Data Warehouse

Faculty of Computer Science

ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN

VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY

한-베정보통신기술대학교

Chapter 6: Extraction, Transformation, and Loading (ETL)

Chapter Content

- Business Process Modeling Notation
- Conceptual ETL Design using BPMN
- Conceptual Design of the Northwind ETL process
- Integration Services and Kettle
- The Northwind ETL process in Integration Services
- The Northwind ETL process in Kettle

Extraction, Transformation, and Loading (ETL)

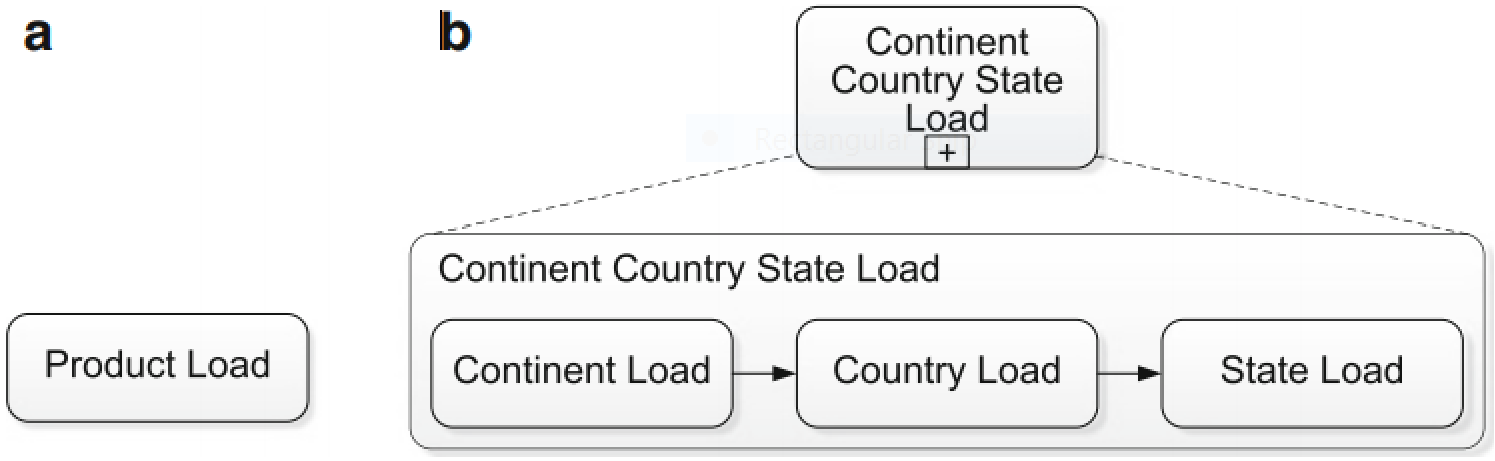
- Extract data from internal and external sources, transform data, and load data into a data warehouse
- No agreed way to specify ETL at a conceptual level
- We study conceptual ETL design
- Conceptual model based on the Business Process Modeling Notation (BPMN)
  - Users already familiar with BPMN do not need to learn another language to design ETL
  - BPMN provides a conceptual and implementation-independent specification of processes
  - Processes expressed in BPMN can be translated into executable specifications(e.g., Microsoft's Integration Services)

Business Process Modeling Notation (BPMN)

- Business process: A collection of related activities or tasks whose goal is to produce a specific service or product
- Business process modeling: Activity of representing the business processes of an organization, so that the current processes may be analyzed and improved
- Many techniques to model business process proposed over the years
- No formal semantics for these techniques
- Formal techniques (e.g., Petri Nets): Well-defined semantics but hard to understand by business users
- A standardization process resulted in the Business Process Modeling Notation (BPMN) released by the Object Management Group (OMG). Current version is BPMN 2.0
- BPMN: Graphical notation for defining, understanding, and communicating the business procedures of an organization them in a standard manner
- Four basic categories of elements: flow objects, connecting objects, swimlanes, and artifacts

Flow Objects: Activities

- An activity is a work performed during a process
  - Can be atomic or nonatomic
  - Can be a task or a subprocess
- Subprocess: An encapsulated process whose details we want to hide

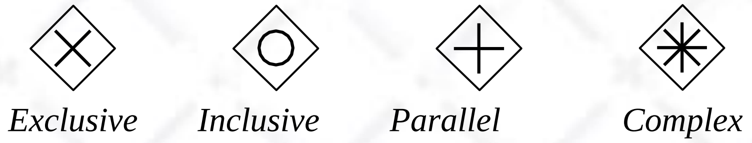


Activities. (a) Single task. (b) Collapsed and expanded subprocess

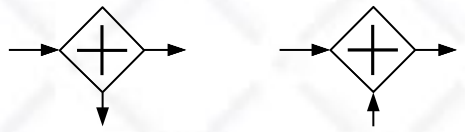
Flow Objects: Gateways

- Control the activity sequence in a process, based on conditions
- Represent only logic, not activities
- Exclusive gateways model OR-split decisions
- Inclusive gateways select or merge one or more flows
- Parallel gateways allow the synchronization between outgoing and incoming flows
  - Splitting parallel gateway: Analogous to an AND-split
  - Merging parallel gateway: Synchronizes the flow and merges all the incoming flows into a single outgoing one
- Complex gateways can represent complex conditions

Different types of gateways



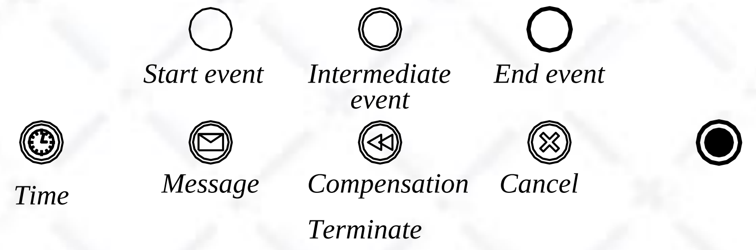
Splitting and merging gateways



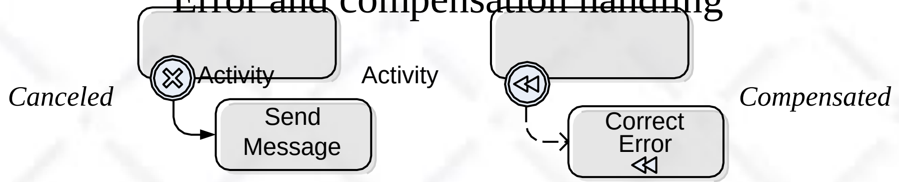
Flow Objects: Events

- Represent something that happens that affects the sequence and timing of the workflow activities
- Start and end events indicate the beginning and ending of a process
- Time event: represents situations when a task must wait for some period of time before continuing
- Message event represents communication
- Compensation event represents error detection and recovery by launching compensation activities
- Cancel event listens to the process errors and notifies them by an explicit or implicit action
- Terminate When it is reached, the entire process is stopped, including all parallel processes.

Examples of events

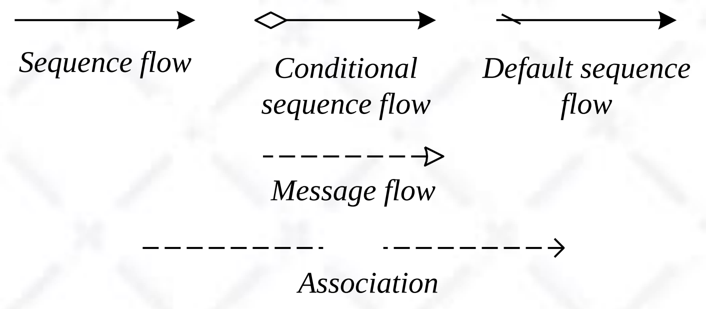


Error and compensation handling



Connecting Objects

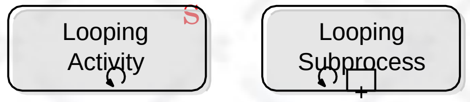
- Represent how objects are connected
- Sequence flow: A sequencing constraint between flow objects
  - If two activities are linked by a sequence flow, the target one starts when the source one has finished
  - If multiple sequence flows outgo from a flow object, all of them will be activated after its execution
- Conditional sequence flow: Adds a condition to the sequence flow
- A sequence flow may be set as the default flow in case of many outgoing flows (e.g., if no other condition is true in a gateway, the default flow is followed
- A message flow represents the only way of sending and receiving of messages between pools An association relates artifacts (e.g., annotations) to flow objects.



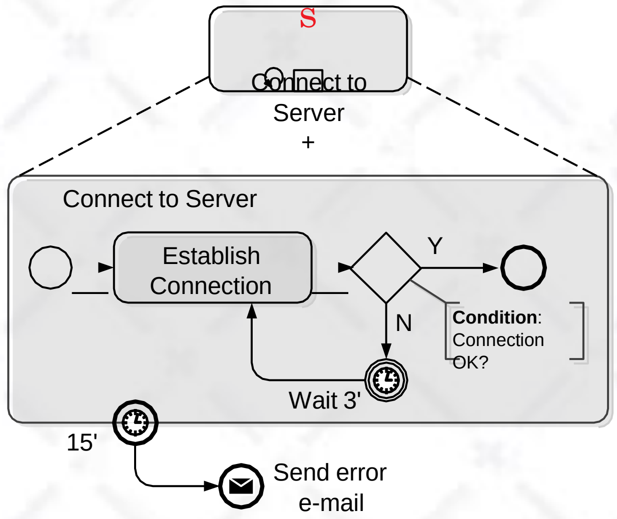
Loops and Subprocesses

- Loop: Execution control feature representing repeated execution of a process
- Conditions checked before or after activity. Loop ended if its condition evaluates to false.
- Figure: ETL process representing the connection to a server task
- At a high abstraction level, the subprocess activity hides the details
- Expansion shows details: server waits 3 minutes (time event). If connection not established, request launched again. If no connection after 15 minutes, task stopped, and error email sent (message event).

Loops

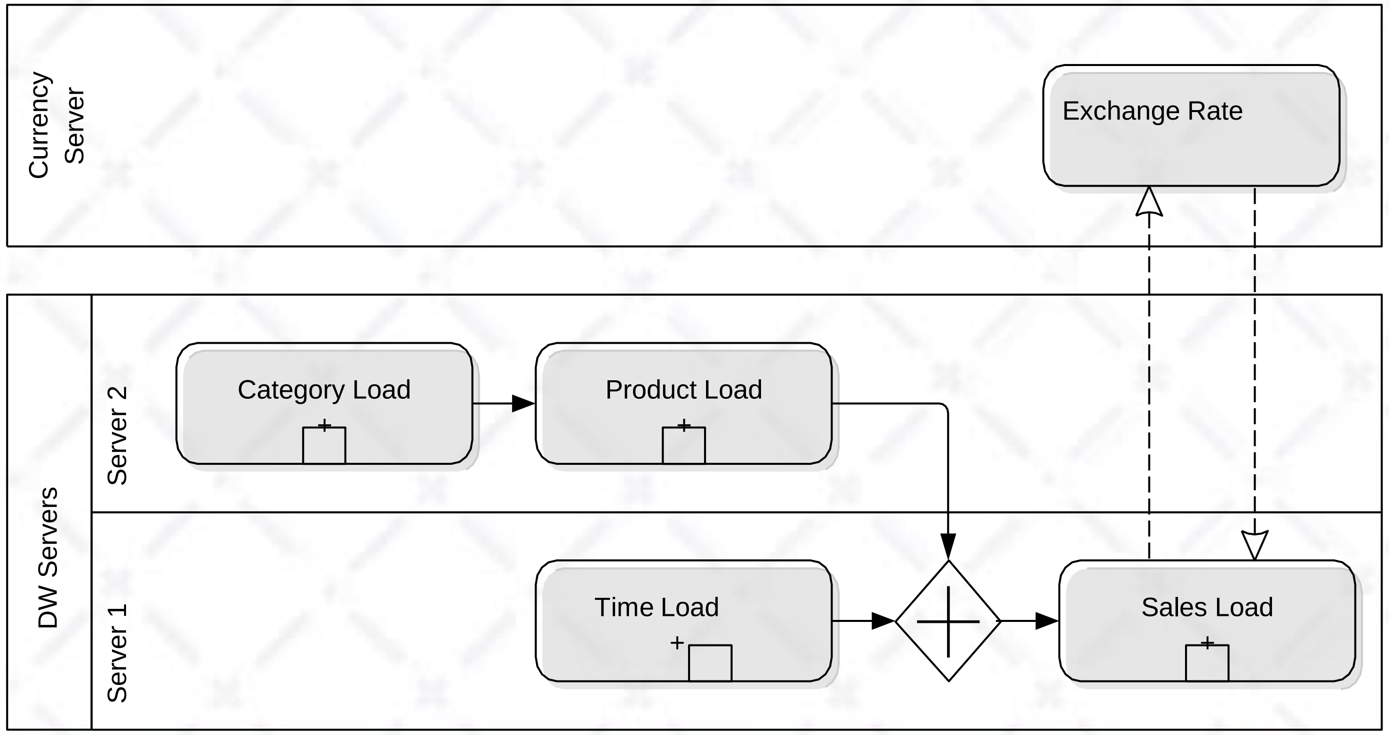


Subprocesses



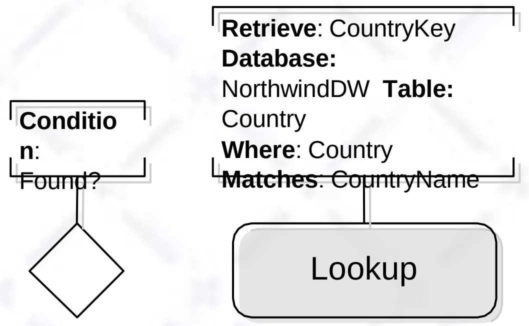
Swimlanes

- A structuring object that comprises pools and lanes
- Both allow the definition of process boundaries
- Only messages allowed between two pools, not sequence flows
- A workflow must be contained in only one pool
- One pool may be subdivided into many lanes, which represent roles or services



