여 관리할 수 있습니다. SSIS 패키지의 운영이나 수정과 관련된 역할(Role) 등을 적용할 수 있습니다.

  ③ msdb DB에 테이블 형태로 저장되기 때문에 일반 DB 백업과 같은 방식으로 패키지를 백업하여 관리할 수 있습니다.

  ④ SQL 쿼리를 이용하여 패키지에 대한 정보를 조회할 수 있습니다.

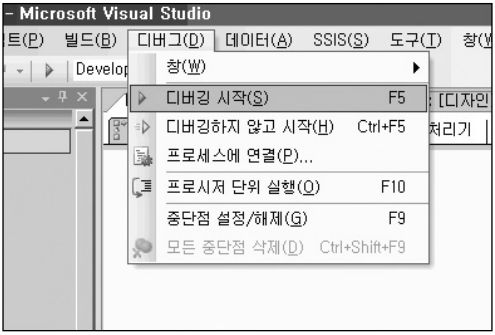
  ➔ 자주 수정이 일어나지 않는 형태인 운영 환경에서는 SQL Server 형태의 관리가 용이합니다.

패키지 실행 및 마이그레이션

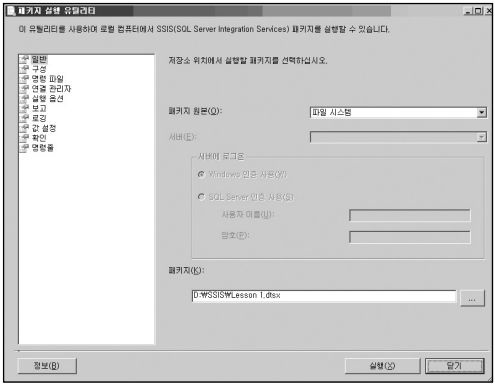
SSIS 패키지 실행하기

패키지를 수동으로 실행하는 방법

패키지를 수동으로 실행하는 방법에는 세 가지가 있습니다. 첫 번째는 BIDS에서 직접 실행하는 방법입니다.

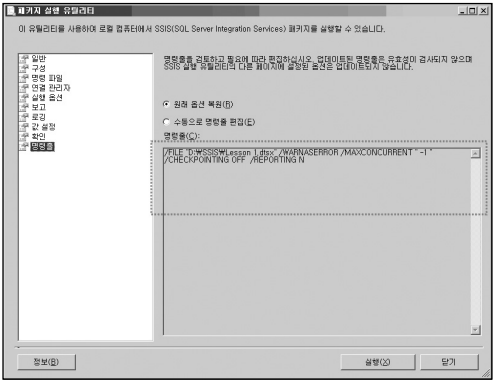


두 번째는 패키지 실행 유틸리티인 DTEXECUI를 이용하는 방법입니다. [시작] → [실행] 에서 dtexecui.exe를 입력하여 이를 실행한 후 수행하고자 하는 패키지를 지정하여 수행할 수 있으며, 간단히 수행할 패키지 파일(.dtsx)을 더블 클릭하여 DTEXECUI를 실행할 수도 있습니다.



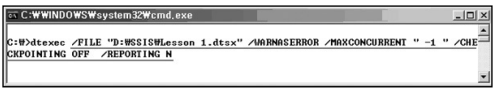
위의 그림과 같이 **구성**, **명령 파일**, **연결 관리자**, **실행 옵션**, **보고**, **로깅**, **값 설정**, **확인**, **명령줄** 등 여러 가지의 옵션들을 설정할 수 있는 탭으로 구성되어 있습니다.

연결 관리자 탭에서 이미 지정되어 있는 연결과는 다른 연결을 설정할 수 있으며, **보고** 탭에서 수행될 때 출력할 상태 정보의 수준을 설정할 수도 있습니다. 실행에 필요한 설정을 마친 후, 실행 버튼을 누르면 해당 패키지가 실행됩니다.



