Нов Български Университет

Бакалавърски факултет

Департамент “Информатика”

КУРСОВА РАБОТА

Курс:

CITB655 Data warehouse

Тема:

Изготвяне на проект на бизнес OLTP и OLAP система. Извличане на данните от OLTP системата, обработване с ETL процес и зареждането им в OLAP базата данни.

Изготвил:
Тихомир Младенов

Проверил:
доц. д-р Димитър Атанасов

Специалност:
Бизнес Информатика
Курс: Четвърти
Фак. Н: F-61480

НБУ София, юни 2019 г.

Съдържание

1. Представяне на бизнес модел и OLTP схема на фирмата

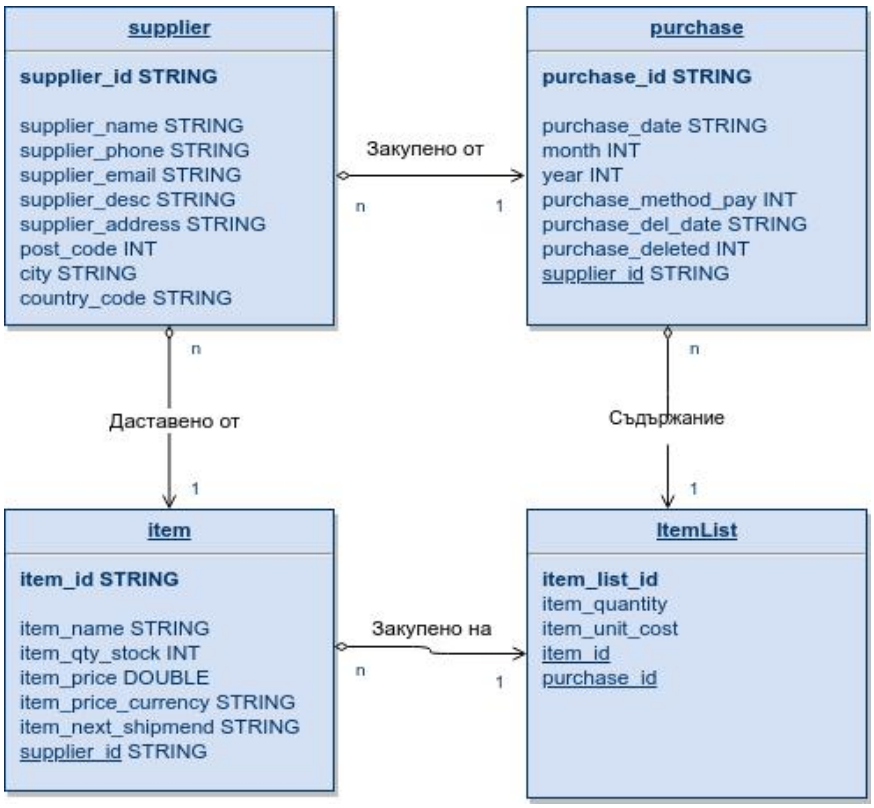
Ether Technologies е фирма занимаваща се с разработка на Internet of Things устройства. Тя работи с други фирми-партньори, от които си набавя нужните модули за устройствата, които произвежда. С цел обезпечаване на наличностите в складовата си база и свързаните с нея дейности, фирмата поддържа база-данни за “покупка-продажба”.

Фирмата пази информация за своите доставчици. Това са уникален ID, име, телефонен номер, имейл, мета описание, адрес, пощенски код, град, код държава.

Всяка покупка има уникален ID, дата на осъществяване, месец, година, метод на плащане (кеш 0, банков превод 1), дата на изтриване на покупката (0 ако не е изтрита). Всяка покупка има външен ключ към доставчик / доставчици, който/които ще я изпълнят/доставят.

Поръчаните артикули не винаги идват с една поръчка. В списъкът с елементи се съдържа ID номер на списъка, брой елементи, единична цена, ID номер на елемента, ID номер на покупката.

В таблица “артикул” с уникален ID се бележи всеки вид елемент, неговото име, налично количество, цена на брой, валута на цената, дата на следваща доставка и от кой доставчик.



1.1 Структура и съдържание на таблицата за доставчици

Supplier table									
supplier_id	supplier_name	supplier_phone	supplier_email	supplier_email	supplier_desc	supplier_address	post_code	city	country_code
S_1	ARM	555-123-4567	help@arm.com	help@arm.com	CPU	Peterhouse Technology Park, Cambridge, USA	45002	Cambridge	UK
S_2	Olimex	456-789-1011	order@olimex.com	order@olimex.com	PCB	Pravda St, 2, Plovdiv, 4000, Bulgaria	NULL	Plovdiv	BG
S_3	Samsung	123-456-9876	custserv@samsung.com	custserv@samsung.com	MEM	2 Fleet Pl, Farringdon, London, EC4A 4AD, UK	NULL	London	NULL
S_4	Intel	202-122-2324	sales@intel.com	sales@intel.com	NET	Pfizerstrasse 1, Karlsruhe, 76139, Germany	NULL	Karlsruhe	NULL

1.2 Структура и съдържание на съдържанието на таблицата за артикулите

Item table						
Item_id	Item_name	Item_q_stock	Item_price	Item_price_curr	Item_next_ship	Supplier_id
I_1	ARM-Cortex-A53	1200	3	USD	30-JUL-2019	S_1
I_2	1GB LPDDR2-900	600	1,5	USD	1-AUG-2019	S_3
I_3	500MB LPDDR2-900	600	1	USD	15-AUG-2019	S_3
I_4	SoC_PCB_SMT	900	2	USD	15-JUL-2019	S_2
I_5	8P8C_Ethern	1200	2	USD	30-JUL-2019	S_4

