OSCON, July 23 2009

Roland Bouman – http://rpbouman.blogspot.com
Business Intelligence Developer, Strukton Rail
Author of “Pentaho Solutions” (Wiley, ISBN: 978-0-470-48432-6)

Taming your Data

Practical Data Integration Solutions with Kettle

OSCON, July 23 2009

- Data Integration
- Kettle
- IMDB Database Import
- XML Database Import / Export
- Clustering

Topics

- Data is everywhere...
  - Sources: Flat files, Databases, Spreadsheets, Web
  - Formats: CSV, SQL, XML, Binary
- ...but how can we make sense of it?
- Data Integration
  - ETL: Extract, Transform, Load
  - Load data from multiple sources
  - Validate, Clean, Standardize
  - Kimball: “the kitchen in the BI restaurant”

Data is everywhere....

http://www.flickr.com/photos/pedromourapinheiro/2022655147/

Information: a well-prepared meal

http://www.flickr.com/photos/lizardwisdom/2462711805/

Extraction:
Rough, dangerous hardhat job

http://www.flickr.com/photos/wendycrockett/2647516433/

Cleansing and Validating: tedious and at times disgusting

http://www.flickr.com/photos/anjuli_ayer/3320839877/

Staging: storing data for further processing

http://www.flickr.com/photos/anjuli_ayer/3320848365/

Transform: Changing data structure

http://www.flickr.com/photos/mcav0y/2696641615/

Load:
Store permanently in data warehouse

Data Integration Solutions

- Programming, Scripting
  - Pros: flexible
  - Cons: require programmers, code quality, documentation, scalability
- Dedicated tool
  - Pros: works “out of the box”, visual representation, scalability
  - Cons: Expensive?

Data Integration Solutions

- Proprietary DI tools
  - Integration Services (Microsoft)
  - Business Objects Data Integrator (SAP)
  - Powercenter (Informatica)
  - Datastage (IBM)
- Open Source DI tools:
  - Kettle (Pentaho)
    - LGPL
  - Talend Open Studio (Talend)
    - GPL (v2)

Data Integration Tools

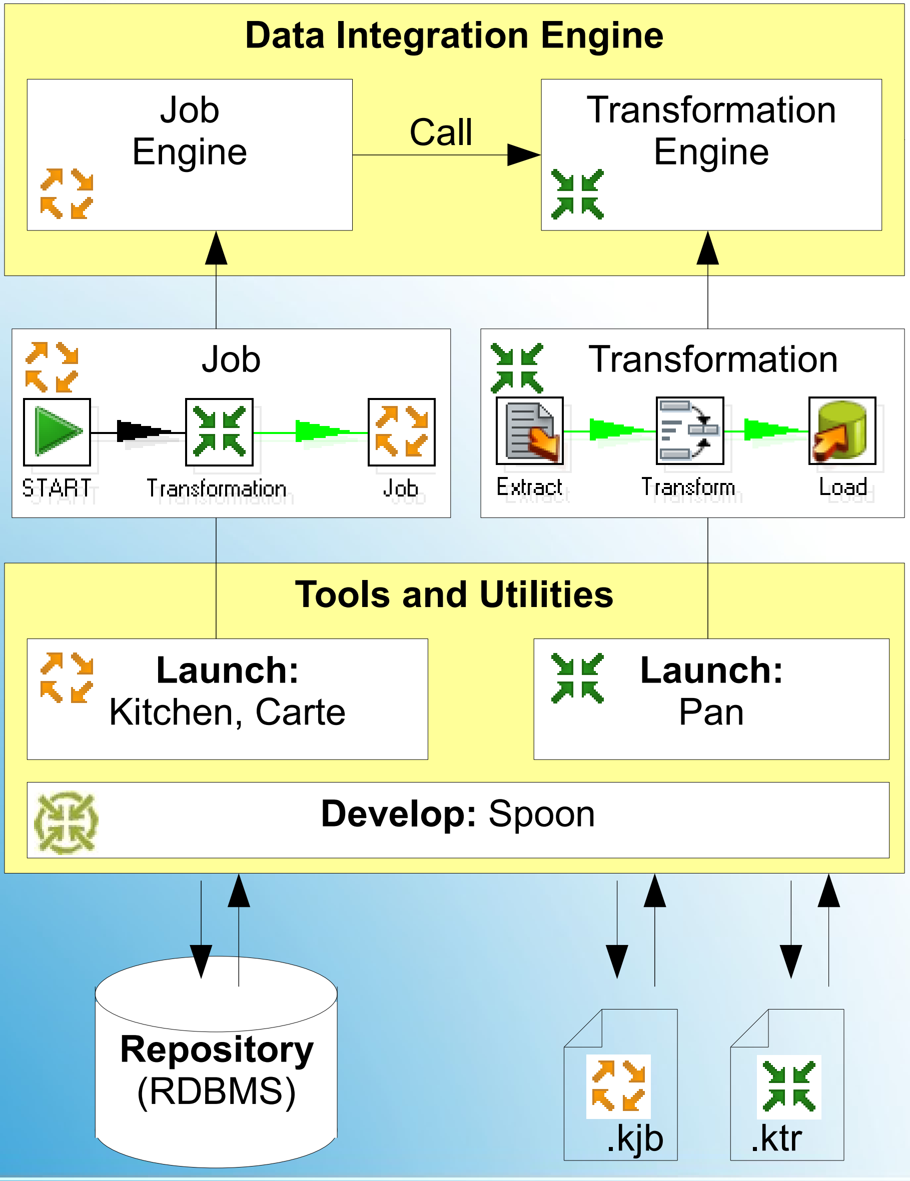
Kettle

(Pentaho Data Integration)

- K.E.T.T.L.E:
  - Kettle Extraction, Transformation, Transportation Loading Environment
  - Aka Pentaho Data Integration (PDI)
- Some random facts:
  - Java 1.5, user Interface: SWT
  - Single .zip, < 80 Mb, unzip to install
  - http://sourceforge.net/projects/pentaho
  - http://wiki.pentaho.com/ (then, Kettle > Home)
  - Extensible (Plug-in Architecture)

Kettle (Pentaho Data Integration)

- Stream Engine-based
  - Not a code generator
  - Jobs
  - Transformations
- Developer Tools:
  - Spoon
- Launcher Tools:
  - Kitchen, Pan
- Server:
  - Carte



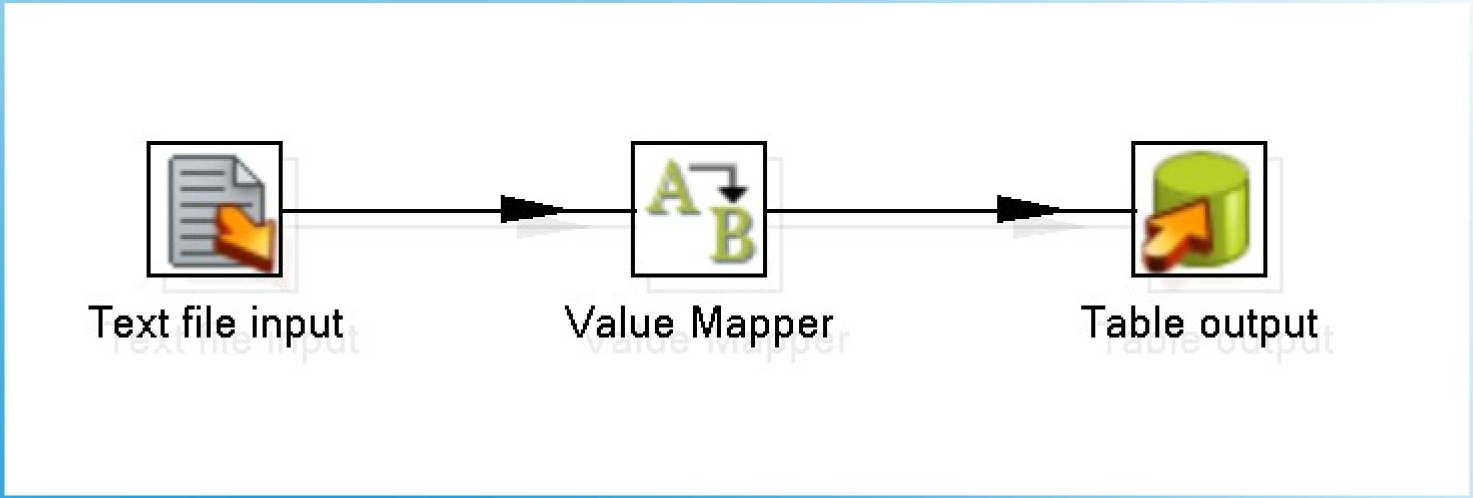
Kettle Overview

Kettle Transformations

- Transformations
  - Steps: record stream operators
  - Hops: connects steps, channels record streams
  - Steps run asynchronously
- Step Categories:
  - Input: create record stream from a resource
  - Transformation: generates output stream(s) based on input stream(s)
  - Output: serializes record stream to a resource

Kettle Transformations

- Transformation Demo
  - Extract: CSV Data (www.hometheaterinfo.com)
  - Transform: Map `UNK` (unknown) year to `NULL`
  - Load: (MySQL) Database Table



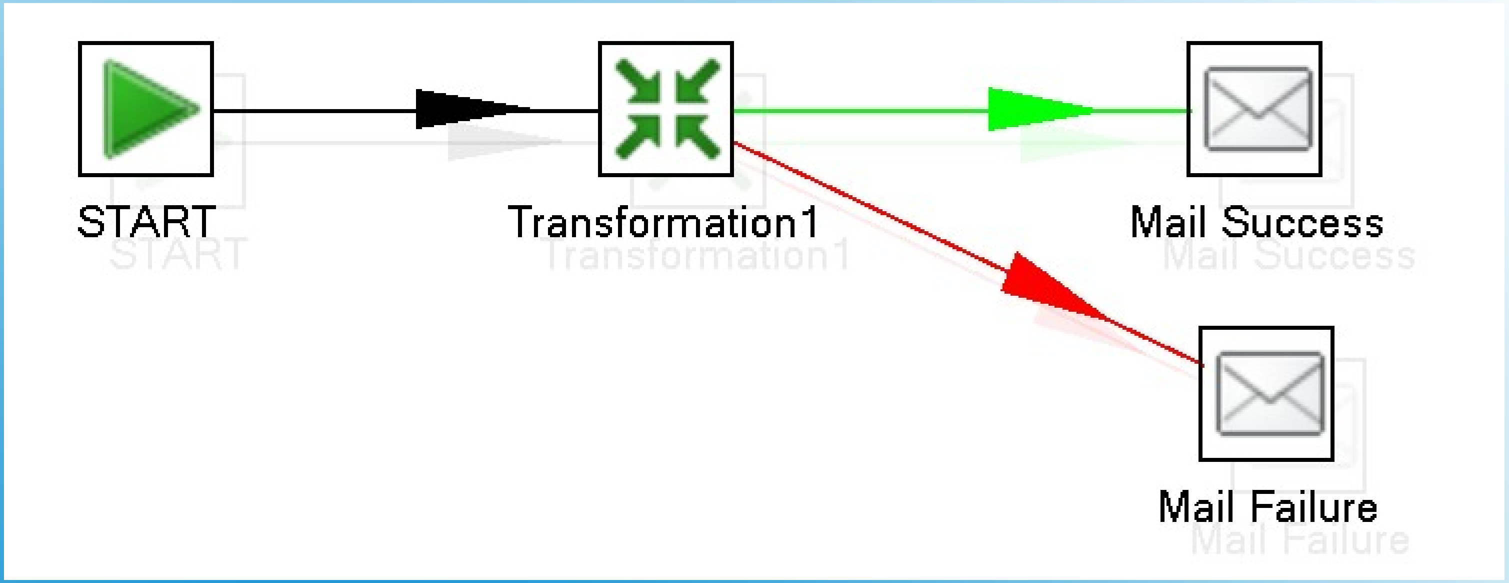
Kettle Transformation Demo

Kettle Jobs

- Jobs
  - Job entries: execute a task (synchronously)
  - Hops: defines flow of control
- Job Hops:
  - Unconditional
  - Conditional (in case of success or failure)
- Job Entries:
  - Job (job calls another job)
  - Transformation (job calls a transformation)
  - ...and many others.

