Міністерство освіти і науки України
Сумський державний університет
Факультет електроніки
та інформаційних технологій

5889 Методичні вказівки

до самостійної роботи
з дисципліни «**Сховища даних**»
для здобувачів спеціальності *122 «Комп'ютерні науки»*
освітнього ступеня ***«магістр»***
усіх форм здобуття вищої освіти

Суми
Сумський державний університет
2024

Методичні вказівки до самостійної роботи з дисципліни «Сховища даних» / укладачі: А. В. Неня, В. В. Курінна. – Суми : Сумський державний університет, 2024. – 173 с.

Кафедра інформаційних технологій факультету ЕлІТ

ЗМІСТ

ВСТУП

Методичні вказівки призначені для самостійної роботи в процесі підготовки до виконання лабораторних робіт із підтримки життєвого циклу сховищ даних засобами SQL Server Management Tools у рамках курсу дисципліни «Сховища даних».

У методичних вказівках наведений теоретичний огляд питань реалізації моделей сховищ даних, налаштування їх продуктивної роботи, завантаження та витягання даних тощо. Рекомендованим є ознайомлення з теоретичним матеріалом до початку виконання завдань лабораторних робіт.

Наприкінці огляду теоретичних матеріалів наведений перелік тестових питань із варіантами відповідей для самоперевірки до початку виконання завдань лабораторної роботи. Відповіді на наведені тестові питання не перевіряє викладач.

Аналітичні питання, наведені в цих методичних вказівках, призначені для поглибленого опрацювання практичних навичок підтримки життєвого циклу сховищ даних. Відповіді на аналітичні питання можуть бути надані на перевірку викладачеві за бажанням здобувача. Оцінювання розгорнутих відповідей на аналітичні питання відбувається згідно з регламентом оцінювання завдань курсу дисципліни «Сховища даних».

МАТЕРІАЛ ДЛЯ ПІДГОТОВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ 2

Теоретичний матеріал

Огляд схем моделі сховища даних

Існує дві схеми моделей сховищ даних: «зірка» та «сніжинка». Ці схеми були введені в 1980 р. Схему «зірка» настільки широко застосовують у наші дні, що стала своєрідним неформальним стандартом для всіх типів додатків бізнес-аналітики (*business intelligence*).

Схема «зірка» проілюстрована на рисунку 2.1 ліворуч.

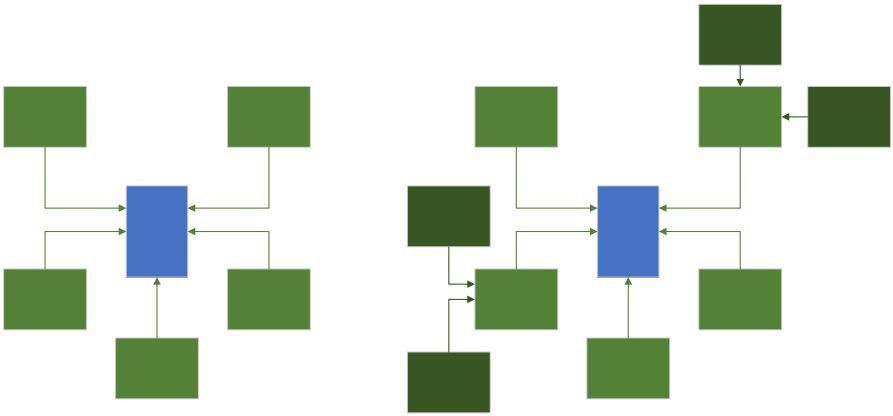

Рисунок 2.1 – Ілюстрація схем даних «зірка» та «сніжинка»

У ній є одна центральна таблиця, яку називають *таблицею фактів (fact table)*, оточену декількома таблицями, яких називають *вимірами (dimensions)*. Одна схема «зірка» охоплює *окрему сферу бізнесу*. Таблиця фактів з'єднана з усіма таблицями вимірів зовнішніми ключами. Зазвичай зовнішні ключі, взяті всі разом, однозначно визначають кожен рядок у таблиці фактів і таким чином формують унікальний ключ, тому всі зовнішні ключі можна використовувати як складений первинний ключ таблиці фактів. За потреби може бути доданий і простіший ключ. У зв'язках із таблицями вимірів таблиця фактів знаходиться на стороні «багатьом». Опис рядка з таблиці фактів

може звучати так: «Замовник A купив товар B у день C кількістю D на суму E».

Схема «зірка» прийшла із концептуальної моделі куба. Можна уявити всі продажі як великий куб. Під час пошуку проблеми в даних про продажі застосовують метод зрізів (або перерізів) куба за різними категоріями замовників, товарів або часових періодів. Іншими словами, куб «нарізається» за його вимірами. Отже, замовники, товари та часові періоди представляють у концептуальній моделі куба продажів три виміри. З подібної концептуальної моделі таблиці вимірів отримали свою назву (виміри). У логічній моделі зі схемою «зірка» можна подати більше ніж три виміри. Таким чином, схема «зірка» представляє *багатовимірний гіперкуб* (*multidimensional hypercube*).

У таблицях вимірів є зовнішні ключі, які пов'язують їх із кількома таблицями фактів. Виміри, пов'язані з кількома таблицями фактів, називають *загальними* (*shared*) чи *узгодженими* (*conformed*) вимірами.

Приватні виміри – це виміри, що стосуються єдиної схеми «зірка». Водночас використовуючи приватні виміри, можуть бути втраченими зв'язки з різними таблицями фактів, тобто не буде можливості порівняти дані в різних таблицях фактів за одними і тими самими вимірами. Наприклад, ви не зможете порівняти продажі та бухгалтерські дані для одного і того самого замовника, якщо таблиці фактів про продаж та облікові дані не користуються одним і тим самим виміром.

Схема «сніжинка» проілюстрована на рисунку 2.1 праворуч.

Ще один приклад сховища даних за схемою «сніжинка» проілюстрований на рисунку 2.2. Таблиця DimProduct містить не назву підкатегорії, а лише значення зовнішнього ключа ProductSubcategoryKey для *таблиці підстановки* (*lookup table*) DimProductSubcategory. Таблиця DimProductSubcategory не містить назви категорії; у ній є лише зовнішній ключ ProductCategoryKey із таблиці DimProductCategory (рис. 2.2).

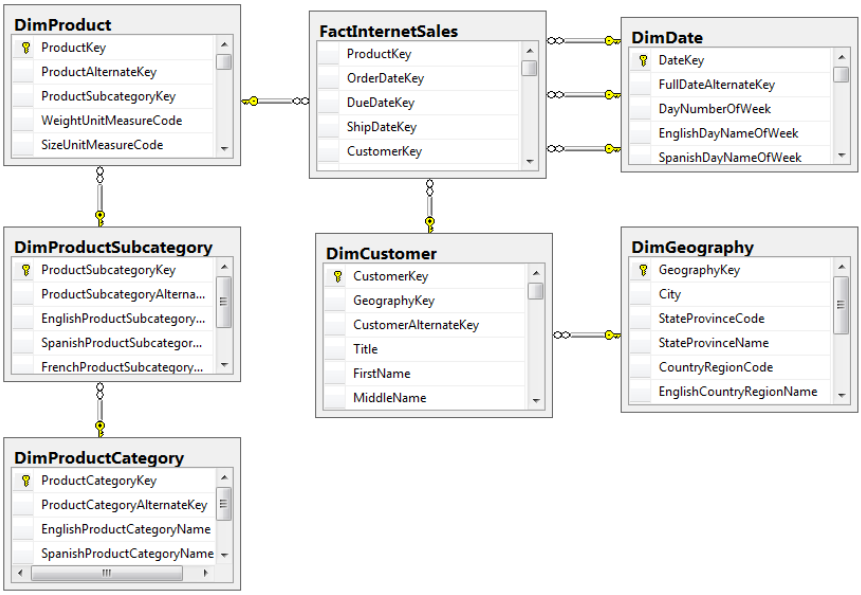

Рисунок 2.2 – Ілюстрація додаткових таблиць

У такій конфігурації «зірка» починає перетворюватися на «сніжинку». Отже, схему «зірка» з нормалізованими вимірами називають схемою «сніжинка».

У більшості довгострокових проєктів необхідно проєктувати схеми «зірка». Оскільки схема «зірка» простіша за схему «сніжинка», її легше супроводжувати. Запити у схемі «зірка» простіші і виконуються швидше за запити у схемі «сніжинка», оскільки включає менше з'єднань. Схема «сніжинка» більше підходить для короткострокових пробних проєктів, тому що вона ближча до нормалізованої схеми бази даних і тому вимагає менше часу на розробку.

Аудит, походження та перетворення даних

Крім таблиць для звітів, сховище даних може містити таблиці аудиту. За кожного оновлення сховища даних (далі – СД) необхідно контролювати, хто виконав оновлення, коли воно було зроблено і скільки рядків було передано в кожний вимір та кожну таблицю фактів у СД. Перевірка витраченого часу для кожного завантаження дозволяє обчислити продуктивність і

вжити заходів для підвищення продуктивності. Ця інформація зберігається в таблиці або таблицях аудиту. Але ви повинні розуміти, що аудит не допоможе, якщо не аналізувати інформацію регулярно.

Таблиці результатів аудиту містять інформацію пакетного рівня про регулярні завантаження в СД, але у вас може виникнути бажання або необхідність отримати більш детальну інформацію. Наприклад, ви захочете знати, звідки у вимір та/або таблицю фактів надійшов кожен рядок і коли вона була додана.

У таких випадках можна вставити відповідні стовпці таблиці вимірювань і фактів. У термінології СД таку інформацію називають життєвим шляхом даних (lineage). Для збору як даних аудиту, так і інформації про походження та перетворення або життєвий шлях даних потрібно відповідним чином змінити процес вилучення-перетворення-завантаження (extract-transform-load, ETL).

Типи стовпців у вимірах

Виміри надають контекстну інформацію для звітів. Типовий аналіз включає зведені таблиці і зведені графіки. Вони представлені зведенням за стовпцями одного чи кількох вимірів, такі стовпці в термінології СД і OLAP називаються *атрибутами*. Немає сенсу формувати зведені дані для атрибутів із безперервними значеннями чи атрибута з великим набором різних значень. Уявіть собі, який буде мати вигляд зведена таблиця або діаграма з 1 000 стовпців. Для одержання зведених даних найкраще підходять дискретні атрибути з невеликою кількістю різних значень.

Стовпці з унікальними значеннями ідентифікують рядки. Ці стовпці є *ключами*. У сховищі даних ключі потрібні так само, як і в базах даних. Ключі однозначно ідентифікують об’єкти.

У вимірі СД вам знадобиться один або кілька стовпців, які будуть використовуватися для об’єктів. Замовник зазвичай має адресу, номер телефону та адресу електронної пошти.

Дані в цих стовпцях ви не аналізуєте. Вони не потрібні для формування звіту. Але у звіті часто потрібні такі відомості,

як адреса. Якщо цих даних не буде в СД, доведеться вилучати їх із бази даних, можливо, за допомогою розподіленого запиту. Набагато простіше зберегти ці дані в СД. Стовпці, які застосовуються у звітах лише як мітки, а не для формування звітів, називають *властивостями елементів* (member properties).

У таблицях вимірів стовпцям із властивостями елементів ви можете давати назви кількома мовами, надаючи *переклад* мовами, які повинні підтримуватися. У звітах, що формуються зі сховища даних, доведеться вручну обирати стовпці з перекладами потрібною мовою.

Крім типів стовпців у вимірах, призначених для ідентифікації, іменування, складання звітів та внесення до звітів позначень і міток, можуть застосовуватися стовпці з інформацією *про життєвий шлях даних* (походження та хронології перетворень). Між стовпцями з відомостями про життєвий шлях даних та стовпцями інших типів є істотна різниця: перші ніколи не показуються кінцевим користувачам і ніколи не відображаються у звітах для кінцевих користувачів.

У результаті вимір може містити стовпці таких типів:

– ключі (keys) – застосовуються для ідентифікації об’єктів;

– стовпці імен (name columns) – використовуються для зрозумілих людині назв об’єктів;

– атрибути (attributes) – застосовуються для формування зведень під час аналізу даних;

– властивості елементів (member properties) – використовуються для міток або позначень у звітах;

– стовпці з відомостями про життєвий шлях даних (lineage columns) – застосовуються для аудиту і ніколи не надаються кінцевим користувачам.

Природні ієрархії

До складу виміри DimDate навчальної бази даних входять такі атрибути (рис. 2.3):

– MonthNumberName;
– CalendarQuarter;
– CalendarYear.

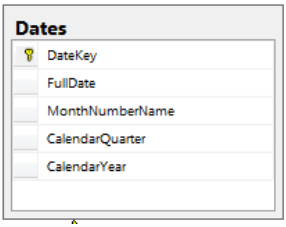

Рисунок 2.3 – Ілюстрація виміру часу

Ви відразу помітите, що ці атрибути пов’язані. Між ними є функціональна залежність, таким чином вони не відповідають третій нормальній формі.

Наведені атрибути утворюють ієрархію. Ієрархії особливо корисні для формування зведених даних та аналітичного оброблення OLAP – вони забезпечують природний шлях деталізації. За допомогою ієрархій можна виконувати аналіз розбиттям або декомпозицією.

Ієрархії мають рівні. Виконуючи деталізацію, ви переходите із батьківського рівня на дочірній. Наприклад, шлях деталізації календаря у вимірі DimDate проходить через такі рівні: CalendarYear → CalendarQuarter → MonthNumberName → → DateKey.

На кожному рівні є елементи. Наприклад, елементи на рівні місяця (month) – це January (січень), February (лютий), March (березень), April (квітень), May (травень), June (червень), July (липень), August (серпень), September (вересень), October (жовтень), November (листопад) та December (грудень). У професійній термінології (жаргоні) СД та OLAP рядки на рівні листя (на нижньому рівні ієрархії) – реальні рядки у вимірі – називаються елементами (members). Ось чому стовпці вимірів, що використовуються у звітах як мітки чи позначення, називаються властивостями елементів (member properties).

Виміри, що повільно змінюються

У сховищах даних існує одна загальна проблема, пов’язана з вимірами: згодом дані у вимірах змінюються. В OLTP-додатку це зазвичай не проблема; коли фрагмент даних змінився, ви просто оновлюєте його. Але у сховищі даних ви

повинні зберігати історію. Питання лише в тому: як це робити. На професійному жаргоні СД цю проблему називають проблемою *вимірів, що повільно змінюється* (*Slowly Changing Dimension, SCD*).

У СД у вас можуть бути ті самі дані, що й у базі даних OLTP. Ви можете, наприклад, використовувати у вимірі Customer той самий ключ, що й у базі даних. Ви могли б оновити стовпець City, отримавши сповіщення від OLTP-системи, і таким чином переписати історію. Такий варіант керування змінами називають *Type 1 SCD*. Іншими словами, Type 1 означає переписування історії атрибуту та всіх більш високих рівнів ієрархій, до яких належить цей атрибут.

Але можливо, ви надаєте перевагу зберегти історію, закріпити факт вкладу замовника в обсяг продажу в іншому місті, країні чи регіоні. У цьому разі ви не можете просто перезаписати дані; замість цього потрібно вставити новий рядок, який містить нові дані. Звичайно, значення в інших стовпцях, які не змінилися, залишаються незмінними. Але постає інша проблема. Якщо просто додати новий рядок для замовника з тим самим значенням ключа, ключ перестане бути унікальним. Насправді, якщо спробувати застосувати первинний ключ або обмеження унікальності як ключ, обмеження відкине таку вставку. Розв'язком може стати введення нового ключа, ключа сховища даних (data warehouse key). У термінології СД цей вид ключа називають *сурогатним ключем* (*surrogate key*).

Збереження історії змін під час додавання нових рядків називається обробленням *Type 2 SCD*. Коли ви реалізуєте Type 2 SCD для спрощення виконання запитів, то зазвичай вставляєте прапорець, щоб позначити рядок, актуальні для елемента виміри. Інший варіант: додати два стовпці, що відображають період дії або коректності значення. Тип даних обох стовпців повинен бути Date, і вони повинні містити значення Valid From (Дійсно з) та Valid To (Дійсно до). Для чинного значення у стовпці Valid To (Справді до) повинне стояти значення NULL.

Незважаючи на те, що здебільшого оброблення змін Type 1 і Type 2 застосовується найчастіше, існують інші

варіанти реалізації. Особливо добре відомий варіант *Type 3 SCD,* у якому виконується керування обмеженою порцією хронологічних даних за допомогою додавання стовпців з історією. Але він застосовується дуже рідко.

Типи стовпців у таблицях фактів

Таблиці фактів – це набори кількісних показників, пов'язаних із конкретним бізнес-процесом. Зберігаються ці показники у стовпцях. Цей тип стовпців називають *мірою* (measure). Міри становлять суть таблиці фактів. Вони зазвичай чисельні та можуть об'єднуватися. Вони зберігають важливі для бізнесу значення, такі як сума продажів, кількість замовлень і знижок.

Таблиця фактів включає зовнішні ключі з усіх вимірів. Ці ключі – другий тип стовпців у таблиці фактів. У взаємовідношеннях із вимірами таблиця фактів знаходиться на стороні «до багатьох». Зазвичай зовнішні ключі всі разом однозначно визначають кожен рядок і можуть застосовуватися як складений первинний ключ.

Часто в таблицю включають додатковий сурогатний ключ. Цей ключ коротший і складається лише з одного-двох стовпців. Сурогатний ключ – це зазвичай бізнес-ключ із таблиці, яка є основним джерелом даних для таблиці фактів.

Останній тип стовпців, що використовуються в таблицях фактів, – стовпці з відомостями про життєвий шлях даних, якщо ви включаєте такі відомості. Як і у випадку вимірів, подібна інформація ніколи не відображається кінцевим користувачам. Отже, у таблиці фактів можуть бути стовпці таких типів:

– зовнішні ключі;
– міри;
– відомості про життєвий шлях даних (необов'язково);
– стовпці з бізнес-ключом із основної вихідної таблиці (необов'язково).

Адитивність мір

Адитивність мір не є проблемою проєктування сховища даних. Але ви повинні розглянути, які статистичні або агрегатні

функції та для яких мір застосовуватимете у звітах, а також які статистичні функції використовуватимете під час групування за конкретним виміром.

Найпростіші типи мір – міри, які можуть бути об'єднані за допомогою агрегатної функції SUM за всіма вимірами, наприклад суми або кількості. Наприклад, якщо сума продажів для товару А була 200 доларів, а сума продажів для товару В – 150 доларів, загальна сума продажів була 350 доларів. Якщо сума вчорашніх продажів була 100 доларів, а позавчорашніх – 130 доларів, то було продано товарів на суму 230 доларів. Міри, які можуть бути підсумовані за всіма вимірами, називають *адитивними мірами (additive measures).*

Деякі міри неадитивні за будь-яким виміром. До прикладів відносять ціни та відсоткові норми, наприклад відсоток знижки. Для таких мір зазвичай застосовують агрегатну функцію AVERAGE, або вони зовсім не зазнають статистичного оброблення. Такі міри називають *неадитивними (non-additive)*. Часто в таких випадках додаються адитивні міри, а потім обчислюються неадитивні міри від адитивних агрегатів. Наприклад, для одержання середньої ціни можна вирахувати загальну суму продажів і потім розділити її на сумарну кількість замовлень. На вищих рівнях групування чи агрегування обчислена ціна – це середня ціна, на найнижчому рівні це дані як такі, тобто обчислена ціна – це реальна ціна. Таким чином ви можете розкладати запити на складові.

Для деяких мір можливе застосування агрегатної функції SUM за всіма вимірами за винятком часу. До прикладів можна віднести ранги та баланси. Такі міри називають *напівадитивними (semi-additive measures).*

Наприклад, якщо на банківському рахунку клієнт А має 2 000 доларів, а клієнт В – 3 000 доларів, у них разом є 5 000 доларів. Але якщо клієнт А вчора мав на рахунку 5 000, а сьогодні в нього залишилося лише 2 000 доларів, це зовсім не означає, що клієнт А має лише 7 000 доларів. Ви повинні уважно стежити за способом статистичного оброблення таких мір у

звітах. Для вимірювання часу можна обчислювати середнє значення або використовувати як агрегат останнє значення.

Тестові питання для підготовки

Для перевірки знання матеріалу дайте відповідь на такі запитання з поясненням чому та чи інша відповідь правильна або неправильна.

A. Нормалізація таблиць бази даних зумовлює зберігання:

- ☐ однорідних даних у пов’язаних таблицях;
- ☐ різнорідних даних в одній таблиці;
- ☐ однорідних даних в одній таблиці.

B. Яка таблиця є центральною у схемі «зірка» сховища даних:

- ☐ таблиця виміру;
- ☐ таблиця фактів?

C. Як пов’язані таблиці вимірів із таблицею фактів у схемі «зірка»:

- ☐ зовнішніми ключами;
- ☐ первинними ключами?

D. Схема «сніжинка» – це:

- ☐ нормалізована схема «зірка»;
- ☐ деталізована схема «зірка»;
- ☐ агрегована схема «зірка»?

E. Формувати звіти у схемі «зірка» простіше, ніж у нормалізованій схемі для оперативного оброблення транзакцій (online transactional processing, OLTP). Чому виникає бажання спростити формування звітів (Виберіть усе, що підходить):

- ☐ у схемі «зірка» зазвичай менше таблиць, ніж у нормалізованій схемі. Тому запити простіші, оскільки вимагають менше з’єднань;
- ☐ у схемі «зірка» краща підтримка числових типів даних, ніж у нормалізованій реляційній схемі, тому легше формувати агрегати;
- ☐ у мові Transact-SQL є спеціальні вирази, які застосовують до схем «зірка»;

☐ схема «зірка» стандартизована й інформативна; ви можете швидко знайти відомості, необхідні для формування звіту?

F. Ви створюєте короткостроковий пробний проєкт. Яка схема найбільше підходить для такого проєкту:

☐ схема «зірка»;
☐ нормалізована схема;
☐ схема «сніжинка»;
☐ XML-схема?

G. У схемі «зірка» застосовуються таблиці двох типів. Які це типи (Виберіть усе, що підходить.):

☐ таблиці підстановки;
☐ виміри;
☐ міри;
☐ таблиці фактів?

H. Ви реалізуєте для конкретного стовпця варіант Type 2 вирішення проблеми SCD. Що потрібно зробити під час одержання з вихідної системи зміненого значення для цього стовпця:

☐ вставити в таблицю стовпець для зберігання попереднього значення; перемістити поточне значення стовпця, що оновлюється, у новий стовпець; замінити поточне значення новим значенням із вихідної системи;

☐ вставити новий рядок для того самого елемента виміру з новим значенням в оновлюваному стовпці; застосувати сурогатний ключ, оскільки бізнес-ключ тепер не унікальний; додати прапор, що свідчить про те, який рядок елемента актуальний або дійсний;

☐ нічого не робити, оскільки у сховищі даних підтримуються відомості про історію даних, вам не потрібно оновлювати дані у вимірах;

☐ оновити значення в стовпчику точно так само, як це було зроблено у вихідній системі?

I. Який тип стовпця не є частиною виміру:

☐ атрибут;
☐ міра;

☐ ключ;
☐ властивість елемента;
☐ ім’я?

J. Як знайти природні ієрархії у схемі «сніжинка»:

☐ необхідно проаналізувати вміст атрибутів у всіх вимірах;
☐ природні ієрархії зберігаються в таблицях уточнювальних запитів;
☐ схема «сніжинка» не підтримує природні ієрархії;
☐ необхідно перетворити схему «сніжинка» на схему «зірка» і тоді можна буде відразу виявити природні ієрархії?

K. За яким виміром не потрібно застосовувати агрегатну функцію SUM для напівадитивних мір:

☐ Customer;
☐ Product;
☐ Date;
☐ Employee?

L. Які міри, на вашу думку, неадитивні (Виберіть усі можливі варіанти.):

☐ Price (Ціна);
☐ Debit (Дебет);
☐ SalesAmount (Сума продажів);
☐ DiscountPct (Відсоток знижки);
☐ UnitBalance (Залишок в одиницях товару)?

M. Стовпець якого типу не є частиною таблиці фактів:

☐ відомості про життєвий шлях даних;
☐ міра;
☐ ключ;
☐ властивість елемента?

Аналітичні питання

Завдання 1. Короткостроковий пробний проєкт.

Вас взяли на роботу для реалізації короткострокового пробного проєкту сховища даних. Ви повинні підготувати схему для даних про продажі. Спеціалісти відділу продажів вашого

замовника хотіли б аналізувати дані про клієнтів, товари та період часу. Дайте відповіді на питання:

1.1. Яку схему ви почали б використовувати?

1.2. Якими могли б бути виміри у вашій схемі?

1.3. На які типи мір ви орієнтовані?

Завдання 2. Розширення короткострокового пробного проєкту.

Після того, як ви реалізували короткостроковий пробний проєкт сховища даних про продажі із завдання 1, ваш замовник замовив розширення проєкту до реального довгострокового сховища даних. Для відповіді на питання необхідно врахувати такі побажання спеціалістів:

– спеціаліст із відділу продажів: «Я не бачу коректних агрегатів за регіонами для ретроспективних даних»;

– адміністратор бази даних, що створює звіти: «Мої запити все одно ускладнені, з багаточисельними підключеннями».

Дайте відповіді на питання:

2.1. Як би ви вирішили проблему спеціаліста відділу продажів?

2.2. Схему якого типу ви би реалізували для довгострокового СД?

2.3. Як би ви вирішували проблему адміністратора бази даних?

МАТЕРІАЛ ДЛЯ ПІДГОТОВКИ ДО ВИКОНАННЯ ПРАКТИЧНОЇ РОБОТИ 3

Теоретичний матеріал

Реалізація вимірів і таблиць фактів

Головні об'єкти бази даних – це виміри і таблиці фактів. Для прискорення ETL-процесу у вашому сховищі даних можна включити додаткові об'єкти, такі як послідовності, що зберігаються процедури, та проміжні таблиці. Після того, як об'єкти створені, їх потрібно протестувати, завантаживши тестові дані.

Під час створення бази даних сховища даних необхідно розглянути кілька особливостей. СД містить перетворену копію LOB-даних (дані з транзакційної бази даних). Ви завантажуєте дані у своє СД періодично, за розкладом, зазвичай у нічні години. Для СД немає потреби створювати резервну копію журналу транзакцій як для бази даних LOB. Отже, для СД необхідно вибрати просту модель відновлення.

SQL Server підтримує три моделі відновлення.

1. У моделі повного відновлення (*full recovery model*) усі транзакції реєструються повністю з усіма пов'язаними даними. Необхідно регулярно створювати резервну копію журналу транзакцій. У цьому випадку можна відновити дані на будь-який момент часу. Відновлення на заданий момент особливо корисно, коли виникають помилки, допущені людьми.

2. Модель відновлення з неповним протоколюванням (*bulk logged recovery model*) – додаток до повної моделі відновлення, що дозволяє виконувати високопродуктивні операції масового копіювання. Масові операції, наприклад, створення індексу або масове завантаження тексту або XML даних, можуть протоколюватися в мінімальному обсязі. Для таких операцій SQL Server може реєструвати лише інструкцію Transact-SQL без усіх пов'язаних даних. Проте все одно необхідно регулярно створювати резервну копію журналу транзакцій.

3. У простій моделі відновлення (*simple recovery model*) SQL Server автоматично звільняє місце у журналі зафіксованих транзакцій. SQL Server підтримує мінімальні вимоги до місця на диску для журналу транзакцій, істотно знижуючи потребу в керуванні відведеним місцем.

Проста модель відновлення зручна на етапі розроблення, тестування та баз даних із величезним переважанням операцій зчитування даних. Оскільки у сховищі даних ви переважно використовуєте дані в режимі «лише читання», для СД найбільше підходить проста модель відновлення. Якщо застосовуються повна модель чи модель із неповним протоколюванням, необхідно регулярно створювати резервні копії журналу, який постійно збільшується з кожним новим завантаженням даних.

Дані бази даних SQL Server та файли журналів можуть розростатися та стискатися автоматично. Але зростання відбувається у невідповідний час – коли ви завантажуєте нові дані, накладаючись на ваше завантаження та сповільнюючи завантаження. Численні операції, що викликають незначне збільшення, можуть фрагментувати ваші дані. Автоматичне стиснення може фрагментувати дані навіть більше. Для запитів, які зчитують великий обсяг даних та виконують перегляд великих таблиць, виникає бажання обмежити фрагментування наскільки це можливо. Отже, ви повинні перешкодити автостисканню та автозбільшенню. Переконайтеся, що режим бази даних *Auto Shrink* вимкнено. Від початку необхідно зарезервувати на диску достатньо місця для даних та реєстраційних журналів, щоб перешкодити їхньому автозбільшенню.

Підрахувати необхідну кількість дискового простору дуже легко. Сховище даних містить дані впродовж декількох років, зазвичай 5 або 10 років. Завантажте тестові дані за обмежений період, наприклад, за рік (або за місяць, якщо ви маєте справу з дуже великими вихідними базами даних). Перевірте розмір файлів вашої бази даних та екстраполюйте його на повний обсяг даних за 5 або 10 років. Крім того,

необхідно додати щонайменше 25 % додаткового вільного простору до ваших файлів даних. Цей додатковий простір дозволить без фрагментації перебудовувати чи створювати заново індекси.

Незважаючи на те, що в простій моделі відновлення журнал транзакцій не зростає, ви все одно повинні задати його досить великим, щоб у ньому вмістилася найбільша транзакція. Звичайні інструкції мови маніпулювання даними (DML), включаючи INSERT, DELETE, UPDATE та MERGE, завжди протоколюються повністю, навіть у простій моделі. Потрібно перевіряти виконання цих інструкцій та оцінювати необхідний розмір журналу транзакцій.

У сховищі даних зазвичай більшу частину простору займають великі таблиці фактів. Оптимізувати виконання запитів та керування великими таблицями фактів можна за допомогою секціонування (*Partitioning*). Секціонування таблиць полегшує керування та забезпечує виграш у продуктивності. Запити часто торкаються лише підмножини секцій, і SQL Server вміє виключати решту секцій на ранній стадії процесу виконання запиту.

У базі даних може бути безліч файлів, згрупованих у численні файлові групи (*filegroups*). Немає загальної рекомендації щодо кількості файлових груп, які потрібно створювати у вашому сховищі даних. Проте для більшості проєктів СД – найкращий варіант: одна файлова група для кожної секції. Кількісті файлів у файловій групі залежить від обсягу вашого дискового простору. Зазвичай потрібно створювати один файл на фізичний диск.

Завантаження даних із вихідних систем часто дуже складне. Для полегшення завдання можна створити у своєму СД проміжні таблиці. Ви можете навіть помістити проміжні таблиці та інші об’єкти в окрему базу даних. Проміжні таблиці (*staging tables*) застосовуються для тимчасового збереження вихідних даних до їх очищення та об’єднання з даними інших джерел. Крім того, проміжні таблиці також відіграють роль проміжного шару між СД та вихідними таблицями. Якщо у вихідних

таблицях щось змінилося, наприклад, якщо початкова база даних оновлена, ви повинні змінити лише запит, який зчитує вихідні дані та завантажує їх у проміжні таблиці. Після цього ваш регулярний ETL-процес повинен виконуватися так само, як це робилося до внесення змін до вихідної системи. Частина СД, що містить проміжні таблиці, називається проміжною областю даних (*data staging area, DSA*).

Проміжні таблиці ніколи не показуються кінцевим користувачам. Якщо вони становлять частину вашого СД, можна зберігати їх не разом із звичайними таблицями зі схеми «зірка», а в іншій схемі. Зберігаючи проміжні таблиці в окремій схемі, ви можете надати кінцевим користувачам дозволи для звичайних таблиць СД, обмеживши ці дозволи лише відповідною схемою, що спростить адміністрування. У типовому сховищі даних достатньо двох схем: одна із звичайними таблицями СД та інша з проміжними таблицями. Звичайні таблиці СД можна зберігати у схемі dbo, і за необхідності створити окрему схему для проміжних таблиць.

Реалізація вимірів

Реалізація виміру передбачає створення таблиці, що містить усі необхідні стовпці. Крім бізнес-ключів, ви повинні включити сурогатний ключ у всі виміри, що потребують керування змінами *Type 2 Slowly Changing Dimension (SCD)*. Під час реалізації цього способу керування виміром необхідно так само вставити стовпець-прапор для позначення актуального рядка або два стовпці для зберігання дат, що позначають термін дії чи актуальності рядка.

Для сурогатних ключів можна використовувати прості послідовні цілі числа. SQL Server може створити автонумерацію у такому стовпці. Для створення послідовних чисел можна скористатися властивістю IDENTITY.

Послідовність – це визначений користувачем, незалежний від таблиці (отже, прив’язаний до схеми) об’єкт. SQL Server застосовує послідовності для генерації низки числових значень відповідно до заданих вами параметрами. Можна генерувати зростаючі та спадні послідовності, задаючи

інтервал можливих значень. Можна навіть генерувати циклічні (повторювані) послідовності. Керування зв'язком між послідовностями і таблицями у вашому ETL-додатку відбувається за допомогою послідовностей, якими можна координувати значення ключів у кількох таблицях.

Ви повинні застосовувати замість стовпців ідентифікаторів послідовності у таких сценаріях:

– установлення наступного числа до виконання вставки в таблицю;

– спільне використання однієї послідовності чисел у кількох таблицях і навіть у кількох стовпцях в одній таблиці;

– досягнення заданого числа через повторний запуск генерації послідовних чисел спочатку (створення циклічної послідовності);

– сортування послідовних значень, відсортованих за другим стовпцем. Функція NEXT VALUE FOR, яка викликається для виділення послідовних значень, може використовувати речення OVER. В OVER можна згенерувати послідовність у порядку, заданому через ORDER BY, включеним у речення OVER;

– коли вам потрібно привласнити кілька номерів в один і той самий час. Запит значень ідентифікаторів може призвести до розривів у послідовності номерів, якщо інші користувачі в цей момент теж генерували послідовні номери. Для того щоб отримати кілька послідовних номерів одночасно, можна викликати системну процедуру *sp_sequence_get_range*;

– коли потрібно змінити параметри послідовності, наприклад, значення збільшення;

– якщо потрібно досягти вищої продуктивності, ніж у разі застосування стовпців ідентифікаторів. Можна використовувати для створення послідовності параметр *CACHE*. Він підвищує продуктивність, мінімізуючи кількість дискових операцій ведення / виведення, яка потрібна для генерації чисел послідовності. Якщо розмір кешу 50 (це значення за замовчуванням), SQL Server зберігає в кеші лише поточне значення та кількість значень, залишених у кеші, тобто

необхідний обсяг пам’яті еквівалентний двом екземплярам із типом даних об’єкта-послідовності.

Крім звичайних стовпців, ви також можете додавати у вимір стовпці, що обчислюються (*computed columns*). Стовпець, що обчислюється, – це віртуальний стовпець у таблиці. Значення стовпця визначається виразом.

Описуючи у таблицях обчислювані стовпці, ви спрощуєте запити. Вони також можуть допомогти підвищити продуктивність. Ви можете зберігати та індексувати обчислюваний стовпець, якщо дотримані такі попередні умови:

- вимоги до володіння;
- вимоги до детермінованості;
- вимоги до точності;
- вимоги до типу даних;
- вимоги до параметра SET.

Обчислювані стовпці можна застосовувати для дискретизації значень, що безперервно змінюються, у вихідних стовпцях. Обчислювані стовпці особливо корисні для мінливих значень у стовпцях. Прикладом значення, що безперервно змінюється, може бути вік. Припустимо, що у вас є дати народження замовників чи співробітників; для аналітичного оброблення, можливо, вам знадобиться їхній вік. Вік змінюється кожного дня, з кожним новим завантаженням. Ви можете дискретизувати вік у дві групи. Після цього значення не будуть змінюватися так часто. Крім того, вам не потрібно підтримувати та індексувати обчислюваний стовпець. Якщо значення стовпця, що обчислюється, не зберігаються, SQL Server обчислює їх на льоту, коли вони будуть потрібні запиту. Якщо ви застосовуєте SQL Server Analysis Services (SSAS), можна зберігати цей стовпець фізично в базі даних SSAS, і таким чином підтримувати його в SSAS.