1.3 Структура и съдържание на съдържанието на таблицата за покупка

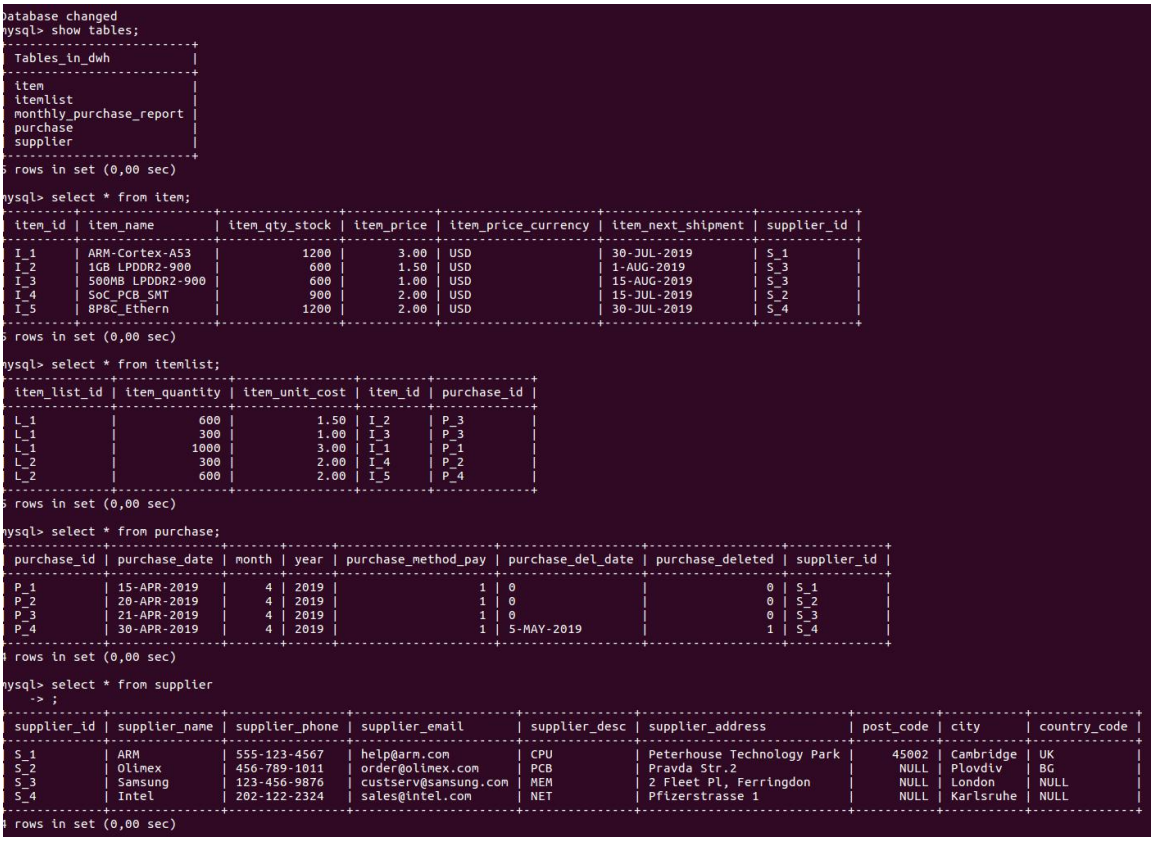
Purchase table							
purchase_id	purchase_date	Month	year	purchase_meth_pay	purchase_del_date	Purchase_deleted	supplier_id
P_1	15-APR-2019	4	2019	1	0	0	S_1
P_2	20-APR-2019	4	2019	1	0	0	S_2
P_3	21-APR-2019	4	2019	1	0	0	S_3
P_4	30-APR-2019	4	2019	1	5-MAY-2019	1	S_4

1.4 Структура и съдържание на съдържанието на таблицата със списък на артикулите

ItemList table				
item_list_id	item_quantity	item_unit_cost	item_id	purchase_id
L_1	600	1,5	I_2	P_3
L_1	300	1	I_3	P_3
L_1	1000	3	I_1	P_1
L_2	300	2	I_4	P_2
L_2	600	2	I_5	P_4

Компанията има своя ERP система и следят какви, колко и на каква стойност артикули имат в наличност и за кога са поръчани нови бройки. Не винаги се налага да се поръча само един елемент. В случая компанията е поставила една поръчка за I_2 (1GB RAM модули), I_3 (500MB RAM модули) от доставчика Самсунг. В друга пък се поръчвани само RISC процесори от ARM. В последната се очаква доставка на заготовка / платка за SMT монтаж, от Olimex. Доставката на мрежови ethernet модули от Intel е отказана и не я виждаме в таблицата горе.

Примерен изглед на транзакционната БД:



```
Database changed
mysql> show tables;
+-------------------------+
| Tables_in_dwh           |
+-------------------------+
| item                    |
| itemlist                |
| monthly_purchase_report |
| purchase                |
| supplier                |
+-------------------------+
5 rows in set (0,00 sec)

mysql> select * from item;
+---------+-----------------+---------------+------------+----------------------+---------------------+-------------+
| item_id | item_name       | item_qty_stock | item_price | item_price_currency | item_next_shipment | supplier_id |
+---------+-----------------+---------------+------------+----------------------+---------------------+-------------+
| I_1     | ARM-Cortex-A53  |          1200 |       3.00 | USD                  | 30-JUL-2019        | S_1         |
| I_2     | 1GB LPDDR2-900  |           600 |       1.50 | USD                  | 1-AUG-2019         | S_3         |
| I_3     | 500MB LPDDR2-900|           600 |       1.00 | USD                  | 15-AUG-2019        | S_3         |
| I_4     | SoC_PCB_SMT     |           900 |       2.00 | USD                  | 15-JUL-2019        | S_2         |
| I_5     | 8P8C_Ethern     |          1200 |       2.00 | USD                  | 30-JUL-2019        | S_4         |
+---------+-----------------+---------------+------------+----------------------+---------------------+-------------+
5 rows in set (0,00 sec)

mysql> select * from itemlist;
+--------------+---------------+----------------+---------+-------------+
| item_list_id | item_quantity | item_unit_cost | item_id | purchase_id |
+--------------+---------------+----------------+---------+-------------+
| L_1          |           600 |           1.50 | I_2     | P_3         |
| L_1          |           300 |           1.00 | I_3     | P_3         |
| L_1          |          1000 |           3.00 | I_1     | P_1         |
| L_2          |           300 |           2.00 | I_4     | P_2         |
| L_2          |           600 |           2.00 | I_5     | P_4         |
+--------------+---------------+----------------+---------+-------------+
5 rows in set (0,00 sec)

mysql> select * from purchase;
+-------------+---------------+-------+------+---------------------+-----------------+-----------------+-------------+
| purchase_id | purchase_date | month | year | purchase_method_pay | purchase_del_date | purchase_deleted | supplier_id |
+-------------+---------------+-------+------+---------------------+-----------------+-----------------+-------------+
| P_1         | 15-APR-2019   |     4 | 2019 |                   1 | 0               |               0 | S_1         |
| P_2         | 20-APR-2019   |     4 | 2019 |                   1 | 0               |               0 | S_2         |
| P_3         | 21-APR-2019   |     4 | 2019 |                   1 | 0               |               0 | S_3         |
| P_4         | 30-APR-2019   |     4 | 2019 |                   1 | 5-MAY-2019      |               1 | S_4         |
+-------------+---------------+-------+------+---------------------+-----------------+-----------------+-------------+
4 rows in set (0,00 sec)

mysql> select * from supplier
    -> ;
+-------------+---------------+----------------+-------------------+---------------+-------------------------+-----------+-----------+--------------+
| supplier_id | supplier_name | supplier_phone | supplier_email    | supplier_desc | supplier_address        | post_code | city      | country_code |
+-------------+---------------+----------------+-------------------+---------------+-------------------------+-----------+-----------+--------------+
| S_1         | ARM           | 555-123-4567   | help@arm.com      | CPU           | Peterhouse Technology Park |     45002 | Cambridge | UK           |
| S_2         | Olimex        | 456-789-1011   | order@olimex.com  | PCB           | Pravda Str.2            |      NULL | Plovdiv   | BG           |
| S_3         | Samsung       | 123-456-9876   | custserv@samsung.com | MEM        | 2 Fleet Pl, Ferringdon  |      NULL | London    | NULL         |
| S_4         | Intel         | 202-122-2324   | sales@intel.com   | NET           | Pfizerstrasse 1         |      NULL | Karlsruhe | NULL         |
+-------------+---------------+----------------+-------------------+---------------+-------------------------+-----------+-----------+--------------+
4 rows in set (0,00 sec)
```