명령줄 탭을 클릭하면 위의 그림과 같이 설정한 사항들이 매개변수 형식으로 표시됩니다. 이 명령줄은 패키지를 수행하는 세 번째 방법인 dtexec.exe 에서 사용할 수 있는 정보입니다.

패키지를 실행하는 세 번째 방법은 DTEXEC 명령을 이용하는 것입니다. 이는 DTEXECUI 와는 달리 콘솔 모드에서 수행됩니다.
필요한 여러 옵션들을 직접 지정하여 패키지를 수행할 수도 있지만, 간단히 두 번째 방법의 마지막 그림에서와 같이 **명령줄** 옵션에서 표시된 정보를 복사한 후 명령 창에 붙여 넣는 방식으로 수행하면 됩니다.



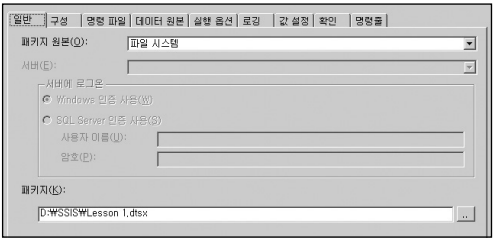
성능 상으로는 콘솔 모드에서 실행되는 DTEXEC를 이용하여 여러 정보들을 나타내지 않는 옵션으로 수행하는 것이 가장 우수합니다. DTEXECUI를 이용하여 패키지 실행에 필요한 여러 설정들을 지정한 후, **명령줄** 탭에서 자동으로 생성되는 매개변수 정보를 복사하여 DTEXEC에 이용하는 방식으로 수행하는 것이 효율적입니다.

패키지를 SQL Server 에이전트에 등록하는 방법

Management Studio 에서 [SQL Server 에이전트] → [작업]에서 **새 작업(N)**을 클릭합니다.

작업 속성에서 적절한 이름과 소유자를 지정합니다.
단계 탭에서 **새로 만들기**를 클릭한 후, 유형에서 SQL Server Integration Services 패키지를 선택합니다.

패키지 원본에서 **파일 시스템**을 선택하고 패키지를 지정합니다.



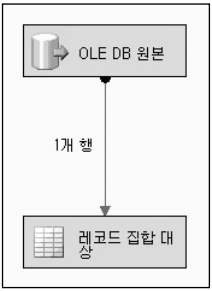
구성, 명령 파일 등의 탭은 이전 부분에서 설명한 패키지 실행 유틸리티(DTEXECUI) 와 동일합니다.

패키지 실행에 필요한 여러 옵션들을 설정한 후, **작업 속성** 탭에서 적절한 일정을 등록하고 확인을 누르면 SQL Server 에이전트 작업으로 등록됩니다.

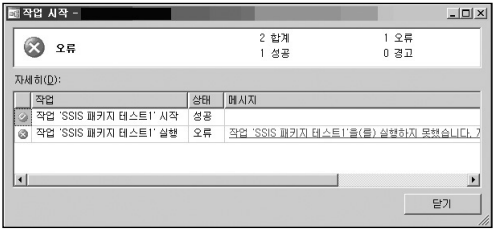
SQL Server 에이전트에서 SSIS 패키지가 실행되지 않는 경우

로컬 PC 또는 서버에서는 패키지를 실행하면 제대로 수행되지만, 패키지 파일(*.dtsx)을 서버에 저장한 후 SQL Server 에이전트에 등록하여 실행하면 오류가 발생하는 경우가 있습니다.

다음과 같이 BIDS에서 직접 패키지를 실행하면 정상적으로 수행됩니다.



이 패키지를 SQL Server 에이전트에 등록한 후 실행하면 다음과 같은 오류가 발생합니다.



이러한 오류가 발생하는 원인은 다음과 같습니다.

SSIS 패키지의 보안 수준 속성인 ProtectionLevel은 기본값으로 EncryptSensitiveWithUserKey로 설정되어 있습니다.