І нарешті, якщо вам потрібна інформація про життєвий шлях даних, які її містять стовпці, також потрібно вносити до ваших вимірів.

Реалізація таблиць фактів

Після створення вимірів необхідно сформувати у сховищі таблиці фактів. Таблиці фактів необхідно створювати після реалізації вимірів. У взаємозв'язку з виміром таблиця фактів знаходиться на стороні «*до багатьох*», тому, якщо потрібно створити обмеження зовнішнього ключа, батьківський об'єкт вже повинен існувати.

Для полегшення керування та підвищення продуктивності можна секціонувати велику таблицю фактів.

Стовпці у таблиці фактів містять зовнішні ключі та виміри. Зовнішні ключі у базі даних визначаються у вимірах. Усі разом зовнішні ключі зазвичай однозначно визначають кожен рядок таблиці фактів. Якщо вони справді однозначно визначають кожен рядок, їх можна використовувати як складений ключ. Можна також додати додатковий первинний сурогатний ключ, який може бути ключем, успадкованим із таблиці транзакційної системи. Наприклад, якщо починати побудову в СД таблиці фактів із даними про продаж із транзакційної таблиці, що містить подробиці про замовлення на продаж, можна використовувати ключ цієї таблиці для таблиці фактів про продаж вашого СД.

У виробничому середовищі можна видалити обмеження зовнішнього ключа, щоб підвищити продуктивність завантажень. Якщо ці обмеження є, SQL Server змушений перевіряти їх під час завантаження. Проте рекомендовано залишати обмеження зовнішніх ключів на час розроблення та на етапах тестування. Якщо визначено зовнішні ключі, простіше створювати діаграми баз даних. Крім того, якщо обмеження будуть порушені, буде одержано повідомлення про помилки під час тестування. Помилки свідчать, що з даними щось негаразд; якщо виникає порушення для зовнішнього ключа, швидше за все, у вимірі пропущено батьківський рядок для одного або кількох рядків із таблиці фактів. Подібні помилки надають інформацію щодо якості даних.

Якщо у виробничому середовищі ухвалено рішення видалити зовнішні ключі, потрібно так організувати

ETL-процес, щоб він був стійкий до подібних помилок. У ETL-процесі, коли в таблиці фактів з'являється невідомий ключ, потрібно додавати рядок у вимір. Рядок, що додається у вимір під час завантаження таблиці фактів, називають виведеним елементом (*inferred member*). Під час завантаження таблиці фактів значення в усіх стовпцях рядка виведеного елемента виміру, крім ключів, невідомі і мають значення NULL. Це означає, що в усіх стовпцях виміру (крім ключів) повинні бути дозволені значення NULL. Майстер SCD (повільно змінюваних вимірів) у службах SQL Server Integration Services (SSIS) допоможе обробити під час завантаження елементи, що виводяться у вимірі. Проблему виведених елементів також називають проблемою вимірів, що запізнюються або пізно надходять (*late-arriving dimensions problem*).

Таблиці фактів, як і виміри, можуть містити стовпці, що обчислюються. Можна заздалегідь виконати низку обчислень і спростити запити. І звичайно, таблиці фактів, як і виміри, можуть мати стовпці з інформацією про життєвий шлях даних, які додаються до таблиці, якщо в них є потреба.

Тестові питання для підготовки

A. Який зі стовпців таблиць сховища даних найбільш підходить для створення кластеризованого індексу:

- ☐ сурогатні ключі;
- ☐ атрибути таблиць вимірів;
- ☐ зовнішні ключі?

B. Яка з моделей відновлення даних найбільш «економічна» на етапі розроблення сховища даних:

- ☐ bulck logged recovery model;
- ☐ full recovery model;
- ☐ simple recovery model?

C. Проміжні таблиці сховища даних призначені для зберігання даних:

- ☐ до очищення;
- ☐ до об'єднання;
- ☐ після очищення;
- ☐ після об'єднання.

D. Який із двох варіантів (ідентифікатор чи послідовність) необхідно використовувати для ключа за умови зміни значення поля на три одиниці:

☐ послідовність;

☐ ідентифікатор?

E. Які об’єкти бази даних і які властивості об’єкта можна використовувати для формування автоматичної нумерації (Виберіть усі відповідні варіанти.):

☐ властивість IDENTITY;

☐ об’єкт SEQUENCE;

☐ обмеження PRIMARY KEY;

☐ обмеження CHECK?

F. Які стовпці вставляються в таблицю для оброблення змін способом Type 2 SCD (Виберіть усі відповідні варіанти.):

☐ властивості елементів;

☐ позначка поточного (актуального) рядка;

☐ стовпці з інформацією про життєвий шлях даних;

☐ сурогатний ключ?

G. Що таке виведений елемент:

☐ рядок у таблиці фактів, що вставляється під час завантаження виміру;

☐ рядок з агрегатними значеннями;

☐ рядок у вимірі, що вставляється під час завантаження таблиці фактів;

☐ обчислюваний стовпець у таблиці фактів?

H. Які види стиснення даних підтримуються в SQL Server (Виберіть усі відповідні варіанти.):

☐ Bitmap;

☐ Unicode;

☐ Рядок;

☐ сторінок?

I. Які оператори отримують виграш від застосування пакетного оброблення (Виберіть усі відповідні варіанти.):

☐ Hash Join;

☐ Merge Join;

☐ сканування;

☐ Nested Loops Join;

☐ фільтр?

J. Чому потрібно застосовувати індексовані подання: (Виберіть усі відповідні варіанти.):

☐ для прискорення запитів, що агрегують дані;

☐ для прискорення завантаження даних;

☐ для прискорення вибіркових запитів;

☐ для прискорення запитів, що включають кілька з'єднань?

K. Об'єкт бази даних, що відображає секції таблиці на файлові групи, називають:

☐ вирівняний індекс;

☐ функція секціонування;

☐ стовпець секціонування;

☐ схема секціонування.

L. Якщо ви хочете переключити вміст із несекціонованої таблиці на секцію секціонованої таблиці, які умови повинна задовольняти несекціонована таблиця (Виберіть усі відповідні варіанти.):

☐ в неї повинні бути ті самі обмеження, що й у секціонованої таблиц;

☐ в неї повинен бути той самий тип стиснення, що й у секціонованої таблиці;

☐ вона повинна бути в особливій схемі PartitionedTables;

☐ в неї повинне бути таке перевірне обмеження для стовпця секціонування, яке гарантує розміщення всіх її даних рівно в одній секції секціонованої таблиці;

☐ в неї повинні бути такі самі індекси, як і в секціонованої таблиці?

M. Які з перелічених функцій T-SQL не дуже корисні для збору інформації про життєвий шлях даних:

☐ APP_NAME();

☐ USER_NAME();

☐ DEVICE_STATUS();

☐ SUSER_SNAME()?

Аналітичні питання

Завдання 1. Повільні звіти сховища даних.

Припустимо, що ви створили сховище даних і заповнили його. Користувачі почали використовувати його для одержання звітів. Також вони почали скаржитися на швидкодію. У звіті дайте відповіді на такі питання:

1.1. Що можна змінити у вашому сховищі даних для прискорення формування звітів з обчисленням зростаючих підсумкових значень?

1.2. Чи має сенс перевіряти початкові запити, які формують звіти зі зростаючими підсумками?

Завдання 2. Проблеми адміністрування сховищ даних.

Ваші користувачі задоволені швидкодією формування звітів зі сховища даних, але під час розмови з адміністратором ви дізналися про потенційні проблеми. Журнал транзакцій кожну ніч збільшується на 10 Гбайт. Крім того, користувачі почали створювати звіти з проміжних таблиць, і ці звіти містять суперечливі дані. Користувачі скаржаться адміністраторові бази даних на те, що не можуть довіряти вашому сховищу даних, якщо у звітах одержують такі брудні дані. У звіті дайте відповіді на такі питання:

2.1. Як можна впоратися з нестримним зростанням обсягу журналу?

2.2. Як можна запобігти кінцевим користувачам застосовувати проміжні таблиці?

МАТЕРІАЛ ДЛЯ ПІДГОТОВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ 4

Теоретичний матеріал

Розроблення пакетів SSIS

Переміщення даних – важлива складова керування даними. Дані клієнтських додатків доставляються на сервер даних для зберігання і передаються назад із бази даних клієнтові для використання та керування. У сховищах даних переміщення даних представляє особливо важливий компонент, який враховує такі потреби СД:

– необхідність імпорту даних з однієї або декількох операційних баз даних;

– необхідність очищення та об'єднання даних;

– необхідність перетворення, що дозволяє зберігати та належним чином обслуговувати дані у сховищі даних.

Microsoft SQL Server пропонує спеціально розроблені продукти для цього набору конкретних потреб: служби *SQL Server Integration Services* (*SSIS*). На відміну від операцій галузевого керування даними, у яких окремі бізнес-елементи почергово обробляються в клієнтському додатку людьми, операції СД виконуються над наборами бізнес-елементів автоматично. Зважаючи на цю істотну різницю, *SSIS* надає засоби для ефективного виконання операцій над великими обсягами даних і, наскільки можливо, без втручання людини.

Ще одна відмінність систем керування галузевих даних та СД – час виконання операцій; операції галузевого керування даними переважно виконуються у звичайні робочі години, а операції СД – у періоди технічного обслуговування. Зазвичай таке обслуговування в СД призначається на час із найменшим робочим навантаженням або повної його відсутності (наприклад, у нічний час). Робиться це для зниження впливу ресурсомістких операцій на галузеву систему та зменшення впливу мінливості даних на СД.

Сценарії переміщення даних за ступенем складності можна поділити на дві групи:

– прості переміщення даних (*simple data movements*), коли дані доставляють із джерела в кінцевий пункт призначення, «як є» (без змін);

– складні переміщення даних (*complex data movements*), коли дані перед збереженням потребують перетворення і виникає необхідність додаткового програмного алгоритму злиття нових та/або модифікованих даних, що надійшли з вихідної системи, з існуючими даними, які вже наявні в кінцевому місці призначення.

З урахуванням цього набір інструментів *SQL Server* надає два різні підходи до розроблення способів переміщення даних:

– *SQL Server Import and Export Wizard* (*Майстер імпорту та експорту SQL Server*), який можна застосовувати для проєктування (і виконання) простих переміщень даних, наприклад, передавання даних з однієї бази даних в іншу;

– *SQL Server Data Tools* (*Комплект інструментів*), який має повне інтегроване середовище, що надає користувачам служб *SQL Server Integration Services* (*SSIS*) можливість розробляти навіть найскладніші способи переміщення даних.

Складові процесу переміщення даних

У процесі складного переміщення даних необхідно звертати увагу на три різні компоненти:

1. Дані вилучаються із вихідної системи (бази операційних даних).
2. Дані перетворюються (очищаються, конвертуються, реорганізуються та реструктуруються) відповідно до вимог кінцевої моделі даних.
3. Дані завантажуються до кінцевого сховища (наприклад, сховища даних).

Цей структурований процес називають *вилученням – перетворенням – завантаженням* або *ETL* (*extract – transform – load*). У простих переміщеннях даних компонентів перетворення немає, і залишаються лише два компоненти: витяг і завантаження.

Переміщення даних у сховищах даних

У типових сценаріях сховищ даних лише деякі переміщення даних не вимагають жодних перетворень; переважній більшості переміщень необхідні принаймні структурні зміни, щоб дані стали відповідати моделі даних, застосовується у сховищі даних.

Для сценаріїв простих переміщень даних, особливо якщо на розроблення відведено зовсім мало часу, застосування багатофункціонального середовища розробки з усіма доступними інструментами та функціями може призвести до безлічі непотрібних витрат. Дійсно, за простого переміщення даних потрібні лише вихідна система, кінцеве місце призначення та спосіб ініціювання передавання. *SQL Server* пропонує спрощений інтерфейс розробки, власне покроковий майстер, який добре підходить для найпростіших переміщень даних, *SQL Server Import and Export Wizard* (*Майстер імпорту та експорту SQL Server*).

Якщо перетворення не потрібні, майстер *Import and Export Wizard може бути* засобом, який підходить для виконання цієї роботи. Якщо ж перетворення потрібні, можливо, ви все одно зможете використати *Import and Export Wizard* за умови, що всі перетворення можуть бути виконані всередині запиту *SELECT*, який застосовується під час видалення даних із вихідної системи.

Якщо в кінцевому місці призначення немає даних (наприклад, місце призначення ще не існує), можна застосовувати майстер *Import and Export Wizard*. Його також можна використовувати, якщо дані в кінцевому місці призначення вже є, але не потрібно поєднувати їх з новими даними (наприклад, коли в кінцевому місці призначення дозволені дублікати).

Незважаючи на те, що *Import and Export Wizard* дозволяє попутно створити базу даних у кінцевому місці призначення, він дозволяє вказати для новостворених файлів бази даних лише початковий розмір та величину збільшення; але не дозволяє задати місцезнаходження файлів. Якщо в місцезнаходженні за

замовчуванням достатньо вільного простору, можна використовувати *Import and Export Wizard*.

Створення пакетів SSIS

SQL Server Import and Export Wizard не має функціональних можливостей для модифікації (перетворення) даних перед їх збереженням у кінцевому місці призначення, а також для відповідного об'єднання нових модифікованих даних із наявними даними, отже, майстер – невідповідний засіб для сценаріїв складного переміщення даних.

Проте майстер можна застосовувати, наприклад, для швидкого розроблення пакета *SSIS*, його тестування і запуску для того, щоб якнайшвидше ввести в дію сховище даних, а пізніше використовувати середовище *SQL Server Data Tools* (*SSDT*) і додати функціональні можливості, що полегшують повторне використання та керованість розробленого пакета.

З іншого боку, сценарії розроблення сховищ даних зазвичай не відносять до проєктів, для яких найважливіше швидке розроблення ухваленого рішення; здебільшого проєкти сховищ даних, перед тим як будуть здійснені будь-які реальні кроки щодо їх розроблення, вимагають великої дослідницької роботи та ретельного планування, спрямованих на створення закінченого готового до роботи розв'язку щодо переміщення даних. Коли етап планування закінчено, швидко розроблений на ранній стадії процес переміщення даних, можливо, вже застаріє, і його доведеться змінювати для введення сховища в експлуатацію. Таким чином, малоймовірно, що виграш від раннього розроблення переважить необхідність перероблення та можливої реконструкції процесу переміщення даних після того, як розробка сховища даних дійсно дозріє для введення в експлуатацію. Іншими словами, розроблення *SSIS* зазвичай виконується з нуля із застосуванням *SSDT*.

У службах *SSIS* застосовується спеціальна декларативна мова програмування, точніше кажучи, спеціальний програмний інтерфейс для визначення порядку та умов виконання операцій. Додатки з упором на автоматизацію, обслуговування СД, сильно відрізняються від більшості додатків тим, що не підтримують

інтерфейси користувача. Моніторинг, контроль, пошук та усунення несправностей забезпечуються функціональними засобами аудиту та протоколювання чи реєстрації.

Методи, що застосовуються в проєктуванні та розробленні *SSIS*, можуть істотно відрізнятися від інших програмних методів і здаватися особливо дивними адміністраторам баз даних або розробникам, які більше звикли до інших мов програмування (наприклад, Transact-SQL або Microsoft.NET). Більшість розроблень у *SSIS* виконується графічно – за допомогою мишки, а не введенням команд із клавіатури. Візуальний підхід до розроблення дозволяє не лише налаштувати операції, визначити їх порядок та зазначити умови їх виконання, але також забезпечує варіант програмування WYSIWYG.

Після того як пакети *SSIS* введені в дію, вони зазвичай виконуються автоматично, наприклад під час використання *SQL Server Agent*, але їх можна запускати і на вимогу за допомогою утиліт, запропонованих платформою, або за допомогою інтерфейсу прикладного програмування (*application programming interface, API*).

Вступ до SSDT

SSDT – спеціальна версія *Visual Studio*, основного інтегрованого середовища розроблення корпорації Microsoft. *SSDT* підтримує розроблення різних проєктів *SQL Server*, таких як проєкти *SQL Server Analysis Services Multidimensional* та *Data Mining*, проєкти *Analysis Services Tabular*, проєкти *Report Server служб SQL Server Reporting Services* та проєкти служб *Integration Services* (*SSIS*). Для всіх проєктів перелічених типів пакет *SSDT* надає повне інтегроване середовище розробки, особливо налаштоване на певний тип проєкту.

Для проєктів служб *Integration Services SSDT* пропонує цілий арсенал завдань керування даними та компонентів, які забезпечують значною мірою будь-які потреби сховищ даних (безліч різноманітних методів вилучення, перетворення та завантаження даних). Модель розробки SSIS розширюється: набір вбудованих інструментів можна поповнити, додавши

завдання користувача та/або компоненти, розроблені сторонніми організаціями або безпосередньо користувачами-розробниками.

Будь-який пакет *SSIS* містить три важливих компоненти.

Диспетчери з'єднань (*connection managers*) забезпечують підключення до сховищ даних, як до вихідних систем, так і до кінцевих місць призначення. Оскільки те саме сховище даних може відігравати роль і джерела даних, і місця призначення, диспетчери з'єднань дозволяють в одному пакеті (або проєкті) визначати з'єднання один раз і застосовувати багаторазово.

Потік керування (*control flow*) визначає як порядок операцій, і умови виконання. Пакет може складатися з однієї або кількох операцій, що представлені завданнями потоку керування (*control flow tasks*). Порядок виконання визначається тим, як пов'язані окремі завдання один з одним. Завдання, у яких немає попереднього завдання, як і завдання, у яких те саме попереднє завдання виконуються паралельно.

Потік даних (*data flow*) включає компоненти переміщення даних, ETL:

– один або кілька компонентів-джерел, що позначають сховища даних, з яких дані виймаються;

– один або кілька компонентів місць призначення, що позначають сховища даних, які дані будуть завантажуватися;

– один або кілька компонентів перетворень, що позначають перетворення, яких зазнають дані.

Інтегроване середовище розроблення *SSDT* пропонує уніфікований та комплексний підхід до розроблення баз даних; результуючі файли служб *Analysis Services, Reporting Services* та *Integration Services* обслуговуються одним і тим самим середовищем IDE з цілком зрозумілими необхідними налаштуваннями, що враховують відмінності різних моделей, що розробляються.

Навіть якщо говорити лише про продукти служб *Integration Services*, проєкт *SSIS* у всій своїй повноті може бути складнішим (і зазвичай набагато складнішим), ніж лише переміщення даних. Середовище IDE надає можливість розробляти, обслуговувати та розгортати множинні процеси

переміщення даних, які в реальному житті становлять ту саму закінчену логічну одиницю оброблення, що й один проєкт.

На додачу до сказаного, зазвичай у сценаріях зберігання даних переміщення даних насправді представляють лише одне з кількох елементів одержання, обслуговування і споживання даних, необхідні підтримки бізнес-середовища. Потреби бізнесу також повністю підтримуються інтегрованим середовищем розробки *SSDT* – множинні проєкти, що охоплюють численні елементи платформи *SQL Server*, що є будівельними блоками єдиної бізнес-моделі, можуть розроблятися і обслуговуватися як єдиний розв'язок *SSDT*.

Для забезпечення ізоляції розроблення й тестування від виробничих операцій розроблення пакет *SSIS* повинен проводитись у спеціалізованих середовищах, в ідеалі без безпосереднього доступу до робочого середовища. Лише після завершення розроблення та проведення належних тестів рішень готові пакети *SSIS* необхідно розгортати в робочому середовищі. Але навіть на стадіях планування та розроблення ви повинні не лише знати про відмінності між середовищем розробки та виробничим середовищем, а й брати до уваги ще перед спробою розгортання пакета.

Зазвичай основна відмінність між середовищем розробки та виробничим робочим середовищем – у конфігурації сховищ даних. У середовищі розробки всі дані можуть міститися на одному сервері (навіть в одній і тій самій базі даних). Дійсно, оскільки для розроблення необхідна (і є в розпорядженні) лише підмножина даних, усі дані розробки легко можуть вміститися в персональному комп'ютері розробника. Отже, під час розроблення рішень SSIS ви повинні враховувати такі відмінності в середовищі розробки та виробничого середовища:

– *З'єднання.* У робочому середовищі вихідне сховище даних та сховище призначення найчастіше будуть міститися на різних серверах.

– *Платформи даних.* Робочі версії платформ даних можуть відрізнятися від тих, які застосовувалися під час розроблення або можуть застосовуватися зовсім різні

платформи (наприклад, для розроблення SQL Server, а в робочому середовищі – інша СУБД).

– *Безпека.* Зазвичай не потрібно, щоб машина розробника була частиною того самого домену операційної системи, що й робочі сервери. Більше того, робочі сервери, що зберігають вихідні сховища даних та кінцеві, можуть розміщуватись у різних доменах.

Починаючи з SQL Server 2012 функція налаштування замінена параметризацією (*parameterization*), яка, по суті, надає ті самі функціональні можливості (наприклад, можливість керувати всіма оголошеними властивостями будь-якого об'єкта, що настроюється, в пакеті SSIS через конфігурацію, що дозволяє адміністраторові налаштувати пакет відповідно до середовища, в якому він застосовуватиметься).

Тестові питання

A. Переміщення даних у системах зі сховищами даних зумовлене необхідністю:

☐ переміщення даних зі сховища до клієнтських додатків;

☐ переміщенням даних між вітринами сховищ даних;

☐ переміщенням даних із різних джерел даних до сховища даних.

B. Чим відрізняються процеси простого та складного переміщення даних:

☐ наявністю додаткових операцій оброблення даних;

☐ наявністю декількох адресатів переміщення даних;

☐ потребою в додатковому обладнанні для переміщення даних?

C. Диспетчер з'єднань пакета SSIS відповідає за:

☐ під'єднання до джерел даних;

☐ під'єднання до сховищ даних;

☐ під'єднання до адресатів даних.

D. Вам потрібно перемістити дані з робочої бази даних у тестову базу даних. Необхідно витягти дані з декількох об'єктів у вихідній базі даних, але ваш менеджер попросив скопіювати лише близько 10 % рядків із найбільших робочих таблиць. База

даних, що тестується, вже існує, але в ній немає жодних таблиць. Як ви вирішити це завдання (Виберіть усі відповідні варіанти.):

☐ застосуйте Import and Export Wizard, скопіюєте всі таблиці з вихідної бази даних у порожню базу даних кінцевого місця призначення й видалите зайві рядки з найбільших таблиць;

☐ застосуйте Import and Export Wizard кілька разів – один раз для таблиць поменше і по одному разу для кожної великої таблиці, використовуючи варіант Write a query to specify the data to transfer (написати запит, що зазначає дані для передавання), щоб обмежити кількість рядків;

☐ застосуйте Import and Export Wizard, скопіюєте всі таблиці з вихідної бази даних у порожню базу даних кінцевого місця призначення й обмежите кількість рядків у кожній великій таблиці за допомогою натискання кнопки Edit SQL (Редагувати SQL) з вікна Column Mappings (Порівняння стовпців);

☐ застосуйте Import and Export Wizard, налаштуйте його на копіювання всіх таблиць із вихідної бази даних у порожню базу даних у кінцевому місці призначення, збережіть пакет SSIS, а потім перед його виконанням відредагуєте його за допомогою середовища SSDT, щоб обмежити кількість рядків, що витягуються з великих таблиць?

E. Ви повинні перемістити дані з операційної бази даних у сховище даних уперше. Сховище даних уже створено і містить деякі еталонні дані. Ви щойно закінчили в операційній базі даних підготовку подань, що відповідають вимірам і таблицям фактів сховища даних. Як ви будете вирішувати це завдання (Виберіть усі можливі варіанти.):

☐ застосуйте Import and Export Wizard і скопіюєте дані з подань вимірів і фактів в операційній базі даних у таблиці сховища даних, застосувавши для кожної непорожньої таблиці призначення параметра Drop and re-create the destination table (Видалити і заново створити таблицю призначення) у вікні Column Mappings (Порівняння стовпців);

☐ застосуйте Import and Export Wizard, налаштуйте його на копіювання даних із представлень вимірів і фактів операційної бази даних у таблиці сховища даних, збережіть пакет SSIS і потім відредагуєте його, використовуючи середовище SSDT і вставляючи відповідні засоби злиття даних для усіх таблиць призначення;

☐ застосуйте Import and Export Wizard і скопіюєте дані з представлень вимірів і фактів в операційній базі даних у таблиці сховища даних, використовуючи параметр Merge data into the destination table (Об'єднати дані в таблицю призначення) у вікні Column Mappings для кожної непустої таблиці призначення;

☐ застосуйте замість Import and Export Wizard середовище SSDT, тому що в майстра немає відповідних функціональних можливостей для перетворення та об'єднання даних.

F. Коли пакети SSIS зберігаються у файли DTSX, який формат використовується для збереження визначень пакета SSIS:

☐ вони зберігаються як двійкові файли;

☐ вони зберігаються як звичайні текстові файли;

☐ вони зберігаються як XML;

☐ вони зберігаються як особливі документи MS Word?

G. Які речення найкраще описують SQL Server Data Tools (SSDT) (Виберіть усі відповідні варіанти.):

☐ SSDT – це розширення середовища SQL Server Management Studio, яке може використовуватися для створення пакетів SSIS за допомогою спеціального майстра;

☐ SSDT – це спеціальна версія SQL Server Management Studio, розроблена для підвищення кваліфікації розробників, недостатньо знайомих з адмініструванням баз даних;

☐ SSDT – це спеціальна версія Visual Studio, що поставляється разом із SQL Server і пропонує великий набір інструментів для розроблення баз даних;

☐ SSDT – це служба SQL Server, яку можна застосовувати для виконання завдань технічного

обслуговування SQL Server, таких як переміщення даних і процеси, подібні до переміщення даних?

H. Які з тверджень про прості та складні переміщення даних є справедливими (Виберіть усі відповідні варіанти.):

☐ у простих переміщень даних єдине джерело даних і одне кінцеве місце призначення;

☐ у складних переміщеннях даних дані перед збереженням у місці призначення необхідно перетворити;

☐ у простих переміщеннях даних перетворення даних обмежені конвертуванням типів даних;

☐ у складних переміщеннях даних для об'єднання вихідних даних і даних у кінцевому місці призначення необхідна додаткове програмне оброблення?

I. Які з тверджень правильні (Виберіть усі відповідні варіанти.):

☐ пакет SSIS може містити одне або кілька рішень SSDT, кожне з яких виконує конкретну операцію керування даними;

☐ проєкт SSIS може містити один або кілька пакетів SSIS;

☐ проєкт SSIS може містити лише один пакет SSIS;

☐ пакети SSIS містять програмне опрацювання, що застосовується в переміщеннях даних і операціях перетворення даних?

J. Завдання Execute SQL Task дозволяє застосовувати до сховища даних інструкції SQL і команди. Які у вашому розпорядженні є інструменти під час розроблення пакетів SSIS для створення та тестування команди SQL (Виберіть усі відповідні варіанти.):

☐ середовище SQL Server Management Studio (SSMS);

☐ середовище SQL Server Data Tools (SSDT);

☐ редактор Execute T-SQL Statement Task Editor;

☐ SQL Server Enterprise Manager (SSEM)?

K. Ви повинні після завершення завдання Execute SQL Task виконати паралельно дві операції потоку даних. Як домогтися цього (Виберіть усе, що підходить.):

☐ в одному пакеті SSIS неможливо виконувати дві операції потоку даних паралельно;

☐ ви можете помістити обидва потоки даних в одне й те саме завдання потоку даних і створити елемент керування черговістю, що йде від завдання Execute SQL Task до завдання потоку даних;

☐ ви можете створити два окремих завдання потоку даних, а потім створити два елементи керування черговістю і спрямувати кожен із них від попереднього завдання Execute SQL Task до окремого завдання потоку даних;

☐ ви можете створити два окремі завдання потоку даних, помістити їх у третє завдання потоку даних і створити елемент керування черговістю, що йде від попереднього завдання Execute SQL Task до третього завдання потоку даних?

L. Який елемент керування черговістю можна застосувати, щоб дозволити завданню Б виконуватися після завдання А, навіть якщо завдання А завершилося з помилкою:

☐ елемент керування черговістю зі значенням failure (завершення з помилкою), що йде від завдання А до завдання Б.

☐ елемент керування черговістю зі значенням success (успішне завершення), що йде від завдання А до завдання Б.

☐ елемент керування черговістю зі значенням completion (завершення), що йде від завдання А до завдання Б.

☐ застосувати два елементи керування черговістю, що йдуть від завдання А до завдання Б – один зі значенням success, а інший зі значенням failure?

Аналітичні питання

Завдання 1. Копіювання виробничих даних до середовища розробки.

У межах виконання проєкту розробник розгорнув свій високотехнологічний пакет у виробництві (версія 1). Проєкт перейшов до другого етапу – розроблення нової версії. З метою збереження найбільш «свіжих» виробничих даних менеджер компанії поставив завдання розробити пакет для переміщення даних, який буде регулярно застосовуватися для копіювання

підмножини виробничих даних із робочого сховища даних у сховище даних у середовищі розробки.

Дайте відповіді на питання:

1.1. Який метод ви будете використовувати для передавання даних за вимогою?

1.2. Як зробити пакет максимально придатним для багаторазового використання?

Завдання 2. Параметризація диспетчерів з'єднань.

Пакети з обслуговування сховищ даних переросли в наявну в компанії інфраструктуру, тому повинні бути встановлені нові сервери і цього разу всі додатки сховищ даних будуть використовувати виділену мережу. На стадії 1 усі ваші пакети SSIS повинні бути заново розгорнуті на нових серверах, і системний адміністратор вирішив, що проєкти SSIS заслуговують більшої пропускної здатності, тому на стадії 2 вам буде дозволено виділити для ваших процесів переміщення даних більшу частину пропускної спроможності мережі.

Дайте відповіді на питання:

2.1. Скільки роботи з додаткового розроблення вам доведеться виконати на стадії 1?

2.2. Що необхідно реалізувати на стадії 2 для переналаштування всіх диспетчерів з'єднань на використання більших мережевих пакетів?

МАТЕРІАЛ ДЛЯ ПІДГОТОВКИ ДО ВИКОНАННЯ ПРАКТИЧНОЇ РОБОТИ 5

Теоретичний матеріал

Диспетчери з'єднань

Сервіси SSIS підтримують різні сховища даних (такі як файли, системи керування (реляційними) базами даних, бази даних служб SQL Server Analysis Services, Web-сервери, FTP-сервери, сервери електронної пошти, Web-служби, інструментарій Windows Management Instrumentation, черги повідомлень та об'єкти SQL Server Management Objects). У проєктах служб SSIS те саме сховище даних може з'явитися в кількох ролях – як джерело даних, як кінцеве місце призначення і як джерело посилальних даних. У завданнях потоку керування та компонентах потоку даних доступ до даних надається через спеціальні об'єкти SSIS, які називаються диспетчерами з'єднань (*connection managers*).

Для спрощення розроблення, налаштування та використання з'єднання їх властивості описуються незалежно від ролі, в якій фігурує сховище даних, лише один раз, а застосовуватися можуть стільки разів, скільки потрібно для різних ролей сховищ даних, для різних завдань та/або компонентів.

Залежно від типу сховища даних, а іноді від постачальника даних, що використовується для встановлення з'єднання, диспетчери з'єднань можуть застосовуватися для вилучення або модифікації даних до сховища даних (наприклад, для відправлення у сховище команд та запитів мовою DML (мова маніпулювання даними)), а також для виконання у сховищі даних команд опису та керування даними (наприклад, для відправлення у сховище даних команд мовами DDL (Data Definition Language, мова опису даних) або DCL (Data Control Language, мова керування даними)). Наприклад, можна створити за допомогою завдання Execute SQL Task (Виконання SQL) тимчасову таблицю, використовувати цю таблицю в завданні потоку даних і видалити її, коли вона більше не знадобиться.

Більшість типів диспетчерів з'єднань установлюється як частина екземпляра SQL Server. Додаткові диспетчери з'єднань можна знайти в Інтернеті, а якщо потрібно, ви можете розробити власний диспетчер з'єднань.

Область видимості диспетчерів з'єднань

Середовище *SQL Server Database Tools* (SSDT) підтримує два способи опису диспетчера з'єднань, які забезпечують два рівні доступності.

Диспетчери з'єднань, видимі в пакеті, доступні в контексті пакета служб SSIS, в якому вони були створені, і не можуть повторно використовувати інші пакети з одного і того самого проєкту SSIS.

Диспетчери з'єднань, видимі в усьому проєкті, доступні для всіх пакетів проєкту, в якому вони були створені.

Застосовуйте диспетчери з'єднань, видимі в пакеті, для з'єднань, які повинні бути доступні лише в конкретному пакеті, а диспетчери з'єднань, видимі в усьому проєкті, для з'єднань, які повинні використовуватися разом із кількома пакетами в проєкті.

Відповідно до рекомендованої методики застосування концепцій програмування середовища SSDT (Visual Studio) і приведення їх у відповідність до потреб реального життя диспетчери з'єднань проєкту дозволяють застосовувати один і той самий набір з'єднань в усьому операційному блоці, представленому декількома пакетами, якщо пакети згруповані в одному і тому самому проєкті служб SSIS.

Параметризація

Як уже обговорювалося в попередньому матеріалі, для полегшення розгортання та обслуговування пакетів SSIS плануйте параметризацію диспетчерів з'єднань, особливо тих властивостей, які можуть залежати від робочого середовища. Зазвичай необхідно параметризувати рядок з'єднання або його окремі елементи, які залежать від середовища, в якому буде розгортатися пакет, наприклад, ім'я екземпляра (ServerName) та ім'я бази даних (InitialCatalog). Параметризувати весь рядок

з’єднання або окремі його складові – швидше питання особистих переваг, важливо так чи інакше параметризувати. Насправді, параметризація повинна відповідати стандартам, прийнятим в організації. У цьому разі будь-який розробник в організації зможе розраховувати на те, що для вирішення загальної проблеми можна використати спільний метод.

Планування складного переміщення даних

Потік керування – важливий елемент пакета SSIS. Він визначає операції та зв’язки між ними, установлюючи порядок та умови їх виконання. Операції потоку керування подані завданнями потоку керування (або завданнями, для стислості), кожна з яких представляє єдину логічну операцію (незалежно від її реальної складності).

На відміну від простих переміщень даних, у яких дані переміщуються з джерела місце призначення «як є» (незмінними), у складних переміщеннях даних перед збереженням місця призначення дані перетворюються. Зазвичай це будь-який із перелічених далі видів перетворень або все одразу.

Очищення даних. Небажані або зіпсовані фрагменти даних видаляються або замінюються на коректні дані. Багато різних операцій відповідають цьому опису – все, від базового очищення (наприклад, обрізки рядків або заміни десяткових ком десятковими точками) до ретельно розробленого синтаксичного аналізу (наприклад, вилучення значних фрагментів даних за допомогою Regular Expressions (Регулярні вирази).

Нормалізація даних. Найпростішим визначенням нормалізації вважатимемо перетворення складних типів даних на прості типи даних (наприклад, вилучення окремих елементарних) значень із XML-документа або атомарних елементів із рядка з роздільниками).

Перетворення типів даних. У джерелі даних може бути інша система типів, ніж у місці призначення. Перетворення типів даних забезпечує перетворення лише на рівні типу окремих значень із типу даних джерела тип даних місця призначення.

Перетворення подання даних. У джерелі та місці призначення можуть використовуватися різні домени. Перетворення подання забезпечує на рівні домену заміщення окремих значень із домену-джерела еквівалентним значенням із домену місця призначення (наприклад, символ «F», що означає стать людини в джерелі даних, замінюється рядком «female», що позначає те саме в місці призначення).

Перевірка значень, що вводяться. Це контроль і доповнення бізнес-правил до окремих значень даних (наприклад, людина не може важити більше ніж одна тонна) кортежів (наприклад, точно дві різні людини утворюють сімейну пару) та/або множини (наприклад, лише одна людина може бути президентом країни в будь-який заданий час).

Обчислення даних та групування даних. У сховища даних зазвичай потрібно завантажувати не лише окремі значення, що представляють різні факти або міри, але також і обчислені (або заздалегідь скомпоновані) величини, одержані з вихідних значень (наприклад, джерело – чиста ціна і податок, а в місце призначення потрібно помістити ціну з урахуванням податку).