Kettle Jobs

- Job Demo
  - Run Transformation
  - Send Email in case of success
  - Send Email in case of failure



Kettle Job Demo

Practical Examples

- Internet Movie Database (www.imdb.com)
- XML examples
  - Importing XML into multiple database tables
  - Export database multiple tables into XML
- Clustering examples
  - Running a transformation on multiple nodes

Practical Examples

IMDB Examples

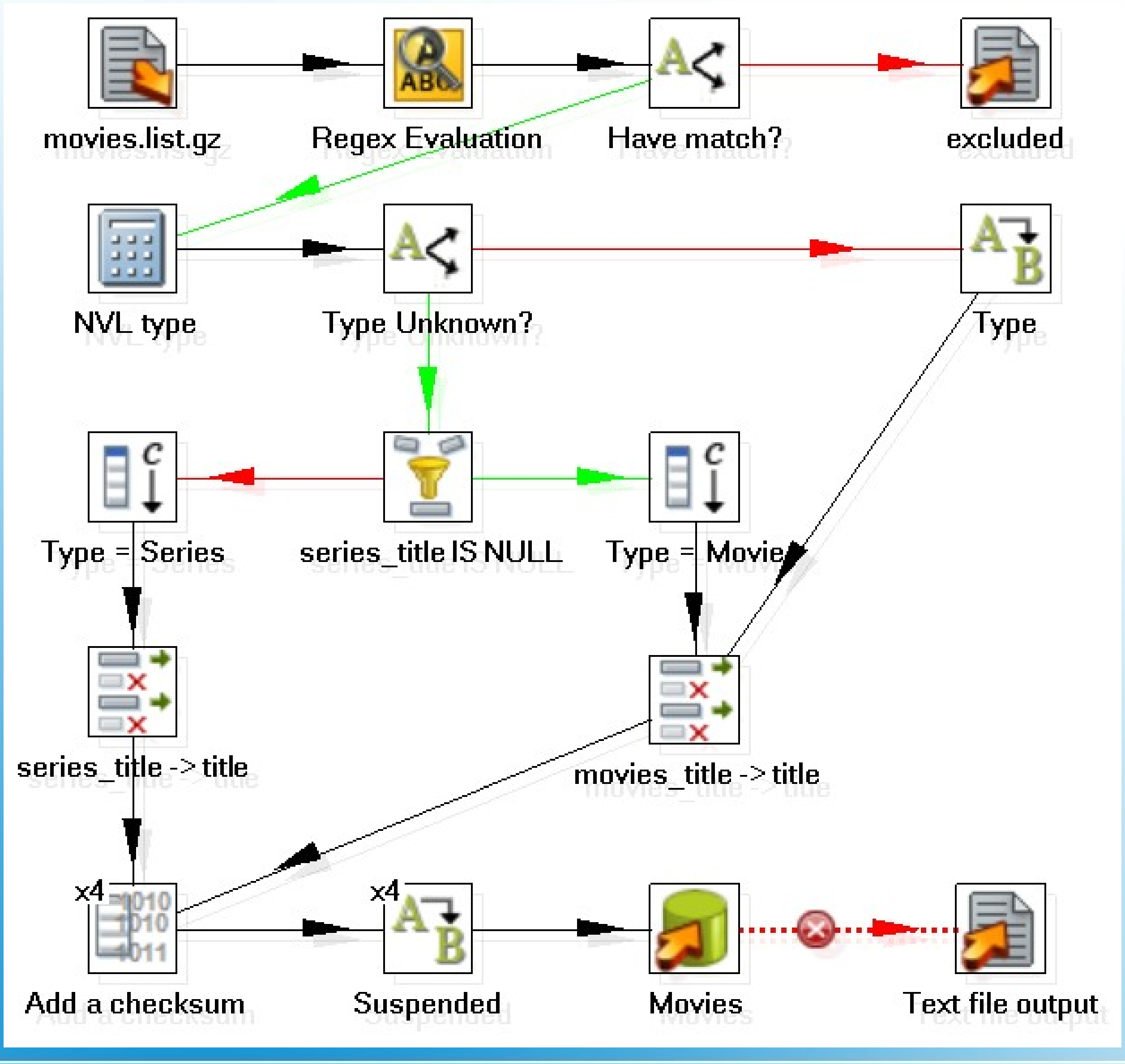
IMDB Examples

- http://www.imdb.com/interfaces#plain
- Gzipped Plaintext “lists”
- Kettle Examples
  - Movies list (using regexes)
  - Directors list (loading master-details data)
  - Laserdiscs list (pivoting rows to columns)

- Title, year, type (opt.)
- Bunch of tabs, year, extra information
- Sample Data:

```
#1 (2005)                                2005
#1 Fan: A Darkomentary (2005) (V)                2005
$100,000 Pyramid DVD Game, The (2006) (VG)               2006
$50,000 Challenge, The (1989) (TV)                   1989    (unreleased)
'Columbia' Winning the Cup (1901/I)                  1901
'Columbia' Winning the Cup (1901/II)                 1901
1 Second Film, The (2008) {{SUSPENDED}}                  2008
21 Days (1940)                               1940    (shot 1937)
Andrey Rublyov (1966)                            1966    (shot 1964-1965)
"#1 College Sports Show, The" (2004)                 2004-????
"#1 Single" (2006) {Cats and Dogs (#1.4)}                ????
"$1,000,000 Chance of a Lifetime" (1986)     1986-1987
"$10,000 Pyramid, The" (1973)                1973-1988,1991-1992
"$10,000 Pyramid, The" (1973) {(1973-03-26)}     1973
"10 Years Younger" (2004/I)                  2004
"10 Years Younger" (2004/I) {(#2.8)}             2005
```

IMDB Examples: Movies list



IMDB Examples: Movies list transformation (1/4)

- Text File Input
  - Read directly from gzip compressed file
  - Extract entire line as one field
- Regex Evaluation
  - Regular expression parses line into 'real' fields
- Check regex match
  - Unmatched titles logged to file
- Switch case according to type
  - Known type codes remapped to friendly names

IMDB Examples: Movies list transformation (2/4)

- Unknown types are either movies or series
  - Series titles are enclosed in double quotes
- Filter step separates series from movies
- Series
  - Add constant `series` for `type` field
  - Map `series_title` field to `title`
- Movies
  - Add type constant **`movie`**
  - Map `series_title` field to `title`

IMDB Examples: Movies list transformation (3/4)

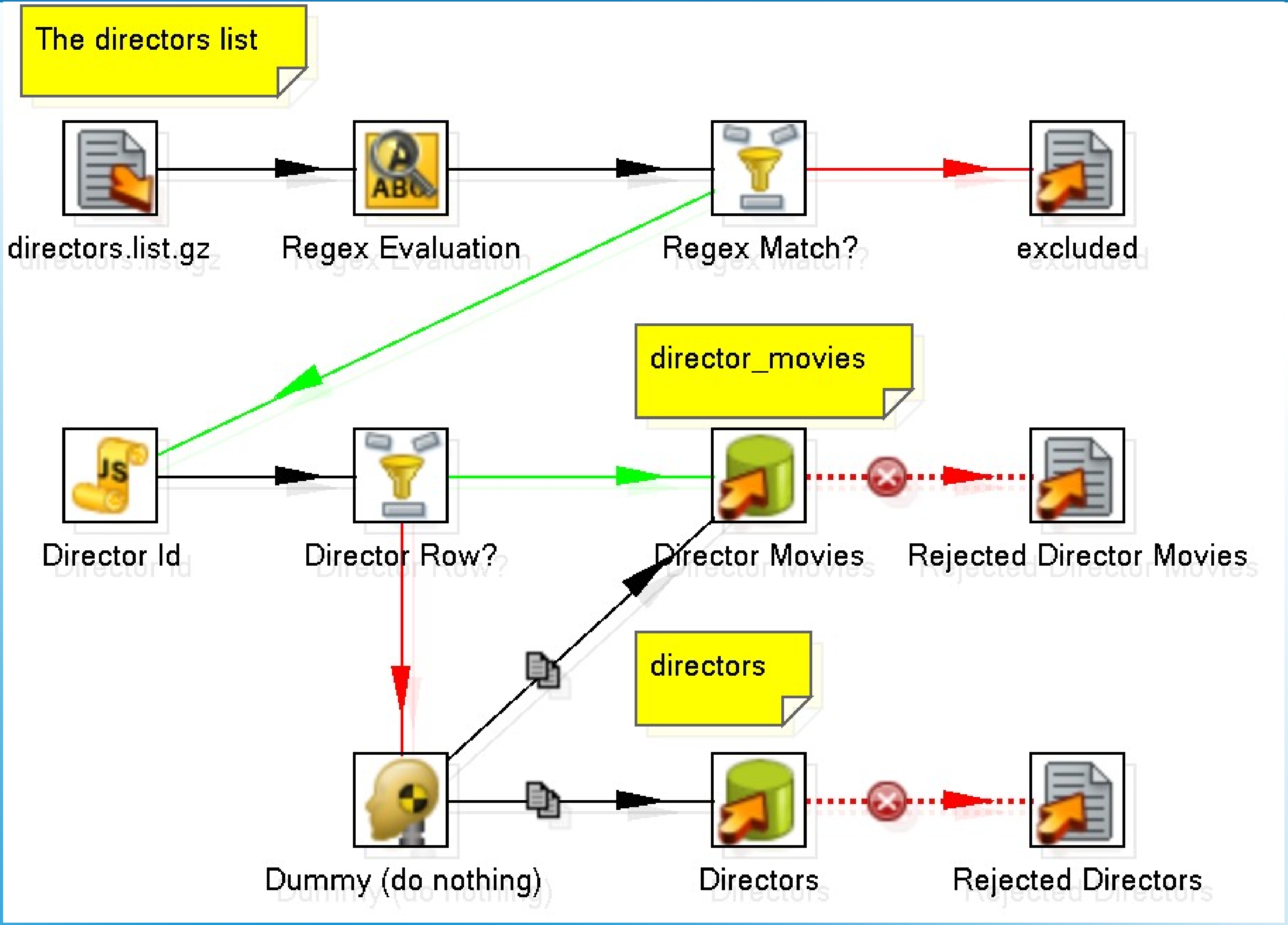
- Add a checksum
  - Expensive, use multiple copies to use more threads
- Check “suspended”
- Load into database table
  - Log errors in rejected file

IMDB Examples: Movies list transformation (4/4)

