pygrametl: A Powerful Programming Framework for Extract-Transform-Load Programmers

By Christian Thomsen and Torben Bach Pedersen

Sari Haj Hussein[1]

[1]Department of Computer Science
Aalborg University

2012-11-23

Concept

Extract-Transform-Load (ETL)

- Extract data from heterogeneous sources
- Transform this data
- Load the transformed data into a DW

Note

- Highly complex and time-consuming process

Concept

Extract-Transform-Load (ETL)

- Extract data from heterogeneous sources
- Transform this data
- Load the transformed data into a DW

Note

- Highly complex and time-consuming process

Contribution

pygrametl (py-gram-e-t-l)

- A Python-based programming framework for ETL programmers
- Perform ETL via writing code

Some Features

- Easy to insert data into dimensions and fact tables via one iteration through the source data
- Easy to insert data into snowflaked dimensions that span several underlying tables
- Easy to add new kinds of data sources

Contribution

pygrametl (py-gram-e-t-l)

- A Python-based programming framework for ETL programmers
- Perform ETL via writing code

Some Features

- Easy to insert data into dimensions and fact tables via one iteration through the source data
- Easy to insert data into snowflaked dimensions that span several underlying tables
- Easy to add new kinds of data sources

Rationale

- Challenge the GUI-based approach to performing ETL
  - It is difficult to express ETL solutions using the graphical editor components
  - It is faster to express the desired operations in some lines of code

Source Data

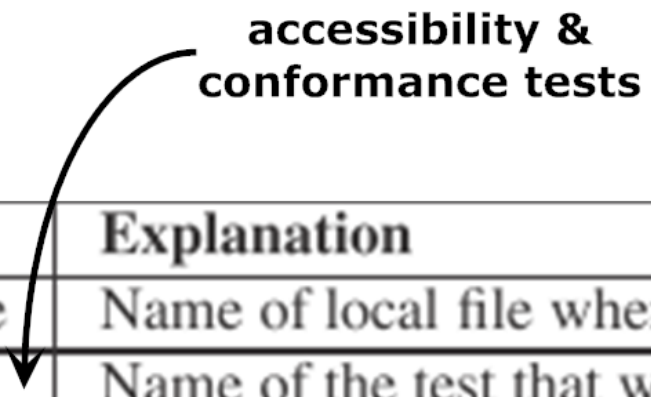
- A web crawler that downloads web pages from different web sites into local files

Source Data

Field	Explanation
localfile	Name of local file where the page was stored
url	URL from which the page was downloaded
server	HTTP header's Server field
size	Byte size of the page
downloaddate	When the page was downloaded
lastmoddate	When the page was modified