Всеки месец, компания изтегля информация от OLTP базата си за наличните закупените артикули, броя им, единична цена, кога и от кой са закупени. Преобразува я и я вкарва в OLAP базата си чрез изпълняване на следната заявка:

```
SELECT item.item_id             AS item_id,
       item.item_name           AS item_name,
       itemlist.item_quantity   AS item_quantity,
       itemlist.item_unit_cost AS item_unit_cost,
       purchase.purchase_date,
       supplier.supplier_name
FROM   item
       LEFT JOIN itemlist
              ON item.item_id = itemlist.item_id
       INNER JOIN purchase
               ON purchase.purchase_id = itemlist.purchase_id
       INNER JOIN supplier
               ON purchase.supplier_id = supplier.supplier_id
WHERE  purchase.purchase_deleted = 0;
ORDER  BY purchase.purchase_date;
```

Резултат от изпълнението на горната заявка към OLTP базата:

item_id	item_name	item_quantity	item_unit_cost	purchase_date	supplier_name
I_1	ARM-Cortex-A53	1000	3.00	15-APR-2019	ARM
I_4	SoC_PCB_SMT	300	2.00	20-APR-2019	Olimex
I_2	1GB LPDDR2-900	600	1.50	21-APR-2019	Samsung
I_3	500MB LPDDR2-900	300	1.00	21-APR-2019	Samsung

2. Цел на внедряването на DWH

DWH ще трябва да следи складовите наличности на елементите през за всеки месец през годините на принципа на snapshot, който ще се изпълнява всеки месец. Основните мерни единици ще са налично количество, единична стойност на всеки от елементите, обща стойност на всеки един елемент според бройката и тотал стойност на всички покупки.

След като се агрегира достатъчно информация, примерно за всяка четвъртина, с информацията ще може да се изчислява:
- средно налично количество на елемент за даден времеви период;
- начален и краен баланс за всеки времеви период, тотал или по елемент;
- следене на примяната в налични количества между последователни и паралални периоди;
- минимални и максимални складови количества във времеви период;
- добавена стойност на наличен елемент към общата стойност на складовите наличности;

2.1 План на DWH

Димeнсиите в DWH ще са Calendar, Item, Supplier, данните от които идват и се преработват от транзакционната БД, и факт таблица, която ще се казва Report, която първоначално приема преработени данни от транзакционната БД, но също така се ъпдейтва и с данни от DWH след всеки snapshot.

2.1.1 Димeнсия Supplier

Ще се състои от технически ключ за самата DWH таблица, версия на записа, дата-от, дата-до (тези трите се ползват при добавянето на променени вече съществуващи записи в димeнсията), ID, име, телефон, град, пощенски код, имейл и код държава на доставчика.

Димeнсията я създаваме със следната команда:

```
CREATE TABLE `Supplier`
  (
     `supplier_tk`     BIGINT(20) NOT NULL,
     `version`         INT(11) DEFAULT NULL,
     `date_from`       DATETIME DEFAULT NULL,
     `date_to`         DATETIME DEFAULT NULL,
     `supplier_id`     INT(11) DEFAULT NULL,
     `supplier_name`   VARCHAR(20) DEFAULT NULL,
     `supplier_phone`  VARCHAR(15) DEFAULT NULL,
     `city`            VARCHAR(15) DEFAULT NULL,
     `post_code`       INT(11) DEFAULT NULL,
     `supplier_email`  VARCHAR(20) DEFAULT NULL,
     `country_code`    VARCHAR(2) DEFAULT NULL,
     PRIMARY KEY (`supplier_tk`),
     KEY `idx_supplier_lookup` (`supplier_id`),
     KEY `idx_supplier_tk` (`supplier_tk`)
  );
```

2.1.2 Димeнсия Calendar

Ще се състои от calendar_tk, който е технически ключ за DHW базата, месец, година, номер артикул, версия, дата-от, дата-до. Последните три данни ще служат за отбелязване на добавяне на променени данни към димeнсията. Таблицата съдържа данните кога кой артикул е закупен.

Димeнсията я създаваме с командата:

```
CREATE TABLE `Calendar`
  (
     `calendar_tk` INT(11) NOT NULL auto_increment,
     `calendar_m`  INT(11) DEFAULT NULL,
     `calendar_y`  INT(11) DEFAULT NULL,
     `item_id`     INT(11) DEFAULT NULL,
     `version`     INT(11) DEFAULT NULL,
     `date_from`   DATETIME DEFAULT NULL,
     `date_to`     DATETIME DEFAULT NULL,
     PRIMARY KEY (`calendar_tk`),
     KEY `idx_calendar_lookup` (`item_id`)
  );
```

2.1.3 Димeнсия Item

Ще се съдържа item_tk - технически ключ за DWH, ID на артикул, име на артикул, версия, дата-от, дата-до. Идеята на тази димeнсия е при нужда да добавя повече семантика към Data Mart-а като се добавят имената на артикулите.

