- **L3 MIASHS/Ingémath/METIS**
- **Université Paris Cité**
- Année 2024-2025
- Course Homepage

- Moodle

Raoul Taburin, vélociste (marchand-réparateur de bicyclettes), a décidé de s'équiper d'un système d'information. Il a fait réaliser une analyse par un cabinet de conseil réputé.
Le vélociste vend des produits (dérailleurs, jantes, freins, selles, guidons, …). Chaque produit est identifié par un numéro de catalogue (`product_id`). Un produit est vendu par un fabricant connu par son nom (Campagnolo, Shimano, Simplex,…). Un produit possède une description (texte). Un produit relève d'une catégorie identifiée par un numéro (`cat_id`) et munie d'une description (pédalier, freins à | disque, tige de selle télescopique, …). Un produit relève aussi d'une gamme (Ultegra, 105, Tiagra, …).
Un produit possède un prix unitaire.
À chaque produit du catalogue correspond une entrée dans l'inventaire. Dans cette entrée on reporte le numéro du produit, le nombre d'exemplaires en stock (`en_stock`) et le nombre d'exemplaires déjà vendus (`vendus`).

Pour chaque produit, le vélociste est amené à effectuer des réapprovisionnements (commandes). Chaque commande concerne un produit, elle est efféctuée à une date notée `date_com`. La commande porte sur une quantité notée `qte`. La commande est livrée au vélociste (si tout se passe bien) à la date `date_liv`. Pour une commande qui n'a pas encore été livrée, `date_liv` est réputée `NULL`.

Le vélociste possède des fidèles clients. Chaque fidèle client est identifié par un numéro (`client_id`), possède un nom, une adresse (texte), un numéro de téléphone. Sur chaque client, le vélociste possède des renseignements démographiques (année de naissance, sexe, profession, taille).

Lorsqu'un fidèle client effectue un achat, le vélociste émet une facture identifiée par un numéro de facture `fact_id`. La facture comporte le numéro du client, une date et un montant global (`montant`).

Une facture se compose de lignes. Chaque ligne comporte une référence à un produit désigné par son numéro de catalogue, une quantité (`qte`) et un montant (calculé à partir de `qte` et du prix unitaire du produit). Chaque ligne de facture est identifiée par un numéro de ligne `num_ligne` (relativement à la facture).

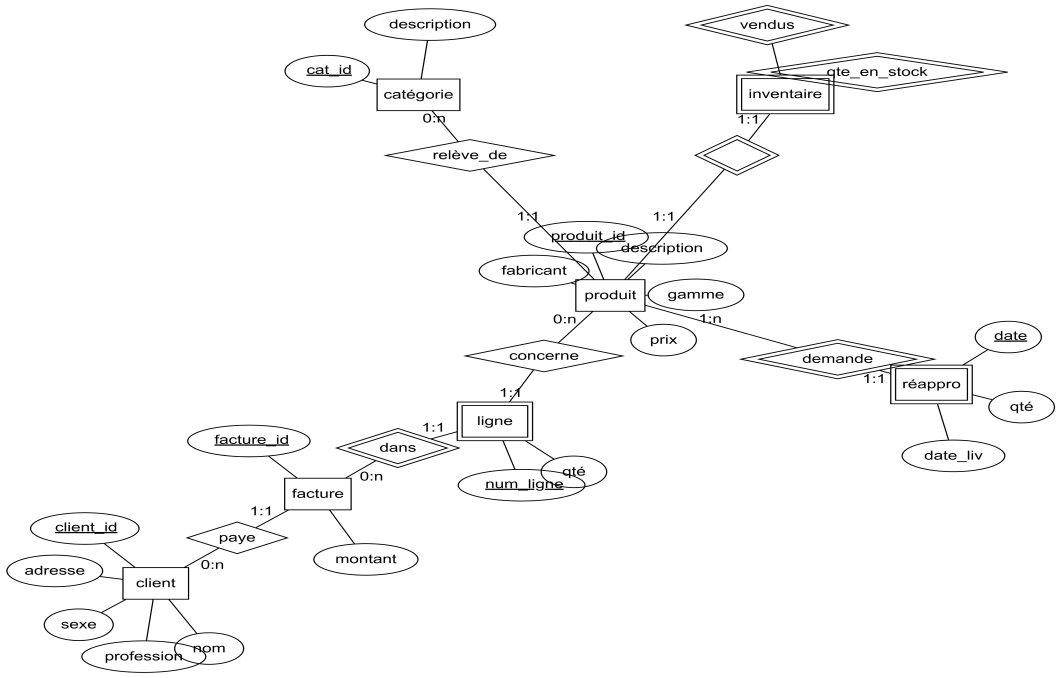
Question 1 (4pts)

