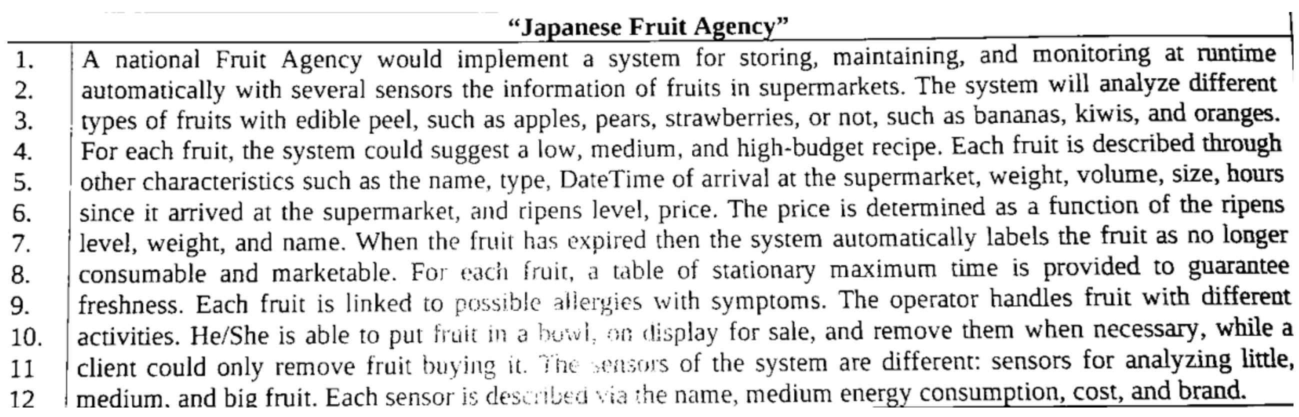
DATABASE SYSTEM EXAM

A.Y 2021/2022

Contents



"Japanese Fruit Agency"

A national Fruit Agency would implement a system for storing, maintaining, and monitoring at runtime automatically with several sensors the information of fruits in supermarkets. The system will analyze different types of fruits with edible peel, such as apples, pears, strawberries, or not, such as bananas, kiwis, and oranges. For each fruit, the system could suggest a low, medium, and high-budget recipe. Each fruit is described through other characteristics such as the name, type, DateTime of arrival at the supermarket, weight, volume, size, hours since it arrived at the supermarket, and ripens level, price. The price is determined as a function of the ripens level, weight, and name. When the fruit has expired then the system automatically labels the fruit as no longer consumable and marketable. For each fruit, a table of stationary maximum time is provided to guarantee freshness. Each fruit is linked to possible allergies with symptoms. The operator handles fruit with different activities. He/She is able to put fruit in a bowl, on display for sale, and remove them when necessary, while a client could only remove fruit buying it. The sensors of the system are different: sensors for analyzing little, medium, and big fruit. Each sensor is described via the name, medium energy consumption, cost, and brand.

Conceptual Design

1. Choose the right level of abstraction:

- The type of the fruit could be SMALL, MEDIUM, LARGE base on its dimension
- DateTime of arrival is compose of the following format dd-MM-yyyy hh:mm:ss
  - dd: 2 values for represent days
  - MM: 2 values for represent months
  - yyyy: 4 values for represent years
  - hh: 2 values for represent hours
  - mm: 2 values for represent minutes
  - ss: 2 values for represent seconds
- The ripens level is within the range 0 - 1 where:
  - 0 – rotten fruit
  - 1 – fresh fruit

2. Identify synonyms and homonyms

No synonyms found

3. Standardize the sentences

For <u>FRUIT</u> we hold name, edible peel info, date of arrival, weight, volume, dimension, ripens level, price, freshness info.

For <u>ALLERGY</u> we hold name, symptoms.

For <u>RECIPE</u> we hold name and description.

For <u>SENSOR</u> we hold name, medium consumption info, cost, brand, scan dimension.

For <u>OFFER</u> we hold discounted price and discounted weight.

For <u>USER</u> we hold name, surname, type (customer/operator).

For <u>FRESHNESS</u> stationary table we hold the name of the fruit and the freshness days.

4. Glossary

Term	Description	Synonym	Links
FRUIT	The fruit under analysis	-	ALLERGY, RECIPE, SENSOR, OFFER, FRESHNESS
ALLERGY	A damaging immune response by the body to a fruit	-	FRUIT
RECIPE	A set of instructions for making fruit dishes	-	FRUIT
SENSOR	A device which detects or measures a physical property and records, indicates, or otherwise responds to it.	-	SENSOR
OFFER	A deduction from the usual cost	-	FRUIT, USER
USER	A client/customer that interact with this system	Customer / Client	OFFER
FRESHNESS	The quality of being new	-	FRUIT

5. Reorganizing for Keywords phrases

Phrases of general nature
• A national Fruit Agency would implement a system for storing, maintaining, and monitoring at runtime automatically with several sensors the information of fruits in supermarkets.

Phrases relating to FRUIT
• The system will analyze different types of fruits with edible peel, such as apples, pears, strawberries, or not, such as bananas, kiwis, and oranges. • Each fruit is described through other characteristics such as the name, type, DateTime of arrival at the supermarket, weight, volume, size, hours since it arrived at the supermarket, and ripens level, price. • The price is determined as a function of the ripens level, weight, and name. • When the fruit has expired then the system automatically labels the fruit as no longer • consumable and marketable.

Phrases relating to ALLERGY
Each fruit is linked to possible allergies with symptoms.

Phrases relating to RECIPE
For each fruit, the system could suggest a low, medium, and high-budget recipe.

Phrases relating to SENSOR
The sensors of the system are different: sensors for analyzing little, medium and big fruit. Each sensor is desribed via the name, medium energy consumption, cost, and brand.

Phrases relating to OFFER
He/She is able to put fruit in a bowl, on display for sale …

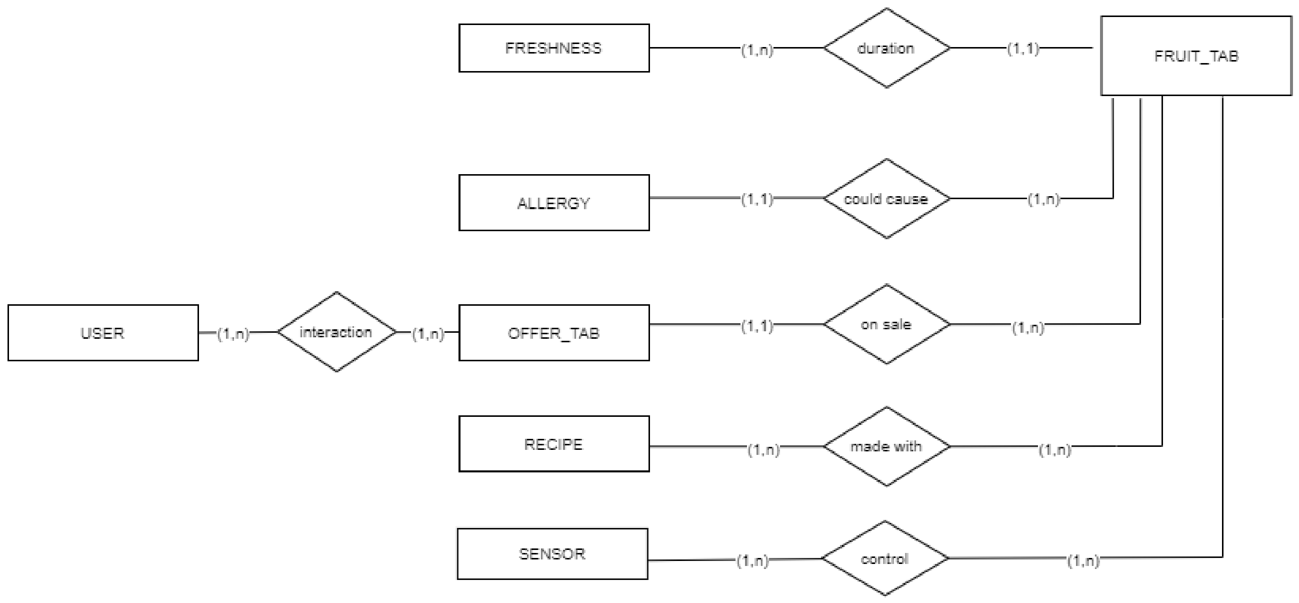
Phrases relating to FRESHNESS
For each fruit, a table of stationary maximum time is provided to guarantee Freshness.

