

# Business intelligence

## Tutorial 4 – Pentaho Data Integration - Extension



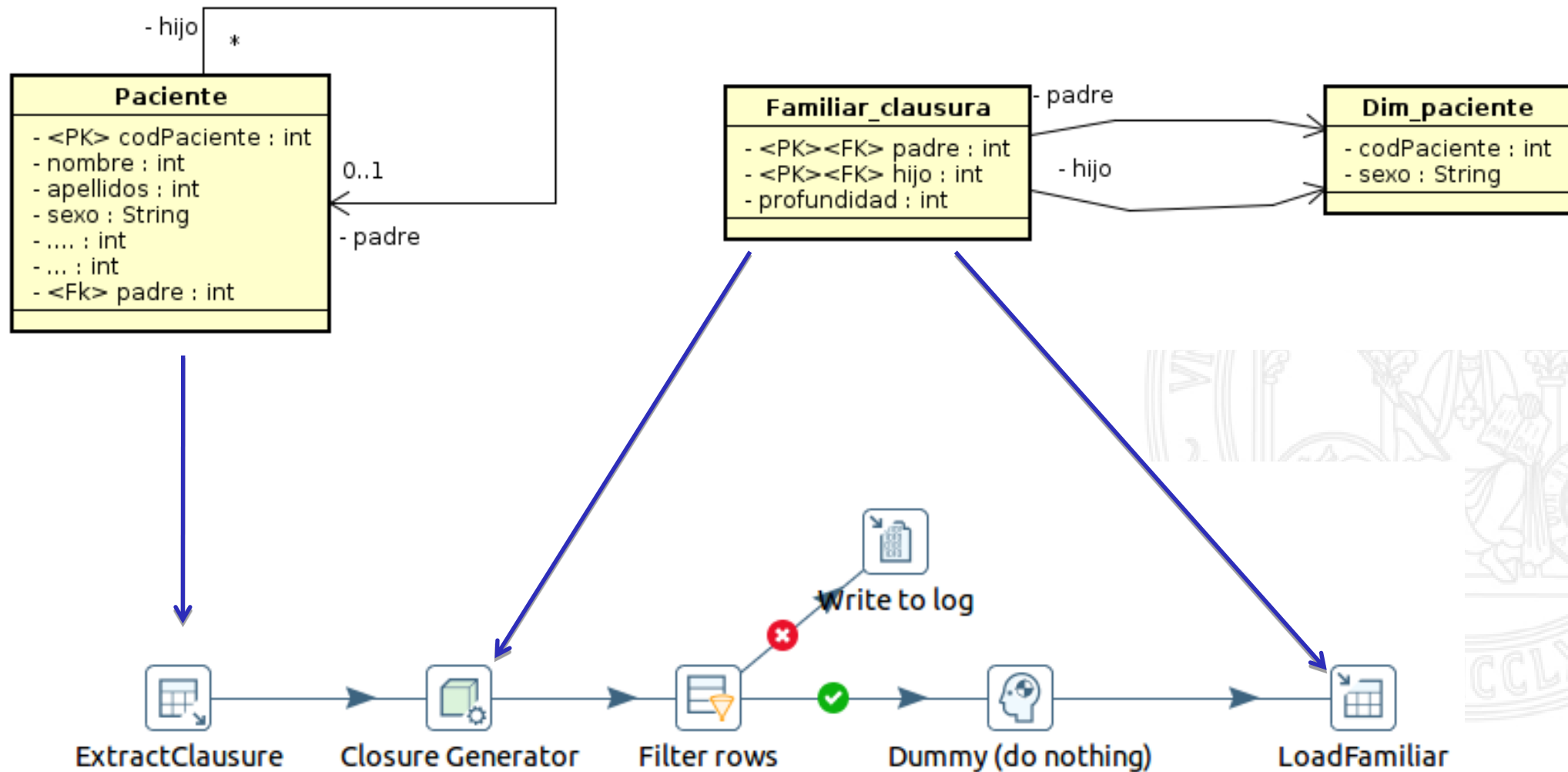
UNIVERSIDAD DE  
MURCIA

- 1 – Load closure
- 2 – Load dimension
  - Read XML with XPath
  - Set and read global variables
  - Update dimension
- 3 – Load prescription
  - Extract from CSV file
  - Stream lookup
  - Javascript



- We will extend the tut3 database with new tables
- Uncompress IN-2017-T4-extra\_subir.rar
- Load SQL scripts
  - sql2-alter-schema-pacientes.sql
  - sql3-alter-schema-alumno.sql
    - **IMPORTANT: change the “dw\_prof” by your own schema.**
  - sql-1-borrar-datos-esquema.sql: **USE IT WHEN NEEDED.**
- We use the same database connection as in tutorial 3

- Create a new PDI transformation



# 1- Load closure

- Input-> Table input
- Schema: "PACIENTES" clausura
- Fields: codpaciente, padre

Table input

Step name: ExtractClausure

Connection: tut4

SQL:

```
SELECT
  codpaciente
, padre
FROM pacientes.paciente
```

Line 3 Column 2

Enable lazy conversio ☐

Replace variables in s ☐

Insert data from step

Execute for each row: ☐

Limit size: 0

Buttons: Help, OK, Preview, Cancel

- Closure:
  - Transform -> Closure generator
  - Set "padre", "hijo", "profundidad"

Closure Generator Dialog

Step name: Closure Generator

Parent ID field: padre

Child ID field: codpaciente

Distance field name: profundidad

Root is zero (Integer)? ☐

Buttons: Help, OK, Cancel

# 1- Load closure

- Filter:
  - Flow -> Filter
  - True: Dummy step
  - False: Write to log
  - Condition
- Flow -> Dummy
- Utility -> Write to log
  - Set fields and log level

**Filter rows**

Step name: Filter rows

Send 'true' data to step: Dummy (do nothing)

Send 'false' data to step: Write to log

The condition:

☐ To edit a subcondition,

codpaciente IS NOT NULL

AND

padre IS NOT NULL

Buttons: Help, OK, Cancel

**Write to log**

Step name: Write to log

Log level: Basic

Print header: ☒

Limit rows?: ☐

Nr of rows to print: 0

Write to log:

Fields:

	Field
1	codpaciente
2	padre
3	profundidad

Buttons: Help, OK, Get Fields, Cancel

- Store dimension:
  - Output -> Table
  - Target: “familiar\_clausura”
  - Map fields
    - Padre -> padre
    - Hijo -> codpaciente
    - Profundidad -> profundidad

**Table output**

Step name: LoadFamiliar

Connection: tut4

Target schema: dw\_in00

Target table: familiar\_clausura

Commit size: 1000

Truncate table: ☐

Ignore insert errors: ☐

Specify database field: ☒

**Database fields**

Fields to insert:

	Table field	Stream field
1	padre	padre
2	hijo	codpaciente
3	profundidad	profundidad