Proposer un diagrammme entité-association (EA) correspondant à cette modélisation.

Conseils

- Distinguer entités fortes et faibles
- Distinguer associations fortes et faibles
- Pour chaque entité préciser l'identifiant (éventuellement relatif)
- Préciser les cardinalités pour chaque participation à une association

Solution



Remarque

- Nous n'avons pas décrit tous les attributs, pour ne pas surcharger un schéma déjà chargé.
- Définir `inventaire` comme une entité faible subordonnée à l'entité `produit` est un choix discutable. Comme l'inventaire n'est pas historicisé (c'est l'inventaire à l'instant courant), on peut aussi bien ajouter à l'entité `produit` les attributs de l'entité `inventaire`.

- Un réapprovisionnement est identifié par le produit et la date de commande. Il s'agit d'une entité faible associée à `produit` par une association faible `demande`

- `ligne` est une entitée faible identifiée par l'identifiant relatif `num_ligne` et l'identifiant de l'entité forte associée `facture`
- L'association entre `ligne` et `produit` n'est pas identifiante.

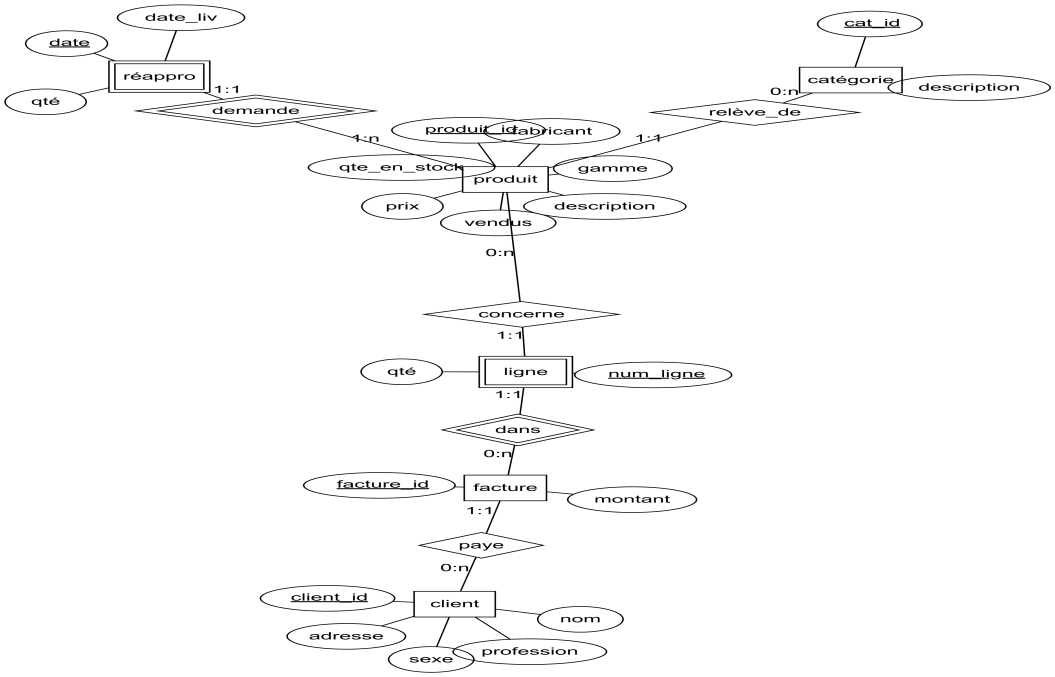
Avec Chat-GPT 3.5

Voici une proposition pour un schéma entité-association (EA) basé sur votre description du système du vélociste :
La notation des cardinalités n'est pas exactement celle du cours. Les entités et associations faibles ne sont pas mises en évidence, mais le gros, l'essentiel du travail est effectué.