- Director name (followed by tabs)
- List of movies, directed by that director
- Optional token provides extra information
- Sample data:

```
Name                Titles
----                -------
13, Phoenix         Action Man (1998)  (V)
               Action Man 2 (1998)  (V)

93, Powers          Pas ma gueule (2006)   (co-director)

a'Hiller, Lejaren  Sleep of Cyma Roget, The (1920)   (unconfirmed)

A. Solla, Ricardo  Foto, La (2001)
               Última parada (Lo peor de todo), La (1999)
               "7 días al desnudo" (2005) {Dos vidas (#1.3)}
               "7 días al desnudo" (2005) {Fuera hace frío (#1.5)}
```

IMDB Examples: Directors list



IMDB Examples: Directors list transformation (1/3)

IMDB Examples: Directors list transformation (2/3)

- Text File Input
  - `line_number` included in stream
- Regex Evaluation
- Check match, log unmatched
- Javascript to add `director_id` to films
  - Keep copy of `row` built-in variable at start of group

```
var first_row;
var director_id;

if (last_name.getString() != null) {
    first_row = row;
}
director_id = parseInt(first_row.getString("line_number","-"));
```

- Use filter to split the stream:
  - “director” row
  - “film” rows
- “director” rows to Dummy step:
  - Duplicate the stream (as opposed to distribute)
  - Feed one back into the ordinary “film” pipeline
  - Feed the other into the “director” pipeline
- Both pipelines:
  - Insert into table
  - Log rejected rows

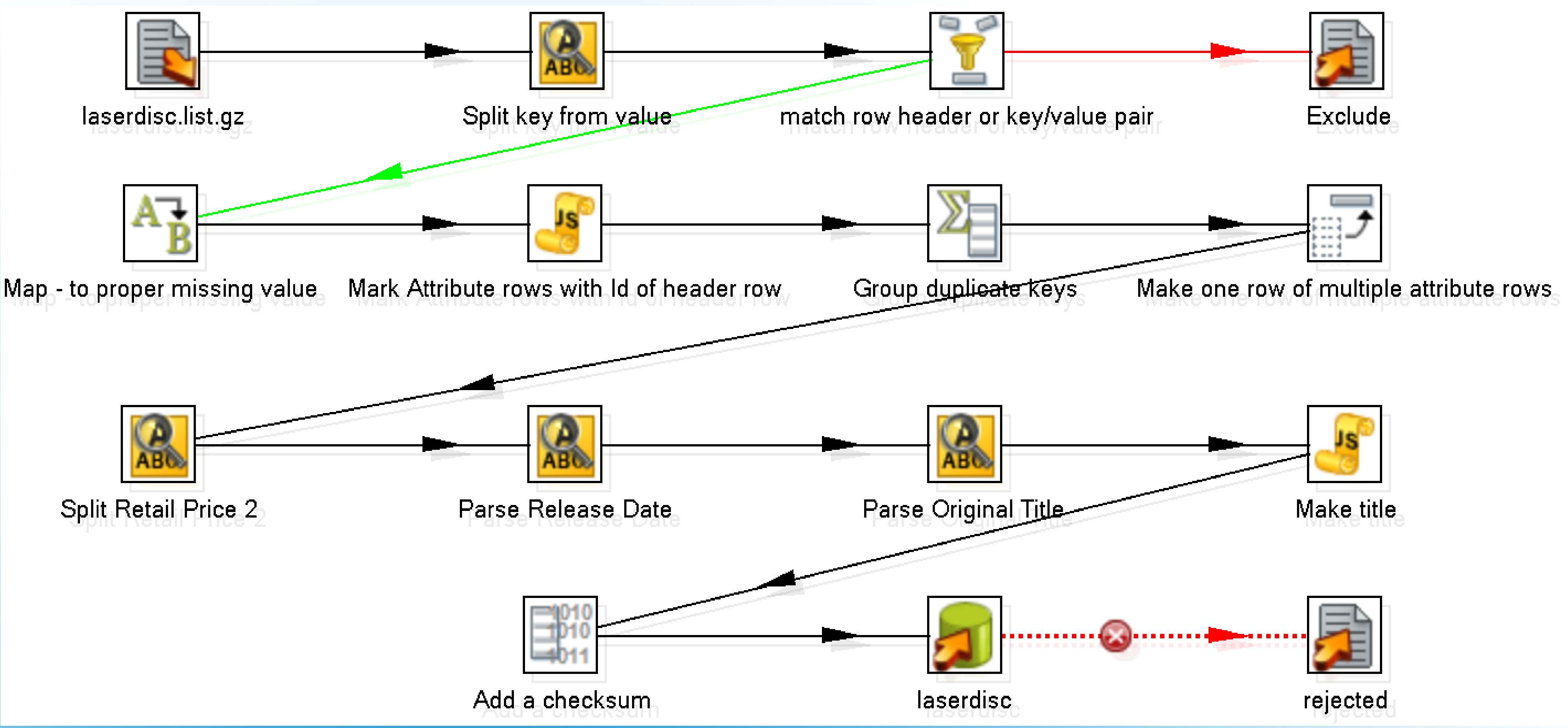
IMDB Examples: Directors list transformation (3/3)

IMDB Examples: Laserdisc list (1/2)

```
-----------------------------------------------------------------
OT: "Absolutely Fabulous" (1992)

LB: FOX Video
CN: 8289-80

LT: Absolutely Fabulous

....more lines like this...

SU: -
LE: 348

RD: 15 August 1995
ST: Available
PR: $ 99.98
RC: USA

CC: CC
QP: -
IN: Box set. Episodes: "Fashion", "Fat", "France", "Iso Tank", "Birthday",
IN: "Magazine", "Hospital", "Death", "Morocco", "New Best Friend", "Poor",
IN: "Birth".
```

- Line of dashes separates laserdiscs
- Multiple lines of key/value pairs
  - `<code>: <value>`
  - Example codes: `OT` = original title, `PR` = price
- Not all laserdisc records use all keys
- Some keys appear multiple times in the same laserdisc record

IMDB Examples: Laserdisc list (2/2)



IMDB Examples: Laserdisc list transformation (1/5)

- Initially same routine as directors list:
  - Text file Input
  - Regex Evaluation
  - Check match, log unmatched
  - Map – (dash) to `NULL`
  - Javascript adds `laserdisc_id`

```
var first_row;
var laserdisc_id;

if (header.getString()) {
    first_row = row;
}
laserdisc_id = parseInt(first_row.getString("line_number","-"));
```

IMDB Examples: Laserdisc list transformation (2/5)

IMDB Examples: Laserdisc list transformation (3/5)

- Group duplicate codes
  - Group by `laserdisc_id` and `key`
  - Must be sorted on grouping fields
  - Concatenate values

```
-----------------------------------------------------------------
OT: "Absolutely Fabulous" (1992)

....more lines like this...

IN: Box set. Episodes: "Fashion", "Fat", "France", "Iso Tank", "Birthday",
IN: "Magazine", "Hospital", "Death", "Morocco", "New Best Friend", "Poor",
IN: "Birth".
```

  - ...becomes:

```
IN: Box set. Episodes: "Fashion", "Fat", "France", "Iso Tank", "Birthday",
"Magazine", "Hospital", "Death", "Morocco", "New Best Friend", "Poor",
"Birth".
```

- Roll row-per-key to fields into a single row
  - Group by `laserdisc_id`
  - Pivot keys, map to fields

```
-----------------------------------------------------------------
OT: "Absolutely Fabulous" (1992)
LB: FOX Video
....more lines like this...
-----------------------------------------------------------------
OT: "Addams Family, The" (1964)
LB: Criterion Television
....more lines like this...
```

  - ...becomes:

```
OT                            | LB                      | ...more columns...
------------------------------+-----------------------+-------------------
"Absolutely Fabulous" (1992) | FOX Video               | ...more
"Addams Family, The" (1964)  | Criterion Television |        values...
```

IMDB Examples: Laserdisc list transformation (4/5)

- More regexes to parse data from values:
  - `retail_price`, `release_date`, `original_title`
- More javascript to construct the **`title`**
- Checksum on `original_title`
  - For joining to the data from the movies list
- Finally, store into table, and log errors

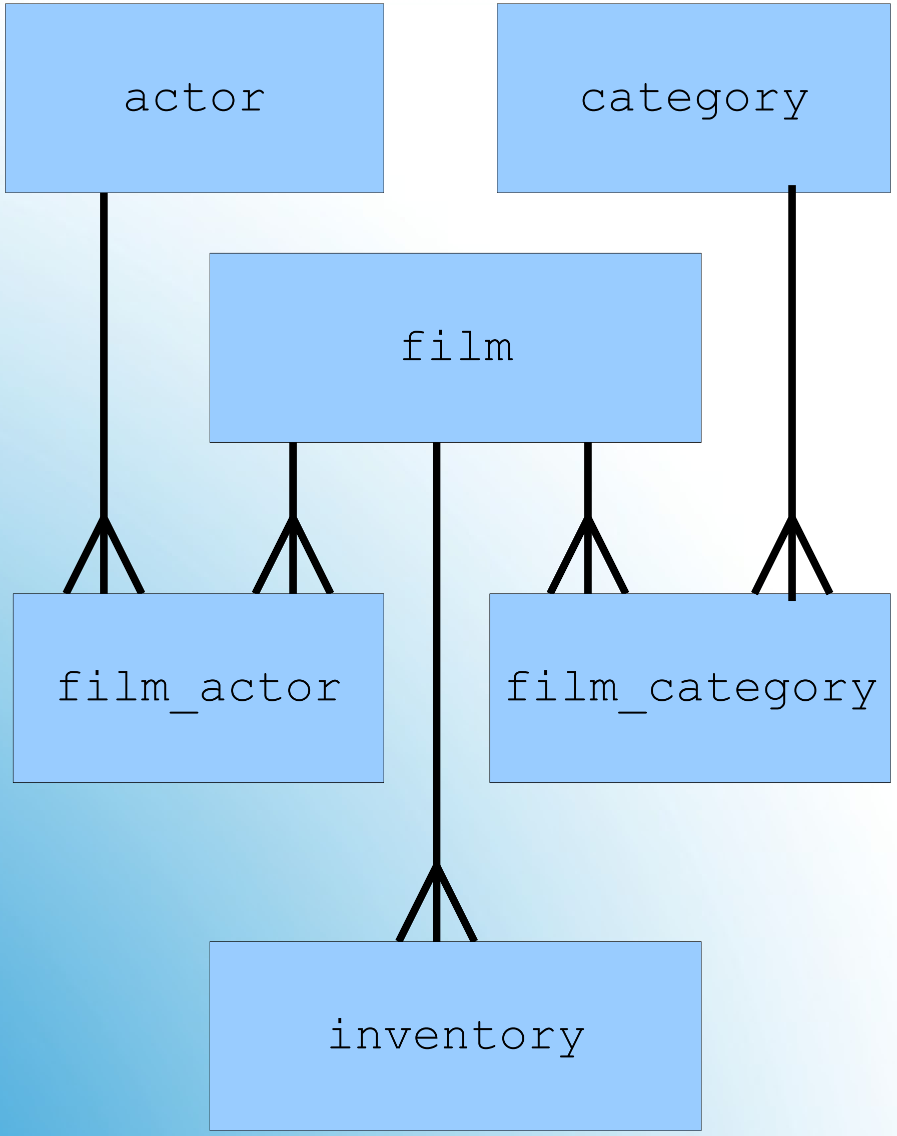
IMDB Examples: Laserdisc list transformation (5/5)