Artifacts

- Allow to visually represent objects outside the actual process
- Can represent data or notes that describe the process, or they can be used to organize tasks or processes
- Can be data objects, groups, and annotations
- A data object represents either data that are input to a process, data resulting from a process, data that needs to be collected, or data that needs to be stored.
- A group organizes tasks or processes that have some kind of significance in the overall model
- Annotations are used to express semantics about the flow objects (e.g., to indicate the attributes involved in a lookup task, or a gateway condition)


Conditio n: Found?
Retrieve: CountryKey **Database:** NorthwindDW **Table:** Country **Where**: Country **Matches**: CountryName
Lookup

Chapter 6: Extraction, Transformation, and Loading

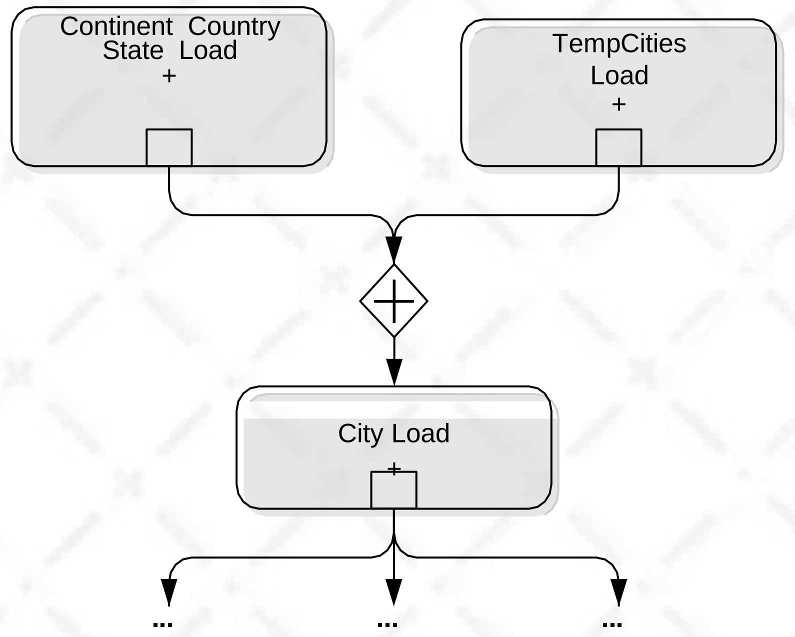
Outline

Conceptual ETL Design using BPMN

- Basic assumption for using BPMN as conceptual model: ETL process is a type of business process
- There is no standard model for defining ETL processes
- Each tool provides its own model, too detailed to be conceptual
- Using BPMN constructs we define the most common ETL tasks and define a BPMN notation for ETL
- ETL process: A combination of control and data processes
  - Control processes manage the coarse-grained groups of tasks
  - Data processes detail how input data are transformed and output data are produced
- Two kinds of tasks in ETL conceptual modeling
  - Control tasks highlight the control procedures provided by BPMN. Represent a workflow (arrows represent the precedence between activities)
  - Data tasks refer to the tasks that directly manipulate data during an ETL process. Represent a data flow (arrows represent data ‘flowing’ along them)

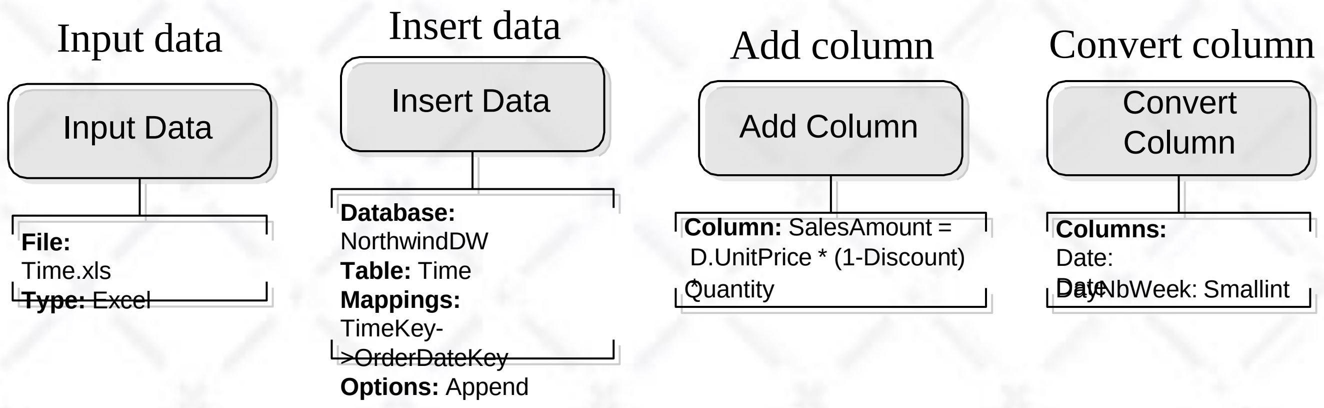
Control Tasks

- Represent the workflow sequence or orchestration of the ETL process independently of the data flow
- Control tasks are represented by means of BPMN constructs described
- For example, gateways are used to control the sequence of activities in an ETL process
- The most used types of gateways in an ETL context are exclusive and parallel

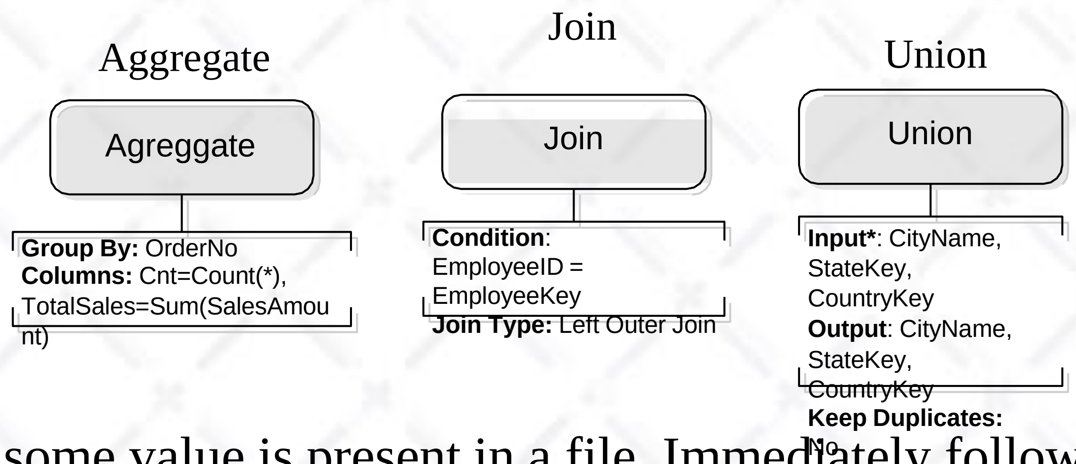


Data Tasks

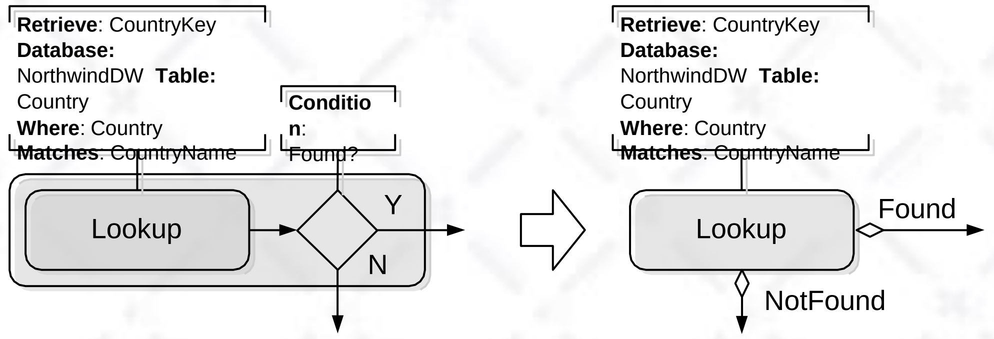
- Show how data are manipulated within an activity
- At lower abstraction level than control tasks
- Represent activities typically carried out to manipulate data: input and output data, data conversion and transformation (for instance, change the data type of an attribute, add a column, remove duplicates, and so on)
- We denote these tasks unary data tasks since they receive one input flow
- *n*-ary data tasks receive as input more than one flow (e.g., this is the case of union, join, difference,...)
- Row operations are transformations applied to the source or target data on a row-by-row basis, e.g., updating the value of a column
- Rowset operations deal with a set of rows, e.g., aggregation is a rowset operation



Rowset Data Tasks


Aggregate
Agreggate
Group By: OrderNo
Columns: Cnt=Count(*), TotalSales=Sum(SalesAmount)
Join
Join
Condition: EmployeeID = EmployeeKey
Join Type: Left Outer Join
Union
Union
Input*: CityName, StateKey, CountryKey
Output: CityName, StateKey, CountryKey
Keep Duplicates: No

Lookup Data Tasks check if some value is present in a file. Immediately followed by an exclusive gateway with a branching condition. We use a shorthand replacing these two tasks by 2 conditional flows.


Retrieve: CountryKey
Database: NorthwindDW Table: Country
Where: Country Matches: CountryName
Condition: Found?
Lookup
Y
N
Retrieve: CountryKey
Database: NorthwindDW Table: Country
Where: Country Matches: CountryName
Lookup
Found
NotFound