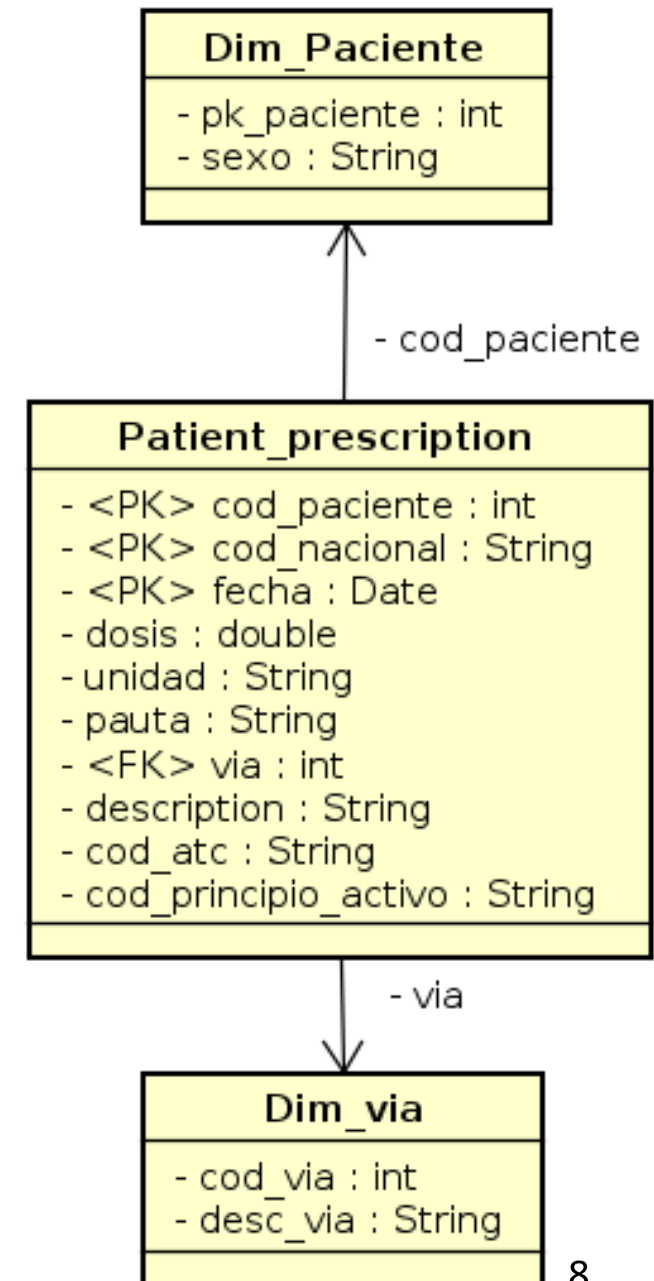
Buttons: Get fields, Enter field mapping, Help, OK, Cancel, SQL

**Edit Data - IN (155.54.205.172)**

	padre integer	hijo integer	profundidad integer
1	16	1	4
2	32	1	5
3	64	1	6
4	128	1	7
5	256	1	8
6	512	1	9
7	1	1	0
8	2	1	1
9	4	1	2
10	8	1	3
11	16	2	3
12	32	2	4

## 2- Load dimension

- Steps
  - Configure global variables
  - Load XML VIAS – Xml
    - Read path from global variable
    - Read XML
    - Update dimension





## 2 – Use global variables from config files

- Files:

- `$HOME/.kettle/shared.xml`
- `$HOME/.kettle/kettle.properties`
  - Set `pathPrescription=MIRUTA1`
    - Unzip there the file given
  - Set `rutaErrors=MIRUTA2`

- Note1: without blank around “=”.

- Note2: set simple path (C:\folder).

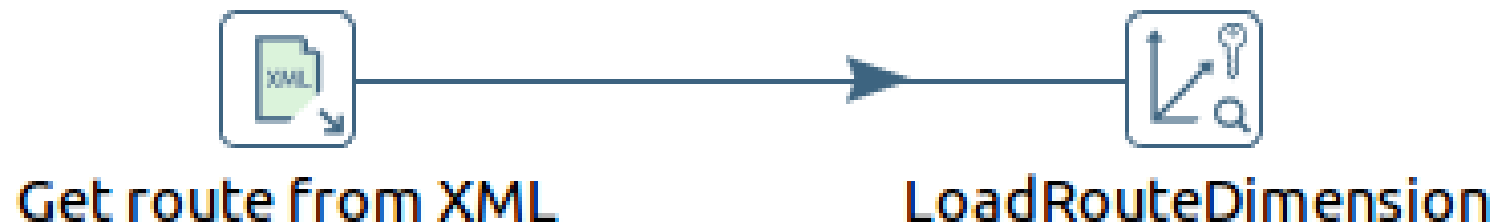
```
Terminal
Terminal
mcampos@periquito:~/.kettle$ ls
db.cache          db.cache-6.0.0.0-353  shared.xml.backup
db.cache-5.2.0.0   kettle.properties     xulSettings.properties
db.cache-5.3.0.0-213 shared.xml
mcampos@periquito:~/.kettle$
```

```
mcampos@periquito:~/.kettle$ cat shared.xml
<?xml version="1.0" encoding="UTF-8"?>
<sharedobjects>
  <connection>
    <name>IN_LOCAL</name>
    <server>localhost</server>
    <type>POSTGRESQL</type>
    <access>Native</access>
    <database>jjj</database>
    <port>5433</port>
    <username>inbd</username>
    <password>Encrypted 2be98afc86aa7f2e4cb79ce10d79cadde<
    <servername/>
    <data_tablespace/>
    <index_tablespace/>
    <attributes>
      <attribute><code>FORCE_IDENTIFIERS_TO_LOWERCASE</code>
      <attribute><code>FORCE_IDENTIFIERS_TO_UPPERCASE</code>
      <attribute><code>IS_CLUSTERED</code><attribute>N</att
    </attributes>
  </connection>
  <PORT_NUMBER</code><attribute>5433</
  <PRESERVE_RESERVED_WORD_CASE</code><
  <QUOTE_ALL_FIELDS</code><attribute>N
  <SUPPORTS_BOOLEAN_DATA_TYPE</code><a
  <SUPPORTS_TIMESTAMP_DATA_TYPE</code>
  <USE_POOLING</code><attribute>N</att
  </sharedobjects>
</sharedobjects>
```

```
kettle.properties + (~/.kettle) - VIM
Terminal
Terminal
kettle.properties + (~/.kettle...
1 pathPrescription=/home/mcampos/tutorials/t4-paths/prescripcion_201407
2 pathErrors=/home/mcampos/tutorials/t4-paths/errors
-- INSERTAR --
1,36 Todo
```