1. **Produit**
   - Attributs : `product_id` (PK), `nom_produit`, `description`, `cat_id`, `nom_gamme`, `prix_unitaire`
   - Association :
     - Relié à `Inventaire` (1,1) → (`product_id`)
     - Relié à `Commande` (1,n) → (`product_id`)
     - Relié à `LigneFacture` (1,n) → (`product_id`)
2. **Catégorie**
   - Attributs : `cat_id` (PK), `description`
3. **Inventaire**
   - Attributs : `product_id` (FK, PK), `en_stock`, `vendus`
   - Association : Relié à `Produit` (1,1)
4. **Commande**
   - Attributs : `commande_id` (PK), `product_id` (FK), `date_com`, `qte`, `date_liv`
   - Association : Relié à `Produit` (n,1)
5. **Client**
   - Attributs : `client_id` (PK), `nom`, `adresse`, `telephone`, `annee_naissance`, `sexe`, `profession`, `taille`
   - Association :
     - Relié à `Facture` (1,n)
6. **Facture**
   - Attributs : `fact_id` (PK), `client_id` (FK), `date_fact`, `montant`
   - Association :
     - Relié à `Client` (n,1)
     - Relié à `LigneFacture` (1,n)
7. **LigneFacture**
   - Attributs : `fact_id` (FK), `num_ligne` (PK relatif), `product_id` (FK), `qte`, `montant`
   - Association :
     - Relié à `Facture` (n,1)
     - Relié à `Produit` (n,1)

Associations principales

- Chaque **produit** appartient à une **catégorie**.
- Un produit a des entrées associées dans l'**inventaire** et des commandes de réapprovisionnement.
- Les **clients** peuvent avoir plusieurs **factures**, chaque facture pouvant contenir plusieurs lignes relatives à un ou plusieurs produits.



Question 2 (2pts)

Si besoin, transformez votre schéme de façon à ce qu'il ne comporte plus d'associations d'arité supérieure ou égale à 3, ou d'associations plusieurs-plusieurs (0 :n pour plusieurs entités participantes).

Notre schéma ne comporte ni associations plusieurs-plusieurs, ni association d'arité supérieure à 2. Il n'y a donc pas de réécriture à effectuer. Nous pouvons immédiatement proécéder à la traduction en pattes de corbeau.

Question 3 (2pts)