Създаваме димeнсията по следния начин:

```
CREATE TABLE `Item`
  (
     `item_tk`    INT(11) NOT NULL auto_increment,
     `item_id`    INT(11) DEFAULT NULL,
     `item_name`  VARCHAR(20) DEFAULT NULL,
     `version`    INT(11) DEFAULT NULL,
     `date_from`  DATETIME DEFAULT NULL,
     `date_to`    DATETIME DEFAULT NULL,
     PRIMARY KEY (`item_tk`),
     KEY `idx_item_lookup` (`item_id`)
  );
```

2.1.4 Факт таблица Report

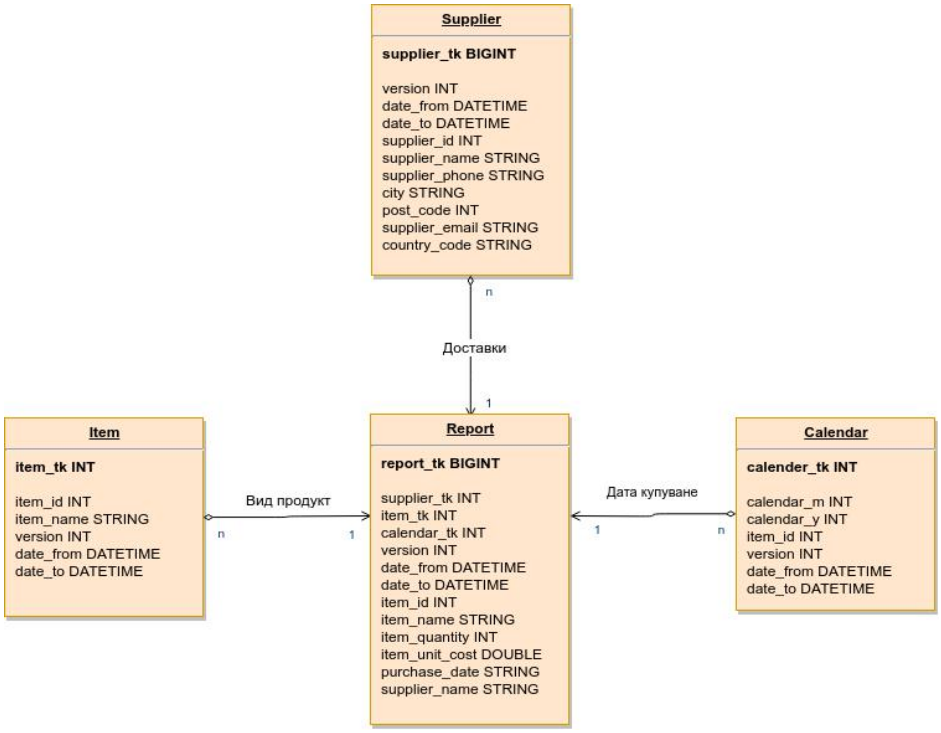
Мерителните данни, които идват от OLTP системата, ще се запишат във факт таблицата Report в OLAP системата, която събира данни от таблиците item, itemlist, supplier и purchase на OLTP базата данни:

```
CREATE TABLE `report`
  (
     `report_tk`       BIGINT(20) NOT NULL auto_increment,
     `supplier_tk`     INT(11) DEFAULT NULL,
     `item_tk`         INT(11) DEFAULT NULL,
     `calendar_tk`     INT(11) DEFAULT NULL,
     `version`         INT(11) DEFAULT NULL,
     `date_from`       DATETIME DEFAULT NULL,
     `date_to`         DATETIME DEFAULT NULL,
     `item_id`         INT(11) DEFAULT NULL,
     `item_name`       VARCHAR(20) DEFAULT NULL,
     `item_quantity`   INT(11) DEFAULT NULL,
     `item_unit_cost`  DOUBLE DEFAULT NULL,
     `purchase_date`   VARCHAR(20) DEFAULT NULL,
     `supplier_name`   VARCHAR(20) DEFAULT NULL,
     PRIMARY KEY (`report_tk`),
     KEY `idx_report_lookup` (`item_id`),
     KEY `idx_report_tk` (`report_tk`)
  );
```

В тази факт таблица / димeнсия, се събират техническите ключове от всеки запис в OLAP базата данни, версиите на записите, кога за променяни, ID, име, количество, цена единична, дата купуване и доставчик на артикул.

2.2 OLAP Схема

Нагледно, DWH базата данни ще бъде базирана на схема звезда. Ето как изглежда тя нагледо с горепосочените димeнсии / таблици:

