← SQL beginner. Team01

Revie

**Submit the project**

Finish project

**Private Git project**

**Task**

# Team 01 - Piscine SQL

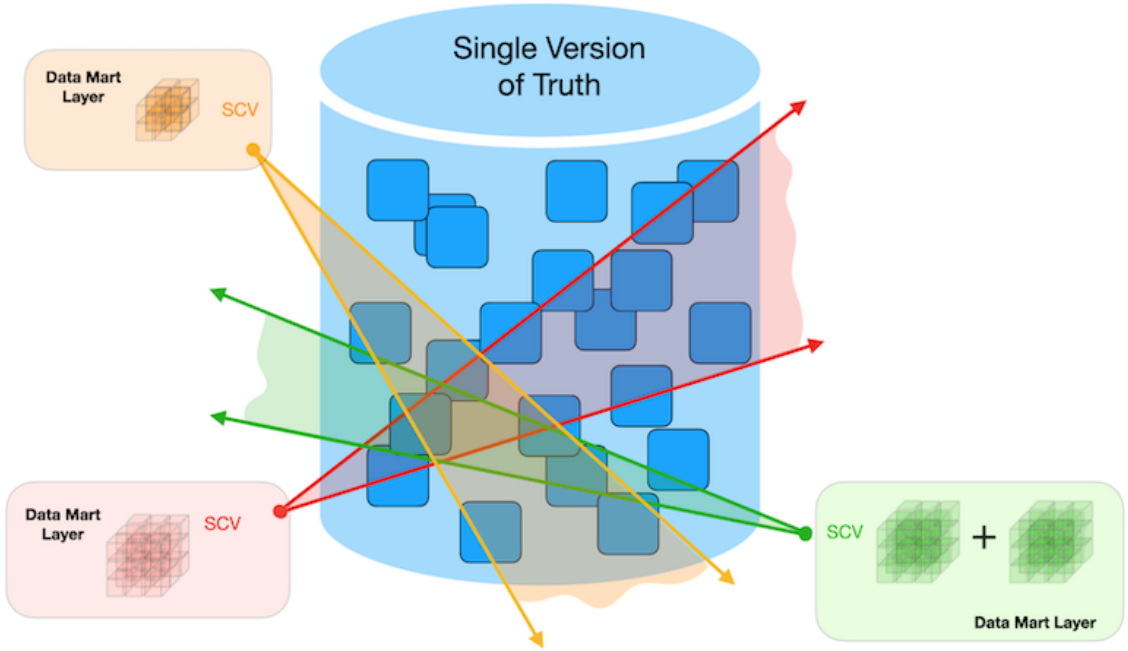
## *Data Warehouse*

Resume: Today you will know what DWH is and how to create a first ETL process

## Contents

# Chapter I

## Preamble



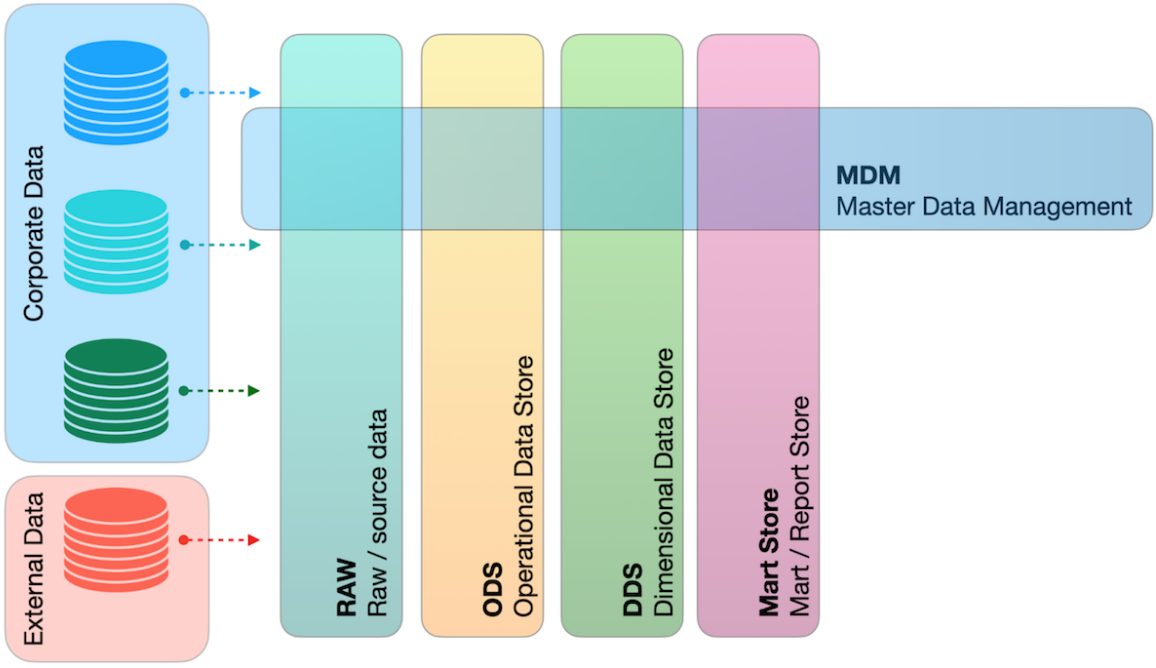

A Data Warehousing (DWH) is a process for collecting and managing data from varied sources to provide meaningful business insights. A Data warehouse is typically used to connect and analyze business data from heterogeneous sources. The data warehouse is the core of the BI system which is built for data analysis and reporting.

There are 2 DWH- “fathers” with opposite opinions on how to make a better DWH from logical data layers.

| | |
|---|---|

| | |
|---|---|
| “A DWH is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management’s decisions” (Bill Inmon) |  |
|  | “A DWH is a system that extracts, cleans, conforms, and delivers source data into a dimensional data store and then supports and implements querying and analysis for the purpose of decision making” (Ralph Kimball) |

