# Task

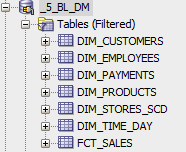
Create required objects:

* Create all required fact table objects on ST Layer (T\_...).
* Grant all required privileges to ST\_CL (Cleansing Layer).
* Create packages to load fact data (one package = one fact table) on ST\_CL.
* Prepare document with description of your load and screenshot of data in the Fact Table.
* Test Procedure for Repeatable execution (it should produce same consistent results each time on a same source data).

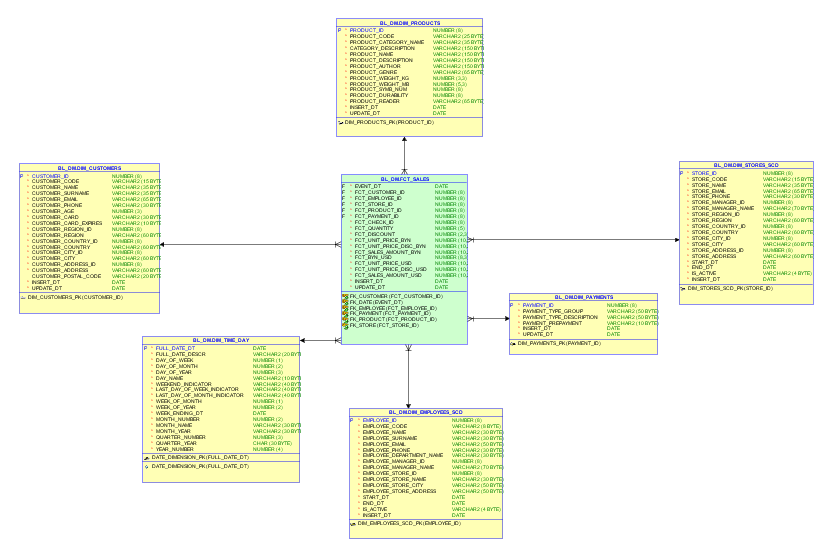
# Solution

## DDL

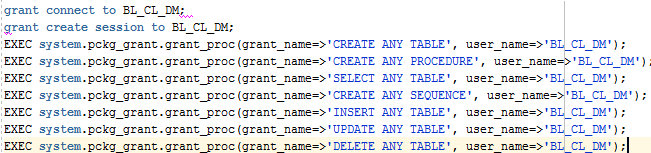
Все необходимые таблицы на уровне BL\_CM были созданы (скрипты в папке).



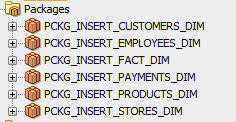
Результат выполнения:



## Гранты

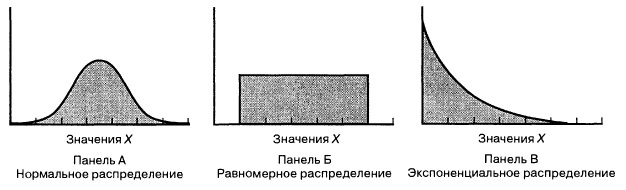


## Пакеты



## Генерация факта

Стояла цель сгенерировать значения, которые бы распределялись нормально, а не равномерно:



Так как функцию dbms\_random.normal() очень тяжело контролировать и нельзя передать промежуток, в котором должны находиться числа, я написала две функции, где сымитировала нормальное распределение.

## Randomize\_max func

Эта функция для тех значений, промежуток которых может изменяться и заранее неизвестен (id магазинов, клиентов, работников итд).

FUNCTION randomize\_max(

table\_name IN VARCHAR2,

col\_name IN VARCHAR2)

RETURN NUMBER;

В функцию в качестве входных параметров передаются имя таблицы и значение поля, для которых генерируются значения:

randomize\_max('bl\_3nf.ce\_customers', 'customer\_id')

randomize\_max('bl\_3nf.ce\_employees', 'employee\_id')

Затем с помощью EXECUTE IMMEDIATE находится максимальное значение в колонке ( то есть значение, до которого надо генерировать).