(a) DownloadLog.csv

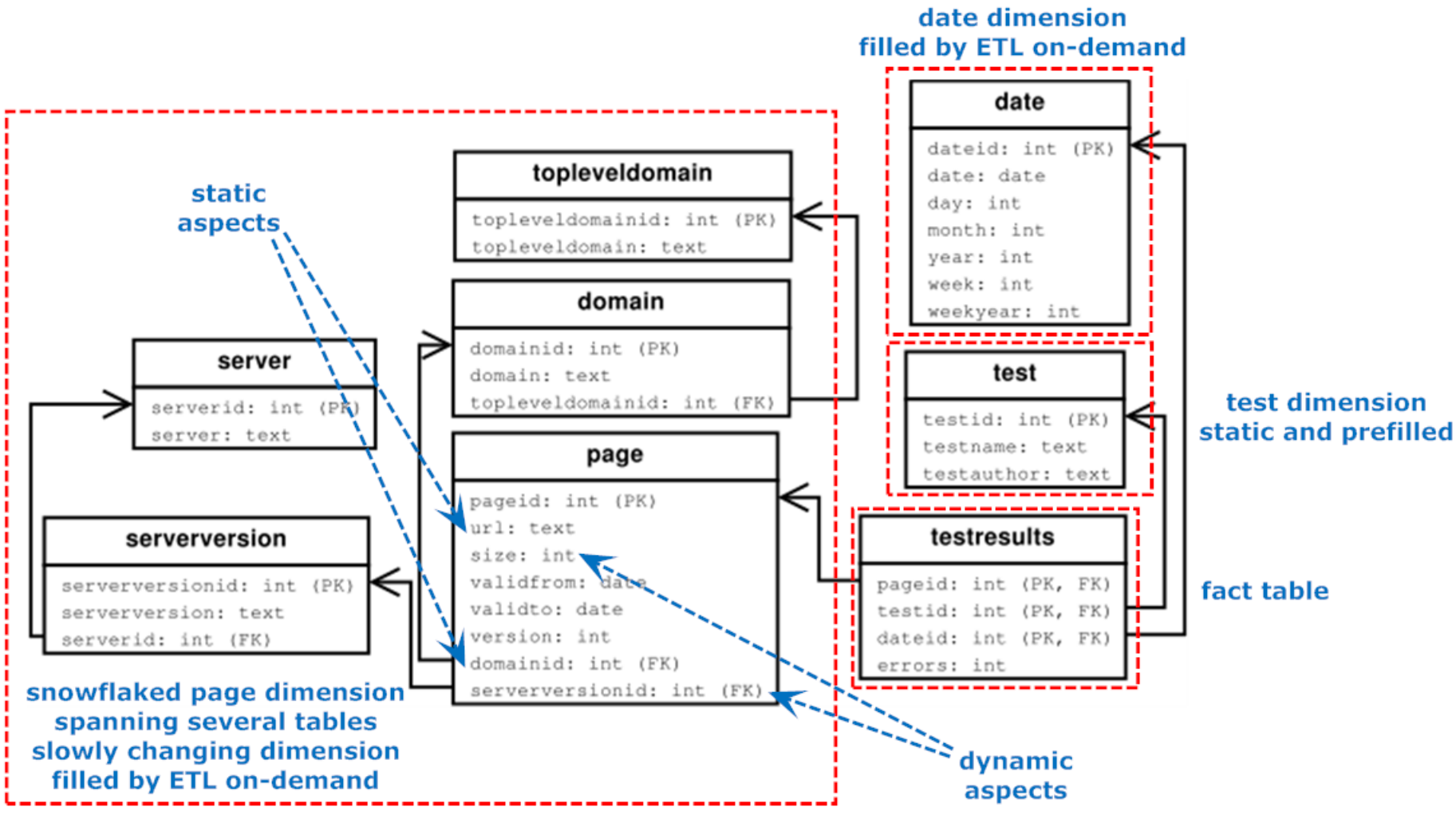
Source Data



Field	Explanation
localfile	Name of local file where the page was stored
test	Name of the test that was applied to the page
errors	Number of errors found by the test on the page

(b) TestResults.csv

DW Schema


date dimension
filled by ETL on-demand
date
dateid: int (PK)
date: date
day: int
month: int
year: int
week: int
weekyear: int
static
aspects
toplevledomain
topleveldomainid: int (PK)
topleveldomain: text
domain
domainid: int (PK)
domain: text
topleveldomainid: int (FK)
server
serverid: int (PK)
server: text
test
testid: int (PK)
testname: text
testauthor: text
test dimension
static and prefilled
page
pageid: int (PK)
url: text
size: int
validfrom: date
validto: date
version: int
domainid: int (FK)
serverversionid: int (FK)
serverversion
serverversionid: int (PK)
serverversion: text
serverid: int (FK)
testresults
pageid: int (PK, FK)
testid: int (PK, FK)
dateid: int (PK, FK)
errors: int
fact table
snowflaked page dimension
spanning several tables
slowly changing dimension
filled by ETL on-demand
dynamic
aspects

Overview

- Programmers create objects for each dimension and fact table in the DW
- Objects have convenient methods (insert, lookup, etc) that hide all details of caching, key assignment, SQL insertion
- These methods take rows (Python dictionaries) as arguments

Data Source Classes

SQLSource Class

- A data source that returns the rows of an SQL query

CSVSource Class

- A data source that returns the lines of a delimiter-separated file
- `testresults = CSVSource(file('TestResults.csv', 'r'), delimiter='\t')`

MergeJoiningSource Class

- A data source that equijoins rows from two other data sources
- `inputdata = MergeJoiningSource(testresults, 'localfile', downloadlog, 'localfile'))`

Data Source Classes

SQLSource Class

- A data source that returns the rows of an SQL query

CSVSource Class

- A data source that returns the lines of a delimiter-separated file
- `testresults = CSVSource(file('TestResults.csv', 'r'), delimiter='\t')`

MergeJoiningSource Class

- A data source that equijoins rows from two other data sources
- `inputdata = MergeJoiningSource(testresults, 'localfile', downloadlog, 'localfile'))`

Data Source Classes

SQLSource Class

- A data source that returns the rows of an SQL query

CSVSource Class

- A data source that returns the lines of a delimiter-separated file
- `testresults = CSVSource(file('TestResults.csv', 'r'), delimiter='\t')`