EncryptSensitiveWithUserKey는 중요한 데이터(Sensitive Data)를 현재 패키지가 개발되는 사용자 프로필 기반의 키를 이용하여 암호화한다는 의미입니다. **중요한 데이터**란 DB 연결 정보 또는 암호, XML 노드 정보 등과 같은 연결 정보입니다. 따라서 윈도우 인증이 아닌 암호를 이용한 **OLE DB 연결**이나 **XML 작업** 등이 포함된 경우 중요한 데이터가 패키지에 포함된 상태이기 때문에 패키지가 암호화됩니다.

수동으로 실행할 때에는 패키지가 암호화된 개인키와 실행 환경의 개인키가 동일하기 때문에 정상적으로 실행됩니다. 하지만, 서버로 파일을 복사한 후 SQL Server 에이전트에서 실행하거나, 현재의 개발 PC가 아닌 다른 곳에 패키지를 복사해서 수행하면 패키지가 암호화된 개인키와 실행 환경의 개인키가 다르기 때문에 오류가 발생합니다.

따라서 개발 환경과는 다른 곳에서 패키지를 실행하거나 SQL Server 에이전트를 이용하여 수행해야 할 경우에는 다음 두 가지 방법을 참고하여 패키지의 보안 수준을 조절해야 합니다.

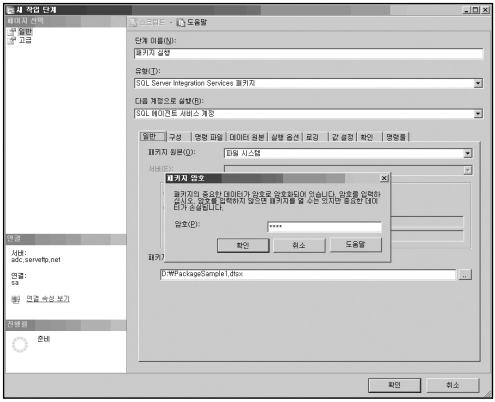
[방법 1]

패키지의 ProtectionLevel을 EncryptSensitiveWithUserKey 대신, EncryptSensitiveWithPassword 또는 EncryptAllWithPassword로 설정한 후, PackagePassword 항목에 암호를 설정합니다.

이런 방식으로 저장하면 패키지의 중요한 정보 또는 패키지 정보 전체가 개인키 대신 암호를 기반으로 암호화됩니다.

이후, SQL Server **에이전트**에서 이 패키지를 실행하기 위해 설정하는 단계에서는 다음과 같은 암호 입력 창이 나타납니다.

(일반 탭에서 실행할 패키지를 지정한 후, 다른 탭을 클릭할 경우 나타납니다.)



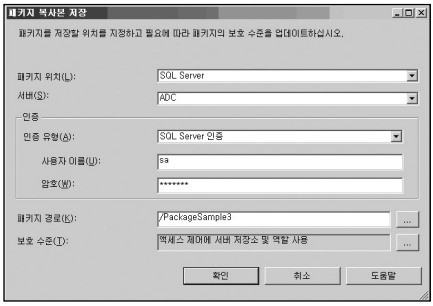
위와 같은 패키지 암호 설정 창에서 지정한 암호를 입력한 후 패키지를 실행하면 정상적으로 수행됩니다.

[방법 2]

패키지의 ProtectionLevel을 ServerStorage로 임시 변경한 후, 패키지를 실행할 SQL Server에 저장합니다.

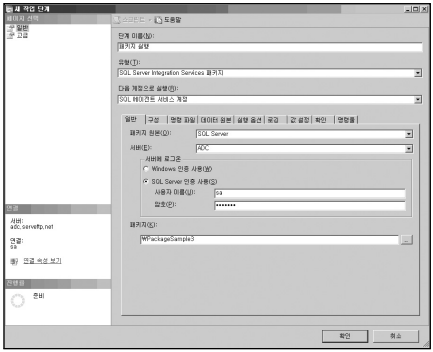
ProtectionLevel을 ServerStorage로 설정하는 것은 패키지의 보안과 관련된 사항을 SQL Server에게 관리하도록 넘기는 것입니다. 이 때, 주의할 점은 이 설정으로 변경한 경우에는 패키지는 파일 형태로 저장할 수 없고 SQL Server로만 저장할 수 있습니다.
개발 화면의 상단 **파일(F)** 메뉴 중, 다른 이름으로 … .dtsx의 복사본 **저장(C)**을 클릭합니다.