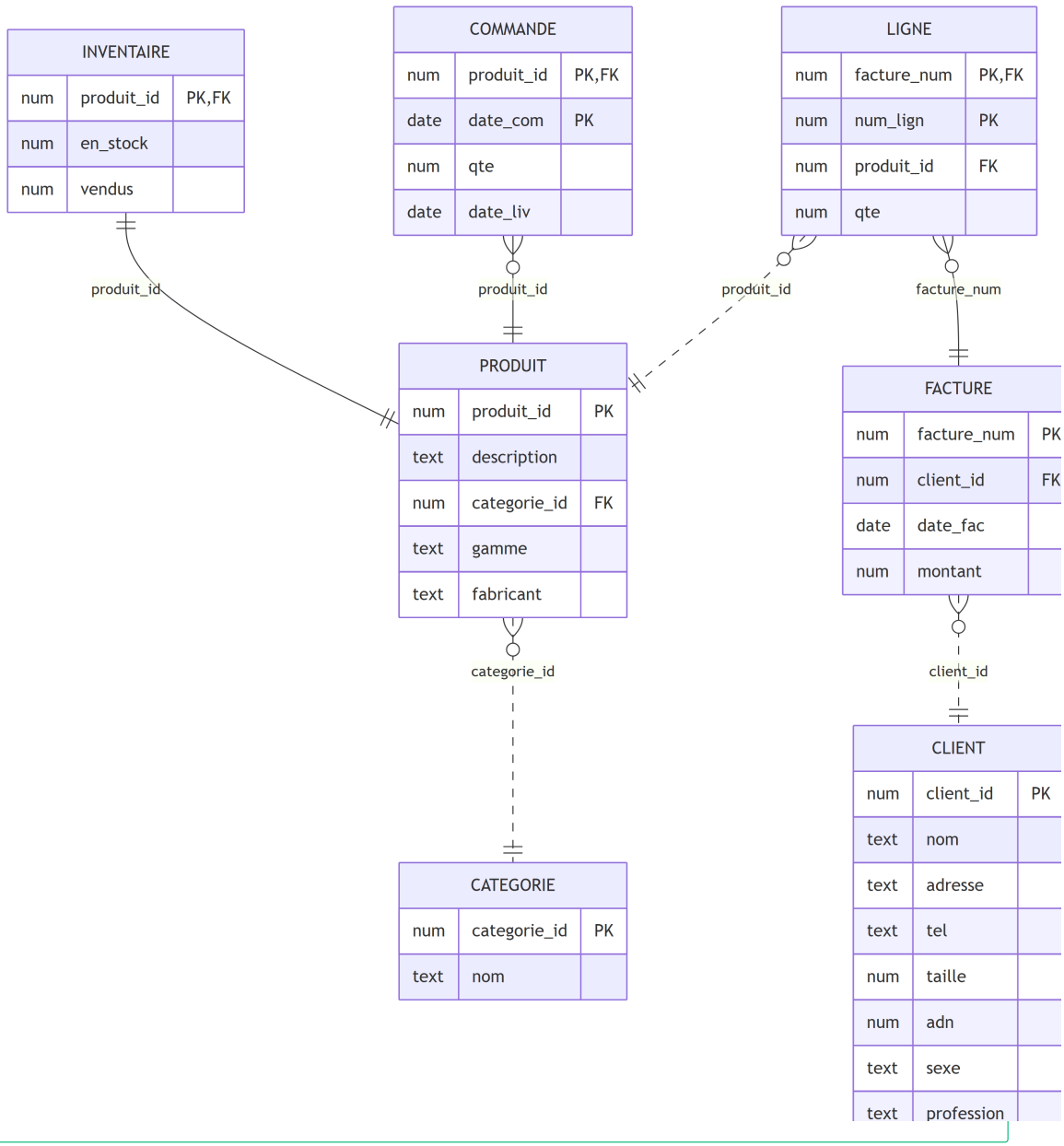
Proposer une traduction en pattes de corbeau du diagramme EA proposé en réponse à la première question.