Phrases relating to USER
The operator handles fruit with different activities. He/She is able to put fruit in a bowl, on display for sale, and remove them when necessary, while a client could only remove fruit buying it.

6. DESIGN

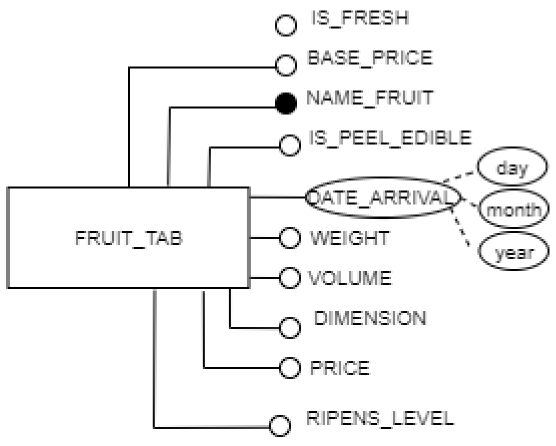
A hybrid type strategy was used for schema design. Using this approach, the various components was developed individually and then merged together in order to obtain the final schema.

Let's start with the one that will characterize the basis of the schema that provides a view on the different entities and relationships without going into detail



Now we can go into the detail on each entity

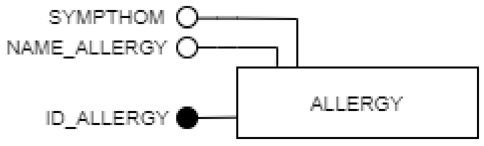
6.1 FRUIT TABLE



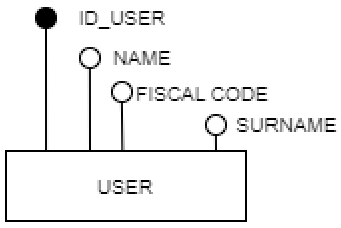
Ripens level is a value in 0-1

IS_PEEL_EDIBLE, IS_FRESH are Boolean value True, False

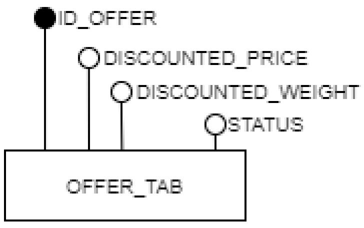
6.2 ALLERGY TABLE



6.3 USER TABLE



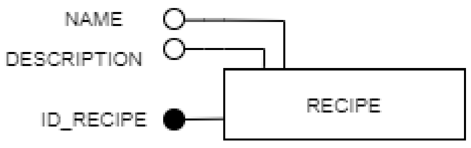
6.4 OFFER TABLE



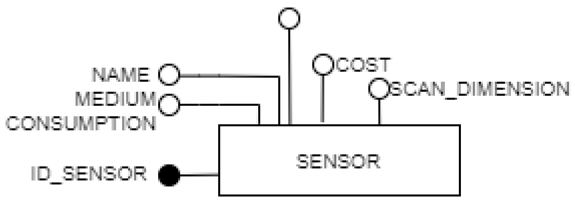
The status is one of:

- SOLD → customer buy offer fruit
- ON SALE → available fruit offer
- REMOVED → the fruit ripens level are 0 or the customer delete the offer.

6.5 RECIPE TABLE

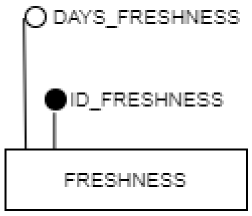


6.6 SENSOR TABLE

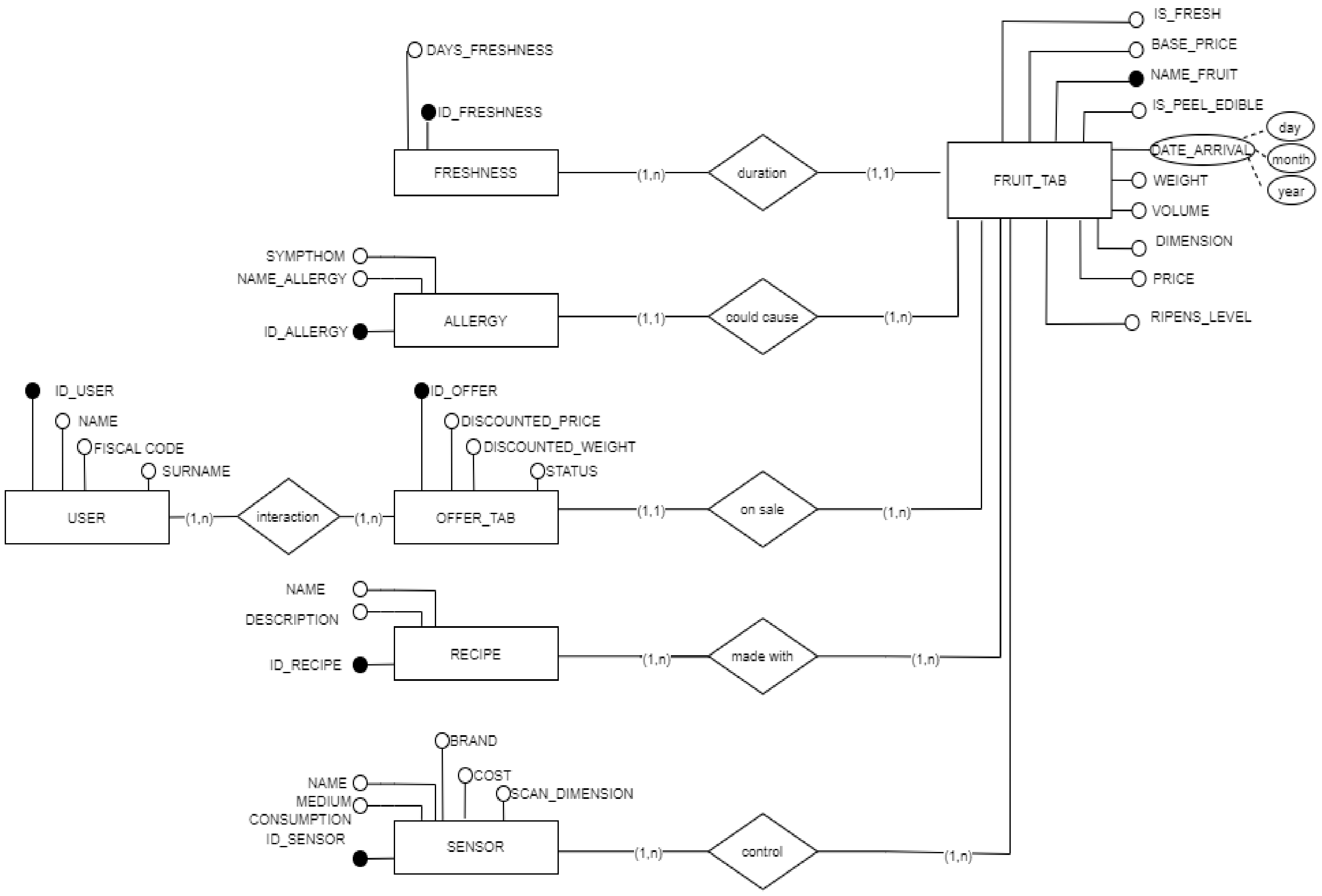


The scan dimensions corresponds to the fruit dimensions : SMALL, MEDIUM, LARGE

6.7 FRESHNESS TABLE



6.8 FINAL SCHEMA



7. Add constraints

Below there are some of the constraints that cannot be represented in the ER scheme