XML Examples

- http://www.stylusstudio.com/examples/videos.xml
  - Videos, Actors
- Map to the MySQL Sakila Sample database
  - http://dev.mysql.com/doc/sakila/en/sakila.html
- Examples:
  - Import XML and add to the Sakila database
  - Export from the Sakila database to XML

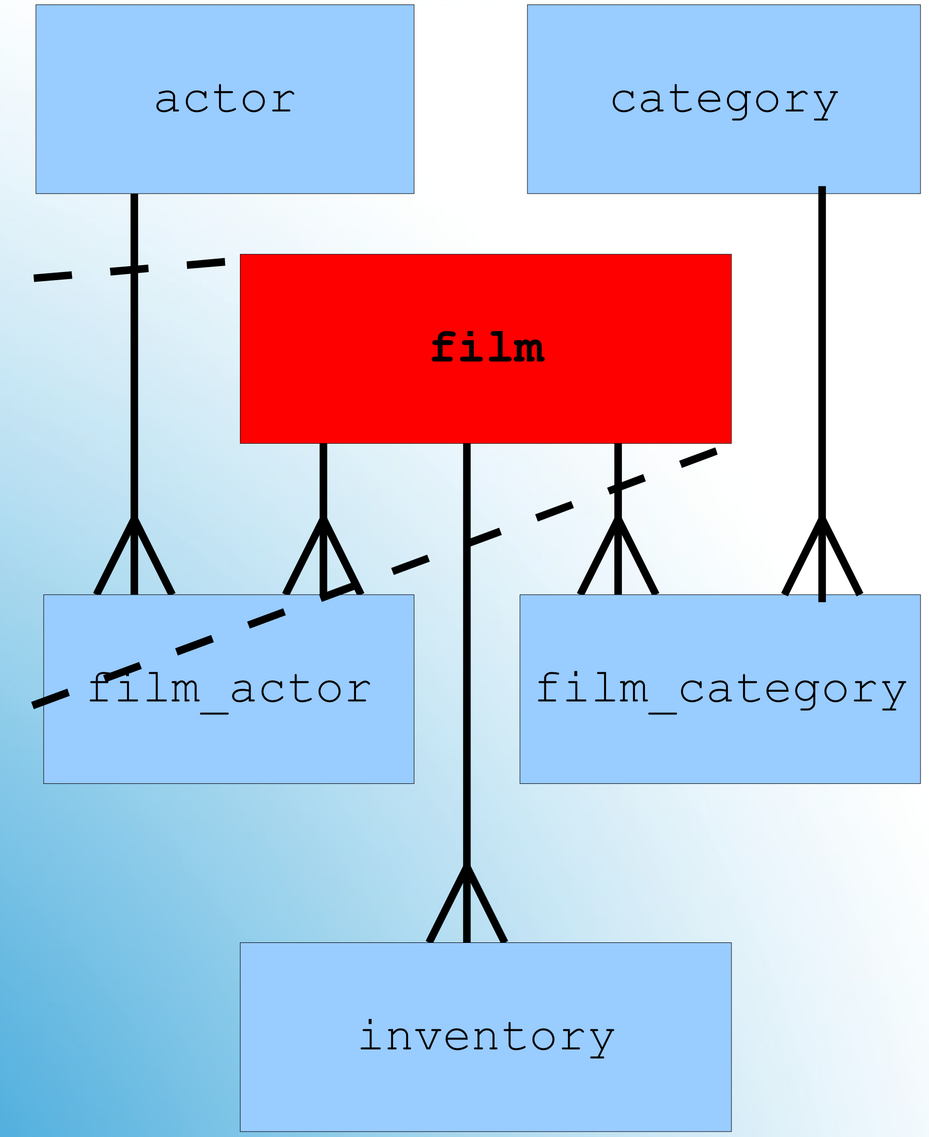
XML Examples

```
<?xml version="1.0"?>
<result>

 <actors>
  <actor id="0001">Anderson, Jeff</actor>
  ...many more <actor>...</actor>'s...
 </actors>

 <videos>
  <video id="id1235AA0">
   <title>The Fugitive</title>
   <genre>action</title>
   <rating>PG-13</rating>
   <summary>Tommy Lee and ...etc.</sumary>
   <year>1997</year>
   <actorRef>0001</actorRef>
   <actorRef>0005</actorRef>
   ...more <actorRef>...</actorRef>'s...
   <dvd>14.99</dvd>
   <dvd_stock>125</dvd_stock>
  </video>
  ...many more <video>...</video>'s...
 </videos>

</result>
```

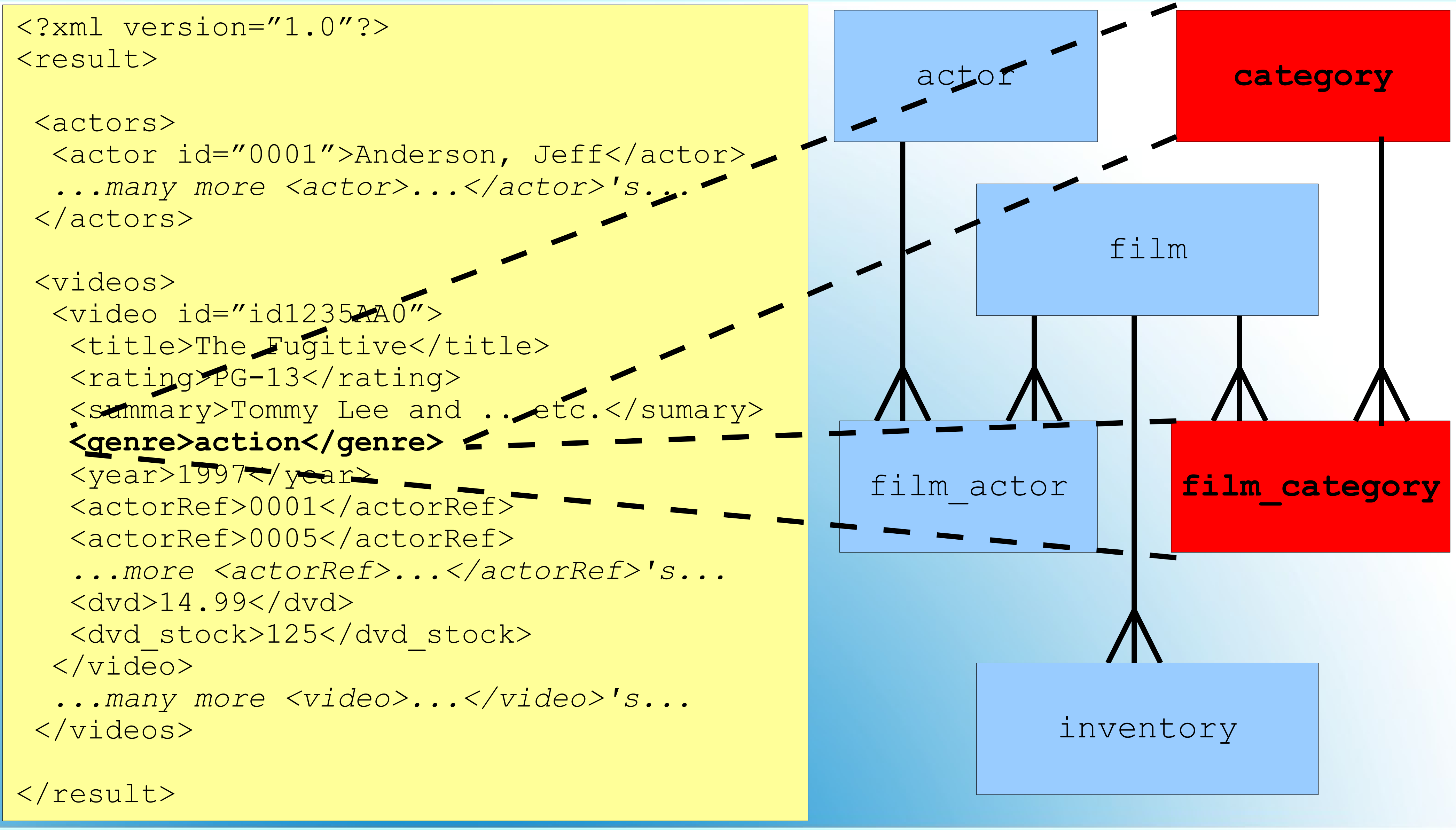


XML sample data and target database schema

```
<?xml version=”1.0”?>
<result>

 <actors>
  <actor id=”0001”>Anderson, Jeff</actor>
  ...many more <actor>...</actor>'s...
 </actors>

 <videos>
  <video id=”id1235AA0”>
   <title>The Fugitive</title>
   <rating>PG-13</rating>
   <summary>Tommy Lee and ...etc.</sumary>
   <genre>action</genre>
   <year>1997</year>
   <actorRef>0001</actorRef>
   <actorRef>0005</actorRef>
   ...more <actorRef>...</actorRef>'s...
   <dvd>14.99</dvd>
   <dvd_stock>125</dvd_stock>
  </video>
  ...many more <video>...</video>'s...
 </videos>

</result>
```



XML <videos> maps to film

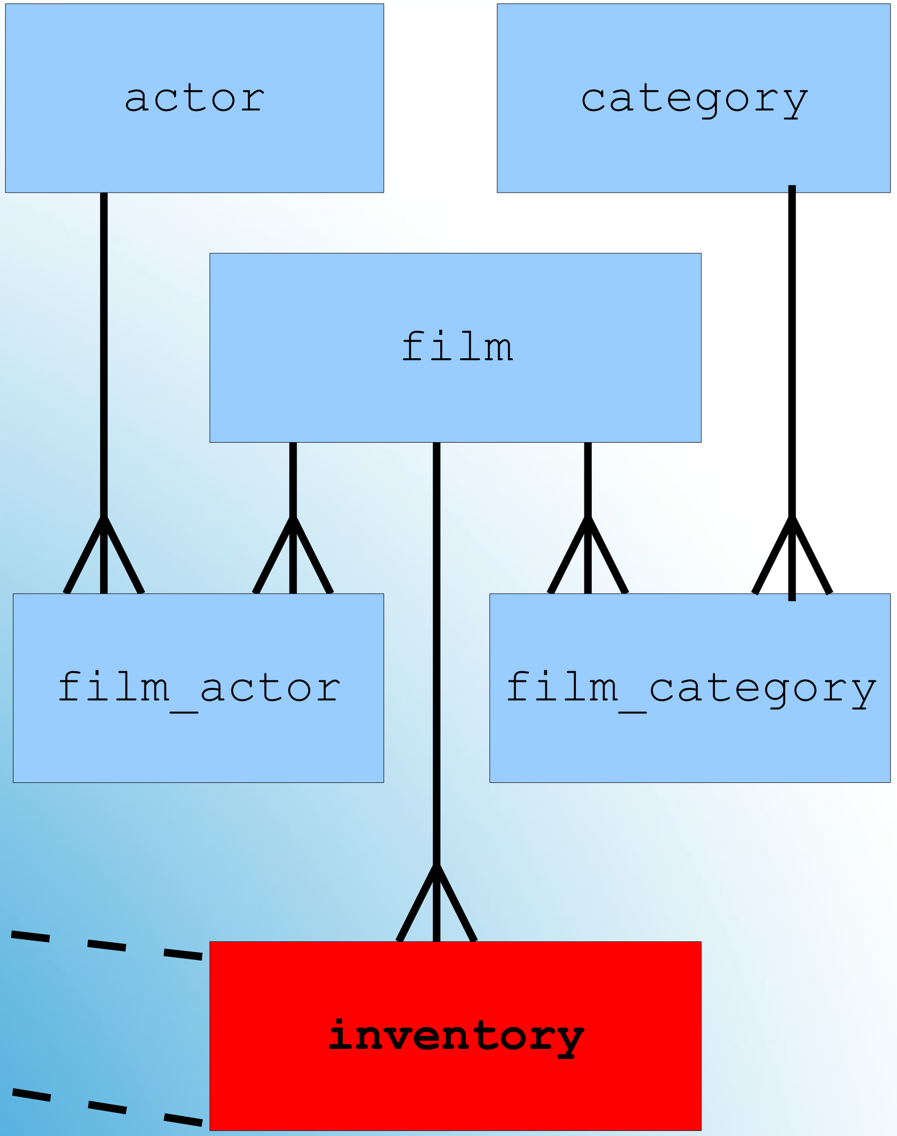


```
<?xml version="1.0"?>
<result>

 <actors>
  <actor id="0001">Anderson, Jeff</actor>
  ...many more <actor>...</actor>'s...
 </actors>

 <videos>
  <video id="id1235AA0">
   <title>The Fugitive</title>
   <rating>PG-13</rating>
   <summary>Tommy Lee and ...etc.</sumary>
   <genre>action</genre>
   <year>1997</year>
   <actorRef>0001</actorRef>
   <actorRef>0005</actorRef>
   ...more <actorRef>...</actorRef>'s...
   <dvd>14.99</dvd>
   <dvd_stock>125</dvd_stock>
  </video>
  ...many more <video>...</video>'s...
 </videos>

</result>
```