패키지 위치를 **SQL Server**로 지정한 후, 인증 유형을 설정합니다. Windows 인증이나 SQL Server 인증이나 상관없이 연결할 수 있는 적절한 권한(msdb의 db_dtsadmin 역할 또는 db_dtsltduser 역할)이 있는 계정으로 지정하면 됩니다. **패키지 경로(K)** 부분에서는 저장할 패키지 이름을 지정합니다.



단, 이 때 SQL Server에 저장한 후, 다시 로컬 파일로 저장할 경우 저장이 되지 않습니다. 이는 앞서 언급한 바와 같이 ProtectionLevel을 ServerStorage로 설정하였기 때문입니다. 따라서 패키지를 별도의 파일로 저장하기 위해서는 ProtectionLevel의 속성값을 Server Storage 이외의 값으로 변경해야 합니다.

이후, SQL Server **에이전트**에서 패키지를 실행하는 작업을 등록할 경우, 다음 그림과 같이 **패키지 원본**을 SQL Server로 지정한 후 저장한 패키지를 선택하면 됩니다.



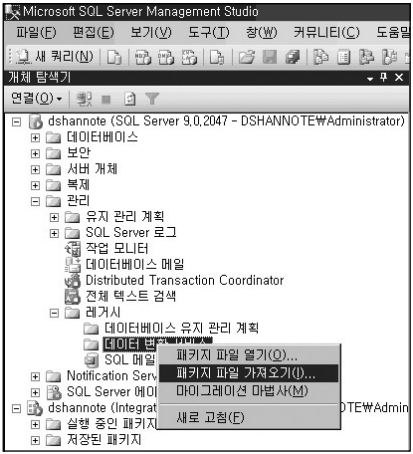
DTS 패키지를 SSIS로 업그레이드 및 마이그레이션하기

SQL 2000 DTS 패키지를 SQL Server 2005에서 사용하기 위한 방법으로는 **업그레이드** 또는 **마이그레이션**을 수행하는 방법이 있습니다. 업그레이드는 SQL 2000 DTS 패키지를 아무런 변경 없이 SQL 2005 환경에서 사용하는 것이며, 마이그레이션은 **패키지 마이그레이션 마법사**를 이용하여 SQL 2000 DTS 패키지를 2005 SSIS 패키지로 변환한 후 사용하는 것입니다. **패키지 마이그레이션 마법사**는 SQL Server 2005 설치 시 Integration Services를 설치하도록 선택하면 자동으로 설치됩니다.

2005에서 기존 DTS 패키지 그대로 이용할 수 있도록 업그레이드하기

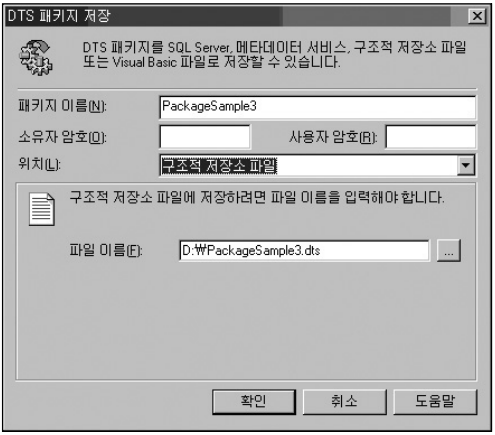
SQL Server 2005에서는 SQL 2000 DTS 패키지를 그대로 사용할 수도 있습니다.

