LAB 1

Excercice 1.3 :

1. number of customers per country

```
SELECT COUNTRY, COUNT(*) FROM CUSTOMERS GROUP
BY COUNTRY;
```

2. number of orders per country,(country and city), and in total. Results are ordered by alphabetical order on country then city.

```
SELECT SHIP_COUNTRY, SHIP_CITY, COUNT(*) FROM
ORDERS GROUP BY ROLLUP(SHIP_COUNTRY,SHIP_CITY)
ORDER BY SHIP_COUNTRY,SHIP_CITY;
```

3. number of orders and quantity of items shipped (according to order details) for each pair of Customer country and Supplier country. Order result by customer country first, then supplier country.

```
SELECT CUSTOMERS.COUNTRY AS C_COUNTRY,
SUPPLIERS.COUNTRY, COUNT(*) AS NBORDERS,
SUM(ORDER_DETAILS.QUANTITY) AS QUANTITY FROM
CUSTOMERS,SUPPLIERS,ORDERS,ORDER_DETAILS,PRODUC
TS WHERE ORDER_DETAILS.ORDER_ID =
ORDERS.ORDER_ID AND ORDER_DETAILS.PRODUCT_ID =
PRODUCTS.PRODUCT_ID AND PRODUCTS.SUPPLIER_ID =
SUPPLIERS.SUPPLIER_ID AND ORDERS.CUSTOMER_ID =
CUSTOMERS.CUSTOMER_ID GROUP BY
CUSTOMERS.COUNTRY,SUPPLIERS.COUNTRY ORDER BY
CUSTOMERS.COUNTRY,SUPPLIERS.COUNTRY;
```

4.

```
SELECT CUSTOMERS.COUNTRY AS C_COUNTRY,
SUPPLIERS.COUNTRY, COUNT(*) AS NBORDERS,
SUM(ORDER_DETAILS.QUANTITY) AS QUANTITY FROM
CUSTOMERS,SUPPLIERS,ORDERS,ORDER_DETAILS,PRODUC
TS WHERE ORDER_DETAILS.ORDER_ID =
ORDERS.ORDER_ID AND ORDER_DETAILS.PRODUCT_ID =
PRODUCTS.PRODUCT_ID AND PRODUCTS.SUPPLIER_ID =
SUPPLIERS.SUPPLIER_ID AND ORDERS.CUSTOMER_ID =
CUSTOMERS.CUSTOMER_ID GROUP BY
CUBE(CUSTOMERS.COUNTRY,SUPPLIERS.COUNTRY)
ORDER BY CUSTOMERS.COUNTRY,SUPPLIERS.COUNTRY;
```

5.

```
GROUP BY SHIP_COUNTRY,
ROLLUP(SHIP_REGION,SHIP_CITY)
GROUP BY GROUPING SETS(
(SHIP_COUNTRY,SHIP_REGION,SHIP_CITY),
(SHIP_COUNTRY,SHIP_REGION),
SHIP_COUNTRY)
```

6. modify your query from question 2 so that the string 'whole country' is displayed instead of NULL on everyrow that aggregates all cities of a single country.

```
SELECT SHIP_COUNTRY,
DECODE(GROUPING(SHIP_CITY),1,'WHOLE
COUNTRY',SHIP_CITY), COUNT(*) FROM ORDERS
GROUP BY ROLLUP(SHIP_COUNTRY,SHIP_CITY)
ORDER BY SHIP_COUNTRY,SHIP_CITY;
```

Exercice 1.4 :

1/

```
select ship_country, ship_city, count(*) as nborders,
sum(count(*)) over(partition by ship_country) as nbordcty,
max(count(*)) over (partition by ship_country) as
nbormaxcty
from orders
group by ship_country, ship_city
```

2/

```
select ship_country, ship_city, count(*) as nborders,
dense_rank() over (partition by ship_country order by
(count(*)) as rankd
from orders
group by ship_country, ship_city
```

