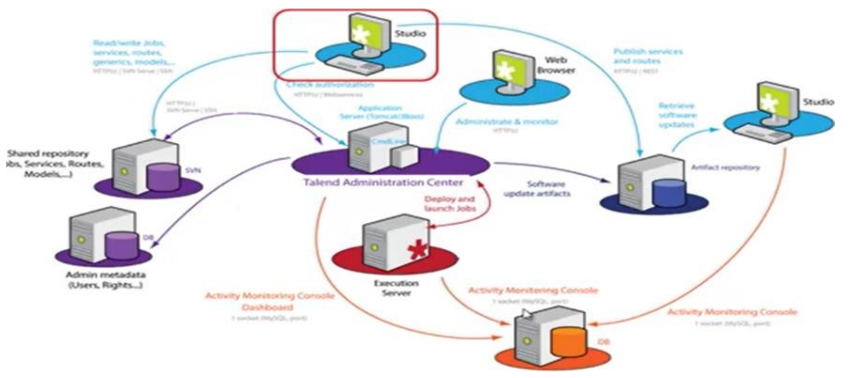
Fondement des Systèmes repartis

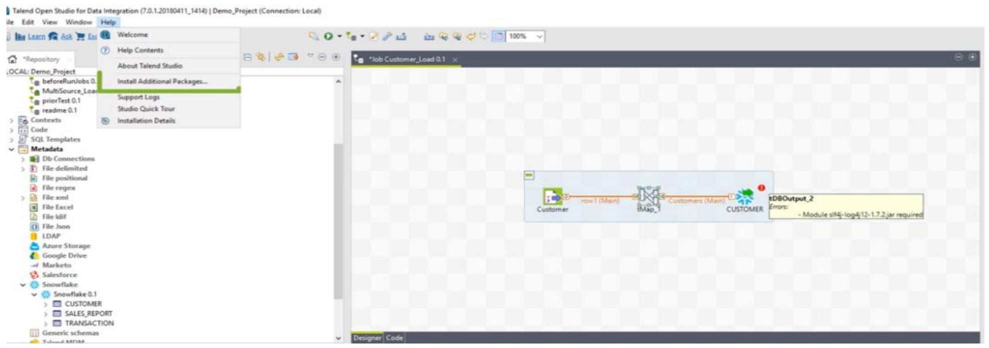
TP2 : Données distribuées : Synchronisation des bases de données avec RabbitMQ


Studio
Read/write Jobs, services, routes, generics, models...
Check authorization
Web Browser
Publish services and routes
Studio
Retrieve software updates
Application Server (Tomcat/JBoss)
Administrate & monitor
Shared repository Jobs, Services, Routes, Models...)
SVN
Talend Administration Center
Software update artifacts
Artifact repository
Deploy and launch Jobs
Admin metadata (Users, Rights...)
DB
Execution Server
Activity Monitoring Console Dashboard
Activity Monitoring Console
Activity Monitoring Console
DB

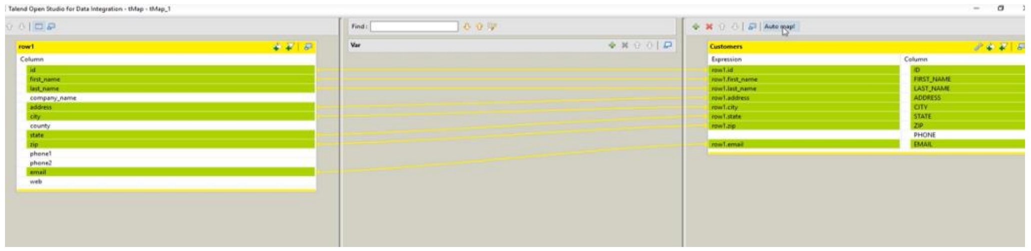
Talend : Open Studio for Data Integration Features Components

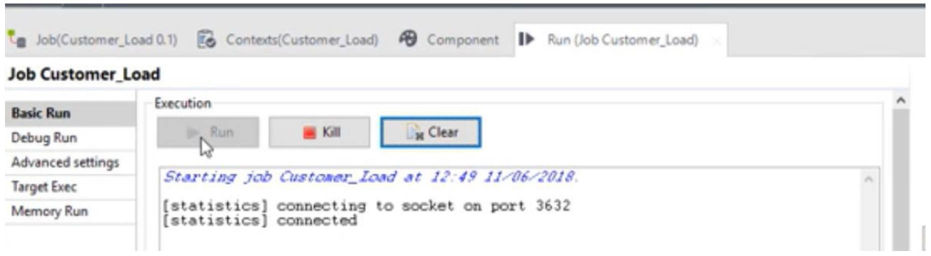
- Control and orchestrate data flows and data integrations with master jobs
- Map, aggregate, sort, enrich, and merge data
- Connectors : RDBMS: Oracle, Teradata, Microsoft SQL server, and more
- SaaS: Marketo, Salesforce, NetSuite, and more
- Packaged Apps: SAP, Microsoft Dynamics, Sugar CRM, and more
- Technologies: Dropbox, Box, SMTP, FTP/SFTP, LDAP, and more
- Design and Productivity Tools : Eclipse-based developer tooling and job designer, ETL and ELT support , Versioning, Export and execute standalone jobs in runtime environment