1. The fruit is rotten (not fresh) / not salabe in two cases:
    a. The days of guaranteed freshness are over
    b. The ripens level are 0
2. Clients can only buy offer instead of Customer that can add/delete offer
3. An offer deleted by an operator user is different from an offer bought by a customer
4. Fruits increase their ripens level every day
5. There are three types of sensor: BIG, MEDIUM, SMALL and for each size, a sensor could scan only a smaller or equal fruit dimension.
    a. BIG SENSOR could scan BIG, MEDIUM, SMALL fruits
    b. MEDIUM SENSOR could scan MEDIUM and SMALL fruits
    c. SMALL sensor could scan only SMALL fruits
6. The fact that an OFFER on a not FRESH fruit is not allowed
7. The fact that the customer should see only an offer on FRESH fruits

8. Logical Design

8.1 Analysis of redundancy

It was decided to use date instead of dateTime for represent the date of arrival of a fruit in the store.

This imply that the attribute ‘hours of arrival’ is unnecessary. For this reason this attribute is deleted.

8.2 Removal of composite and multivalued attribute

The only multivalued attribute is the date format. It is composed by three fields:

- Day
- Month
- Year

It was decided to maintain this multivalued attribute because oracle databases can support this data format.

8.3. Removal of hierarchies

No hierarchies found.

8.4. Partitioning/Merging of concepts

No partitioning or merging.

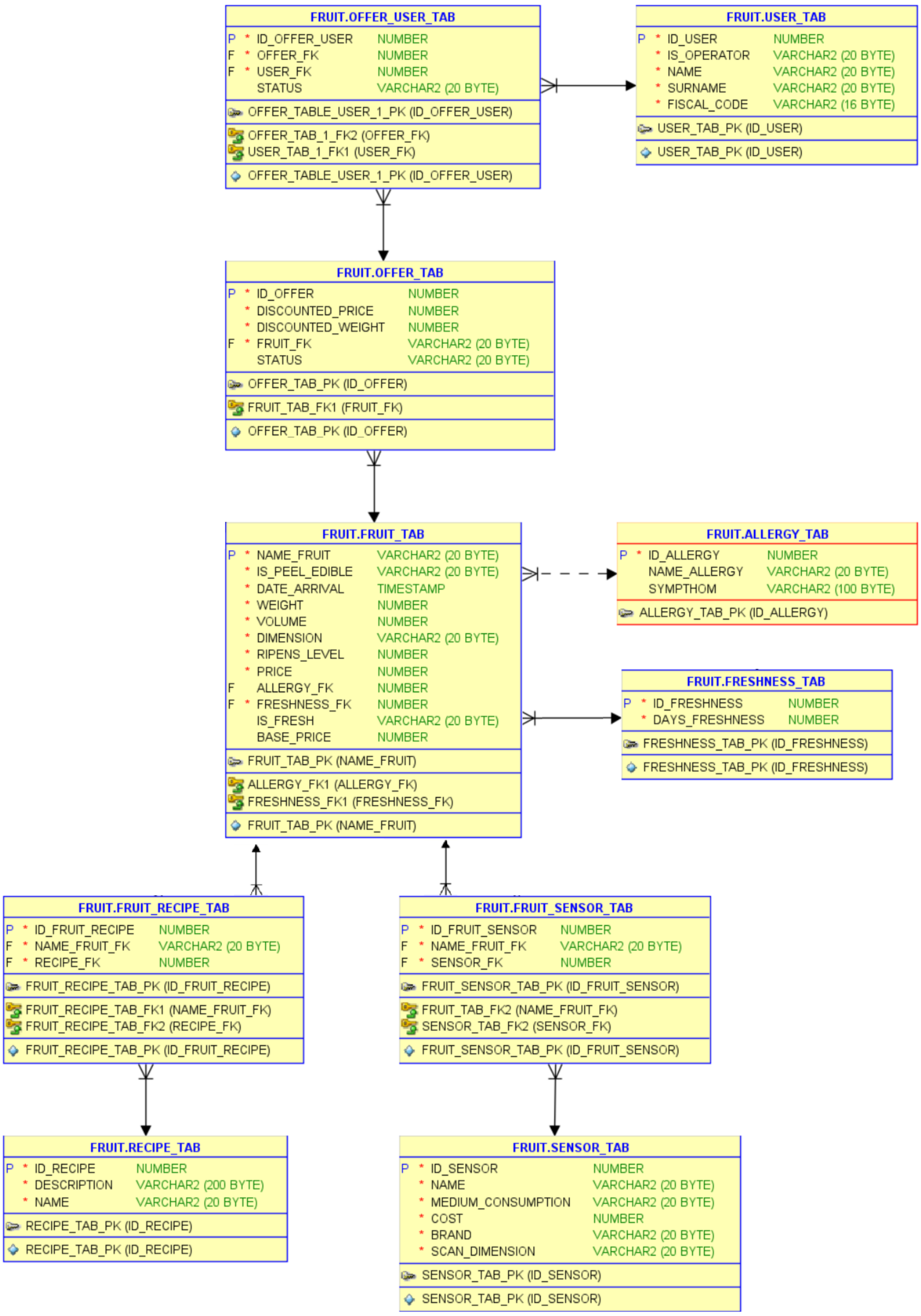
8.5. Choice of primary keys

The primary key of all entity expect FRUIT Table are number and represent a unique code for each entity.

For fruit table the attribute ‘name’ was chosen as primary key.

8.6 New Business Rules

No new business rules.