Conseil

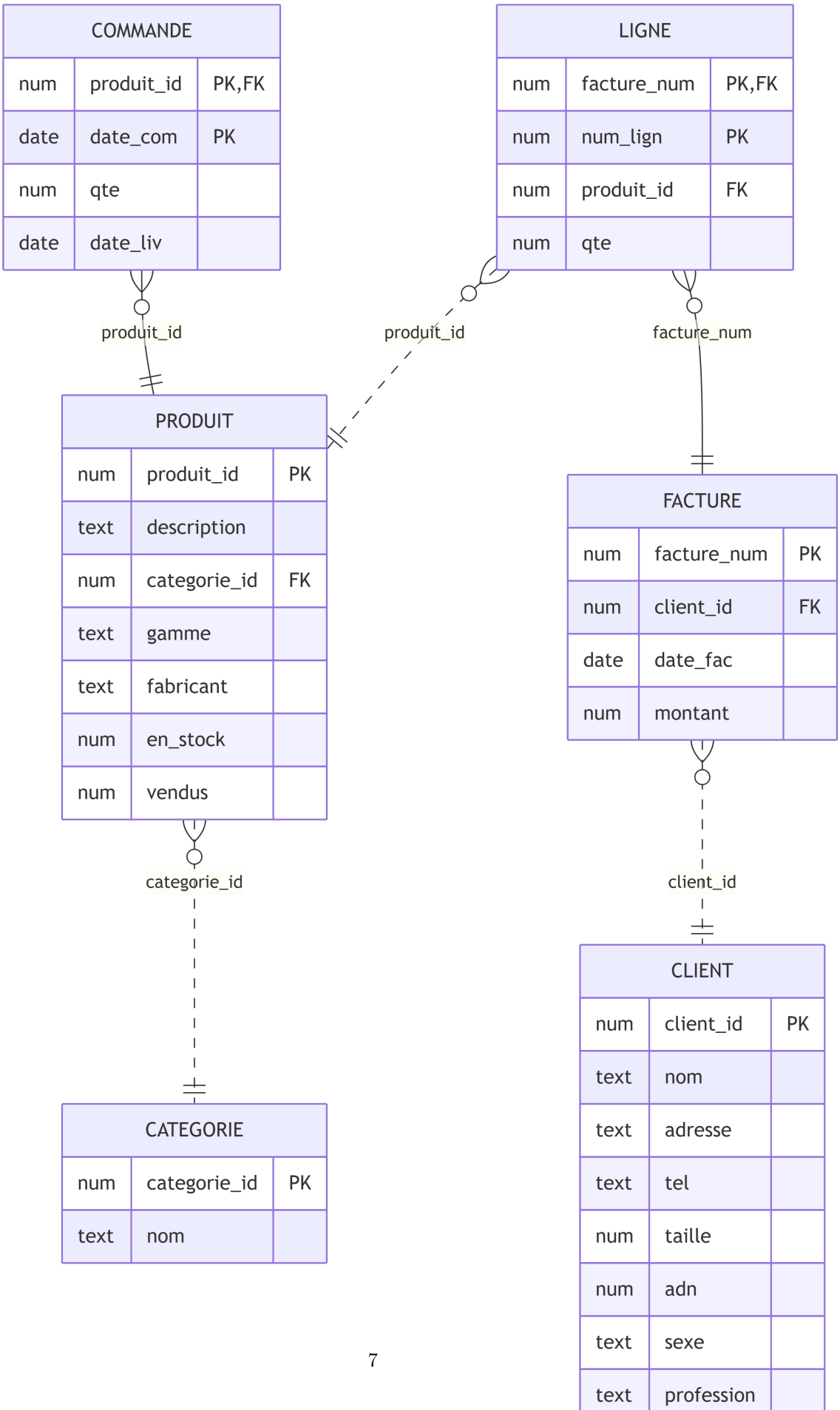
Aidez-vous aussi de votre réponse à la deuxième question.
Précisez

- une clé primaire pour chaque table,
- les tables dites intermédiaires,
- pour les liens matérialisant les contraintes référentielles, préciser s'ils sont identifiant ou non.