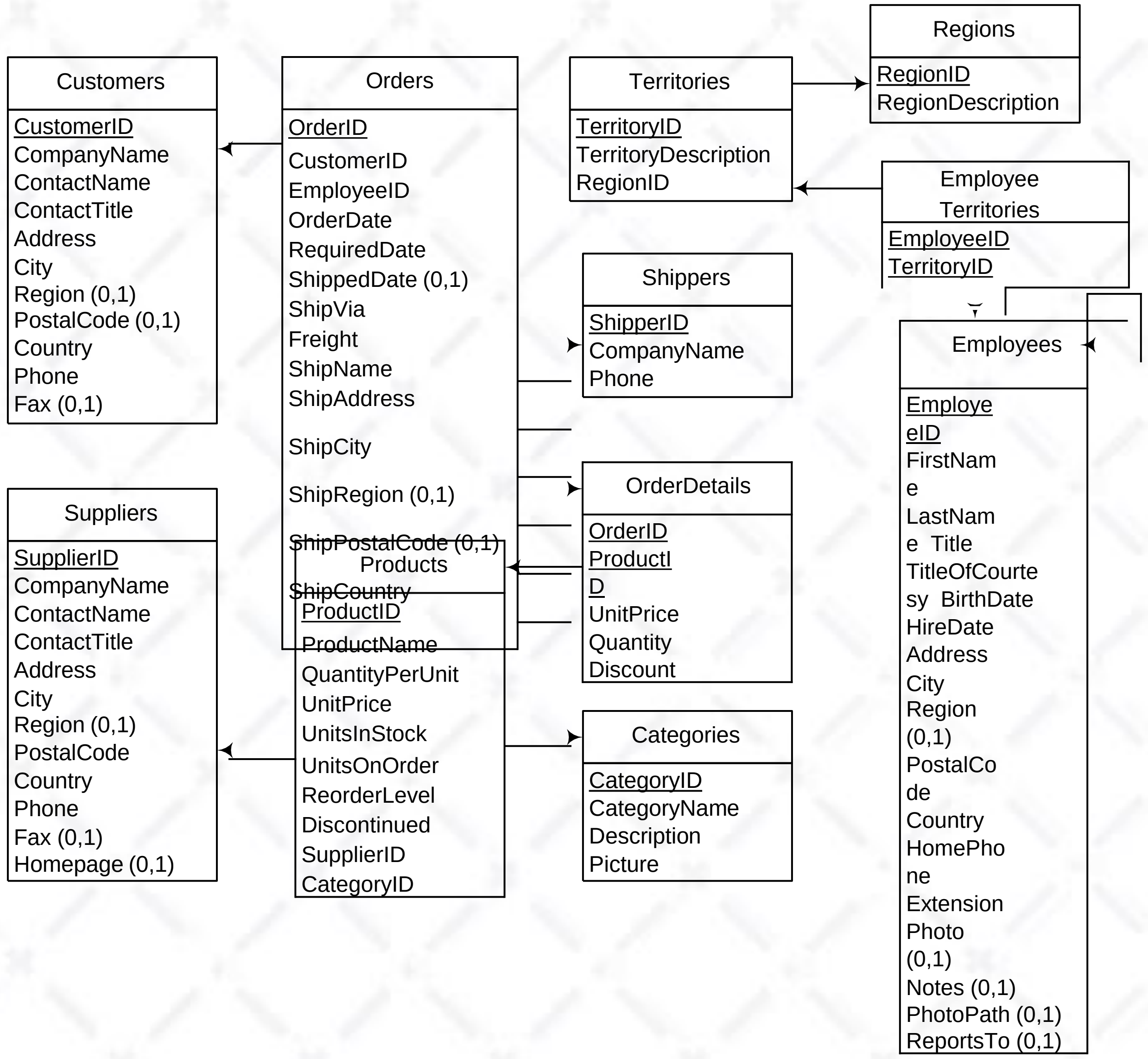
Shorthand notation for the lookup task

Chapter 6: Extraction, Transformation, and Loading

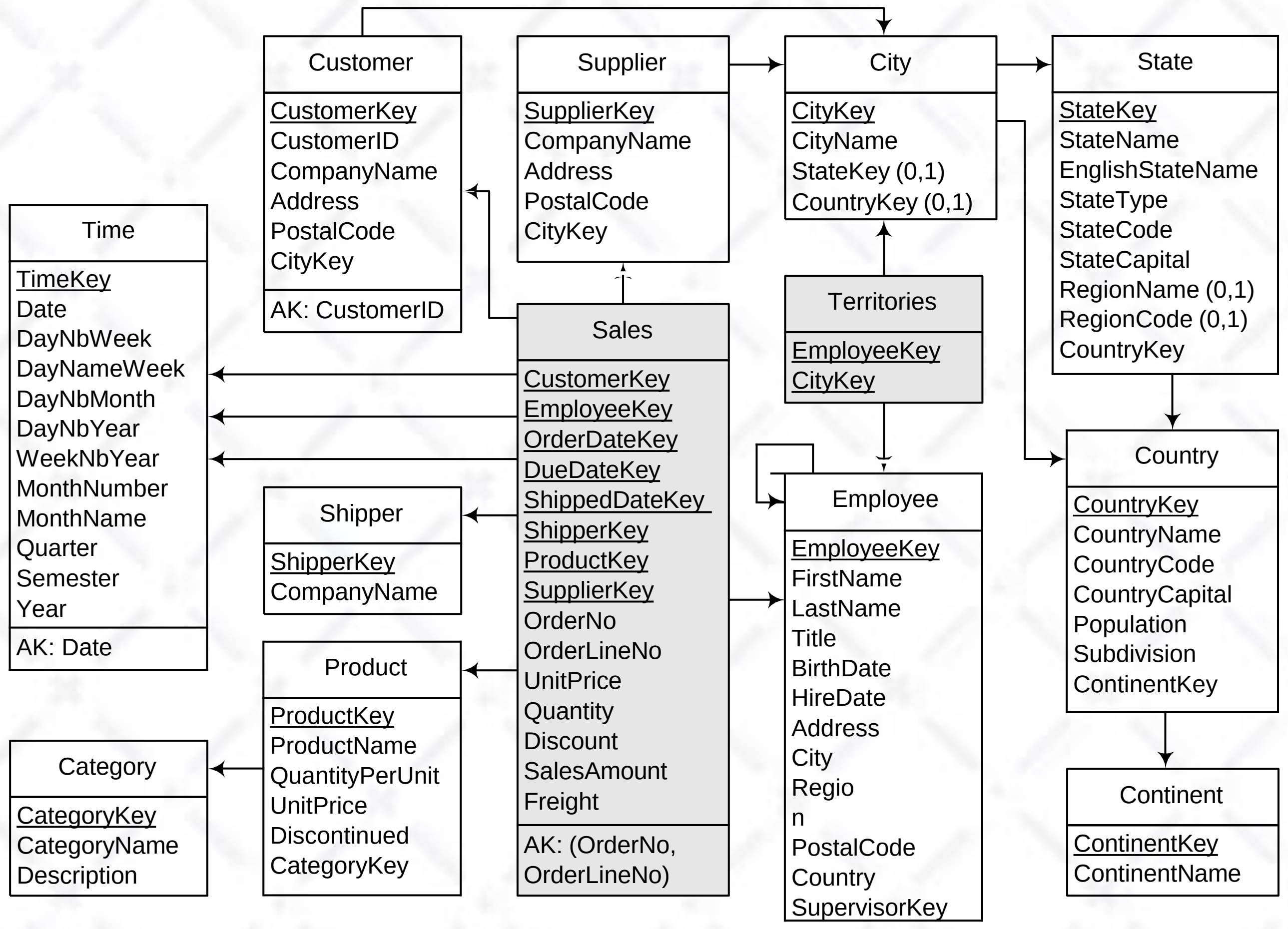
Outline

- ◆Business Process Modeling Notation
- ◆Conceptual ETL Design using BPMN
- ② Conceptual Design of the Northwind ETL process
- ◆Integration Services and Kettle
- ◆The Northwind ETL process in Integration Services
- ◆The Northwind ETL process in Kettle