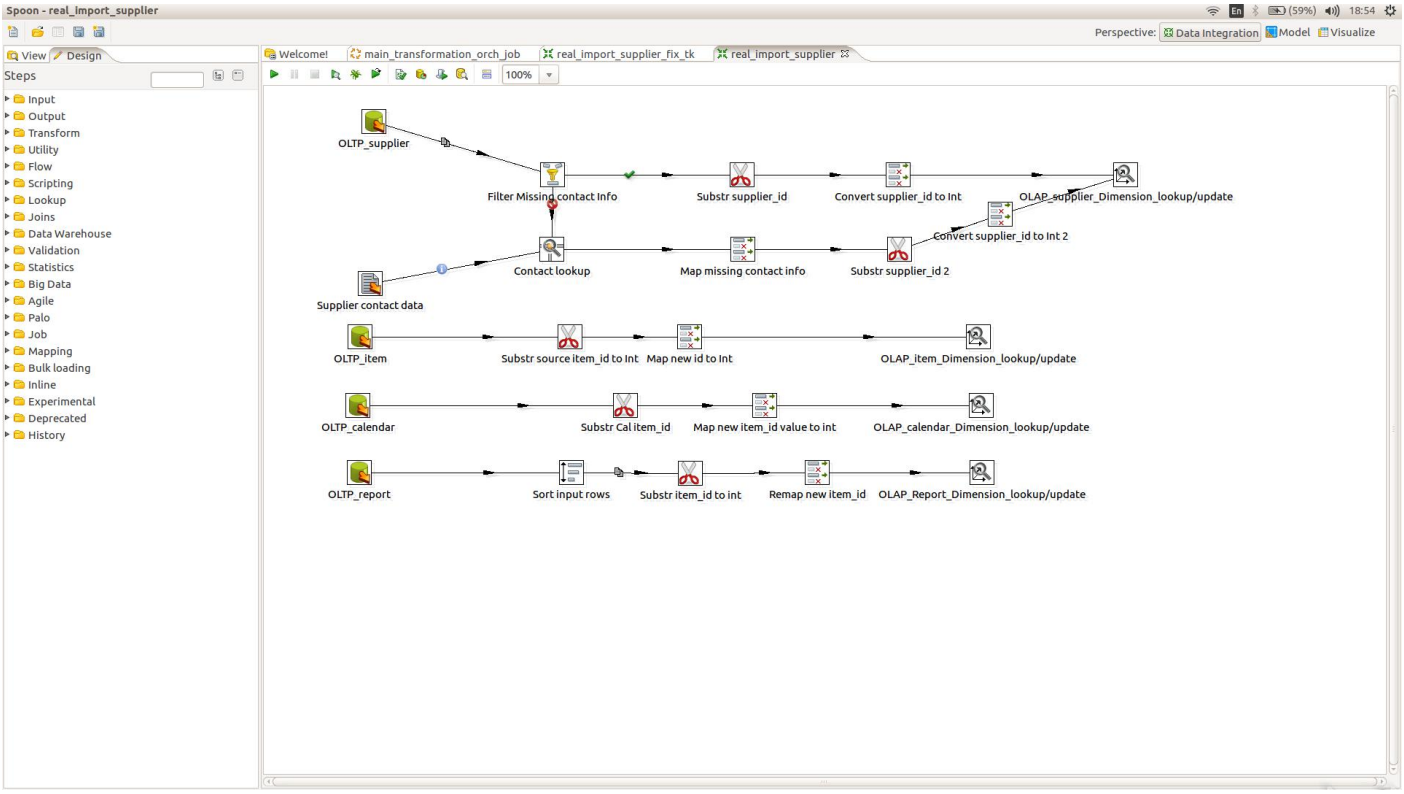


3. Extraction, Transform, Load процес

За да демонстрирам работещ ETL модел, реших да използвам community версията на Pentaho Data Integration (PDI). Той е част от семейството BI софтуер собственост на Hitachi Data Systems. Като цяло може да се каже, че интерфейсът му е структуриран с модули за извличане на информация от различни източници като *.csv файлове или бази данни, проверка и преработване на данни в различни типове с готови модули или regex функции, записване на данните във изходящи *.csv файлове или тяхното качване в други бази данни подготвени за DWH или Data mart. Софтуерът представя метод на преглеждане на Fact таблицата чрез Model и Visualize plug-in, който се казва Agile BI (Pentaho Agile Business Intelligence (Agile BI)). Така не сме задължени да използваме BI софтуер за визуализирането на информацията в таблици или кубове (кубовете също са като таблица, но можем да си представим, че по Z оста са разположени последователно данните по дати или години), които можем

да филтрираме в Agile BI, за да направим нужното изчисление и неговото визуализиране в графика.

В PDI се създават “jobs” и “transformations”. В трансформациите (transformations) се задават стъпки за четене и изваждане, преработване и записване на информацията. При промяна на схемата на данните в dwh / data mart-а можем много лесно да преконфигурираме ETL процеса, който изглежда така:



Ще обясня работния плот на PDI. Вляво в таб View са задачите (jobs) за трансформации, връзки към БД и стъпки. В таб Design, който е на снимката горе, се виждат типовете средства, с които PDI (още наричан Spoon и Kettle), разполага. С drag-and-drop ги разполагаме на централния плот според нуждите ни и конфигурираме стъпките. Отляво са входящите данни, в центъра е преработката, вдясно е изход/записване на данните. За да не стане тази курсова работа твърде дълга, ще спестя обяснението как се конфигурира всяка стъпка, и само ще обясня какво прави.

3.1 ETL на таблица supplier в OLTP

Взимаме данните от таблицата в OLTP, със следната заявка:

```
SELECT supplier_id, supplier_name, supplier_phone, city, post_code,
supplier_email, country_code
FROM supplier;
```

При въвеждането, оперативният персонал не е въвел повечето данни за пощенски код и код държава на всеки производител. За да се поправи това, четем данните от *.csv файл (“Supplier contact data”), където се намира информацията за всяка държава, всеки град пощенския му код. На стъпка “Filter Missing contact Info” (който е реално if (city != NULL && post_code != NULL && country_code != NULL)), редовете, в които която и да е колона от трите удебелени е празна, се препращат в стъпка “Contact lookup”, където по зададен критерий, по град, си набавяме липсващата информация и на стъпка “Map missing contact info”, заменяме непълните редове с новите, които са с пълната информация за пощенски код и код на държава (BG, UK, etc). Тези стъпки се прескачат, ако при проверката всичко е налично на реда, и информацията се изпраща към стъпката “Substr supplier_id”, която взима ID-то на доставчика от OLTP, формат S_* (низ), изрязва цифрата, и на стъпка “Convert supplier_id to int” я обръща в целочислен тип. Накрая информацията се записва в базата данни чрез стъпка “OLAP_supplier_Dimenstion_lookup/update”. Там съм задал критерий PDI да сравнява дали “push”-ваната информация е нова, и ако да, я праща към DWH DB, ако не - я пропуска. Т.е. Не ъпдейтваме вече въведени редове.

Почти същото правим и за OLTP таблиците item и calendar, където проверка за пълност на информацията не се прави, понеже по сценарий всичко задължително трябва да е попълнено в “input” полетата на front-end-а. За тях взимаме информацията със следните заявки:

- За Item:

```
SELECT item.item_id, item.item_name
FROM item
INNER JOIN itemlist ON item.item_id = itemlist.item_id
INNER JOIN purchase ON purchase.purchase_id = itemlist.purchase_id
WHERE purchase_del_date=0;
```

- За Calendar:

```
SELECT purchase.month AS month, purchase.year AS year, itemlist.item_id
AS item_id
FROM purchase
LEFT JOIN itemlist
ON purchase.purchase_id=itemlist.purchase_id
WHERE purchase_deleted=0;
```

Данните за факт таблицата Report в DWH базата ги взимаме от OLTP, сортираме ги по възходящ ред, преобразуваме ID на артикула в INT, и записваме в DWH DB. Данните извличаме от OLTP със заявката:

```
SELECT
item.item_id AS item_id, item.item_name AS item_name,
itemlist.item_quantity AS item_quantity,
itemlist.item_unit_cost AS item_unit_cost, purchase.purchase_date,
supplier.supplier_name
FROM item
LEFT JOIN itemlist ON item.item_id = itemlist.item_id
INNER JOIN purchase
ON purchase.purchase_id = itemlist.purchase_id
INNER JOIN supplier
ON purchase.supplier_id = supplier.supplier_id
WHERE purchase.purchase_deleted=0
ORDER BY purchase.purchase_date;
```

След преработката, данните в Report изглеждат ето така в DWH DB:

```
mysql> SELECT * FROM Report;
+-----------+-------------+---------+--------------+---------+---------------------+---------------------+---------+-----------------+---------------+----------------+---------------+---------------+
| report_tk | supplier_tk | item_tk | calendar_tk | version | date_from           | date_to             | item_id | item_name       | item_quantity | item_unit_cost | purchase_date | suppliер_name |
+-----------+-------------+---------+--------------+---------+---------------------+---------------------+---------+-----------------+---------------+----------------+---------------+---------------+
|         1 |        NULL |    NULL |         NULL |       1 | NULL                | NULL                |    NULL | NULL            |          NULL |           NULL | NULL          | NULL          |
|         2 |        NULL |    NULL |         NULL |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       1 | ARM-Cortex-A53  |          1000 |              3 | 15-APR-2019   | ARM           |
|         3 |        NULL |    NULL |         NULL |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       2 | 1GB LPDDR2-900  |           600 |            1.5 | 21-APR-2019   | Samsung       |
|         4 |        NULL |    NULL |         NULL |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       3 | 500MB LPDDR2-900|           300 |              1 | 21-APR-2019   | Samsung       |
|         5 |        NULL |    NULL |         NULL |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       4 | SoC_PCB_SMT     |           300 |              2 | 20-APR-2019   | Olimex        |
+-----------+-------------+---------+--------------+---------+---------------------+---------------------+---------+-----------------+---------------+----------------+---------------+---------------+
```