3/

```
select ship_country, ship_city,count(*) as nborders,
dense_rank() over (partition by ship_country order by
(count(*)) as rankd,
count(*)/sum(count(*)) over (partition by ship_country) as
percentage
from orders
group by ship_country, ship_city
```

ou alors pour le pourcentage : ratio_to_report(count(*)) over (partirion by ship_country)

4/

```
WITH TEMP AS(
SELECT ORD.ORDER_ID AS
ORDERID,SUM(ORDDE.UNIT_PRICE*ORDDE.QUANTITY) AS
PRICE,
LAG(SUM(ORDDE.UNIT_PRICE*ORDDE.QUANTITY))OVER(
ORDER BY ORD.ORDER_ID)
AS PRICE_PREV
FROM ORDERS ORD,ORDER_DETAILS ORDDE
WHERE ORD.ORDER_ID=ORDDE.ORDER_ID
GROUP BY ORD.ORDER_ID
ORDER BY ORD.ORDER_ID
)
SELECT ORDERID,PRICE
FROM TEMP
WHERE PRICE<1.1*PRICE_PREV
```

5/

```
with temp as(
select extract(year from od.order_date) as year,
p.product_name as product_name,
sum(odd.quantity) as qtity,
max(sum(odd.quantity)) over(partition by extract(year
from od.order_date)) as maxqt
from orders od, order_details odd, products p
where od.order_id = odd.order_id and p.product_id =
odd.product_id
group by extract(year from od.order_date),
p.product_name
)
select year, product_name,qtity
from temp where qtity = maxqt
order by year DESC
```

Exercice 1.5 : Use a hierarchical query on the DUAL table to create a table listing integers from 1 to 60.

```
WITH test(p) AS
( select 1 p from DUAL union all select p+1 from test where p<60)
select p from test;

WITH
count_to_60 (id) AS
(
SELECT 1 id
FROM DUAL
UNION ALL
SELECT  id+1
FROM count_to_60
WHERE id<60
)
SELECT id
FROM count_to_60
ORDER BY id;
```

Lab. Ex 1.6 Generate the list of the next 30 months (format: MON-YY) starting from today.

```
WITH
months(mois) AS
(
select TO_DATE('10/2016','MM/yyyy') mois
from DUAL
UNION ALL
select ADD_MONTHS(mois,1)
from months
where
mois<ADD_MONTHS(TO_DATE('10/2016','MM/yyyy'),29)
)
select mois
from months
order by mois;
```

LAB EXAM:

Ex 1 (4pt)

number of orders for each combination of employee country, customer country and supplier country, as well as each combination that could be obtained by a rollup to the top level on one or several of these 3 dimensions
(ex: total number of orders, number of orders per employee country and supplier country. . .)

```
select c.country,s.country,e.country,count(distinct od.order_id) as NBOrder
From Order_details od, orders o, Customers c, products p,suppliers s, Employees e
where od.order_id = o.order_id
And o.customer_id = c.customer_id
AND od.product_id = p.product_id
AND p.supplier_id = s.supplier_id
AND o.employee_id = e.employee_id
group by cube(c.country,s.country,e.country)
order by c.country,s.country,e.country
```

Ex 2 (3pt)

number of orders for each combination of employee country, customer country and supplier country, as well as for each employee country. The records corresponding to a total per employee country should display the string "global" in both customer country and supplier country columns.

```
select e.country,
DECODE(GROUPING(s.country),1,'global',s.country)s_country,
DECODE(GROUPING(c.country),1,'global',c.country) c_country,
count(distinct od.order_id) as NBOrder
From Order_details od, orders o, Customers c, products p,suppliers s, Employees e
where od.order_id = o.order_id
And o.customer_id = c.customer_id
AND od.product_id = p.product_id
AND p.supplier_id = s.supplier_id
AND o.employee_id = e.employee_id
group by e.country, rollup(c.country,s.country)
order by e.country,s.country,c.country
```