[SQL Server Management Studio에서 사용하기]

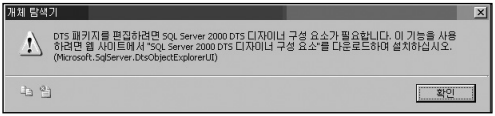


SQL Server Management Studio의 개체 탐색기 중 [관리] → [레거시] → [데이터 변환 서비스]에서 DTS 패키지를 가져올 수 있습니다. DTS 패키지가 .dts 형태의 패키지 파일로 저장되어 있는 경우 **패키지 파일 열기(O)**를 이용하여 열 수 있으며, 파일 형태의 DTS 패키지를 SSIS로 가져오기 위해서는 **패키지 파일 가져오기(I)** 또는 **마이그레이션 마법사(M)**를 사용합니다.

SQL Server 2000의 엔터프라이즈 관리자의 [데이터 변환 서비스] → [로컬 패키지]에 저장되어 있는 DTS 패키지를 확장자가 .dts인 파일로 저장한 후, 이를 SQL Server 2005에서 사용할 수 있습니다. DTS 패키지를 파일 형태로 저장하기 위해서는 DTS 패키지의 상단 메뉴 중 [다른 이름으로 저장]을 선택한 후, **위치(L)**를 **구조적 저장소**로 선택하면 됩니다.



단, SQL Server Management Studio에서 **패키지 파일 열기(O)**를 이용하여 DTS 패키지 파일을 열 때에는 **SQL Server 2000 DTS 디자이너 구성요소**가 설치되어 있어야 하며, 이는 SQL Server 2005와는 별도로 설치해야 합니다. 이 구성요소를 설치하지 않은 경우에는 다음과 같은 경고 메시지가 나타납니다.

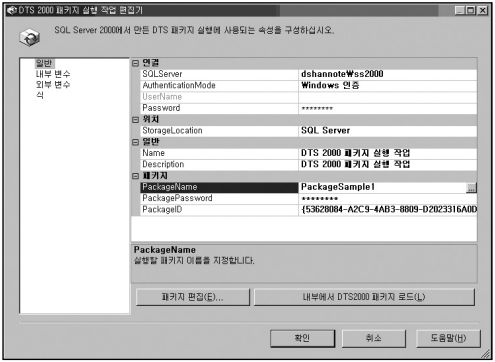


[참고]
SQL Server 2000 DTS 디자이너는 다음 링크에서 받을 수 있습니다.
http://download.microsoft.com/download/4/4/D/44DBDE61-B385-4FC2-A67D-48053B8F9FAD/SQLServer2005_DTS.msi

SSIS 패키지 내에서 기존 DTS 패키지 이용하기

SQL 2005 SSIS에는 SQL 2000 DTS 패키지를 직접 수행할 수 있는 **DTS 2000 패키지 작업**이 별도의 작업 개체로 포함되어 있습니다.

제어 흐름 항목 중 **DTS 2000 패키지 실행 작업**을 선택하여 **제어 흐름** 작업 영역에 추가한 후, 파일 형태 혹은 SQL Server 2000에 저장되어 있는 DTS 패키지를 호출하여 실행할 수 있습니다.



패키지 마이그레이션 하기

SQL Server 2005에서는 SQL 2000 DTS 패키지를 마이그레이션하기 위한 **패키지 마이그레이션 마법사**를 제공합니다. 패키지 마이그레이션 마법사는 SQL Server 2005 Standard Edition 및 Enterprise Edition, Developer Edition에서 사용할 수 있습니다.

패키지 마이그레이션 마법사를 이용하여 SQL 2000 DTS 패키지에 포함된 대부분의 작업들을 마이그레이션 할 수 있지만, **ActiveX 스크립트 작업**이나 **동적 속성 작업**과 같은 일부 작업 개체에 대해서는 마이그레이션이 완전하게 수행되지 않을 수 있습니다.

[작업 매핑표]

SQL 2000 DTS의 작업 개체를 SQL 2005 SSIS로 마이그레이션 시 다음 표와 같이 변경됩니다.