Solution

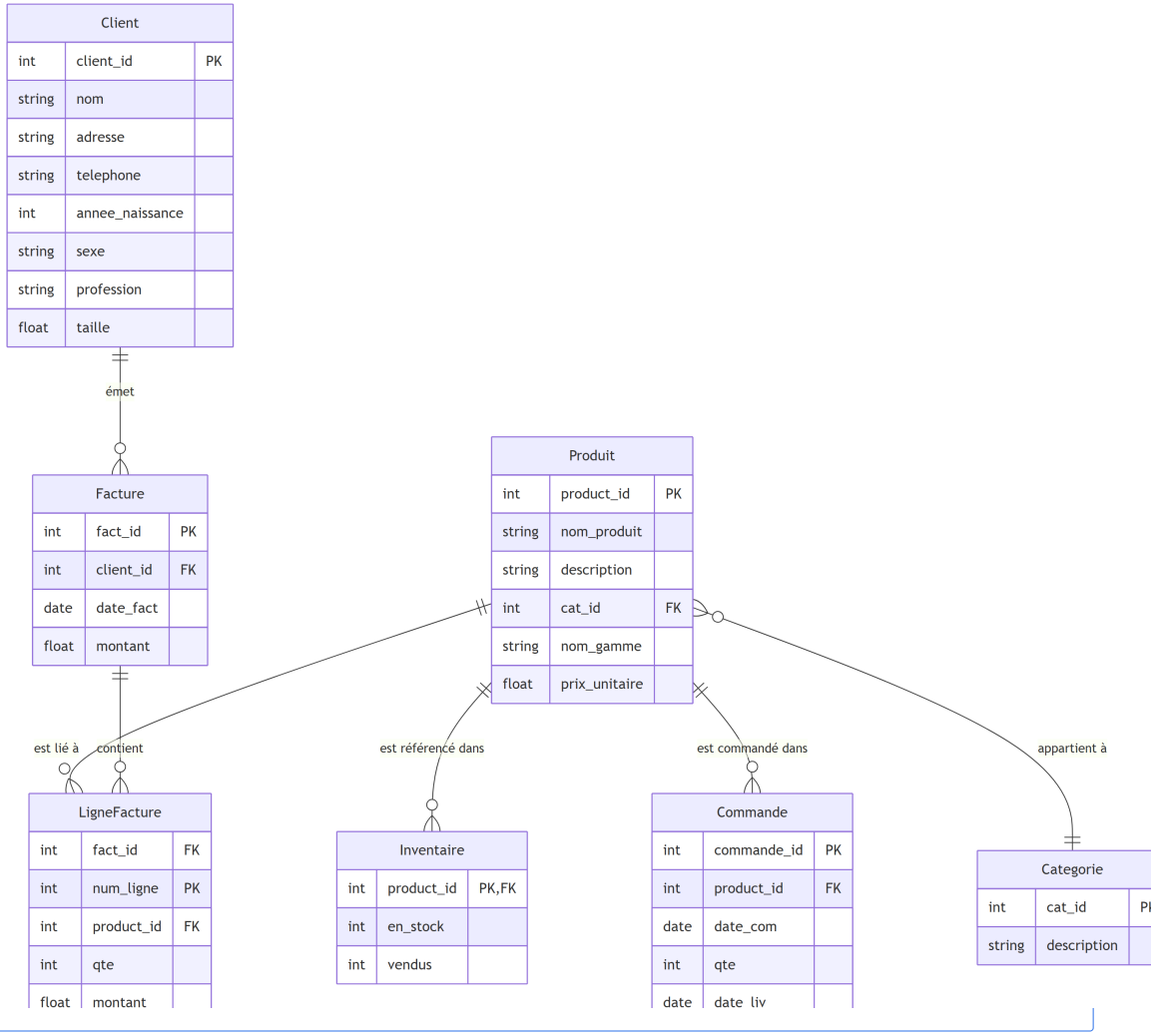

INVENTAIRE
num produit_id PK,FK
num en_stock
num vendus
COMMANDE
num produit_id PK,FK
date date_com PK
num qte
date date_liv
LIGNE
num facture_num PK,FK
num num_lign PK
num produit_id FK
num qte
produit_id
produit_id
produit_id
facture_num
PRODUIT
num produit_id PK
text description
num categorie_id FK
text gamme
text fabricant
FACTURE
num facture_num PK
num client_id FK
date date_fac
num montant
categorie_id
client_id
CLIENT
num client_id PK
text nom
text adresse
text tel
num taille
num adn
text sexe
text profession
CATEGORIE
num categorie_id PK
text nom