MergeJoiningSource Class

- A data source that equijoins rows from two other data sources
- `inputdata = MergeJoiningSource(testresults, 'localfile', downloadlog, 'localfile'))`

Dimension Class

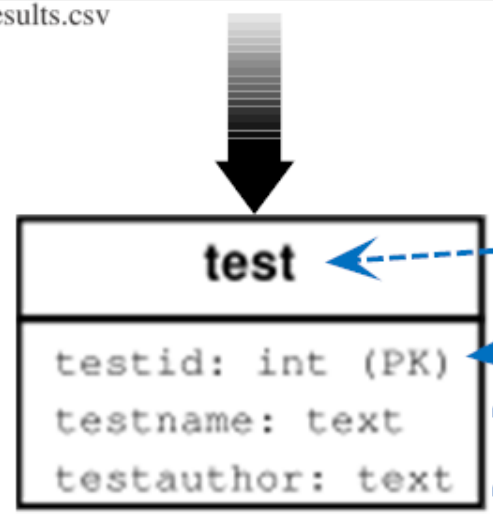
Dimension Class

- Used for dimensions that has exactly one table in the DW

Dimension Class

Field	Explanation
localfile	Name of local file where the page was stored
test	Name of the test that was applied to the page
errors	Number of errors found by the test on the page

(b) TestResults.csv



used when inserting a row that does not have a value for the dimension's key

```
testdim = Dimension(
    name='test',
    key='testid',
    defaultidvalue=-1,
    attributes=['testname', 'testauthor'],
    lookupatts=['testname'])
```

used to lookup the key value, e.g., lookup testid from inserted CSV test name

Dimension Class

Methods

- lookup attribute $\rightsquigarrow$ `lookup` $\rightsquigarrow$ key of the dimension member
- key value $\rightsquigarrow$ `getbykey` $\rightsquigarrow$ corresponding dimension member
- subset of a dimension member attributes $\rightsquigarrow$ `getbyvals` $\rightsquigarrow$ full, corresponding dimension member
- input row (key can be missing) $\rightsquigarrow$ `insert` $\rightsquigarrow$ row is added to the dimension (key is looked up when needed)
- input row $\rightsquigarrow$ `ensure` $\rightsquigarrow$ `lookup` key, and if it does not exist, `insert`

CachedDimension Class

CachedDimension Class

- It internally uses memory caching of dimension members to speed up lookups

SlowlyChangingDimension Class

SlowlyChangingDimension Class

- Supports type 1 and 2 changes in slowly changing dimensions

Reminder

- Type 1 change $\rightsquigarrow$ overwriting
- Type 2 change $\rightsquigarrow$ versioning (version number or validity date range)

SlowlyChangingDimension Class

SlowlyChangingDimension Class

- Supports type 1 and 2 changes in slowly changing dimensions

Reminder

- Type 1 change $\rightsquigarrow$ overwriting
- Type 2 change $\rightsquigarrow$ versioning (version number or validity date range)

SlowlyChangingDimension Class

page
pageid: int (PK)
url: text
size: int
validfrom: date
validto: date
version: int
domainid: int (FK)
serverversionid: int (FK)

```
pagedim = SlowlyChangingDimension(
name='page',
key='pageid',
attributes=['url', 'size', 'validfrom', 'validto',
'version', 'domainid' 'serverversionid'],
lookupatts=['url'],
fromatt='validfrom',
fromfinder=pygrametl.datereader('lastmoddate'),
toatt='validto',
versionatt='version')
```