Зведення даних та скасування відомостей. Може знадобитися реструктурувати або реорганізувати вихідні дані для того, щоб вони відповідали моделі даних призначення (наприклад, дані у вигляді entry-attribute-value (EAV) потрібно реструктурувати в стовпці або навпаки).

Ще одна відмінна характеристика складних переміщень даних порівняно з простими – дозвіл зв'язків між новими або модифікованими вихідними даними та будь-якими даними, що вже є в місці призначення.

Ця конкретна вимога принципово важлива в організації сховищ даних не лише через додавання та модифікації, які повинні виконуватися у сховищах даних безперервно та коректно для надання надійної (безперебійної та достовірної) служби, але й тому, що всі ретроспективні дані організації зазвичай зберігаються та обслуговуються виключно у сховищі даних.

Складність переміщення даних залежить від набору перетворень, які необхідно застосувати до вихідних даних перед тим, як їх можна буде завантажувати в місце призначення, і від набору додаткових операцій, необхідних для коректного поєднання нових або модифікованих вихідних даних із даними у місці призначення. У міру зростання складності збільшуються і потреби в ресурсах і часі, необхідних для вирішення завдання. Як згадувалося раніше, операції з обслуговування сховищ даних зазвичай виконуються під час перерв на профілактичне обслуговування – ці періоди можуть бути тривалими (наприклад, уся ніч) або короткими (наприклад, кілька хвилин певну кількість разів упродовж дня). Під час планування переміщень даних необхідно завжди намагатися по можливості повністю використовувати для процесу обслуговування всі доступні ресурси, щоб час оброблення ніколи не перевищував часових рамок відповідного періоду, відведеного для обслуговування.

Знання робочого навантаження допоможе вам спроєктувати потік керування ваших пакетів SSIS так, щоб мінімізувати використання ресурсів (скажімо, збалансувати операції ЦПУ та введення / виведення для зменшення затримки) та час виконання. Наприклад, виконуючи трудомісткі операції ЦПУ з особливо ємними операціями введення / виведення (такими, як складні перетворення і підготовка до завантаження вимірів) паралельно з інтенсивними операціями введення / виведення, поєднаними з не особливо ємними операціями ЦПУ (наприклад, завантаження таблиць фактів, завантаження кешу уточнювального запиту (lookup cache) і оновлення великого обсягу), можна істотно знизити період бездіяльності як ЦПУ, так і системи введення / виведення.

Завдання

Оброблення служб SSIS можна визначити як систему операцій, що забезпечує автоматичне керування даними та/або сховищами даних та усуває на етапі виконання необхідність втручання людини, зберігаючи за ним це право лише під час розроблення та виявлення й усунення несправностей.

Принципове завдання служб SSIS можна описати як прагнення домогтися автоматизації максимально можливої кількості детермінованих одноманітних операцій, що повторюються, щоб їх виконували машини, дозволяючи людям зосередитися на тому, що вони роблять найкраще – на вирішенні реальних проблем, а не на виконанні детермінованих процедур; на творчому розумовому процесі, а не на багаторазовому механічному виконанні рутинних операцій.

У службах SSIS завдання людини – проєктувати (тобто визначати, як автоматизувати конкретні завдання), розробляти (тобто реалізовувати проєкт), розгортати (тобто виконувати пакети) і супроводжувати (тобто стежити за виконанням, вирішувати можливі проблеми і насамперед навчатися на прикладах і бути готовим до проєктування нових рішень), а виконання рішень автоматизовано.

Служби SSIS пропонують велику колекцію інструментів, які потрібні в операціях керування даними. Серед них є і прості, і дуже складні, але в усіх вони мають одну спільну властивість – кожен з інструментів представляє єдиний блок оброблення, що відповідає логічному набору дій, необхідний для вирішення реальних виробничих завдань.

Завдання SSIS можна розділити на кілька груп відповідно до концепцій, на яких вони базуються. У наступних розділах описані ці групи та завдання, що входять до них.

Завдання підготовки даних. Ці завдання застосовують для підготовки джерел даних для подальшого оброблення; підготовка може бути простою, як, наприклад, копіювання джерела на сервер, або складною, як профіль даних, визначення їх інформаційного значення або навіть з'ясування того, що вони є насправді.

Завдання робочого процесу. Ці завдання полегшують робочий процес, що є бізнес-процес, поданий у вигляді структури, яка включає зв'язки із середовищем і пов'язаними процесами; ці завдання автоматизують взаємодію між окремими процесами служб SSIS та/або взаємодію процесів SSIS із

зовнішніми процесами (процесами, що існують поза SQL Server).

Завдання переміщення даних. Ці завдання або беруть участь у переміщення даних, або забезпечують переміщення.

Завдання адміністрування SQL Server. Завдання адміністрування SQL Server також може бути автоматизовано з допомогою рішень SSIS. Таким чином, служби SSIS надають набір засобів, що підтримують типові завдання адміністрування. Усі ці завдання за звернення до вихідних та кінцевих екземплярів SQL Server покладаються на диспетчери з'єднань SMO (SQL Management Object). Більшість цих завдань вимагає, щоб користувачеві, що їх виконує, були надані дозволи високого рівня, необхідні виконання певних дій. (Наприклад, для передавання бази даних користувачеві потрібно бути членом зумовленої ролі сервера sysadmin, як на початковому екземплярі, так і на екземплярі сервера призначення.)

Завдання обслуговування SQL Server. Обслуговування SQL Server також можна автоматизувати за допомогою рішень SSIS, тому служби SSIS надають різні завдання обслуговування. Насправді плани обслуговування SQL Server реалізовувалися як пакети SSIS, починаючи з версії SQL Server 2005.

Завдання служб Analysis Services. Ці завдання створюють, змінюють, видаляють та обробляють об'єкти служб Analysis Services, а також виконують операції з вилучення даних. Усі ці завдання для з'єднання з базами даних SSAS використовують диспетчери з'єднань Analysis Services.

Завдання Script Task. Це спеціальне завдання розкриває програмну модель SSIS через її реалізацію серед .NET Framework для забезпечення розширюваності рішень SSIS. Завдання Script Task дозволяє об'єднати користувацькі операції керування даними із пакетами SSIS. Налаштування можна виконати, використовуючи будь-яку мову програмування, підтримувану середовищем Microsoft Visual Studio Tools for Applications (VSTA). Зазвичай завдання Script Task може застосовуватися для надання функцій, що не забезпечуються будь-якими стандартними вбудованими завданнями, та

об’єднання зовнішніх пакетів із пакетами служб SSIS або для надання доступу до зовнішніх пакетів або служб через їх прикладні програмні інтерфейси (API).

Для розроблення сценарію середовище VSTA надає інтегроване середовище розробки, що переважно є скороченою версією Visual Studio. Остаточний сценарій заздалегідь компілюється і потім вбудовується для визначення пакета SSIS. За умови, що програмний алгоритм розширення може бути інкапсульований повністю в єдиному сценарії (тобто без залежностей від зовнішніх бібліотек, які можуть бути доступні або недоступні на сервері розгортання) і поки що розширення не потребує багаторазового використання (тобто сценарій застосовується в єдиному пакеті SSIS або в досить обмеженій кількості пакетів), завдання Script Task – відповідний вибір.

Завдання користувача. Основна перевага завдання Script Task – його здатність розширити функціональні можливості служб SSIS без звичайних витрат на повний цикл розробки; процес розроблення простого сценарію можна вважати частиною циклу розробки пакета SSIS.

Але якщо багаторазове використання стає важливою складовою (наприклад, коли той самий програмний алгоритм необхідно реалізувати в множинних пакетах SSIS), розгортання та обслуговування проєктів SSIS, залежить від фрагмента коду, вбудованого у файл пакета SSIS, перестають бути очевидними.

Розгортання та обслуговування також ускладнюються, якщо бізнес-проблема не вкладається в бізнес-алгоритм всередині одного сценарію. У відповідь на подібні труднощі служби SSIS підтримують завдання користувача. Порівняно із завданнями Script Task для їх розроблення потрібні більші зусилля – їхній власний цикл розробки, але водночас вони значно підвищують можливість багаторазового використання, істотно знижують потенційні проблеми, пов’язані із залежностями, та значною мірою покращують практичні показники розгортання та обслуговування.

Завдання користувача можна розробляти незалежно від пакета SSIS. Це не лише підвищує ефективність поділу праці в

групі розробників, а й дозволяє поширювати завдання користувача незалежно від пакетів SSIS, у яких її передбачається використати.

Контейнери

Коли в службах SSIS реалізуються реальні виробничі ідеї, результуючі операції можуть бути скомпоновані з одного або кількох завдань. Для того щоб ці завдання, що формують логічно єдиний блок, могли поводитися як єдиний блок, у служби SSIS введені контейнери.

Контейнери (*containers*) забезпечують структуру (наприклад, завдання, що представляють той самий логічний блок, можна згрупувати в одному контейнері для поліпшення як легкозчитуваності, так і керованості), інкапсуляцію (наприклад, завдання, укладені в контейнер «Цикл за елементами» (loop container), будуть виконуватися багаторазово як єдиний блок) і область видимості (наприклад, ресурсів, обмеженим діапазоном дії контейнера, можуть отримати доступ завдання, поміщені в той самий контейнер, але не завдання, що знаходяться поза контейнером).

Одна з причин угрупування завдань – логіка процесу, інша – пошук та усунення помилок. У середовищі SSDT у режимі налагодження можуть виконуватися пакет SSIS цілком, окремі завдання і група завдань, укладена в контейнер.

У службах SSIS підтримуються три типи контейнерів.

1. *For Loop Container* (Цикл елементів). Цей контейнер виконує включені до нього завдання багаторазово, ґрунтуючись на виразі – цикл триває доти, доки результат обчислення виразу дорівнює True; він ґрунтується на тій самій концепції, що й цикл For у більшості мов програмування.

2. *Foreach Loop Container* (Цикл кожного елемента). Цей контейнер виконує включені до нього завдання багаторазово, кожного елемента обраного лічильника; він заснований на тій самій концепції ітерації, що й цикл Foreach у більшості сучасних мов програмування. Цей контейнер підтримує такі лічильники: лічильник ADO, лічильник за

набором рядків схеми ADO.NET, лічильник файлів, лічильник елементів, лічильник вузлів списку, лічильник SMO

3. Sequence Container (Послідовність). Цей контейнер не має програмного алгоритму, він забезпечує структуру для інкапсуляції завдань, що формують єдиний логічний блок, і для формування в службах SSIS області видимості змінних, які будуть доступні лише конкретному набору завдань, або області видимості транзакції для набору завдань.

Тестові питання

A. Що визначає потік керування диспетчера з'єднання:

☐ операції;

☐ послідовність операцій;

☐ взаємозв'язок операцій;

☐ параметри з'єднання?

B. Оберіть можливі функціональні призначення диспетчера з'єднання:

☐ вилучення даних;

☐ модифікація даних;

☐ опис даних;

☐ керування даними;

☐ створення об'єктів сховища даних.

C. Чи залежить опис властивостей з'єднання від ролі, для якої він використовується:

☐ ні;

☐ так?

D. Призначення контейнерів полягає в:

☐ організації сукупного зберігання семантично аналогічних даних;

☐ організації сукупного зберігання логічно споріднених операцій (завдань);

☐ організації доступу до даних.

E. Вам потрібно витягти дані з текстових файлів із роздільниками. Диспетчер з'єднань якого типу ви оберете:

☐ диспетчер з'єднань Flat File;

☐ диспетчер з'єднань OLE DB;

☐ диспетчер з'єднань ADO.NET;

☐ диспетчер з’єднань File?

F. Деякі дані, з тих, які обробляє ваша компанія, надсилаються партнерами електронною поштою. Як ви налаштуєте диспетчер з’єднань SMTP, щоб витягувати файли з повідомлень електронної пошти:

☐ у диспетчері з’єднань SMTP задати властивості OperationMode значення Send and Receive;

☐ у цій ситуації неможливо використовувати диспетчер з’єднань SMTP, тому що в службах SSIS його можна застосовувати лише для надсилання повідомлень електронної пошти:

☐ диспетчер з’єднань SMTP за замовчуванням підтримує надсилання й одержання повідомлень електронної пошти, тому жодного додаткового налаштування не потрібно;

☐ для цієї мети неможливо використовувати диспетчер з’єднань SMTP, потрібно замість нього застосувати диспетчер з’єднань IMAP (Internet Message Access Protocol)?

G. Вам потрібно витягти дані з таблиці бази даних SQL Server. Диспетчери з’єднань яких типів ви можете застосувати (Виберіть усі відповідні варіанти.):

☐ диспетчер з’єднань ODBC;

☐ диспетчер з’єднань OLE DB;

☐ диспетчер з’єднань File;

☐ диспетчер з’єднань ADO.NET?

H. У вашому рішенні SSIS необхідно в найкоротший термін завантажити великий набір рядків у базу даних. Рядки зберігаються в текстовому файлі з роздільниками, і лише один стовпчик джерела потребує перетворення типу даних із типу String (використаного у вихідному стовпчику) на тип Decimal (що використовується в стовпці призначення). Яке завдання потоку керування більше за інших підходить для цієї операції:

☐ у цьому разі відмінно підійде завдання File System Task, оскільки воно вміє читати дані з файлів і його можна налаштувати для виконання перетворень типів даних;

☐ найбільш підходяще – завдання Bulk Insert Task, тому що воно найшвидше і може виконувати перетворення типів даних;

☐ необхідно застосувати завдання потоку даних, оскільки перед завантаженням у таблицю дані потребують перетворення;

☐ немає такого завдання керування потоком, яке могло б самостійно впоратися з цією операцією, оскільки дані потрібно витягти з файлу-джерела, перетворити і завантажити в таблицю призначення. Доведеться використовувати щонайменше три завдання: завдання Bulk Insert Task для завантаження даних у проміжну базу даних, завдання Data Conversion Task для відповідного перетворення даних і на завершення завдання Execute SQL Task для об'єднання перетворених даних із даними, наявними в місці призначення?

I. Частина вашого процесу консолідації даних включає в собі витяг даних із робочих книг Excel. Іноді в даних містяться помилки, які неможливо виправити автоматично. Як ви можете вирішити цю проблему засобами служб SSIS:

☐ перенаправити зіпсований потік даних у завдання External Process Task, відкрити проблемний файл робочої книги в Excel і повідомити користувача про необхідність коригування файлу перед продовженням процесу консолідації даних;

☐ перенаправляти зіпсований потік даних у завдання File System Task, яка перемістить файл із помилками у виділене місце зберігання, де співробітник інформаційного відділу пізніше зможе його відкоригувати;

☐ якщо помилка не може бути виправлена автоматично, у служб SSIS немає способу продовжити процес автоматичної консолідації даних;

☐ жоден із наведених варіантів відповіді не правильний. Відповідно до суворих правил перевірки правильності даних програми Excel файл із робочою книгою Excel не може містити помилкові дані?

J. У вашому ETL-процесі необхідно під час виконання витягти кілька значень із бази даних, ґрунтуючись на іншому

значенні, яке буде доступне під час виконання, і ці значення не можуть бути витягнуті як частина якого-небудь потоку даних. Яке завдання можна використовувати в цьому разі:

- ☐ завдання Execute T-SQL Statement Task;
- ☐ завдання Execute SQL Task;
- ☐ завдання Expression Task;
- ☐ завдання Execute Process Task?

K. Як у пакеті служб SSIS визначається порядок виконання або послідовність операцій:

- ☐ обробник середовища виконання служб SSIS визначає порядок виконання автоматично, ґрунтуючись на типі операцій, доступних програмних і апаратних ресурсах і обсязі даних;
- ☐ послідовність визначається за допомогою елементів керування черговістю;
- ☐ послідовність визначається за допомогою контейнера Sequence;
- ☐ послідовність визначається під час розроблення за допомогою елементів керування черговістю та контейнерів Sequence, але під час виконання обробник служб SSIS виконує операції в порядку, встановленому найбільш підходящим планом виконання для досягнення максимальної продуктивності?

L. Як елемент керування черговістю зі значенням failure (збій) впливає на порядок виконання:

- ☐ наступне завдання буде виконуватися, якщо попереднє завдання завершилося з помилкою;
- ☐ наступне завдання буде виконуватися, якщо попереднє завдання завершилося без помилок;
- ☐ наступне завдання не буде виконуватися ніколи, тому що цей елемент керування застосовується лише на стадії розроблення;
- ☐ наступне завдання буде виконуватися незалежно від того, що попереднє завдання завершилося з помилкою, але помилка буде записана в журнал служб SSIS?

М. У вашому ETL-процесі є три зовнішні процеси, які повинні виконуватися послідовно, але ви не хочете зупиняти виконання, якщо один із них дав збій. Чи можна цього домогтися за допомогою елементів керування черговістю? Якщо так, то які елементи керування черговістю необхідно застосувати:

☐ ні, цього не можна домогтися за допомогою лише елементів керування черговістю;

☐ так, цього можна домогтися, застосувавши елементи керування черговістю зі значенням completion (завершено) між першим і другим та між другим і третім завданнями Execute Process Task і елемент керування черговістю зі значенням success (успішно) між третім завданням Execute Process Task і наступним завданням;

☐ так, цього можна домогтися, застосувавши елементи керування черговістю зі значенням completion (завершено) між першим і другим, між другим і третім завданнями Execute Process Task і між третім завданнями Execute Process Task і наступним завданням;

☐ так, цього можна домогтися, застосувавши елементи керування черговістю зі значенням failure (збій) між першим і другим та між другим і третім завданнями Execute Process Task і елемент керування черговістю зі значенням completion (завершено) між третім завданням Execute Process Task і наступним завданням?

Аналітичні питання

Завдання 1. Розроблення процесу очищення.

У вашій системі керування даними є два сховища даних (не враховуючи операційного сховища даних); головне сховище даних містить усі дані включно із ретроспективними. Допоміжні сховища даних використовуються вебдодатками, надаючи клієнтам доступ і відповідно повинні містити виключно поточні дані. Ваш пакет для сховища даних уже працює, переміщуючи дані до обох сховищ.

Поставлене завдання створити додатковий процес, який визначає, які дані перестали бути поточними або актуальними і відповідно повинні бути видалені з допоміжного сховища.

Дайте відповіді на питання:

1.1. Як ви визначите, які рядки необхідно видалити?

1.2. Який метод видалення ви застосуєте для досягнення максимальної ефективності?

Завдання 2. Інтеграція зовнішніх процесів.

У свої сценаріях керування даними ваша компанія приймає суміш із особистих пакетів, розроблених вашою групою та іншими організаціями, готових пакетів, які для вас, досвідченого розробника, є просто «чорними ящиками» − ви можете лише спиратися на те, що ці пакети працюють як потрібно, без найменшої здогадки про те, як це робиться насправді.

У вашому пакеті для збереження даних необхідно консолідувати дані із вашого особистого пакета та з двох різнотипних «чорних ящиків», у одного з яких є спеціальний інструмент вилучення даних (автономний додаток), що отримує дані із внутрішнього сховища і зберігає їх у файлах файлової системи, а другий надає прикладний інтерфейс, що забезпечує доступ до своїх внутрішніх засобів одержання даних.

Дайте відповіді на питання:

2.1. Які функціональні засоби пропонують служби SSIS для інтеграції таких різнотипних пакетів у єдиний процес SSIS?

2.2. Як ви будете застосовувати платформу SQL Server для вирішення цієї проблеми?

МАТЕРІАЛ ДЛЯ ПІДГОТОВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ 6

Теоретичні матеріали

Джерела даних та місця призначення. Створення потоку даних

Завдання потоку даних – одна з найважливіших та найскладніших завдань потоку керування у службах SQL Server Integration Services (SISS). У неї включена підсистема оброблення потоку даних (data flow engine), яка витягує, перетворює та завантажує дані із джерел даних у місця призначення даних. Підсистема оброблення потоку даних використовує архітектуру із застосуванням буферів в оперативній пам'яті для ефективного керування різними видами наборів даних. Основна одиниця оброблення для всіх компонентів завдання потоку даних – рядок. Рядки групуються в буфери (buffers), а буфери використовуються для переміщення рядків конвеєром (pipeline) даних. Термін «конвеєр» з'явився тому, що дані вливаються в завдання потоку даних, рухаються через нього і потім випливають із нього.

Під час розроблення процесу ETL у будь-якому пакеті може бути кілька завдань потоку даних, одне чи жодного. Для вставлення завдання потоку даних у пакет необхідно перетягнути мишкою завдання потоку даних із панелі SSIS Toolbox (Панель елементів служб SSIS) у потік керування, або двічі клацнути на завданні кнопкою мишки, щоб вставити його в область потоку керування. Перебуваючи в конструкторі потоку даних (вкладку Data Flow), можна переглянути різні завдання потоку даних, вибираючи їх у списку Data Flow Task (див. рис. 6.1).

На панелі SSIS Toolbox є три типи компонентів завдання потоку даних:

- адаптери джерела потоку даних;
- перетворення потоку даних;
- адаптери призначення потоку даних.

Адаптери потоку даних (data flow adapters) надають можливість витягувати дані з джерел даних та завантажувати дані до джерел.

Перетворення потоку даних (data flow transformations) використовують дані, надані адаптерами потоку даних для застосування до них під час передавання, наприклад, у сховищі даних, широкого набору можливих модифікацій, починаючи з простих взаємно-однозначних зіставлень і закінчуючи складними бізнес-алгоритмами.

У SQL Server Data Tools (SSDT) компоненти потоку даних на панелі SSIS Toolbox за замовчуванням згруповані в п’ять категорій:

– Favorites (Вибрані);
– Common (Загальні);
– Other Transformations (Інші перетворення);
– Other Sources (Інші джерела);
– Other Destinations (Інші призначення).

Будь-який компонент панелі SSIS Toolbox можна додати на вкладку Data Flow у конструкторі служб SSIS, перетягнувши його мишкою або двічі клацнувши кнопкою мишки.

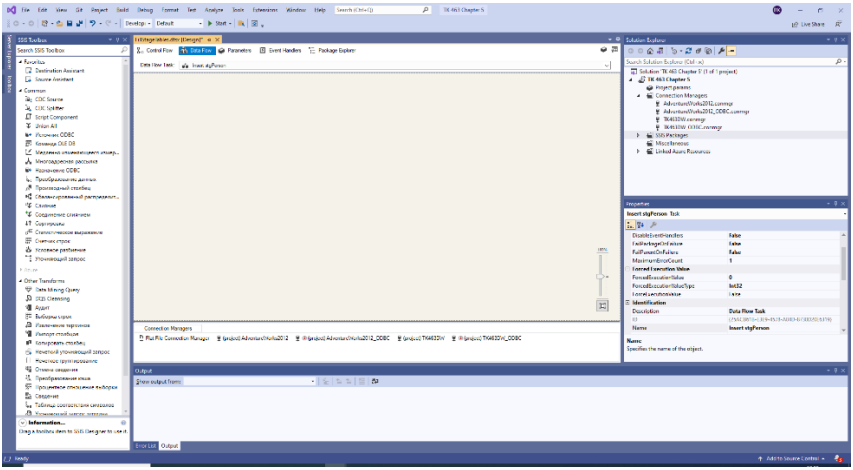

Рисунок 6.1 – Ілюстрація вкладки Data Flow з відкритою панеллю SSIS Toolbox

Визначення та налаштування адаптерів джерел потоку даних

Адаптер джерела потоку даних одержує дані з джерела і робить їх доступними іншим компонентам у потоці даних. Адаптери джерела потоку даних використовують з'єднання служб Integration Services, які можуть визначатися на рівні пакета або на рівні проєкту і вказують джерела даних на конкретні екземпляри сервера або розміщення файлів. Усі активні з'єднання перелічені у вікні Connection Managers (Диспетчери з'єднань). До винятків належать адаптер Raw File (Необроблений файл) та адаптер XML, які не використовують з'єднання пакета або проєкту. У таблиці 6.1 наведено джерела потоку даних та їх призначення.

У більшості адаптерів джерела даних подібні параметри налаштування. Наприклад, на рисунку 6.2 показаний розділ Connection Manager (Диспетчер з'єднань) діалогового вікна OLE DB Source Editor (Редактор джерела «OLE DB»), призначеного для читання даних джерела з таблиці Person.Person, що належить базі даних AdventureWorks2012, та переміщення даних надалі до бази даних TK463DW.

У цьому прикладі режим доступу до даних заданий як Table or view (Таблиця або подання) та вибрана таблиця джерела. Якщо встановити режим доступу до даних SQL Command (Команда SQL), ви зможете записати власну SQL-інструкцію SELECT.

Кожен ряд або кожен рядок (залежно від джерела) всередині адаптера джерела буде перетворюватися на стовпці SSIS. Ви можете встановити стовпці, які будуть використовуватися в потоці даних, вибравши їх на сторінці Columns (Стовпці) у редакторі будь-якого джерела.

Властивості для будь-якого адаптера джерела даних можна встановити, вибравши об'єкт і переглядаючи вікно Properties (Властивості). Наприклад, установити кількість секунд до блокування команди через перевищення відведеного часу можна за допомогою властивості CommandTimeout.

Таблиця 6.1 – Джерела потоку даних та їх призначення

Джерело потоку даних	Призначення
ADO.NET source	Це джерело надає з'єднання з таблицями та запитами через постачальника ADO.NET
CDC source	Change Data Capture (CDC, відстеження змінених даних) – компонент, що дозволяє вилучати лише змінені дані (з операцій insert, update або delete) у вихідній системі. Він використовує постачальника ADO.NET для підключення до таблиці, включеної підсистемою CDC
Excel source	Це джерело дозволяє вилучати дані з робочого листа Microsoft Excel, заданого у файлі Excel
Flat File source	Це джерело дозволяє вилучати дані з файлів із роздільниками або файлів фіксованої ширини, створених за допомогою різних кодових сторінок. Він використовує диспетчер з'єднань Flat File
ODBC source	Це джерело підключається до заданого джерела ODBC за допомогою чистого ODBC, а не постачальника OdbcDataProvider з ADO.NET
OLE DB source	Це джерело приєднується до встановлених постачальників OLE DB, таким як SQL Server, SQL Server Analysis Services (SSAS) та Oracle
Raw File source	Джерело Raw File читає дані з власного файлу даних служб SSIS, записаних призначенням Raw File. Оскільки подання даних «рідне» для джерела, даним не потрібно трансляції та майже жодного синтаксичного аналізу. Це означає, що джерело Raw File може читати дані швидше за інші джерела
XML source	Джерело XML дозволяє вилучати з XML-файлу необроблені дані. Йому потрібна XML-схема для визначення зв'язків даних

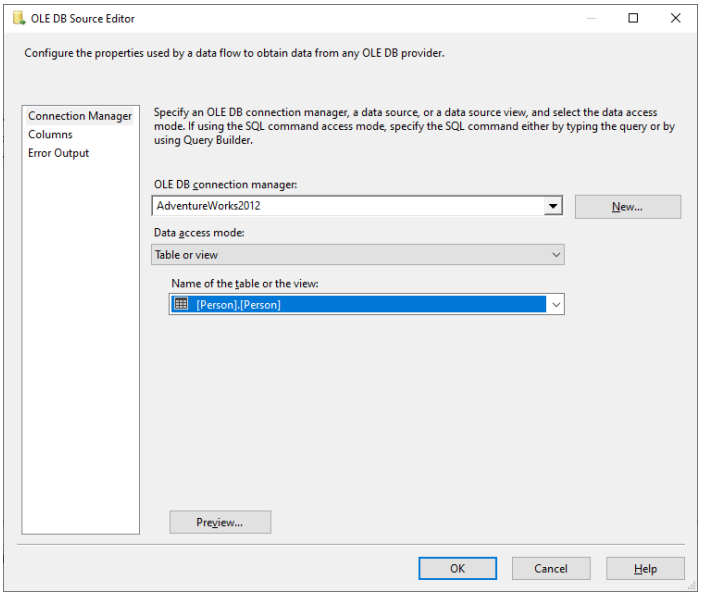

Рисунок 6.2 – Сторінка Connection Manager у вікні OLE DB Source Editor

Визначення та налаштування адаптерів призначення потоку даних

Місця призначення (або призначення) потоку даних аналогічні джерелам даних, оскільки вони також використовують з'єднання проєкту або пакета. Але призначення – це кінцеві точки в завданні потоку даних, що визначають місце, в яке дані необхідно передати. Якщо є потреба записати дані в неструктурований файл або конкретну таблицю в екземплярі SQL Server, необхідно вибрати відповідний адаптер призначення. На панелі SSIS Toolbox можна знайти декілька додаткових призначень, які не мають відповідних їм адаптерів джерел даних і навпаки (наприклад, адаптер CDC-джерела не має відповідного призначення). Ви можете скористатися новим компонентом Destination Assistant (Помічник зі створення призначення або призначенням) або безпосередньо вибрати потрібний адаптер призначення.

Місця призначення потоку даних:

ADO.NET Destination (Призначення ADO.NET). Застосовується для вставлення даних за допомогою постачальника ADO.NET.

Data Mining Model Training (Навчання моделі інтелектуального аналізу даних). Дозволяє передавати дані з потоку даних до моделі інтелектуального аналізу служб SSAS.

DataReader Destination (Призначення DataReader). Дозволяє передавати дані до набору даних ADO.NET, на який можна посилатися програмно.

Dimension Processing (Оброблення вимірів). Завантажує та обробляє вимірів служб SQL Server Analysis Services.

Excel Destination (Призначення Excel). Застосовується для запису даних на конкретний аркуш в Excel.

Flat File Destination (Призначення «Неструктурований файл»). Дозволяє вставляти дані в неструктурований файл, наприклад, із комами або табуляціями як роздільниками.

ODBC Destination (Призначення ODBC). Дозволяє вставляти дані за допомогою постачальника ODBC. Він підтримує режим послідовного пакетного або рядкового доступу до даних при вставці. У середовищі сховища даних через великих обсягів даних рекомендується пакетний режим.

OLE DB Destination (Призначення «OLE DB»). Застосовує постачальника OLE DB для вставлення рядків у систему призначення, що допускає з'єднання OLE DB.

Partition Processing (Оброблення секцій). Дозволяє обробляти секцію служб SSAS безпосередньо з даних, що проходять через потік даних.

Raw File Destination (Призначення «Необроблений файл»). Зберігає дані у своєму форматі служб SSIS як двійковий файл. Дуже корисно в сценаріях, які потребують у тимчасовому розміщенні даних, наприклад, якщо сервер призначення недоступний і ви не бажаєте або не можете обробляти дані джерела повторно.

Recordset Destination (Призначення «Набір записів»). Приймає дані потоку даних та створює набір записів у змінній

пакета типу Object. Дані можуть використовуватися поза потоком даних іншими об'єктами потоку керування.

SQL Server Compact Destination (Призначення «SQL Server Compact»). Дозволяє надсилати дані на мобільний пристрій із запущеною версією SQL Mobile.

SQL Server Destination (Призначення «SQL Server»). Надає високошвидкісне призначення для локальної бази даних SQL Server. Пакет служб SSIS повинен запускатися на тому самому сервері, що й база даних SQL Server, що використовується як місце призначення.

Перед налаштуванням адаптера призначення даних завдання потоку даних потрібно створити як мінімум одне місце призначення. На рисунку 6.3 показаний простий потік даних з одним джерелом та одним місцем призначення. Потік даних одержує записи з таблиці Person.Person у базі даних AdventureWorks2012 і вставляє їх у таблицю stg.Person у базі даних TK463DW.

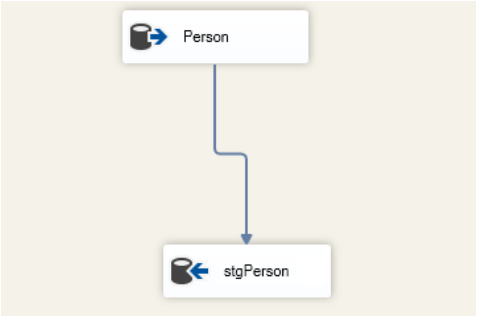

Рисунок 6.3 – Простий потік даних, що витягує дані з джерела адаптера і записує їх в адаптер призначення

Між адаптером джерела та адаптером призначення є шлях потоку даних (data flow path). Він з'єднує вихід одного компонента із входом іншого компонента. Шляхи задають послідовність всіх компонентів усередині потоку даних (адаптери джерел, перетворення, адаптери призначень), дозволяють вставляти інструкції в потік даних, дозволяти посилання між стовпцями або переглянути джерело стовпця, як ви побачите пізніше у цьому розділі.

Як і адаптер джерела, адаптеру призначення потрібне налаштування. Спочатку необхідно з'єднати шлях потоку даних з адаптером призначення, щоб мати можливість задати потрібні зіставлення стовпців. Як показано на рисунку 6.4, для призначення OLE DB необхідно задати диспетчер з'єднань і таблицю призначення.

Урахуйте, що за замовчуванням призначення OLE DB встановлює у списку, що розкривається, варіант режиму доступу до даних Table or view – fast load (Швидке завантаження таблиці або подання). Це означає, що рядки будуть оброблятися інструкцією масової вставки, а не один за одним. За замовчуванням також установлено прапорець Table lock (Блокування таблиці), тобто під час запису даних таблиці встановлюється параметр TABLOCK. Це усуває витрати на збільшення блокування, але якщо одночасно пишеться з кількох потоків даних в одну й ту саму таблицю, необхідно відключити цей параметр, інакше під час виконання виникне помилка.

На рисунку 6.5 показано сторінку Mappings (Співставлення) того самого редактора. На ній можна порівняти стовпці, доступні потоку даних, зі стовпцями призначення в адаптері призначення. Сторінка Mappings є в будь-якого адаптера призначення.

Якщо необхідно залишити без порівняння один зі стовпців, в області Input Column (Вхідний стовпець), на сторінці Mappings, можна встановити значення джерела варіант <ignore> (Пропустити). Те саме стосується і стовпців призначення.

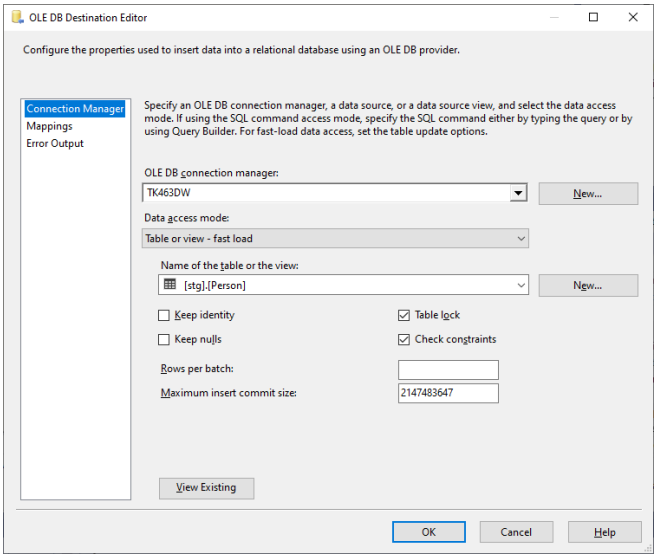

Рисунок 6.4 – Сторінка Connection Manager у вікні OLE DB Destination Editor

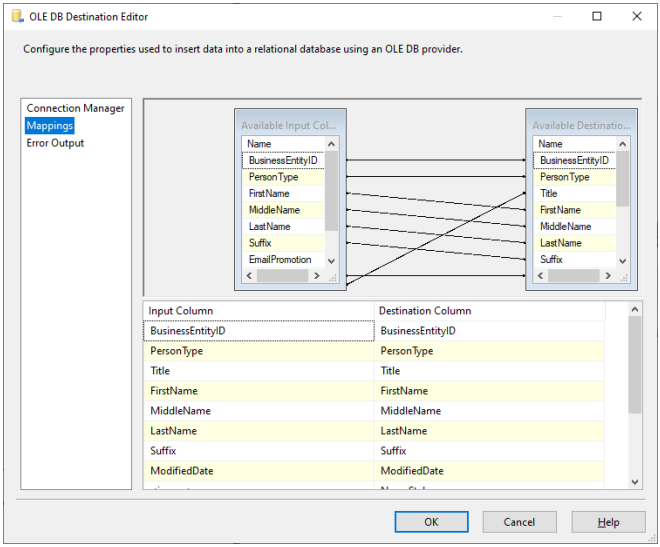

Рисунок 6.5 – Порівняння стовпців для адаптера призначення у вікні OLE DB Destination Editor

Типи даних служб SSIS

У разі використання адаптера джерела даних типи даних джерела порівнюються із загальними типами служб SSIS. Це означає, що після одержання даних адаптером джерела всі операції в потоці даних виконуються на типах даних SSIS. У проєкті може бути декілька баз даних, файлів тощо із специфічними типами даних; підсистема оброблення потоку даних перетворюватиме кожен тип даних на відповідний тип даних SSIS. Наприклад, числовим даним присвоюється числовий тип даних, рядкам – символьний тип даних, датам – тип даних для дат. Іншим даним, таким як ідентифікатори GUID і великі двійкові об'єкти Binary Large Object Blocks (BLOBs), також надаються відповідні типи даних. Якщо в даних наявний тип, який не перетворюється на тип даних служб Integration Services, виникає помилка.