Data integration is the process of combining data from different sources into a unified view: starting from ingestion, to cleansing, mapping and transforming to a target sink, and finally making data more actionable and valuable to those accessing it.



https://info.talend.com/en_di_di_dummies.html?type=productspage





```
Starting job Customer_Load at 12:49 11/06/2018.

[statistics] connecting to socket on port 3632
[statistics] connected
```

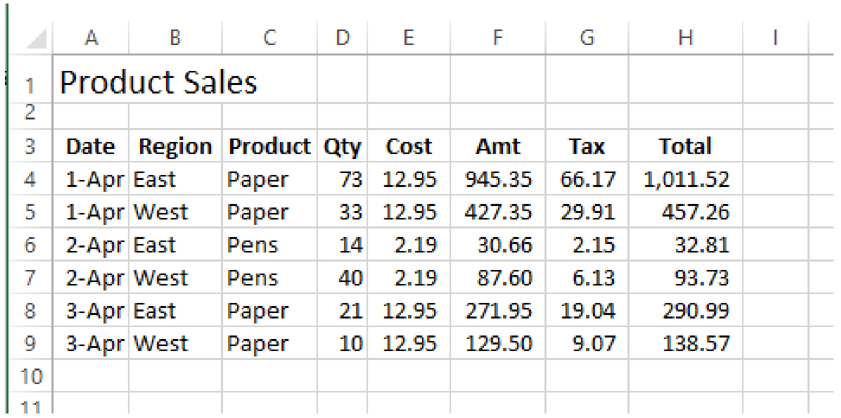
The Problem

Obviously, there are tons of different ways to synchronize distributed databases. Let's imagine that we have an unusual situation with restrictions below:

- A future system will have some Head Office (**HO**) and a couple of Branch Offices (**BO**s).
- All offices are located in different places, and some of them have limitation with the internet connection. It could even be a situation where the internet is available for 1-2 hours per day.

The perfect solution is to write your own DB sync mechanism data between branches using RabbitMQ Message queues.

For this lab we assume one Head Office (HO) and 2 Branch offices (BO) for sales. The 2 sales branches are physically separated from the Head office. They manage their databases independently and they need to synchronise their data to the Head office that maintain the hole data of sales. We assume that the database are based on the product sales table with the following structure.



	A	B	C	D	E	F	G	H	I
1	Product Sales								
2									
3	Date	Region	Product	Qty	Cost	Amt	Tax	Total	
4	1-Apr	East	Paper	73	12.95	945.35	66.17	1,011.52	
5	1-Apr	West	Paper	33	12.95	427.35	29.91	457.26	
6	2-Apr	East	Pens	14	2.19	30.66	2.15	32.81	
7	2-Apr	West	Pens	40	2.19	87.60	6.13	93.73	
8	3-Apr	East	Paper	21	12.95	271.95	19.04	290.99	
9	3-Apr	West	Paper	10	12.95	129.50	9.07	138.57	
10									
11									

The main of this lab is to create a distributed application that synchronisation databases from the product sales tables. This application needs to use the RabbitMQ to send data on the related queues. We run 2 distributed processes that synchronise data from first BO to HO and the second BO to the HO.

For this lab we use Java with JDBC connector for MySQL databases for sales. So, 3 databases are needed for the 2 BO and one HO. The synchronisation is made by the 2 BO.

Appendix: java codes

JdbcRetrieve.java

```java
package com.zetcode;

import java.sql.PreparedStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class JdbcRetrieve {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/testdb?useSSL=false";
        String user = "testuser";
        String password = "test623";

        String query = "SELECT * FROM Authors";

        try (Connection con = DriverManager.getConnection(url, user, password);
                PreparedStatement pst = con.prepareStatement(query);
                ResultSet rs = pst.executeQuery()) {

            while (rs.next()) {

                System.out.print(rs.getInt(1));
                System.out.print(": ");
                System.out.println(rs.getString(2));
            }

        } catch (SQLException ex) {

            Logger lgr = Logger.getLogger(JdbcRetrieve.class.getName());
            lgr.log(Level.SEVERE, ex.getMessage(), ex);
        }
    }
```

JdbcPreparedTesting.java

```java
package com.zetcode;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class JdbcPreparedTesting {

    public static void main(String[] args) {

        String cs = "jdbc:mysql://localhost:3306/testdb?useSSL=false";
        String user = "testuser";
        String password = "test623";

        String sql = "INSERT INTO Testing(Id) VALUES(?)";

        try (Connection con = DriverManager.getConnection(cs, user, password);
                PreparedStatement pst = con.prepareStatement(sql)) {

            for (int i = 1; i <= 5000; i++) {

                pst.setInt(1, i * 2);
                pst.executeUpdate();
            }

        } catch (SQLException ex) {

            Logger lgr = Logger.getLogger(JdbcPreparedTesting.class.getName());
            lgr.log(Level.SEVERE, ex.getMessage(), ex);
        }
    }
}
```

https://www.vogella.com/tutorials/MySQLJava/article.html

http://www.mysqltutorial.org/mysql-jdbc-tutorial/

http://zetcode.com/db/mysqljava/

http://www.mysqltutorial.org/mysql-jdbc-transaction/