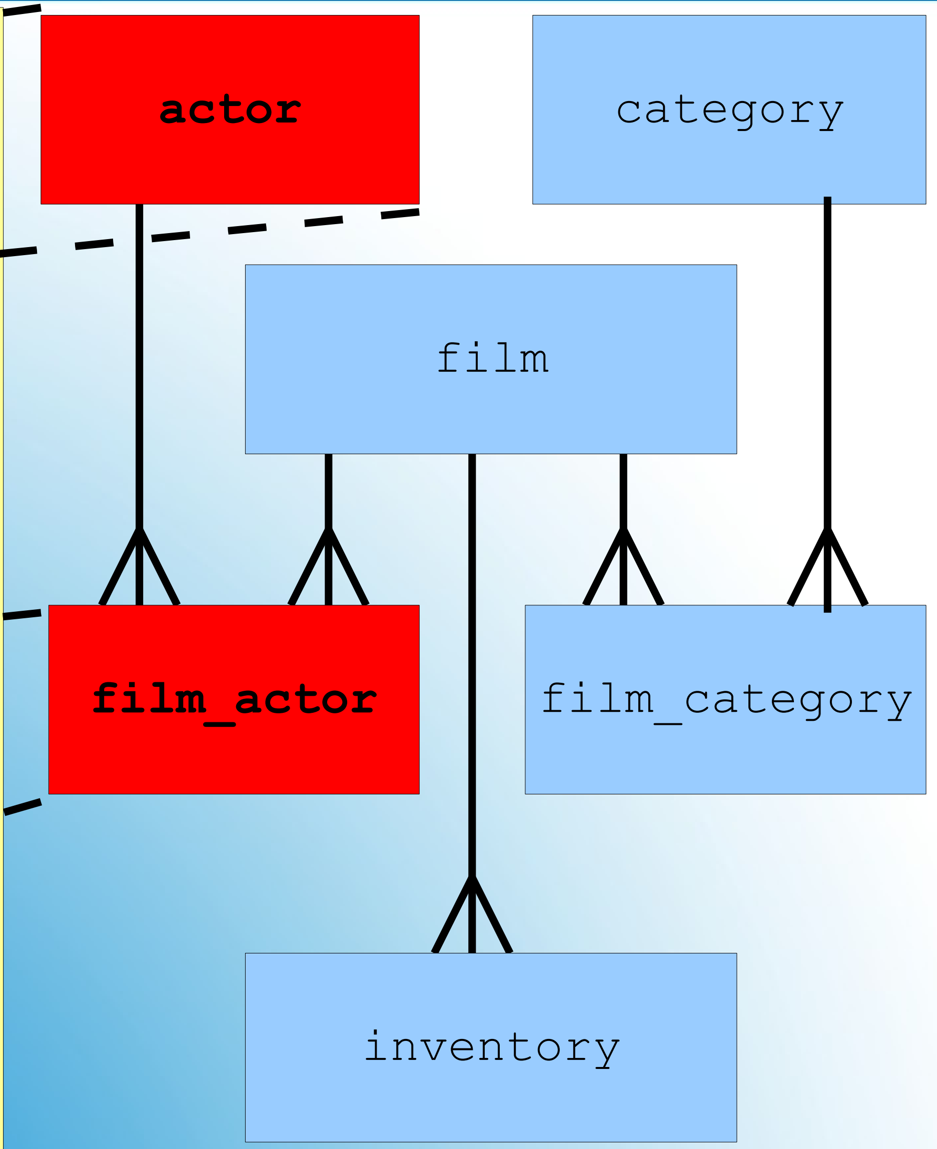
XML <genre> maps to category and film_category

XML <dvd_stock> relates to inventory

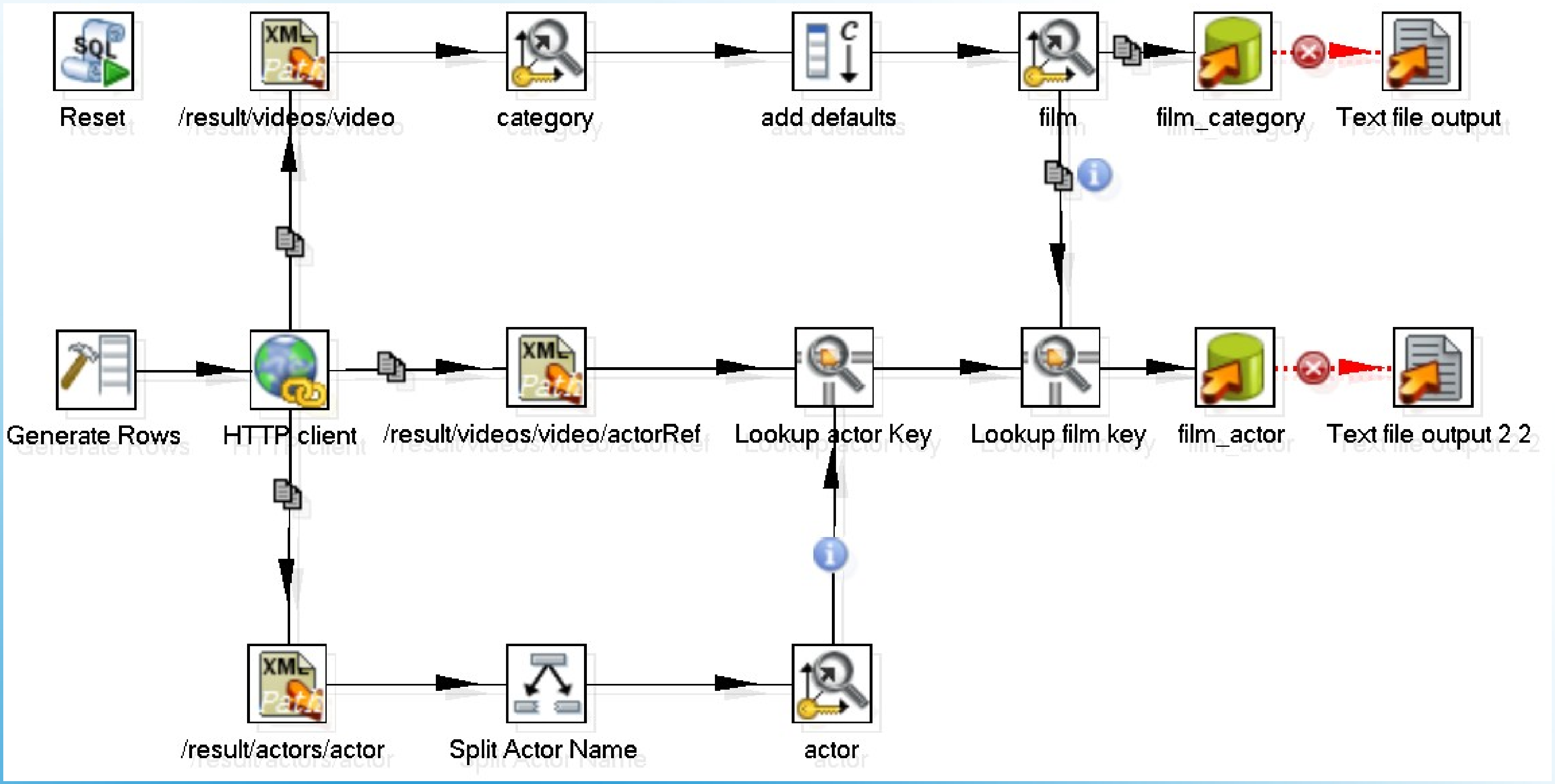
```
<?xml version="1.0"?>
<result>

 <actors>
  <actor id="0001">Anderson, Jeff</actor>
  ...many more <actor>...</actor>'s...
 </actors>

 <videos>
  <video id="id1235AA0">
   <title>The Fugitive</title>
   <rating>PG-13</rating>
   <summary>Tommy Lee and ...etc.</sumary>
   <genre>action</genre>
   <year>1997</year>
   <actorRef>0001</actorRef>
   <actorRef>0005</actorRef>
   ...more <actorRef>...</actorRef>'s...
   <dvd>14.99</dvd>
   <dvd_stock>125</dvd_stock>
  </video>
  ...many more <video>...</video>'s...
 </videos>

</result>
```