Както се вижда, техническите ключове са празни. Затова ще направя втора, отделна трансформация специално за DWH базата, за да ги намеря от всяка една DWH таблица и поставя в Report:

Заявка към DWH базата и резултатът ще бъде ще бъде:

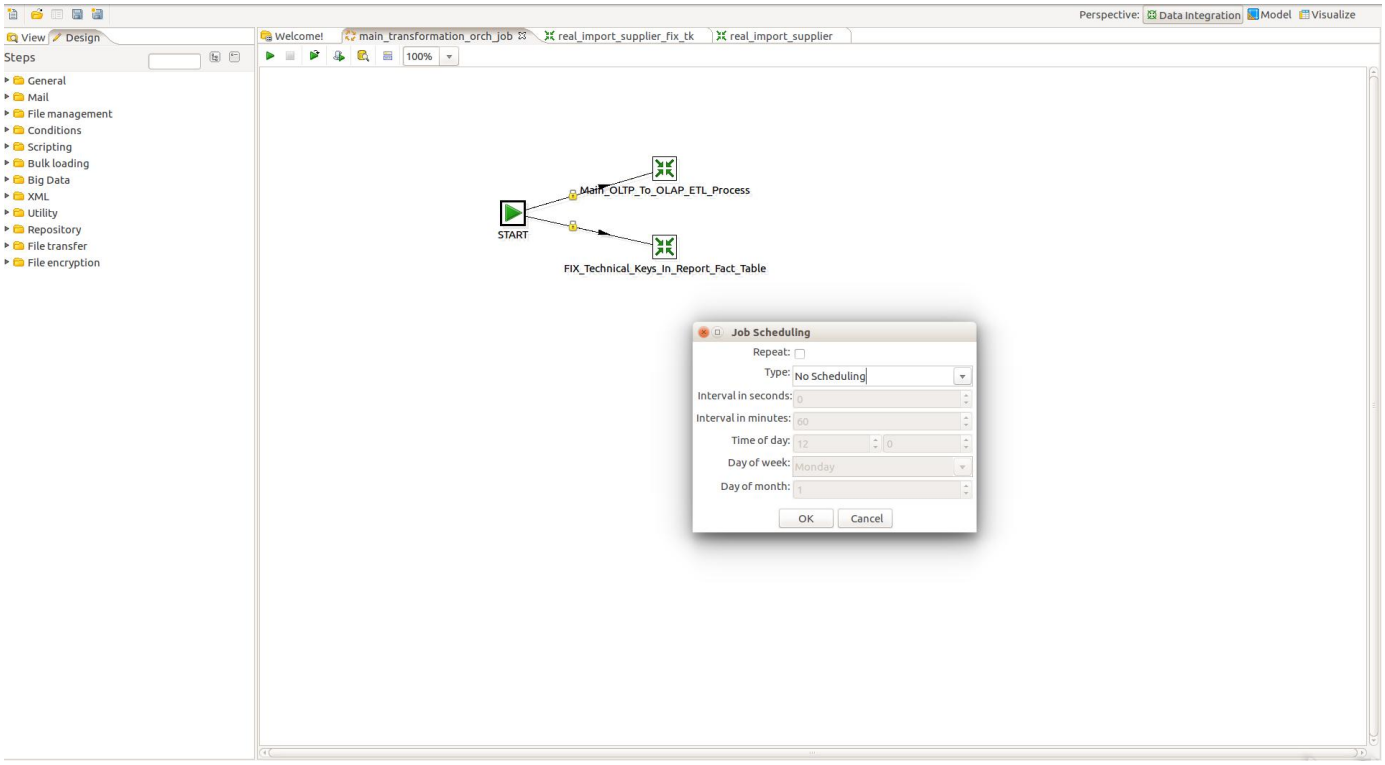
```
SELECT Report.item_id, Item.item_tk AS item_tk, Calendar.calendar_tk AS
calendar_tk, Supplier.supplier_tk AS supplier_tk
FROM Report
INNER JOIN Supplier ON Report.supplier_name = Supplier.supplier_name
INNER JOIN Calendar ON Report.item_id = Calendar.item_id
INNER JOIN Item ON Report.item_id = Item.item_id;
```

```
mysql> mysql> SELECT * FROM Report;
+-----------+-------------+---------+--------------+---------+---------------------+---------------------+---------+-----------------+---------------+----------------+---------------+---------------+
| report_tk | supplier_tk | item_tk | calendar_tk | version | date_from           | date_to             | item_id | item_name       | item_quantity | item_unit_cost | purchase_date | suppliер_name |
+-----------+-------------+---------+--------------+---------+---------------------+---------------------+---------+-----------------+---------------+----------------+---------------+---------------+
|         1 |        NULL |    NULL |         NULL |       1 | NULL                | NULL                |    NULL | NULL            |          NULL |           NULL | NULL          | NULL          |
|         2 |           1 |       2 |            3 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       1 | ARM-Cortex-A53  |          1000 |              3 | 15-APR-2019   | ARM           |
|         3 |           3 |       4 |            5 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       2 | 1GB LPDDR2-900  |           600 |            1.5 | 21-APR-2019   | Samsung       |
|         4 |           3 |       5 |            6 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       3 | 500MB LPDDR2-900|           300 |              1 | 21-APR-2019   | Samsung       |
|         5 |           2 |       3 |            4 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       4 | SoC_PCB_SMT     |           300 |              2 | 20-APR-2019   | Olimex        |
|         6 |        NULL |    NULL |         NULL |       1 | NULL                | NULL                |    NULL | NULL            |          NULL |           NULL | NULL          | NULL          |
+-----------+-------------+---------+--------------+---------+---------------------+---------------------+---------+-----------------+---------------+----------------+---------------+---------------+
```

В термини на PDI, моят ETL процес се съдържа в два отделни файла (трансфомации). Последната трансформация за техническите ключове трябва да се извърши най-накрая и е много проста. Тъй като е проста и бърза, не мога да я изпълня в същия файл за трансформации. В PDI стъпките в една трансформация се изпълняват паралелно а не

последователно, което ще рече, че простите и бързи операции ще се изпълнят преди дългите и сложните. Да я оставя в същия файл ще значи, че ще се опитвам да намирам технически ключове в DWH базата за записи, които все още не съществуват.