FRUIT.OFFER_USER_TAB
P * ID_OFFER_USER NUMBER
F * OFFER_FK NUMBER
F * USER_FK NUMBER
STATUS VARCHAR2 (20 BYTE)
OFFER_TABLE_USER_1_PK (ID_OFFER_USER)
OFFER_TAB_1_FK2 (OFFER_FK)
USER_TAB_1_FK1 (USER_FK)
OFFER_TABLE_USER_1_PK (ID_OFFER_USER)
FRUIT.USER_TAB
P * ID_USER NUMBER
* IS_OPERATOR VARCHAR2 (20 BYTE)
* NAME VARCHAR2 (20 BYTE)
* SURNAME VARCHAR2 (20 BYTE)
* FISCAL_CODE VARCHAR2 (16 BYTE)
USER_TAB_PK (ID_USER)
USER_TAB_PK (ID_USER)
FRUIT.OFFER_TAB
P * ID_OFFER NUMBER
* DISCOUNTED_PRICE NUMBER
* DISCOUNTED_WEIGHT NUMBER
F * FRUIT_FK VARCHAR2 (20 BYTE)
STATUS VARCHAR2 (20 BYTE)
OFFER_TAB_PK (ID_OFFER)
FRUIT_TAB_FK1 (FRUIT_FK)
OFFER_TAB_PK (ID_OFFER)
FRUIT.FRUIT_TAB
P * NAME_FRUIT VARCHAR2 (20 BYTE)
* IS_PEEL_EDIBLE VARCHAR2 (20 BYTE)
* DATE_ARRIVAL TIMESTAMP
* WEIGHT NUMBER
* VOLUME NUMBER
* DIMENSION VARCHAR2 (20 BYTE)
* RIPENS_LEVEL NUMBER
* PRICE NUMBER
F ALLERGY_FK NUMBER
F * FRESHNESS_FK NUMBER
IS_FRESH VARCHAR2 (20 BYTE)
BASE_PRICE NUMBER
FRUIT_TAB_PK (NAME_FRUIT)
ALLERGY_FK1 (ALLERGY_FK)
FRESHNESS_FK1 (FRESHNESS_FK)
FRUIT_TAB_PK (NAME_FRUIT)
FRUIT.ALLERGY_TAB
P * ID_ALLERGY NUMBER
NAME_ALLERGY VARCHAR2 (20 BYTE)
SYMPTHOM VARCHAR2 (100 BYTE)
ALLERGY_TAB_PK (ID_ALLERGY)
FRUIT.FRESHNESS_TAB
P * ID_FRESHNESS NUMBER
* DAYS_FRESHNESS NUMBER
FRESHNESS_TAB_PK (ID_FRESHNESS)
FRESHNESS_TAB_PK (ID_FRESHNESS)
FRUIT.FRUIT_RECIPE_TAB
P * ID_FRUIT_RECIPE NUMBER
F * NAME_FRUIT_FK VARCHAR2 (20 BYTE)
F * RECIPE_FK NUMBER
FRUIT_RECIPE_TAB_PK (ID_FRUIT_RECIPE)
FRUIT_RECIPE_TAB_FK1 (NAME_FRUIT_FK)
FRUIT_RECIPE_TAB_FK2 (RECIPE_FK)
FRUIT_RECIPE_TAB_PK (ID_FRUIT_RECIPE)
FRUIT.FRUIT_SENSOR_TAB
P * ID_FRUIT_SENSOR NUMBER
F * NAME_FRUIT_FK VARCHAR2 (20 BYTE)
F * SENSOR_FK NUMBER
FRUIT_SENSOR_TAB_PK (ID_FRUIT_SENSOR)
FRUIT_TAB_FK2 (NAME_FRUIT_FK)
SENSOR_TAB_FK2 (SENSOR_FK)
FRUIT_SENSOR_TAB_PK (ID_FRUIT_SENSOR)
FRUIT.RECIPE_TAB
P * ID_RECIPE NUMBER
* DESCRIPTION VARCHAR2 (200 BYTE)
* NAME VARCHAR2 (20 BYTE)
RECIPE_TAB_PK (ID_RECIPE)
RECIPE_TAB_PK (ID_RECIPE)
FRUIT.SENSOR_TAB
P * ID_SENSOR NUMBER
* NAME VARCHAR2 (20 BYTE)
* MEDIUM_CONSUMPTION VARCHAR2 (20 BYTE)
* COST NUMBER
* BRAND VARCHAR2 (20 BYTE)
* SCAN_DIMENSION VARCHAR2 (20 BYTE)
SENSOR_TAB_PK (ID_SENSOR)
SENSOR_TAB_PK (ID_SENSOR)

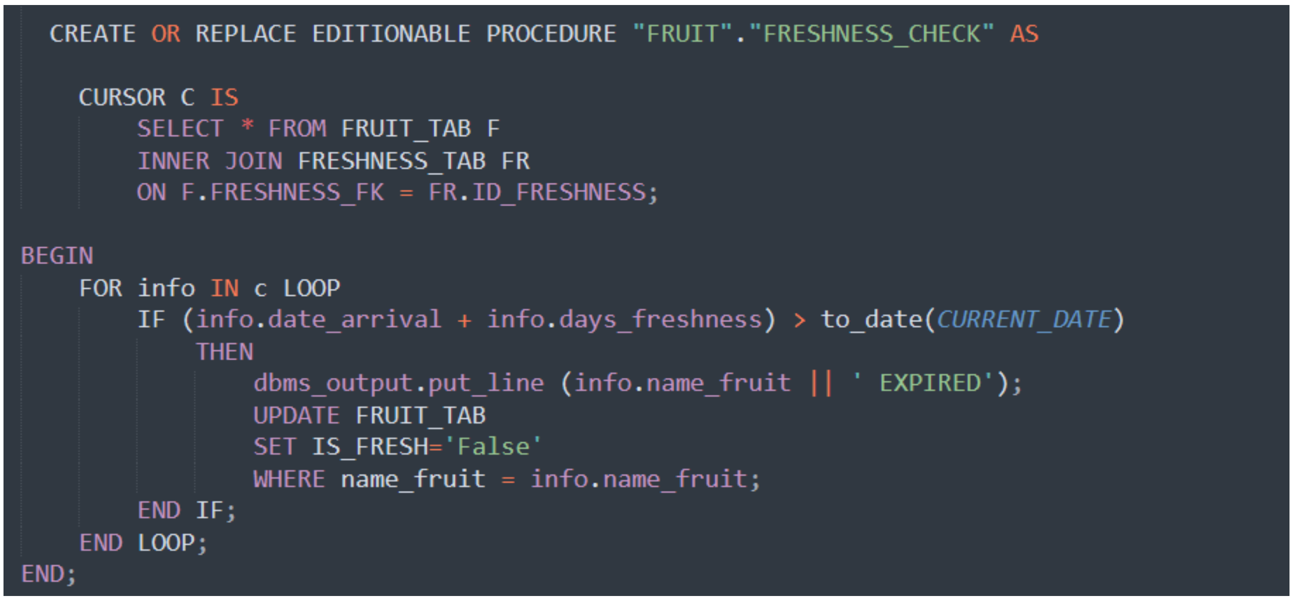
Francesco Ranieri

9.0 Physical Design

9.1 Procedures

All the procedures are used in job schedulers.

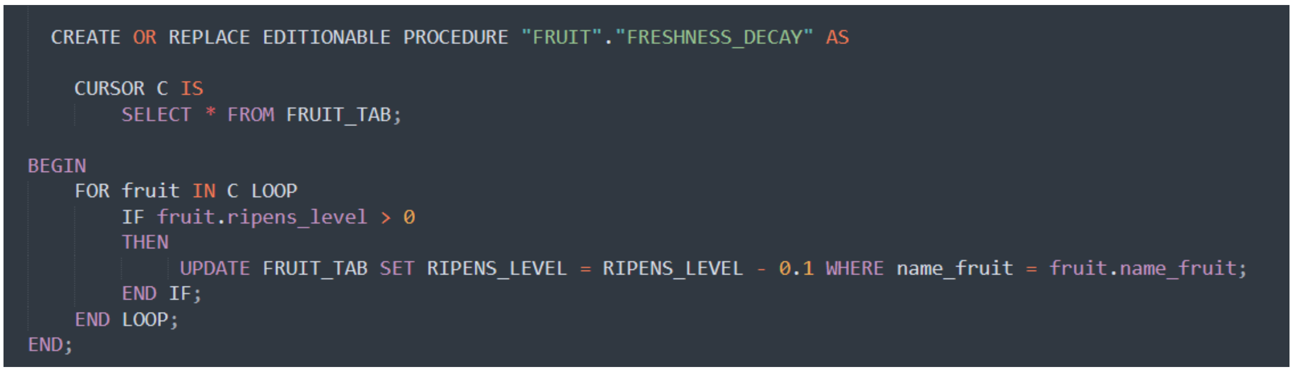
FRESHNESS CHECK PROCEDURE



```
 CREATE OR REPLACE EDITIONABLE PROCEDURE "FRUIT"."FRESHNESS_CHECK" AS

    CURSOR C IS
        SELECT * FROM FRUIT_TAB F
        INNER JOIN FRESHNESS_TAB FR
        ON F.FRESHNESS_FK = FR.ID_FRESHNESS;

BEGIN
    FOR info IN c LOOP
        IF (info.date_arrival + info.days_freshness) > to_date(CURRENT_DATE)
            THEN
                dbms_output.put_line (info.name_fruit || ' EXPIRED');
                UPDATE FRUIT_TAB
                SET IS_FRESH='False'
                WHERE name_fruit = info.name_fruit;
        END IF;
    END LOOP;
END;
```

This procedure checks if a fruit or not. To do that the procedure checks if the date of arrival of the fruit in the store, added to the number of days of freshness guaranteed, is greater than the daily date. If so, then set the attribute is_fresh of that fruit to False because it is rotten and no longer fresh.