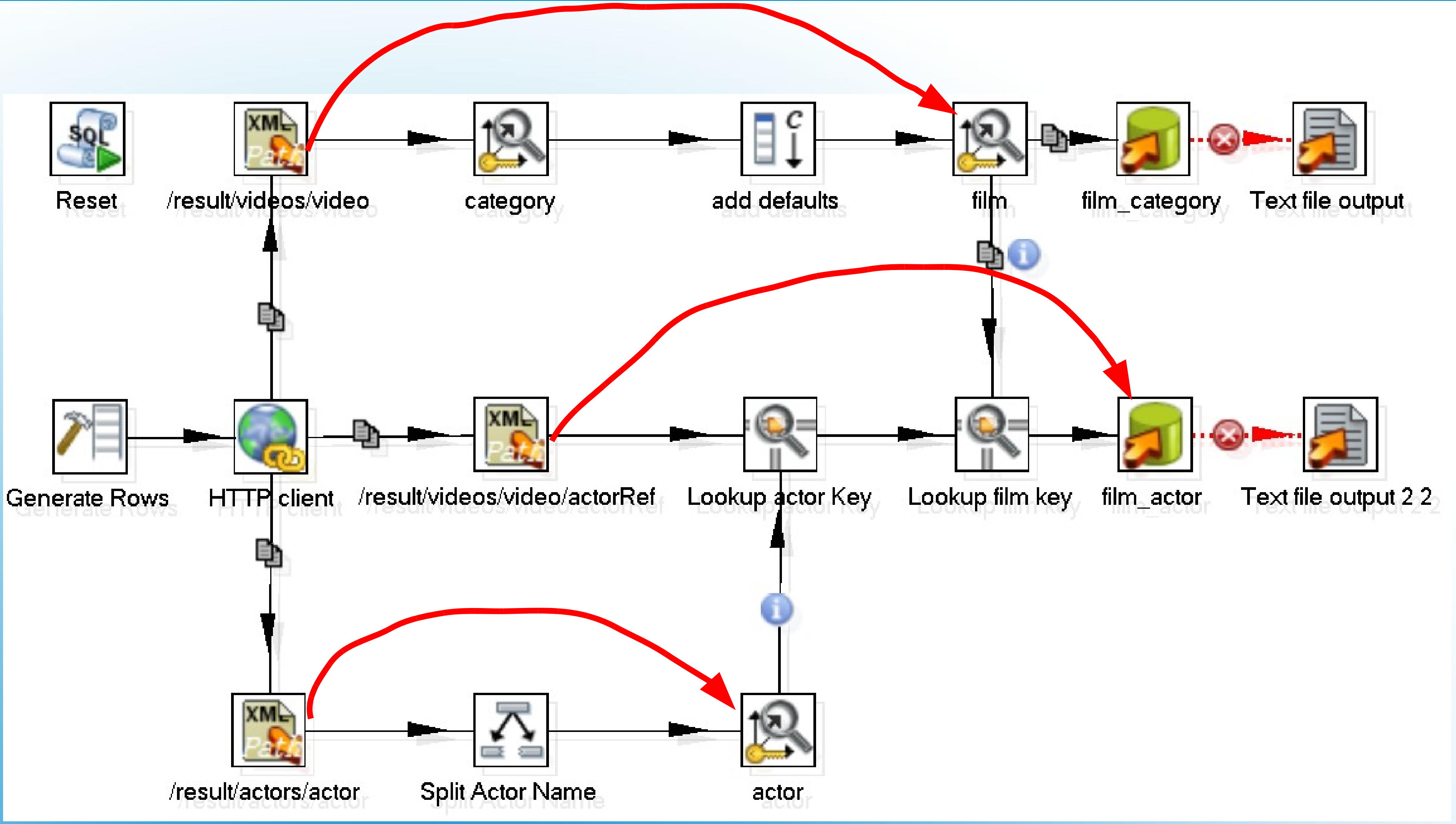
```
<?xml version="1.0"?>
<result>

 <actors>
  <actor id="0001">Anderson, Jeff</actor>
  ...many more <actor>...</actor>'s...
 </actors>

 <videos>
  <video id="id1235AA0">
   <title>The Fugitive</title>
   <rating>PG-13</rating>
   <summary>Tommy Lee and ...etc.</sumary>
   <genre>action</genre>
   <year>1997</year>
   <actorRef>0001</actorRef>
   <actorRef>0005</actorRef>
   ...more <actorRef>...</actorRef>'s...
   <dvd>14.99</dvd>
   <dvd_stock>125</dvd_stock>
  </video>
  ...many more <video>...</video>'s...
 </videos>

</result>
```



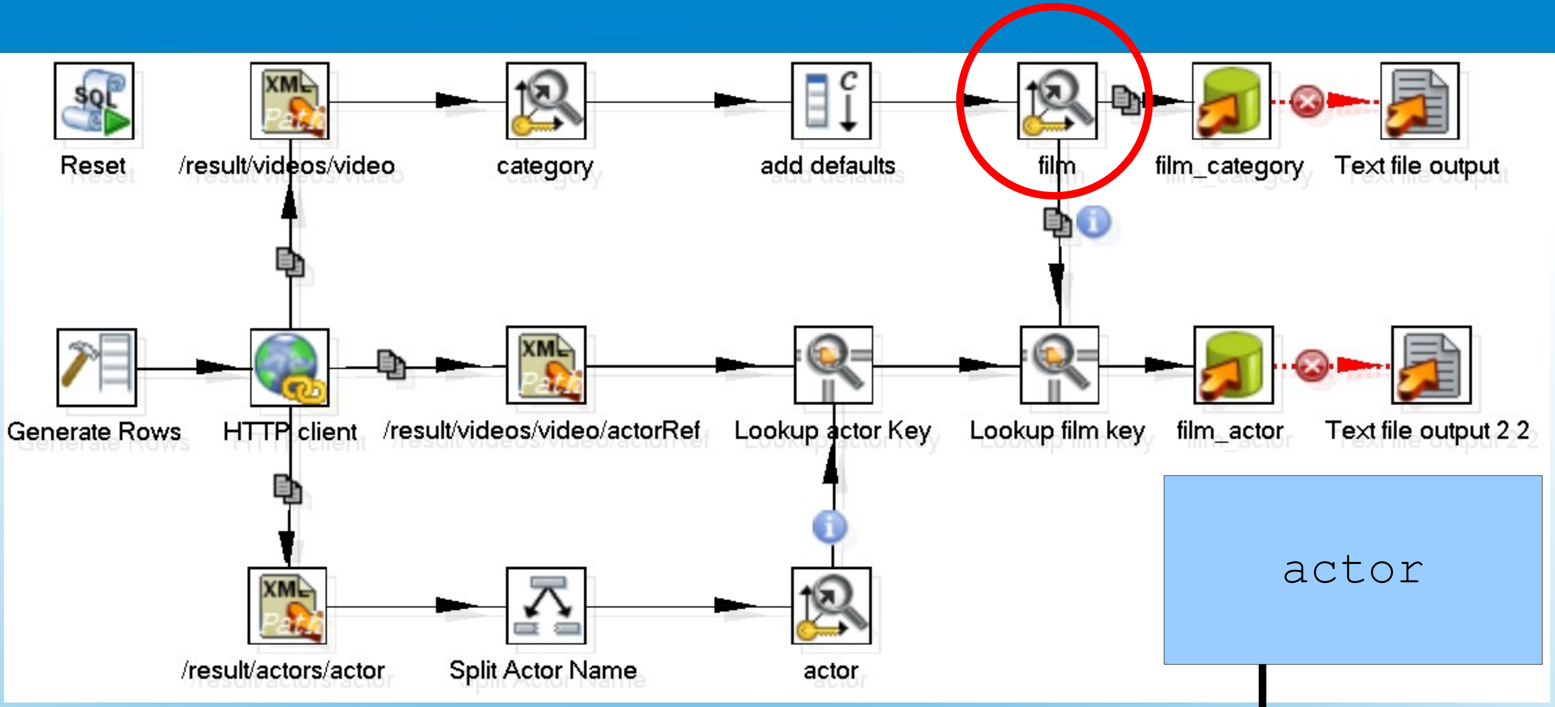
XML <actors> and <actorRef> map to actor and film_actor