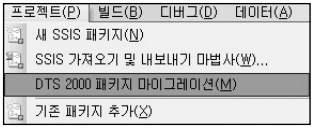
SQL Server 2000 DTS 작업	SQL Server 2005 SSIS 작업
ActiveX 스크립트 작업	ActiveX 스크립트 작업
대량 삽입 작업	대량 삽입 작업
SQL Server 개체 복사 작업	SQL Server 개체 전송 작업
데이터 마이닝 예측 작업	데이터 마이닝 쿼리 작업
패키지 실행 작업	DTS 2000 패키지 실행 작업
프로세스 실행 작업	프로세스 실행 작업
SQL 실행 작업	SQL 실행 작업
파일 전송 프로토콜 작업	FTP 작업
메시지 큐 작업	메시지 큐 작업
메일 보내기 작업	메일 보내기 작업

SQL Server 2000 DTS 작업	SQL Server 2005 SSIS 작업
데이터베이스 전송, 오류 메시지 전송, 작업 전송, 로그인 전송 및 master 저장 프로시저 전송 작업	SQL Server 개체 전송 작업, 데이터베이스 , 전송 작업 오류 메시지 전송 작업, 작업 전송 작업, 로그인 전송작업 및 마스터 저장 프로시저 전송 작업

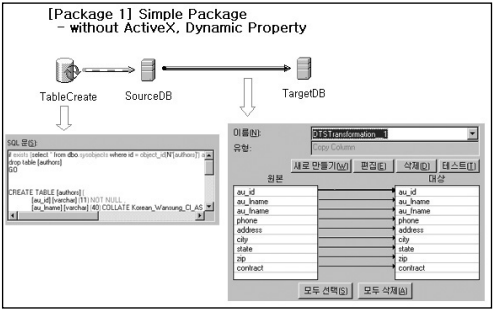
Analysis Service와 관련된 작업, 데이터 기반 쿼리 작업, 동적 속성 작업, 데이터 변환 작업 등은 SSIS 작업으로 변환되지 않습니다.

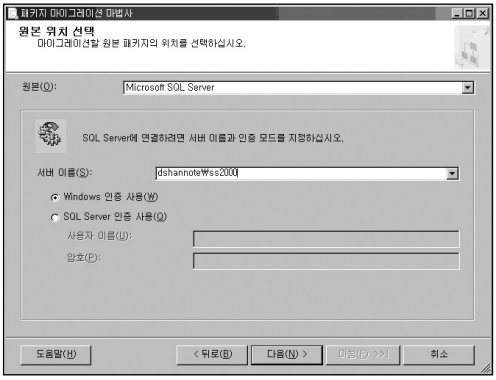
패키지 변환 마법사 실행하기

패키지 변환 마법사는 BIDS의 [프로젝트(P)] → [DTS 패키지 마이그레이션 마법사(M)] 메뉴 혹은 솔루션 탐색기의 SSIS 패키지에서 실행할 수 있으며, SQL Server Management Studio의 [관리] → [레거시] → [데이터 변환 서비스] 에서 **마이그레이션 마법사(M)**를 선택하여 실행할 수 있습니다.

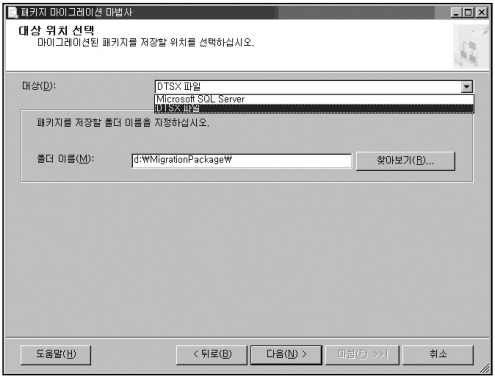


다음과 같이 **SQL 실행 작업**이나 **데이터 펌프 작업** 등으로 이루어진 단순한 DTS 패키지를 **패키지 변환 마법사**를 이용하여 변환하는 방법을 설명합니다.