Schema of the Northwind Operational Database



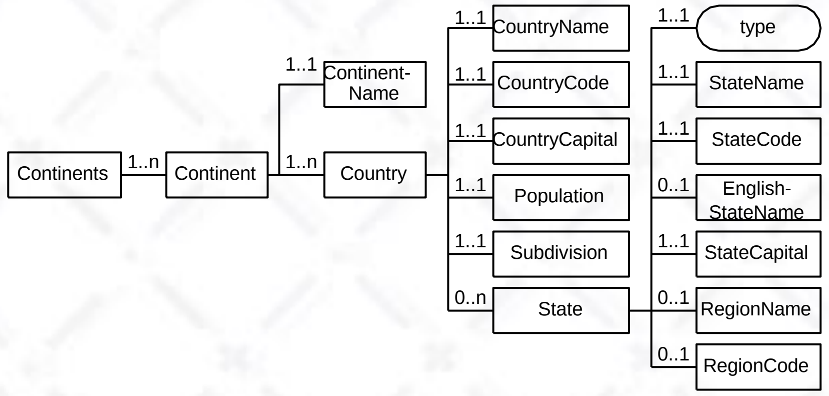
Schema of the Northwind Data Warehouse



Conceptual Design of the Northwind ETL: Data Sources

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Continents>
    <Continent>
                  <ContinentName>Europe</ContinentName>
        <Country>
            <CountryName>Austria</CountryName>
            <CountryCode>AT</CountryCode>
            <CountryCapital>Vienna</CountryCapital>
            <Population>8316487</Population>
            <Subdivision>Austria is divided into nine Bundeslnder,  or simply Lnder
                (states; sing. Land).</Subdivision>
            <State type="state">
                <StateName>Burgenland</StateName>
                <StateCode>BU</StateCode>
                <StateCapital>Eisenstadt</StateCapital>
            </State>
            <State type="state">
                <StateName>Krnten</StateName>
                <StateCode>KA</StateCode>
                <EnglishStateName>Carinthia</EnglishStateName>
                <StateCapital>Klagenfurt</StateCapital>
            </State>
            ...
```

- File Time.xls contains data for loading the Time dimension, spanning the dates in table Orders of the operational database
- Dimensions Customer and Supplier share the geographic hierarchy starting at the City level
- Data for the hierarchy State → Country → Continent loaded from Territories.xml



XML Schema of Territories.xml

Start of the file Territories.xml

Conceptual Design of the Northwind ETL: Data Sources

- File called Cities.txt identifies to which state or province a city belongs
- Contains three fields separated by tabs and begins as shown below
- For cities located in countries that do not have states (e.g., Singapore), second field is set to null
- The file is also used to identify to which state corresponds the city in the attribute TerritoryDescription of table Territories

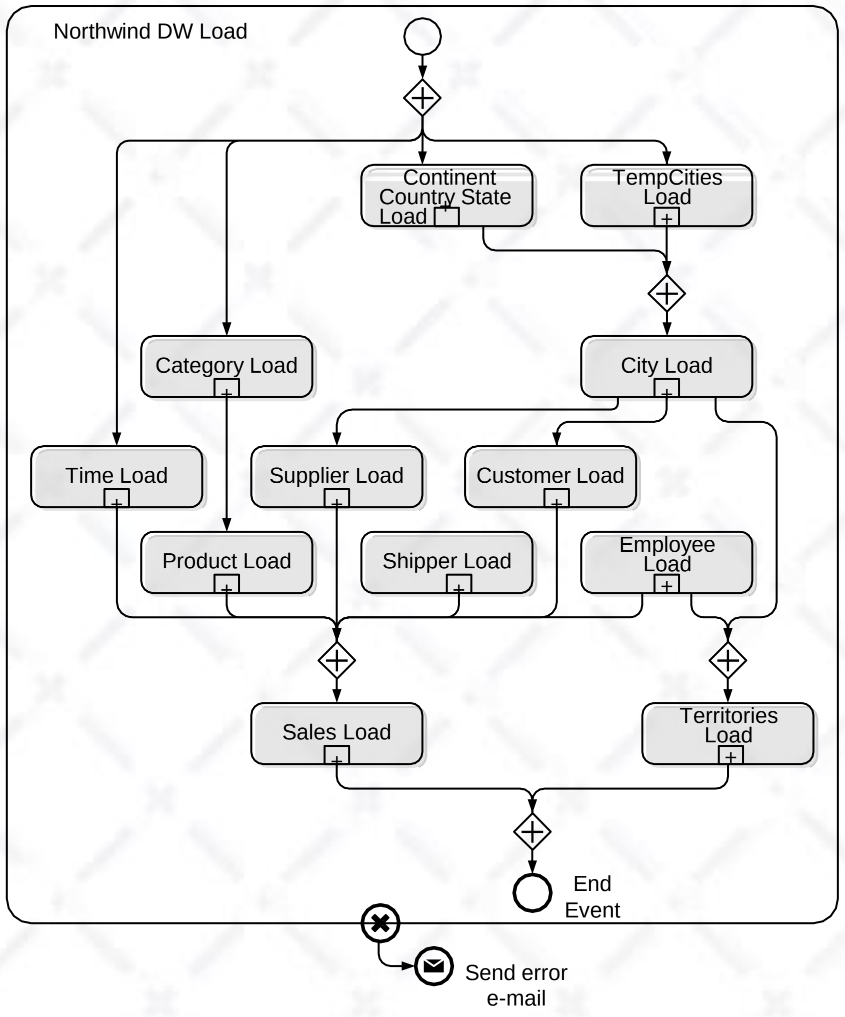
```
City → State → Country
Aachen → North Rhine-Westphalia → Germany
Albuquerque → New Mexico → USA
Anchorage  → Alaska → USA
Ann Arbor → Michigan → USA
Annecy → Haute-Savoie → France
...
```

Begining of the file Cities.txt

TempCities
City State Country

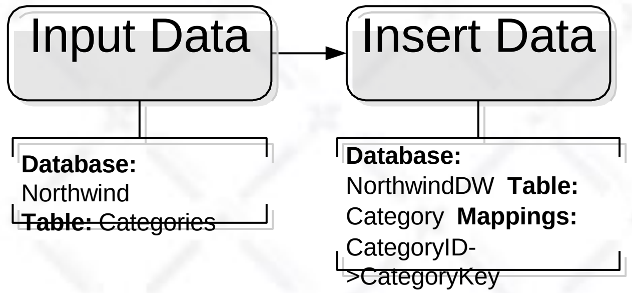
Associated table TempCities

Conceptual Design of the Northwind ETL: Overall View



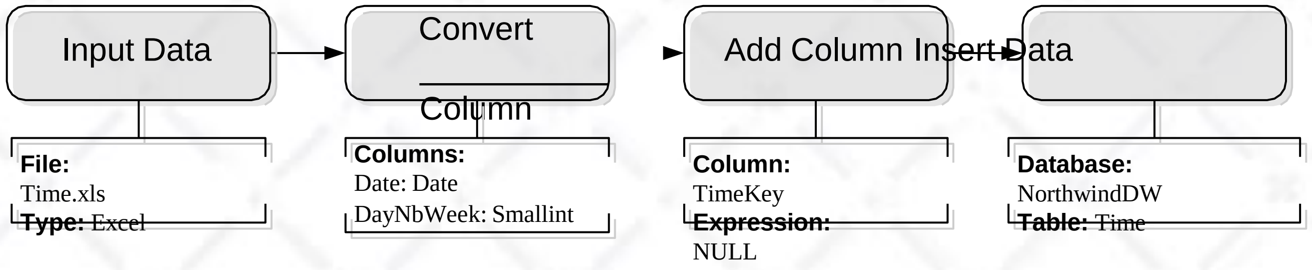
Conceptual Design of the Northwind ETL

- Load of the Category dimension table


Input Data
Insert Data
Database: Northwind
Table: Categories
Database: NorthwindDW Table: Category Mappings: CategoryID->CategoryKey

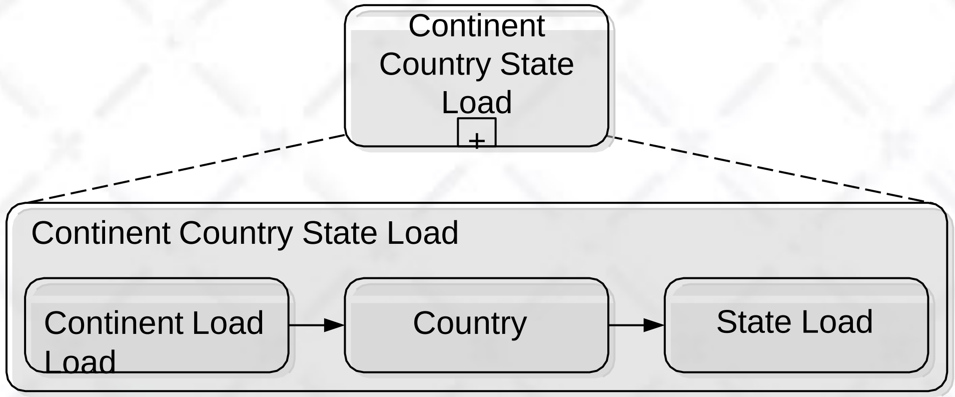
  - Input task loads table Categories from the operational database
  - Insert task loads the table Category in the data warehouse, mapping CategoryID to CategoryKey attribute in the Category table

- Loading the Time dimension table from an Excel file is similar, but includes a data type conversion, and an addition of the column TimeKey

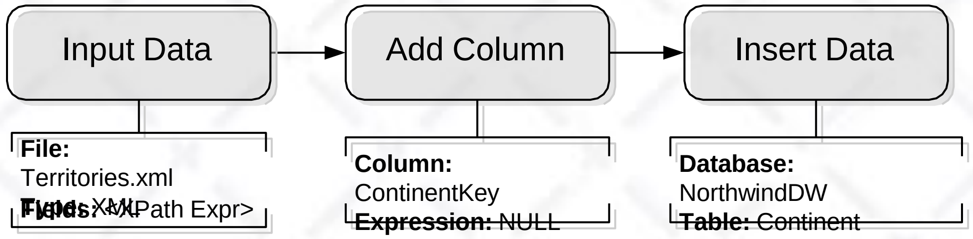


Conceptual Design of the Northwind ETL

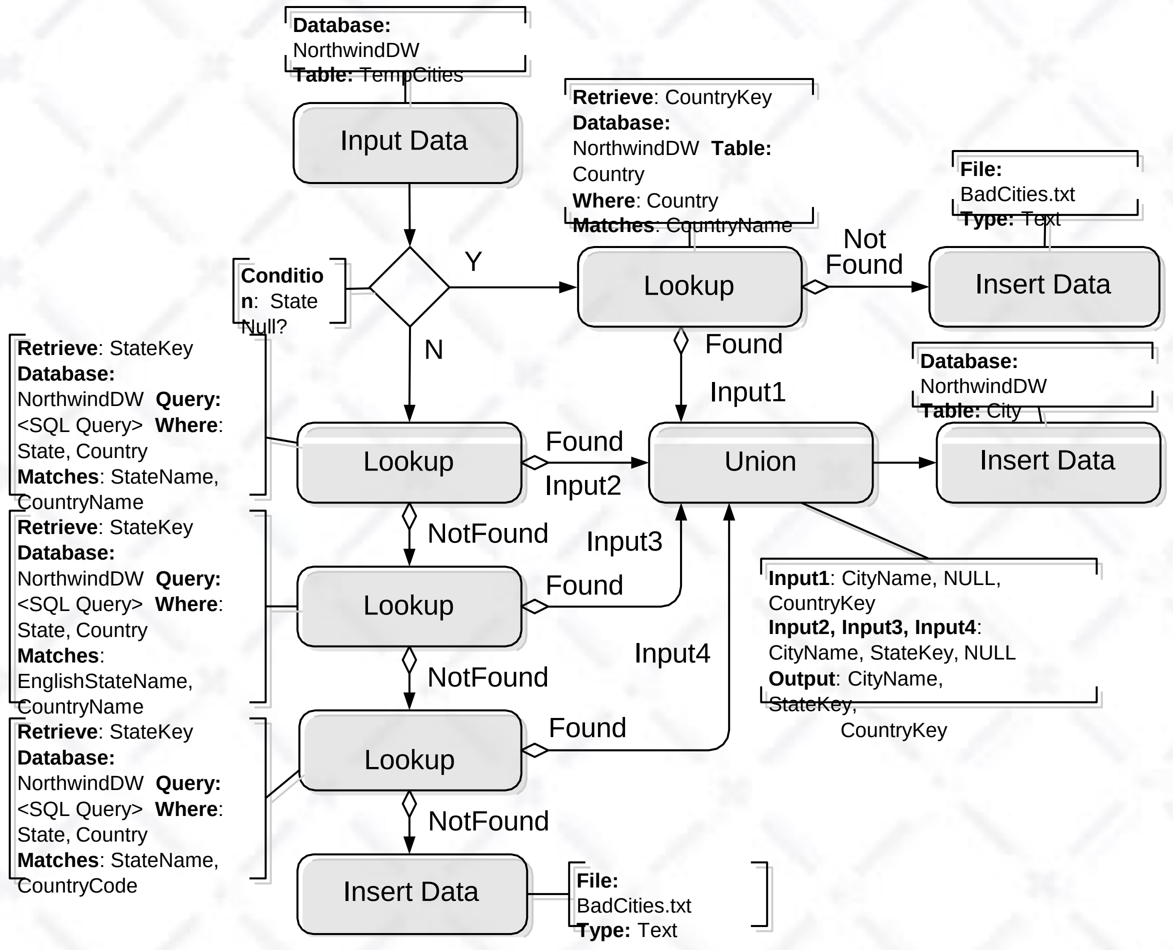
- Loading the City level first requires loading the Geography hierarchy State → Country → Continent
- Associated control task



- Load of the Continent table



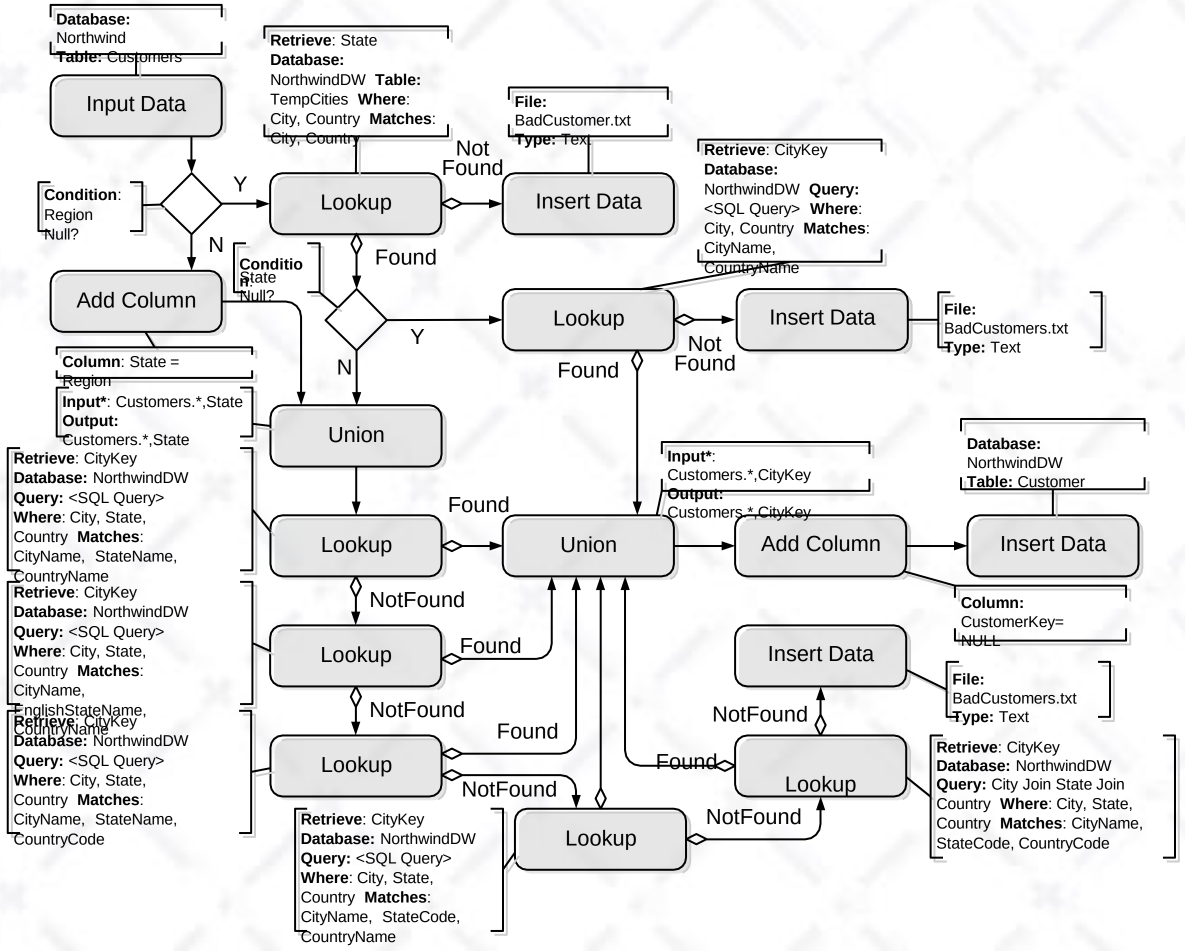
Load of the City Level


Database: NorthwindDW Table: TempCities
Input Data
Condition: State Null?
Y
N
Retrieve: CountryKey Database: NorthwindDW Table: Country Where: Country Matches: CountryName
Lookup
Not Found
File: BadCities.txt Type: Text
Insert Data
Found
Input1
Database: NorthwindDW Table: City
Insert Data
Retrieve: StateKey Database: NorthwindDW Query: <SQL Query> Where: State, Country Matches: StateName, CountryName
Lookup
Found
Input2
Union
NotFound
Input3
Retrieve: StateKey Database: NorthwindDW Query: <SQL Query> Where: State, Country Matches: EnglishStateName, CountryName
Lookup
Found
Input4
Input1: CityName, NULL, CountryKey
Input2, Input3, Input4: CityName, StateKey, NULL
Output: CityName, StateKey, CountryKey
NotFound
Retrieve: StateKey Database: NorthwindDW Query: <SQL Query> Where: State, Country Matches: StateName, CountryCode
Lookup
Found
NotFound
Insert Data
File: BadCities.txt Type: Text

Load of the City Level

- Assume that a table TempCities(City,State,Country) has been created and populated from Cities.txt
- First task is an input data over TempCities