Nowadays, Big Data is coming more and more and we need more resources to control, structure and further research our data. To support classical DataWareHouse systems there is a new pattern called LakeHouse (based on λ-architecture) = DataLake + DataWareHouse. From a logical point of view, we can imagine a modern LakeHouse like a set of logical data layers.




Therefore, to be a Data Architect you need to know a “bit more” than Relational Modeling. I would like to show a list of existing Data Models Patterns.

- Relational Model
- Temporal Model
- BiTemporal Model
- USS Model
- EAV Model
- Star / Snowflake Models
- Galaxy Model
- Data Vault Model
- Anchor Model
- Graph Model

## Chapter II

## General Rules

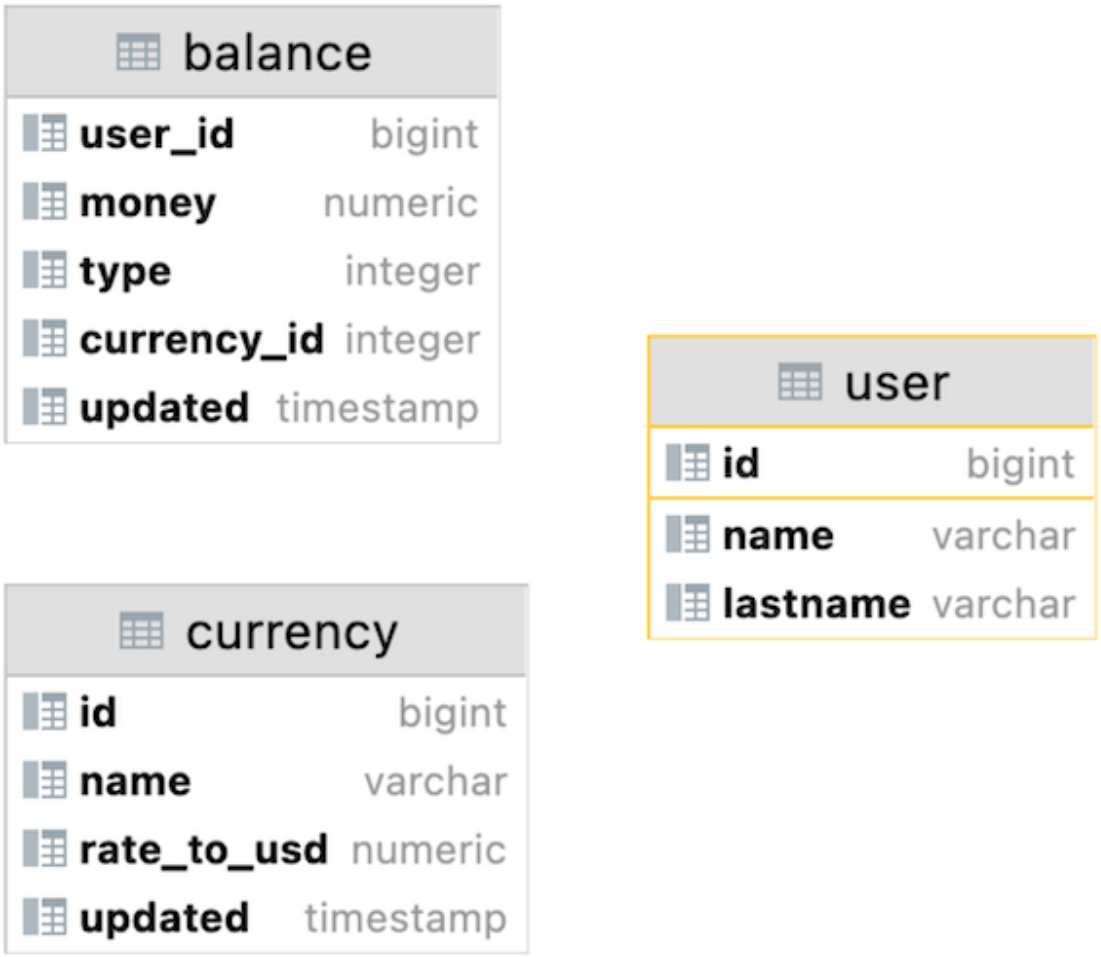
- Use this page as the only reference. Do not listen to any rumors and speculations on how to prepare your solution.
- Please make sure you are using the latest version of PostgreSQL.
- That is completely OK if you are using IDE to write a source code (aka SQL script).
- To be assessed your solution must be in your GIT repository.