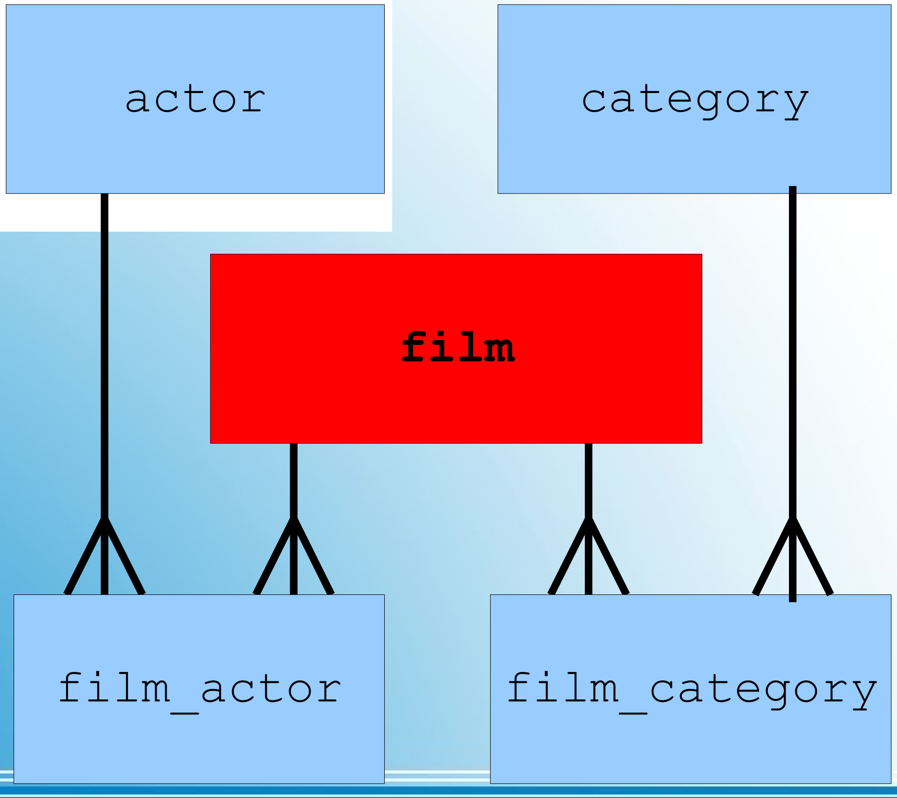


XML Examples: import `videos.xml` into db

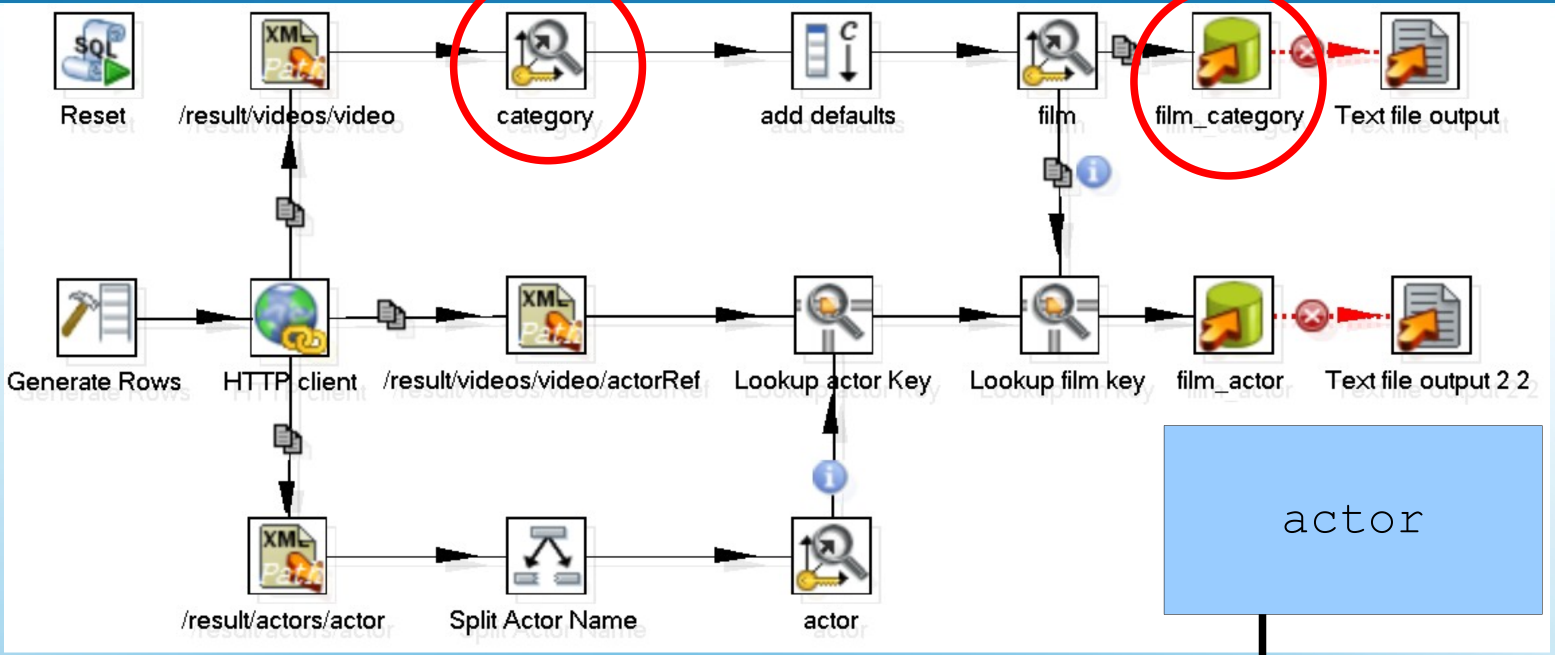


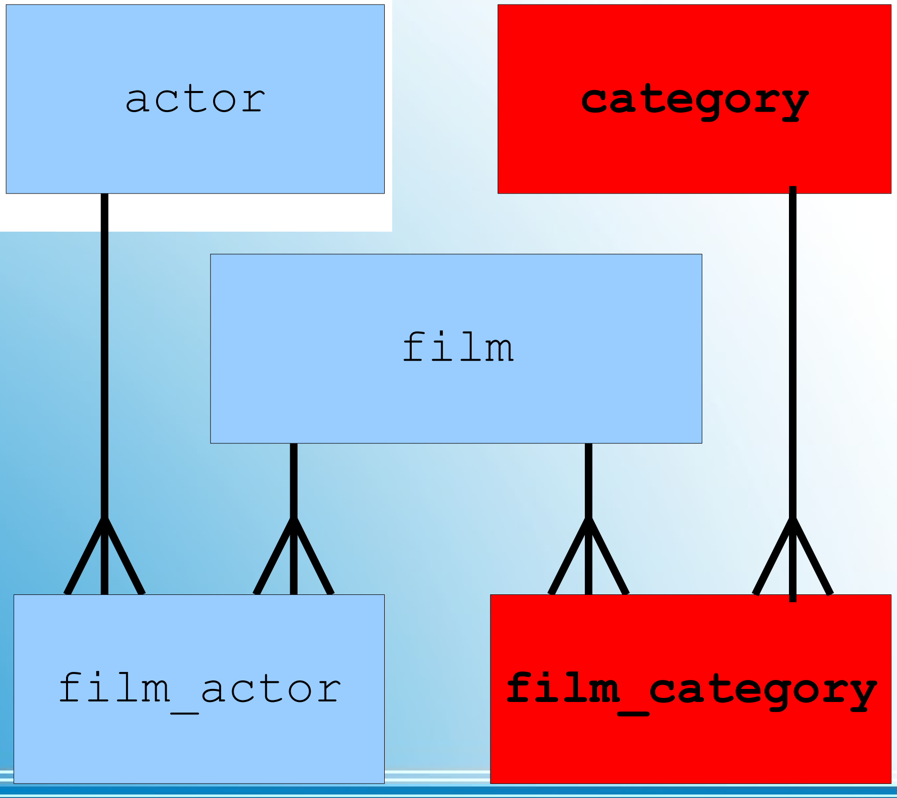
XML Import Example: Main Data Flow



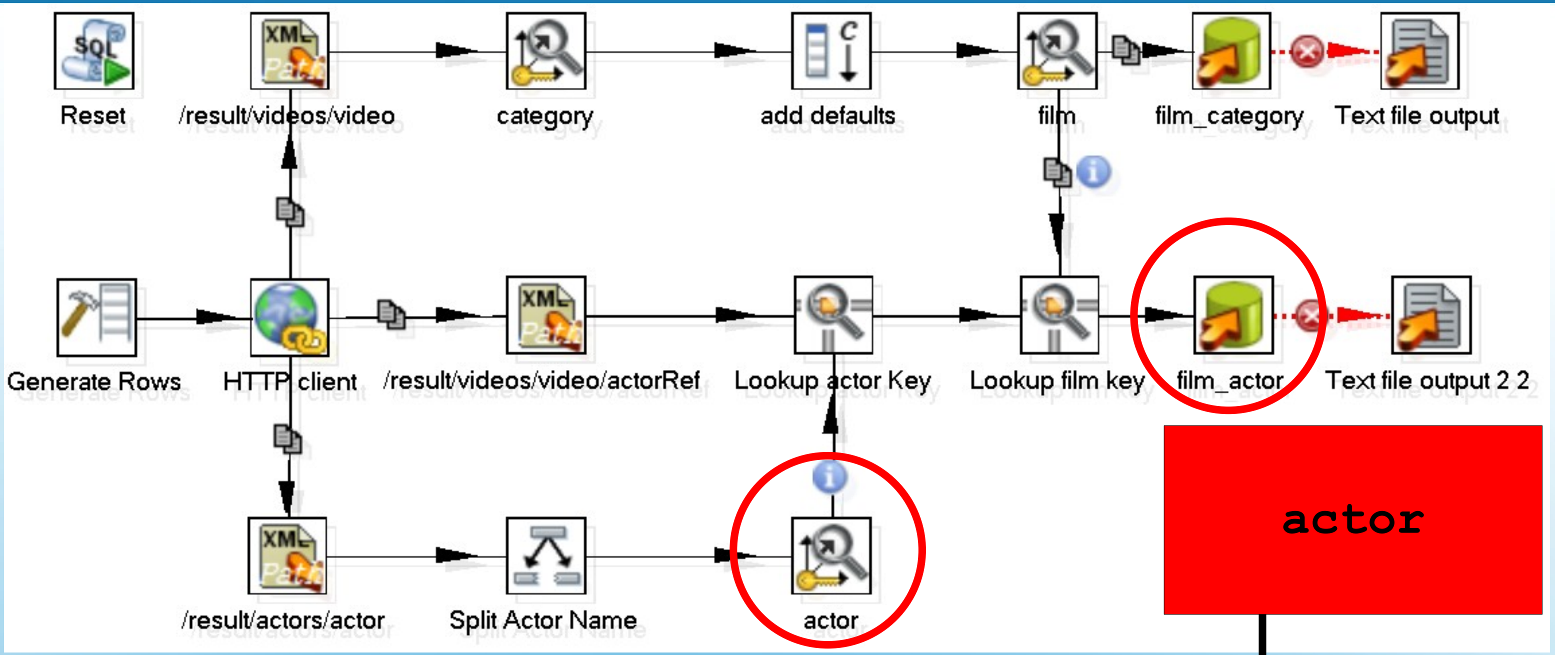


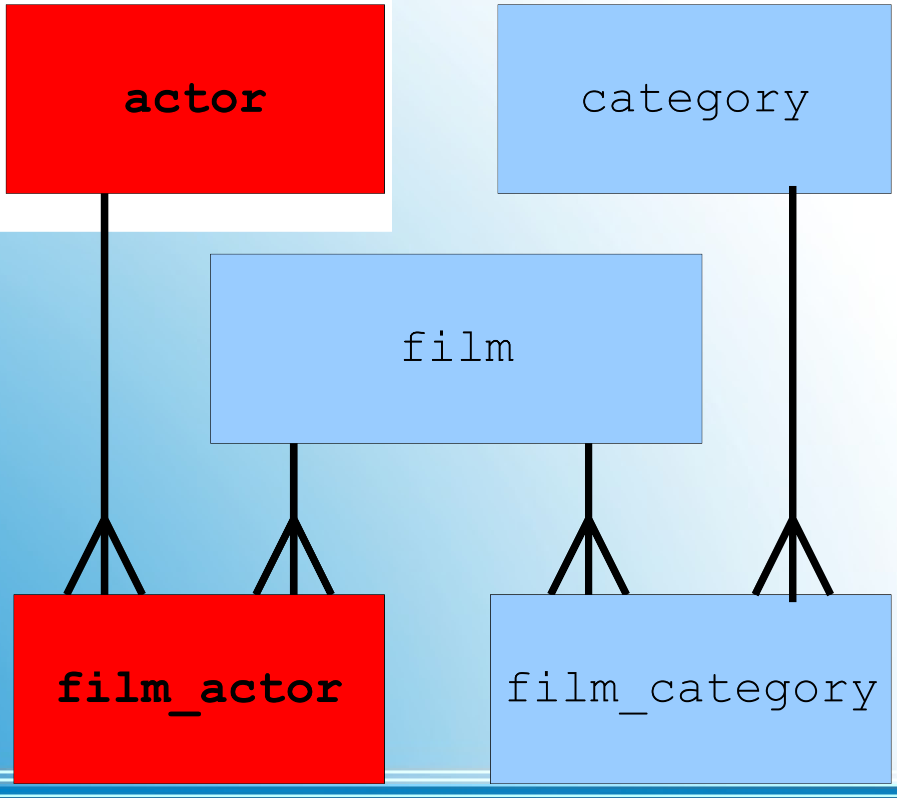
XML Import Example: `film_table`



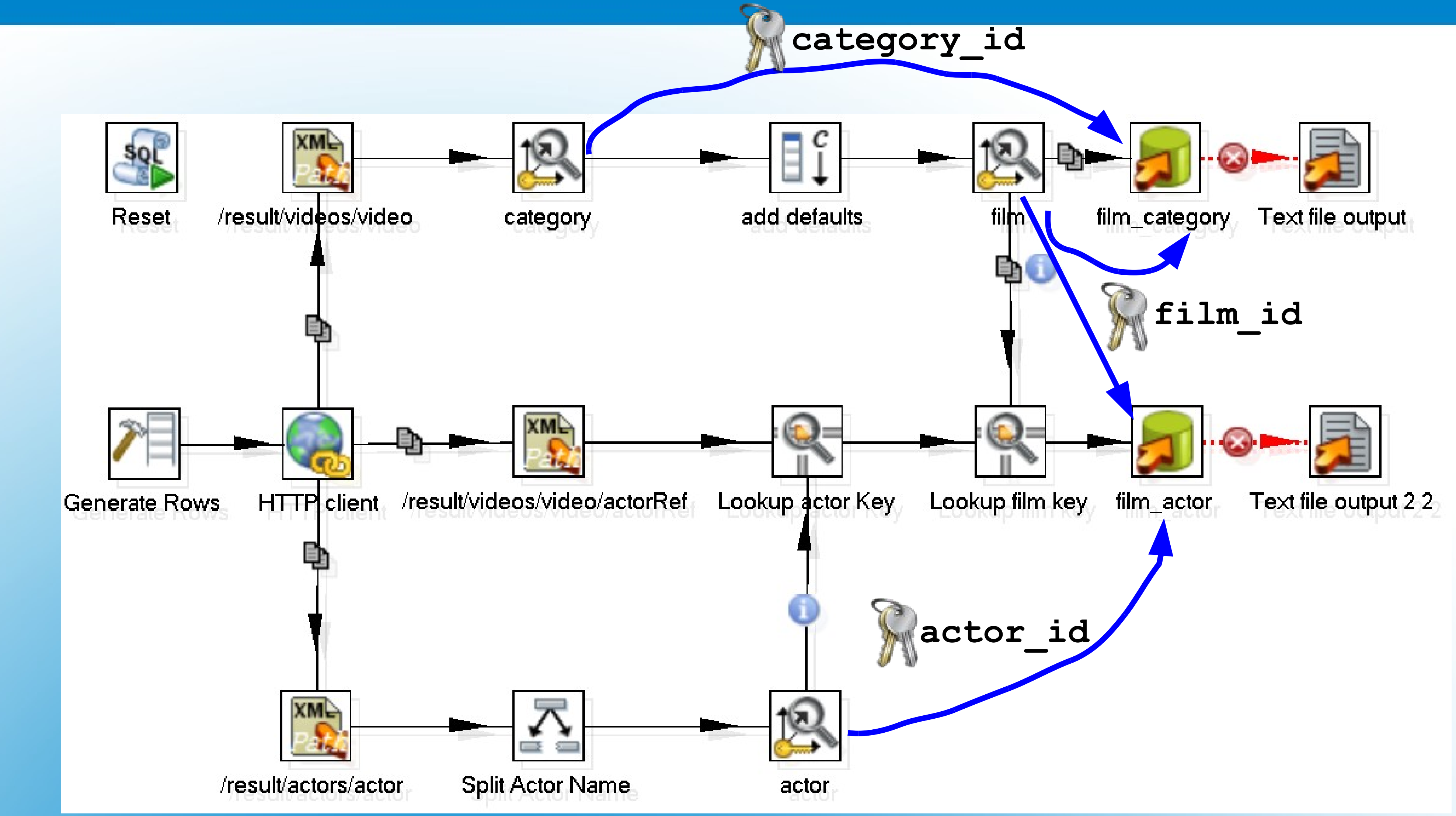


XML Import Example: `category` and `film_category`





XML Import Example: actor and film_actor



XML Import Example: Key Management

XML Examples: import `videos.xml` into db (1/4)

