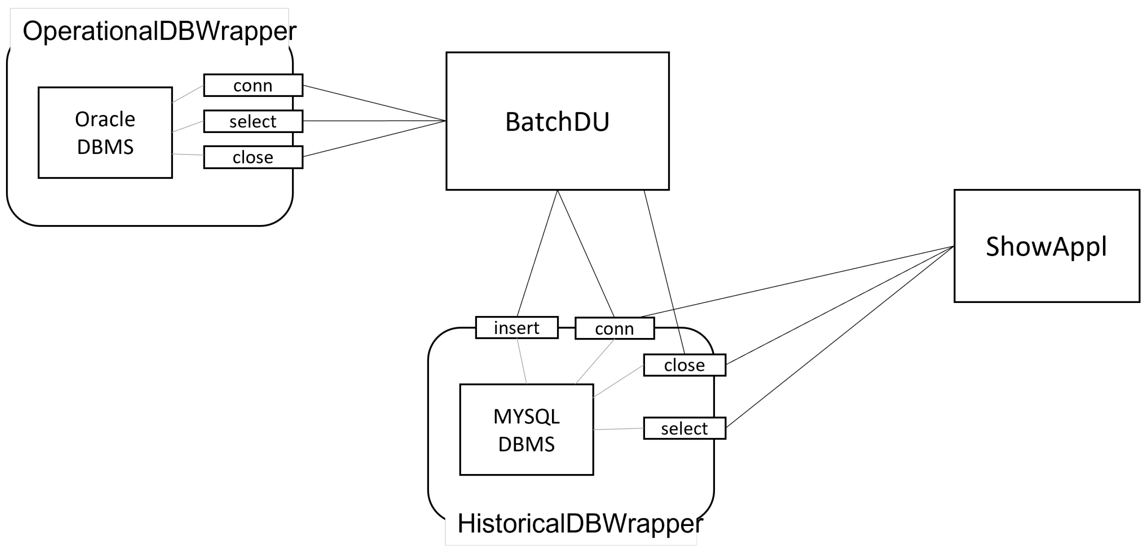
ASSIGNMENT 1

Author: Valerio Ghirimoldi

Architecture



The architecture is shown in the figure above, info about the classes are in the .py files.

Connections with the pillars

- **Framework for satisfying the requirements and a starting point for the implementation**.
  The architecure provides a way to read data stored in the operational database. The BatchDU component takes data related to a time interval from the operational database (when not under pressure, using the interface provided by the wrapper) and store them in the historical database.
  On the historical database it's possible to execute more specific queries (again, using the wrapper methods), for example projecting only specific columns, in order to show the data for example in a web application.

- **It avoids the lock-in problem**.
  Both the wrappers offer an interface on a DBMS, if the DBMS needs to be changed it's only a matter of changing the implementation of the related wrapper, the BatchDU application and other applications which may use the wrappers are not influenced by these modifications.
  (A strong assumption is that the DBMS are both relational).

- **It allows reusability**.
  The components are general and can be used in different context where a database is involved since they provide a way to interact with them in a controlled way.

- **It enforces security**.
  The wrappers provide only a few methods to read and/or write from/to the databases.
  It's not possible to directly execute arbitrary queries on them, only controlled queries are permitted.
  The wrappers also offer a method to log the operations executed (e.g. method invocations with their parameters) in order to analyze them in batch to search for suspicious activities and errors.

The BatchDU process extracts data from the operational database ordered by their timestamp, one row at a time (since getting all of them toghether may be unfeasible because of their size), and write them in the historical database.
If an error occurs during the execution of the process the state of both the databases remain consistent. To resume the process one can read the timestamp of the last row inserted in the historical database and restart the process with this timestamp as the starting one (e.g. the process wrote only half of the rows selected from the operational database before an error occured, I can re-execute it with $start\_ts =$ the timestamp of the last row inserted).
The classes are also compatible with a timestamp-based CDC pattern, which can be used periodically to select only the fresh data from the operational database and store them in the historical one (e.g. at the end of each month mm the BatchDU process is executed with start_ts = 01/mm/yyyy).