FRESHNESS DECAY PROCEDURE



```
 CREATE OR REPLACE EDITIONABLE PROCEDURE "FRUIT"."FRESHNESS_DECAY" AS

    CURSOR C IS
        SELECT * FROM FRUIT_TAB;

BEGIN
    FOR fruit IN C LOOP
        IF fruit.ripens_level > 0
        THEN
            UPDATE FRUIT_TAB SET RIPENS_LEVEL = RIPENS_LEVEL - 0.1 WHERE name_fruit = fruit.name_fruit;
        END IF;
    END LOOP;
END;
```

This procedure subtracts 0.1 to the ripens level of each fruit.

CREATE OFFER ON LOW RIPENS LEVEL PROCEDURE

```
  CREATE OR REPLACE EDITIONABLE PROCEDURE "FRUIT"."OFFER_LOW_RIPENS_LEVEL" AS
    CURSOR C IS
        SELECT * FROM FRUIT_TAB;
BEGIN
    FOR fruit IN c LOOP
        IF fruit.weight > 0
            THEN
                IF fruit.ripens_level > 0 AND fruit.ripens_level < 0.5
                    THEN
                        UPDATE OFFER_TAB
                            SET
                                discounted_price = fruit.price - 0.2,
                                discounted_weight = fruit.weight,
                                fruit_fk = fruit.name_fruit,
                                status = 'ON SALE'
                            where
                                fruit_fk = fruit.name_fruit;
                        IF (sql%notfound)
                            THEN
                                INSERT INTO OFFER_TAB(discounted_price, discounted_weight, fruit_fk)
                                VALUES (fruit.price - 0.2, fruit.weight, fruit.name_fruit);
                        END IF;
                ELSE
                    IF fruit.ripens_level = 0
                    THEN
                        UPDATE OFFER_TAB
                            SET
                                status = 'REMOVED'
                            where
                                fruit_fk = fruit.name_fruit;
                        UPDATE FRUIT_TAB SET is_fresh = 'False' where name_fruit = fruit.name_fruit;
                    END IF;
                END IF;
        END IF;
    END LOOP;
END;
```

The objective of this procedure is to simulate the creation of an offer when the level of ripeness of a fruit is very high. Of course, all the fruit weight in the system is involved in the offer.

In this way the aim is to sell the entire stock of ripe fruit as soon as possible. To do that every time this procedure is execute the discounted price of the fruit is decreased by 0.20€.

When this procedure detects a potential fruit to put on offer, it tries to update an offer with the new price and, if no rows are affected from this update query, we are in the case that there is no offer for that fruit and so the next step is to create a new one.

Finally, when this procedure finds a fruit with ripens level 0, then all the offers on that fruit are set to 'REMOVED' and customer are no longer able to see that offer.

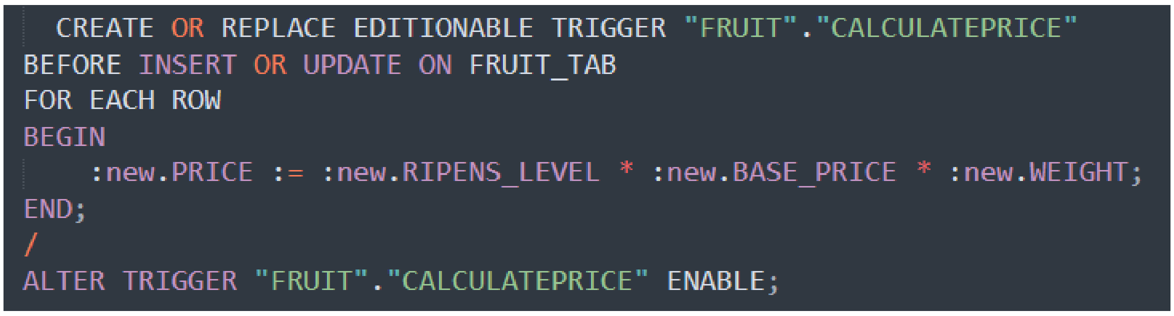
REFILL PROCEDURE

```
  CREATE OR REPLACE EDITIONABLE PROCEDURE "FRUIT"."REFILL_FRUITS" AS

BEGIN
    UPDATE FRUIT_TAB SET weight = weight + 30;
END REFILL_FRUITS;
```

Increate fruits quantity by 30 kg.

9.2 Triggers

CALCUALTE PRICE TRIGGER



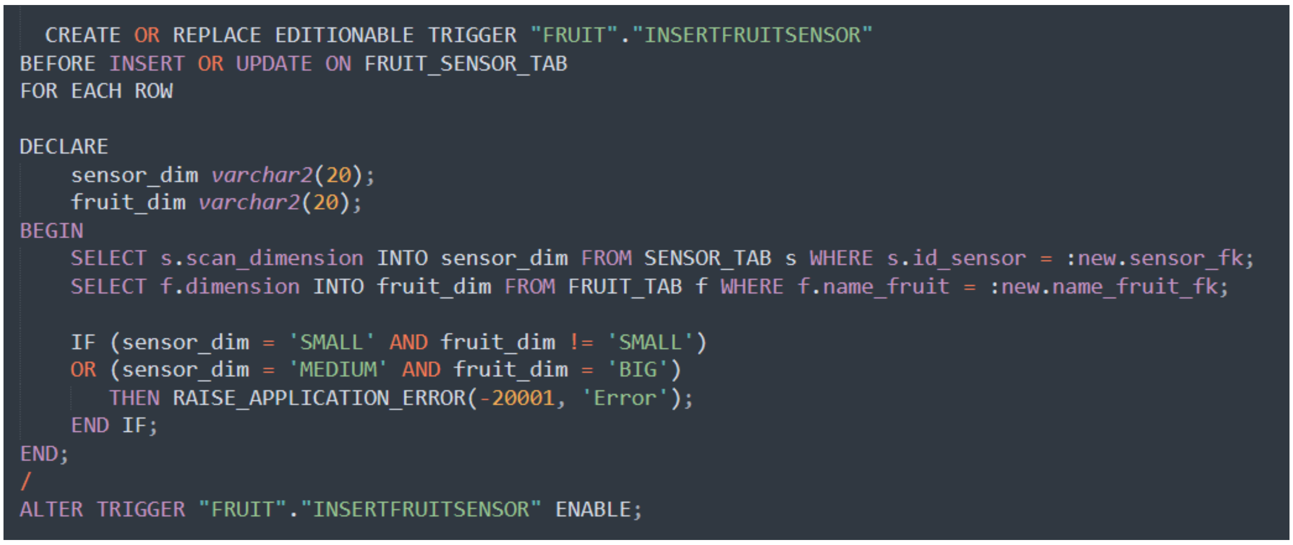
```
  CREATE OR REPLACE EDITIONABLE TRIGGER "FRUIT"."CALCULATEPRICE"
BEFORE INSERT OR UPDATE ON FRUIT_TAB
FOR EACH ROW
BEGIN
    :new.PRICE := :new.RIPENS_LEVEL * :new.BASE_PRICE * :new.WEIGHT;
END;
/
ALTER TRIGGER "FRUIT"."CALCULATEPRICE" ENABLE;
```

This trigger recalculates the price of the fruit based on the base price, the ripens level and the weight.