- Execute SQL to clean up database
  - Executes once in initialization phase
- Generate a row, pass url in field
  - We need this to drive the HTTP Client step
- HTTP Client `GET`s XML document
- Feed XML into Get Data From XML steps
  - Use XPath query to fetch rows
  - `<video>`, `<actor>`, `<actorRef>`
  - Use more XPath queries to get field values

- `<actor>` stream:
- Split actor name
  - `first_name`, `last_name`
- Lookup `actor_id` in `actor` table
  - INSERT new row in `actor` table if necessary
  - Add generated key value to stream
  - Output fed to `film_actor`

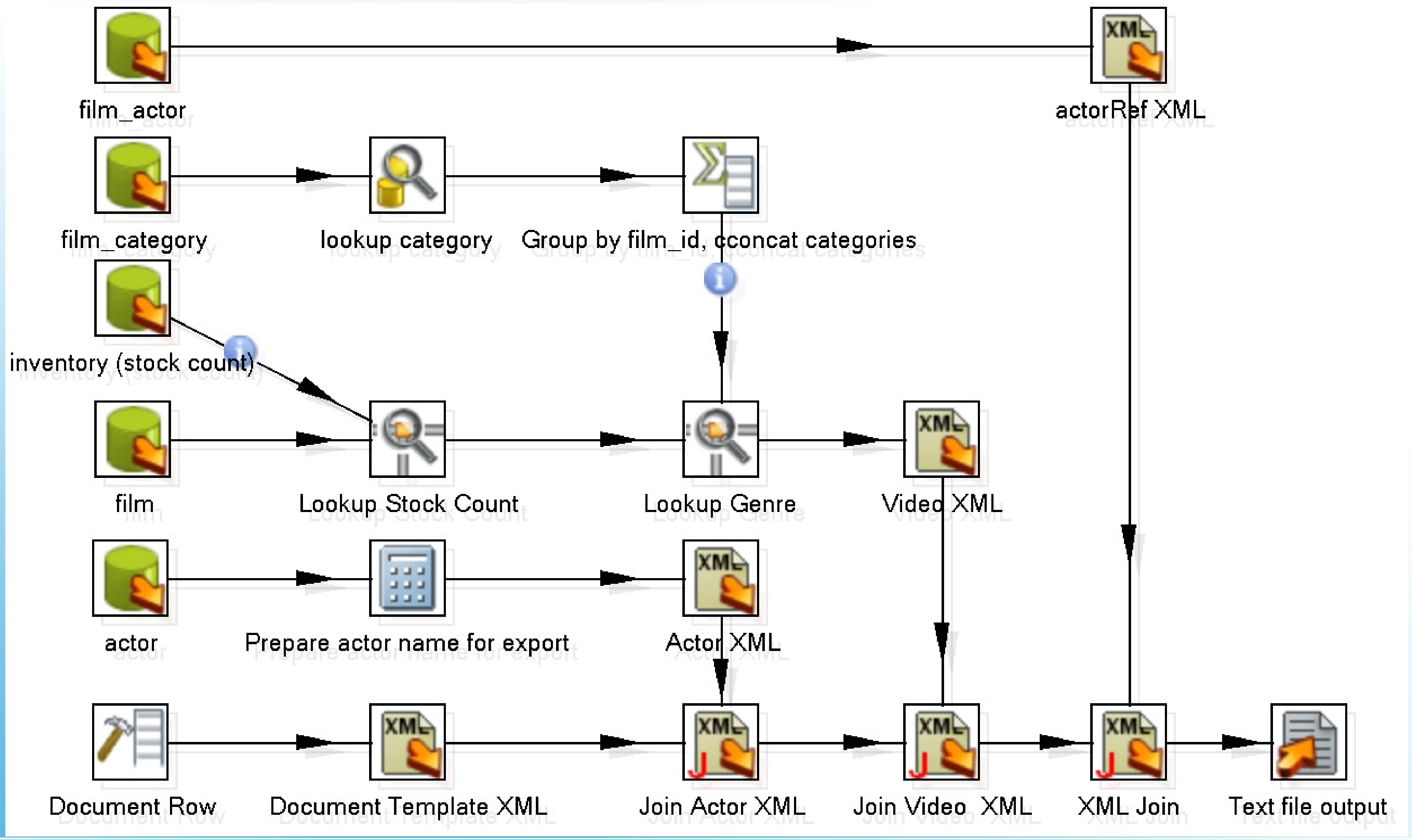
XML Examples: import `videos.xml` into db (2/4)

- `<video>` stream:
- Lookup `category_id` in `category` table
  - INSERT new row in `category` table if necessary
- Add defaults for `film` table
- INSERT new row in `film` table
  - Add generated key value to stream
  - Output fed to `film_actor` and `film_category`
- INSERT row to `film_category` table
  - Log rejected rows to text file

XML Examples: import `videos.xml` into db (3/4)

- `<actorRef>` stream:
- Lookup `actor_id` from `<actor>` stream
- Lookup `film_id` from `<video>` stream
- INSERT row to `film_actor` table
  - Log rejected rows to text file

XML Examples: import `videos.xml` into db (4/4)



XML Examples: export `videos.xml` from db (1/6)

- Generate 1 row for static elements:
  - actors, videos
- Generate XML document template:

```
<?xml version="1.0"?>
<result>
 <actors>
 </actors>
 <videos>
 </videos>
</result>
```

- XML Joins to merge sections into document:
  - Unqualified: `<actor>`, `<video>`
  - Qualified: `<actorRef>`

XML Examples: export `videos.xml` from db (2/6)

- `actor` stream:
  - Get rows from `actor` table
  - Get actor full name
    - `last_name + ', ' + first_name`
    - Initcap each word
    - `PENOLOPE GUINNES` → `'Guinness, Penolope'`
  - Generate XML `<actor>` elements
  - Merge into `<actors>` in the document
    - Simple join on XPath: `/result/actors`

XML Examples: export `videos.xml` from db (3/6)

- `film_category` Stream:
  - Get rows from `film_category` table
    - `ORDER BY film_id`
  - Lookup row from **`category`** table
  - Group multiple categories per `film_id`
    - Already ordered by `film_id` in SQL statement
  - Fed into film stream for lookup
    - Use `film_id` for match
    - Lookup categories (as `genre` field)

XML Examples: export **`videos.xml`** from db (4/6)

- **`film`** stream:
  - Get rows from `film` table
  - Look up stock count
  - Look up genre
  - Generate XML `<video>` elements
  - Merge into `<videos>` in the document
    - Simple join on XPath: `/result/films`

XML Examples: export `videos.xml` from db (5/6)

- `film_actor` stream:
  - Get rows from `film_actor` table
  - Generate XML `<actorRef>` elements
  - Merge into corresponding `<film>`
    - Complex join Xpath:
    - `/result/videos/video[@id=?]`
    - `?` placeholder is parameterized with `film_id`

XML Examples: export `videos.xml` from db (6/6)

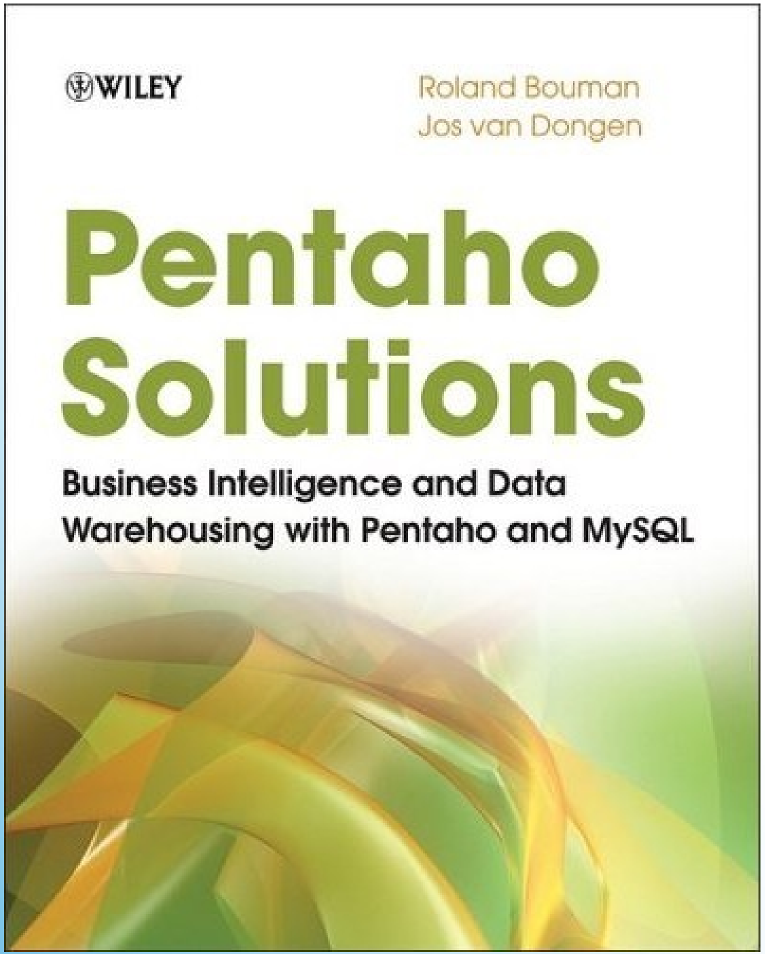
Clustering Kettle

- Slave Server
- Carte Service
  - Based on Jetty (Servlet Container)
  - Remote Execution of Jobs and Transformations
- Cluster
  - Group of slave servers

Clustering Kettle With Carte

Upcoming Book: Pentaho Solutions

- Pentaho Solutions
  - September 2009
  - 630+ page paperback, $50.00
  - OSCON Discount: 20%
- 3 Chapters on Kettle:
  - Getting Started (38 pages), Design (47 pages), Deployment (35 pages)
- ...and much more: Data Warehousing, Reporting, OLAP, Dashboarding, Data Mining


WILEY
Roland Bouman
Jos van Dongen
Pentaho Solutions
Business Intelligence and Data Warehousing with Pentaho and MySQL