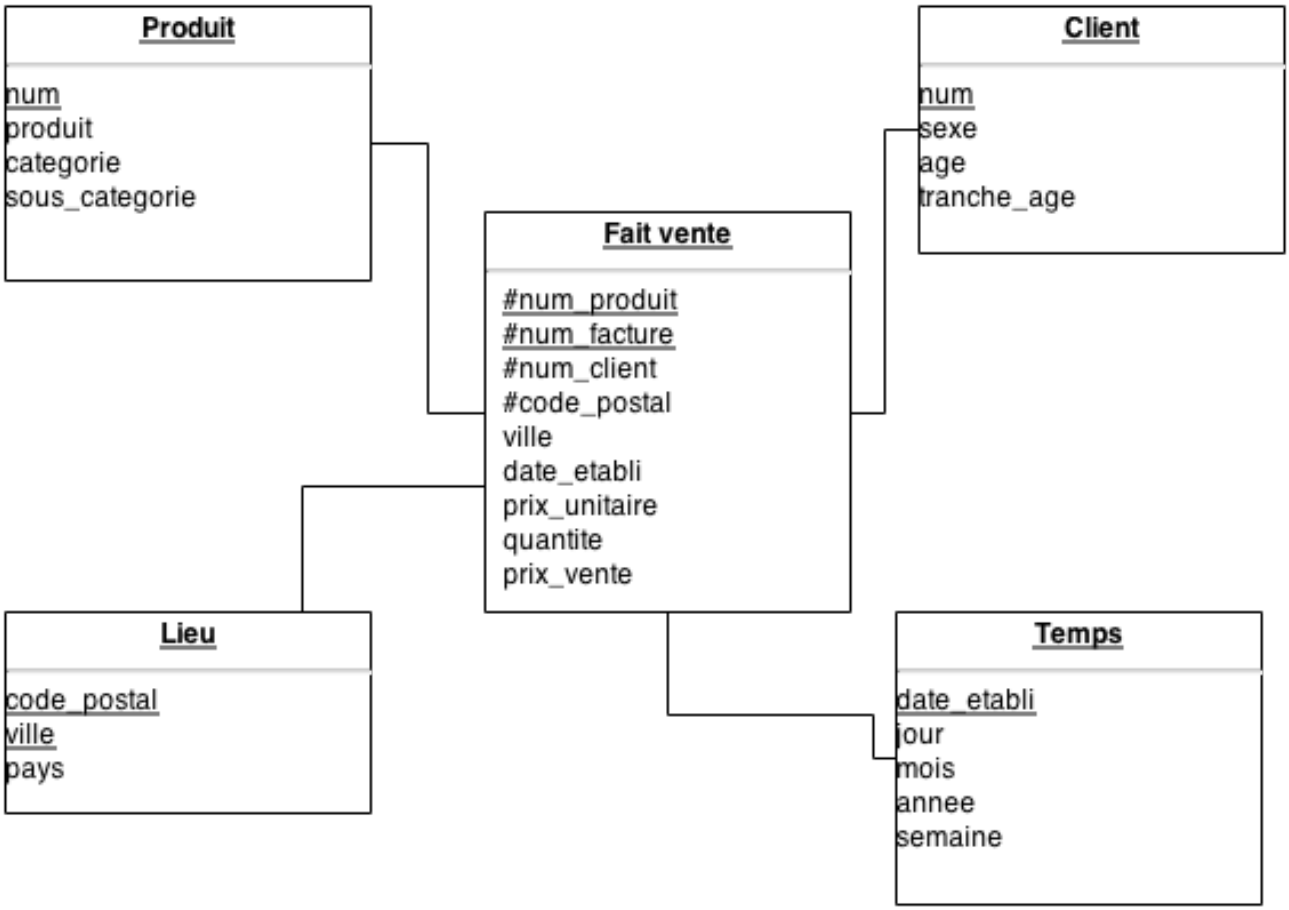
Delobelle Anthony
Delporte Jérôme
Marquette Camille

TP2 Entrepôt de données

Modélisation dimensionnelle



Implémentation et alimentation

1- Requetes SQL Creation Vues :

```
Create materialized view produit_vm
Build immediate
Refresh force
Enable query rewrite as
select num, regexp_substr(designation, '[^.]*', 1, 1) as produit, regexp_substr(designation, '[^.]*', 1,
3) as categorie, regexp_substr(designation, '[^.]*', 1, 5) as sous_categorie
from produit;
```

```
Create materialized view client_vm
Build immediate
Refresh force as
select num,
sexe,
floor(months_between(SYSDATE, date_nais)/12) as age,
Case
when floor(months_between(SYSDATE, date_nais)/12) <30 then '<30'
when floor(months_between(SYSDATE, date_nais)/12) <46 then '30-45'
when floor(months_between(SYSDATE, date_nais)/12) <61 then '46-60'
else '>60'
end as tranche_age
from client;
```

```
Create materialized view lieu_vm
Build immediate
Refresh force as
select  distinct regexp_substr(adresse, '[^,]*', 1, 3) as code_postal, regexp_substr(adresse, '[^,]*', 1,
5) as ville , regexp_substr(adresse, '[^,]*', 1, 7) as pays
from client;
```

```
Create materialized view temps_vm
Build immediate
Refresh force as
select distinct date_etabli, extract(day FROM date_etabli) as jour , extract(month FROM
date_etabli) as mois , extract(year FROM date_etabli) as annee,
to_number(to_char( date_etabli,'IW')) as semaine from facture;
```

```
Create materialized view vente_vm
Build immediate
Refresh force as
select produits.pid as num_produit, client.num as num_client, regexp_substr(adresse, '[^,]*', 1, 3) as
code_postal, regexp_substr(adresse, '[^,]*', 1, 5) as ville, facture.date_etabli as date_vente,
facture.num as num_facture, prix_date.prix as prix_unitaire, ligne_facture.qte as quantite,
ligne_facture.qte * prix_date.prix as prix_vente
from produits
join ligne_facture on produits.pid = ligne_facture.produit
join facture on ligne_facture.facture = facture.num
join client on facture.client = client.num
join prix_date on prix_date.produit = produits.pid;
```