The formula is the following:

NEW_PRICE = BASE_PRICE * WEIGHT * RIPENS_LEVEL

INSERT FRUIT SENSOR TRIGGER



```
  CREATE OR REPLACE EDITIONABLE TRIGGER "FRUIT"."INSERTFRUITSENSOR"
BEFORE INSERT OR UPDATE ON FRUIT_SENSOR_TAB
FOR EACH ROW

DECLARE
    sensor_dim varchar2(20);
    fruit_dim varchar2(20);
BEGIN
    SELECT s.scan_dimension INTO sensor_dim FROM SENSOR_TAB s WHERE s.id_sensor = :new.sensor_fk;
    SELECT f.dimension INTO fruit_dim FROM FRUIT_TAB f WHERE f.name_fruit = :new.name_fruit_fk;

    IF (sensor_dim = 'SMALL' AND fruit_dim != 'SMALL')
    OR (sensor_dim = 'MEDIUM' AND fruit_dim = 'BIG')
       THEN RAISE_APPLICATION_ERROR(-20001, 'Error');
    END IF;
END;
/
ALTER TRIGGER "FRUIT"."INSERTFRUITSENSOR" ENABLE;
```

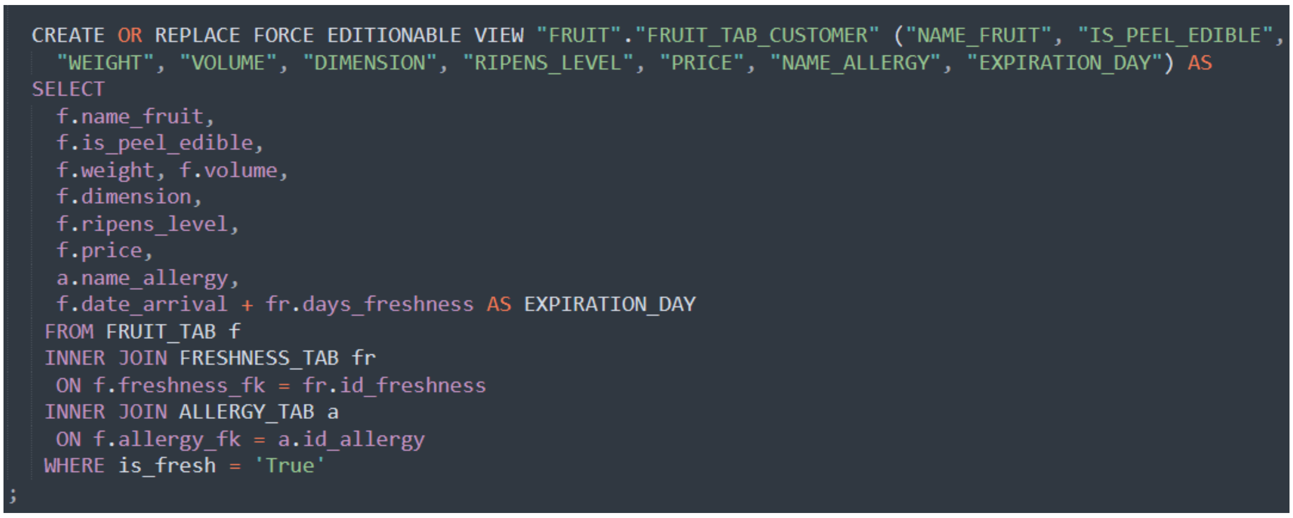
This trigger guarantee that for each size, a sensor could scan only a smaller or equal fruit dimension.

- BIG SENSOR could scan BIG, MEDIUM, SMALL fruits
- MEDIUM SENSOR could scan MEDIUM and SMALL fruits
- SMALL sensor could scan only SMALL fruits

9.3 Views

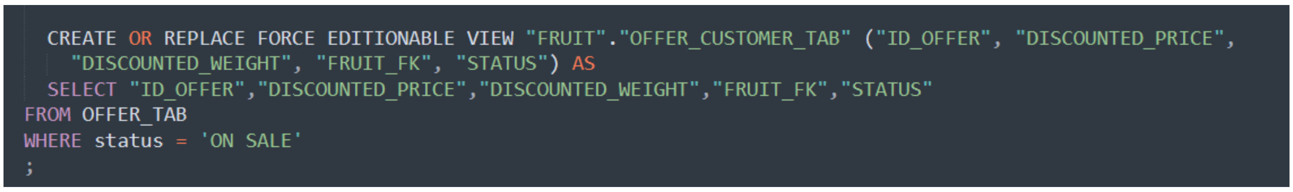
Two views are created for CUSTOMER experience. In this way the customer are able to:

1) view all the info on a fruit



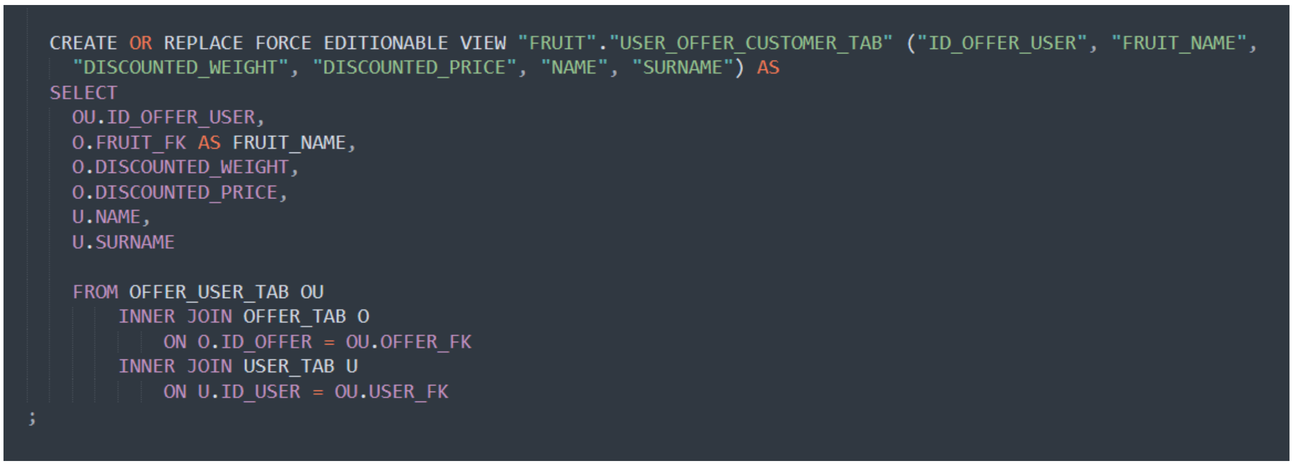
```sql
 CREATE OR REPLACE FORCE EDITIONABLE VIEW "FRUIT"."FRUIT_TAB_CUSTOMER" ("NAME_FRUIT", "IS_PEEL_EDIBLE",
   "WEIGHT", "VOLUME", "DIMENSION", "RIPENS_LEVEL", "PRICE", "NAME_ALLERGY", "EXPIRATION_DAY") AS
 SELECT
   f.name_fruit,
   f.is_peel_edible,
   f.weight, f.volume,
   f.dimension,
   f.ripens_level,
   f.price,
   a.name_allergy,
   f.date_arrival + fr.days_freshness AS EXPIRATION_DAY
  FROM FRUIT_TAB f
  INNER JOIN FRESHNESS_TAB fr
   ON f.freshness_fk = fr.id_freshness
  INNER JOIN ALLERGY_TAB a
   ON f.allergy_fk = a.id_allergy
  WHERE is_fresh = 'True'
;
```

2) view only 'ON SALE' offer which means view all available offer on fresh fruits



```sql
  CREATE OR REPLACE FORCE EDITIONABLE VIEW "FRUIT"."OFFER_CUSTOMER_TAB" ("ID_OFFER", "DISCOUNTED_PRICE",
    "DISCOUNTED_WEIGHT", "FRUIT_FK", "STATUS") AS
  SELECT "ID_OFFER","DISCOUNTED_PRICE","DISCOUNTED_WEIGHT","FRUIT_FK","STATUS"
FROM OFFER_TAB
WHERE status = 'ON SALE'
;
```

The last view is created to keep track of the customer purchases. This view show all info about a customer and the offer that he/she has bought.