- Your solutions will be evaluated by your piscine mates.
- You should not leave in your directory any other file than those explicitly specified by the exercise instructions. It is recommended that you modify your `.gitignore` to avoid accidents.
- Do you have a question? Ask your neighbor on the right. Otherwise, try with your neighbor on the left.
- Your reference manual: mates / Internet / Google.
- Read the examples carefully. They may require things that are not otherwise specified in the subject.
- And may the SQL-Force be with you!
- Absolutely everything can be presented in SQL! Let's start and have fun!

## Chapter III

## Rules of the day

- Please make sure you have an own database and access for it on your PostgreSQL cluster.
- All tasks contain a list of Allowed and Denied sections with listed database options, database types, SQL constructions etc. Please have a look at the section before you start.
- Please download a script with Database Model here and apply the script to your database (you can use command line with psql or just run it through any IDE, for example DataGrip from JetBrains or pgAdmin from PostgreSQL community).
- Please take a look at the Logical View of our Database Model.

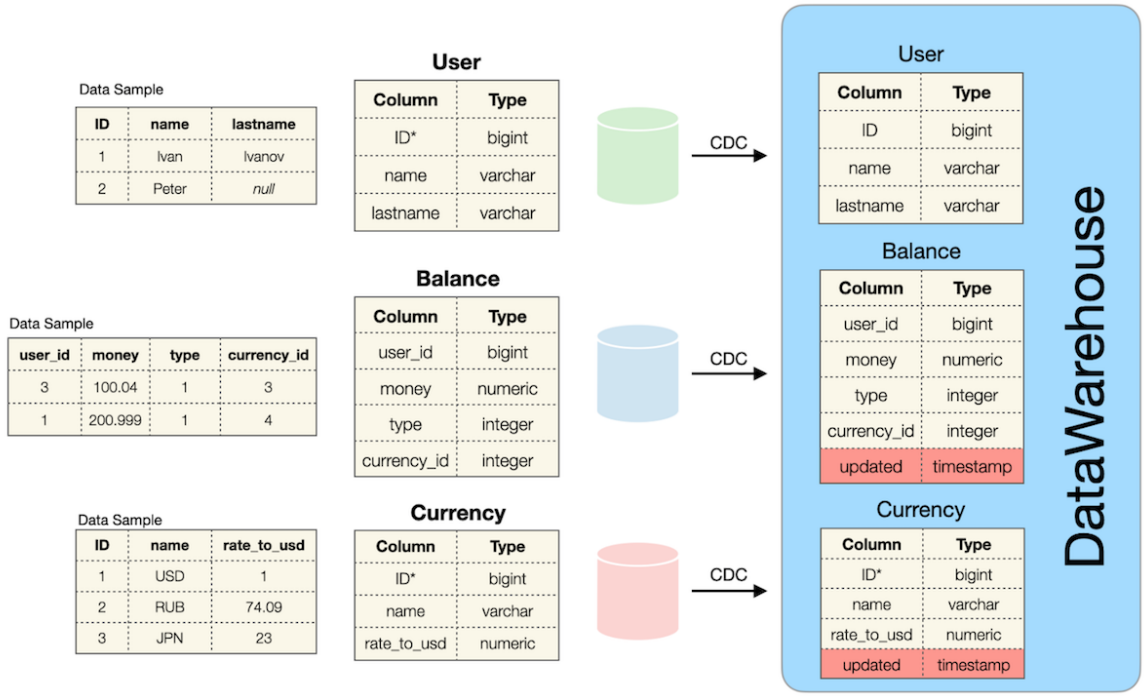



# Chapter IV

## Exercise 00 - Classical DWH

| Exercise 00: Classical DWH | |
|---|---|
| Turn-in directory | ex00 |
| Files to turn-in | `team01_ex00.sql` |
| **Allowed** | |
| Language | SQL |

Let’s take a look at the data sources and first logical data layer (ODS - Operational Data Store) in the DWH.




`User` table Definition (in a Green Source Database):

| Column Name | Description |
|---|---|
| ID | Primary Key |
| name | Name of User |
| lastname | Last name of User |

`Currency` table Definition (in a Red Source Database):

| Column Name | Description |
|---|---|
| ID | Primary Key |
| name | Currency Name |
| rate_to_usd | Ratio to USD currency |

`Balance` table Definition (in a Blue Source Database):

| Column Name | Description |
|---|---|
| user_id | “Virtual Foreign Key” to User table from other source |
| money | Amount of money |

| Column Name | Description |
|---|---|
| type | Type of balance (can be 0,1,...) |
| currency_id | “Virtual Foreign Key” to Currency table from other source |

Green, Red and Blue databases are independent data sources and satisfy the pattern of microservice. It means, there is a high risk of data anomalies (presented below).