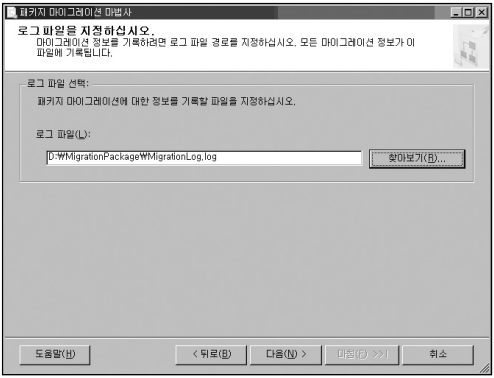


패키지 변환 마법사에서 DTS 패키지 원본 지정

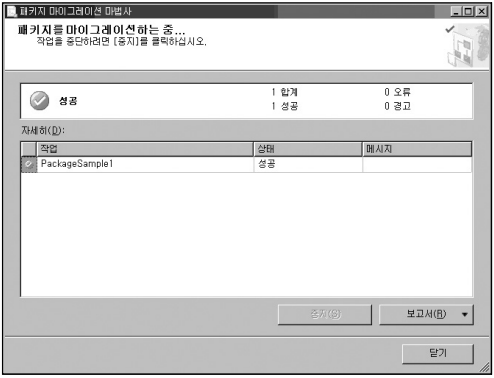


SSIS 패키지로 변경 후 저장할 위치 선택

마이그레이션할 패키지 선택



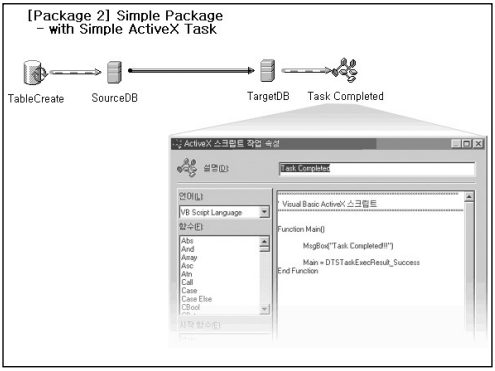
마이그레이션 로그 파일 지정



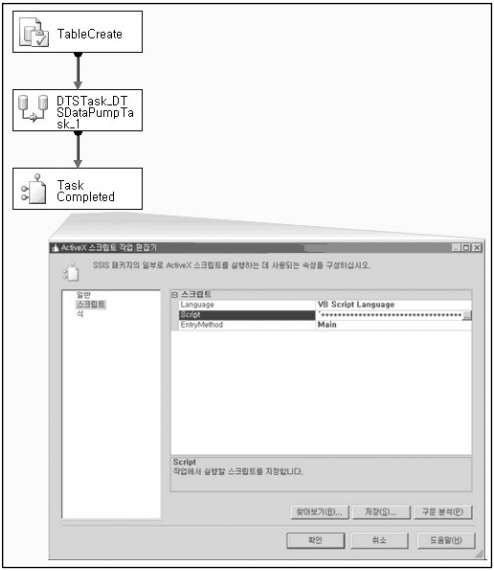
마이그레이션 수행

패키지 마이그레이션 마법사를 사용하여 DTS 패키지 마이그레이션하기(2)

ActiveX 스크립트 작업 중 단순한 형태의 스크립트는 패키지 마이그레이션 마법사를 사용하여 변환한 후 그대로 사용할 수 있습니다.