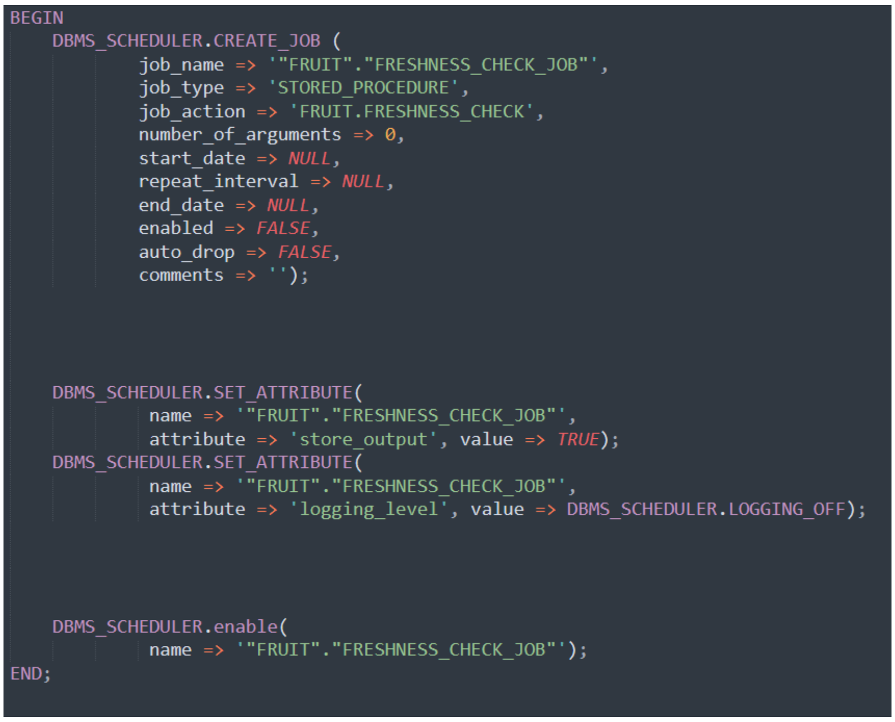


```sql
  CREATE OR REPLACE FORCE EDITIONABLE VIEW "FRUIT"."USER_OFFER_CUSTOMER_TAB" ("ID_OFFER_USER", "FRUIT_NAME",
    "DISCOUNTED_WEIGHT", "DISCOUNTED_PRICE", "NAME", "SURNAME") AS
  SELECT
    OU.ID_OFFER_USER,
    O.FRUIT_FK AS FRUIT_NAME,
    O.DISCOUNTED_WEIGHT,
    O.DISCOUNTED_PRICE,
    U.NAME,
    U.SURNAME

    FROM OFFER_USER_TAB OU
        INNER JOIN OFFER_TAB O
            ON O.ID_OFFER = OU.OFFER_FK
        INNER JOIN USER_TAB U
            ON U.ID_USER = OU.USER_FK
;
```

9.4 JOB SCHEDULERS

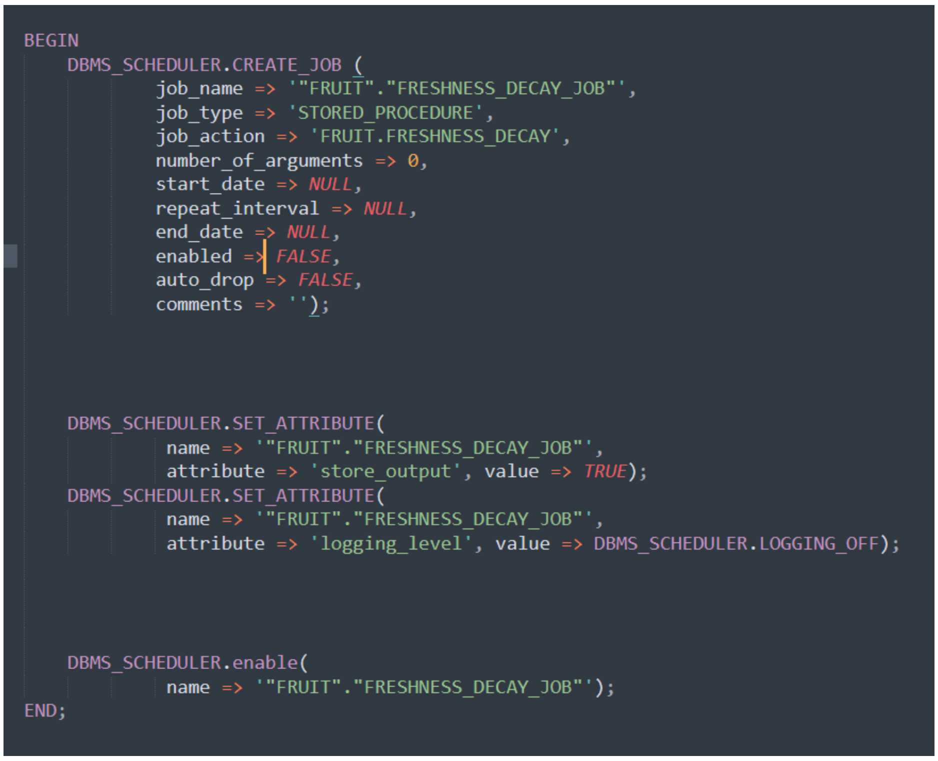
I created also 4 job schedulers, one per each procedure and in the following configuration the repeating time of the job is set to 1 minute but in a real case scenario all these jobs are designed to be run daily.