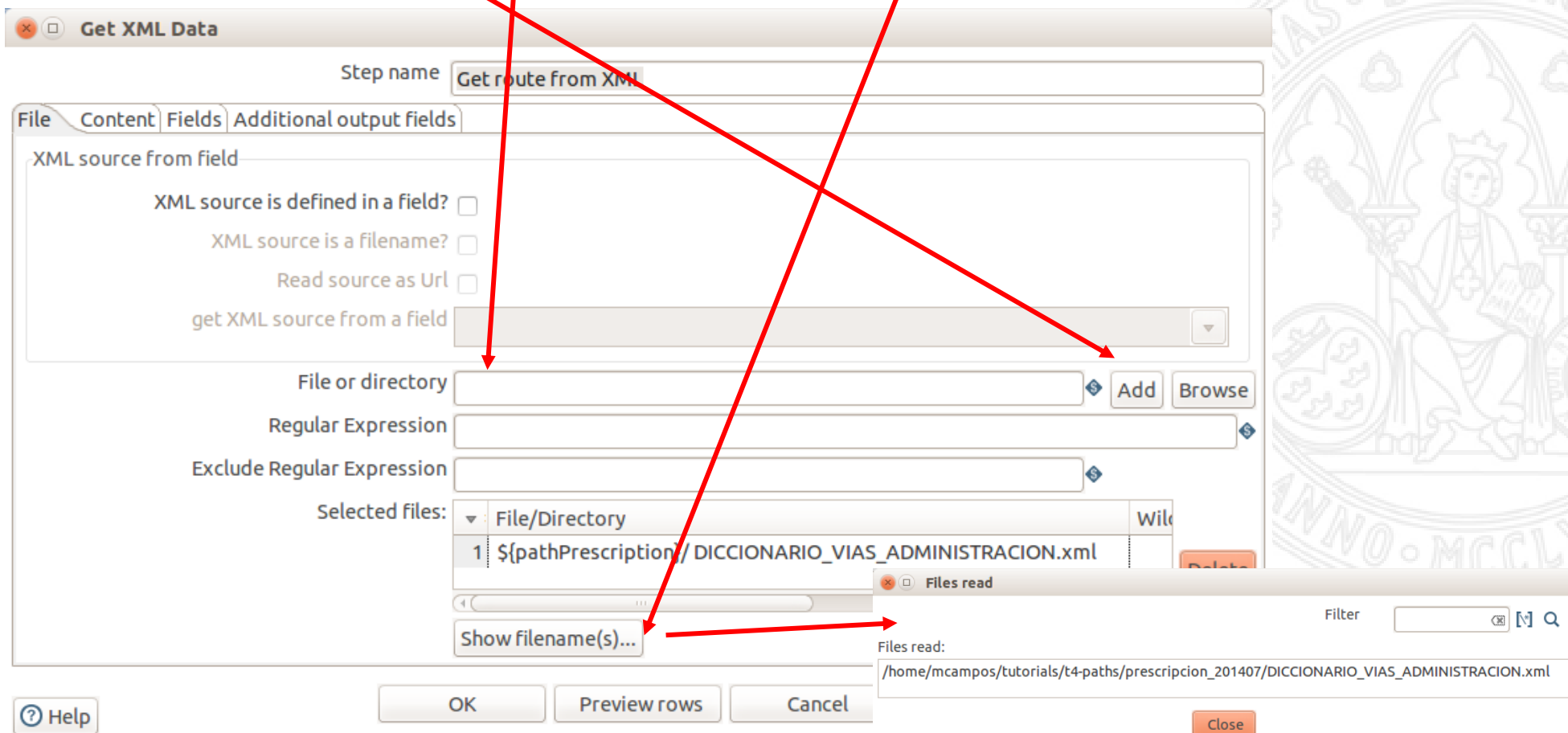
- Transformation

- Input -> Get data from XML
  - Read content with XPath expressions
- Datawarehouse -> Combination lookup/update
  - For SCD use Dimension lookup/update



## 2- Read XML file

- Read file using global variable previously defined
  - `${pathPrescription}/DICCIONARIO_VIAS_ADMINISTRACION.xml`
  - **NOTE: Restart PDI to read global variables**
- Click on “Add”. Check with “Show filenames” that file is ok.

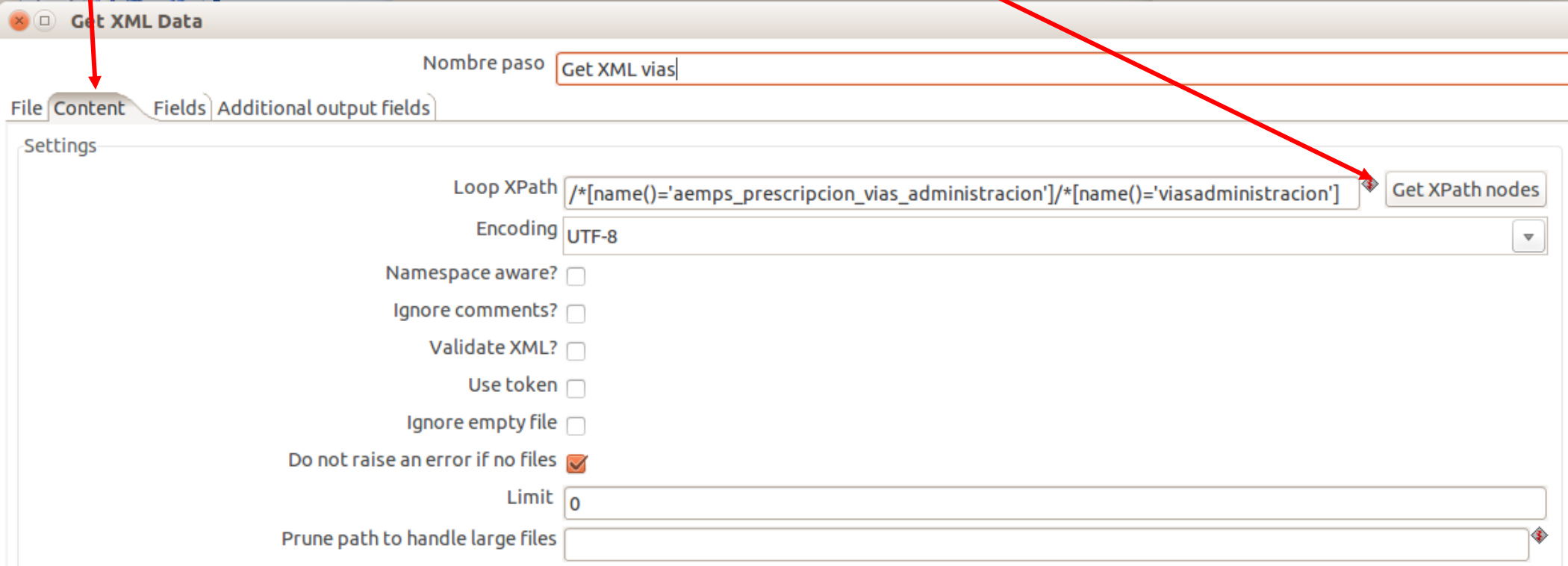
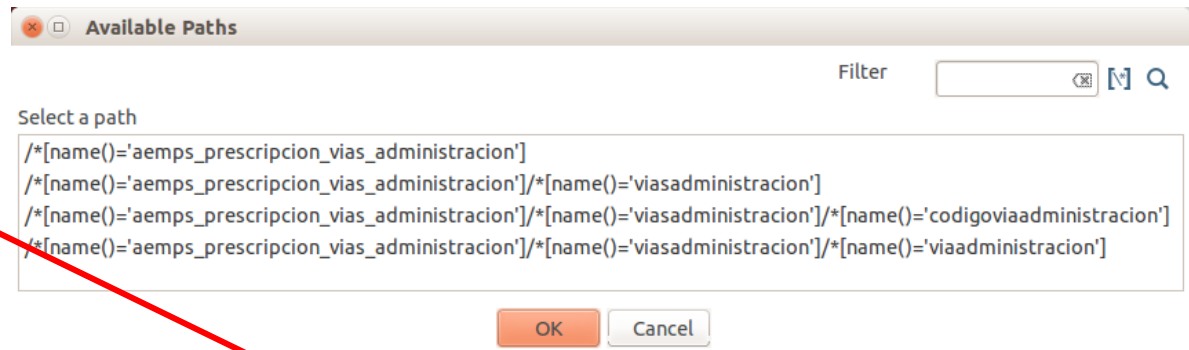