- An exclusive gateway tests whether State is null or not
  - If so, lookup obtains the CountryKey
  - If not, we match (State, Country) pairs in TempCities to values in the State and Country tables
- Finally, union performed with the results of the four flows, and table is loaded with an insert data task
- Records for which the state and/or country are not found are stored into a BadCities.txt file.

Load of the Customer Level



Database: Northwind Table: Customers
Input Data
Condition: Region Null?
Y
N
Add Column
Column: State = Region
Input*: Customers.*,State Output: Customers.*,State
Retrieve: State Database: NorthwindDW Table: TempCities Where: City, Country Matches: City, Country
Lookup
Not Found
File: BadCustomer.txt Type: Text
Insert Data
Found
Condition: State Null?
Y
N
Retrieve: CityKey Database: NorthwindDW Query: <SQL Query> Where: City, Country Matches: CityName, CountryName
Lookup
Not Found
Insert Data
File: BadCustomers.txt Type: Text
Found
Union
Retrieve: CityKey Database: NorthwindDW Query: <SQL Query> Where: City, State, Country Matches: CityName, StateName, CountryName
Lookup
Found
Union
Input*: Customers.*,CityKey Output: Customers.*,CityKey
Add Column
Column: CustomerKey= NULL
Database: NorthwindDW Table: Customer
Insert Data
NotFound
Retrieve: CityKey Database: NorthwindDW Query: <SQL Query> Where: City, State, Country Matches: CityName, EnglishStateName, CountryName
Lookup
Found
NotFound
Retrieve: CityKey Database: NorthwindDW Query: <SQL Query> Where: City, State, Country Matches: CityName, StateName, CountryCode
Lookup
Found
NotFound
Retrieve: CityKey Database: NorthwindDW Query: <SQL Query> Where: City, State, Country Matches: CityName, StateCode, CountryName
Lookup
NotFound
Found
Lookup
Retrieve: CityKey Database: NorthwindDW Query: City Join State Join Country Where: City, State, Country Matches: CityName, StateCode, CountryCode
NotFound
Insert Data
File: BadCustomers.txt Type: Text

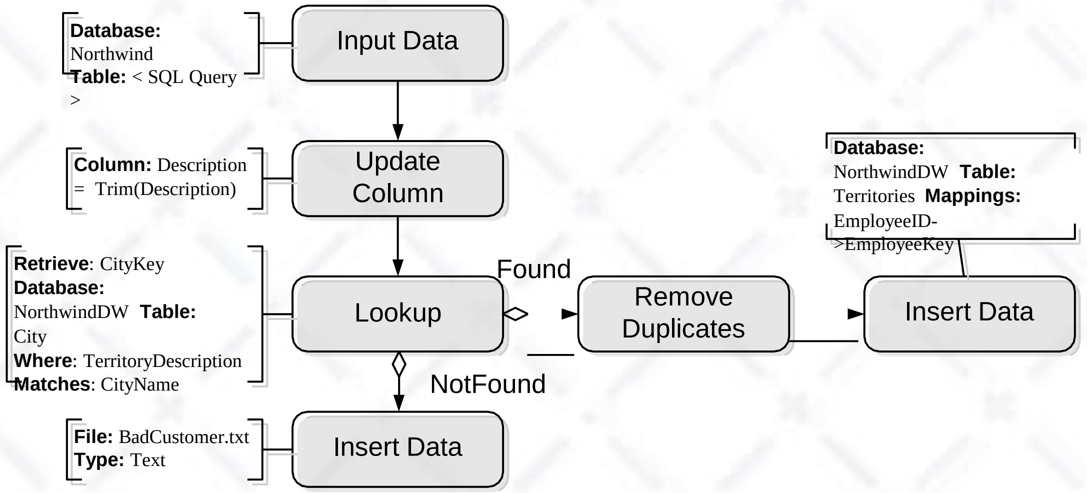
Load of the Customer Level

- The input table Customers is read from the operational database using an input data task
- Region (optional) in Customers is actually a state name or a state code → the first exclusive gateway checks whether this attribute is null or not
  - If Region is not null, add new column State initialized with the values of Region
  - Otherwise, check if the (City, Country) pair matches a pair in TempCities, and retrieve the State

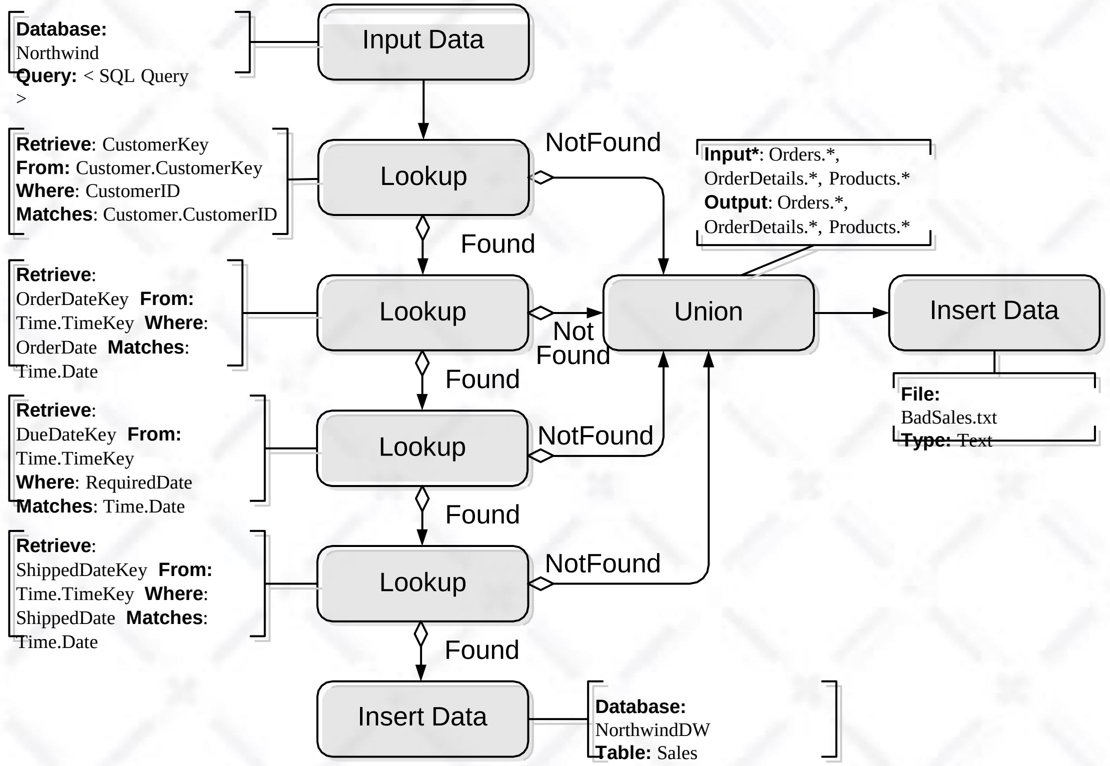
    attribute, creating a new column
- A second exclusive gateway over the new State column accounts for countries without states
- Then perform a union over the two flows
- Finally, perform the union of all flows, and add the column CustomerKey for the surrogate key ini- tialized to null

Load of the Territories Bridge Table

- The input is an SQL query joining EmployeeTerritories and Territories
- Then, an update column task removes ('trims') the leading spaces from attribute TerritoryDescription
- The city key is then obtained with a lookup over City in the D
- Finally, Territories is populated with an insert data task


Database: Northwind Table: < SQL Query >
Input Data
Column: Description = Trim(Description)
Update Column
Retrieve: CityKey Database: NorthwindDW Table: City Where: TerritoryDescription Matches: CityName
Lookup
Found
Remove Duplicates
Database: NorthwindDW Table: Territories Mappings: EmployeeID->EmployeeKey
Insert Data
NotFound
File: BadCustomer.txt Type: Text
Insert Data

Load of the Sales Fact Table



Database: Northwind
Query: < SQL Query >
Input Data
NotFound
Lookup
Retrieve: CustomerKey
From: Customer.CustomerKey
Where: CustomerID
Matches: Customer.CustomerID
Found
Input*: Orders.*, OrderDetails.*, Products.*
Output: Orders.*, OrderDetails.*, Products.*
Retrieve: OrderDateKey From: Time.TimeKey Where: OrderDate Matches: Time.Date
Lookup
Not Found
Union
Insert Data
Found
File: BadSales.txt
Type: Text
Retrieve: DueDateKey From: Time.TimeKey
Where: RequiredDate
Matches: Time.Date
Lookup
NotFound
Found
Retrieve: ShippedDateKey From: Time.TimeKey Where: ShippedDate Matches: Time.Date
Lookup
NotFound
Found
Insert Data
Database: NorthwindDW
Table: Sales

Load of the Sales Fact Table

- Task performed once all the other ones done
- Columns for order line number, sales amount, and freight must be created (Add Column data tasks)
- The process starts with an input data task that obtains data from the operational database via the query:

```
SELECT O.CustomerID, EmployeeID AS EmployeeKey, O.OrderDate,
          O.RequiredDate AS DueDate, O.ShippedDate,
          ShipVia AS ShipperKey, P.ProductID AS ProductKey,  P.SupplierID AS
          SupplierKey, O.OrderID AS OrderNo,  ROW NUMBER() OVER
          (PARTITION BY D.OrderID
          ORDER BY D.ProductID) AS OrderLineNo, D.UnitPrice, Quantity, Discount,  D.UnitPrice
          * (1-Discount) * Quantity AS SalesAmount,  O.Freight/COUNT(*) OVER (PARTITION BY
          D.OrderID) AS Freight
FROM      Orders O, OrderDetails D, Products P
WHERE  O.OrderID = D.OrderID AND D.ProductID = P.ProductID
```

- A sequence of lookups follows, which obtains the missing foreign keys for the dimension tables
- Finally, the fact table is loaded with the data retrieved

Chapter 6: Extraction, Transformation, and Loading

Integration Services

- SQL Server component to perform data migration tasks, and implement and execute ETL processes
- Components of Integration Services
  - Package: A workflow containing a collection of tasks executed in an orderly fashion
  - A package consists of a control flow and, optionally, one or more data flows
  - Control flow: three kinds of elements
    - Tasks: Individual units of work that provide functionality to a package
      - Tasks: data flow tasks, data preparation tasks, Analysis Services tasks, workflow tasks
    - Containers: Group tasks logically into units of work, and are used to define variables and events
      - Ex: Sequence Container and For Loop Container
    - Precedence constraints: Connect tasks, containers, and executables defining execution order
- Creating a control flow in Integration Services requires:
  - Adding containers
  - Adding tasks
  - Connecting containers and tasks, using precedence constraints
  - Adding connection managers, when a task connects to a data source

Integration Services: Data Flows

- Extract data into memory, transform them, and write them to a destination
- Three kinds of components:
  - Sources: Extract data from data stores (OLE DB data sources, Excel files, flat files, and XML files, among other)
  - Transformations: Modify, summarize, and clean data (split, divert, or merge the flow)
    - Example: Conditional Split, Copy Column, and Aggregate.
  - Destinations: Load data into data stores or create in-memory datasets
- Creating a data flow includes the following steps
  - Adding one or more sources
  - Adding the transformations to satisfy the package requirements
  - Connecting data flow components
  - Adding one or more destinations to load data into data stores
  - Configuring error outputs
  - Including annotations to document the data flow

Kettle

- Main components:
  - Transformations: Logical tasks consisting in steps connected by hops, essentially data flows to extract, transform, and load data
    - Steps: Perform a specific tasks, e.g., reading data from a file, filtering rows, writing to a database
      - Steps grouped according to their function, such as input, output, scripting, etc.
    - Hops: Data paths connecting steps to each other, so records can pass from one step to another
  - Jobs: Workflows that orchestrate the individual pieces of functionality implementing an entire ETL process
  - Jobs are composed of:
    - Jobs entries: Primary building blocks of a job, correspond to the steps in data transformations
    - Jobs hops: Specify the execution order of job entries and the conditions
    - Jobs settings: Options that control the behavior of a job and the logging method
- Important: loops are not allowed in transformations, but allowed in jobs

Kettle

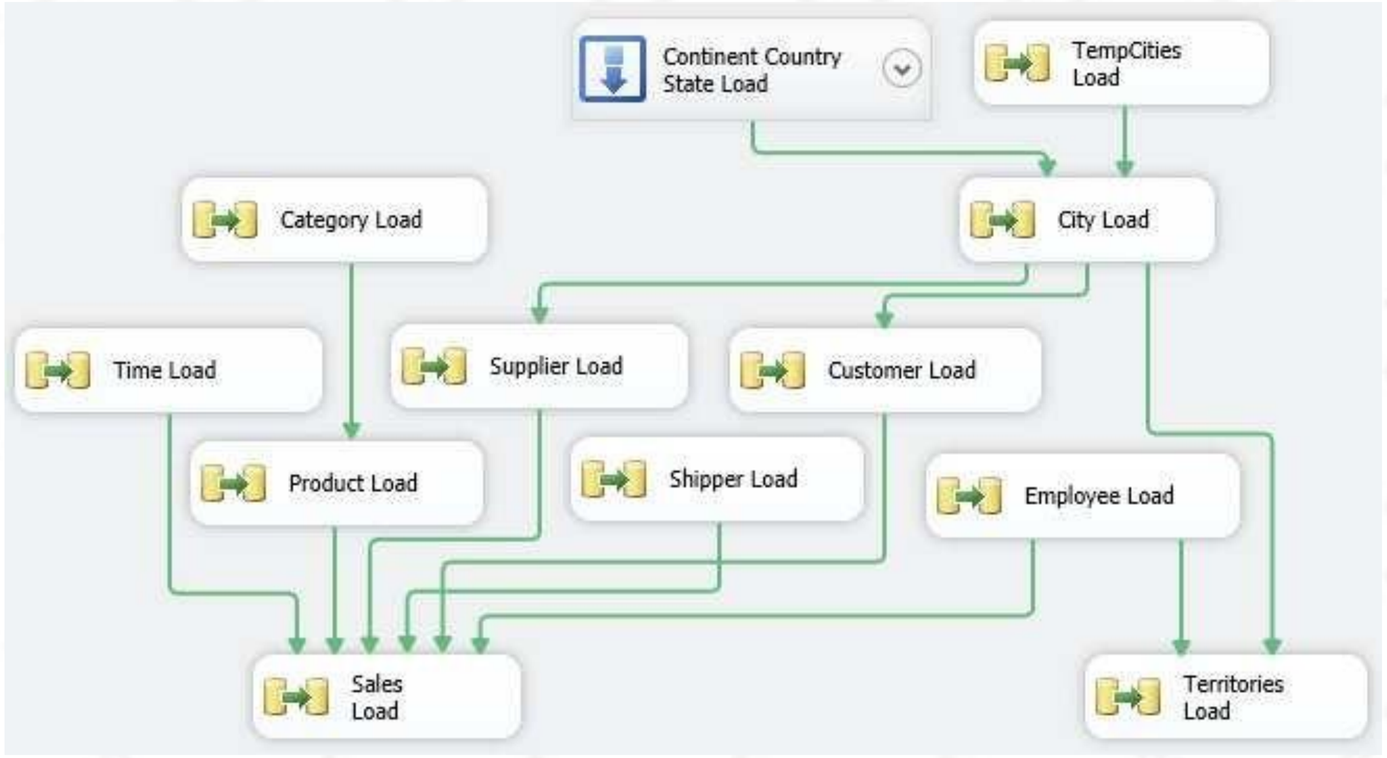
- Kettle is composed of the following components:
  - Data Integration Server: Performs the actual data integration tasks
    - Executes jobs and transformations
    - Defines and manages security
    - Provides content management
    - Schedules and monitor activities
  - Spoon: A graphical user interface for designing jobs and transformations
    - Transformations can be executed locally within Spoon, or in the Data Integration Server
  - Pan: A standalone command line tool for executing transformations
  - Kitchen: A standalone command line tool for executing jobs Jobs are usually scheduled to run in batch mode at regular intervals.
  - Carte: A lightweight server for running jobs and transformations on a remote host

Chapter 6: Extraction, Transformation, and Loading

Outline

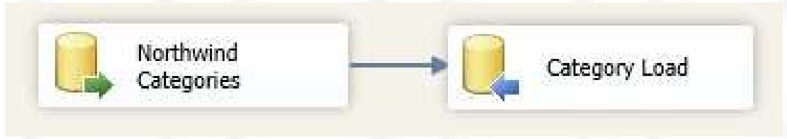
The Northwind ETL in Integration Services

- We just need to translate the conceptual constructs to the equivalent Integration Services ones
- Overall view of the ETL process

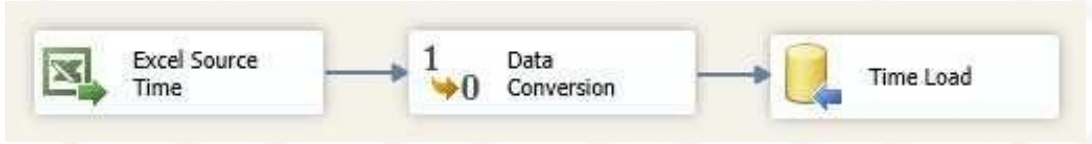


Data Flow Tasks

- Many data flow tasks are simple
- These data flow tasks are composed of an OLE DB Source task that reads the table from the operational database and an OLE DB Destination task that receives the output and stores it in the DW
- Loading the Category dimension table

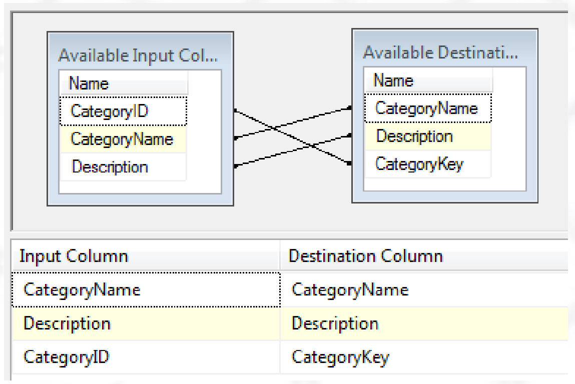


- Similar data flows are used for loading the Product, Shipper, and Employee tables
- Also straightforward is the data flow that loads the Time dimension from the source Excel file after a data conversion

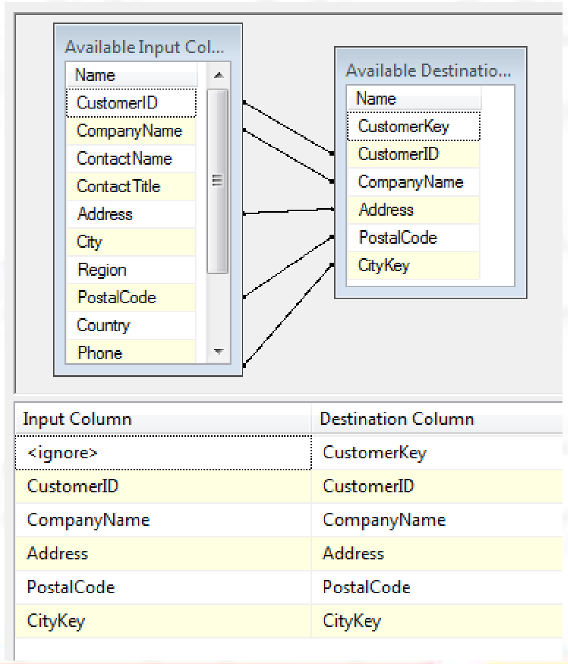


Keys in the Data Warehouse

- Keys of the operational database are reused in the DW where dimensions do not have an alternate key
- For example, for table Category we reuse CategoryID as the key in the DW (CategoryKey)
- For table Customer the CustomerID key is an CustomerAltKey column in the DW
- A new value for CustomerKey is generated during the insert in the DW
- Mappings of the source and destination columns depending on the reuse of the key



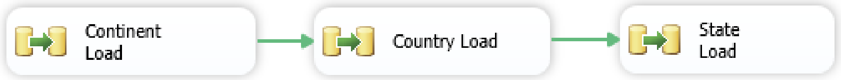
Input Column	Destination Column
CategoryName	CategoryName
Description	Description
CategoryID	CategoryKey



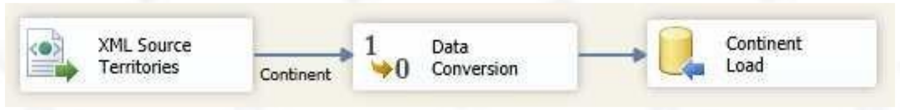
Input Column	Destination Column
<ignore>	CustomerKey
CustomerID	CustomerID
CompanyName	CompanyName
Address	Address
PostalCode	PostalCode
CityKey	CityKey

Load of the Continent → Country → State Hierarchy

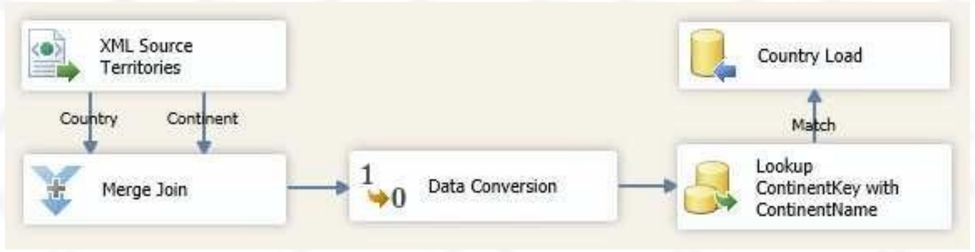
- Sequence container used for the three data flows that load the tables of the hierarchy



- Load of the Continent level

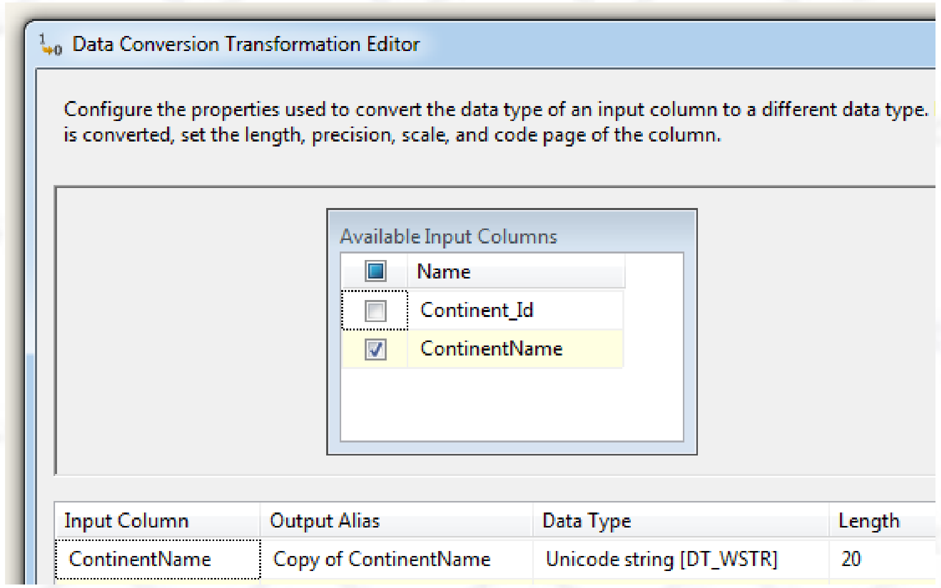


- Load of the Country level



- First produce a key to reference Continent from Country
- Data conversion tasks detailed in the next slide
- In the data flow that loads Country a merge join obtains the ContinentName for a given Country

Conversion of the Data Input from the XML File



Data Conversion Transformation Editor

Configure the properties used to convert the data type of an input column to a different data type.
is converted, set the length, precision, scale, and code page of the column.

Available Input Columns

	Name
☐	Continent_Id
☑	ContinentName

Input Column	Output Alias	Data Type	Length
ContinentName	Copy of ContinentName	Unicode string [DT_WSTR]	20

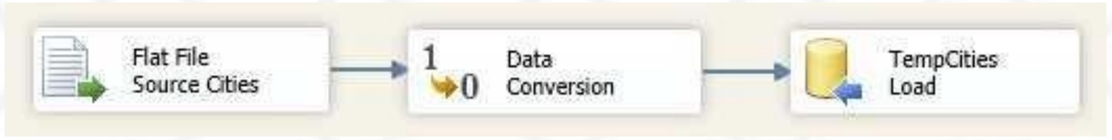
- A data conversion transforms the data types from the XML file into the data types of the database
- The ContinentName read from the XML file is by default of length 255, and it is converted into a string of length 20

Load of the TempCities Table

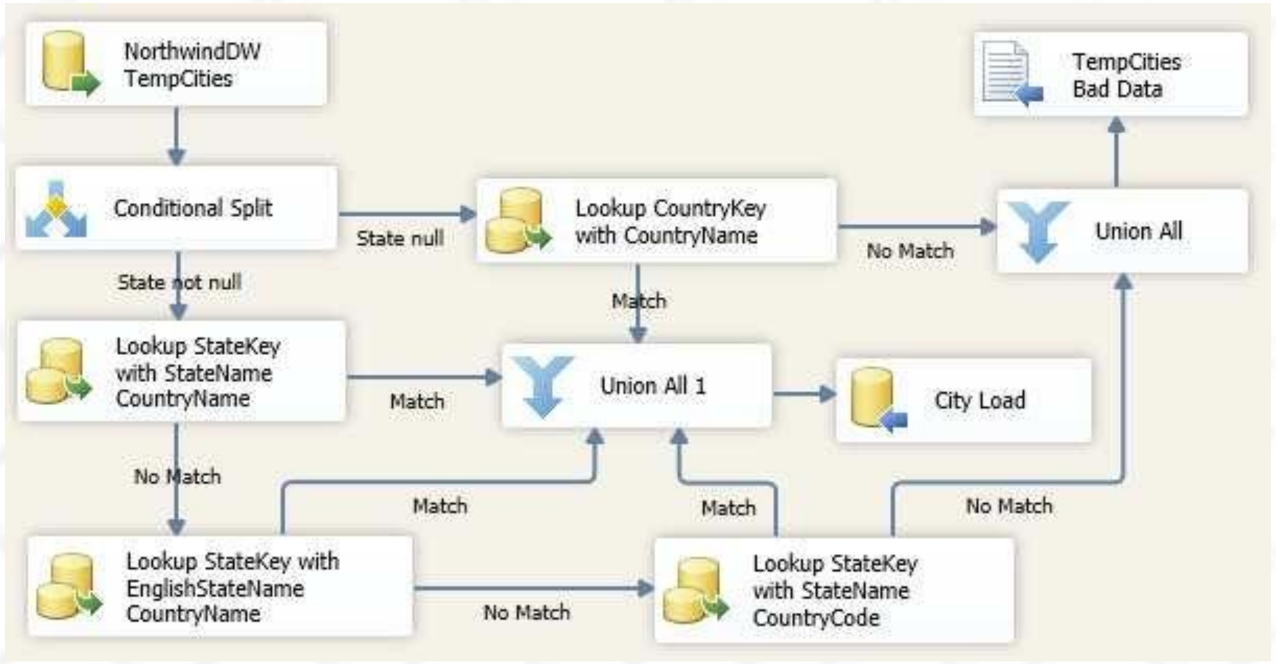
- TempCities: A temporary table needed to load the geographic hierarchy associated to dimensions Customer and Supplier
  - TempCities: Obtained from the text file Cities.txt
- Structure of the temporary table

TempCities
City
State
Country

- We assume that this table already exists in the database
- A data conversion transformation is needed to transform the default types obtained from the text file into the database types

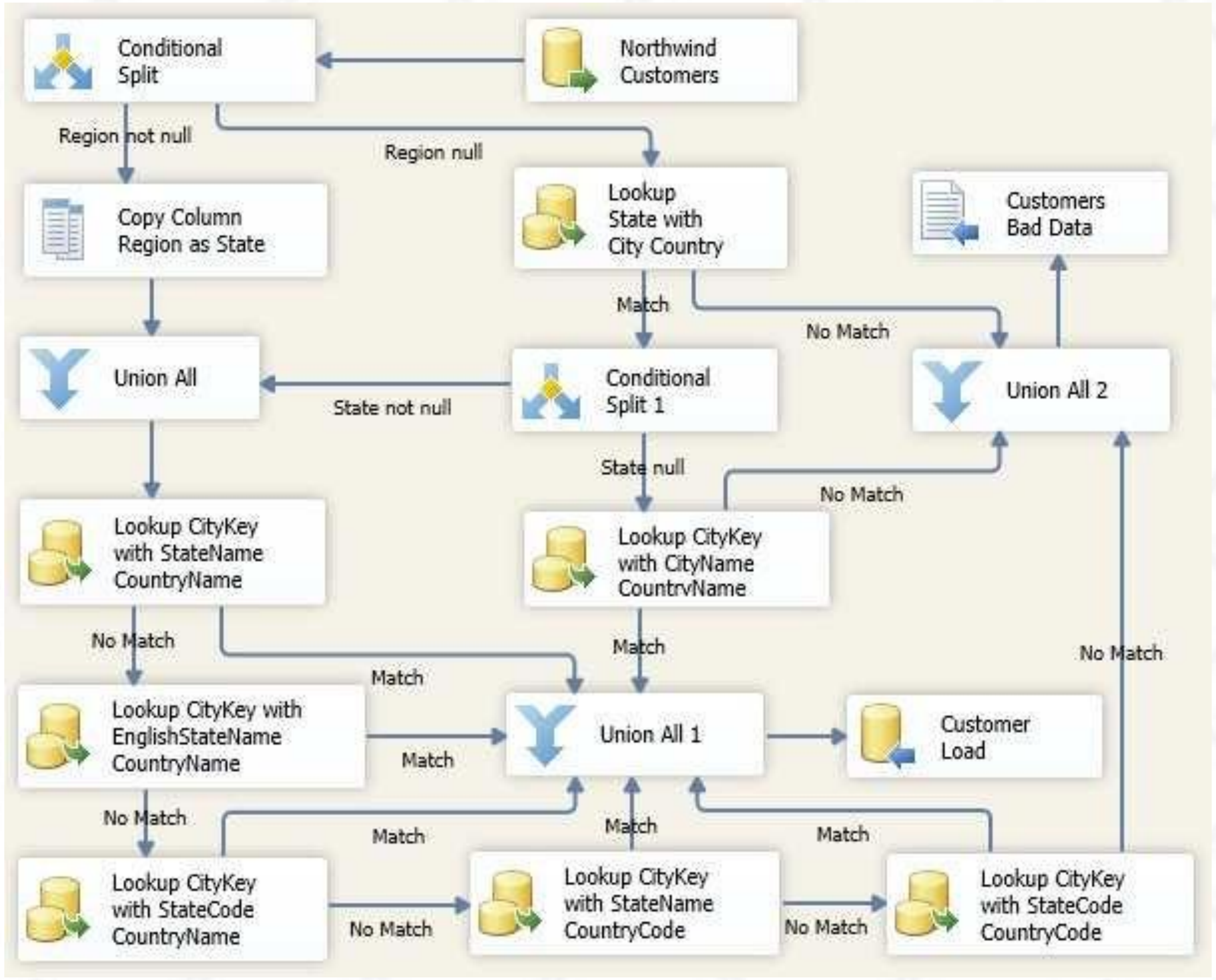


Load of the City Level


NorthwindDW TempCities
Conditional Split
State null
State not null
Lookup CountryKey with CountryName
No Match
Match
Union All
TempCities Bad Data
Lookup StateKey with StateName CountryName
Match
Union All 1
City Load
No Match
Match
Match
No Match
Lookup StateKey with EnglishStateName CountryName
No Match
Lookup StateKey with StateName CountryCode

- The data flow associates to each city in TempCities, either a StateKey or a CountryKey, depending on whether or not the corresponding country is divided in states. For this:
  - The conditional split tests if the State is null or not
  - If so, a lookup is needed for obtaining the CountryKey

Load of the Customer Level


Conditional Split
Northwind Customers
Region not null
Region null
Copy Column Region as State
Lookup State with City Country
Customers Bad Data
Match
No Match
Union All
State not null
Conditional Split 1
Union All 2
State null
No Match
Lookup CityKey with StateName CountryName
Lookup CityKey with CityName CountryName
No Match
Match
Match
No Match
Lookup CityKey with EnglishStateName CountryName
Match
Union All 1
Customer Load
No Match
Match
Match
Match
Lookup CityKey with StateCode CountryName
No Match
Lookup CityKey with StateName CountryCode
No Match
Lookup CityKey with StateCode CountryCode

Load of the Customer Level

- Starts with a conditional split: if a customer has a null value in Region, a lookup adds a column State

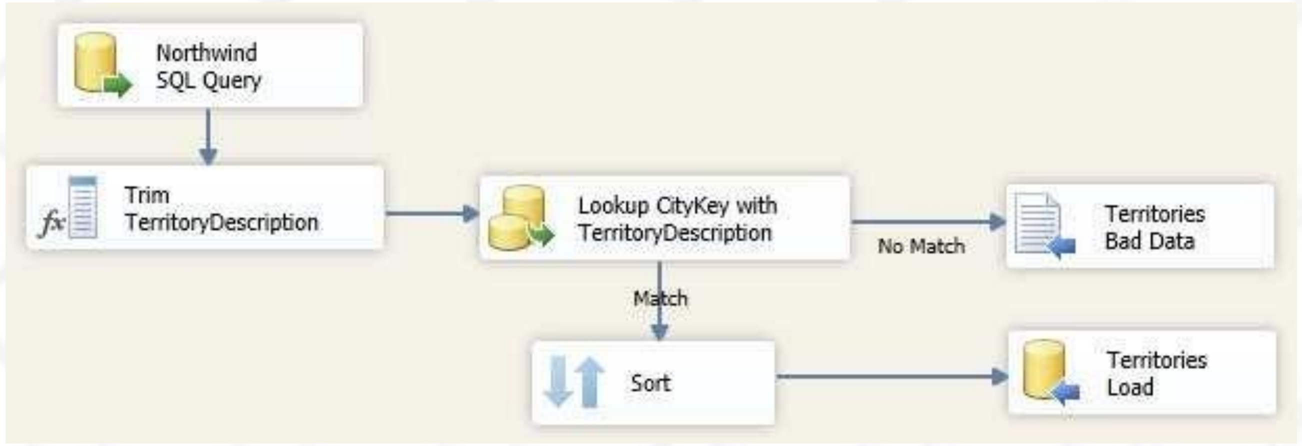
  by matching City and Country from Customers with City and Country from TempCities
- The value State obtained may be null for countries without states ⇒ a conditional split is needed
- If state is null, then a lookup tries to find a CityKey matching values of City and Country in a lookup table built as a join between City and Country