- Tables are not in data consistency. It means there is a User, but does not have any rows in the Balance table, or vice versa, there is a Balance , but no rows in the User table. The same situation between Currency and Balance tables. (other words, doesn’t exist explicit Foreign Keys between them)
- Possible NULL values for name and lastname in User table
- All tables are working under OLTP (OnLine Transactional Processing) SQL traffic. It means there is an actual state of data at one time, historical changes are not saved for every table.

These 3 listed tables are data sources for the tables with the similar data models in the DWH area.

`User` table Definition (in a DWH Database):

| Column Name | Description |
|---|---|
| ID | Primary Key |
| name | Name of User |
| lastname | Last name of User |

`Currency` table Definition (in a DWH Database):

| Column Name | Description |
|---|---|
| ID | Mocked Primary Key |
| name | Currency Name |
| rate_to_usd | Ratio to USD currency |
| updated | Timestamp of event from source database |

`Mocked Primary Key` means there are duplicates with the same ID, because a new updated attribute was added that transforms our Relational Model to Temporal Relational Model.

Please take a look at the Data Sample for “EUR” currency below. This sample is based on SQL statement

```
SELECT *
FROM Currency
WHERE name = ‘EUR’
ORDER BY updated DESC;
```

| ID | name | rate_to_usd | updated |
|---|---|---|---|
| 100 | EUR | 0.9 | 03.03.2022 13:31 |
| 100 | EUR | 0.89 | 02.03.2022 12:31 |
| 100 | EUR | 0.87 | 02.03.2022 08:00 |
| 100 | EUR | 0.9 | 01.03.2022 15:36 |
| ... | ... | ... | ... |

`Balance` table Definition (in a DWH Database):

| Column Name | Description |
|---|---|
| user_id | “Virtual Foreign Key” to User table from other source |
| money | Amount of money |
| type | Type of balance (can be 0,1,...) |
| currency_id | “Virtual Foreign Key” to Currency table from other source |
| updated | Timestamp of event from source database |

Please take a look at the Data Sample below. This sample is based on SQL statement

```
SELECT *
FROM Balance
WHERE user_id = 103
ORDER BY type, updated DESC;
```

| user_id | money | type | currency_id | updated |
|---|---|---|---|---|
| 103 | 200 | 0 | 100 | 03.03.2022 12:31 |
| 103 | 150 | 0 | 100 | 02.03.2022 11:29 |
| 103 | 15 | 0 | 100 | 03.03.2022 08:00 |
| 103 | -100 | 1 | 102 | 01.03.2022 15:36 |
| 103 | 2000 | 1 | 102 | 12.12.2021 15:36 |
| ... | ... | ... | ... | ... |

All tables in DWH inherit all anomalies from source tables as well.