## 2- Read XML file

- Content tab

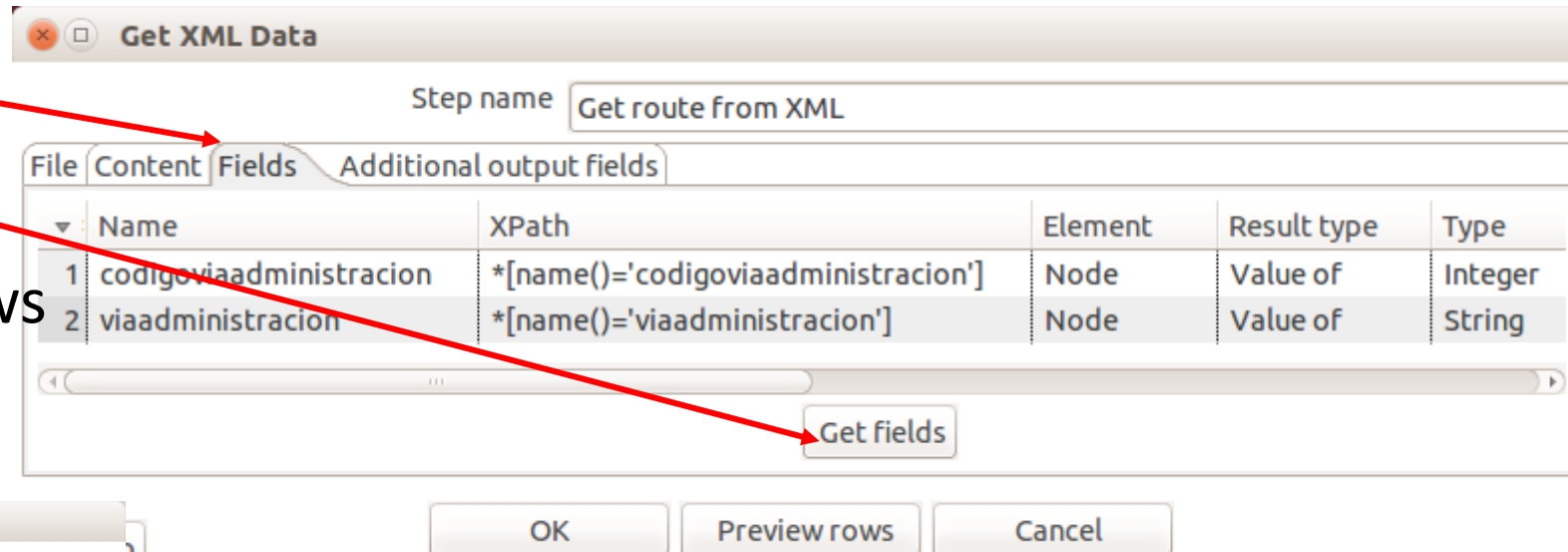
- Click “Get XPath Nodes”

- Select Xpath to “viasadministracion”-> We read its leaves



## 2- Read XML file

- Fields tab
  - Read fields
  - Preview rows



## 2- Update dimension

- Set target table: dw\_YOURSCHEMA.dw\_dim\_via
- Set key fields
- Set table column for subrogate key: cod\_via
- Create subrogate key
  - As max table+1
- It will insert a new row when necessary

Combination Lookup / Update

Step name: LoadRouteDimension

Connection: tut4 [Edit...] [New...] [Wizard...]

Target schema: dw\_in00 [Browse...]

Target table: dw\_dim\_via [Browse...]

Commit size: 100 Cache size: 9999

Pre-load the cach ☐

Key fields (to look up row in table):

	Dimension field	Field in stream
1	desc_via	viaadministracion

Technical key field: cod\_via

Creation of technical key

☒ Use table maximum + 1

☐ Use sequence [ ]

☐ Use auto increment field

Remove lookup fields? ☐

Use hashcode? ☐

Hashcode field in table

Date of last update field (of [ ]

[?] Help OK Cancel Get Fields SQL

## 2 – Load Dimension

- When running Check variables
- Now rerunning doesn't try to overwrite values of dimensions

**Run Options**

Environment Type

☒ Local The transformation will run on the machine you are using.

☐ Server

☐ Clustered

Options

☒ Clear log before running Log level: Basic

☐ Enable safe mode

☒ Gather performance metrics

Parameters Variables

Variable	Value
Internal.Entry.Current.Directory	file:///home/mcampos/Escritorio/Pentaho_6/t4-test-
Internal.Job.Filename.Directory	Parent Job File Directory
Internal.Job.Filename.Name	Parent Job Filename
Internal.Job.Name	Parent Job Name
Internal.Job.Repository.Directory	Parent Job Repository Directory
pathPrescription	/home/mcampos/tutorials/t4-paths/prescripcion_201

☒ Always show dialog on run

Help Run Cancel

### Execution Results

Execution History Logging Step Metrics Performance Graph Metrics

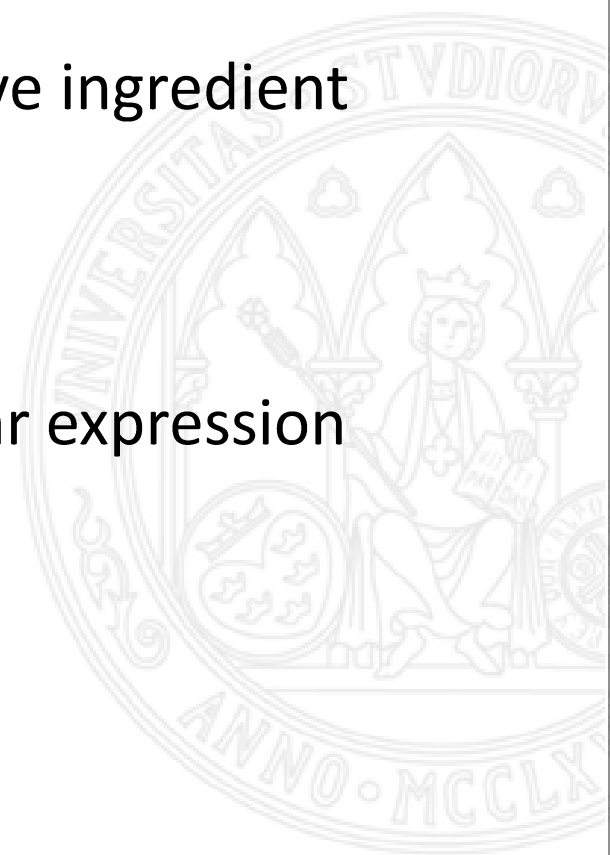
Stepname	Copypnr	Read	Written	Input	Output
Get route from XML	0	0	50	50	0
LoadRouteDimension	0	50	50	50	50

### Execution Results

Execution History Logging Step Metrics Performance Graph Metrics

Stepname	Copypnr	Read	Written	Input	Output
Get route from XML	0	0	50	50	0
LoadRouteDimension	0	50	50	50	0