```
SELECT CityKey, CityName, CountryName
FROM City C JOIN Country T ON  C.CountryKey = T.CountryKey
```

- For customers with nonnull Region, the values of this column are copied into a new column State
- Then, 5 lookup tasks are needed, where each one tries to match a couple of values of State and Country to values in the lookup table built as a join between the City, State, and Country tables:

```
SELECT C.CityKey, C.CityName, S.StateName,
       S.EnglishStateName,S.StateCode, T.CountryName,
       T.CountryCode
FROM City C JOIN State S ON C.StateKey = S.StateKey  JOIN Country
       T ON S.CountryKey = T.CountryKey
```

Load of the Territories Fact Table

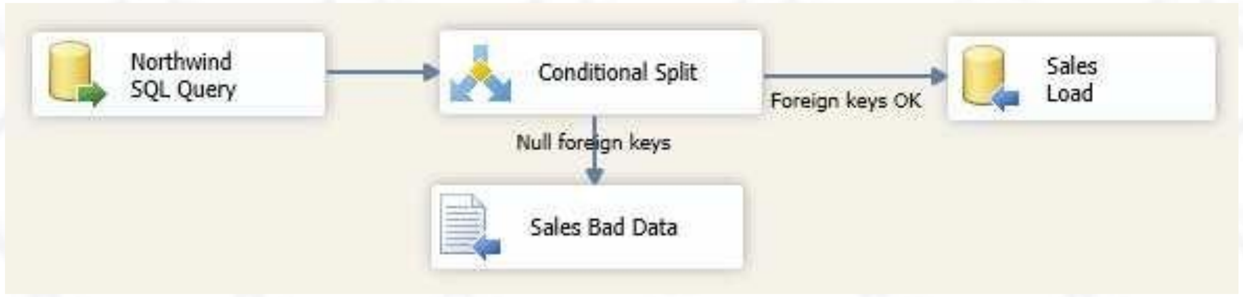


- The data flow task starts with an OLE DB Source task, an SQL query:

  SELECT E.*, TerritoryDescription

  FROM EmployeeTerritories E JOIN Territories T ON E.TerritoryID = T.TerritoryID
- Continues with a derived column transformation that removes the trailing spaces in the values of TerritoryDescription
- A lookup transformation searches the corresponding values of CityKey in City
- Then a sort transformation removes duplicates

Load of the Sales Fact Table



- The first OLE DB Source task is a query that combines data from the operational DB and the DW (see the query in next slide)
- Then, a conditional split transformation task selects the records obtained in the query containing a null value in the columns CustomerKey or ShippedDateKey, and stores them in a flat file.
- The correct records are inserted in the data warehouse.

Load of the Sales Fact Table

- Query of the OLE DB source task

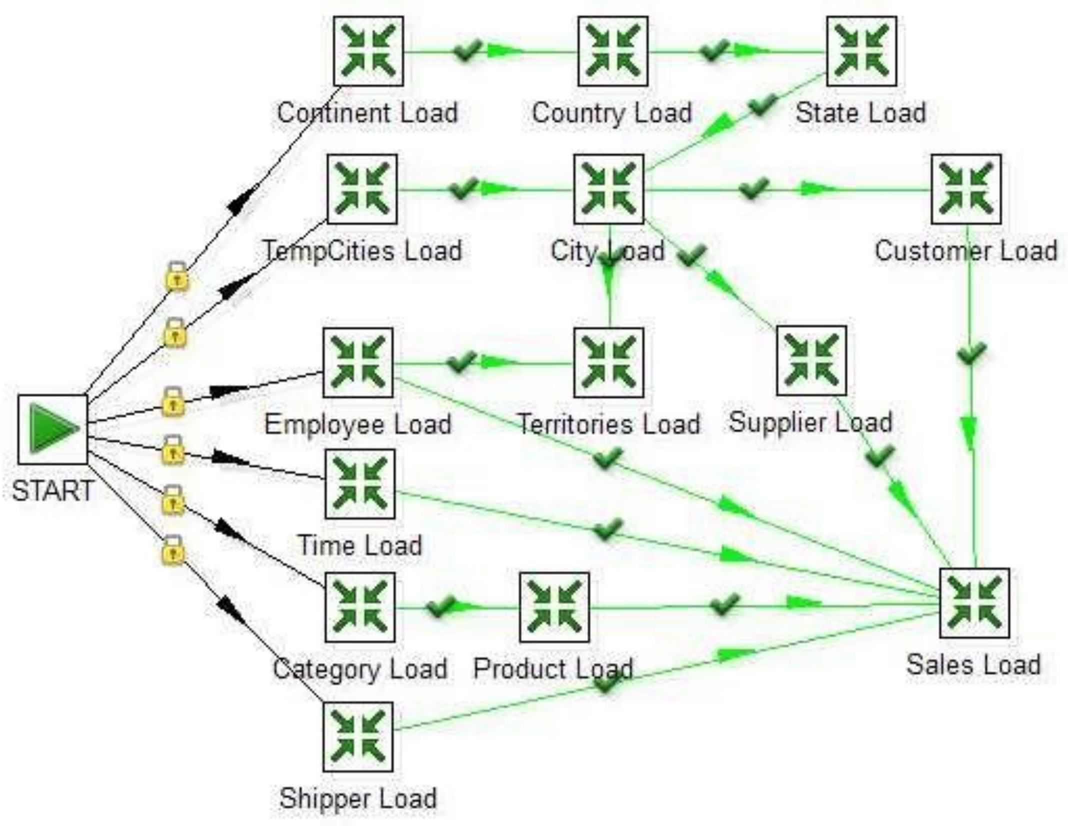
```
SELECT
    ( SELECT CustomerKey FROM dbo.Customer C
      WHERE C.CustomerID = O.CustomerID) AS CustomerKey,  EmployeeID AS
    EmployeeKey,
    ( SELECT TimeKey FROM dbo.Time T
      WHERE T.Date = O.OrderDate) AS OrderDateKey,
      ( SELECT TimeKey FROM dbo.Time T
      WHERE T.Date = O.RequiredDate) AS DueDateKey,
      ( SELECT TimeKey FROM dbo.Time T
      WHERE T.Date = O.ShippedDate) AS ShippedDateKey,  ShipVia AS
    ShipperKey, P.ProductID AS ProductKey,  SupplierID AS SupplierKey,
    O.OrderID AS OrderNo,
    CONVERT(INT, ROW NUMBER() OVER (PARTITION BY D.OrderID
      ORDER BY D.ProductID)) AS OrderLineNo, D.UnitPrice, Quantity, Discount,  CONVERT(MONEY,
    D.UnitPrice * (1-Discount) * Quantity) AS SalesAmount,  CONVERT(MONEY, O.Freight/COUNT(*)
    OVER (PARTITION BY D.OrderID)) AS Freight
FROM Northwind.dbo.Orders O, Northwind.dbo.OrderDetails D,  Northwind.dbo.Products P
WHERE  O.OrderID = D.OrderID AND D.ProductID = P.ProductID
```

Chapter 6: Extraction, Transformation, and Loading

Outline

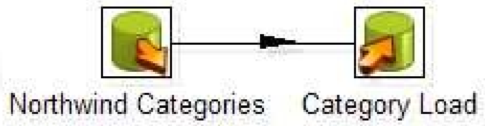
- Business Process Modeling Notation
- Conceptual ETL Design using BPMN
- Conceptual Design of the Northwind ETL
- Integration Services and Kettle
- The Northwind ETL process in Integration Services
- The Northwind ETL process in Kettle

Overall View of the ETL Process

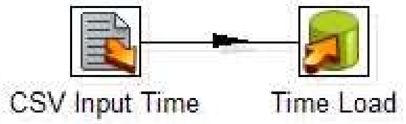


Load of the Category and Time Dimension Tables

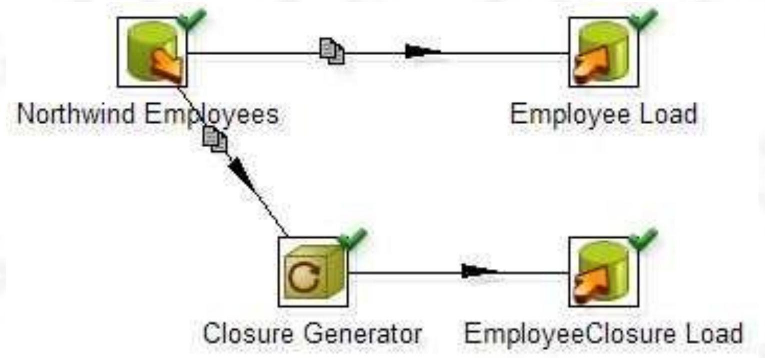
- Loading the Category dimension table is similar to the data flow in Integration Services



- Loading the Time dimension table: Specified in the transformation step that reads the CSV file



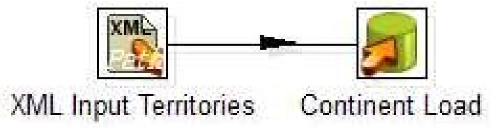
Load of the Employee Dimension Table



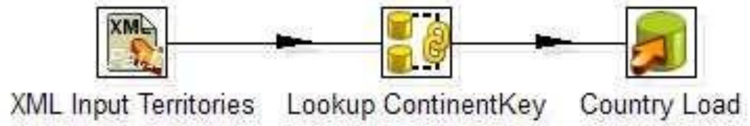
- Transformation requires a closure table containing the transitive closure of Supervision hierarchy
- After reading the Employees table the rows read are sent in parallel to the steps that load the Employee
  and the EmployeeClosure tables

Load of the Continent and Country Levels

- Load of the Continent level

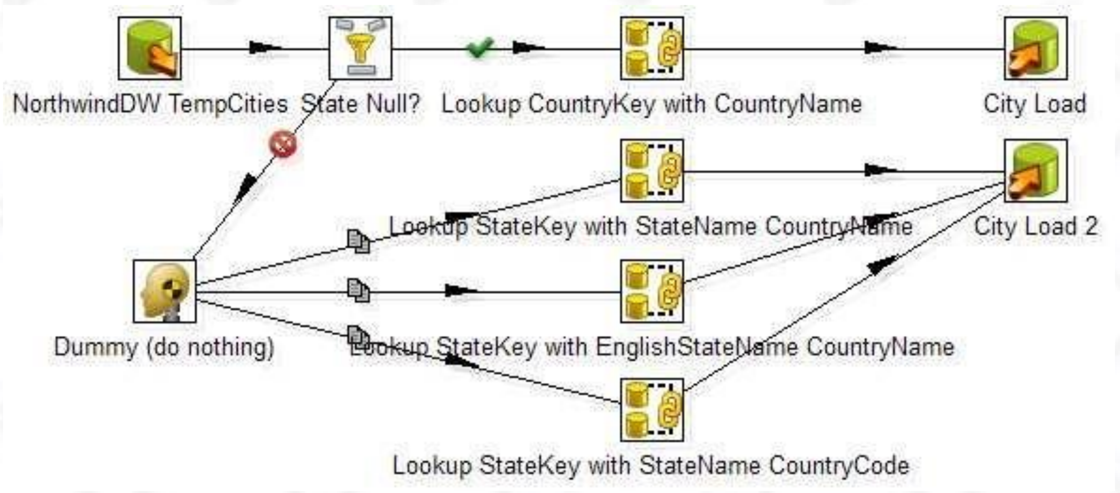


- With respect to IS, the conversion task is not required in Kettle
- Load of the Country level



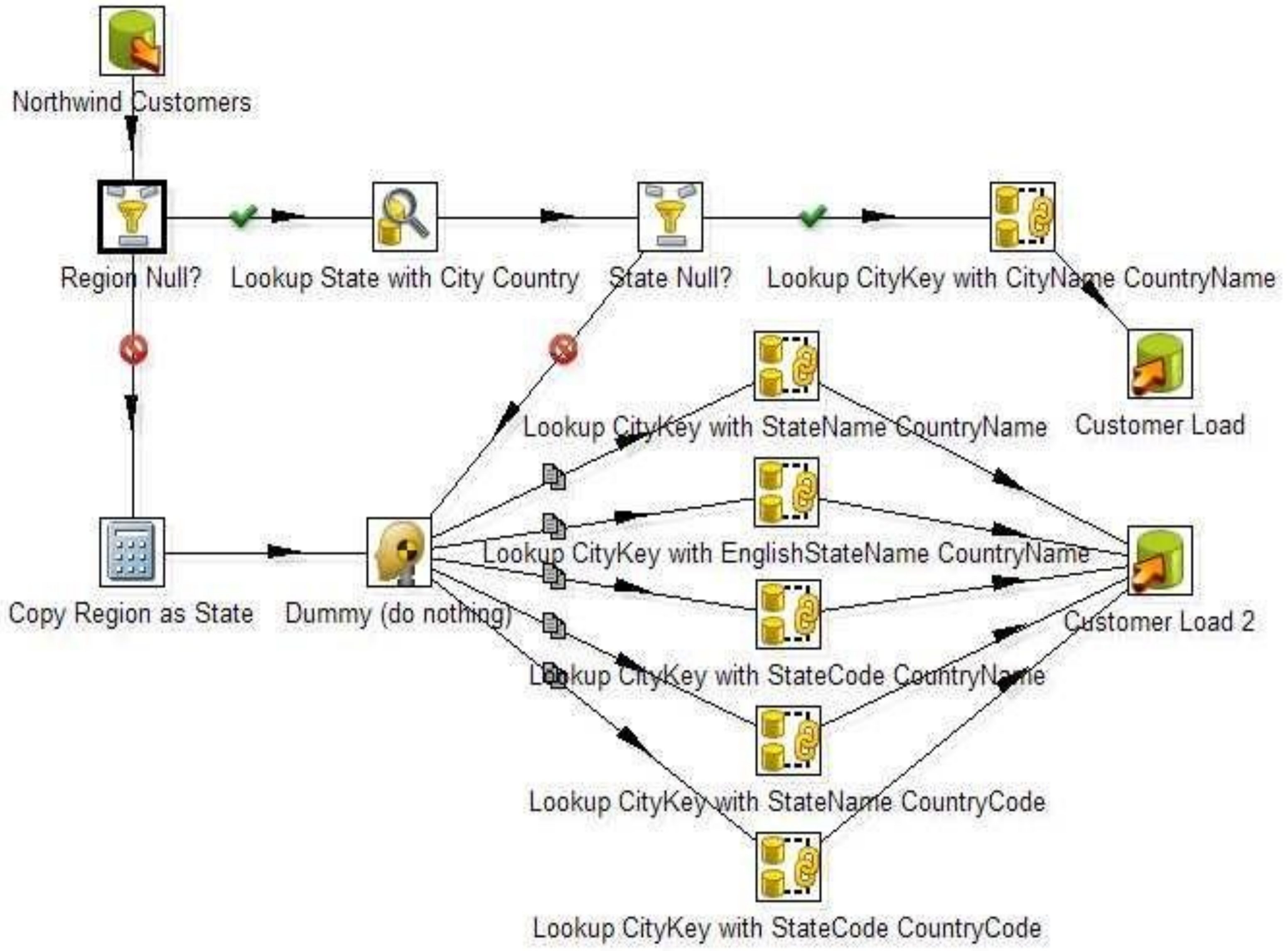
- With respect to IS, in Kettle we can find the ContinentName associated to a Country using an XPath expression, as in the conceptual design

Load of the City Level


NorthwindDW TempCities
State Null?
Lookup CountryKey with CountryName
City Load
Lookup StateKey with StateName CountryName
City Load 2
Dummy (do nothing)
Lookup StateKey with EnglishStateName CountryName
Lookup StateKey with StateName CountryCode

- Significant differences with IS
- No possible to cascade lookup steps in Kettle as it is done with lookup tasks in IS
- Cascade lookups must be implemented as a collection of parallel flows.
- In Kettle we do not have tasks that load records for which a lookup value was not found in a text file
- The rows that do not have a null value sent in parallel to all the subsequent lookup tasks
- A dummy task is needed
- In Kettle there is no need to explicitly include a union task but all fields in the input flows have the same name → need one step for CountryKey, and other for StateKey

Load of the Customer Level

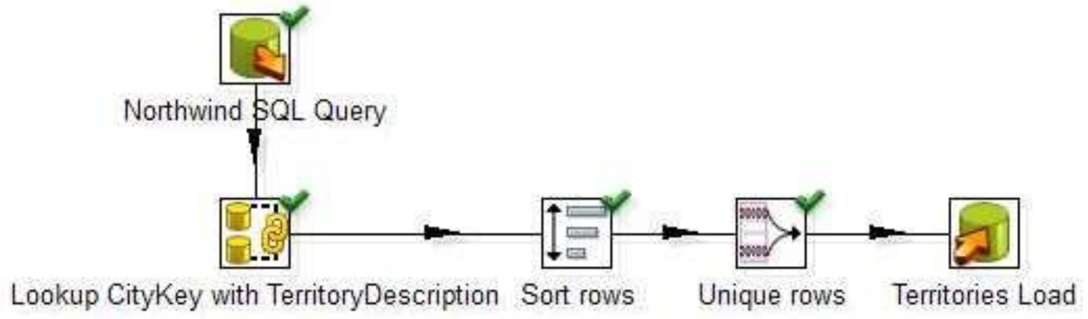

Northwind Customers
Region Null?
Lookup State with City Country
State Null?
Lookup CityKey with CityName CountryName
Customer Load
Lookup CityKey with StateName CountryName
Lookup CityKey with EnglishStateName CountryName
Copy Region as State
Dummy (do nothing)
Customer Load 2
Lookup CityKey with StateCode CountryName
Lookup CityKey with StateName CountryCode
Lookup CityKey with StateCode CountryCode

Load of the Customer Level

- Two different steps for performing lookups (different icons):
  - One that looks for State, and the other ones that look for CityKey
  - The former lookup type looks for values *in a single table* and sends all rows to the output flow
  - The second type of lookup looks for values *in an SQL query* and only sends to the output stream the rows with matching value
- A dummy task is needed in Kettle
- Dummy step sends the input rows to all subsequent lookup tasks
- SQL query used in the lookup step that looks for CityKey with StateName and CountryName