Ex 3 (3pt)
Same question as Ex 1, but display additionally the rank of each record among all records based on the same combination.

```
select c.country,s.country,e.country,count(distinct od.order_id) as NBOrder,
Dense_Rank() over (partition by c.country,s.country,,e.country order by count(od.order_id)) RK
From Order_details od, orders o, Customers c, products p,suppliers s, Employees e
where od.order_id = o.order_id
And o.customer_id = c.customer_id
AND od.product_id = p.product_id
AND p.supplier_id = s.supplier_id
AND o.employee_id = e.employee_id
group by cube(c.country,s.country,e.country)
order by c.country,s.country,e.country
```

Ex 4 (3pt)
Number of orders per employee. You are not allowed to use any GROUP BY clause (nor an extension of it).

```
SELECT DISTINCT LASTNAME, FIRSTNAME,
COUNT(*) OVER (PARTITION BY LASTNAME) AS NB
FROM EMPLOYEES,ORDERS
WHERE ORDERS.EMPLOYEE_ID = EMPLOYEES.EMPLOYEE_ID;
```

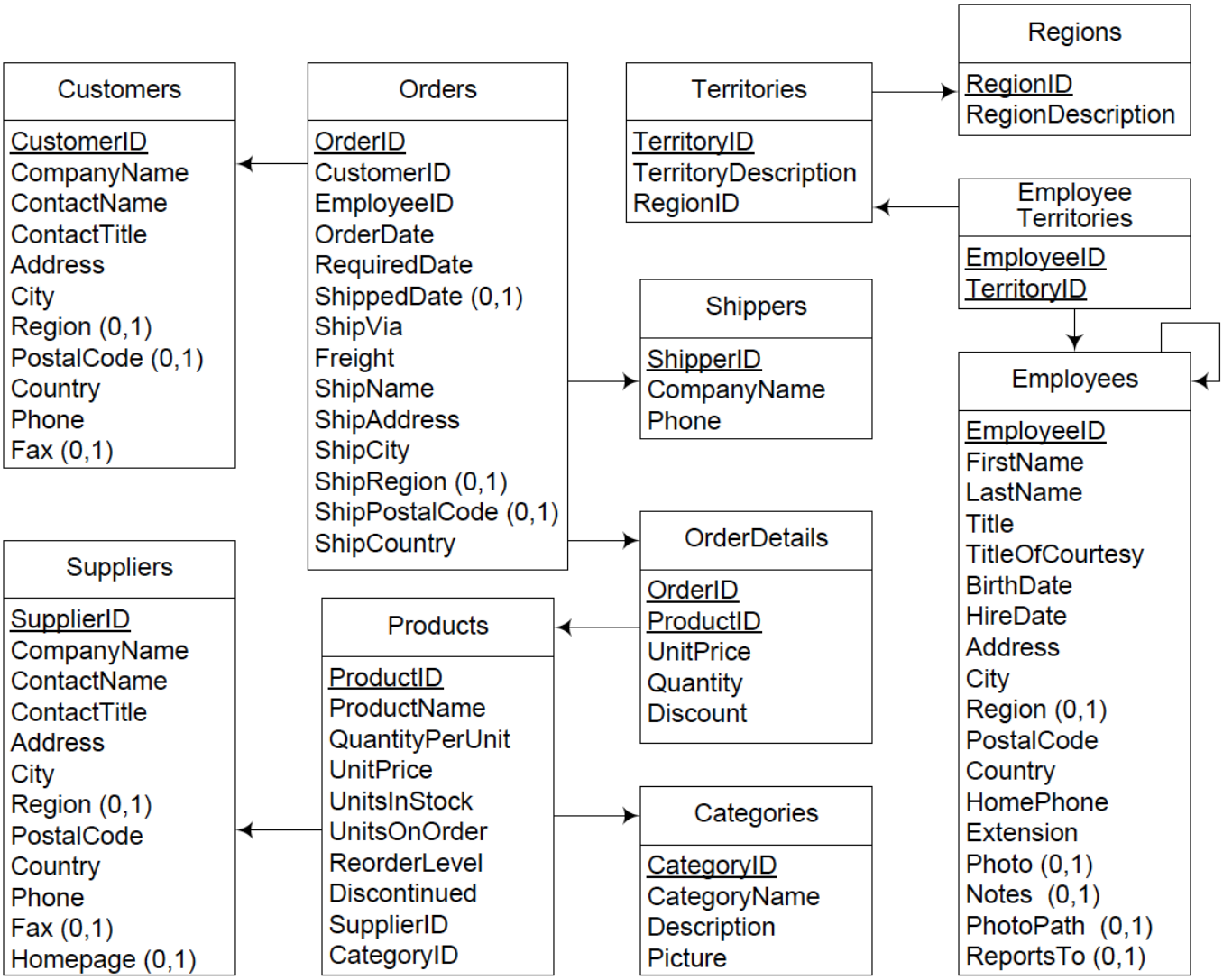

Figure 1: Relational schema for the Northwind database

Dimensions in Oracle SQL: star

Time

date
day_of_week
holiday_flag
calendar_month
calendar_month_number_in_year
calendar_year
fiscal_month
fiscal_year

Dimension definition

```
CREATE DIMENSION date_dim
  LEVEL day IS time.date
  LEVEL month IS time.calendar_month
  LEVEL year IS time.calendar_year
  LEVEL fis_month IS time.fiscal_month
  LEVEL fis_year IS time.fiscal_year
  HIERARCHY cal_rollup (
    day CHILD OF
    month CHILD OF
    year
  )
  HIERARCHY fis_rollup (
    day CHILD OF
    fis_month CHILD OF
    fis_quarter
  )
  ATTRIBUTE day DETERMINES
    (day_of_week, holiday_flag)
  ATTRIBUTE month DETERMINES
    (calendar_month_number_in_year)
;
```


Dimensions in Oracle SQL: snowflake

Products

prod_key (PK)
prod_name
prod_desc
category_key (FK)

Categories

cat_key (PK)
cat_name
cat_desc

Dimension definition

```
CREATE DIMENSION products_dim