- Tables are not in data consistency.
- Possible NULL values for name and lastname in User table

Please write a SQL statement that returns the total volume (sum of all money) of transactions from user balance aggregated by user and balance type. Please be aware, all data should be processed including data with anomalies. Below presented a table of result columns and corresponding calculation formula.

| Output Column | Formula (pseudocode) |
|---|---|
| name | source: user.name if user.name is NULL then return '' value |
| lastname | source: user.lastname if user.lastname is NULL then return '' value |
| type | source: balance.type |
| volume | source: balance.money need to summarize all money "movements" |
| currency_name | source: currency.name if currency.name is NULL then return '' value |
| last_rate_to_usd | source: currency.rate_to_usd. take a last currency.rate_to_usd for corresponding currency if currency.rate_to_usd is NULL then return 1 |

| Output Column | Formula (pseudocode) |
|---|---|
| total_volume_in_usd | source: volume , last_rate_to_usd. make a multiplication between volume and last_rate_to_usd |

Please take a look at a sample of output data below. Sort the result by user name in descending mode and then by user lastname and balance type in ascending mode.

| name | lastname | type | volume | currency_name | last_rate_to_u |
|---|---|---|---|---|---|
| Петр | | 2 | 203 | | 1 |
| Иван | Иванов | 1 | 410 | EUR | 0.9 |
| ... | ... | ... | ... | ... | ... |

# Chapter V

## Exercise 01 - Detailed Query

| Exercise 01: Detailed Query | |
|---|---|
| Turn-in directory | ex01 |
| Files to turn-in | `team01_ex01.sql` |
| **Allowed** | |
| Language | ANSI SQL |

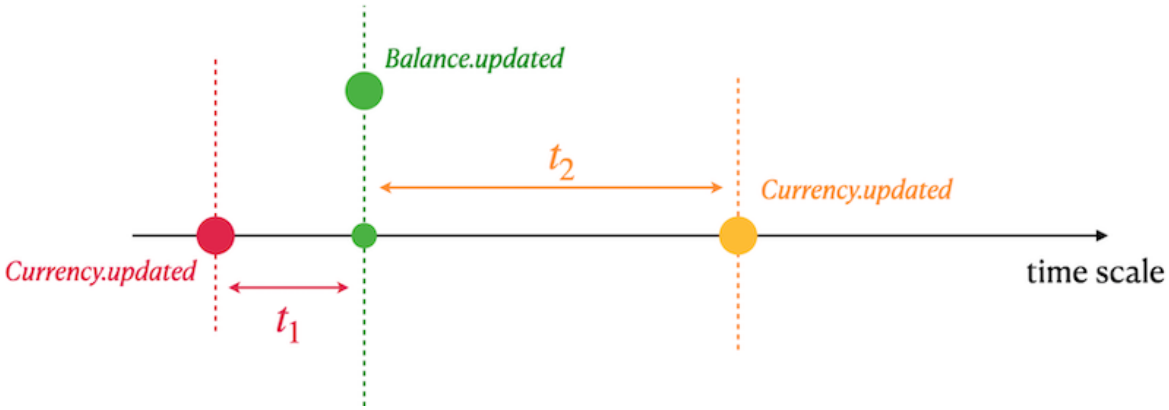
Before deeper diving into this task please apply INSERTs statements below.

```
insert into currency values (100, 'EUR', 0.85, '2022-01-01 13:29');
insert into currency values (100, 'EUR', 0.79, '2022-01-08 13:29');
```

Please write a SQL statement that returns all Users , all Balance transactions (in this task please ignore currencies that do not have a key in the `Currency` table ) with currency name and calculated value of currency in USD for the nearest day.

Below presented a table of result columns and corresponding calculation formula.

| Output Column | Formula (pseudocode) |
|---|---|
| name | source: user.name if user.name is NULL then return '' value |
| lastname | source: user.lastname if user.lastname is NULL then return '' value |
| currency_name | source: currency.name |
| currency_in_usd | involved sources: currency.rate_to_usd, currency.updated, balance.updated.Take a look at a graphical interpretation of the formula below. |




- need to find a nearest rate_to_usd of currency at the past (t1)
- if t1 is empty (means no any rates at the past) then find a nearest rate_to_usd of currency at the future (t2)
- use t1 OR t2 rate to calculate a currency in USD format

Please take a look at a sample of output data below. Sort the result by user name in descending mode and then by user lastname and currency name in ascending mode.

| name | lastname | currency_name | currency_in_usd |
|---|---|---|---|
| Иван | Иванов | EUR | 150.1 |
| Иван | Иванов | EUR | 17 |
| ... | ... | ... | ... |