2- Requête création clé primaires / clé étrangères :

Clé primaire pour produit déjà généré par oracle

Pour client :

```
ALTER materialized view Client_vm
ADD CONSTRAINT pk_view_client_ID PRIMARY KEY (num);
```

Pour lieu :

```
ALTER materialized view lieu_vm
ADD CONSTRAINT pk_view_lieu_ID PRIMARY KEY (code_postal, ville);
```

Pour temps :

```
ALTER materialized view temps_vm
ADD CONSTRAINT pk_view_temps_ID PRIMARY KEY (date_etabli);
```

Pour produit :

```
ALTER materialized view vente_vm
ADD CONSTRAINT pk_view_vente_ID PRIMARY KEY (num_produit, num_facture);

ALTER materialized view vente_vm
ADD CONSTRAINT pk_view_vente_lieu_fk Foreign KEY (code_postal, ville) references
lieu_vm;

ALTER materialized view vente_vm
ADD CONSTRAINT pk_view_vente_client_fk Foreign KEY (num_client) references client_vm;

ALTER materialized view vente_vm
ADD CONSTRAINT pk_view_vente_temps_fk Foreign KEY (date_vente) references temps_vm;

ALTER materialized view vente_vm
ADD CONSTRAINT pk_view_vente_produit_fk Foreign KEY (num_produit) references
produit_vm;
```

4- Creation des index bitmap :

```
CREATE BITMAP INDEX IX_produit_categorie
ON produit_vm (categorie);

CREATE BITMAP INDEX IX_produit_sscategorie
ON produit_vm (sous_categorie);

CREATE BITMAP INDEX IX_client_tranche
ON client_vm (tranche_age);

CREATE BITMAP INDEX IX_temps_annee
ON temps_vm (annee);
```

5- Déclaration des dimensions :

```
CREATE DIMENSION produits_dim
LEVEL produit IS(produit_vm.num)
LEVEL sous_categorie IS (produit_vm.sous_categorie) SKIP WHEN NULL
LEVEL categorie IS (produit_vm.categorie)
HIERARCHY prod_rollup (
produit CHILD OF sous_categorie
CHILD OF categorie);

CREATE DIMENSION age_dim
LEVEL age IS(client_vm.age)
LEVEL tranche_age IS (client_vm.tranche_age)
HIERARCHY prod_rollup (
age CHILD OF tranche_age);
```

Questions d'exploitation – Requêtes SQL

1- Chiffre d'affaire par produit:

```
select produit_vm.PRODUIT, sum(vente_vm.PRIX_VENTE) as CA
from vente_vm join produit_vm
on vente_vm.NUM_PRODUIT = produit_vm.NUM
group by produit_vm.PRODUIT
```

2- Chiffre d'affaire par catégorie et mois:

```
select produit_vm.CATEGORIE, temps_vm.MOIS, sum(vente_vm.PRIX_VENTE) as CA
from vente_vm join produit_vm
on vente_vm.NUM_PRODUIT = produit_vm.NUM
join temps_vm
on vente_vm.DATE_VENTE = temps_vm.DATE_ETABLI
group by produit_vm.CATEGORIE, temps_vm.MOIS
order by produit_vm.CATEGORIE
```

Chiffre d'affaire par catégorie:

```
select produit_vm.CATEGORIE, sum(vente_vm.PRIX_VENTE) as CA
from vente_vm join produit_vm
on vente_vm.NUM_PRODUIT = produit_vm.NUM
join temps_vm
on vente_vm.DATE_VENTE = temps_vm.DATE_ETABLI
group by produit_vm.CATEGORIE
order by produit_vm.CATEGORIE
```

Chiffred'affaire global:

```
select sum(vente_vm.PRIX_VENTE) as CA_global
from vente_vm
```

4- Chiffre d'affaire par tranche d'âge, en donnant le rang de chaque tranche:

```
select client_vm.TRANCHE_AGE, sum(vente_vm.PRIX_VENTE) as CA,
rank() over (order by sum(vente_vm.PRIX_VENTE) desc)as rang
from vente_vm join client_vm
on vente_vm.NUM_CLIENT = client_vm.NUM
group by client_vm.TRANCHE_AGE
order by rang
```

5- Les 3 produits les plus vendus en quantité:

```
select produit, quantite_totale from(
select produit_vm.PRODUIT, sum(vente_vm.QUANTITE) as quantite_totale, rank() over ( order
by sum(vente_vm.QUANTITE) desc) as rang
from vente_vm join produit_vm
on vente_vm.NUM_PRODUIT = produit_vm.NUM
group by produit_vm.PRODUIT
order by quantite_totale desc)
where rang < 4
```