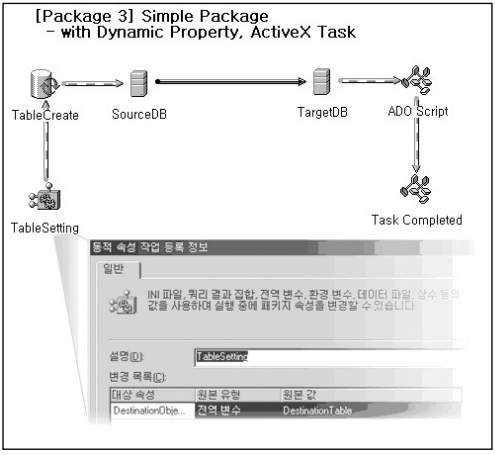
ActiveX 스크립트 작업이 포함된 패키지


TableCreate
DTSTask_DTSDataPumpTask_1
Task Completed
ActiveX 스크립트 작업 편집기
SSIS 패키지의 일부로 ActiveX 스크립트를 실행하는 데 사용되는 속성을 구성하십시오.
일반
스크립트
작
스크립트
Language
VB Script Language
Script
EntryMethod
Main
Script
작업에서 실행할 스크립트를 지정합니다.
찾아보기(B)...
저장(S)...
구문 분석(P)
확인
취소
도움말(H)

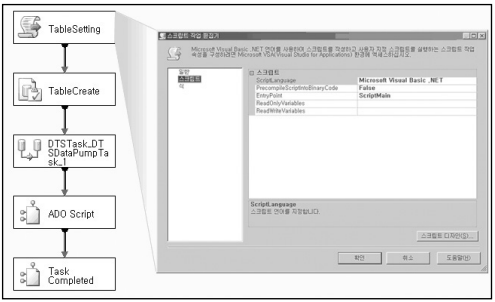
마이그레이션 수행 후

패키지 마이그레이션 마법사를 사용하여 DTS 패키지 마이그레이션 하기(3)

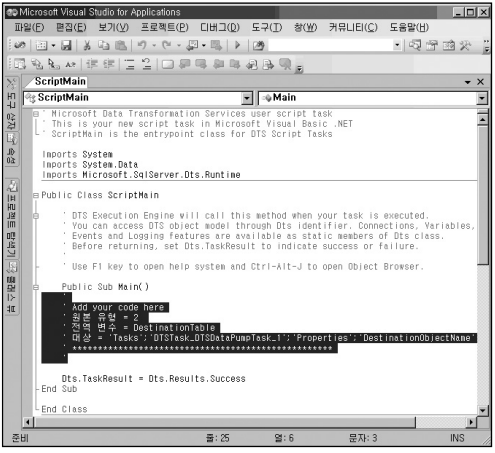
동적 속성 작업이 포함된 패키지의 경우, 주석 처리가 된 스크립트 형태의 작업으로 변환됩니다. 이러한 변환 결과를 참고하여 SSIS의 **구성** 및 **속성 식** 등을 이용하여 별도로 수정해야 합니다.



동적 속성 작업이 포함된 패키지



마이그레이션 수행 후



동적 속성 작업이 변환된 스크립트 작업

패키지 마이그레이션 시 참고 사항

패키지 마이그레이션 마법사는 원본 DTS 패키지를 변경하지 않고 그대로 유지합니다.

SSIS는 DTS에서와는 달리 패키지 이름에 잘못된 문자(/ \ : [ ] . =)가 포함되어 있거나 등록되어 있지 않은 개체가 포함되어 있는 경우, 마이그레이션이 수행되지 않습니다.

패키지를 마이그레이션해도 패키지의 암호는 함께 마이그레이션되지 않습니다.

비매품

SQL Server 2005 Integration Services 포켓북

- 저 자: 한 대 성
- 감 수: 하 성 희
- Contents 관련문의: dshan@adconsulting.co.kr

- 발 행: 한국마이크로소프트(유)
- 제 작: 에이디컨설팅
- 초판 발행일: 2007년 1월

Microsoft
SQL Server 2005

SQL Server 2005 Integration Services 포켓북

Microsoft
한국마이크로소프트(유)

서울특별시 강남구 대치동 892번지 포스코센터 서관 5층
고객지원센터 : 1577-9700
인터넷 : http://www.microsoft.com/korea/sql

SQL-200701-01