```
SELECT C.CityKey
FROM City C JOIN State S ON C.StateKey = S.Statekey  JOIN Country T
ON S.CountryKey = T.CountryKey
WHERE  ? = CityName AND ? = StateName AND ? = CountryName
```

Load of the Territories Fact Table



- Flow starts by obtaining the assignment of employees to territories from the Northwind database using an SQL query
- In Kettle there is no step that removes the trailing spaces in the TerritoryDescription column
- This was taken into account in the SQL query of the subsequent lookup step:

```
SELECT CityKey  FROM City
WHERE  TRIM(?) = CityName
```

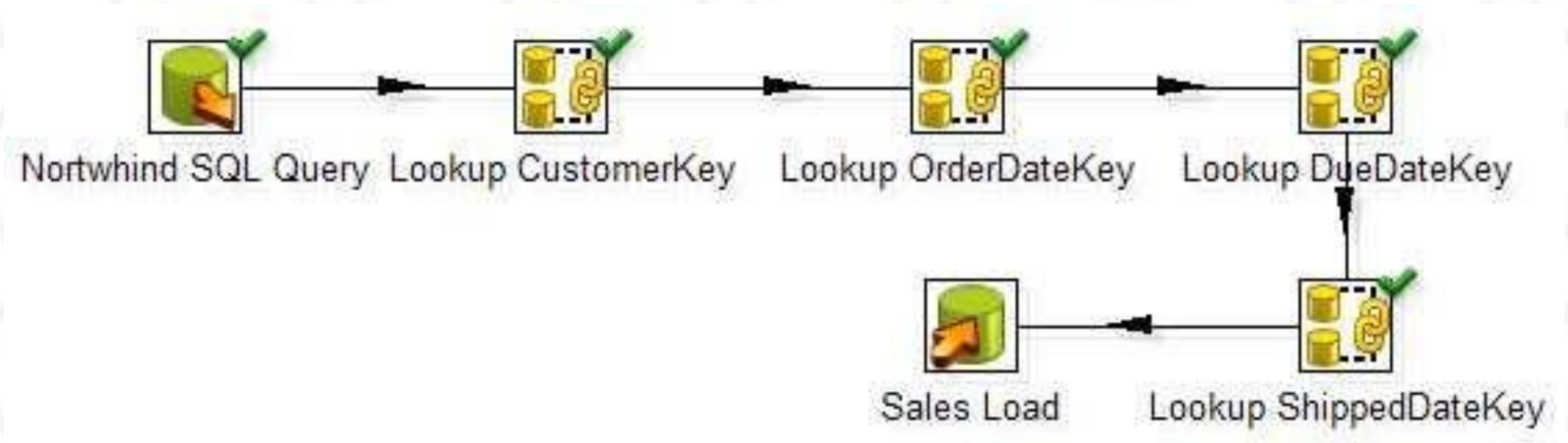
- After the lookup of the CityKey, Kettle requires a sort but this does not remove duplicates, like in IS

Load of the Territories Fact Table

◆The flow starts by obtaining values from the following SQL query addressed to the Northwind database, like in the conceptual design

```
SELECT O.CustomerID, EmployeeID AS EmployeeKey,
        O.OrderDate, O.RequiredDate, O.ShippedDate,  ShipVia
        AS ShipperKey, P.ProductID AS ProductKey,
        P.SupplierID AS SupplierKey, O.OrderID AS OrderNo,
        ROW NUMBER() OVER (PARTITION BY D.OrderID
        ORDER BY D.ProductID) AS OrderLineNo,
        D.UnitPrice, Quantity, Discount,
        D.UnitPrice * (1-Discount) * Quantity AS SalesAmount,
        O.Freight/COUNT(*) OVER (PARTITION BY D.OrderID) AS
        Freight
FROM      Orders O, OrderDetails D, Products P
WHERE  O.OrderID = D.OrderID AND D.ProductID = P.ProductID
```

Load of the Sales Fact Table



- ◆In IS it is possible to query both the Northwind operational database and the Northwind data ware- house in a single query
- ◆Not possible in PostgreSQL
- ◆Thus, additional lookup steps are needed in Kettle for obtaining the surrogate keys
- ◆Additional task needed in IS for removing the records with null values for surrogate keys
- ◆These are automatically removed in the lookup steps in Kettle

Enjoy the Course…!