SlowlyChangingDimension Class

Methods

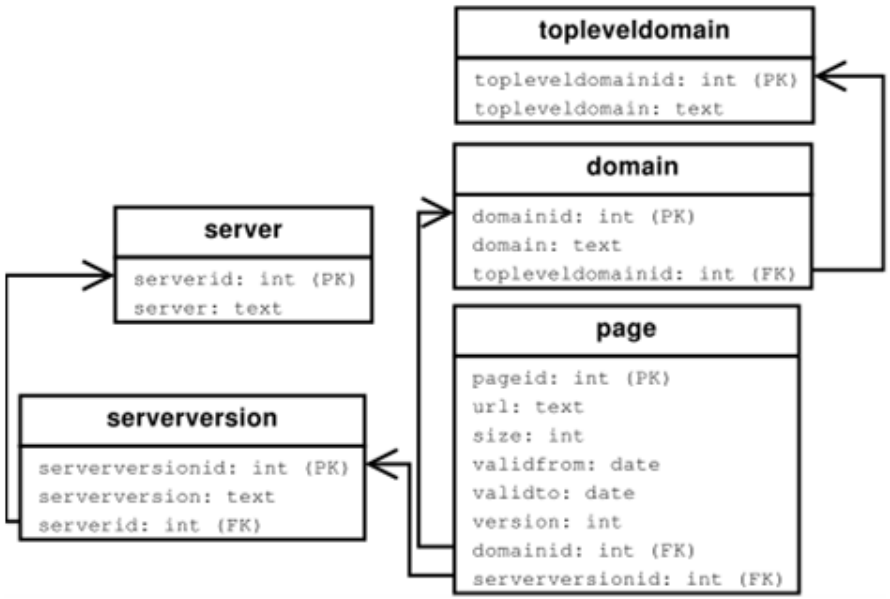
- lookup attribute $\rightsquigarrow$ `lookup` $\rightsquigarrow$ key of the newest version of the dimension member
- input row $\rightsquigarrow$ `scdensure` $\rightsquigarrow$ `ensure` & handle type 1 and 2 changes

SnowFlakedDimension Class

SnowFlakedDimension Class

- Supports filling a dimension in a snowflake schema through adding data to all underlying tables

SnowFlakedDimension Class




```
pagesf = SnowflakedDimension([
(pagedim, [serverversiondim, domaindim]),
(serverversiondim, serverdim),
(domaindim, topleveldim)
])
```

Fact Table Classes

FactTable Class

- Provides the basic representation of a fact table

BatchFactTable Class

- It does not insert rows immediately. Instead it waits until a user-configurable number of rows is available

BulkFactTable Class

- `insert` writes to a file. When a user-configurable number of rows has been added to the file, the file content is bulkloaded into the fact table

Fact Table Classes

FactTable Class

- Provides the basic representation of a fact table

BatchFactTable Class

- It does not insert rows immediately. Instead it waits until a user-configurable number of rows is available

BulkFactTable Class

- `insert` writes to a file. When a user-configurable number of rows has been added to the file, the file content is bulkloaded into the fact table

Fact Table Classes

FactTable Class

- Provides the basic representation of a fact table

BatchFactTable Class

- It does not insert rows immediately. Instead it waits until a user-configurable number of rows is available

BulkFactTable Class

- `insert` writes to a file. When a user-configurable number of rows has been added to the file, the file content is bulkloaded into the fact table

Flow Support

- Support steps and data flow between these steps
- The developer can create
  - A step for data extraction
  - A step for data cleansing
  - A step for data insertion into the DW

Step Classes

Step Class

- Provides the basic class for flow support
- `worker` $\rightsquigarrow$ applied on each row passing through the step
- `redirect` $\rightsquigarrow$ direct a row to a specific step
- `inject` $\rightsquigarrow$ inject a new row into the flow

DimensionStep Class

- `worker` calls `ensure` on a Dimension instance for each row passing through the step

MappingStep Class

- It applies a certain function on each row passing through the step (e.g., type conversion function)

Step Classes

Step Class

- Provides the basic class for flow support
- `worker` $\rightsquigarrow$ applied on each row passing through the step
- `redirect` $\rightsquigarrow$ direct a row to a specific step
- `inject` $\rightsquigarrow$ inject a new row into the flow

DimensionStep Class

- `worker` calls `ensure` on a Dimension instance for each row passing through the step

MappingStep Class

- It applies a certain function on each row passing through the step (e.g., type conversion function)

Step Classes

Step Class

- Provides the basic class for flow support
- `worker` $\rightsquigarrow$ applied on each row passing through the step
- `redirect` $\rightsquigarrow$ direct a row to a specific step
- `inject` $\rightsquigarrow$ inject a new row into the flow

DimensionStep Class

- `worker` calls `ensure` on a Dimension instance for each row passing through the step

MappingStep Class

- It applies a certain function on each row passing through the step (e.g., type conversion function)

Step Classes

ValueMappingStep Class

- It maps from one value to another, e.g., Sweden $\rightsquigarrow$ .se or Russia $\rightsquigarrow$ .ru

ConditionalStep Class

- It applies a given condition on each row, and if the condition evaluates to True, the row is passed on to the next step

AggregationStep Class

- It performs aggregation on rows, e.g., averaging

Step Classes

ValueMappingStep Class

- It maps from one value to another, e.g., Sweden $\rightsquigarrow$ .se or Russia $\rightsquigarrow$ .ru