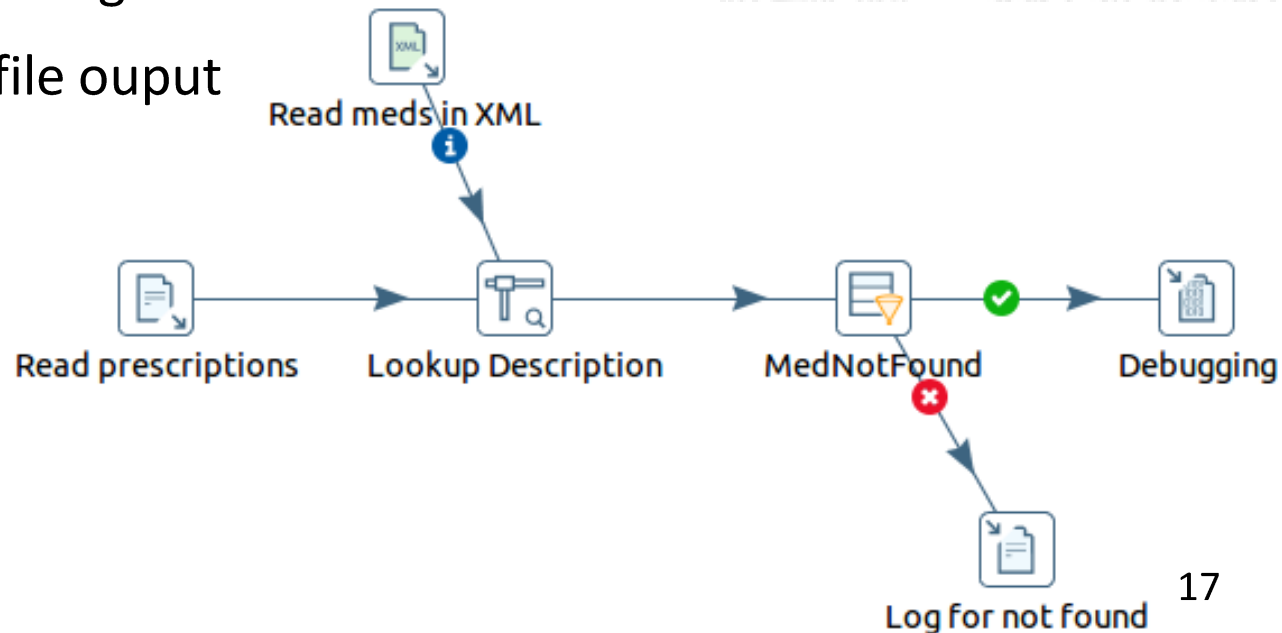
- Part 1
  - Read csv with prescriptions for patients. With fields
    - Active ingredient, dosis, route, etc.
  - Read XML of meds to get description of active ingredient
    - Stream lookup
- Part 2
  - Validate hour format with javascript + regular expression
    - Set constant value if invalid
  - Lookup route in dimension
  - Merge date and time in a “timestamp” field
    - Include metadata
  - Store in database





### 3 - Load prescriptions

- Patient prescriptions: Input -> CSV file input
- Meds description: Input -> Get data from XML
- Lookup description of active ingredient: Lookup -> Stream lookup
- Filter prescription without description (not found in stream):
  - Flow -> Filter rows
  - If true: Utility -> Write to log
  - If false: Output -> Text file output



# 3- Read CSV file

- Set filename using global var
  - `${pathPrescription}/patient_prescription.csv`
- Set delimiter “;”
- Enable lazy conversion
- Check header present
- Click “Get fields”
- **Important: data format**
- Include new column with row number for debugging: “row\_number”

The screenshot shows the 'CSV Input' dialog box with the following settings:

- Step name: Read prescriptions
- Filename: `${pathPrescription}/patient_prescription.csv` (with a 'Browse...' button)
- Delimiter: ; (with an 'Insert TAB' button)
- Enclosure: "
- NIO buffer size: 50000
- Lazy conversion? ☒
- Header row present? ☒
- Add filename to result: ☐
- The row number field name (opt): row\_number
- Running in parallel? ☐
- New line possible in fields? ☐
- File encoding: (dropdown menu)

Below the settings is a table showing the detected fields:

Name	Type	Format	Length	Precision	Cu
1 id_patient	Integer	#	15	0	€
2 dose	Number	##	3	1	€
3 unit	String		5		€
4 route	String		15		€
5 interval	String		4		€
6 cod_drug	String		6		€
7 desp	String		6		€
8 prescription_date	Date	dd/MM/yyyy			€
9 prescription_time	Date	HH:mm	5		€

At the bottom are buttons: Help, OK, Get Fields, Preview, and Cancel. Red arrows from the list on the left point to the following fields: 'Delimiter', 'Lazy conversion?', 'Header row present?', 'The row number field name (opt)', 'Important: data format', and 'Get Fields'.

# 3- Read XML file

- Same as before but “Prescription.xml”
- XPath for ‘prescription’

The screenshot displays the 'Get XML Data' tool interface, which is used for extracting data from XML files. The interface is divided into several sections:

- Step name:** 'Get data from XML'
- File tab:** Contains options for XML source configuration.
  - XML source from field:** Includes checkboxes for 'XML source is defined in a', 'XML source is a filename:', and 'Read source as Url'. A dropdown menu for 'get XML source from a field' is also present.
  - File or directory:** A text input field with 'Add' and 'Browse' buttons.
  - Regular Expression:** A text input field.
  - Exclude Regular Expression:** A text input field.
  - Selected files:** A list showing 'File/Directory' and '1 \${pathPrescription}/Prescripcion.xml' with a 'Delete' button.
- Fields tab:** Contains the 'Settings' section.
  - Loop XPath:** A text input field containing the XPath expression `/*[name()='aemps_prescripcion']/*[name()='prescription']` and a 'Get XPath nodes' button.
  - Encoding:** A dropdown menu set to 'UTF-8'.

Two red arrows originate from the list items on the left. One arrow points from 'Same as before but “Prescription.xml”' to the file path `${pathPrescription}/Prescripcion.xml` in the 'Selected files' list. The other arrow points from 'XPath for ‘prescription’' to the XPath expression in the 'Loop XPath' field.

# 3- Read XML file

- Tab “Fields” -> Get fields and preview rows.
  - Get only ‘cod\_nacion’, ‘des\_prese’, ‘cod\_atc’, ‘cod\_principio\_activo’
  - **Note: Codigo\_nacional change data type: String**

The screenshot shows the 'Get XML Data' dialog box with the 'Fields' tab selected. The 'Step name' is 'Get data from XML'. The 'Fields' table lists four fields: 'cod\_nacion', 'des\_prese', 'cod\_atc', and 'cod\_principio\_activo'. The 'Type' column for the first three fields is 'String', and for the last field, it is 'Integer'. A red circle highlights the 'Type' column. Below the dialog box, the 'Examine preview data' window shows the first five rows of the data, with the first row highlighted in red.