Це означає, що під час розроблення потоку даних відомо, який обсяг пам'яті йому знадобиться для кожного рядка з конкретного адаптера джерела даних. Для доступу до налаштувань порівняння даних із типами даних служб SSIS необхідно перейти до розширеного редактора Show Advanced Editor.

На рисунку 6.6 показано вкладку Input and Output Properties (Властивості входів та виходів) розширеного редактора для адаптера джерела даних. Можна розкрити на вкладці Вузол Source Output (Вихід джерела) та в категорії Output Columns (Вихідні стовпці) переглянути або змінити будь-який тип даних служб SSIS в області Data Type Properties (Властивості типу даних). Наприклад, стовпець FirstName порівняний із типом даних SSIS Unicode string [DT_WSTR] довжиною 50 символів.

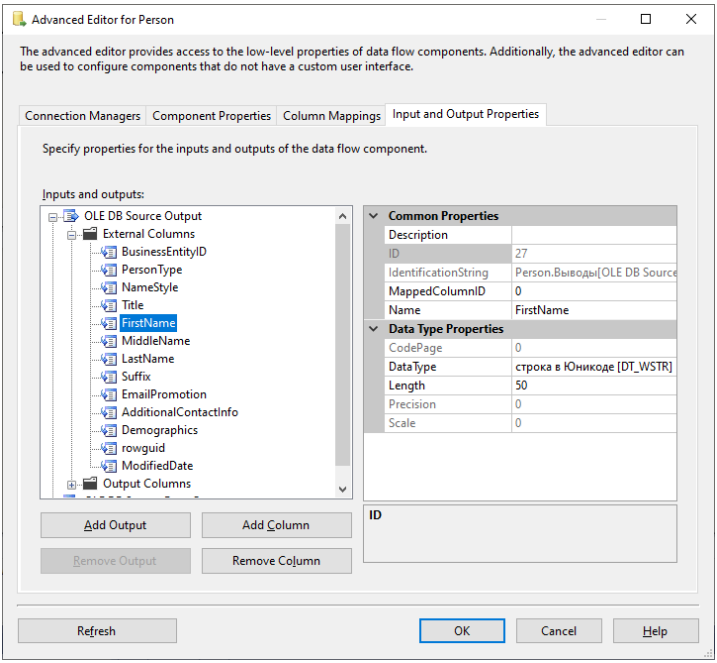

Рисунок 6.6 – Вкладка Input and Output Properties у розширеному редакторі

Використання швидкого синтаксичного аналізу

Швидкий синтаксичний аналіз (Fast Parse) – це набір операцій у службах SSIS, який можна використовувати для швидкого завантаження даних із неструктурованого файлу. Під час завантаження даних за допомогою адаптера джерела у вигляді неструктурованого файлу без аналізу даних, що залежать від мови та регіональних стандартів (локалі) (такі як формати дат, десяткові символи, наприклад кома, або символи валют), служби SSIS можуть застосовувати для дуже швидко завантаження даних параметр Fast Parse.

Швидкий синтаксичний аналіз підтримує лише підмножина форматів дат, часу та цілих форматів.

Для встановлення властивості Fast Parse використовується розширений редактор:

1. У діалоговому вікні Advanced Editor (Розширений редактор) треба перейти на вкладку Input and Output Properties (Властивості входів та виходів).

2. На панелі Inputs and outputs (Входи та виходи) клацніть кнопкою миші на стовпці, для якого хочете встановити властивість Fast Parse.

3. У вікні властивостей розкрийте вузол Custom Properties (Властивості користувача) і потім встановіть для властивості FastParse значення True.

Ця властивість доступна на рівні стовпця, тому що одні стовпці можуть залежати від локалі, інші – ні.

Робота з перетвореннями потоку даних

Вибір перетворень. Перетворення дозволяють змінювати та обробляти дані в потоці даних. Важливо зрозуміти, що робить кожне з перетворень і як воно впливає на потік даних загалом із погляду вимог до оброблення даних та продуктивності.

Перетворення може оперувати одним рядком даних у конкретний час або одночасно кількома рядками даних. Операції в деяких перетвореннях подібні одна на одну; отже, перетворення можна розділити на природні групи з подібними компонентами. Крім такого угруповання, кожному перетворенню відповідає вид блокування, що виникає в завданні потоку даних.

Є три види блокування:

– У перетвореннях без блокування (non-blocking transformations) кожен рядок проходить через перетворення без будь-якого очікування. Для позначення далі використовується буква N.

– У перетвореннях із частковим (partial-blocking) блокуванням накопичується значна кількість рядків, і потім усі вони зазнають перетворення. Для позначення далі використовується буква P.

– У перетвореннях із блокуванням (blocking) усі рядки повинні бути раховані в перетворення, як воно зможе їх обробити. Для позначення далі використовується літера B.

Перетворення рівня рядка. Найпоширеніші перетворення, що легко налаштовуються, – перетворення рівня рядка, що виконують операції на рівні окремого рядка і не потребують інших рядків із джерела. У таблиці 6.2 описано перетворення на рівні рядка.

Деякі поширені випадки цього обчислення в сценаріях сховищ даних включають створення обчислюваних стовпців із кількох вихідних стовпців, математичні обчислення, перетворення типів даних конкретних значень та заміну значення NULL іншими значеннями. З погляду продуктивності та зростання робочого навантаження перетворення Import Column (Імпорт стовпця) та Export Column (Експорт стовпця) відрізняються від інших перетворень рівня рядка. Обидва вони дозволяють читати та записувати конкретний стовпець як дані двійкового типу. Наприклад, за допомогою перетворення Import Column можна вставити в потік даних зображення, що зберігаються в окремих файлах.

Таблиця 6.2 – Види перетворень на рівні рядка

Перетворення потоку даних	**Призначення**	**Тип блокування**
1	2	3
Audit (Аудит)	Вставляє до кожного рядка додаткові стовпці, ґрунтуючись на системних змінних пакетах, таких як ExecutionStartTime та PackageNam	N
Cache Transform (Перетворення кешу)	Дозволяє за допомогою диспетчера з'єднань із кешем записувати дані в кеш. Пізніше дані можуть використовуватися перетворенням Lookup (Уточнювальний запит). Це корисно під час застосування кількох перетворень Lookup до тих самих даних, оскільки служби SSIS збережуть у кеші	N

Продовження таблиці 6.2

1	2	3
	потрібні дані лише один раз, а не для кожного компонента Lookup	
Character Map (Таблиця символів)	Виконує звичайні операції з текстом, наприклад заміну малих літер великими, і дозволяє виконувати розширені операції побітового перетворення	N
Copy Column (Копіювання стовпця)	Дублює значення стовпця в кожному рядку в новий іменований стовпець	
Data Conversion (Конвертація даних)	Створює в кожному рядку новий стовпець із даними нового типу, конвертованими з вихідного стовпця. Прикладом може бути конвертація тексту в числові дані або тексту в текст Unicode	
Derived Column (Похідний стовпець)	Для кожного рядка створює або замінює стовпець на основі заданого виразу SSIS. Це найчастіше застосовуване перетворення рівня рядка, тому що воно дозволяє замінювати значення в стовпцях або створювати нові стовпці на основі наявних стовпців, змінних та параметрів	
Export Column (Експорт стовпця)	Експортує у файл стовпці типу binary large objects (BLOB) по черзі з кожного рядка	N
Import Column (Імпорт стовпця)	Завантажує двійкові файли, наприклад, зображення конвеєр (pipeline), призначений для призначення даних типу BLOB	N
Row Count (Підрахунок рядків)	Підраховує кількість рядків, що пройшли через перетворення, та зберігає кількість у змінній пакета після оброблення останнього рядка	N

Перетворення з кількома входами та виходами. Перетворення з кількома входами та виходами можуть працювати з кількома входами даних та генерувати кілька виходів даних відповідно. Такі перетворення дозволяють комбінувати різні відгалуження шляхів потоків даних на один шлях або створювати кілька відгалужень шляхів потоків даних з одного шляху. У таблиці 6.3 перелічені перетворення з кількома входами та виходами.

У сценаріях сховищ даних ви дуже часто використовуватимете ці перетворення. Найпоширеніший сценарій – застосування компонента Lookup (Уточнювальний запит) у режимі Full Cache (Повне кешування) під час вставлення даних у таблицю фактів. У цих випадках для одержання відповідного сурогатного ключа з таблиці вимірів (за винятком великих таблиць вимірів із кількістю рядків більше ніж 10 млн; тут більше підійде перетворення Merge Join) дуже підходить перетворення Lookup.

Таблиця 6.3 – Види перетворень із кількома входами та виходами

Перетворення потоку даних	**Призначення**	**Тип блокування**
1	2	3
CDC Splitter (Розділювач CDC)	Розділяє єдиний потік змінених рядків із компоненти Source CDC на множинні потоки даних, ґрунтуючись на типі змін вихідних даних (тобто це операція вставки, оновлення чи видалення). CDC Splitter направляє дані у три можливі виходи, аналізуючи стовпець __$operation. Це перетворення – особливий варіант перетворення Conditional Split, яке автоматично обробляє стандартні значення стовпця __$operation	N

Продовження таблиці 6.3

1	2	3
Conditional Split (Умовне розбиття)	Спрямовує або відфільтровує дані на основі умовного виразу в один або кілька виходів, із яких кожен рядок може бути відправлений лише в один вихідний шлях	N
Lookup (Уточнювальний запит)	Виконує операцію порівняння між поточним рядком та зовнішнім набором даних по одному або декількох стовпцях. Додаткові стовпці можуть бути додані в потік даних із зовнішнього набору даних	N
Merge (Злиття)	Зливає рядки двох аналогічних відсортованих вхідних наборів даних, один за одним, на основі заданого ключа сортування	P
Merge Join (З'єднання злиттям)	Поєднує рядки двох відсортованих вхідних наборів даних, ґрунтуючись на заданих стовпцях або стовпцях з'єднання, додаючи стовпці з кожного джерела	P
Multicast (Багатоадресна доставка)	Генерує один або кілька ідентичних вихідних наборів даних, причому кожен рядок відправляється на кожен вихід. Це перетворення створює логічну копію даних	N
Union All (Об'єднати все)	Об'єднує один або кілька аналогічних вхідних наборів, розміщуючи рядки один за одним на основі значень у стовпчиках порівняння. Кількість рядків у вихідному наборі перетворення Union All виходить підсумовуванням лічильників рядків усіх вхідних наборів даних	P

Як описано в таблиці 6.3, перетворенням Merge і Merge Join потрібні відсортовані вхідні набори даних. Обидва перетворення з частковим блокуванням, тобто рядки не можуть

негайно вирушати у вихідний шлях. Це пояснюється тим, що перетворення чекає на рядки з усіх входів, зберігаючи відсортованим вихідний набір або звіряючи відсортовані рядки на базі певного порядку сортування.

Перетворення набору рядків. Перетворення набору рядків виконують оброблення на підставі критеріїв кількох вхідних рядків або генерують множинні вихідні рядки з одного вхідного рядка. Перетворення набору рядків вимагають більше операційних витрат і споживають більше оперативної пам'яті, але дуже важливі задоволення бізнес-потреб. У таблиці 6.4 перелічені перетворення набору рядків.

Найчастіше використовувані перетворення набору рядків – Aggregate (Статистичне оброблення) і Sort (Сортування). Перетворення Aggregate застосовується у середовищах сховищ даних для заповнення кіосків даних із високим ступенем гранулярності. Припустимо, у сховищі даних підприємства є кіоск даних, у якому зберігаються щоденні дані, і за ним фінансовому відділу потрібен щомісячний звіт; у цьому разі у вас може з'явитися бажання статистично опрацювати дані цього кіоску на місячній основі. Інший приклад: ви хочете застосувати перетворення Merge Join, а вихідний потік даних не відсортовано, у цьому разі використовується компонент Sort.

Таблиця 6.4 – Види перетворень набору рядків

Перетворення потоку даних	**Призначення**	**Тип блокування**
1	2	3
Aggregate (Статистичне оброблення)	Зв'язує рядки на основі заданого групування та генерує агрегати, такі як SUM, MAX, MIN та COUNT	B

Продовження таблиці 6.4

1	2	3
Percent Sampling (Відсоткова вибірка)	Фільтрує вхідні рядки, дозволяючи лише заданому відсотку рядків потрапити у вихідний шлях потоку даних	N
Pivot (Зведення)	Приймає множинні вхідні рядки та зводить їх для створення виходу з додатковими стовпцями, що ґрунтуються на значеннях вихідних рядків	P
Row Sampling (Вибірка рядків)	Генерує фіксовану кількість рядків, випадковим чином вибираючи рядки з усього вхідного набору, причому неважливо, наскільки вхідний набір більший за заданий вихідний	B
Sort (Сортування)	Упорядковує вхідний набір даних, ґрунтуючись на заданих стовпцях сортування та порядку сортування. Це перетворення також дозволяє видалити дублікати в стовпцях, якими ведеться сортування	B
Unpivot (Скасувати відомості)	Приймає єдиний рядок і генерує множинні рядки, переміщуючи значення стовпців у новий рядок, ґрунтуючись на заданих стовпцях	P

У випадках перетворень Sort, Aggregate та Row Sampling усі вхідні рядки повинні бути зчитані, перш ніж вони можуть бути відправлені у вихідний шлях. Ось чому ці перетворення застосовують повне блокування.

Додаткові перетворення для підготовки даних. Остання група перетворень дозволяє виконувати додаткові операції над рядками в конвеєрі потоку даних, вона описана в таблиці 6.5.

Таблиця 6.5 – Види додаткових перетворень для підготовки даних

Перетворення потоку даних	**Призначення**	**Тип блокування**
1	2	3
DQS Cleansing (Очищення DQS)	Перевіряє коректність рядків, виконуючи автоматичне очищення за допомогою наявної бази знань у службах Data Quality Services (DQS)	P
OLE DB Command (Команда OLE DB)	Виконує операції над базою даних, такі як оновлення або видалення, рядково, ґрунтуючись на зіставленні параметрів із вхідних рядків	N
Slowly Changing Dimension (Повільно змінний вимір)	Генерує перетворення, необхідні для підтримки завантаження таблиць вимірів у сценаріях сховищ даних. Це перетворення обробляє вимірювання SCD Type 1 та Type 2 і має підтримку виведених елементів вимірювань	N
Data Mining Query (Запит інтелектуального аналізу даних)	Додає вхідні рядки до моделі інтелектуального аналізу для прогнозування	P
Fuzzy Grouping (Нечітке групування)	Усуває дублікати, зважаючи на подібності рядкових значень у вибраних стовпцях	B
Fuzzy Lookup (Нечіткий уточнювальний запит)	З’єднує вхід потоку даних із таблицею посилань на підставі подібності стовпців. Параметр Similarity Threshold (Поріг подібності) задає ступінь близькості дозволених порівнянь: чим вищий поріг, тим	B

Продовження таблиці 6.5

1	2	3
	більш подібними повинні бути значення	
Script Component (Скрипт)	Застосовує функціональні можливості сценаріїв .NET до рядків, стовпців, вхідних та вихідних наборів у конвеєрі потоку даних. Це найпотужніший компонент	N
Term Extraction (Вилучення термінів)	Аналізує текстові вхідні стовпці, знаходячи іменники та словосполучення з іменників англійською мовою	P
Term Lookup (Уточнюючий запит терміна)	Перевіряє текстові вхідні стовпці на відповідність набору слів, визначеному користувачем	P

Застосування перетворень

Як і у випадку адаптерів джерела та призначення, за допомогою мишки можна перетягнути перетворення з панелі SSIS Toolbox на вкладку Data Flow конструктора SSIS Designer. У кожного перетворення є вікно для редагування, в якому описується спосіб застосування операції до даних. Відкрити редактор можна або двічі клацнувши кнопкою мишки на перетворенні, або клацнувши на ньому правою кнопкою мишки та вибравши команду Edit. Наприклад, перетворення Derived Column (Похідний стовпець) задає вираз, що генерує новий стовпець у потоці даних або заміщає наявний стовпець. На рисунку 6.7 можна побачити, що додається один новий стовпець FullName за допомогою конкатенації стовпців FirstName і LastName, яка задається таким виразом SSIS:

[LastName] + " " + [FirstName].

Перетворення Derived Column може замінювати існуючі стовпці. У цьому прикладі стовпець Suffix замінюється виразом, що перевіряє наявність значення NULL і повертає N/A, якщо воно є, і зберігає колишнє значення стовпця за його відсутності.

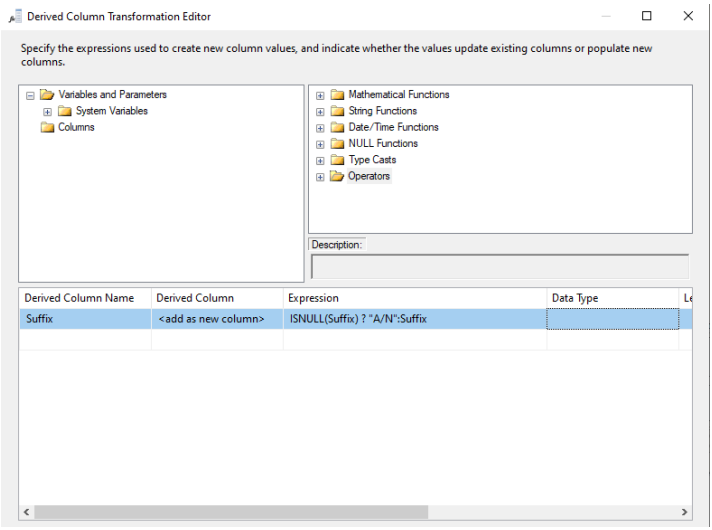

Рисунок 6.7 – Вікно Derived Column Transformation Editor

Повний алгоритм оброблення потоку даних описується шляхом з'єднання адаптерів джерел даних, перетворень та адаптерів призначень шляхами даних, що створюються перетягуванням стрілки виходу одного компонента на інший компонент потоку даних. Сині стрілки даних призначені для рядків, успішно перетворених, червоні стрілки – для рядків, у яких перетворення завершилося з помилкою, наприклад округлення або конвертації. На рисунку 6.8 показано частину потоку даних для оновлення виміру про замовників.

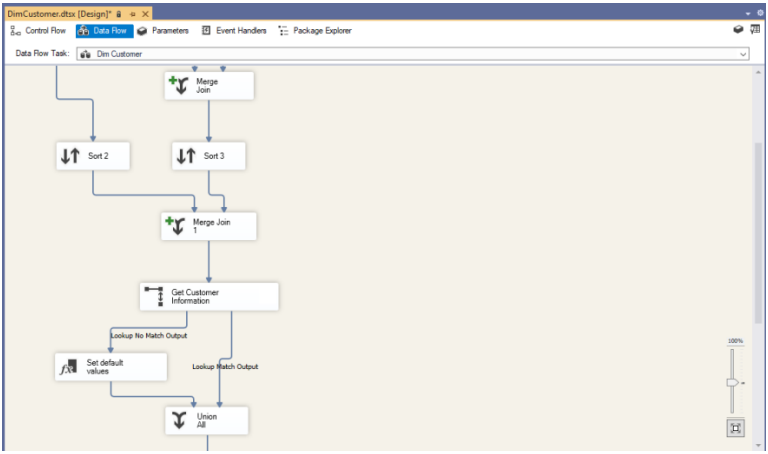

Рисунок 6.8 – Потік даних із декількома перетвореннями, з'єднаних шляхами даних

Дозвіл посилань на стовпці. Коли метадані про стовпці в завданні потоку даних змінюються, потрібно відкоригувати посилання на стовпці. Зазвичай така необхідність виникає, коли додаються нові стовпці з джерела, видаляються деякі стовпці у конкретних перетвореннях і навіть видаляються деякі перетворення, вимагаючи нового порівняння наявних компонентів.

До служби SSIS внесено редактор Resolve References (Редактор дозволу посилань), який можна застосовувати для швидкого одержання зіставлення вхідних та вихідних стовпців між компонентами. На рисунку 6.9 показаний редактор, який можна відкрити, клацнувши правою кнопкою мишки на шляху даних між двома перетвореннями та вибравши команду Resolve Reference (Дозволити посилання).

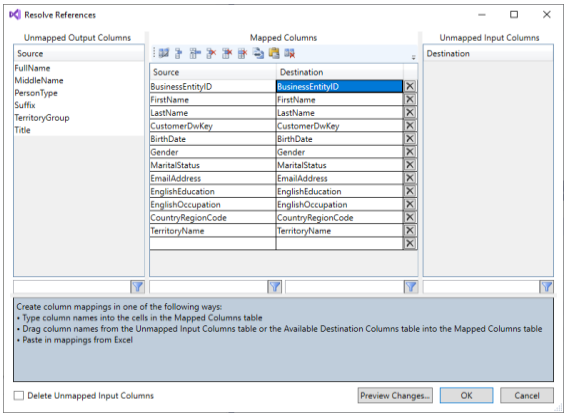

Рисунок 6.9 – Редактор Resolve References

У редакторі дозволу посилань можна пов’язати непоставлені вихідні стовпці з непорівнянними вхідними стовпцями по всіх шляхах, що формують шлях потоку даних. Ви також можете застосовувати його для перевірки коректності порівнянь та виявлення непорівнянних стовпців.

Вибір відповідних стратегій ETL та інструментів

Обсяг даних, якими повинні керувати сховища даних, зростає з кожним днем, і інтеграція даних стає найбільшою проблемою поряд із зростанням потреби щодо реалізації різних аналітичних рішень, починаючи зі сховищ даних підприємства і закінчуючи спеціальними вітринами даних для наукового прогнозування. Найбільша частина вартості та обслуговування процесів інтеграції складних даних займає область масового переміщення даних. Крім форматів стандартного неструктурованого файлу та реляційних даних середовищам, зайнятим інтеграцією даних, доводиться брати до уваги формати XML та неструктурованих даних. З урахуванням цих нових форматів і експоненційного зростання транзакційних даних процес інтеграції даних стає дедалі складнішим і трудомістким.

Стратегія ETL

Стратегія ETL – дуже широкий термін. До стратегії ETL можна віднести всю методологію розроблення, починаючи з

фази аналізу проєкту, на якій ви порівнюєте логічні моделі із системами вихідних даних, або вона може заключати в собі більш спеціальний погляд на процес розроблення, що мінімізує потенційні ризики інтеграції даних. Зосередимося на трьох технічних складових:

– визначення архітектури для ETL;

– виборі частини процесу, що виконується в службах SSIS, і передавання частини, що залишилася на рівень бази даних;

– керування процесом ETL загалом.

Архітектура ETL. В еталонній архітектурі інтеграції даних визначено процеси та конфігурації, що підтримують захват, перевірку якості, оброблення та переміщення даних, транзакційних або масових, необхідних для одного призначення або для кількох. Типовий шар архітектури ETL складається з процесу та зони паркування (landing zone), зазвичай розміщеної всередині бази даних. За кожним процесом йде зона паркування.

Еталонна архітектура складається з:

– вилучення даних (процес);
– області початкового розміщення (зона паркування);
– визначення якості даних (процес);
– області очищення (зона паркування);
– перетворення (процес);
– готової до завантаження публікації (зона паркування);
– завантаження сховища даних підприємства (процес);
– сховища даних підприємства (зона паркування);
– завантаження вітрин даних (процес);
– вітрини даних (зона паркування).

Під час розроблення цієї архітектури необхідно вирішити, які зони паркування матеріалізувати, записавши дані в конкретні таблиці у схемі чи базі даних. Зазвичай початкове розміщення після фази вилучення (Extract) дуже вигідне і може бути повною копією, одержанням змінених даних або додатковою копією. Якщо використовуються різні джерела даних, то це перша область, де можна все уніфікувати в єдиній базі даних і застосувати мову SQL до всіх даних. Наступна фаза

залежить від якості даних. Якщо вона низька, буде потрібна додаткова взаємодія з користувачами підприємства та операторами даних, і в такому разі ця зона паркування буде дуже корисною. Перетворення даних та завантаження їх у сховище даних зазвичай виконуються у службах SSIS без явного проміжного розміщення перетворених даних. Завдання ETL для вітрин даних також може бути досить складним – усе залежить від моделі сховища даних та його розміру.

На підставі цієї архітектури процес ETL, що використовує служби SSIS, повинен складатися з декількох пакетів, кожен із яких відведений для окремого процесу. Краще мати кілька пакетів, ніж безліч потоків даних в одному пакеті. Крім того, не варто забувати про розробку в групі, якою можна керувати за наявності декількох пакетів.

Перетворення Lookup

Під час вставлення даних у таблицю фактів сховища даних необхідно одержати з таблиці вимірів відповідний ключ сховища даних (зазвичай сурогатний ключ). Як було розглянути раніше, стовпці в таблиці фактів містять зовнішні ключі та виміри, а виміри у вашій базі даних визначають зовнішні ключі. Для вирішення цієї проблеми в процесі ETL дуже корисно застосовувати перетворення Lookup (Уточнювальний запит), тому що воно виконує пошук, з'єднуючи дані у вхідних стовпцях зі стовпцями в стандартному наборі даних (reference dataset).

Перетворення Lookup додають, переміщуючи за допомогою мишки компонент із панелі SSIS Toolbox (Елементи служб SSIS) на робоче поле конструктора потоку даних. Еталонний набір даних задається в редакторі Lookup Transformation Editor (Редактор перетворення «Уточнювальний запит»). Він може бути файлом кешу, наявною таблицею або поданням, новою таблицею або результатом SQL-запиту. Для підключення до еталонного набору даних перетворення Lookup використовує диспетчер з'єднань OLE DB або диспетчер з'єднань Cache. На рисунку 6.10 можна переглянути налаштування на сторінці General (Загальні).

Зазначити спосіб збереження еталонного набору в пам’яті можна вибираючи відповідний режим кешування.

– *Повне кешування* (full cache) – режим за замовчуванням. У цьому режимі база даних запитується один раз під час фази виконання потоку даних, і весь еталонний набір даних зберігається в оперативній пам’яті. Операції Lookup будуть виконуватися дуже швидко, але необхідно мати достатньо пам’яті, щоб помістити в неї еталонний набір.

– *Часткове кешування* (partial cache) означає, що кеш перетворення порожній на початку потоку даних. Коли надходить новий рядок, перетворення Lookup перевіряє свій кеш на наявність відповідних значень. Якщо збігів не знайдено, він запитує базу даних. Якщо збіг знайдено в базі даних, значення переносяться в кеш, таким чином, вони можуть використовуватися наступного разу, коли надійде рядок, що збігається.

– Режим *без кешування* (no cache) буде зберігати лише останній рядок, що збігся, тобто перетворення Lookup буде запитувати базу даних для кожного рядка. Під час завантаження у сховище даних таблиць фактів необхідно уникати застосування цього режиму, оскільки завантаження буде надто повільне.

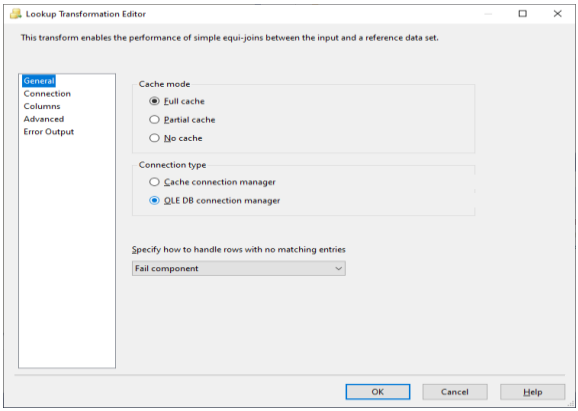

Рисунок 6.10 – Сторінка General у вікні Lookup Transformation Editor

Еталонний набір даних задається на сторінці Connection (З'єднання) діалогового вікна. Оскільки оптимальний вибір – режим Full cache, найкраще написати SQL-запит, який повертає лише потрібні стовпці. Наприклад, якщо ви вибрали режим Use a table or a view (Використовувати таблицю або подання) і ця таблиця вимірів містить 50 або більше стовпців, усі вони зберігатимуться в оперативній пам'яті. Зазвичай вам потрібна лише пара стовпців – ті, які будуть потрібні для з'єднання, і стовпці, які потрібно витягти.

На сторінці Columns (Стовпці) редактора Lookup Transformation Editor задається складне з'єднання (з'єднання кількома стовпцями). Якщо потрібно записати з'єднання із зазначенням діапазону, наприклад між початковою та кінцевою датами, необхідно застосовувати варіант Custom query (Запит користувача) на сторінці Advanced (Додатково). У цьому разі доступний лише режим Partial cache, а не Full cache.

Інший важливий параметр, який можна налаштувати в редакторі Lookup Transformation Editor, – це спосіб оброблення рядків, для яких немає збігів. Ви можете завершити роботу компонента з помилкою, ігнорувати цю помилку або перенаправити рядки або в новий вихід рядків, що не збігаються, або у виведення помилок. Вибір здійснюється на сторінці General (Загальні).

З поглчду оброблення потоку даних здається природним за відсутності збігу застосувати в такому рядку деяке стандартне значення, а потім об'єднати обидва виходи в один і записати дані адаптера призначення. Але з погляду продуктивності та накопиченого практичного досвіду краще застосувати інший підхід. У редакторі Lookup Transformation Editor потік даних налаштовується на пропуск помилки завданням відповідного варіанта в списку Specify how to handle rows with no matching entries (Зазначте метод оброблення рядків без елементів, що збігаються). Це означає, що в результаті з перетворення Lookup потраплять рядки з елементами, що збігаються, і рядки зі значенням NULL в стовпцях результатів пошуку.

Далі додається компонент Derived Column (Похідний стовпець) і записується вираз SSIS Expression для заміни наявного значення в стовпці з результатом пошуку таким логічним виразом:

ISNULL(CustomerDwKey)? 0 : CustomerDwKey.

Другий метод набагато швидший і споживає менше ресурсів. Це тому, що перетворення Union All застосовує часткове блокування.

Застосування конвертування Cache Transform з перетворенням Lookup. Перетворення Cache Transform (Перетворення кешу) записує дані з підключеного джерела даних потоку даних диспетчер з'єднань із кешем. Диспетчер з'єднань із кешем – альтернатива виконанню уточнювальних запитів до таблиці бази даних за допомогою диспетчера з'єднань OLE DB.

Після того, як кеш створений у пакеті служб SSIS, він зберігатиметься в оперативній пам'яті доти, доки пакет не завершить виконання, якщо диспетчер з'єднань із кешем створений на рівні пакета. Якщо він створений на рівні проєкту, кеш може спільно використовуватися кількома пакетами. За такого підходу кеш може багаторазово використовуватися множинними пакетами і потоками даних і спільно застосовуватися в множинних запитах. Він може бути збережений на диск.

Далі перелічені переваги застосування диспетчера з'єднань із кешем.

– Він дозволяє повторно використовувати кеш, знижуючи навантаження.

– Він дозволяє спільно використовувати кеш у кількох уточнювальних запитах, знижуючи цим споживання оперативної пам'яті. Наприклад, розглянемо рольовий вимір (role-playing dimension) у сховищах даних, що становить одну фізичну таблицю, але в таблиці фактів існують численні зв'язки за зовнішніми ключами для кожної ролі виміру (наприклад, для виміру Date ними можуть бути стовпці InvoiceDateID, ShippingDateID, DocumentDateID тощо).

– Можна виконувати уточнювальні запити до інших джерел (не OLE DB).

З погляду режимів кешування та передового практичного досвіду, пов'язаного з ним, застосування диспетчера з'єднань із кешем еквівалентне використанню режиму Full cache. Оскільки по суті кеш – це чистий текст, не рекомендується зберігати у ньому чутливі до регістру дані.

Сортування даних

Якщо дані зчитуються з бази даних – з проміжної області або безпосередньо з транзакційної системи, – то можна перекласти витрати на сортування даних на основну базу даних, якщо задати в компоненті Data Source (Джерело даних) SQL-запит користувача з розділом ORDER BY.

Потім необхідно проінформувати підсистему оброблення потоку даних про те, що джерело даних відсортовано. У діалоговому вікні Advanced Editor (Розширений редактор) для створеного компонента джерела даних у вузлі Output Columns (Вихідні стовпці) на вкладці Input and Output Properties необхідно задати властивість SortKeyPosition для стовпців, які є частиною ключа сортування (1 – для першого стовпця, 2 – для другого тощо). 0 – значення за замовчуванням для всіх стовпців, які не входять до ключа сортування. Необхідно також зазначити, що вихід повністю відсортований, вибравши вихід і встановивши значення True для властивості IsSorted.

Оновлення на основі наборів

У середовищі сховища даних зазвичай потрібно уникати застосування операцій UPDATE та DELETE над даними та у стратегії ETL. Ця рекомендація стосується насамперед великих таблиць фактів. Але все одно є потреба виконувати оновлення даних, якщо це стосується таблиць виміру.

На даний момент єдиний спосіб оновлення в службах SSIS – використовувати перетворення OLE DB Command (Команда OLE DB), що виконує інструкцію SQL із застосуванням стовпців поточного рядка як параметрів. Ця операція виконується для кожного рядка в потоці даних і тому

корисна лише в особливих випадках, коли загальна кількість SQL-інструкцій, що виконуються, мала. Як приклад розглянемо оновлення таблиці вимірювань dbo.Customer, у якій понад 100 000 рядків. Припустимо, що перетворення Lookup повертає 30 000 рядків, що збіглися, які потребують оновлення. Якщо для оновлення цих рядків ви зв'яжете вихід із OLE DB Command, перетворення виконає SQL-інструкцію 30 000 разів. Як бачите, це буде дуже повільне оновлення, якого потрібно уникати.

Для цього можна виконати комбінацію застосування в потоці керування спочатку завдання потоку даних, а потім завдання Execute SQL Task (Виконання SQL). Завдання потоку даних записує дані, що потребують модифікації, в додаткову проміжну таблицю, а потім завдання Execute SQL Task застосує інструкцію UPDATE або MERGE для виконання оновлення на основі набору даних.

Цей приклад демонструє важливість розуміння того, що краще робити на рівні бази даних, а що виконувати у службах SSIS. За такого підходу оновлення 30 000 рядків займе декілька секунд порівняно з годиною під час перетворення OLE DB Command.

Тестові питання

А. Установіть відповідність між типами потоків даних та їх призначеннями.

	Вилучення даних джерел	**Завантаження даних у джерела**	**Модифікація даних за алгоритмами**	**Визначення місця передавання даних**
Адаптери джерел даних	☐	☐	☐	☐
Перетворення потоку даних	☐	☐	☐	☐
Призначення потоку даних	☐	☐	☐	☐

В. Зазначте правильний зв'язок між адаптерами:

☐ адаптер потоку даних пов'язує джерела даних із адаптером призначення;

☐ адаптер джерела визначає потік даних та призначення даних;

☐ адаптер призначення даних з'єднує джерело даних та потік даних.

C. Під час перетворення даних чи можна трансформувати частину рядків даних:

☐ ні, трансформація потоку даних перетворює всю множину рядків даних;

☐ ні, трансформація потоку даних лише переміщує дані з одного джерела в інше;

☐ так, можлива трансформація частково накопичених даних;

☐ так, але ця трансформація відбувається у сховищі даних.