ConditionalStep Class

- It applies a given condition on each row, and if the condition evaluates to True, the row is passed on to the next step

AggregationStep Class

- It performs aggregation on rows, e.g., averaging

Step Classes

ValueMappingStep Class

- It maps from one value to another, e.g., Sweden $\rightsquigarrow$ .se or Russia $\rightsquigarrow$ .ru

ConditionalStep Class

- It applies a given condition on each row, and if the condition evaluates to True, the row is passed on to the next step

AggregationStep Class

- It performs aggregation on rows, e.g., averaging

Test Bed

- They implemented an ETL program for the running example using
  - the code-based tool `pygrametl`
  - the graphical-based tool Pentaho Data Integration (PDI)
- Then they compared
  - the development time
  - the performance

Development Time

pygrametl

- Coding the ETL program took
  - less than one hour on the first use
  - 24 minutes on the second use

PDI

- Coding the ETL program took
  - more than 2 hours on the first use
  - 28 minutes on the second use

Development Time

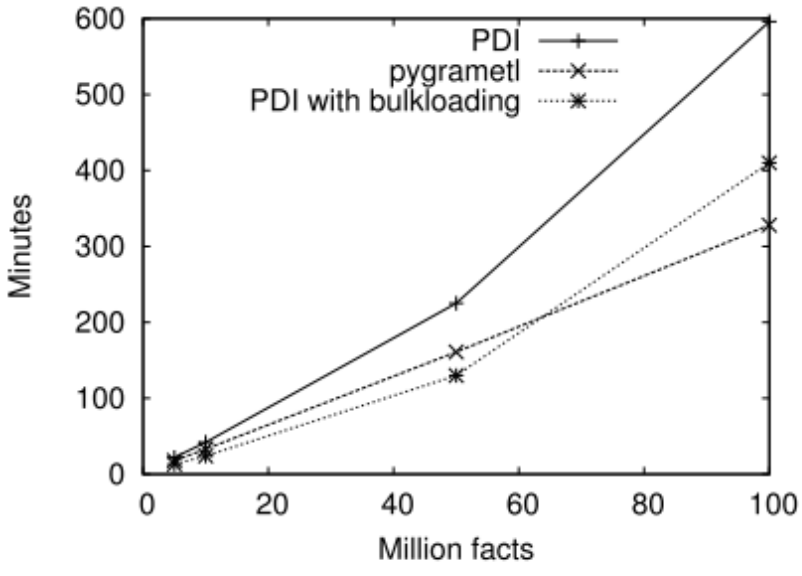
pygrametl

- Coding the ETL program took
  - less than one hour on the first use
  - 24 minutes on the second use

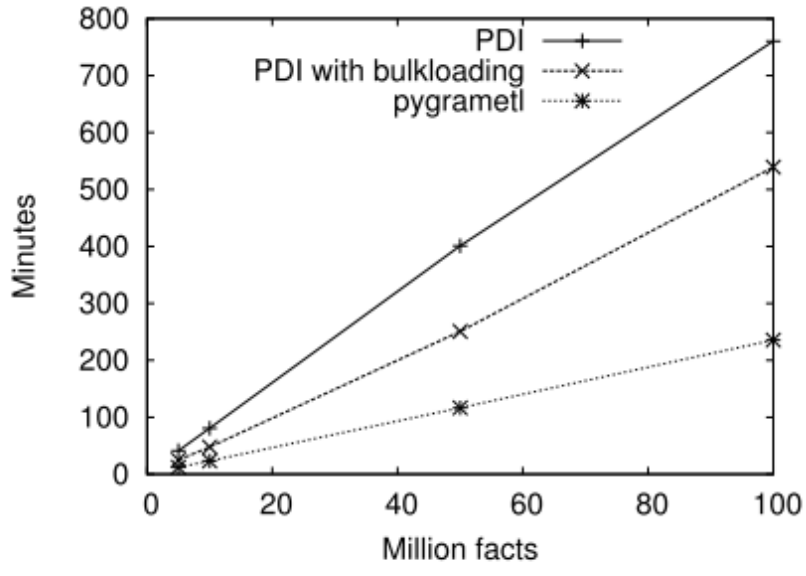
PDI

- Coding the ETL program took
  - more than 2 hours on the first use
  - 28 minutes on the second use

Performance



(a) Elapsed time



(b) CPU time

Thank You!