FRESHNESS CHECK JOB



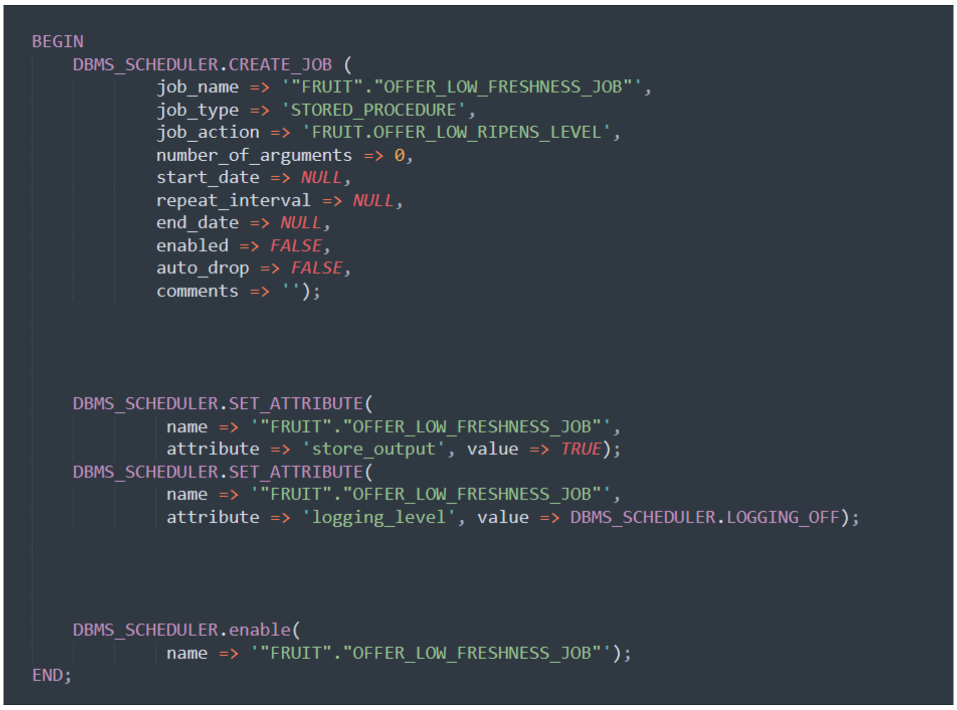
```
BEGIN
    DBMS_SCHEDULER.CREATE_JOB (
            job_name => '"FRUIT"."FRESHNESS_CHECK_JOB"',
            job_type => 'STORED_PROCEDURE',
            job_action => 'FRUIT.FRESHNESS_CHECK',
            number_of_arguments => 0,
            start_date => NULL,
            repeat_interval => NULL,
            end_date => NULL,
            enabled => FALSE,
            auto_drop => FALSE,
            comments => '');



    DBMS_SCHEDULER.SET_ATTRIBUTE(
             name => '"FRUIT"."FRESHNESS_CHECK_JOB"',
             attribute => 'store_output', value => TRUE);
    DBMS_SCHEDULER.SET_ATTRIBUTE(
             name => '"FRUIT"."FRESHNESS_CHECK_JOB"',
             attribute => 'logging_level', value => DBMS_SCHEDULER.LOGGING_OFF);



    DBMS_SCHEDULER.enable(
             name => '"FRUIT"."FRESHNESS_CHECK_JOB"');
END;
```

FRESHNESS DECAY JOB



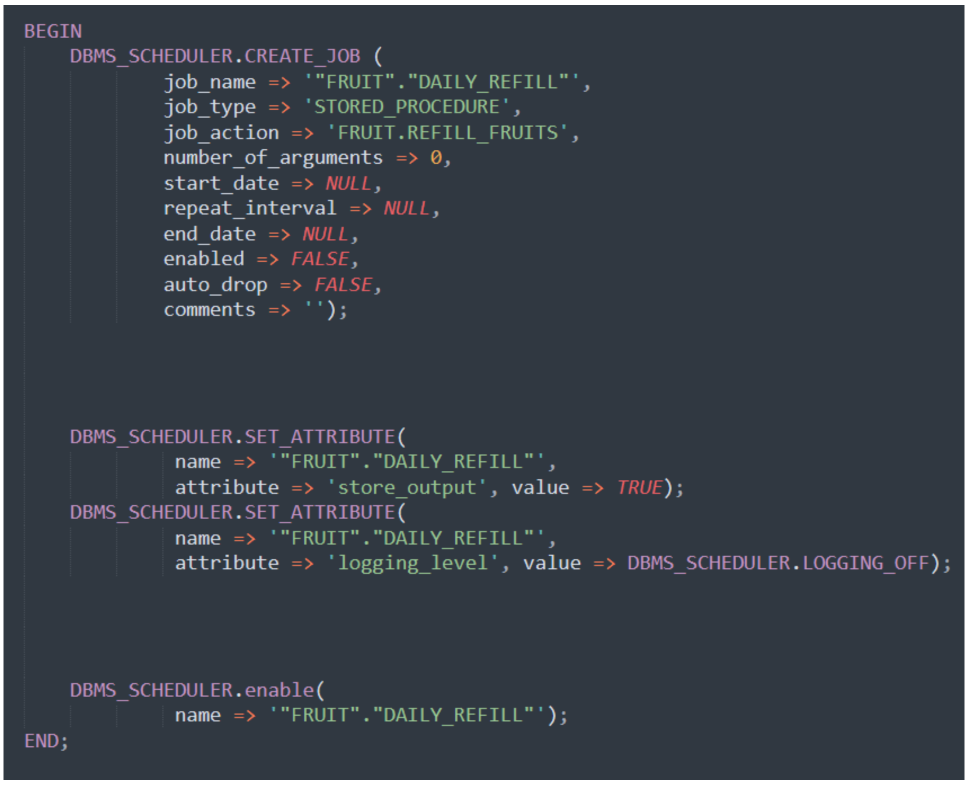
```
BEGIN
    DBMS_SCHEDULER.CREATE_JOB (
            job_name => '"FRUIT"."FRESHNESS_DECAY_JOB"',
            job_type => 'STORED_PROCEDURE',
            job_action => 'FRUIT.FRESHNESS_DECAY',
            number_of_arguments => 0,
            start_date => NULL,
            repeat_interval => NULL,
            end_date => NULL,
            enabled => FALSE,
            auto_drop => FALSE,
            comments => '');



    DBMS_SCHEDULER.SET_ATTRIBUTE(
             name => '"FRUIT"."FRESHNESS_DECAY_JOB"',
             attribute => 'store_output', value => TRUE);
    DBMS_SCHEDULER.SET_ATTRIBUTE(
             name => '"FRUIT"."FRESHNESS_DECAY_JOB"',
             attribute => 'logging_level', value => DBMS_SCHEDULER.LOGGING_OFF);



    DBMS_SCHEDULER.enable(
             name => '"FRUIT"."FRESHNESS_DECAY_JOB"');
END;
```

OFFER ON LOW LEVEL RIPENS JOB



```
BEGIN
    DBMS_SCHEDULER.CREATE_JOB (
            job_name => '"FRUIT"."OFFER_LOW_FRESHNESS_JOB"',
            job_type => 'STORED_PROCEDURE',
            job_action => 'FRUIT.OFFER_LOW_RIPENS_LEVEL',
            number_of_arguments => 0,
            start_date => NULL,
            repeat_interval => NULL,
            end_date => NULL,
            enabled => FALSE,
            auto_drop => FALSE,
            comments => '');



    DBMS_SCHEDULER.SET_ATTRIBUTE(
             name => '"FRUIT"."OFFER_LOW_FRESHNESS_JOB"',
             attribute => 'store_output', value => TRUE);
    DBMS_SCHEDULER.SET_ATTRIBUTE(
             name => '"FRUIT"."OFFER_LOW_FRESHNESS_JOB"',
             attribute => 'logging_level', value => DBMS_SCHEDULER.LOGGING_OFF);



    DBMS_SCHEDULER.enable(
             name => '"FRUIT"."OFFER_LOW_FRESHNESS_JOB"');
END;
```

REFILL PROCEDURE



```
BEGIN
    DBMS_SCHEDULER.CREATE_JOB (
            job_name => '"FRUIT"."DAILY_REFILL"',
            job_type => 'STORED_PROCEDURE',
            job_action => 'FRUIT.REFILL_FRUITS',
            number_of_arguments => 0,
            start_date => NULL,
            repeat_interval => NULL,
            end_date => NULL,
            enabled => FALSE,
            auto_drop => FALSE,
            comments => '');



    DBMS_SCHEDULER.SET_ATTRIBUTE(
             name => '"FRUIT"."DAILY_REFILL"',
             attribute => 'store_output', value => TRUE);
    DBMS_SCHEDULER.SET_ATTRIBUTE(
             name => '"FRUIT"."DAILY_REFILL"',
             attribute => 'logging_level', value => DBMS_SCHEDULER.LOGGING_OFF);



    DBMS_SCHEDULER.enable(
             name => '"FRUIT"."DAILY_REFILL"');
END;
```