D. Які адаптери джерел потоку даних можна застосувати, якщо ви хочете прочитати дані з SQL Server (Виберіть усі відповідні варіанти.):

☐ джерело ADO NET;

☐ джерело Raw File (Джерело «Необроблений файл»);

☐ джерело OLE DB;

☐ джерело ODBC?

E. Які призначення потоку даних можна використовувати, якщо ви хотіли б тимчасово розмістити дані у файловій системі (Виберіть усі відповідні варіанти.):

☐ призначення OLE DB;

☐ призначення Flat File (Призначення «Неструктурований файл»);

☐ призначення Raw File (Призначення «Необроблений файл»);

☐ призначення Recordset (Призначення «Набір записів»)?

F. Які твердження справедливі щодо адаптерів джерел даних (Виберіть усі відповідні варіанти.):

☐ ви можете змінити спосіб порівняння даних із типами даних служб SSIS;

☐ у кожного завдання потоку даних може бути лише один адаптер джерела даних;

☐ з адаптера джерела завжди необхідно вибирати всі стовпці;

☐ за допомогою служб SSIS можна прочитати дані з XML-файлу?

G. Яке перетворення можна застосувати для перетворення даних з одного типу в інший (Виберіть усе, що підходить.):

☐ Audit;

☐ Derived Column;

☐ Data Conversion;

☐ Script Component?

H. Які перетворення повністю блокують (Виберіть усі відповідні варіанти.):

☐ перетворенняLookup;

☐ перетворення Sort;

☐ перетворення Merge Join;

☐ перетворення Aggregate?

I. Які перетворення ви почали б використовувати, якщо потрібно було б об'єднати дані з двох таблиць різних баз даних, що знаходяться на двох різних серверах (Виберіть усе, що підходить.):

☐ перетворення Merge Join;

☐ перетворення Union All;

☐ перетворення Merge;

☐ перетворення Lookup?

J. Припустимо, що ви хочете завантажити дані з неструктурованого файлу і записати їх на SQL Server, у програму Excel і в перетворення Raw File всередині одного завдання потоку даних. Скільки адаптерів Data Source ви могли б використовувати (Виберіть усе, що підходить.):

☐ 0;

☐ 1;

☐ 2;

☐ 3;

К. Яке з тверджень справедливе щодо диспетчера з’єднань із кешем (Виберіть усе, що підходить.):

☐ кеш може повторно використовуватися багатьма перетвореннями Lookup;

☐ під час виконання пакета не можна виконувати додаткове оновлення кешу;

☐ диспетчер з’єднань із кешем можна задати на рівні проєкту;

☐ ви можете виконувати уточнювальні запити до інших (не OLE DB) джерел?

Аналітичні питання

Завдання 1. Система з новим джерелом.

Відділ маркетингу попросив додати до сховища даних інформацію про питому вагу продукції компанії над ринком. Керівництво відділу придбало річну передплату на інформаційну розсилку, і дані доступні щомісяця на FTP-сервер. Дані зберігаються в неструктурованому файлі. Він містить інформацію про товарну групу в першому полі та відомості про частку ринку в наступних 12 полях (окреме поле для кожного місяця). Вам потрібно інтегрувати в наявне сховище даних нове джерело. Дайте відповідь на такі запитання:

1.1. Як ви моделюватимете таблицю фактів для зберігання відомостей про частку ринку?

1.2. Які завдання та перетворення служб SSIS ви застосуєте для заповнення цієї таблиці фактів?

МАТЕРІАЛ ДЛЯ ПІДГОТОВКИ ДО ВИКОНАННЯ ПРАКТИЧНОЇ РОБОТИ 7

Теоретичний матеріал

Удосконалення пакетів SSIS. Удосконалення потоку керування

Пакети служб Microsoft SQL Server Integration Services (SSIS) – автоматичні пакети, їх функціонування не потребує взаємодії з користувачем. Важливий акцент робиться на розробленні SSIS: кожен процес і кожна операція, незалежно від їхньої складності, повинні плануватися і реалізовуватися так, щоб їх виконання можна було автоматизувати. Мається на увазі не лише саме виконання, а й виявлення помилок. Реакція на помилки також повинна бути автоматизована, і в ідеалі це стосується і відновлення виконання після помилок.

Автоматизація стає можливою насамперед завдяки *детермінізму*. Якщо операція може бути описана детерміновано (і перекладена мовою програмування), вона може бути автоматизована. Переважна частина операцій зберігання даних автоматизована; вони виконуються машинами, а не людьми, дозволяючи нам, людям, більшу частину робочого дня витрачати на те, що ми робимо найкраще – міркувати та реагувати на проблеми, які сучасні машини не здатні вирішувати.

Передбачуваність (або прогнозованість) також є дуже важливою для автоматизації. Вона є основою розроблення розв'язків та уможливлює багаторазове використання. Можливість багаторазового використання досягається, якщо один раз спроєктований пакет потім застосовується багаторазово. Багаторазове використання зменшує час, необхідний для розробленя пакета, знижує потребу в ресурсах і полегшує розгортання пакетів та його обслуговування.

Ще одна невід'ємна складова ефективної автоматизації – *адаптованість*. Для того щоб автоматизований пакет міг виявляти проблеми і реагувати на них без втручання людини або навіть насамперед перешкоджати їх появі, він повинен бути

адаптованим, здатним до самоналаштування. Він повинен визначати стан робочого середовища та під час виконання налаштовуватися відповідним чином.

Значний детермінізм, характерний багатьом процесам керування даними, особливо у сховищах даних, дозволяє автоматизувати ці процеси.

Можливість визначити на етапі розроблення, які операції керування даними потрібні, в яких умовах вони виконуватимуться і в якому порядку є основними умовами автоматизації.

Проте детермінований не означає жорстко заданий – хоча більшу частину обставин можна передбачити на етапі розроблення, деякі, на жаль, залишаються невизначеними. Те, що не можна передбачати, не можна попередньо встановити чи налаштувати. Прикладом можуть бути відмінності між середовищем розробки та виробничим середовищем: будь-який елемент автоматизованого процесу, що залежить від середовища і, отже, не налаштований належним чином, може перетворити розгорнутий проект на марний або навіть шкідливий або взагалі перешкодити його виконанню.

Для адекватної реакції на такі проблеми пакети служб SSIS можуть бути параметризовані: це дозволить залишити незмінною суть автоматизованого процесу та відповідним чином відкоригувати властивості, що залежать від робочого середовища чи інших зовнішніх обставин. Зазвичай адміністратор, який відповідає за обслуговування, використовує параметри для керування виконанням процесу за допомогою оголошених властивостей.

Параметризація забезпечує насамперед не лише розгортання процесу, а й розширення можливості повторного застосування пакетів служб SSIS, дозволяючи розгортати один й той самий пакет багаторазово в різних виробничих середовищах.

Високий ступінь детермінізму може навіть дозволити автоматично задавати деякі властивості пакетів SSIS – або тому що вони базуються на параметризованих налаштуваннях (наприклад, якщо кореневий шлях, що вказує місце

розташування файлів даних параметризований і буде в результаті керуватися адміністратором, шляхи до вкладених папок кореневої папки можна задавати автоматично), або тому що вони ґрунтуються на властивостях, що автоматично виявляються (наприклад, кількість процесорів, обсяг доступної системної пам'яті або обсяг вільного дискового простору) і, отже, не потребують налаштування адміністратором.

Змінні SSIS

Змінні SSIS можуть використовуватися для зберігання значень, які визначаються автоматично під час виконання та повинні використовуватись або багаторазово використовуватись у різних елементах пакета. Деякі змінні задаються один раз – наприклад, на початку виконання або безпосередньо перед моментом їх використання – і потім можуть застосовуватися багаторазово, тоді як інші змінні (такі як ім'я файлу, що обробляється контейнером Foreach Loop, що використовує Foreach File Enumerator (Лічильник із циклом за кожним файлом), або значення з поточного рядка в контейнері Foreach Loop, що використовує Foreach ADO Enumerator (Лічильник ADO Foreach)) задаються ітераційно і використовують до наступної ітерації.

Параметри та змінні можуть застосовуватись у пакетах SSIS на рівних підставах, але є кілька відмінностей, про які необхідно знати:

– параметри видно стороні, яка викликає, а змінні не видно. Якщо конкретна властивість повинна задаватися автоматично і не повинна визначатися адміністратором, використовуйте змінну, в протилежному випадку – параметр;

– у пакеті SSIS параметри доступні лише для читання. Вони можуть бути задані лише стороною, яка викликає, і після установлення не можуть бути змінені.

Існують два види змінних служб SSIS.

– Фіксований набір *системних змінних* представляє конкретні властивості різних об'єктів служб SSIS (пакетів, завдань, контейнерів, компонентів та обробників подій).

Системні змінні доступні лише читання; їх значення задаються службами SSIS.

– *Користувальницькі або визначені користувачем змінні*, визначені розробником SSIS і використовувані для зберігання різної інформації, одержаної або сформованої під час виконання. За замовчуванням змінні, визначені користувачем, можна записувати дані, але є можливість обмежити їх застосування лише зчитуванням даних із них.

Крім згаданих обмежень, у поведінці системних і користувацьких змінних немає різниці.

Зазвичай змінні використовуються для зберігання скалярних значень, але можуть застосовуватися для зберігання наборів рядків, наприклад, наборів рядків, які використовуються в контейнері Foreach Loop Container із лічильником Foreach ADO Enumerator. Змінні SSIS можуть використовуватися завданнями потоку керування, контейнерами та компонентами потоку даних, вони також доступні обробникам подій.

Змінні можна створювати за допомогою вікна Variables або діалогового вікна Add Variable, доступного в конкретному завданні та редакторах компонентів. Вони навіть можуть бути додані програмно. Загальний перелік змінних і їх властивостей поданий у таблиці 7.1.

Для відображення системних змінних у вікні Variables можна встановити фільтр у діалоговому вікні Variable Grid Options (Параметри сітки змінних), показаному на рисунку 7.1. Це діалогове вікно доступне з команди панелі інструментів Grid Options (Параметри сітки).

Таблиця 7.1 – Опис властивостей змінних

Властивість	**Опис**
1	2
Name	Ім’я змінної. Воно може надаватися, коли створюються змінні користувача
Scope	Область дії змінної, яка визначає, які об’єкти служб SSIS зможуть отримати доступ до неї. Область дії змінної більш докладно буде обговорюватися пізніше в цьому розділі
Data type	Тип даних змінної. Типи даних змінних більш детально обговорюватимуться пізніше в цьому розділі
Value	Значення змінної. Усі змінні користувача повинні бути ініціалізовані за допомогою значення за замовчуванням за винятком змінних типу Object або DBNull
Namespace	Для змінних двох типів існують два простори імен змінної: системні змінні існують у просторі імен System, а змінні користувача – у просторі імен User
RaiseChangeEvent	Логічне значення, що визначає, чи буде згенерована подія, коли змінюється значення змінної; може бути корисним для реєстрації та виявлення несправностей
Description	Необов’язковий опис змінної, що використовується переважно для документування
Expression	Вираз застосовується для визначення величини змінної. Воно корисне для змінних, чиї значення визначаються автоматично за допомогою виразу на основі інших доступних системних або змінних користувача або параметрів. Вирази змінних докладніше обговорюватимуться пізніше в цьому розділі
EvaluateAsExpression	Логічне значення, яке зазначається, задається величина змінної за допомогою виразу чи ні. Цю властивість не можна задати у вікні Variables, але вона доступна у вікні Properties (Властивості) або у разі програмного додавання змінної

Продовження таблиці 7.1

1	2
Read only	Логічне значення, яке визначає, чи можна змінювати змінну чи вона доступна лише для читання. Цю властивість не можна задати у вікні Variables, але воно доступне в діалоговому вікні Add Variable і у вікні Properties або в разі програмного додавання змінної
IncludeInDebugDump	Логічне значення, яке визначає, включається змінна в налагоджувальні файли дампа чи ні. За замовчуванням значення цієї властивості True, як для системних, так і для змінних користувача. Але значення автоматично встановлюється рівним False для змінних на основі виразів і змінних тип яких змінено на рядковий. Цю властивість не можна задати у вікні Variables, але вона доступна у вікні Properties або в разі програмного додавання змінної

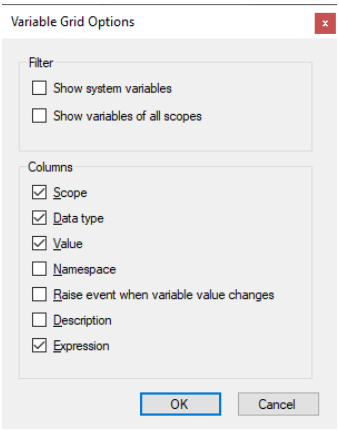

Рисунок 7.1 – Діалогове вікно Variable Grid Options

Типи даних змінних

Під час проєктування змінні налаштовуються, перевіряються (значення, задане в конструкторі, або значення, задане виразом, перевіряється на відповідність певному типу

даних, і в разі невідповідності типів генерується помилка часу проєктування) та заносяться до списку (робляться доступними редакторам різних об'єктів служб SSIS). Але до часу виконання, коли компілюється пакет служб SSIS, змінні досі не створені; у процесі виконання, під час перевірки пакета, змінні перевіряються знову, і якщо виявляється невідповідність типів, генерується помилка часу виконання. Можуть бути використані такі типи даних змінних:

Empty – Порожнє посилання. Цей тип даних є частиною переліку System.TypeCode, але не підтримується середовищем SSDT.

Object – Загальний тип, що представляє будь-яке посилання або значення типу, не представленого явно іншим типом TypeCode. Цей тип можна використовувати для збереження значень типів даних, які не підтримуються в змінних службах SSIS.

DBNull – Неіснуюче значення (стовпець) бази даних. Застосовується для явного передавання значення NULL у завдання, що його використовує, або компоненту.

Boolean – Простий тип, що представляє логічні значення True або False.

Char – Цілочисловий тип, що представляє 16-бітові цілі числа без знака з діапазону 0–65 535. Ця безліч можливих значень типу Char відповідає множині символів Unicode.

SByte – Цілочисловий тип, що представляє 8-бітові цілі числа зі знаком і діапазоном значень мінус 128–127.

Byte – Цілочисловий тип, що представляє 8-бітові цілі числа без знака з діапазоном значень 0–255.

Int16 – Цілочисловий тип, що представляє 16-бітові цілі числа зі знаком і діапазоном значень мінус 32 768–32 767.

UInt16 – Цілочисловий тип, що представляє 16-бітові цілі числа без знака з діапазоном значень 0–65 535.

Int32 – Цілочисловий тип, що представляє 32-бітові цілі числа зі знаком і діапазоном значень мінус 2 147 483 648–2 147 483 647.

UInt32 – Цілочисловий тип, що представляє 32-бітові цілі числа без знака з діапазоном значень 0–4 294 967 295.

Int64 – Цілочисловий тип, що представляє 64-бітові цілі числа зі знаком і діапазоном значень мінус 9 223 372 036 854 775 808–9 223 372 036 854 775 807.

UInt64 – Цілочисловий тип, що представляє 64-бітові цілі числа без знака з діапазоном значень 0–18 446 744 073 709 551 615.

Single – Тип з плаваючою комою, що представляє значення приблизно з діапазону 1,5×10–45–3,4×103 сім десяткових розрядів.

Double – Тип з плаваючою комою, що становить значення приблизно з діапазону 5,0×10–324–1,7×10 308 і точністю 15 або 16 десяткових розрядів.

Decimal – Простий тип, що представляє значення приблизно з діапазону 1,0×10–28–7,9×1 028 з 28 або 29 значущими цифрами.

DateTime – Тип, що представляє значення дати та часу.

String – Тип задрукованого класу (sealed class), що представляє рядки символів Unicode.

Крім набору типів даних, які використовуються змінними SSIS, у службах SSIS є дві додаткові групи типів даних.

– Кожен постачальник даних підтримує власний набір типів даних, які відрізняються в різних постачальників (наприклад, OLE DB, ADO.NET або ODBC).

– У буфері потоку даних застосовується набір спеціальних типів даних, що надаються підсистемою оброблення SSIS і використовуються компонентами потоку даних.

Кожна з цих трьох груп типів даних служить певній меті, підтримуючи загальне принципове завдання служб SSIS – надання спільного середовища для інтеграції даних, що походять із різнорідних сховищ даних. Переважна більшість випадків звернення рішень служб SSIS до даних, що зберігаються в різних сховищах, не повинні мати проблеми сумісності будь-яких типів даних. Але завжди необхідно

пам’ятати, що не кожен тип даних з однієї групи можна порівняти з найбільш відповідним типом з іншої групи. Деякі типи даних навіть не підтримуються певними наборами типів даних у службах SSIS.

Якщо змінні використовуються для зберігання наборів рядків, тобто для зберігання результату запиту мовою Transact-SQL, яка формує один або кілька наборів рядків, як, наприклад, результат завдання Execute SQL Task (Виконання SQL), можна застосувати такі два типи даних.

– Якщо запит формує звичайний набір рядків, можна зберегти результат змінної типу Object. Змінна типу Object може потім використовуватися користувачем цього набору рядків (такі як Foreach Loop Container (Цикл за кожним елементом) з Foreach ADO Enumerator (Лічильник ADO Foreach) або завдання Script Task (Скрипт), або компонент Script Component з відповідним програмним алгоритмом для доступу до даних набору рядків).

– Якщо запит формує XML-результат, його можна зберегти в змінній типу Object або String. Змінна типу Object, що містить XML-представлення набору даних ADO.NET, може використовуватися користувачем XML-даних (таким як Foreach Loop Container з Foreach NodeList Enumerator (Лічильник Foreach NodeList), завдання Script Task або компонент Script Component з програмним алгоритмом користувача доступу до даних у XML-документі).

Область дії змінних

Область дії (scope) змінної визначається ієрархією об’єктів служб SSIS; область дії обмежує доступ і видимість змінної конкретної гілки в ієрархії об’єктів SSIS.

Пакет представляє кореневий елемент (вершину) ієрархії; отже, всі змінні, визначені на рівні пакета, можуть бути доступні будь-яким завданням, контейнерам або компонентам, визначеним усередині цього пакета. Змінні з областю дії лише на рівні пакета можна розглядати як глобальні змінні.

Змінні, визначені на рівні контейнера, доступні лише цьому контейнеру та всім об’єктам, що містяться в ньому

(контейнери можуть містити завдання та інші контейнери). В ієрархії об'єктів служб SSIS об'єкти мають доступ до власних локальних змінних та до змінних відповідних об'єктів-предків.

Змінні, визначені на рівні завдання, доступні лише в даному конкретному завданні, оскільки завдання не можуть містити інші об'єкти служб SSIS. Єдиний виняток із цього правила – завдання потоку даних. Змінні, визначені в завданні потоку даних, можуть використовуватися компонентами, що містяться в цьому завданні даних; в інших випадках змінні, визначені на рівні завдання, не можуть повторно використовуватись будь-де.

На рисунку 7.2 показано, як положення об'єкта в ієрархії впливає область дії змінної. На рисунку наведено три змінні з різними областями дії; за замовчуванням у вікні *Variables* відображаються лише змінні, доступні пакету служб SSIS, виділеному на панелі конструктора. Для виведення на екран змінних із областю дії різних рівнів відкрийте діалогове вікно Variable Grid Options (Параметри сітки змінних) (див. рис. 7.1) та встановіть прапорець Show variables of all scopes (Показати змінні всіх областей).

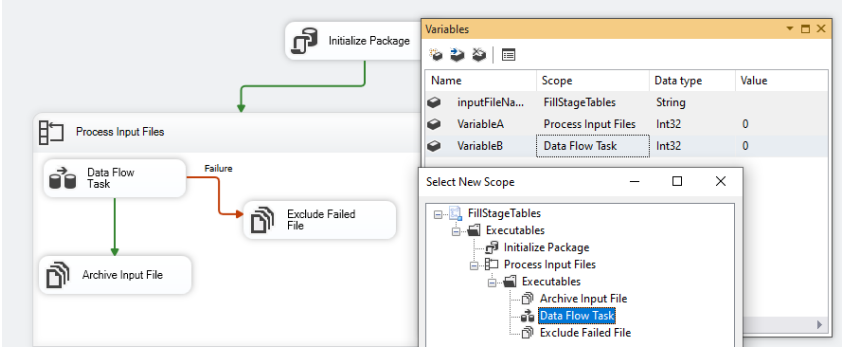

Рисунок 7.2 – Область дії змінної. Вікно Select New Scope

На рисунку 7.2 показано три змінні з різними областями дії.

– Змінна inputFileName визначена на рівні пакета і, отже, доступна всім об'єктам служб SSIS: завдання Initialize Package, контейнеру Foreach Loop з ім'ям Process Input Files, завдання

потоку даних, завдання Archive Input File та завдання Exclude Failed File.

– Змінна variableA визначена лише на рівні контейнера і, отже, доступна лише цьому контейнеру і лише його завданням. Вона не доступна в завданні Initialize Package.

– Змінна variableB визначена лише на рівні завдання і, отже, доступна в завданні потоку даних і компонентах, які містить завдання. Воно не доступне в завданнях Archive Input File та Exclude Failed File, у контейнері Process Input Files або у завданні Initialize Package.

За замовчуванням будь-яка змінна, що створюється, отримує область дії на рівні пакета. Ви можете змінити її, використовуючи команду Move (Перемістити змінну) з панелі інструментів у вікні Variables, яка відкриє вікно Select New Scope (Вибір нової області), показане на рисунку 7.2 праворуч.

Область дії змінної має ще одну корисну властивість. Імена змінних для уникнення двозначності повинні бути унікальні, але лише в межах сфери дії. Дві або кілька змінних можуть використовувати те саме ім'я до того часу, поки вони існують в різних областях дії. У разі двозначності дозволяється користуватись ступенем близькості – використовується найближча змінна в ієрархії об'єктів SSIS.

Розглянемо приклад, наведений на рисунку 7.2; ви можете додати ще одну змінну з іменем variableA в дію завдання потоку даних. Після цього змінна variableA, визначена в контейнері Process Input Files, стане недоступною для завдання потоку даних і компонентів, що містяться в ній, оскільки її перекриє змінна variableA, створена з областю дії, найближчої до завдання потоку даних, тобто її власної.

Змінні, що мають те саме ім'я, але різні області дії, можуть навіть використовувати різні типи даних.

Параметризація властивостей

Параметризація властивостей, що дозволяє задавати конкретні властивості об'єктів SSIS динамічно, може бути реалізована кількома способами.

– Властивості об’єкта SSIS, пов’язані з параметрами, можуть установлюватися динамічно із середовища, що викликає, навіть після розгортання.

– Явне присвоєння значень властивостей із змінних доступне в деяких завданнях та компонентах, що полегшує найпоширеніші випадки вживання та спрощує розгортання. Наприклад, завдання Execute SQL Task підтримує три методи присвоєння значення властивості SQLStatement:

o безпосереднє введення, у разі інструкція SQL – заздалегідь певна константа;

o підключення файлу, інструкція SQL вилучається із вмісту файлу;

o змінна, інструкція SQL надається із змінної.

– Присвоєння із виразів доступне для більшості властивостей об’єктів SSIS, дозволяючи обчислювати значення властивостей за допомогою комбінування констант, змінних, параметрів або функцій виразу.

Диспетчери з’єднань, завдання та вираження елементів керування черговістю

Зазвичай у рішенні SSIS повинні бути параметризовані наступні об’єкти служб SSIS.

– **Диспетчери з’єднань**. Під час розроблення пакета SSIS підключається до сховищ даних у середовищі розробки або тестового середовища, а розгорнутий пакет підключатиметься до виробничих сховищ даних. Задані диспетчери з’єднань також повинні налаштовуватися ітеративно під час виконання.

– **Завдання та компоненти**. Залежно від динамічно визначених налаштувань, якщо операція залежить від значень, що задаються за допомогою додаткових програмних алгоритмів (або на початку виконання, або в кожній ітерації контейнера з циклом), ви можете використовувати завдання Expression Task або завдання Script Task для обчислення значень та збереження їх у змінній SSIS, яка потім буде передавати відповідному завданню або компоненті. До прикладів можна віднести завдання Execute SQL Task або компонент джерела даних, що

використовують динамічно створювані запити, зі змінним набором параметрів.

– **Завдання потоку даних**. Ви можете налаштувати завдання потоку даних, дозволивши їм враховувати поточний стан робочого середовища. Великі переміщення даних – зазвичай ресурсомісткі процеси. Отже, для того щоб вони не вийшли за межі наявних ресурсів, ви можете відрегулювати їх поведінку під час виконання відповідно до реальної наявності ресурсів, застосувавши відповідний програмний алгоритм (за допомогою завдання Expression Task або завдання Script Task), наприклад, установивши розмір пакета або максимальний розмір кешу для завдання потоку даних, зважаючи на доступну системну пам'ять та ресурси системи введення / виведення.

Вирази

Вираз – це комбінація констант, змінних, параметрів, посилань на стовпці, функцій виразів та/або операцій виразів, що дозволяє задати під час розроблення, як визначатиметься конкретне значення під час виконання. Вирази застосовуються для динамічного визначення значень в автоматичному процесі замість призначення значень вручну та завчасного використання констант.

Вирази записують спеціальною мовою виразів, власною мовою служб SSIS. Ця мова використовує синтаксис, подібний на синтаксис мов програмування C++ та C#, він застосовує певний набір операцій, наведених у мануалі до програмного середовища.

Елементарні обчислення можна виконувати за допомогою операцій: додавання, віднімання, множення та поділу для числових даних та конкатенації або злиття для рядків. Порівняння значень також підтримується операціями порівняння; ці операції повертають логічне значення. Деякі завдання та компоненти застосовують логічні функції для визначення своїх операцій (наприклад, перенаправлення рядків із потоку даних на різні виходи залежно від результату заданого порівняння).

Логічні вирази можуть застосовуватися в елементах керування черговістю, щоб розширити вбудовані функціональні можливості, використовувані для визначення потоку керування, що базується лише на успішному, помилковому або просто завершенні попереднього завдання.

Математичні функції виконують обчислення числових значень та повертають числові результати. За допомогою операцій реалізуються основні обчислення, а математичні функції виконують обчислення, які важко чи неможливо виконати лише за допомогою операцій.

Рядкові функції виконують операції над рядковими та шістнадцятковими значеннями та повертають рядкові або числові результати. У мові виразів є типові рядкові функції, але для більш складних маніпуляцій із рядками вам, можливо, доведеться вдатися залежно від ступеня складності, необхідної в конкретному випадку, до завдань або компонентів скриптів або до завдань користувача або компонентів.

Функції дати та часу виконують операції над значеннями дат і часу та повертають дату й час, рядок або числові результати. Вимірювання з даними дат та часу відіграють важливу роль у сховищах даних, і ці функції можуть застосовуватися для спрощення їх обслуговування.

Застосовуйте ці функції для тестування значень NULL або передавання їх споживачам (завдання або компоненту). Хоча за замовчуванням значення NULL не пов’язані з будь-яким типом, споживач може розраховувати на одержання значень певних типів, тому функція NULL приймає аргумент, що вказує тип даних результату.

Вирази властивостей

Вирази доступні у властивостях більшості об’єктів служб SSIS і можуть бути встановлені за допомогою редакторів об’єктів або вікна властивостей об’єкта. Хоча більшість об’єктів SSIS застосування виразів у властивостях зовсім необов’язково, є компоненти (наприклад, компонент Derived Column (Виробничий) стовпець)), у яких висловлювання обов’язкові, без них компонент не можна налаштувати.

Зазвичай вирази використовують для налаштування певних властивостей SSIS, що залежать від даних та умов, не відомих до етапу виконання. Якщо залежність конкретної властивості від однієї чи кількох інших властивостей можна описати детерміновано, значення залежної властивості не потрібно вгадувати під час розроблення, його потрібно обчислити під час виконання.

Якщо результат виразу може використовуватися для завдання налаштувань у кількох об'єктах служб SSIS, можна скористатися завданням Expression Task (Вираз) для одноразового обчислення результату та збереження його в змінній, яка в подальшому зможе використовуватися багаторазово, усуваючи необхідність повторення тих самих обчислень.

Під час розроблення виникає помилка перевірки, якщо результат обчислення виразу відрізняється від очікуваного (бо одне або кілька значень, що впливають на результат, невідомі або не внесені). Для того щоб перешкодити цьому, можна задати значення за замовчуванням або відкласти перевірку до часу виконання, коли необхідні значення вже будуть доступні.

Вирази в елементах керування черговістю

Вирази надають спосіб розширення функціональності елементів керування черговістю за допомогою додаткових умов, які знижують можливу жорсткість застосування лише трьох основних варіантів. Завдяки цьому розширенню умови виконання пакета служб SSIS можна зробити суворішими (наприклад, застосувавши правило «Constraint and Expression» (Обмеження та вираз) або менш суворими (наприклад, за допомогою правила «Constraint or Expression» (Обмеження або вираз)). Крім того, вираз можна застосувати навіть замість обмеження, повністю усунувши з умови успіх (або невдачу) попереднього завдання.

Застосування головного пакета в удосконаленому потоці керування

Головний пакет (master package) – це не особливий тип пакета служб SSIS, це концепція чи підхід до розроблення, розгортання, обслуговування та функціонування у службах SSIS. Реалізація цієї концепції, що використовує переваги стандартних вбудованих функціональних можливостей SSIS, ґрунтується на завданні Execute Package Task (Виконання пакета). Окремі операції або процеси можуть розподілятися між різними пакетами служб SSIS, але для формування центрального потоку керування та спільного використання централізованої настройки окремих пакетів застосовується єдиний пакет, який називають головним.

Повний розв'язок для сховищ даних зазвичай містить безліч різнорідних операцій:

– Вилучення даних. Під час роботи з великими обсягами даних застосування проміжної області знижує вплив на систему-джерело операцій переміщення даних, що передаються в проміжне сховище даних за допомогою простих переміщень, без будь-яких перетворень.

– Підготовка проміжної галузі. Оскільки обсяг даних зростає, ресурси потребують проміжного сховища даних. Для зменшення впливу процесу обслуговування області проміжного зберігання на інші процеси або навіть його повного усунення, можна організувати обслуговування проміжної області як виділений самостійний процес, а не як частина будь-якого іншого процесу.

– Перетворення даних. У процесах перетворень проміжне сховище даних відіграє роль джерела даних, знижуючи залежність цих перетворень від вилучення даних. Проміжне сховище даних може бути тимчасовим місцем зберігання для складних перетворень, які можуть повністю виконуватися в оперативній пам'яті, а також може стати й тимчасовим призначенням даних доти, доки вони переміщені в кінцеве місце призначення.

– Обслуговування сховищ даних. Перебудова та реорганізація індексів, оновлення статистики та операції очищення – це типові операції з обслуговування сховищ даних. Зрозуміло, що вони повинні виконуватися як компонент будь-якого іншого процесу.

– Завантаження даних. Якщо проміжне сховище даних забезпечує тимчасове місце розміщення для даних, то кінцеве пересилання даних може виконуватися як просте переміщення даних, знижуючи вплив на сховище призначення і зменшуючи, якщо не усуваючи, залежність процесів завантаження від процесів перетворень.

Не всі операції завантаження однакові – завантаження в таблиці фактів відрізняються від завантажень у таблиці вимірювань, і дані керуються по-різному залежно від способу поводження з хронологічними даними (наприклад, вимірювання, що повільно змінюються); завантаження даних фактів не потрібно розробляти, розгортати та обслуговувати разом із завантаженнями даних вимірів та завантаження різних вимірів не повинні розроблятися, розгортатися та обслуговуватися разом, незважаючи на те, що вони зазвичай виконуються як єдине ціле.

– Багатовимірне оброблення. Перш ніж дані можуть бути переміщені в багатовимірне сховище даних (наприклад, базу даних служб SQL Server Analysis Services (SSAS)), повинні стати стабільними, тобто всі попередні процеси (вилучення-перетворення-завантаження) повинні завершитися успішно і повністю, насамперед що буде виконано переміщення. З цього цілком очевидно випливає, що багатовимірне оброблення не повинне виконуватися як частина будь-якого іншого процесу.

Тестові питання

А. Установіть відповідність між змінними SSIS та їх призначення

	Збереження властивостей об’єктів SSIS	Збереження даних	Збереження «процесних» даних
Системні	☐	☐	☐

змінні			
Змінні користувача	☐	☐	☐

B. Оберіть усі властивості змінних SSIS:

☐ збереження автоматично визначених значень;

☐ збереження визначених користувачем значень;

☐ збереження значень, визначених у процесі виконання збереження наперед визначених значень;

☐ використання в одному розв'язку;

☐ використання в багатьох розв'язках.

C. На якому (-их) рівні (-ях) доступні глобальні змінні:

☐ рівні пакетів;

☐ рівні контейнерів;

☐ рівні завдань?

D. У вашому пакеті SSIS вам потрібно витягти скалярне значення з таблиці в базі призначення, щоб використовувати його в кількох завданнях. Який спосіб підходить для цього найбільше:

☐ вбудувати підзапит у кожен наявний запит, застосовуваний пакетом, щоб підсистема керування базою даних могла підготувати найбільш відповідний план виконання для вилучення значення під час виконання;

☐ створити змінну і використовувати вираз для одноразового вилучення значення з бази даних, а потім використовувати його впродовж часу виконання;

☐ створити змінну і використовувати завдання Execute SQL Task для одноразового вилучення значення, а потім використовувати його впродовж часу виконання;

☐ створити змінну і використовувати завдання Expression для багаторазового вилучення значення з бази даних у міру необхідності?

E. У вашому пакеті SSIS ви створили змінну на рівні пакета для зберігання значення, яке хочете багаторазово застосовувати в пакеті. Пізніше ви з'ясували, що це значення в одному контейнері має задаватися інакше, але на вихідну змінну це впливати не повинно. Що ви можете зробити:

☐ створити нову змінну на рівні пакета з іншим ім’ям і переналаштувати завдання відповідно для використання нової змінної або первісної;

☐ створити нову змінну на рівні контейнера з іншим ім’ям і переналаштувати на використання нової змінної лише ті завдання, які її містять;

☐ створити нову змінну на рівні контейнера з тим самим ім’ям і залишити завдання без змін;

☐ використовувати параметр рівня пакета, тому що цю проблему не вирішити за допомогою змінних?

F. У вашому процесі SSIS конкретна властивість повинна визначатися у виробничому середовищі адміністратором. Значення, що задається адміністратором, буде використовуватися в різних властивостях і повинно перевизначатися автоматично, якщо під час виконання виникнуть певні умови. Який є найбільш відповідний метод у службах SSIS, що дає можливість домогтися бажаного:

☐ створити параметр і використовувати вирази для присвоєння його значення відповідним властивостям, але застосовувати вираз на початку виконання, щоб якщо потрібно, змінити значення параметра;

☐ створити параметр, на початку виконання, застосувати вирази для присвоювання його значення змінній або перевизначення значення за потреби і потім застосовувати вирази для присвоювання значення змінної різним властивостям;

☐ створити змінну для читання / запису, використати вирази для присвоювання її значення відповідним властивостям і під час виконання присвоїти коректне значення змінній через шляхи до властивостей;

☐ створити параметр і використовувати вирази, щоб присвоїти його значення властивості або, якщо потрібно, перевизначити її?

G. У вашому пакеті SSIS є кілька завдань потоку даних, кожна з яких імпортує дані в базу призначення. Ви повинні записати в журнал кількість рядків, які були вставлені або

оновлені в кожному потоці даних. Які функціональні можливості, що надаються службами SSIS, можна використати для виконання цього завдання (Виберіть усі відповідні варіанти.):

☐ можна використовувати завдання Row Count Task (Підрахунок рядків) для обчислення кількості рядків, що пройшли через потік даних;

☐ можна застосувати компонент Row Count (Лічильник рядків) для обчислення кількості рядків, що пройшли через потік даних;

☐ можна зберігати значення в змінних перед збереженням їх у журналі;

☐ можна використовувати вираз для обчислення загальної кількості оброблених рядків?