Name	XPath	Element	Result type	Type
1 cod_nacion	*[name()='cod_nacion']	Node	Value of	String
2 des_prese	*[name()='des_prese']	Node	Value of	String
3 cod_atc	*[name()='cod_atc']	Node	Value of	String
4 cod_principio_activo	*[name()='composicion_pa']/*[name()='cod_principio_activo']	Node	Value of	Integer

Get field

OK Preview rows Cancel

Examine preview data

Rows of step: Get data from XML (5 rows)

cod_nacion	des_prese
1 600000	AMOXICILINA /ACIDO CLAVULANICO SALA 500/50 mg POLVO PARA SOLUCION INYECTABLE Y PARA PE
2 600008	CITALOPRAM FARMALIDER 10 mg COMPRIMIDOS RECUBIERTOS CON PELICULA, 500 comprimidos
3 600010	INSPIRA 50 mg COMPRIMIDOS RECUBIERTOS CON PELICULA , 200 comprimidos
4 600011	INSPIRA 25 mg COMPRIMIDOS RECUBIERTOS CON PELICULA , 200 comprimidos
5 600017	ONDANSETRON RATIOPHARM 4 mg COMPRIMIDOS RECUBIERTOS CON PELICULA EFG , 500 comprimidos

Close Show Log

# 3- Lookup in stream

- Set stream to lookup in
  - The XML file
- Key field in XML
- Field in CSV
- Fields from XML that will be inserted in the stream
  - Set data type
  - Optional renaming

The screenshot shows the 'Stream Value Lookup' dialog box. Red arrows from the list on the left point to the following elements in the dialog:

- An arrow from 'Set stream to lookup in' points to the 'Step name' field, which contains 'Lookup Description'.
- An arrow from 'Key field in XML' points to the 'Lookup step' field, which contains 'Read meds in XML'.
- An arrow from 'Field in CSV' points to the 'Field' column header in the 'The key(s) to look up the value(s):' table.
- An arrow from 'Fields from XML that will be inserted in the stream' points to the 'Field' column header in the 'Specify the fields to retrieve :' table.

The dialog box contains the following sections:

Step name:

Lookup step:

The key(s) to look up the value(s):

Field	LookupField
1 cod_drug	cod_nacion

Specify the fields to retrieve :

Field	New name	Default	Type
1 des_prese	description		String
2 cod_princip			String
3 cod_atc			String

Preserve memory (costs CPU) ☒

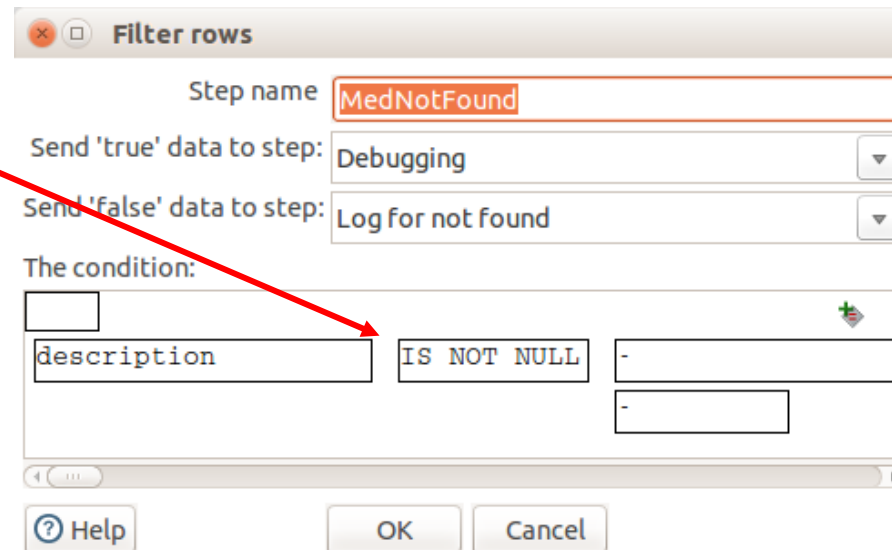
Key and value are exactly one integer field ☐

Use sorted list (i.s.o. hashtable) ☐

Buttons:

# 3- Filter

- Discard if description is null



Filter rows

Step name: MedNotFound

Send 'true' data to step: Debugging

Send 'false' data to step: Log for not found

The condition:

description	IS NOT NULL	-
		-

Help OK Cancel



# 3- Write to log and errors to file

- Set a text file to register errors
  - Set folder `${pathErrors}/pres_error.txt`
- Set fields needed for debugging
  - Row number is usefull

**Text file output**

Step name:

File | Content | Fields

Filename:

Run this as command instead? ☐

Pass output to servlet ☐

Create Parent folder ☒

Do not create file at start ☒

Accept file name from field? ☐

File name field:

Extension:

Include stepnr in filename? ☐

Include partition nr in filename? ☐

Include date in filename? ☒

Include time in filename? ☒

Specify Date time format ☐

Date time format:

Add filenames to result ☐

**Write to log**

Step name:

Log level:

Print header ☒

Limit rows? ☐

Nr of rows to print:

Write to log:

Fields

Field
1 row_number
2 cod_drug
3 cod_atc
4 description

# Stop and verify everything is ok

- Execute the transformation
  - Get data from xml: 19994 rows
  - Text file: 10 rows
- Some error in log writing?
  - Modify “Read prescription” and change data type of column “prescription\_time” to string