EXECUTE IMMEDIATE 'SELECT MAX('||col\_name||') FROM '||table\_name INTO v\_max;

Сама генерация:

Сначала генерируется число от 1 до 10, затем оно проверяется, и, в зависимости от значения, генерируется id которое нам нужно.

i:=dbms\_random.value(1,10);

result:=

CASE

WHEN i<=7 THEN

CEIL(dbms\_random.value(1,(v\_max\*0.45))) --45%

WHEN i>7 AND i<=9 THEN

CEIL(dbms\_random.value((v\_max\*0.45),(v\_max\*0.97))) --52%

WHEN i<10 THEN

CEIL(dbms\_random.value((v\_max\*0.97),(v\_max\*1))) --3%

ELSE

'1'

END ;

Первый промежуток взят от 1, так как преполагается, что 3НФ таблица заполняется равномерно (нет дырок в id) и нумерация начинается с 1.

Также использовалась функция CEIL, чтобы избежать значения 0.

## Randomize func

В жту функцию можно самому передавать значения, в пределах которых будут генерироваться промежутки:

FUNCTION randomize(

a IN NUMBER,

b IN NUMBER,

c IN NUMBER,

d IN NUMBER,

e IN NUMBER)

RETURN NUMBER

Вызов функции:

randomize(1,4,10,20,50) as quantity

Принцип такой же, как в предыдущей функции:

Больше всего значений в таблице будет в первом промежутку, чуть меньше во втором, меньше всего в четвертом промежутке.

i:=dbms\_random.value(1,20);

result:=

CASE

WHEN i<=10 THEN

CEIL(dbms\_random.value(a,b))

WHEN i>10 AND i<=16 THEN

CEIL(dbms\_random.value(b,c))

WHEN i>16 AND i<=19 THEN

CEIL(dbms\_random.value(c,d))

WHEN i<20 THEN

CEIL(dbms\_random.value(d,e))

ELSE

'1'

END ;

Также использовалась функция CEIL, чтобы избежать значения 0.

## Virtual columns

Для колонок, которые вычисляются на основе других колонок, я использовала virtual columns, то есть они будут вычислятся автоматически и нет необходимости передавать их значения в таблицу.

Fct\_unit\_price\_disc\_BYN generated always

AS

(Fct\_unit\_price\_BYN\*Fct\_discount), Fct\_sales\_Amount\_BYN generated always

AS

(Fct\_unit\_price\_BYN\*Fct\_discount \*Fct\_quantity ) , Fct\_BYN\_USD NUMBER(8,3) , Fct\_unit\_price\_USD generated always

AS

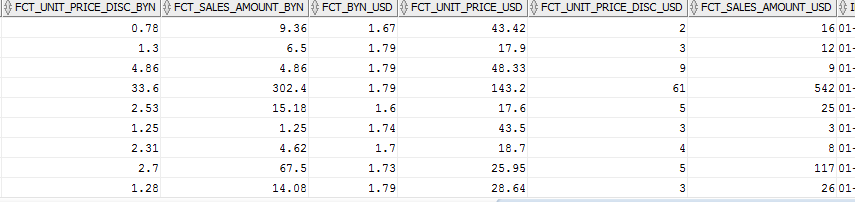
(Fct\_BYN\_USD \* Fct\_unit\_price\_BYN), Fct\_unit\_price\_disc\_USD generated always

AS

(CEIL(Fct\_BYN\_USD \* Fct\_unit\_price\_BYN\*Fct\_discount)) , Fct\_sales\_Amount\_USD generated always

AS

(CEIL(Fct\_BYN\_USD \* Fct\_unit\_price\_BYN\*Fct\_discount \*Fct\_quantity ))



## Repeatable execution

Пакеты можно запускать несколько раз.