H. У вашому пакеті SSIS необхідно задати властивості кількох завдань, ґрунтуючись на інформації, доступній у середовищі виконання. Кожне значення властивостей, які потрібно обчислити, можна підрахувати за допомогою математичного виразу. Який метод буде найбільше підходити:

☐ для створення і тестування виразу застосувати вікно Expression Builder (Побудовник виразів), а потім скопіювати вираз в усі визначення відповідних завдань;

☐ помістити в потік керування завдання Expression Task (Завдання виразу), що передує кожному завданні, властивості якого повинні визначатися динамічно;

☐ застосувати стільки завдань Expression Task, скільки потрібно змінних, що зберігають результати обчислення різних виразів, а потім використовувати ці змінні для присвоєння значень властивостям відповідних завдань;

☐ застосувати одне завдання Expression Task і зберігати всі необхідні обчислені значення у змінній набору рядків (типу Object), яка повинна використовуватися у виразах властивостей для налаштування відповідних завдань?

I. У потік керування вашого пакета SSIS необхідно включити завдання обслуговування, яке буде перебудовувати індекси ваших таблиць вимірювань після їх успішного

заповнення. Ви реалізували завантаження кожної таблиці вимірів в окремому потоці даних. Завдання Execute SQL Task, що містить скрипт перебудови індексу, повинне виконуватися після успішного завершення попереднього завдання потоку даних, але лише по суботах. Назва дня тижня для поточного дня зберігається у змінній. Чи можна отримати до неї доступ у потоці керування SSIS? Якщо так, то як:

☐ ні, це неможливо в потоці керування, тому що перетворення умовного розбиття доступне лише в потоці даних;

☐ так, до неї можна отримати доступ у потоці керування, але лише за допомогою завдання Script Task;

☐ так, цього можна домогтися, застосувавши провідний від завдання потоку даних до завдання Execute SQL Task елемент керування черговістю з перевіркою успішного завершення і виразом, що перевіряє, чи дорівнює значення змінної Saturday (субота);

☐ так, цього можна досягти, застосувавши звичайний елемент керування черговістю з перевіркою успішного завершення, що веде від завдання потоку даних до завдання Execute SQL Task?

J. Які методи служби SSIS надають для налаштування дочірніх пакетів із головного пакета:

☐ параметри проєкту;

☐ глобальні змінні;

☐ параметри пакета;

☐ параметри процесу?

K. Вирази можуть застосовуватися в елементах керування черговістю для (правильні одна або кілька відповідей):

☐ заміни елементів керування черговістю;

☐ вирішення конфліктів між декількома елементами керування черговістю;

☐ додавання умов до елементів керування черговістю;

☐ визначення змінних.

L. Які з пропозицій, що стосуються головного пакета, справедливі (Виберіть усі відповідні варіанти.):

☐ головні пакети застосовуються для налаштування дочірніх пакетів;

☐ головні пакети використовуються для виконання дочірніх пакетів;

☐ головні пакети не дозволяють дочірнім пакетам виконуватися безпосередньо;

☐ головні пакети використовуються для керування порядком виконання дочірніх пакетів?

Аналітичні питання

Завдання 1. Повні розв'язки.

Керівник запропонував розробити повний розв'язок для зберігання даних – процес ETL, багатомірне оброблення та навіть кешування звітів. Він запропонував використати єдиний пакет служб SSIS для цього конкретного розв'язку.

Керівник надав такі висновки за допомогою своєї пропозиції.

– застосування єдиного пакета SSIS сприяє розвертанню та обслуговуванню;

– розміщення всіх елементів в єдиному пакеті дозволить також швидше виконувати будь-які доробки та вдосконалення пакета;

– застосування одного пакета SSIS;

– єдиний спосіб гарантувати правильний порядок виконання столу великої кількості операцій;

– розроблення в службах SSIS не вимагає розподілу праці; розроблення всіх необхідних завдань може виконуватися одним розробником;

– змінні з одного пакета недоступні в інших пакетах, відповідно єдиний пакет надає відповідні можливості багаторазового використання змінних.

1.1. Як би ви віднеслися до цієї пропозиції?

1.2. Чи погоджуєтеся ви з доводами керівника?

Завдання 2. Виконання, що керується даними.

Перед вами поставлене завдання консолідації наявних розв'язків сховища даних, які в даний час реалізовані як окремі

пакети SSIS, що виконують вилучення, перетворення та завантаження даних. Частина розв'язків залежить від двох зовнішніх служб, що надають дані з віддалених сховищ даних, з попереднім їх збереженням у файлах вашої локальної мережі.

Усе оброблення даних повинне виконуватися в нічні періоди обслуговування. Ваша система взаємодіє із зовнішніми службами за допомогою спеціальних засобів, установлених локально. Ці служби не цілком надійні – часом одна з них або обидві не відповідають, але дані, які вони надають, дуже важливі для вашої компанії. Якщо яка-небудь служба дає збій під час формування потрібних файлів, вам хотілося б повідомити про це адміністраторам віддаленої машини для усунення перешкод.

Усі доступні дані все одно повинні бути оброблені та завантажені у сховище даних, але якщо зовнішні дані не доставлені або доставлені частково, зміни не повинні бути передані в базу даних служби Analysis Services, яка використовується для звітів. Керівники середньої ланки вашої компанії віддають перевагу повним аналітичним даним і готові чекати декілька днів, поки дані не будуть оновлені до кінця.

Дайте відповіді на такі питання:

2.1. Які концепції сховища даних можна застосувати для вирішення ваших завдань?

2.2. Які розв'язки, що надаються набором засобів SSIS, ви можете використовувати?

2.3. Як поміщати завантаження неповних даних у базу даних служби SSAS?

МАТЕРІАЛ ДЛЯ ПІДГОТОВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ 8

Теоретичний матеріал

Удосконалення потоку даних. Виміри, що повільно змінюються. Визначення типів атрибутів

Під час підготовки до завантаження таблиць вимірів необхідно визначити для кожного атрибута, що відбудеться в разі вставлення нових значень. Якщо є потреба відстежувати хронологію або передісторію хоча б для одного атрибуту, то необхідно додати два додаткові стовпці, що відображають інтервал валідності (дійсності) значення. Тип даних цих стовпців повинен бути Date або DateTime, і стовпці повинні містити значення Valid From (Діюча с) і Valid To (Діюче до). Для поточного значення у стовпці Valid To повинне бути значення NULL. У процесі моделювання та зіставлення вихідних даних із вашим сховищем даних дуже корисно розділити атрибути на різні типи.

– *Бізнес-ключ.* Атрибут цього типу – частина бізнес-ключа з даних-джерела.

– *Атрибут незмінності* (Fixed Attribute). Значення атрибуту цього типу ніколи не повинно змінюватись у сховищі даних.

– *Атрибут, що змінюється* (Changing Attribute або Type 1 SCD). Атрибути цього типу перезаписують хронологію атрибута.

– *Атрибут із передісторією* (Historical Attribute або Type 2 SCD). Атрибути цього типу зберігають свою хронологію.

– На основі цієї інформації можна розпочати реалізацію пакета служб SSIS для оновлення вимірів. У тих випадках, коли є потреба реалізувати варіант Type 2 SCD, для виміру знадобиться новий ключ сховища даних. Цей ключ називають сурогатним ключем.

Елементи вимірів, що виводяться

У сценаріях сховищ даних можлива ситуація, в якій записи з таблиці фактів надходять із вихідним бізнес-ключем, який ще не завантажений у вимір. На жаргоні СД ця проблема відома ***як виміри, які пізно прибувають (або запізнювальні), або таблиці фактів, що рано прибули***. Якщо до таблиці фактів вставляються записи, для яких немає відповідних сурогатних ключів у вимірі, типовим буде такий розв'язок:

1. Вставити до таблиці виміру рядок, використовуючи ключ системи-джерела, і призначити сурогатний ключ цього елемента. Позначити цей запис для ілюстрації, що він був вставлений як елемент, що виводиться, за допомогою додаткового стовпця у вимірі (наприклад, додавши стовпець InferredMember і надавши йому значення True).

2. Застосувати новостворений сурогатний ключ і надати його запису з таблиці фактів.

3. Змінити процес завантаження виміру для перевірки того, чи вставлено запис як елемент, що виводиться. У цьому випадку необхідно обробляти всі атрибути як атрибути типу Type 1 SCD незалежно від їхньої специфікації, за винятком сурогатного ключа та бізнес-ключа. Якщо цього не зробити, атрибути типу Type 2 SCD створять новий рядок, тому що у рядку, що виводиться, немає значення для цього атрибута. Таким чином, процес завантаження складається з оновлення рядка та присвоєння стовпцю InferredMember значення False.

Якщо не потрібно підтримувати елементи, що виводяться, все одно корисно мати процес, що присвоює у вимірі стандартне значення сурогатного ключа (наприклад, –1) рядкам, для яких немає збігів. Агрегатні значення з СД і вихідних систем збігатимуться, тому що в таблицю фактів завантажені всі дані, але пізніше потрібно буде перезавантажити або оновити дані, коли у вимірі з'явиться відповідний рядок.

Застосування завдання Slowly Changing Dimension

У службах SSIS є вбудоване перетворення Slowly Changing Dimension (SCD), яке проведе через всі етапи процесу створення алгоритму для вставлення та оновлення даних у

таблиці вимірів. Спершу потрібно створити у пакеті хоча б одне завдання потоку даних та визначити один адаптер джерела потоку даних.

Далі перераховані кроки, необхідні для реалізації перетворення SCD.

1. З панелі SSIS Toolbox перетягнути мишкою завдання Slowly Changing Dimension на робоче поле конструктора потоку даних.

2. З'єднайте наявний потік даних із новим завданням Slowly Changing Dimension. На рисунку 8.1 показане робоче поле конструктора потоку даних із новим завданням.

3. Двічі клацнути кнопкою мишки на завданні Slowly Changing Dimension для відкриття Slowly Changing Dimension Wizard (Майстер виміру, що повільно змінюється).

4. На сторінці привітання майстра натиснути кнопку Next (Далі).

5. Установити диспетчер з'єднань та вибрати кінцеву таблицю виміру.

6. На основі цієї інформації майстер перерахує всі стовпці з таблиці виміру (рис. 8.2). Тепер необхідно задати порівняння стовпців даних джерела зі стовпцями виміру. Робиться це вибором відповідних стовпців у графі Input Columns (Вхідні стовпці) кожного стовпця призначення. Далі потрібно зазначити, який стовпець або які стовпці є частиною джерела бізнес-ключа. Для цього в стовпці Key Type (Тип ключа) задається значення Business key.

7. На наступній сторінці для стовпців виміру визначається тип змін. Можливі варіанти: Fixed attribute, Changing attribute (Type 1 SCD) або Historical attribute (Type 2 SCD). Зробити це можна для стовпців, порівняних зі стовпцями джерела у попередньому пункті.

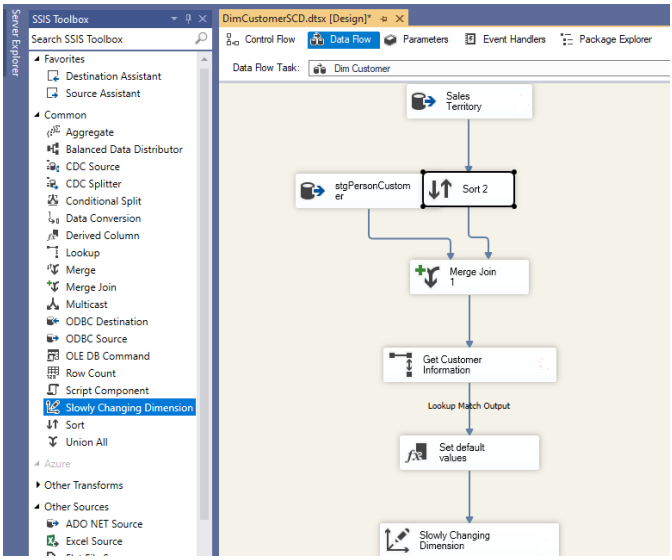

Рисунок 8.1 – Робоче поле конструктора потоку даних
із завданням Slowly Changing Dimension

8. На наступній сторінці під рядком Fixed and Changing Attribute Options (Параметри атрибутів незмінності та атрибутів, що змінюються) можна задати, чи буде перетворення завершуватися з помилкою, якщо виявляться зміни в атрибуті незмінності, і чи оновлюватимуться атрибути типу Type 2 SCD у всіх відповідних записах.

9. Якщо в п. 7 був заданий хоча б один стовпець із типом Historical attribute (Type 2 SCD), на наступній сторінці потрібно задати, як відстежувати його передісторію таблиці виміру. Існують два варіанти:

– Використовувати спільний стовпець для виведення поточних та застарілих записів. У цьому разі задається один стовпець у таблиці виміру, щоб показати, який рядок на даний момент активний. Під час вибору цього варіанта потрібно задати ознаку поточного та застарілого запису (наприклад, True / False або Current / Expired (Поточна / Застаріла)).

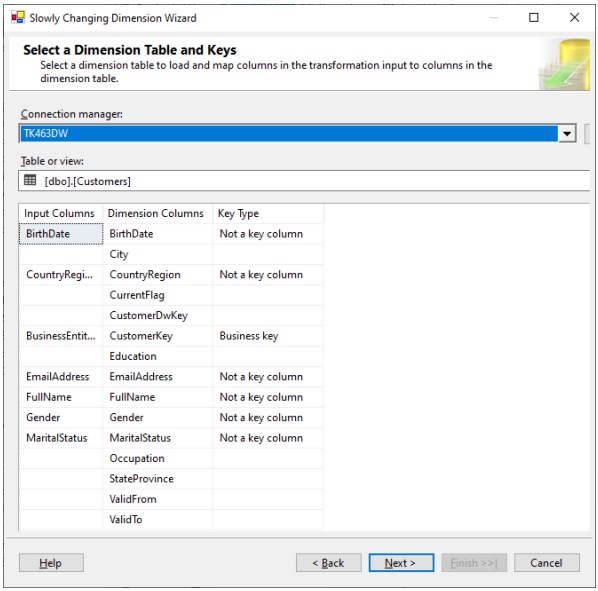

Рисунок 8.2 – Вибір таблиці виміру та ключів у майстрі Slowly Changing Dimension Wizard

– Використовувати дані про дату початку та закінчення подій для ідентифікації поточних та застарілих записів. Під час вибору цього варіанта задається два стовпці в таблиці виміру для відображення дійсності кожного запису. Також задається змінна служб SSIS для вказівки конкретної дати (рис. 8.3). Відомості про дату застосовуються для встановлення значення ValidTo (Діюче до) у попередній версії рядка та значення ValidFrom (Діюче з) у поточній версії рядка.

11. На наступній сторінці майстра Slowly Changing Dimension Wizard можна задати підтримку виведених елементів виміру, встановивши прапорець Enable inferred member support (Увімкнути підтримку виведених елементів вимірювання). Далі зазначити, як позначити рядки, які потрібно оновити, тому що вони були вставлені після завантаження таблиці фактів. Є два варіанти: оновлювати, якщо всі атрибути, позначені вами як змінювані, мають значення NULL, або оновлювати, проаналізувавши додатковий стовпець вашої таблиці вимірювання, що має значення True/False.

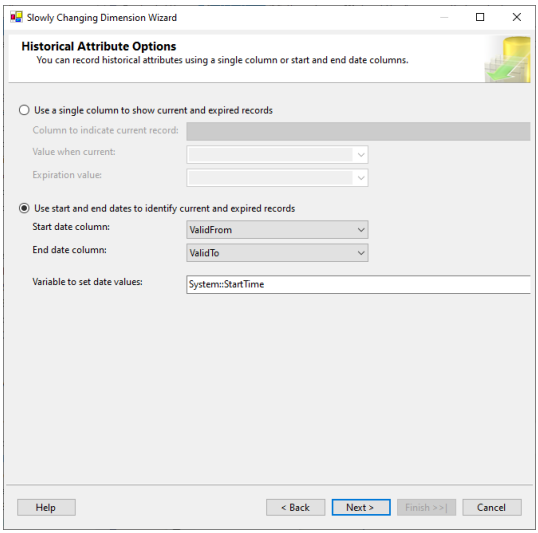

Рисунок 8.3 – Налаштування змінної

12. На останній сторінці можна переглянути конфігурацію.

Після натискання кнопки Finish майстер Slowly Changing Dimension Wizard створить відповідні перетворення у потоці даних для підтримки потрібного алгоритму оновлення вимірювання. Він використовує перетворення Derived Column (Похідний стовпець), OLE DB Command (Команда OLE DB) і Union All (Об’єднати всі) та адаптер місця призначення даних Data Destination (рис. 8.4).

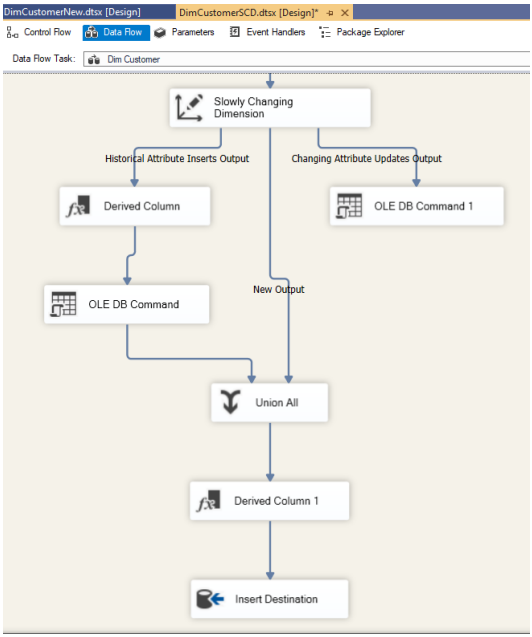

Рисунок 8.4 – Перетворення, створені автоматично майстром Slowly Changing Dimension Wizard

Перетворення Slowly Changing Dimension створює до шести виходів, ґрунтуючись на вимогах до оновлення та вставлення рядків. У таблиці 8.1 перелічено всі можливі виходи.

Виміри, що ефективно оновлюються

Незважаючи на те, що перетворення Slowly Changing Dimension дуже легко використовувати і можна швидко створити цілком процес потоку даних для поновлення вимірів, врахуйте, що з погляду продуктивності воно діє далеко не оптимально, тому його варто застосовувати лише до невеликих вимірів (не більше ніж 10 000 рядків).

Якщо подивитися на перетворення, які використовує майстер, то можна помітити, що він застосовує перетворення OLE DB Command, яке діє на рівні рядка і виконує інструкцію T-SQL UPDATE для кожної зміни (і Type 1, і Type 2 SCD). Це дуже марнотратно з погляду продуктивності й у вимірах великого обсягу може стати зовсім марним. Тому в цьому

розділі розглядається альтернативний підхід до оновлення вимірів засобами служб SSIS.

Таблиця 8.1 – Виходи перетворення Slowly Changing Dimension

Вихід	**Призначення**
Changing Attributes Updates output (Вихідні дані оновлень змінюваних змінних атрибутів)	Цей вихід застосовується до рядків з атрибутами Type 1 SCD
Fixed Attribute output (Вихід атрибута незмінності)	Цей вихід застосовується до рядків з атрибутами незмінності
Historical Attributes Inserts output (Вихід вставок атрибутів із передісторією)	Цей вихід застосовується до рядків з атрибутами Type 2 SCD
Inferred Member Updates output (Вихід оновлень виведених елементів)	Цей вихід складається з усіх рядків, які повинні оновити рядки у вивідних елементах таблиці вимірювань
New output (Новий вихід)	Цей вихід складається з усіх нових рядків
Unchanged output (Незмінений вихід)	Цей вихід застосовується для рядків, що не змінилися

Для оновлення вимірів з атрибутами Type 1 і Type 2 SCD застосовується такий алгоритм:

1. Перевірити, чи були змінені атрибути рядка-джерела.
2. Позначити рядки зі змінами Type 1 або Type 2 SCD залежно від специфікації, заданої для виміру.
3. Застосувати до виміру необхідний алгоритм оновлення:

– у разі зміни атрибутів Type 1 SCD поновіть рядок;

– у разі зміни атрибутів Type 2 SCD поновіть термін дії поточного рядка, зазначивши, що він був дійсний до цього моменту, і вставте новий рядок.

Перевірка змін атрибутів

У разі порівняння значення атрибутів у вихідному рядку і рядку призначення насамперед потрібно перевірити, чи є вже у вашому вимірі бізнес-ключ, наприклад за допомогою перетворення Lookup (Уточнюючий запит). Після цього усі нові рядки повинні потрапити у вихід рядків, що не збігаються (no match) уточнювального запиту, і ви зможете вставити їх у вимір. Набагато складніше перевірити вже наявні рядки, щоб з'ясувати, чи змінилися в них атрибути і який тип змін потрібно застосувати до них надалі.

У реальному житті у вимірі зазвичай багато стовпців, і порівнювати їх усі невигідно. Тому в процесах ETL зазвичай застосовується хеш-функція для зберігання інформації про всі потрібні атрибути, щоб процес міг швидко встановити, чи вносилася зміна. *Хеш-функція* – це будь-який алгоритм або підпрограма, що відображає великі набори даних змінної довжини, названі ключами, на набори даних меншого обсягу з фіксованою довжиною. У цьому разі механізм хешування повинен бути детермінованим (тобто функція завжди повинна повертати один і той самий результат для певного набору вхідних значень, хоч би коли її було викликано) і повинна володіти дуже низькою ймовірністю відображення двох або декількох ключів на одне й те саме хеш-значення. Один із найпопулярніших алгоритмів – криптографічна хеш-функція MD5 Message-Digest Algorithm, що формує 128-бітове (16-байтне) хеш-значення.

Для реалізації хеш-функції в службах SSIS у вас є кілька можливостей:

– написати інструкцію SELECT і вставити обчислювальний стовпець, застосувавши функцію T-SQL HASHBYTES під час читання даних;

– застосувати перетворення Script Component і використовувати платформу Microsoft .NET Framework для обчислення необхідного хеш-значення;

– розгорнути користувацький компонент служб SSIS, який обчислює хеш-значення, що базується на інших алгоритмах.

Звичайно, потрібно доповнити свій вимір одним або кількома стовпцями для зберігання хеш-значень. Якщо є багато стовпців і половина з них повинна розглядатися як атрибути типу Type 2 SCD, також можна додати два хеш-значення: одне – для всіх стовпців, та інше – для стовпців типу Type 2 SCD.

Таким чином, можна перевіряти, чи було змінено рядок і чи потребує він врахування зміни Type 2 SCD.

Алгоритм оновлення на основі наборів даних

Одна з ключових причин повільної роботи перетворення Slowly Changing Dimension – виконання SQL-оновлень на рівні рядка. Ось чому так важливо реалізувати алгоритм на основі наборів даних, коли проєктують процеси SSIS, змушені враховувати зміни Type 1 або Type 2 SCD.

У таблиці 8.2 перелічено поточні дані сховища даних для виміру Customer. Стовпець City – стовпець типу Type 2 SCD, а стовпці FullName і Occupation – типу Type 1 SCD.

Таблиця 8.2 – Поточні дані СД для вимірювання Customer

DWciD	CustomeriD	FullName	City	Occupation	ValidFrom	ValidTo
289	17	Bostjan Strazar	Vienna	Professional	1.1.1900	NULL

Для внесення зміни Type 1 або Type 2 SCD необхідно зберегти всі рядки, які було змінено, у додатковій тимчасовій таблиці в потоці даних і додати стовпці, що вказують, яку зміну враховують, Type 1 або Type 2 SCD. У таблиці 8.3 показано лише змінені вихідні дані після завершального кроку в потоці даних.

Таблиця 8.3 – Змінені вихідні дані, збережені в додатковій тимчасовій таблиці

CustomeriD	FullName	City	Occupation	Type1	Type2
17	Bostjan Strazar	Ljubljana	Teacher	True	True

Далі за допомогою завдання Execute SQL Task у потоці керування необхідно виконати такі інструкції SQL.

1. Оновити рядки виміру, що мають зміни Type 1 SCD (у яких стовпець Type 1 має значення True), ґрунтуючись на бізнес-ключі джерела, у цьому прикладі – стовпець CustomerID. У таблиці 8.4 показано результат у сховищі даних після цього кроку.

Таблиця 8.4 – Дані СД із виміру Customer після внесення зміни Type 1 SCD

DWciD	CustomeriD	FullName	City	Occupation	ValidFrom	ValidTo
289	17	Bostjan Strazar	Vienna	Teacher	1.1.1900	NULL

2. Оновити стовпець ValidTo у рядках виміру, що мають зміну типу Type 2 SCD (у яких у стовпці Type2 значення True) значенням поточної дати (у цьому прикладі 1.1.2012). У таблиці 8.5 показано результат у сховищі даних після цього кроку.

Таблиця 8.5 – Дані СД із виміру Customer з оновленим стовпчиком ValidTo

DWciD	CustomeriD	FullName	City	Occupation	ValidFrom	ValidTo
289	17	Bostjan Strazar	Vienna	Teacher	1.1.1900	1.1.2012

3. Вставити нові рядки, що мають зміни Type 2 SCD, і задайте в стовпчику ValidTo значення NULL. У таблиці 8.6 показано результат у сховищі даних після цього кроку.

Таблиця 8.6 – Остаточні дані СД у вимірі Customer

DWciD	CustomeriD	FullName	City	Occupation	ValidFrom	ValidTo
289	17	Bostjan Strazar	Vienna	Teacher	1.1.1900	1.1.2012
943	17	Bostjan Strazar	Ljubljana	Teacher	1.1.2012	NULL

Застосування динамічного SQL для читання даних

Припустимо, що є таблиця-джерело зі стовпцем, який можна використовувати для фільтрації рядків конкретного вікна додаткового завантаження. Наприклад, у таблиці-джерелі є стовпець OrderDate типу Date, і є потреба у пакеті SSIS читати лише ті рядки, в яких дата замовлення (order date) пізніша, ніж конкретне значення, що зберігається у змінній служб SSIS з ім'ям @IncrementalDate.

Існують різні способи створення і передавання динамічного запиту SQL адаптеру джерела, що залежать від типу адаптера джерела даних, який буде застосовуватися у службах SSIS.

Використання адаптера джерела OLE DB

У разі використання адаптера джерела OLE DB є два варіанти для створення і використання динамічного запиту SQL.

– *Записати команду SQL з параметром*. Можна надати інструкції SQL параметр, задавши у вікні *OLE DB Source Editor,* режим доступу до даних *SQL Command.* Якщо у вікні OLE DB Source Editor натиснути кнопку *Parameters*, відкриється діалогове вікно *Set Query Parameters*. У ньому можна зіставити заповнювачі параметрів SQL зі змінними SSIS або параметрами (рис. 8.5).

– *Використовувати команду SQL зі змінної*. У цьому випадку необхідно зберегти у змінній весь запит. Спочатку потрібно задати змінну типу String, наприклад @SQLQuery, і задати як значення фіксовану інструкцію запиту SQL, щоб можна було читати метадані під час використання змінної як команди SQL всередині OLE DB Source Editor. Якщо

використовувати дані попереднього прикладу, значення змінної повинне мати такий вигляд:

```
SELECT SalesOrderID, OrderDate, CustomerID, TaxAmt, SubTotal
FROM Sales.SalesOrderHeader;
```

Тепер потрібно зазначити, що значення змінної повинне задаватися динамічно під час кожного виконання пакета.

Це можна зробити як у потоці керування, застосувавши нове завдання Expression Task (Завдання виразу) (рис. 8.6), так і завданням виразу у вікні Variables (Змінні), що встановлює значення під час виконання.

Після визначення змінної можна зазначити в OLE DB Source Editor режим доступу до даних зі змінної як SQL Command.

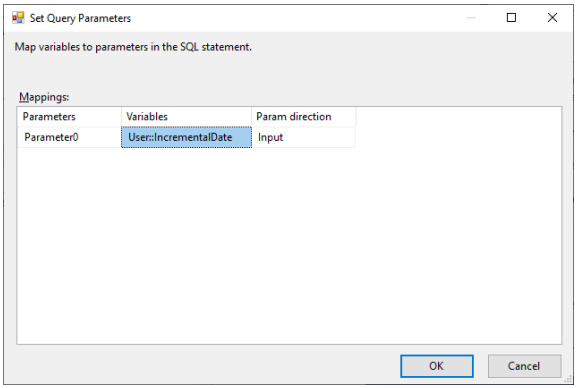

Рисунок 8.5 – Зіставлення параметрів зі змінними SSIS

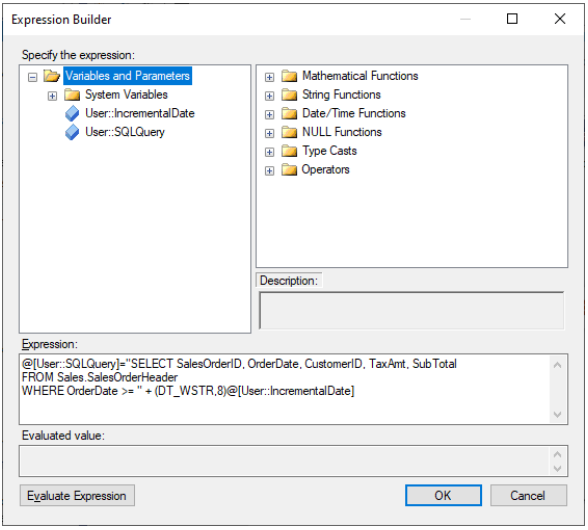

Рисунок 8.6 – Запис динамічного запиту SQL за допомогою виразу служб SSIS

Використання адаптерів джерела ODBC або ADO.NET

Якщо використовується адаптер джерела ADO.NET або новий адаптер джерела ODBC, параметризувати запит SQL можна лише поза завданням потоку даних, використовуючи властивість Expressions. Властивість Expressions завдання служб SSIS дає можливість установлювати різні властивості динамічно за допомогою обчислення під час виконання. Можна встановити деякі загальні властивості завдання потоку даних і деякі властивості конкретних перетворень потоку даних, доданих у ваш потік даних. Під час вставлення адаптера джерела ODBC або ADO.NET ці властивості видно у вікні редактора Property Expressions Editor. Як і раніше, можна задати вираз для властивості SqlCommand, і під час виконання служба SSIS надішле запит SQL, що базується на цьому виразі.

Починаючі з SQL Server 2012 є також можливість застосувати параметри SSIS для модифікації властивості SqlCommand у завданні потоку даних. *Параметри служб SSIS* дозволяють присвоювати значення властивостям у пакетах під час виконання пакетів. Можна створювати параметри проєкту

на рівні проєкту і параметри пакета – на рівні пакета. Параметри проєкту використовуються для надання зовнішнього входу для одного або декількох пакетів у проєкті. Параметри пакета дають можливість змінювати виконання пакета без його редагування та повторного розгортання. Якщо клацнути правою кнопкою мишки на завданні потоку даних і вибрати команду Parameterize (Параметризація), відкриється однойменне діалогове вікно. У ньому ви зможете порівняти конкретну властивість і певний параметр. Важливо пам'ятати, що якщо тут ви задасьте параметр для конкретної властивості, його також буде видно у властивості Expressions, як у попередньому випадку.

Реалізація CDC за допомогою служб SSIS

Увімкнення CDC у базі даних

Відстеження змінених даних (change data capture, CDC) надає легкий спосіб відстеження змін даних у таблицях бази даних, що дає можливість передати ці зміни іншій системі, наприклад, сховищу даних. Якщо є потреба ввімкнути засоби CDC, їх потрібно увімкнути для кожної таблиці, у якій необхідно відстежувати зміни, але спочатку потрібно увімкнути їхню підтримку для самої бази даних. Далі наведено приклад коду мовою T-SQL, який вмикає CDC для бази даних TK463DW і таблиці stg.CDCSalesOrderHeader.

```
USE TK463DW;
-- Увімкнення CDC для бази даних
EXEC sys.sp_cdc_enable_db;
-- Додавання ролі користувача для CDC
CREATE ROLE cdc_role;
-- Увімкнення CDC для таблиці
-- Переконайтеся, що виконується SQL Server Agent
EXEC sys.sp_cdc_enable_table
 @source_schema = N'stg',
 @source_name = N'CDCSalesOrderHeader',
 @role_name = N'cdc_role',
 @supports_net_changes = 1;
```

Обов'язкова умова під час увімкнення CDC – запущена служба SQL Server Agent. Під час увімкнення системи CDC сервер згенерує нову схему cdc, кілька системних таблиць і два

завдання в службі SQL Server Agent. Далі перелічено системні таблиці:

– cdc.captured_columns – список відстежуваних стовпців;

– cdc.change_tables – усі таблиці, для яких увімкнено систему CDC;

– cdc.ddl_history – історія всіх змін визначень даних (DDL) з моменту включення CDC;

– cdc.index_columns – індекси, пов'язані з таблицями CDC;

– cdc.lsn_time_mapping – застосовується для порівняння значень реєстраційного номера транзакції в журналі (LSN) і часу фіксації транзакції;

– cdc.stg_CDCSalesOrderHeader_CT – у кожної таблиці з увімкненою системою.

У CDC є «копія», в якій зберігаються всі зміни з додатковими стовпчиками, що описують, наприклад, тип зміни.

Компоненти CDC у службах SSIS

Для того щоб полегшити відстеження змінених даних, у службах SSIS введено нові компоненти.

– **Завдання CDC Control Task** (Керування CDC). Застосовується для керування життєвим циклом пакетів CDC. Це завдання керує синхронізацією пакета CDC з пакетом початкового завантаження і контролює керування діапазонами реєстраційних номерів транзакцій у журналі (log sequence number, LSN), які обробляються під час виконання пакета CDC (LSN однозначно ідентифікує кожен запис у журналі транзакцій SQL Server). Крім того, це завдання має справу зі сценаріями опрацювання помилок і відновлення та є частиною потоку керування в службах SSIS.

– **Адаптер джерела CDC**. Зчитує діапазон змінених даних із таблиць змін CDC і доставляє зміни далі в напрямку потоку іншим компонентам служб SSIS. Адаптер джерела CDC – один з адаптерів джерел потоку даних.

– **Роздільник CDC**. Ділить єдиний потік змінених рядків із компонента, джерела CDC, на різні потоки даних для операцій вставлення, оновлення та видалення. Роздільник, – по

суті, попередньо налаштоване перетворення Conditional Split (Умовне розбиття), яке автоматично обробляє стандартні значення стовпця __$operation в таблиці, що зберігає змінені дані.

– **Завдання CDC Control Task** зберігає стан пакета CDC у змінній пакета служб SSIS і може також зберегти змінну пакета SSIS у таблиці бази даних, таким чином зберігаючи стан між періодами активізації пакета або декількох пакетів, які разом виконують загальний процес CDC (наприклад, одне завдання може відповідати за початкове завантаження, а інше – за додаткові оновлення). У цієї змінної повинен бути тип даних String. Вона буде завантажуватися, ініціалізуватися й оновлюватися завданням CDC Control Task і використовуватися джерелом CDC, компонентом потоку даних, для визначення поточного діапазону оброблення змінених записів. Діапазон оброблення задається зазначеним значення у випадному списку CDC control operation (Операція керування CDC) вікна CDC Control Task Editor (Редактор завдання «Керування CDC»).

Джерело CDC пропонує п’ять різних способів вилучення змінених даних (рис. 8.7). Формат, у якому ви отримуєте дані, визначається режимом оброблення CDC, обраним у діалоговому вікні CDC Source Editor (Редактор джерела CDC).

Можливі такі варіанти.

– **All** (Усі). У цьому режимі оброблення повертатиметься рядок для кожної зміни, внесеної у вихідну таблицю. Якщо три інструкції UPDATE було застосовано до одного й того самого рядка, ви отримаєте назад три записи. Рядки будуть упорядковані за номерами LSN (тобто в порядку їх оброблення). Цей режим корисний у процесах ETL, які потребують докладного аудиту, але він ускладнює процес ETL, оскільки доведеться об’єднувати зміни самостійно, щоб отримати останню зміну для конкретного запису.