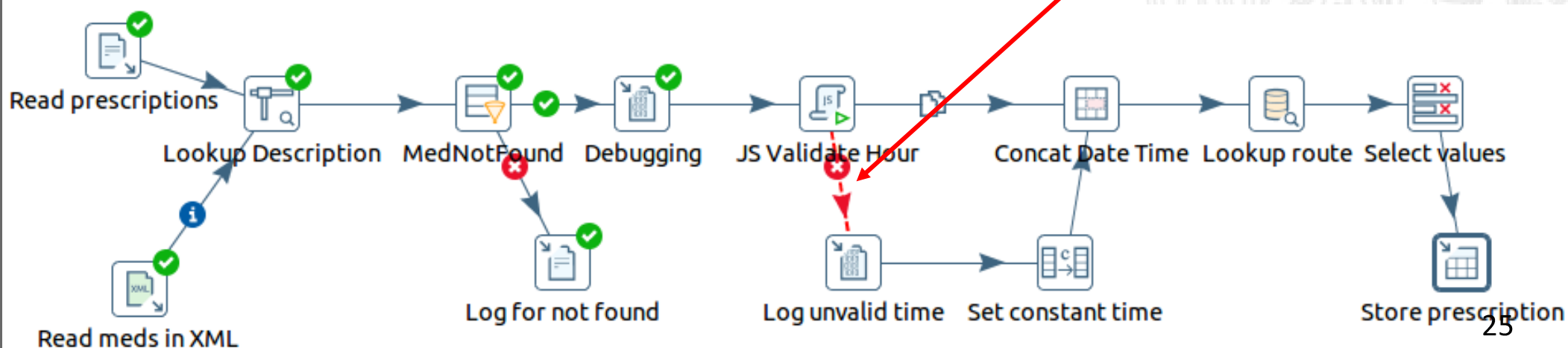




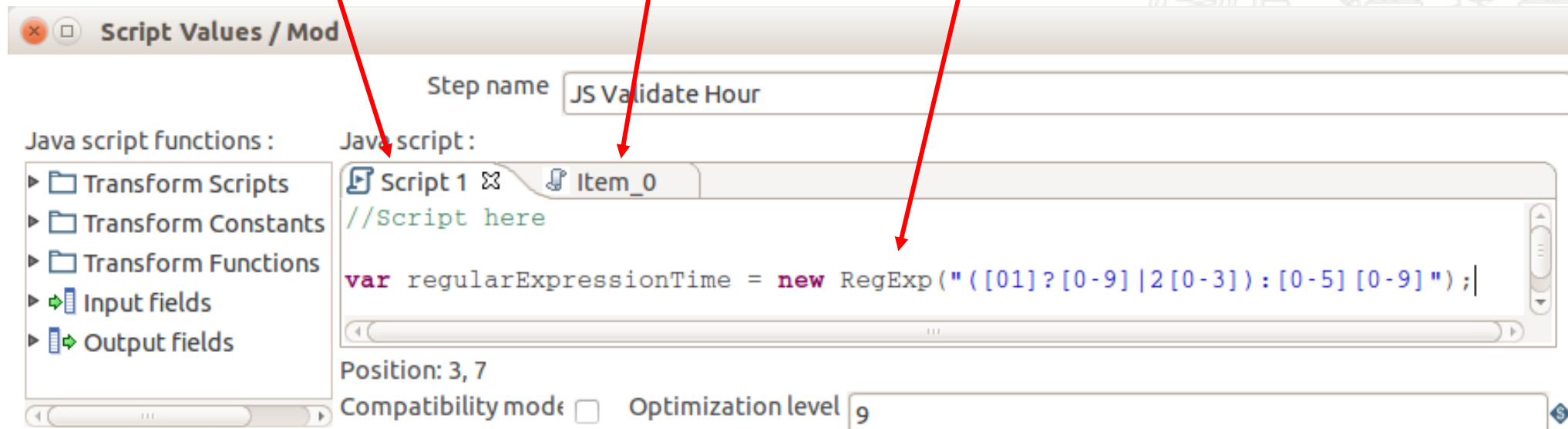
# 3- Load prescriptions - Part 2

- Scripting -> Modified Javascript Value
  - Validate hour format with javascript + regular expression
  - Transform -> Set field value to a constant: Set constant value if invalid
- Transform->Concat fields: Concat date and time in a “timestamp”
- Lookup->Database lookup: lookup route in route dimension
- Transform->Select fields: Remove unneeded fields. Set type, rename.
- Output->Table output: Store in database

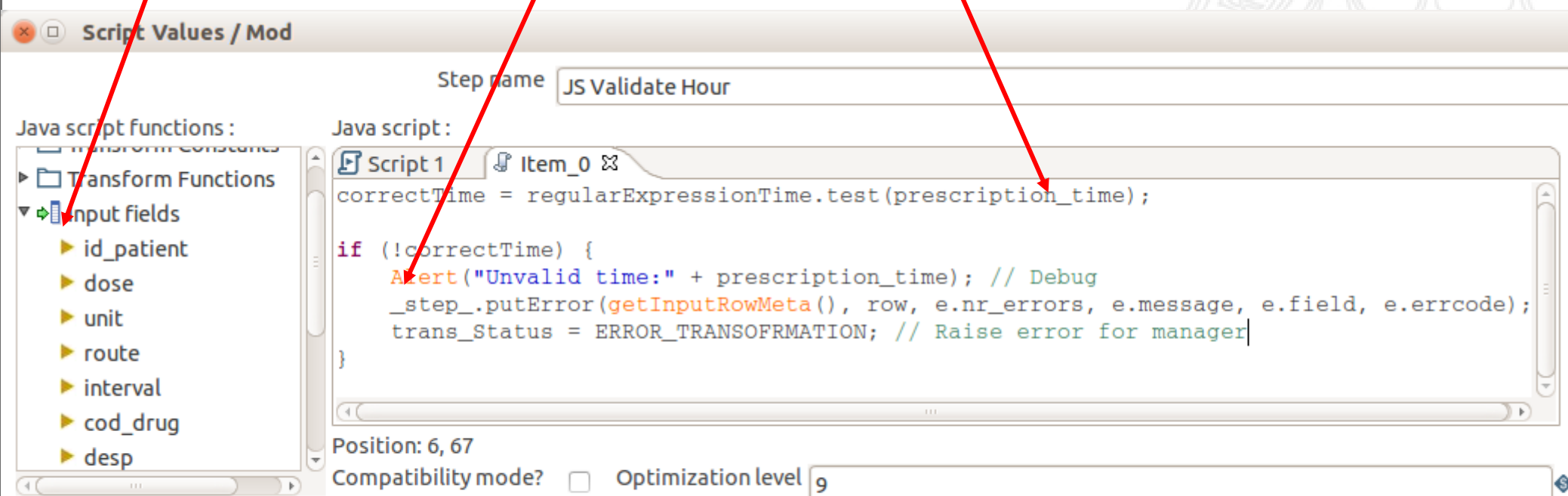
Note: error handling



- Insert new start script -> **Right click on Tab "Script 1"**
  - "Set start script" -> Executed once for the transformation
    - Initialize regular expression: **RegExp("([01]?[0-9]|2[0-3]):[0-5][0-9]")**
  - "Add New" -> "Set transform script" -> Once per row
    - Verify regular expression
  - Alternative: Validation->DataValidator



- Validate regular expression with value
  - `regularExpressionTime.test(prescription_time);`
  - Get variables from “Input fields”
- Launch error in transformation, also for debugging



Fields					
Fieldname	Rename to	Type	Length	Precision	Replace value 'Fieldname' or 'Rename to'

# 3 – JS Validate time

- If invalid time log
- Set constant value in field “prescription\_time”

**Write to log**

Step name

Log level

Print header ☒

Limit rows? ☐

Nr of rows to print

Write to log

Fields

Field
1 id_patient
2 row_number
3 cod_drug
4 prescription_date
5 prescription_time

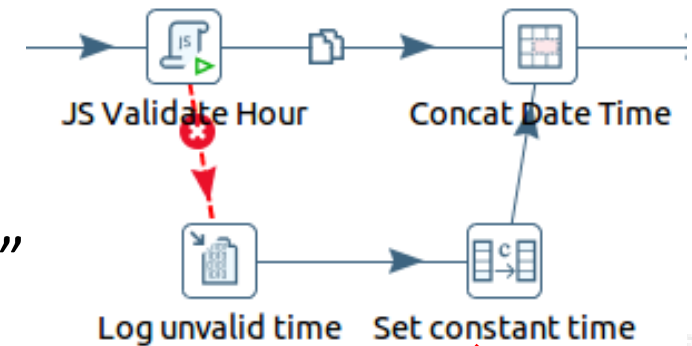
**Set field value to a constant**

Step name

Use variable in constant ☐

Fields

Field	Replace by value	Conversion mask (Date)
1 prescription_time	08:00	HH:mm



### 3 – Concat time and date

- Concat fields “prescription\_date” and “prescription\_time” in new stream field: “new\_date\_time”
- Check data types
- **Note Separator: blank space (not empty field)**

Concat Fields

Step name: Concat Date Time

Target Field Name: new\_date\_time

Length of Target Field: 0

Separator:  Insert TAB

Enclosure: "