  LEVEL product IS products.prod_key
  LEVEL category IS categories.cat_key
  HIERARCHY prod_rollup (
    product CHILD OF
    category
  JOIN KEY (product.category_key) REFERENCES category
  )
  ATTRIBUTE product DETERMINES
    (prod_name, prod_desc)
  ATTRIBUTE category DETERMINES
    (cat_name,cat_desc)
;
```


Dimensions in Oracle SQL: snowflake (2)

Products

prod_key (PK)
prod_name
prod_desc
subcategory_key (FK)
undividedcategory_key

SubCategories

subcat_key (PK)
subcat_name
category_key

Categories

cat_key (PK)
cat_name
cat_desc

Dimension definition (Oracle)

```
CREATE DIMENSION products_dim
  LEVEL product IS products.prod_key
  LEVEL subcategory IS subcategories.subcat_key SKIP WHEN NULL
  LEVEL category IS categories.cat_key
  HIERARCHY prod_rollup (
    product CHILD OF
    category
  JOIN KEY (products.subcategory_key) REFERENCES subcategory
  JOIN KEY (subcategories.category_key) REFERENCES category
  JOIN KEY (product.category_key) REFERENCES category
  )
;
```


Dimensions in Oracle SQL

Modifying dimensions (Oracle)

```
ALTER DIMENSION products_dim DROP HIERARCHY prod_rollup;
ALTER DIMENSION products_dim ADD LEVEL brand IS categories.brand;
ALTER DIMENSION products_dim COMPILE; -- to re-validate
DROP DIMENSION products_dim;
...
```

Dimensions in Oracle SQL: usage

Dimensions are viewed in Oracle as constraints, used for query rewriting.

Validating dimensions

```
BEGIN
  DBMS_DIMENSION.VALIDATE_DIMENSION(
  dimension => 'products_dim',
  incremental => false,
  check_nulls => true,
  statement_id=>'validation run 1');
END;
/
```

- incremental=true: validate new rows only
- check_nulls=true: NULLS occur only in columns with level 'SKIP WHEN NULL'
- statement_id: used to distinguish the rows created in table `dimension_exceptions`

Dimensions in Oracle SQL: usage

Viewing dimension information

```
EXECUTE DBMS_DIMENSION.DESCRIBE_DIMENSION('products_dim');
```