Ето защо в PDI има средство за оркестрация, наречено “Задание” (Job), за което бях загатнал в началото на този раздел. В него може да изберем най-вече дали искаме да изпълняваме заданията последователно (по презумция) или паралално, и да създадем график кога да се извършва процесът автоматично. Това изглежда така:



Като изпълним главното задание, DWH базата ще изглежда така:

```
mysql> select * from Calendar;
+-------------+------------+------------+---------+---------+---------------------+---------------------+
| calendar_tk | calendar_m | calendar_y | item_id | version | date_from           | date_to             |
+-------------+------------+------------+---------+---------+---------------------+---------------------+
|           1 |       NULL |       NULL |    NULL |       1 | NULL                | NULL                |
|           2 |          4 |       2019 |       1 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |
|           3 |          4 |       2019 |       4 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |
|           4 |          4 |       2019 |       2 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |
|           5 |          4 |       2019 |       3 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |
+-------------+------------+------------+---------+---------+---------------------+---------------------+
5 rows in set (0,00 sec)

mysql> select * from Item;
+---------+---------+-----------------+---------+---------------------+---------------------+
| item_tk | item_id | item_name       | version | date_from           | date_to             |
+---------+---------+-----------------+---------+---------------------+---------------------+
|       1 |    NULL | NULL            |       1 | NULL                | NULL                |
|       2 |       1 | ARM-Cortex-A53  |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |
|       3 |       4 | SoC_PCB_SMT     |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |
|       4 |       2 | 1GB LPDDR2-900  |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |
|       5 |       3 | 500MB LPDDR2-900 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |
+---------+---------+-----------------+---------+---------------------+---------------------+
5 rows in set (0,00 sec)

mysql> select * from Supplier;
+-------------+---------+---------------------+---------------------+-------------+---------------+----------------+-----------+----------+-----------------------+--------------+
| supplier_tk | version | date_from           | date_to             | supplier_id | supplier_name | supplier_phone | city      | post_code | supplier_email        | country_code |
+-------------+---------+---------------------+---------------------+-------------+---------------+----------------+-----------+----------+-----------------------+--------------+
|           0 |       1 | NULL                | NULL                |        NULL | NULL          | NULL           | NULL      |     NULL | NULL                  | NULL         |
|           1 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |           1 | ARM           | 555-123-4567   | Cambridge |    45002 | help@arm.com          | UK           |
|           2 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |           2 | Olimex        | 456-789-1011   | Plovdiv   |     2134 | order@olimex.com      | BG           |
|           3 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |           3 | Samsung       | 123-456-9876   | London    |    42131 | custserv@samsung.com  | UK           |
|           4 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |           4 | Intel         | 202-122-2324   | Karlsruhe |    12302 | sales@intel.com       | DE           |
+-------------+---------+---------------------+---------------------+-------------+---------------+----------------+-----------+----------+-----------------------+--------------+
5 rows in set (0,00 sec)

mysql> select * from Report;
+-----------+-------------+---------+-------------+---------+---------------------+---------------------+---------+-----------------+---------------+----------------+---------------+---------------+
| report_tk | supplier_tk | item_tk | calendar_tk | version | date_from           | date_to             | item_id | item_name       | item_quantity | item_unit_cost | purchase_date | supplier_name |
+-----------+-------------+---------+-------------+---------+---------------------+---------------------+---------+-----------------+---------------+----------------+---------------+---------------+
|         1 |        NULL |    NULL |        NULL |       1 | NULL                | NULL                |    NULL | NULL            |          NULL |           NULL | NULL          | NULL          |
|         2 |           1 |       2 |           2 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       1 | ARM-Cortex-A53  |          1000 |              3 | 15-APR-2019   | ARM           |
|         3 |           3 |       4 |           4 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       2 | 1GB LPDDR2-900  |           600 |            1.5 | 21-APR-2019   | Samsung       |
|         4 |           3 |       5 |           5 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       3 | 500MB LPDDR2-900 |           300 |              1 | 21-APR-2019   | Samsung       |
|         5 |           2 |       3 |           3 |       1 | 1900-01-01 00:00:00 | 2200-01-01 00:00:00 |       4 | SoC_PCB_SMT     |           300 |              2 | 20-APR-2019   | Olimex        |
|         6 |        NULL |    NULL |        NULL |       1 | NULL                | NULL                |    NULL | NULL            |          NULL |           NULL | NULL          | NULL          |
+-----------+-------------+---------+-------------+---------+---------------------+---------------------+---------+-----------------+---------------+----------------+---------------+---------------+
```

Базата е обработена и готова за включването си в Data Mart или DWH. Освен с гореспоменатия Agile BI модул на PDI, можем да я достъпим от друг продукт - Pentaho BA Server. Това вече е специализиран софтуер за визуализация на Big Data, където се изплозва и MDX езикът за заявки към OLAP системи, каквато е и Pentaho BA.

Тъй като не е главна цел на тази курсова работа, а и поради липса на време, Pentaho BA не съм включил в проекта, а само PDI.

