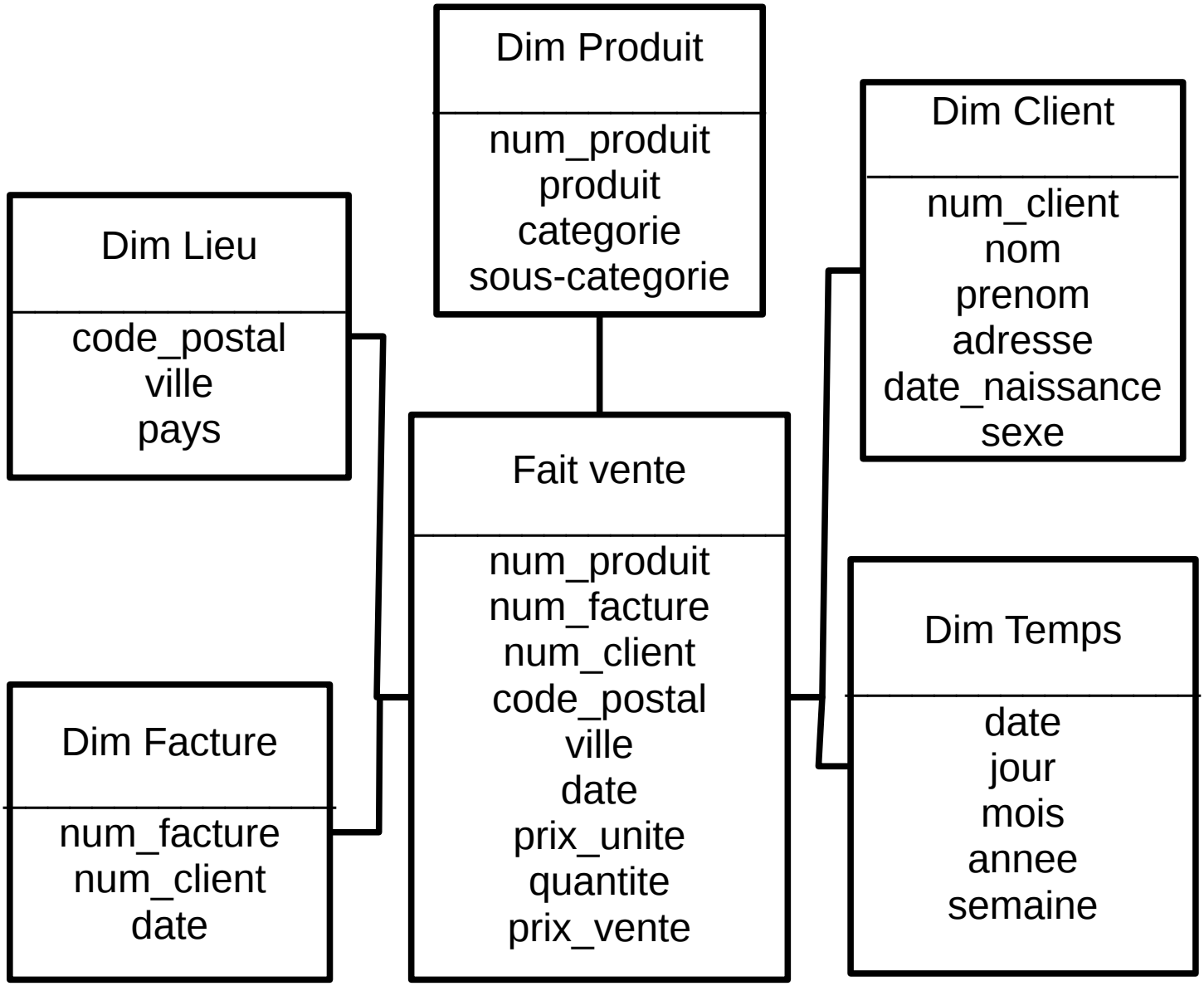
20 février 2015
Emilie Allart
tp2 : SQL / OLAP

***** Modélisation dimensionnelle *****



***** Implémentation et exploitation *****

**** Questions d'implémentation ****

1/Creation de 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 ans'
               WHEN floor(months_between(SYSDATE, date_nais)/12)
<46 then '30-45 ans'
               WHEN floor(months_between(SYSDATE, date_nais)/12)
<61 then '46-60 ans'
               ELSE '>60 ans'
          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 des clés primaires et étrangères :

```
ALTER MATERIALIZED VIEW Client_vm
     ADD CONSTRAINT pk_view_client_ID PRIMARY KEY (num);

ALTER MATERIALIZED VIEW lieu_vm
     ADD CONSTRAINT pk_view_lieu_ID PRIMARY KEY (code_postal,
ville);

ALTER MATERIALIZED VIEW temps_vm
     ADD CONSTRAINT pk_view_temps_ID PRIMARY KEY (date_etabli);

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;
```

3/ Vérification des mise à jour des vues

Ok

4/ Création des index

```
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);

TODO : autres dim et verifications
```

** Questions d'exploitation **

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
```

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
```

```
     global:
SELECT sum(vente_vm.PRIX_VENTE) AS CA_global
     FROM vente_vm
```

3/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
```

4/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
```