Fields Advanced

	Name	Type	Format
1	prescription_date	Date	dd/MM/yyyy
2	prescription_time	Date	HH:mm

Get Fields Minimal width

Help OK Cancel

# 3 – Lookup route

- Select **your\_SCHEMA.dw\_dim\_via**
  - Set lookup field: “route”
  - Set table field: “desc\_via”
  - Return “cod\_via”
    - Include type

The screenshot shows the 'Database Value Lookup' dialog box. Red arrows indicate the configuration steps:

- An arrow from 'your\_SCHEMA' points to the 'Lookup schema' field, which contains 'dw\_in00'.
- An arrow from 'dw\_dim\_via' points to the 'Lookup table' field, which contains 'dw\_dim\_via'.
- An arrow from 'Set lookup field: “route”' points to the 'Field1' column in the 'The key(s) to look up the value(s):' table, which contains 'route'.
- An arrow from 'Set table field: “desc\_via”' points to the 'Table field' column in the same table, which contains 'desc\_via'.
- An arrow from 'Return “cod\_via”' points to the 'Field' column in the 'Values to return from the lookup table:' table, which contains 'cod\_via'.
- An arrow from 'Include type' points to the 'Type' column in the same table, which contains 'Integer'.

Configuration details:

- Step name: Lookup route
- Connection: tut4
- Lookup schema: dw\_in00
- Lookup table: dw\_dim\_via
- Enable cache? ☐
- Cache size in rows (0=cache everything): 0
- Load all data from table ☐

The key(s) to look up the value(s):

Table field	Comparator	Field1	Field2
1 desc_via	=	route	

Values to return from the lookup table:

Field	New name	Default	Type
1 cod_via			Integer

Do not pass the row if the lookup fails ☐

Fail on multiple results? ☐

Order by

Buttons: Help, OK, Cancel, Get Fields, Get lookup fields

### 3 – Select data and introduce metadata

- Tab “Select & alter -> Get fields: all
- Tab “Meta-data:”
- Set “Timestamp” type and format “dd/MM/yyyy HH:mm” for field “new\_date\_time” we concat

Select / Rename values

Step name: Select values

Select & Alter Remove Meta-data

Fields to alter the meta-data for :

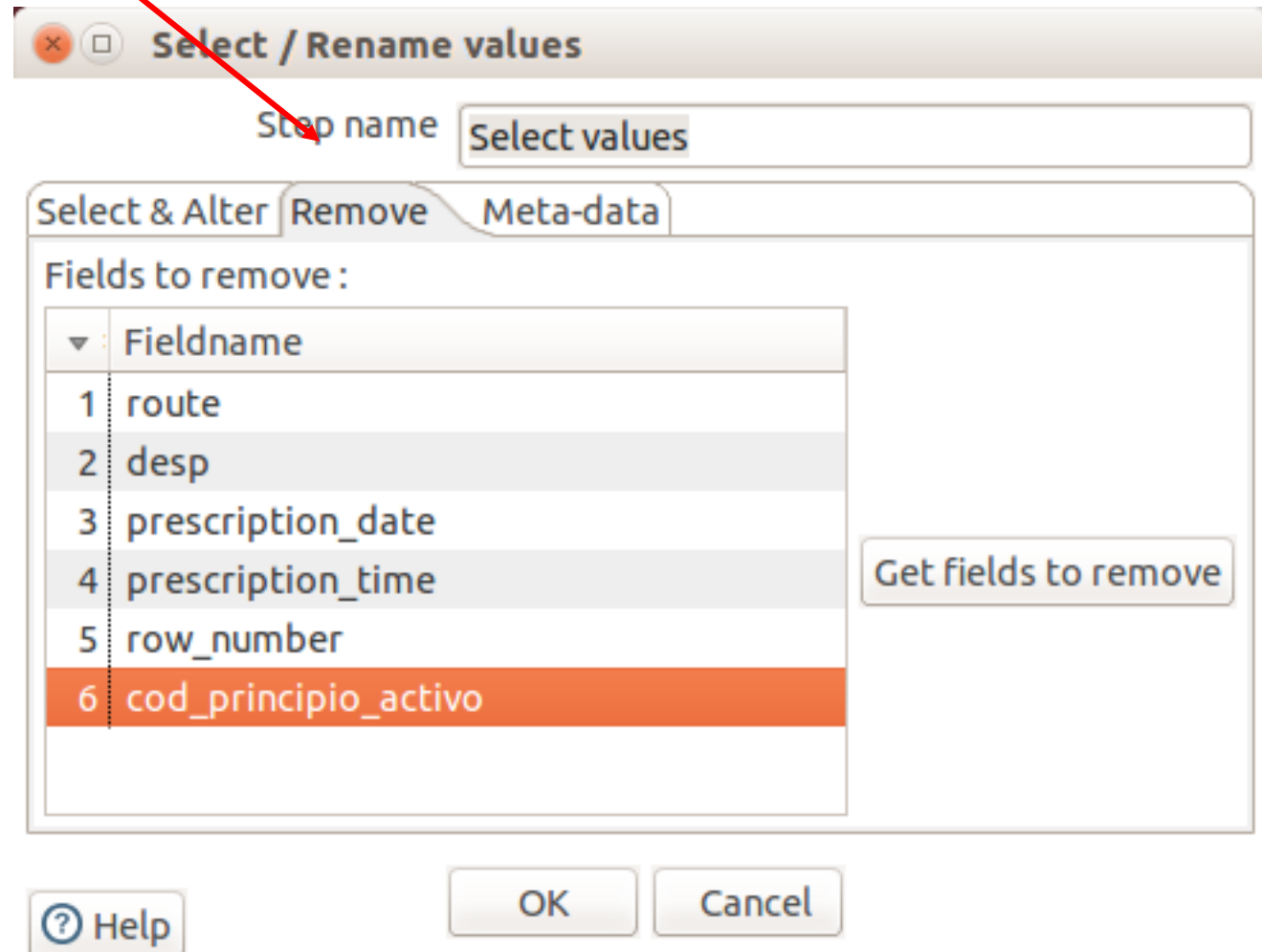
Fieldname	Rename to	Type	Lengt	Precisio	Binar	Format	Date For
1 new_date_time		Date				dd/MM/yyyy HH:mm	

Get fields to change

Help OK Cancel

### 3 – Select data and introduce metadata

- Tab “Remove”
- Include only unneeded fields:





# 3- Store prescriptions

- Select **your** **\_SCHEMA**.patient\_prescription

Salida de Tabla

Nombre paso:

Conexión:

Esquema destino:

Tabla destino:

Tamaño transacción (commit):

Vaciar tabla: ☒

Ignorar errores de inserción: ☐

Specify database fields: ☒

Main options Database fields

Fields to insert:

#	Table field	Stream field
1	cod_paciente	id_patient
2	dosis	dose
3	unidad	unit
4	via	cod_via
5	pauta	interval
6	cod_nacional	cod_drug
7	descripcion	descripcion
8	fecha	fecha_unida
9	cod_principio	cod_atc

- Remember: Before running, verify transformation
  - In case of problem with Date format parsing one of:
    - A) In input disable “lazy load” with type string
    - B) Set prescription\_time with type Date format HH:mm, and not print in in log steps