3.2 Custom ETL Python script

Ако не бях използвал софтуерът на Pentaho от горната точка, и трябваше сам да имплементирам ETL процесът, то той би изглеждал така на Python като следва абсолютно същите стъпки:

```
import mysql.connector

# Connect to OLTP database
mydb = mysql.connector.connect(
  host = "127.0.0.1",
  user = "dwh_export",
  passwd = "123456*",
  database = "dwh"
)

# Connect to OLAP database
mydb_dwh = mysql.connector.connect(
  host = "127.0.0.1",
  user = "dwh_export",
  passwd = "123456*",
  database = "dwh_export"
)

# OLTP cursor, get data from DB with myresult
mycursor = mydb.cursor()
mycursor.execute("SELECT supplier_id, supplier_name, supplier_phone, city, post_code,
supplier_email, country_code FROM supplier;")
myresult = mycursor.fetchall()

# OLAP cursor
mycursor_dwh = mydb_dwh.cursor()

###########Truncating tables for testing purposes
mycursor_dwh.execute("TRUNCATE TABLE Calendar_Python")
mydb_dwh.commit()
mycursor_dwh.execute("TRUNCATE TABLE Item_Python")
mydb_dwh.commit()
mycursor_dwh.execute("TRUNCATE TABLE Supplier_Python")
mydb_dwh.commit()
mycursor_dwh.execute("TRUNCATE TABLE Report_Python")
mydb_dwh.commit()
##############
```

```
# Get the company contact data from the resolver file
correction_file_loc = 'pc_rezolver.csv'
read_correction_file = open(correction_file_loc, 'r')
file_contents = read_correction_file.read()
read_lines = file_contents.split(',')

# Convert all myresult elements from tuple to list, because tuple is immutable
for i in range(len(myresult)):
   myresult[i] = list(myresult[i])

# Go through every row from myresult,
if there 's missing contact information data (post_code or country), populate the empty fields for the
# corresponding by the corresponding city
for i in myresult:
# Positions in the list for post_code, city and country_code
post_code = i[len(i) - 3]
city = i[len(i) - 4]
country_code = i[len(i) - 1]
if post_code is None or country_code is None:
   j = 0

# Compare city entries to the ones in the pc_rezolver file, fix formatting(remove new lines, etc).Replace
#any None fields with the# corresponding information
for country_code and post_code
for k in read_lines:
   j = j + 1
if k == city:
   i[len(i) - 3] = read_lines[j - 2].replace("\r\n", "")
i[len(i) - 1] = read_lines[j].replace("\r\n", "")

# Substring the old IDs, cast them to INT
for the table format in DWH
for i in myresult:
   for j in i:
   id = i[0]
i[0] = int(id[2])
i[4] = int(i[4])
break

# Insert the information into the DWH Supplier_Python table
sql = "INSERT INTO Supplier_Python (supplier_id, supplier_name, supplier_phone, city, post_code,
supplier_email, country_code) VALUES (%s, %s, %s, %s, %s, %s, %s)"
mycursor_dwh.executemany(sql, myresult)
```

```
mydb_dwh.commit()
print(mycursor_dwh.rowcount, " rows were inserted in Supplier_Python OLAP DB.")

# Get the item ID and Name
for each purchase
mycursor.execute("SELECT item.item_id, item.item_name FROM item INNER JOIN itemlist ON item.item_id = itemlist.item_id INNER JOIN purchase ON purchase.purchase_id = itemlist.purchase_id WHERE purchase_del_date=0;")
myresult = mycursor.fetchall()

# Parse query result type from tuple to list
for i in range(len(myresult)):
   myresult[i] = list(myresult[i])

# Substring and parse to INT the old OLTP item_id to correspond to the type used in the OLAP DB
for i in myresult:
   id = i[0]
i[0] = int(id[2])

sql = "INSERT INTO Item_Python (item_id, item_name) VALUES (%s, %s)"
mycursor_dwh.executemany(sql, myresult)
mydb_dwh.commit()
print(mycursor_dwh.rowcount, " rows were inserted in Item_Python OLAP DB.")

mycursor.execute("SELECT purchase.month AS month, purchase.year AS year, itemlist.item_id AS item_id FROM purchase LEFT JOIN itemlist ON purchase.purchase_id=itemlist.purchase_id WHERE purchase_deleted=0;")
myresult = mycursor.fetchall()

for i in range(len(myresult)):
   myresult[i] = list(myresult[i])

# Substring and parse to INT the old OLTP item_id to correspond to the type used in the OLAP DB
for i in myresult:
   id = i[2]
i[2] = int(id[2])

sql = "INSERT INTO Calendar_Python (calendar_m, calendar_y, item_id) VALUES (%s, %s, %s)"
mycursor_dwh.executemany(sql, myresult)
mydb_dwh.commit()
print(mycursor_dwh.rowcount, " rows were inserted in Calendar_Python OLAP DB.")

mycursor.execute("SELECT item.item_id AS item_id, item.item_name AS item_name, itemlist.item_quantity AS item_quantity, itemlist.item_unit_cost AS item_unit_cost,
```

```
purchase.purchase_date, supplier.supplier_name FROM item LEFT JOIN itemlist ON item.item_id = itemlist.item_id INNER JOIN purchase ON purchase.purchase_id = itemlist.purchase_id INNER JOIN supplier ON purchase.supplier_id = supplier.supplier_id WHERE purchase.purchase_deleted=0 ORDER BY purchase.purchase_date;")
myresult = mycursor.fetchall()



for i in range(len(myresult)):
    myresult[i] = list(myresult[i])

# Substring and parse to INT the old OLTP item_id to correspond to the type used in the OLAP DB
for i in myresult:
    id = i[0]
i[0] = int(id[2])

# Insert Item data into OLAP DB
sql = "INSERT INTO Report_Python (item_id, item_name, item_quantity, item_unit_cost, purchase_date, supplier_name) VALUES (%s, %s, %s, %s, %s, %s)"
mycursor_dwh.executemany(sql, myresult)
mydb_dwh.commit()
print(mycursor_dwh.rowcount, " rows were inserted in Report_Python OLAP DB.")

# After all the data from above is entered into the database, this query gets the technical keys from each entry and enters it into the# Report_Python fact table.
mycursor_dwh.execute("SELECT Item_Python.item_tk AS item_tk, Supplier_Python.supplier_tk AS supplier_tk, Calendar_Python.calendar_tk AS calendar_tk, Report_Python.item_id FROM Report_Python INNER JOIN Supplier_Python ON Report_Python.supplier_name = Supplier_Python.supplier_name INNER JOIN Calendar_Python ON Report_Python.item_id = Calendar_Python.item_id INNER JOIN Item_Python ON Report_Python.item_id = Item_Python.item_id;")
myresult_dwh = mycursor_dwh.fetchall()

for i in range(len(myresult_dwh)):
    myresult_dwh[i] = list(myresult_dwh[i])

myresult_dwh = sorted(myresult_dwh, key = lambda myres: myresult_dwh[3])

for i in myresult_dwh:
    mycursor_dwh.execute("UPDATE Report_Python SET item_tk = {}, supplier_tk = {}, calendar_tk={} WHERE item_id={}".format(i[0], i[1], i[2], i[3]))
mydb_dwh.commit()
```

4. Използвана литература

https://www.coursera.org/learn/dwdesign/	Курс за DWH. Използвах лекциите от първите 3 седмици, но наблегнах най-вече на теорията и Pentaho.
https://wiki.pentaho.com/display/EAI/.01+Introduction+to+Spoon#id-.01IntroductiontoSpoon-TransformationDefinitions	PDI документация.
https://help.pentaho.com/Documentation/7.1/0J0/0C0/020	PDI Transformation Tutorial
https://help.pentaho.com/Documentation/7.0/0L0/0Y0/030/030/010	Running PDI Transformation
https://help.pentaho.com/Documentation/5.2/0L0/0Y0/030/030	Visualization Perspective
https://help.pentaho.com/Documentation/7.0/0D0/Pentaho_Business_Analytics	Pentaho BA Server information