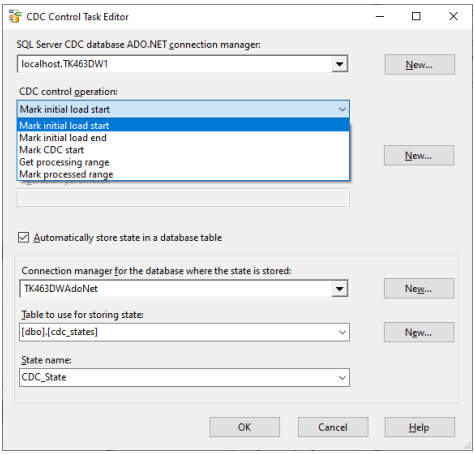

Рисунок 8.7 – Можливі варіанти режиму оброблення CDC

– **All with old values** (Усі зі старими значеннями). Цей режим оброблення аналогічний варіанту All, за винятком того, що ви отримуєте два рядки на кожну інструкцію UPDATE – один зі значеннями Before Update (До поновлення) і другий, що містить значення After Update (Після оновлення). Є й додатковий стовпець для розмежування різних оновлень.

– **Net** (Сумарні). У цьому режимі сервер CDC буде вносити всі зміни, які були зроблені в конкретному рядку, і поверне лише один рядок для кожного унікального запису. Якщо інструкція INSERT вставила рядок і потім було застосовано дві інструкції UPDATE, ви отримаєте назад один рядок, що містить усі зміни. Цей режим зручний у типових сценаріях ETL для СД.

– **Net with update mask** (Сумарні з маскою оновлення). Аналогічний режиму Net, за винятком того, що цей режим додає для кожного стовпця джерела стовпчик типу Boolean, який вказує, чи змінювалося значення стовпця. Це може бути корисно, якщо необхідно надати додаткове програмне оброблення за зміни конкретного поля.

– **Net with merge** (Сумарні зі злиттям). Аналогічний режиму Net, але рядки UPDATE і INSERT групуються разом з одним і тим самим значенням операції. Цей режим розроблено

для підвищення продуктивності запитів, коли важливо зазначити, що операція, яка вносить зміни в дані, – це вставлення або оновлення, і водночас немає потреби їх явно розмежовувати. Потім можна обробити зміни за допомогою інструкції T-SQL MERGE.

Стратегія ETL для поетапного завантаження таблиць фактів

Стратегія ETL для завантаження таблиць фактів завжди повинна починатися з аналізу даних джерела, щоб визначити можливість реалізації поетапного завантаження даних. Крім того, завжди потрібно запитувати у власників бізнесу, які винятки можливі. Наприклад, у деяких країнах допускається виконання проведення записів бухгалтерської книги, внесених на кілька місяців раніше від поточної дати. У таких випадках ви повинні мати гнучкіше ковзне вікно в кілька місяців, а не лише останні дані.

Наступна важлива деталь – керування операціями видалення та оновлення.

Здебільшого цих операцій потрібно уникати, коли справа стосується сховищ даних, тому що вони впливають на продуктивність.

Далі перелічено деякі загальні рекомендації щодо завантаження таблиць фактів.

– Секціонувати таблиці фактів.

– Додаткові дані повинні бути в одній секції, щоб їх легко було видалити.

– Без операції видалення, простим перемиканням секцій.

– Застосовувати таку стратегію завантаження:

  - завантажувати додаткові дані в таблицю з такою самою структурою, що й у таблиці фактів у місці призначення, без стиснення та індексів;
  - застосовувати необхідні індекси та стиснення;
  - переключати завантажену таблицю за допомогою секціонування на таблицю фактів місця призначення.

– Для одержання відповідних сурогатних ключів застосовуйте уточнювальні запити, що повністю кешовані.

З погляду служб SSIS це означає, що під час завантаження додаткових даних необхідно застосовувати комбінацію завдання *Execute SQL Task*, що керує секціями, із завданням потоку даних, що завантажує всі необхідні дані. Ця стратегія завантаження також важлива для нового індексу columnstore, оскільки це індекс лише для читання, і вам доведеться застосувати алгоритм переключення секцій для ефективного завантаження даних.

Потік помилок

Зробити пакети ETL надійними в експлуатації – важливе завдання. Це важко, тому що в реальному світі немає ідеальних джерел даних. Проблема зазвичай у тому, що ви не можете бути впевнені, що дані, які надходять із вихідних систем, завжди структуровані, як було задумано під час розроблення. Це означає, що ви можете надіслати дані, які спричинять збій вашого пакета ETL. Важливо відстежувати такі рядки даних, спрямовувати їх на додаткові перетворення роздування і повідомляти про це адміністраторів або операторів даних для ETL. У цьому занятті ви познайомитеся із застосуванням шляхів потоків помилок (error path) для спрямування збійних рядків в інші групи компонентів.

Застосування потоків помилок

Служби SSIS вміють виводити «погані рядки» з потоку даних і обробляти проблему даних, що виникла, не впливаючи на хороші рядки. Шляхи потоку даних керують рядками даних. Але, крім шляхів даних, існують і шляхи або висновки помилок. На робочому полі потоку даних вони позначені червоними лініями зі стрілками, що з'єднують компоненти потоку даних. Вони включають рядки даних, що викликали збій у компоненті, якщо було задано перенаправлення збійних рядків.

У потоці даних виведення помилок застосовують не всі компоненти. Наприклад, компонент Multicast (Багатоадресна доставка) лише копіює дані; він не виконує жодних операцій над самими даними, тому немає можливості для виникнення помилок і не існує виведення помилок. До компонентів, що

застосовують шляхи потоку помилок, належать усі адаптери джерел, адаптери призначень, перетворення Lookup (Уточнювальний запит), Conditional Split (Умовне розбиття) жодних операцій над самими даними, тому немає можливості для виникнення помилок і не існує виведення помилок. До компонентів, що застосовують шляхи потоку помилок, належать усі адаптери джерел, адаптери призначень, перетворення Lookup (Уточнювальний запит), Conditional Split (Умовне розбиття) тощо.

Для використання шляхів потоку помилок необхідно налаштувати виведення помилок. Існують три способи оброблення помилок у компонентах потоку даних:

– завдання для виведення помилок варіанта Fail Transformation (Компонент завершити з помилкою), який у разі виявлення помилки породжує збій у потоці даних;

– вибір варіанта Ignore Failure (Пропускати помилку) дозволяє рядку продовжити стандартний зелений шлях, але на виході помилкове значення замінюється значенням NULL;

– вказівка варіанта Redirect Rows (Перенаправляти рядок), що відправляє рядок із помилкою червоним шляхом; це єдиний спосіб обробити помилки за допомогою окремих компонентів.

Ці варіанти опрацювання помилок доступні всьому рядку й операціям з окремим стовпцем у рядку. Це не означає, що єдиний стовпець перенаправляється, але можна налаштувати одні стовпці на пропуск помилок, а інші – на перенаправлення. Діалогове вікно Configure Error Output (Налаштування виведення помилок) використовується для завдання властивостей. Для того щоб потрапити в це вікно, можна двічі клацнути на редагованому компоненті й вибрати команду Configure Error Output або просто перетягнути мишкою червону стрілку шляху потоку помилок на наступний компонент, що спричинить відкриття того самого вікна.

До поширених випадків використання виведення помилок належать текстові файли як джерела даних, у яких тип даних не відповідає типу, заданому в потоці даних, або

перетворення Lookup, у якому не знайдено збігу. Часом призначення стикається з помилковим рядком, якщо під час виконання вставлення порушується обмеження. Крім того, рядки з помилками можуть оброблятися по-різному залежно від того, чи потрібно їх відправити в проміжну таблицю для перегляду, або необхідно очистити дані і повернути їх назад у потік даних через перетворення Union All (Об'єднати всі).

Тестові питання

A. Яку (-і) з груп атрибутів можна описувати датами валідності значень бізнес-ключі:

- ☐ фіксовані атрибути;
- ☐ змінні атрибути;
- ☐ історичні атрибути?

B. Які компоненти повинна мати система відстежування зміни даних в My SQL Server:

- ☐ об'єднувач потоку;
- ☐ завдання керування;
- ☐ адаптер джерела;
- ☐ роздільник потоку?

C. Для якої групи атрибутів є необхідним створювати мітку істинного (активного) значення:

- ☐ бізнес-ключі;
- ☐ фіксовані атрибути;
- ☐ змінні атрибути;
- ☐ історичні атрибути?

D. Які функціональні можливості підтримує перетворення Slowly Changing Dimension (Виберіть усі відповідні варіанти.):

- ☐ Type 1 SCD;
- ☐ Type 2 SCD;
- ☐ СКД 3-го типу;
- ☐ елементи, що виводяться?

E. Які перетворення можна використати, щоб з'ясувати, чи були змінені значення конкретних стовпців під час порівняння вихідних даних із таблицею призначення:

- ☐ Data Conversion (Перетворення даних);

☐ Derived Column (Похідний стовпець);
☐ Conditional Split (Умовне розбиття);
☐ Multicast (Багатоадресне доставлення)?

F. Яке твердження справедливе щодо перетворення Slowly Changing Dimension (Виберіть усі відповідні варіанти.):

☐ перетворення Slowly Changing Dimension підтримує оновлення на основі наборів даних;
☐ перетворення Slowly Changing Dimension підтримує елементи, що виводяться;
☐ в одному потоці даних може бути кілька перетворень Slowly Changing Dimension;
☐ майстер Slowly Changing Dimension Wizard підтримує з'єднання лише із базою даних SQL Server?

G. Які режими оброблення в джерелі CDC можна використовувати для безпосереднього завантаження даних без застосування додаткового процесу ETL для отримання поточного значення рядка (Виберіть усі відповідні варіанти.):

☐ Net (Сумарні);
☐ All (Усі);
☐ All with old value (Усі зі старими значеннями);
☐ Net with merge (Сумарні зі злиттям)?

H. Який метод можна застосувати в службах SSIS для завдання динамічного запиту SQL під час виконання, якщо використовується адаптер джерела OLE DB (Виберіть усі відповідні варіанти.):

☐ використовувати збережену процедуру як джерело;
☐ використовувати параметри в інструкції SQL;
☐ застосувати вирази;
☐ використовувати команду SQL зі змінної?

I. Яких рекомендацій корисно дотримуватися, якщо ви застосовуєте стратегію ETL для додаткового завантаження (Виберіть усі відповідні варіанти.):

☐ застосовуйте секціонування таблиць фактів;
☐ уникайте повністю кешованих уточнювальних запитів;

☐ якщо у вас є вікно з вихідними даними, що перекривається, спочатку застосуйте інструкцію DELETE для видалення рядків, що перекриваються, в таблиці фактів;

☐ завантажте додаткові дані в таблицю, що має структуру, таку саму, як у таблиці фактів призначення?

J. У яких компонентів потоку даних є потік помилок (Виберіть усі відповідні варіанти.):

☐ адаптер OLE DB Source (Джерело «OLE DB»);

☐ перетворення Union All (Об'єднати всі);

☐ перетворення Merge (Злиття);

☐ перетворення Lookup (Уточнювальний запит)?

K. Які існують варіанти оброблення помилок на рівні рядків (Виберіть усі відповідні варіанти.):

☐ Ignore failure (Пропускати помилку);

☐ Fail component (Завершувати компонент із помилкою);

☐ Redirect row (Перенаправити рядок);

☐ Delete row (Видалити рядок)?

L. У яких адаптерів джерела даних є потік помилок (Виберіть усі відповідні варіанти.):

☐ OLE DB source;

☐ Raw File source (Джерело «Необроблений файл»);

☐ ODBC source;

☐ Excel source?

МАТЕРІАЛ ДЛЯ ПІДГОТОВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ 9

Теоретичні питання

Створення надійного пакета, що перезапускається. Транзакції пакета

Більшість реляційних баз даних, таких як SQL Server, виконують операції атомарними (неподільними) блоками. Це означає, що одна інструкція або набір інструкцій або успішні та впливають на дані, або неуспішні, і система повертає дані в стан, який був до спроби виконання інструкції. Блок оброблення, який повинен завершитися успішно для того, щоб дані були змінені, називається ***транзакцією*** (***transaction***).

Підтримка транзакцій вбудована у служби SSIS. Ця підтримка може бути задана на рівні різних пакетів, і можна координувати транзакції за допомогою повторного запуску пакета. Пакети можна налаштувати на дозвіл запуску однієї або декількох транзакцій, задавати успадкування транзакцій.

Визначення параметрів транзакції пакета та завдання

Можна задавати транзакції в пакеті на рівні всього пакета або на рівні будь-якого контейнера потоку керування чи завдання. Транзакції у службах SSIS використовують *Microsoft Distributed Transaction Coordinator* (MSDTC, Координатор розподілених транзакцій Microsoft); для функціонування транзакцій на комп'ютері повинна бути запущена служба MSDTC. Будь-яка служба або програма, яка активізується за допомогою MSDTC, може бути частиною транзакції в службах SSIS. Це означає, що служба MSDTC дає можливість виконувати розподілені транзакції, наприклад, оновлення бази даних SQL Server і бази даних Oracle в одній і тій самій транзакції.

Для запуску транзакції в пакеті необхідно задати у властивості завдання або контейнера *TransactionOption* значення *Required*. Властивість *TransactionOption* є на рівні пакета, на рівні контейнера і майже в усіх завдань потоку керування.

Вона може мати одне з таких значень:

– *Required* – якщо транзакція існує, приєднується до неї; якщо не запущена, запускає нову транзакцію;

– *Supported* – якщо транзакція існує, приєднується до неї (це значення за замовчуванням);

– *NotSupported* – пакет, контейнер або завдання не повинні приєднуватися до наявної транзакції.

На рисунку 9.1 показано значення властивостей контейнера послідовності в області потоку керування. Зверніть увагу на те, що в цьому випадку у властивості *TransactionOption* задано значення *Required*.

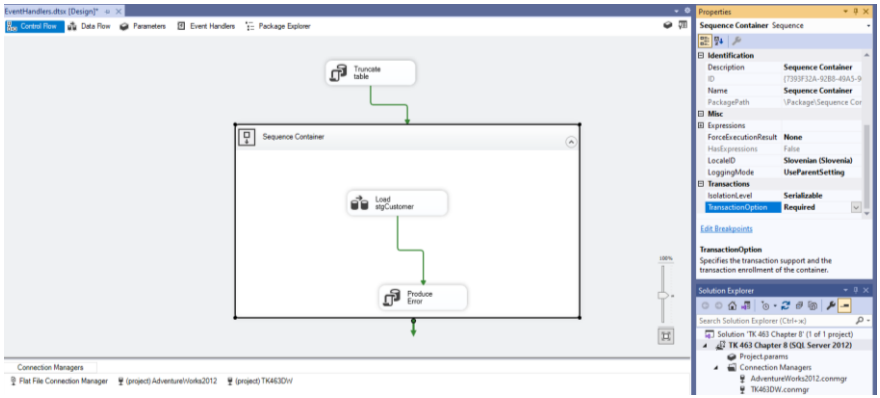

Рисунок 9.1 – Ілюстрація галаштування властивості *TransactionOption*

Дотримуйтеся таких рекомендації під час ухвалення рішення про спосіб реалізації транзакції:

– для підтримки транзакцій у службах *SSIS* необхідно запустити службу *MSDTC* і завдання транзакцій повинні працювати зі службами *MSDTC* за замовчуванням;

– якщо набір завдань повинен виконуватися як єдиний блок, то необхідно помістити ці завдання в контейнер послідовності і задати в його властивості *TransactionOption* значення *Required*;

– завдання може успадкувати налаштування транзакції від свого батька, якщо властивість *TransactionOption* має

значення *Supported*, стандартне значення під час створення завдання або контейнера;

– якщо конкретне завдання повинне бути виключена з транзакції, то необхідно задати у властивості *TransactionOption* значення *NotSupported*;

– якщо задана властивість *TransactionOption* зі значенням *Required* у контейнері циклів *Foreach Loop* або *For Loop*, нову транзакцію створюватимуть у кожному проході циклу;

– транзакції виконуються на рівні потоку керування, а не в потоці даних. Отже, можна запустити транзакцію для завдання потоку даних, але не можна зробити це для окремих компонентів потоку даних; або оброблення даних загалом буде успішним, або відбудеться відкат;

– запуск завдань у транзакціях вимагає додаткових непродуктивних витрат із погляду служб *SSIS* і впливає на продуктивність пакетів, тому застосовувати їх потрібно лише в тих випадках, коли виконання завдання як єдиного блоку обов'язкове;

– налаштування взаємодії служби *MSDTC* з вузлами і периферичними пристроями зазвичай непросте завдання. Для запитів, що довго виконуються, необхідно заздалегідь з'ясувати можливі затримки і несподівані розриви з'єднань;

– для включення потоку даних у транзакцію вихідний постачальник даних, який використовується диспетчером з'єднань, також повинен підтримувати транзакції MSDTC;

– простим переглядом потоку керування в пакеті не визначити, які з контейнерів або із завдань дійсно підтримують транзакції або налаштовані на приєднання до транзакцій. Необхідно перевіряти властивість *TransactionOption* у кожному елементі, щоб отримати потрібну інформацію;

– не кожне завдання здатне виконати відкат, якщо транзакція завершилася з помилкою. Завдання *XML Task* і *File System Task* не підтримують транзакції.

Рівні ізоляції транзакції

Рівні ізоляції транзакції (*transaction isolation levels*) керують рівнем блокування, яке встановлюється під час відбору даних. Установлення рівня ізоляції транзакції для контейнера або завдання у службах *SSIS* дає можливість визначити, наскільки суворо необхідно ізолювати цю транзакцію від інших транзакцій. Мовою бази даних рівні ізоляції транзакції зазначають, які дані бачимо в інструкціях всередині транзакції. Ці рівні безпосередньо впливають на рівень паралельного доступу, визначаючи, яка взаємодія можлива між транзакціями, спрямованими на одне й те саме цільове джерело даних.

У службах SSIS потрібний рівень ізоляції транзакції задається у властивості **IsolationLevel** завдання або контейнера. Підтримуються такі рівні ізоляції:

– *Unspecified* – використовується рівень ізоляції, відмінний від заданого, але він не може бути визначений. Використовується для всього пакета для перевизначення рівнів ізоляції всередині пакета з подальшим використанням у ньому відповідних рівнів для контейнерів і завдань.

– *ReadUncommitted* – не блокуються записи, що зчитуються. Це означає, що незафіксовану зміну може бути зчитано і потім скасовано іншим клієнтом, що призводить до формування локальної копії запису, яка не узгоджується з тією, що зберігається в базі даних. Цей процес називається брудним зчитуванням (*dirty read*), тому що дані суперечливі.

– *Chaos* – поведінка, як у випадку *ReadUncommitted*, але під час операції запису перевіряється рівень ізоляції інших незавершених транзакцій, тому транзакції з вищими рівнями ізоляції не перезаписуються. Крім того, цей рівень ізоляції не підтримується платформою SQL Server, не включений до стандарту ANSI і не дає можливості виконати відкат.

– *ReadCommitted* – під час зчитування записи блокуються, і блокування знімається, щойно зчитування закінчено. Цей рівень перешкоджає зчитуванню змін до моменту їх фіксації, але не заважає іншим клієнтам під час транзакції вставляти, видаляти або змінювати інші записи.

– *RepeatableRead* – під час зчитування записи блокуються, і блокування зберігається до завершення транзакції. Цей рівень гарантує незмінність зчитуваних даних під час транзакції.

– *Serializable* – блокується весь зчитуваний набір даних, і блокування зберігається до завершення транзакції. Цей рівень гарантує на час транзакції незмінність даних і збереження їх упорядкованості в базі даних. Це значення прийнято за замовчуванням у службах *SSIS*.

– *Snapshot* – дані, що зчитуються в транзакції, не відображатимуть зміни, зроблені в інших одночасно виконуваних транзакціях. Транзакція використовує версії рядків даних, які існують на момент початку транзакції. Під час зчитування не застосовується жодне блокування даних. Значення *Snapshot* властивості *IsolationLevel* не сумісне з транзакціями пакета, тому цьому не можна застосовувати властивість *IsolationLevel* для встановлення цього рівня ізоляції для транзакцій пакета. Замість цього, щоб задати рівень ізоляції транзакцій пакета *Snapshot*, можна застосувати запит на SQL.

Транзакції, що обробляються вручну

Крім вбудованого механізму оброблення транзакцій є можливість керувати ними самостійно. Робиться це явним запуском транзакції на рівні бази даних за допомогою завдання ***Execute SQL Task*** і подальшим виконанням інструкції SQL для запуску транзакції. На рисунку 9.2 показано приклад, у якому використовується база даних SQL Server. У цьому прикладі спочатку застосовується інструкція SQL:

```
BEGIN TRAN;
```

Потім у завданні за допомогою завдання потоку даних реалізується необхідна бізнес-логіка і наприкінці фіксується транзакція в додатковому завданні *Execute SQL Task*, що виконує інструкцію

```
COMMIT TRAN;
```

Для керування транзакціями в декількох завданнях SSIS необхідно реалізувати механізм, що дає можливість зберігати одне й те саме з’єднання в завданнях. Зробити це можна,

встановивши у властивості *RetainSameConnection* значення *True*. Після налаштування диспетчер з'єднань намагатиметься повторно використовувати наявне з'єднання, коли завдання запросить його (замість створення щоразу нового з'єднання). За замовчуванням більшість постачальників SQL Server дозволяє встановлення цієї властивості, але потрібно пам'ятати, що властивість *RetainSameConnection* визначається базовим постачальником, і потрібно переконатися, що цей постачальник підтримує цю функціональну можливість.

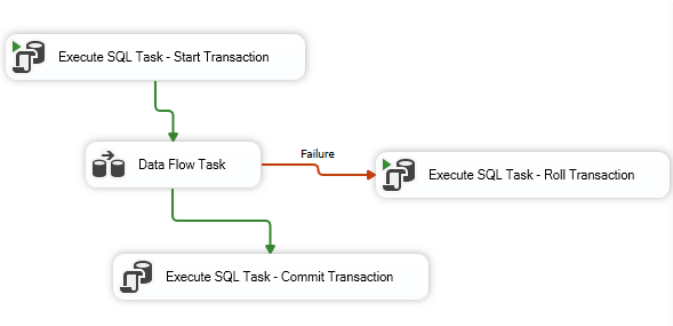

Рисунок 9.2 – Ілюстрація додавання ручного оброблення транзакцій за допомогою вставлення кількох завдань Execute SQL Task

Контрольні точки

Існує можливість налаштувати пакет за повторного виконання на запуск із точки збою або з більш ранньої точки. У службах SSIS цей процес налаштування називають вставленням *контрольних точок* (*checkpoints*). Контрольні точки діють разом із транзакціями, забезпечуючи можливість перезапуску пакета.

Створення контрольних точок для перезапуску пакета

Іноді необхідно мати можливість перезапустити пакет після збою, почавши його виконання з точки, що спричинила збій, особливо якщо виконується робота зі складними пакетами, і пакетами, що довго виконуються. Інакше кажучи, за необхідності повторного запуску пакета може виникнути потреба пропустити виконання завдань, що завершилися

успішно. Це можна реалізувати за допомогою включення до пакета контрольних точок.

Для активізації можливості перезапуску пакета спочатку потрібно ввімкнути в пакеті використання контрольних точок і потім вибрати відповідні завдання і контейнери для запису контрольних точок. Для увімкнення контрольних точок у пакеті необхідно виконати такі дії:

– у пакеті відкрити вікно *Properties* та перейти на вкладку *Control Flow* конструктора *SSIS*;

– задати властивість *SaveCheckpoints* на рівні пакета, що дорівнює *True*;

– у властивості *CheckpointFileName* зазначте коректний шлях та ім'я файлу для файлу контрольних точок. Пакети використовують файли для збереження інформації про свій стан, тому якщо в пакеті виникне помилка і пізніше його буде перезапущено, пакет зможе прочитати файл контрольних точок, щоб визначити, на чому він зупинився й одержати дані про стан в останньому успішно виконаному завданні;

– задати у властивості *CheckpointUsage* значення *IfExists*, що спричинить виконання пакета з початкової точки за відсутності файлу контрольних точок, і виконання з позначеної точки, якщо цей файл є. На рисунку 9.3 показано налаштовані властивості пакета за описаним алгоритмом дій.

Для встановлення контрольних точок у різних завданнях пакета необхідно задати значення *True* у властивості *FailPackageOnFailure* для всіх завдань або контейнерів.

Під час виконання пакета з увімкненими контрольними точками будуть відбуватися дані кроки.

1. Пакет виконує перевірку наявності файлу контрольних точок. Якщо такого файлу немає, пакет починає виконуватися з першого завдання (або паралельних завдань) у потоці керування. Якщо ж файл контрольних точок знайдено, пакет його читає, щоб з'ясувати, звідки стартувати (і як оновити значення змінних і з'єднання в момент останнього збою).

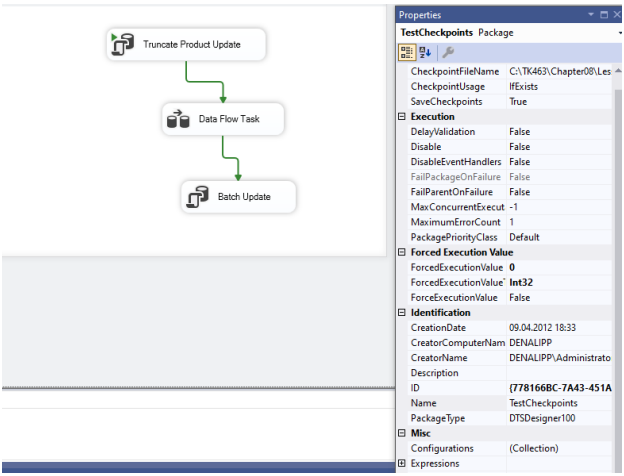

Рисунок 9.3 – Ілюстрація налаштування властивостей на рівні пакета для використання контрольних точок

2. У кожній успішній контрольній точці пакета (якщо в задачі потоку даних у властивості *FailPackageOnFailure* значення *True* і завдання виконано успішно) файл контрольних точок оновлюється.

3. Якщо в пакеті виникає збій, файл контрольних точок залишається у файловій системі і зберігає останнє оновлення з останньої успішної контрольної точки.

4. Якщо пакет виконався без помилок, файл контрольних точок видаляється. Таким чином, коли пакет буде виконуватися наступного разу, файлу контрольних точок не буде, і пакет почне виконання з першого завдання або завдань.

На рисунку 9.4 показано перше виконання тестового пакета. У цей момент немає файлу контрольних точок, тому пакет починається із завдання *Execute SQL Task.*

Під час виконання потоку керування, показаного на рисунку 9.4, завдання *Truncate Update Table* виконалося без помилок і служба *SSIS* записала контрольну точку у файл контрольних точок. Завдання потоку даних завершилося збоєм, оновлення файлу контрольних точок не відбулося, і пакет припинив виконання.

У цей момент помилку в завданні потоку даних було виправлено, і пакет запустили на виконання повторно. На рисунку 9.5 показано потік керування перезапущеного пакета. Як проілюстровано, під час першого виконання пакет завершився на кроці 2. Після коригування помилки друге виконання пакета почалося з кроку 2 і продовжилося до завершення.

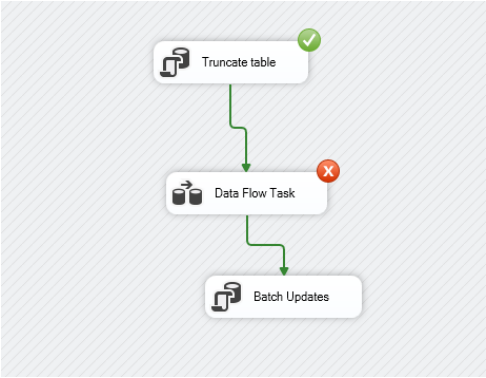

Рисунок 9.4 – Ілюстрація завершення пакета з помилкою в завданні потоку даних

Після завершення останнього кроку, на рисунку 9.5 названого *Batch Updates*, файл контрольних точок було видалено.

Ураховуючи використання контрольних точок, у великих і складних пакетах можна заощадити масу часу в процесі повторних запусків за рахунок пропуску завдань, які в останньому прогоні виконалися успішно, і запуску з того завдання, яка завершилася збоєм. Необхідно пам’ятати, що можна увімкнути для завдання потоку даних використання контрольних точок, але всередині завдання потоку даних їх встановити не можна. Це означає, що, якщо всередині завдання потоку даних є три перетворення і виконання завершилося збоєм на третьому перетворенні, під час повторного запуску пакета його виконання почнеться із завдання потоку даних і

перші два перетворення будуть виконані повторно, перш ніж справа дійде до третього перетворення.

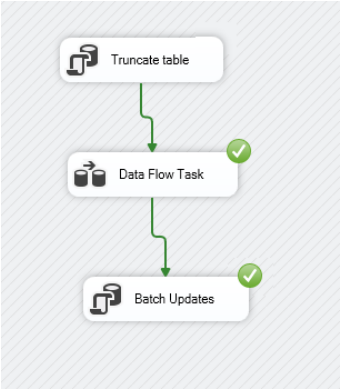

Рисунок 9.5 – Ілюстрація успішного виконання пакета після другого запуску

Обробники подій

Пакети, контейнери та завдання потоку керування під час виконання генерують події. Служби SSIS надають засоби відстеження подій часу виконання та виконання інших операцій під час генерації подій. Ця функціональна можливість SSIS *називається обробленням подій* (*event handling*). Можна створювати користувацькі обробники подій для розширення функціональних можливостей пакета та полегшення керування пакетами під час їхнього виконання. Обробники подій можуть виконувати такі завдання, як очищення області тимчасового зберігання даних після завершення виконання завдання або вилучення системної інформації, що дає можливість оцінити доступні ресурси перед запуском пакета. За допомогою обробників подій ви також можете реалізувати користувацькі розв’язки завдань реєстрації та протоколювання або аудиту у ваших процесах ETL.

Застосування обробників подій

Обробники подій використовують для оброблення робочого процесу моделі потоку керування, що містить ті самі

завдання потоку керування і контейнери, які є на панелі *SSIS Toolbox* в області конструктора потоку керування. Можна визначити будь-яку кількість обробників подій у пакеті.

Для вставлення в пакет обробника подій необхідно виконати такі дії:

– у конструкторі служб SSIS перейти на вкладку *Event Handlers;*

– вибрати для обробника подій виконуваний компонент зі списку *Executable* (рис. 9.6);

– виділити подію в списку *Event handler;*

– з панелі SSIS Toolbox перетягнути мишкою завдання, які будуть виконуватися з урахуванням обраних області дії та події.

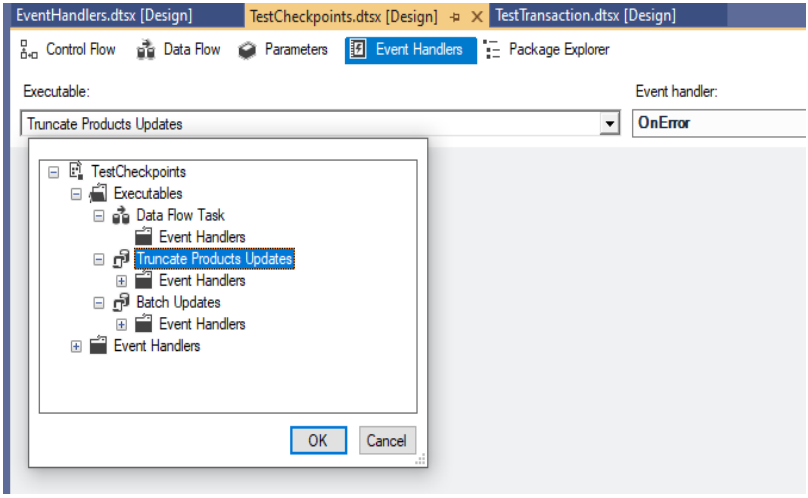

Рисунок 9.6 – Приклад визначення обробників подій на вкладці *Event Handlers* конструктора служб SSIS

Виконуваний компонент – це завдання, контейнер або навіть цілий пакет, що генерує подію. *Обробник події* – це конкретна подія, що викликає запуск робочого процесу події. У таблиці 9.1 описано обробники подій різних типів.

Таблиця 9.1 – Типи обробників подій

Обробник події	**Опис**
1	2
OnError	Запускається, коли виконуваний компонент повідомляє про помилку
OnExecStatusChanged	Запускається для всіх завдань і контейнерів, коли стан виконання змінюється на in process, success або failed
OnInformation	Запускається, коли служби SSIS відображають інформаційні повідомлення під час перевірки та виконання завдань або контейнера
OnPostExecute	Запускається, коли завдання або контейнер успішно завершилися
OnPostValidate	Запускається після успішної перевірки завдання або контейнера
OnPreExecute	Запускається перед виконанням виконуваного компонента
OnPreValidate	Запускається перед перевіркою компонента системою
OnProgress	Виконується, коли підсистемою SSIS надсилається повідомлення про хід виконання, зазначаючи про відчутне просування вперед завдання або контейнера
OnQueryCancel	Запускається, коли завдання Execute SQL Task скасовується під час втручання людини, наприклад, виникає зупинення виконання пакета

Продовження таблиці 9.1

1	2
OnTaskFailed	Аналогічний обробнику OnError, але запускається, якщо завдання завершується зі збоєм, а не за кожного виникнення помилки
OnVariableValueChanged	Запускається, коли змінюється значення змінної, у якої у властивості RaiseChangeEvent встановлено значення True
OnWarning	Запускається, коли завдання повертає попередження, наприклад, про стовпець, який не використовувався в потоці даних

Крім того, обробники подій із призначеною областю дії виконуваного об'єкта поширюються на дочірні події, якщо така подія виникає. Якщо подію призначено контейнеру, дочірні виконувані файли включають завдання і контейнери, вбудовані в батьківський контейнер. Це означає, що, якщо подія *OnError* призначена пакету і подія *OnError* виникає на рівні завдання, обробник події запускається для завдання і пакета (і для будь-яких контейнерів між ними).

Тестові питання

А. На якому (-их) рівні (-ях) можна задати параметри транзакцій пакетів:

- ☐ рівень пакета;
- ☐ рівень потоку;
- ☐ рівень завдання?

В. Чим відрізняються рівні ізоляції транзакцій:

- ☐ послідовністю виконання операцій транзакцій;
- ☐ частотою виконання операцій транзакцій;
- ☐ часовою міткою відкату змін;
- ☐ рівнем блокування даних?

C. Оберіть усі особливості виконання транзакцій:

☐ виконання всіх операцій транзакцій;

☐ виконання частини операцій транзакцій;

☐ зміна даних після успішного виконання всіх операцій транзакцій;

☐ зміна даних після успішного виконання більше ніж 50 % операцій транзакцій.

D. Яке завдання в потоці керування підтримує транзакції (Виберіть усі відповідні варіанти.):

☐ Data flow Task (Завдання потоку даних);

☐ Execute SQL Task (Виконання SQL);

☐ File System Task (Файлова система);

☐ XML Task (XML)?

E. Які рівні ізоляції транзакції не блокують зчитувані записи (Виберіть усі відповідні варіанти.):

☐ Serializable;

☐ Snapshot;

☐ Хаос;

☐ ReadUncommitted?

F. Які інструкції T-SQL можна застосовувати для ручного оброблення транзакцій (Виберіть усі відповідні варіанти.):

☐ BEGIN TRAN;

☐ ROLLBACK TRAN;

☐ END TRAN;

☐ COMMIT TRAN?

G. Які властивості необхідно установити на рівні пакета для увімкнення використання контрольних точок (Виберіть усі відповідні варіанти.):

☐ CheckpointFileName;

☐ CheckpointUsage;

☐ SaveCheckpoints;

☐ FailPackageOnFailure?

H. У яких об’єктах служб SSIS можна активувати контрольні точки (Виберіть усі відповідні варіанти.):

☐ завдання потоку даних;

☐ завдання потоку керування;
☐ контейнер послідовності;
☐ перетворення Sort (Сортування)?

I. Які твердження про контрольні точки справедливі (Виберіть усі відповідні варіанти.):

☐ для одного пакета можна використовувати кілька файлів контрольних точок;
☐ якщо завдання потоку даних завершується з помилкою, ви можете зберегти рядки з помилками у файлі контрольних точок;
☐ якщо пакет завершився успішно, файл контрольних точок видаляється;
☐ якщо ви задали у властивості CheckpointUsage значення Always, обов'язково повинен існувати файл контрольних точок, інакше пакет не запуститься?

J. Які компоненти у службах SSIS можуть діяти як виконувані компоненти, що збуджують подію (Виберіть усі відповідні варіанти.):

☐ контейнер послідовності;
☐ пакет;
☐ завдання потоку даних;
☐ перетворення потоку даних?

K. Обробники подій яких типів можна застосовувати для протоколювання всіх помилок пакета (Виберіть усі відповідні варіанти.):

☐ OnPostExecute;
☐ OnError;
☐ OnWarning;
☐ OnTaskFailed?

L. Обробники яких типів можна використовувати для реєстрації інформації перед запуском завдання в пакеті (Виберіть усі відповідні варіанти.):

☐ OnTaskFailed;
☐ OnProgress;
☐ OnPreExecute;
☐ OnWarning?

МАТЕРІАЛ ДЛЯ ПІДГОТОВКИ ДО ВИКОНАННЯ ПРАКТИЧНОЇ РОБОТИ 10

Теоретичний матеріал

Створення динамічних пакетів

Під час розроблення пакетів служб *Microsoft SQL Server Integration Services* (*SSIS*) корисно робити всі завдання або перетворення настільки динамічними, наскільки це можливо. Такий підхід дасть можливість переміщати пакети з одного середовища в інше (наприклад, із середовища розробки в тестове середовище, а потім у виробниче середовище) без додаткових змін пакета. Також можна встановлювати різні властивості в пакетах під час їхнього виконання.

Для цього можна використовувати параметри і диспетчери з'єднань рівня проєкту, або застосовувати конфігурації пакетів. Застосування динамічних пакетів усуває необхідність внесення змін під час переходу з одного середовища в інше або відкриття проєкту, якщо виникла потреба виконати пакети з іншими значеннями властивостей або змінних.

Параметри та диспетчери з'єднань рівнів пакета та проєкту

Починаючи від SQL Server 2012 у служби *SSIS* введено параметри та диспетчери з'єднань рівня проєкту. Параметри дають можливість присвоювати значення властивостям у пакетах під час їх виконання. Диспетчери з'єднань рівня проєкту дають можливість один раз визначити з'єднання з джерелом даних і потім використовувати його в усіх пакетах, що входять у проєкт. Обидва засоби доступні лише в проєктах, розроблених для моделі розгортання проєктів (*Project Deployment Model*).

Застосування диспетчерів з'єднань рівня проєкту

Диспетчери з'єднань рівня проєкту дають можливість установлювати диспетчер з'єднань для проєкту. Можна визначити з'єднання для всього проєкту і використовувати його в усіх пакетах, що входять до цього проєкту.

Для створення з'єднання рівня проєкту необхідно визначити новий диспетчер з'єднань, клацнувши правою кнопкою мишки на папці *Connection Managers* на панелі *Solution Explorer* під ім'ям проєкту та вибрати варіант *New Connection Manager*.

Альтернативний метод – перетворити наявне з'єднання рівня пакета на з'єднання рівня проєкту. Для цього потрібно відкрити пакет, у вікні *Connection Managers* клацнути правою кнопкою мишки на з'єднанні, яке хочете перетворити, і вибрати команду *Convert to Project Connection*, як показано на рисунку 10.1.

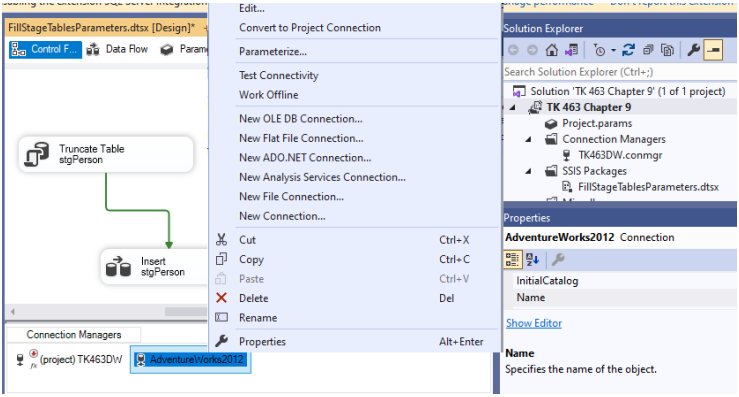

Рисунок 10.1 – Перетворення з'єднання пакета на з'єднання рівня проєкту

Параметри

Параметри дають можливість присвоювати значення властивостям пакетів під час їхнього виконання. Параметри також можна застосовувати у виразах служб SSIS – наприклад, скористатися завданням *Expression Task* для встановлення значення змінної, що базується на певному значенні параметра. Є два типи параметрів: параметри проєкту, створені на рівні проєкту, і параметри пакета, створені на рівні пакета.

Установлювати і використовувати параметри можна лише в проєктах, розроблених для моделі розгортання проєктів. Якщо застосовується модель розгортання проєктів, проєкти

розгортаються в каталозі *Integration Services* (*Integration Services catalog*).

Використання параметрів. Можна використовувати один параметр для присвоєння значення кільком властивостям пакета і застосовувати один параметр у кількох виразах служб SSIS. Залежно від місця проєкту в життєвому циклі розгортання проєктів ви можете присвоїти до трьох різних типів значень кожному параметру. Ці три типи перелічені в тому порядку, в якому вони застосовуються до параметра.

– *Значення проєкту (Design Default Value*) – значення за замовчуванням, яке задається під час створення або зміни проєкту в середовищі *SQL Server Data Tools*. Це значення зберігається разом із проєктом.

– *Значення сервера* (*Server Default Value*) – значення за замовчуванням, що присвоюється під час розгортання проєкту або пізніше, коли проєкт розміщується в каталозі *SSIS*. Це значення перевизначає значення проєкту.

– *Значення виконання* (*Execution Value*) – значення, що присвоюється в посиланні на конкретний екземпляр виконуваного проєкту. Це присвоювання перевизначає всі інші значення, але застосовується до єдиного екземпляра виконуваного проєкту.

Визначення параметрів. Визначення параметрів виконується в середовищі *SSDT*. Зазвичай параметр проєкту створюється у вузлі *Project.params* під іменем проєкту на панелі *Solution Explorer* (рис. 10.2). Для додавання параметра пакета спочатку потрібно відкрити пакет і потім перейти на вкладку параметрів в області конструктора пакета.

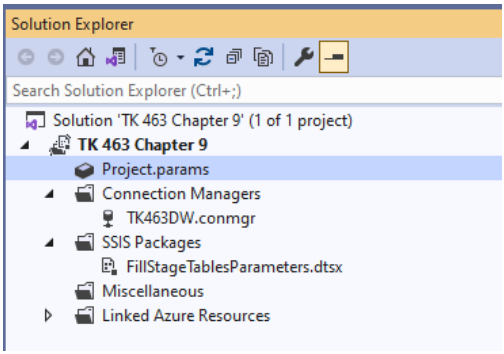

Рисунок 10.2 – Панель Solution Explorer із виділеною папкою Project.params

Описати новий параметр можна, клацнувши кнопкою мишки на значку *Add Parameter* (Додати параметр) відкритого вікна параметрів проєкту або пакета (перший ліворуч значок, синій кубик). На рисунку 10.3 показано вікно параметрів з одним параметром *pTK463DWConnectionString*.

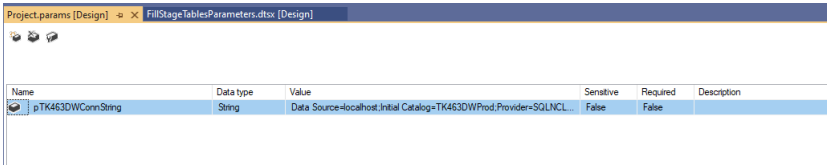

Рисунок 10.3 – Вікно параметрів

У середовищі *SSDT* під час створення параметра є кілька властивостей, які потрібно встановити.

– *Name* (Ім’я) – ім’я параметра. Перший символ імені повинен бути літерою або символом підкреслення.

– *Data type* (Тип даних) – тип даних параметра.

– *Value* (Значення) – стандартне значення параметра, що присвоюється на стадії розроблення, відоме як значення проєкту.

– *Sensitive* (Конфіденційний) – якщо значення цієї властивості True, значення параметра в каталозі шифруються і виводяться як значення NULL під час відображення в

інструкціях Transact-SQL або в середовищі SQL Server Management Studio.

– *Required* (Обов'язкове) – перед виконанням пакета потрібно задати значення, відмінне від заданого значення проєкту.

– *Description* (Опис) – опис параметра для спрощення обслуговування. У середовищі *SSDT* необхідно додати опис параметра у вікно *Visual Studio Properties* (Властивості Visual Studio), коли параметр виділено у відповідному вікні параметрів.

Параметри можна редагувати у списку вікна параметрів або використовувати вікно *Properties* для зміни значень параметрів. Видалити параметр можна за допомогою кнопки *Delete Parameter* на панелі інструментів. За допомогою кнопки *Add Parameters To Configurations* на панелі інструментів (остання кнопка праворуч) можна задати значення параметра в конкретній конфігурації побудови, яка застосовується лише під час виконання пакета в середовищі *SQL Server Data Tools*.

Для використання іншого неявного способу створення параметра клацніть правою кнопкою мишки на завданні потоку керування або диспетчері з'єднань і виберіть команду *Parameterize*. У діалоговому вікні (рис. 10.4) можна зазначити, яка властивість повинна обчислюватися динамічно, під час виконання, і чи потрібно створити параметр (останнє зазначається вибором перемикача *Create new parameter*).

Конфігурації побудови у службах SQL Server Integration Services

Конфігурації побудови в середовищі Visual Studio надають спосіб збереження декількох варіантів структури і властивостей проєкту. До активної конфігурації забезпечується швидкий доступ, її можна змінювати, легко створюючи кілька конфігурацій побудови для одного й того самого проєкту.

Створення конфігурацій побудови

У проєкті може бути набір певних властивостей проєкту для кожної унікальної комбінації конфігурації та платформи.

Створити додаткову конфігурацію проєкту або розв’язок можна за допомогою диспетчера *Configuration Manager*.

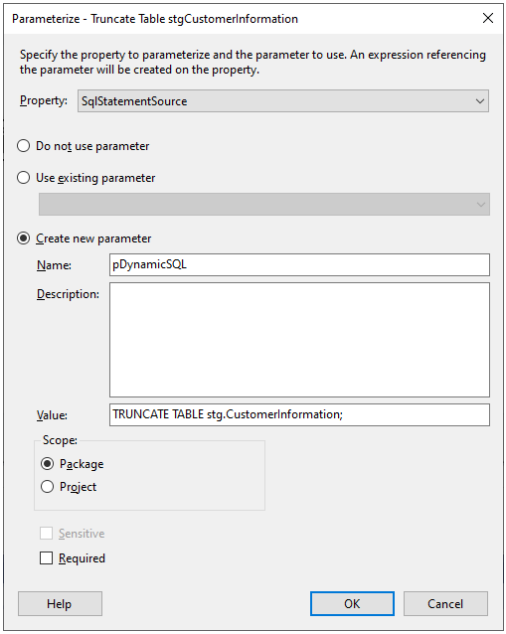

Рисунок 10.4 – Діалогове вікно Parameterize

Для відкриття вікна диспетчера *Configuration Manager* необхідно вибрати рядок *Configuration Manager* зі списку елемента *Configurations* на стандартній панелі інструментів. Другий спосіб одержати доступ до диспетчера: спочатку відкрити діалогове вікно *Property Pages* проєкту (подвійний клік правою кнопкою мишки по назві проєкту на панелі *Solution Explorer* і вибір команди *Properties*) і натиснути кнопку *Configuration Manager* (рис. 10.5).

У випадному списку *Active Solution Configuration* необхідно вибрати варіант *New*. У діалоговому вікні *New Solution Configuration* ввести ім’я конфігурації для створюваного вами розв’язку. Необхідно установити прапорець *Create new project configurations* для створення і конфігурування проєкту. Тепер можна змінити активну конфігурацію, вибравши її

безпосередньо у випадному списку *Solution Configurations* на стандартній панелі інструментів у діалоговому вікні *Configuration Manager*.

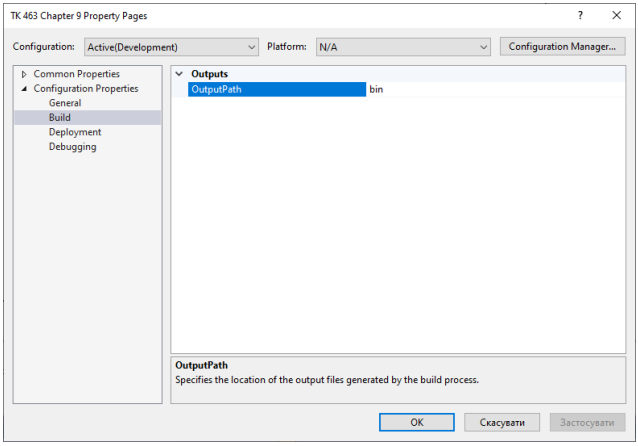

Рисунок 10.5 – Ілюстрація вмісту діалогового вікна *Property Pages* для проєкту

Застосування конфігурацій побудови

Зв'язати значення параметрів із конфігураціями побудови можна за допомогою вікна параметрів. Для цього необхідно виконати такі дії:

1. Відкрити вікно параметрів проєкту або пакета.
2. У вікні параметрів виділити праворуч кнопку на панелі інструментів *Add Parameters To Configurations*.
3. У діалоговому вікні *Manage Parameter Values* можна вставити значення для будь-якого параметра і будь-якої конфігурації. Для вставлення параметра в конфігурацію необхідно натиснути кнопку *Add*.
4. У вікні додавання параметрів *Add Parameters* можна виділіть параметри та натиснути кнопку *OK*.
5. Останнім кроком необхідно задати значення параметрів, які підходять для поточних конфігурацій (рис. 10.6).

На рисунку 10.6 проілюстровано, що параметр *pNoOfRows* матиме значення 10 000 під час виконання пакета,

що використовує конфігурацію *Production*, і 100 – якщо під час виконання буде використано конфігурацію *Development*. Це означає, що якщо є кілька параметрів і необхідно змінити значення параметрів під час проєктування, коли проєкт SSIS розробляється або налагоджується, потрібно лише перейти до іншої конфігурації побудови, і всі значення параметрів зміняться одночасно.

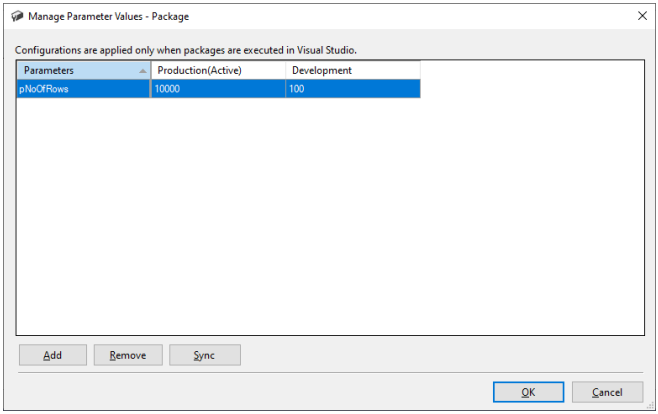

Рисунок 10.6 – Керування значеннями параметрів

Конфігурації побудови можна також застосовувати для встановлення властивостей у конфігураціях для побудови, розгортання та налагодження проєкту. Для присвоювання різних значень властивостям у різних конфігураціях потрібно клацнути правою кнопкою мишки по імені проєкту на панелі *Solution Explorer* і вибрати команду *Properties*. На рисунку 10.7 показано властивості розгортання в діалоговому вікні проєкту *Property Pages*. Якщо потрібно розгорнути проєкт на кількох серверах, можна задати різні значення у властивостях *Server Name* і *Server Project Path* для кожної конфігурації, щоб за допомогою конфігурацій можна було швидко перейти між середовищами розгортання на стадії розроблення проєкту.

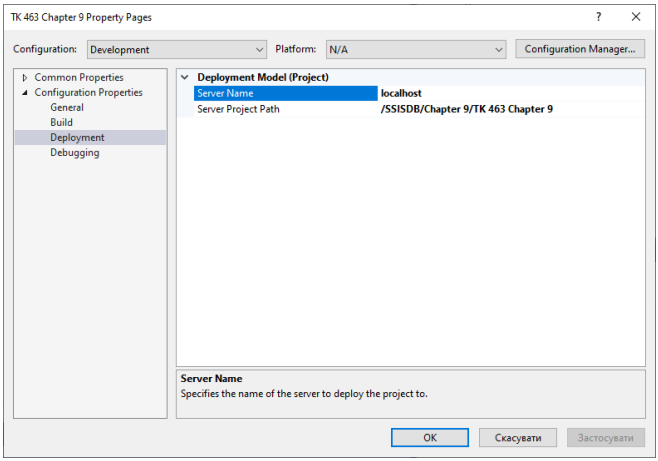

Рисунок 10.7 – Властивості конфігурації *Deployment* у діалоговому вікні проєкту *Property Pages*

Вирази властивостей

Вирази служб SSIS, що застосовуються для оновлення властивостей потоку керування під час виконання пакета, називають *виразами властивостей* (*property expressions*). Використовувати вираз властивості можна двома способами. По-перше, можна задати властивість у вигляді виразу у вікні властивостей, натиснувши кнопку з три крапки (...) властивості *Expressions*. Відкриється вікно *Property Expressions Editor*, показане на рисунку 10.8. У цьому діалоговому вікні можна вибрати властивість зі списку і потім ввести вираз.

Другий шлях створення виразу властивості – застосування редактора завдання або контейнера, які пропонують один або кілька способів завдання властивостей із використанням виразів. На рисунку 10.9 показано вікно *Execute SQL Task Editor*. Задати вирази властивостей можна на сторінці *Expressions*.

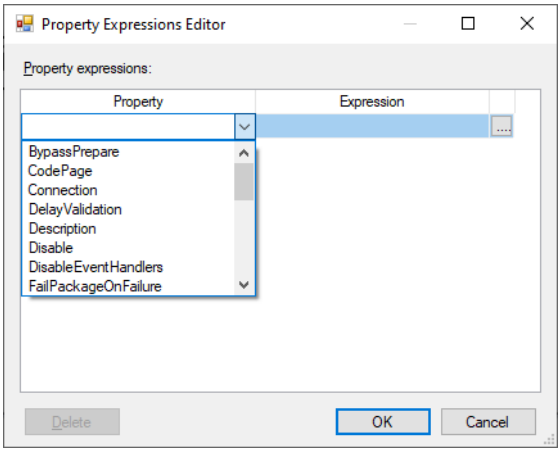

Рисунок 10.8 – Вікно редактора *Property Expressions Editor*

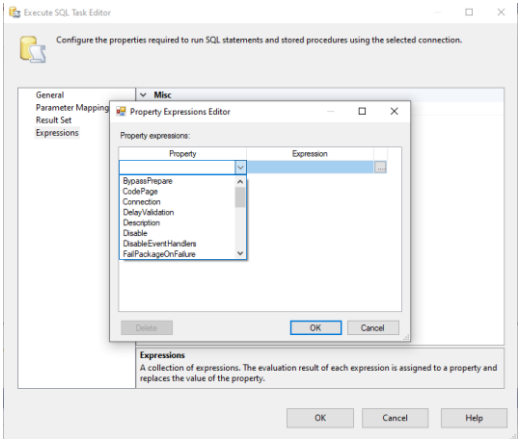

Рисунок 10.9 – Вікно Property Expressions Editor з активною *сторінкою Expressions*

Конфігурації пакета

Робота з конфігураціями пакета. Коли виконується пакет у моделі розгортання пакетів, пакет насамперед переглядає свої конфігурації та перевизначає свої поточні налаштування новими налаштуваннями з конфігурацій. За допомогою конфігурацій часто налаштовуються такі елементи:

– *Властивості з'єднання*. До них належать властивості, що задають рядок з'єднання, ім'я сервера, ім'я користувача і пароль.

– *Властивості змінних пакета*. Можна задати значення змінних, описи змінних і властивість *RaiseChangeEvent*.

– *Властивості пакета*. До них належать будь-які властивості, що визначаються на рівні пакета, такі як налаштування контрольних точок і безпеки.

Перш ніж з'явиться можливість включити конфігурацію пакета, необхідно перетворити проєкт SSIS на модель розгортання пакета. За замовчуванням новий пакет у службах *SSIS* встановлюється для моделі розгортання проєкту. Цю установку можна змінити, вибравши команду *Convert To Package Deployment Model* на головній панелі інструментів під ім'ям проєкту. Потрібно враховувати, що перетворити можна лише ті проєкти, у яких немає жодних параметрів або диспетчерів з'єднань, визначених на рівні проєкту.

За замовчуванням конфігурацію пакета в усіх пакетах вимкнено. Для ввімкнення та налаштування конфігурацій застосовується *Package Configuration Organizer*, за допомогою якого можна виконати такі завдання:

– увімкнути або вимкнути конфігурації пакета у вашому пакеті;

– додати або видалити конфігурації, призначені для вашого пакета;

– визначити порядок застосування конфігурацій.

Для відкриття вікна *Package Configurations Organizer* необхідно відкрити пакет, для якого потрібно ввімкнути конфігурації, і потім вибрати із меню служб SSIS команду *SSIS Configurations*. Для ввімкнення конфігурацій необхідно встановити прапорець *Enable package configurations* у верхній частині діалогового вікна. На рисунку 10.10 показане діалогове вікно *Package Configurations Organizer* з увімкненими конфігураціями.

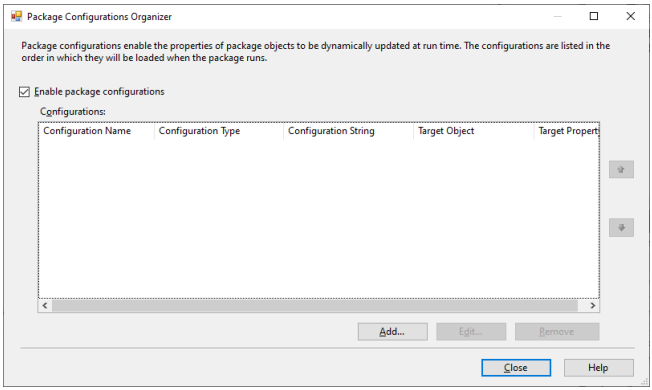

Рисунок 10.10 – Діалогове вікно *Package Configurations Organizer*

Створення конфігурації. Для створення нової конфігурації потрібно натиснути кнопку *Add* у діалоговому вікні *Package Configurations Organizer*, щоб запустити майстер *Package Configuration Wizard*. Спочатку потрібно вибрати відповідне значення для типу конфігурації зі списку, що розкривається (рис. 10.11). Служби *SSIS* підтримують конфігурації пакета різних типів; у таблиці 10.1 перелічено типи конфігурацій, які можна використовувати.

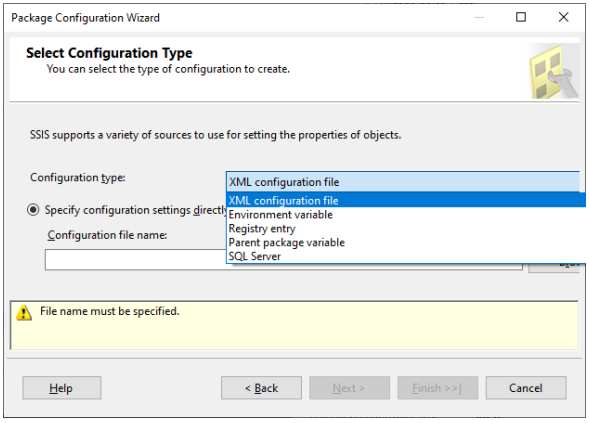

Рисунок 10.11 – Вибір типу конфігурації в майстрі Package Configuration Wizard

Таблиця 10.1 – Типи конфігурацій пакетів

Тип	Опис
XML configuration file (XML-файл конфігурації) (XML-файл конфігурації)	Зберігає налаштування конфігурації в XML-файлі, у файловій системі. В одному файлі конфігурації можна зберігати кілька конфігурацій
Environment variable (Змінна середовища)	Зберігає інформацію про конфігурацію в наборі системних глобальних змінних, які називаються змінними середовища (environment variable). У конфігурації змінної середовища можна зберігати лише одну властивість
Registry entry (Елемент реєстру)	Дає можливість зберігати властивості пакета та налаштування в реєстрі комп'ютера
Parent package variable (Змінна батьківського пакета)	Надає пакету спосіб успадкування значення змінної з батьківського пакета. Якщо пакет виконується з іншого пакета SSIS за допомогою завдання *Execute Package Task*, значення його змінних доступні дочірньому пакету завдяки конфігурації цього типу. З конфігурацією цього типу можна зберегти установку лише для однієї властивості пакета в конкретний момент часу
SQL Server	Зберігає налаштування конфігурації в таблиці SQL Server. В одній таблиці можна зберігати кілька конфігурацій

Створення XML-файлу конфігурації. Якщо ви вибираєте тип XML configuration file (XML-файл конфігурації), можна

задати розміщення вашого файлу конфігурації. Є два способи зазначення розміщення файлу.

– Ввести розміщення файлу безпосередньо в текстове поле *Configuration file name*. Застосовувати цей варіант доцільно, якщо маєте намір завжди використовувати одне й те саме місце зберігання та ім'я файлу конфігурації.

– Використовувати змінну середовища, яка містить розташування та ім'я файлу конфігурації. Для застосування цього підходу потрібно створити системну змінну у властивостях вашої операційної системи. Значення змінної повинне містити повний шлях, ім'я та розширення файлу.

Застосування змінної середовища для зазначення розміщення файлу називається *непрямою конфігурацією* (*indirect file location approach*); цей варіант потрібно використовувати, якщо місцезнаходження або ім'я XML-файлу можуть змінитися в майбутньому або вже змінюються під час переходу з одного середовища в інше.

Кілька пакетів можуть використовувати один і той самий XML-файл конфігурації, це стосується і конфігурацій інших типів. Якщо у вас є кілька пакетів зі спільними властивостями, такими як рядки з'єднання, всі вони можуть використовувати один XML-файл конфігурації.

Після того, як розміщення файлу та його ім'я визначено, визначаються налаштування та властивості сервера, які повинен містити XML-файл конфігурації. Оскільки ці характеристики загальні для всіх типів конфігурацій, перед описом серверних налаштувань і визначеннями властивостей у цьому розділі розглядається початкова установка конфігурації SQL Server.

Створення конфігурації SQL Server. Для збереження конфігурацій пакета в таблиці *SQL Server* у випадному списку *Configuration type* майстра *Package Configuration Wizard* виберіть варіант *SQL Server*. Вибір типу *SQL Server* як механізму зберігання для ваших конфігурацій потребуватиме завдання іншої групи налаштувань порівняно з іншими типами конфігурацій, такими як XML-файл конфігурації. На

рисунку 10.12 показано параметри конфігурації SQL Server, доступні для налаштування в конфігураціях.

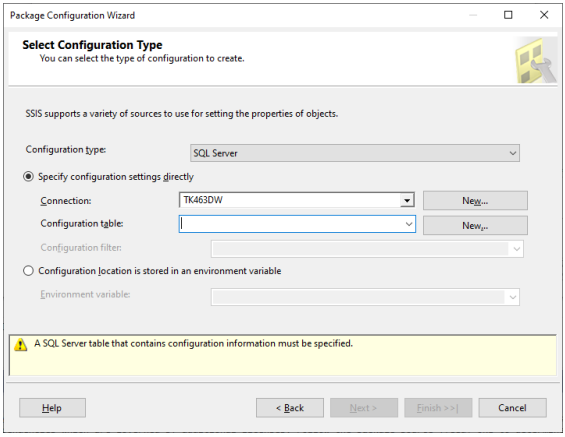

Рисунок 10.12 – Майстер *Package Configuration Wizard* для конфігурації у вигляді таблиці SQL Server

Як і у разі XML-файлу конфігурації, можна для зазначення розміщення конфігурації задати змінну середовища (наприклад, ім'я джерела даних для конфігурації SQL Server) або зазначити налаштування конфігурації явно.

Є три налаштування, які точно визначають розміщення таблиці.

– *Connection* (*З'єднання*). Це повинне бути з'єднання на основі SQL Server, яке задає сервер і базу даних, де буде зберігатися ваша конфігурація, і звідки вона буде читатися. Якщо потрібне з'єднання не визначено, можна натиснути кнопку *New* поруч зі списком *Connection*, щоб відкрити діалогове вікно *Configure OLE DB Connection Manager*.

– *Configuration table* (*Таблиця конфігурації*). Це ім'я таблиці, у якій будуть перебувати конфігурації. У таблиці заздалегідь визначено імена стовпців та їхні типи даних, які змінювати не можна. Для створення таблиці потрібно натиснути кнопку *New* поруч зі списком *Configuration table*, щоб відкрити діалогове вікно *Create Table*, у якому можна змінити ім'я

таблиці та виконати інструкцію створення таблиці для з'єднання, заданого в попередньому полі.

– *Configuration filter* (*Фільтр конфігурації*). Кілька конфігурацій SQL Server можуть зберігатися в одній таблиці, і вибрати потрібну конфігурацію можна за допомогою розкривного списку Configuration filter. Можна ввести новий фільтр або використовувати наявний. Ім'я, яке ви оберете або введете в поле цього списку, використовується як значення в стовпці Configuration Filter базової таблиці.

Вставлення властивостей у вашу конфігурацію. Незалежно від типу використовуваної конфігурації SSIS, на наступній сторінці майстра можна вибрати властивості для експорту (*Properties To Export*), тобто задати властивості пакета *SSIS* або об'єкта, які повинні використовуватися в конфігурації. Після того як визначені властивості для конфігурації в майстрі *Package Configuration Wizard*, необхідно натиснути кнопку *Next*.

У цей момент служби SSIS запропонують перевірити, чи є вже елементи конфігурації для обраного вами типу конфігурації. Якщо такі елементи є, їх можна повторно використовувати або перезаписати. У такому разі натисніть кнопку *Reuse Existing*. Якщо потрібно очистити наявні елементи конфігурації та створити нові, необхідно натиснути кнопку *Overwrite*.

Якщо в цій конфігурації ще немає елементів, або ви натиснули кнопку Overwrite, ви побачите сторінку *Select Properties to Export*, подібну на показану на рисунку 10.13.

На сторінці *Select Properties to Export* подано деревоподібну структуру властивостей пакета, що дає можливість зазначити властивості для обраної конфігурації SSIS. Властивості згруповано в такі папки:

– *Variables* – перелічено всі змінні пакета, які разом із їхніми властивостями можна вибрати як елементи конфігурації;

– *Connection Managers* – перелічено всі з'єднання пакета, з яких можна вибрати конкретні властивості для ваших з'єднань;

– *Log Providers* – дає можливість динамічно встановити конфігурацію протоколювання та реєстрації;

– *Properties* – відображає список властивостей рівня пакета, які можна використовувати для його налаштування;

– *Executables* – містить деревоподібну структуру ваших завдань і контейнерів. Переміщаючись по цьому дереву, можна налаштувати певні властивості ваших завдань і контейнерів.

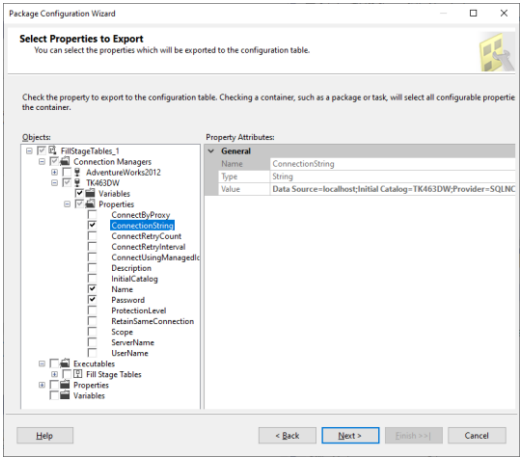

Рисунок 10.13 – Сторінка Select Properties to Export майстра Package Configuration Wizard

Якщо застосовується XML-файл конфігурації або конфігурації SQL Server чи елемент реєстру, можна відразу задати кілька властивостей у конфігурації, встановивши кілька прапорців властивостей.

Спільне використання, упорядкування та редагування конфігурації. Якщо у списку визначені кілька конфігурацій, то можна визначити порядок їх застосування в пакеті. Конфігурації викликаються в тому порядку, в якому вони перераховані у вікні *Package Configurations Organizer*. Це зауваження важливе, якщо є кілька конфігурацій, які оновлюватимуть одну й ту саму властивість, або якщо є конфігурації, що залежать від попередніх конфігурацій. Наприклад, є конфігурація, яка оновлює рядок з’єднання, який

потім використовується як місце розташування для елементів із наступної конфігурації. Необхідно мати на увазі, що останнє застосоване оновлення властивості буде тим значенням, яке використовується в пакеті.

Спільне використання конфігурацій різними пакетами – поширений підхід. Водночас можуть існувати елементи конфігурації, які застосовуються в одному пакеті і не використовуються в іншому. На виконання пакета це не впливає, але можна отримати попередження, яке повідомляє про те, що якоїсь властивості конфігурації в пакеті не існує.

Тестові питання

A. Які типи параметрів доступні в службах SSIS у SQL Server (Виберіть усі відповідні варіанти.):

- ☐ параметри рівня проєкту;
- ☐ параметри рівня розв'язку (solution);
- ☐ параметри рівня потоку керування;
- ☐ параметри пакета?

B. Які властивості можна задати за допомогою конфігурацій побудови (Виберіть усі відповідні варіанти.):

- ☐ значення параметрів;
- ☐ значення змінних;
- ☐ властивості завдання потоку керування;
- ☐ властивість Deployment Server Name (Ім'я сервера розгортання)?

C. Які властивості можна задати за допомогою виразів властивостей (Виберіть усі відповідні варіанти.):

- ☐ інструкцію SQL для завдання Execute SQL Task;
- ☐ значення змінних;
- ☐ властивості завдання потоку керування;
- ☐ властивість SqlCommand у перетворенні Lookup (Уточнювальний запит)?

D. Конфігурації яких типів можна використовувати для зберігання значень конфігурації (Виберіть усі відповідні варіанти.):

- ☐ XML configuration file (XML-файл конфігурації);
- ☐ SQL Server;

☐ будь-яка система реляційної бази даних;

☐ Registry entry (Елемент реєстру)?

E. Для яких об’єктів можна задати динамічні властивості за допомогою конфігурацій пакетів (Виберіть усі відповідні варіанти.):

☐ параметри;

☐ змінні;

☐ перетворення потоку даних;

☐ Sequence Container Task (Завдання контейнера послідовності?

F. Які елементи служб SSIS можна налаштувати за допомогою конфігурацій пакетів (Виберіть усі відповідні варіанти.):

☐ властивості з’єднання;

☐ властивості змінних пакета;

☐ властивості параметрів;

☐ властивості пакета?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Sarka D. Training Kit (Exam 70-463): Implementing a Data Warehouse with Microsoft SQL Server 2012 / D. Sarka, G. Jerkic, M. Lah // O'Reilly Media, 2012. – 484 p.

2. Integration Services (SSIS) Development and Management Tools [Electronic resource]. – Access mode : learn.microsoft.com/uk-ua/sql/integration-services/.

3. SQL Server Data Tools [Electronic resource]. – Access mode : https://learn.microsoft.com/uk-ua/sql/ssdt/sql-server-data-tools?view=sql-server-ver16.

4. Korotkevitch D. Expert SQL Server Transactions and Locking: Concurrency Internals for SQL Server Practitioners / D. Korotkevitch. – Apress, 2018. – 320 p. – ISBN 1484239571, 978148423957.

Електронне навчальне видання

Методичні вказівки

до самостійної роботи
з дисципліни «**Сховища даних**»
для здобувачів спеціальності *122 «Комп'ютерні науки»*
освітнього ступеня ***«магістр»***
усіх форм здобуття вищої освіти

Відповідальна за випуск В. В. Шендрик
Редакторка Н. З. Клочко
Комп'ютерне верстання А. В. Нені

Формат 60×84/16. Ум. друк. арк. 10,75. Обл.-вид. арк. 10,45.

Видавець і виготовлювач
Сумський державний університет,
вул. Харківська, 116, м. Суми, 40007
Свідоцтво про внесення суб'єкта господарювання до Державного реєстру видавців,
виготовлювачів та розповсюджувачів видавничої продукції ДК № 8193 від 15.10.2024.