Remarque


COMMANDE
num produit_id PK,FK
date date_com PK
num qte
date date_liv
LIGNE
num facture_num PK,FK
num num_lign PK
num produit_id FK
num qte
produit_id
produit_id
facture_num
PRODUIT
num produit_id PK
text description
num categorie_id FK
text gamme
text fabricant
num en_stock
num vendus
FACTURE
num facture_num PK
num client_id FK
date date_fac
num montant
categorie_id
client_id
CATEGORIE
num categorie_id PK
text nom
CLIENT
num client_id PK
text nom
text adresse
text tel
num taille
num adn
text sexe
text profession

Proposition Chat-GPT 3.5

Le lien entre `inventaire` et `produit` n'est pas décoré correctement, car il n'y a pas d'historicisation. Les liens sont tous présentés comme identifiants. Cela ne devrait pas être le cas. En dehors de cela c'est bon.



Le vélociste a explicité les contraintes suivantes :

- Pour un même produit, les intervalles de temps `[date_com, date_liv)` correspondant à deux commandes différentes ne peuvent se recouvrir,
- Une gamme de produits appartient à un seul fabricant (Ultegra est une gamme de Shimano, Campagnolo et autres ne peuvent pas utiliser ce nom),
- Dans une gamme donnée, un fabricant propose au plus un produit de catégorie donnée.

Question 4 (1pt)

- Préciser parmi ces contraintes, celles qui sont des dépendances fonctionnelles
- Proposer un mécanisme pour mettre en place ces contraintes en SQL (langage de définition de données)

Solution

- La contrainte de recouvrement est une contrainte d'exclusion (`EXCLUDE`). Ce n'est pas un dépendance fonctionnelle.

```
ALTER TABLE reappro
  ADD CONSTRAINT xcl_prod
  EXCLUDE USING gist (
    product_id WITH =,
    (date, date_liv) WITH &&
  ) ;
```

- Le fait qu'une gamme de produits appartient à un seul fabricant, définit une dépendance fonctionnelle `gamme` → `fabricant`. Cette contrainte peut aussi s'exprimer à l'aide d'une contrainte `EXCLUDE`

```
ALTER TABLE produit
  ADD CONSTRAINT xcl_gamme_fabriquant
  EXCLUDE USING gist (
    gamme WITH =,
    fabriquant WITH <>
  ) ;
```

- Le fait que pour une gamme, un fabricant ne propose au plus un produit d'une catégorie donnée définit une contrainte `gamme`, `fabricant`, `cat_id` → `produit_id`. Cette contrainte est une dépendance fonctionnelle. Comme on a par ailleurs `gamme` → `fabricant`, on peut la simplifier en `gamme`, `cat_id` → `produit_id`.

Là encore on peut utiliser la construction

```
ALTER TABLE produit
  ADD CONSTRAINT xcl_gamme_cat_prod
  EXCLUDE USING gist (
    gamme WITH =,
    cat_id WITH =,
    produit_id WITH <>
  ) ;
```

On suppose que le schéma est muni des dépendances fonctionnelles déduites de la question précédente et de celles qui se déduisent des contraintes de clé primaire. On note cet ensemble de dépendances fonctionnelles Σ.

Question 5 (2pts)

- Préciser pour chaque table si elle est en FNBC par rapport à Σ
- Si un ou plusieurs tables ne sont pas en FNBC, proposer une décomposition sans perte d'information (SPI) telle que toutes les tables soient en FNBC.

Solution

Les tables qui ne possèdent pas de DF en dehors de celle impliquées par la donnée de la clé primaire sont en FNBC.
La table `produit` n'est pas en FNBC : le déterminant de `gamme` → `fabricant` n'est pas une super-clé. Les clés de `produit` sont `product_id`, et `gamme, cat_id`.

Question 6 (2pts)

Soit le schéma $\mathcal{A} = \{\texttt{A, B,C, D, E, G, H}\}$.
Soit $\Sigma = \{\{\texttt{D,E}\} \longrightarrow \{\texttt{F}\}, \{\texttt{H}\} \longrightarrow \{\texttt{B}\}, \{\texttt{B,C}\} \longrightarrow \{\texttt{D}\}, \{\texttt{C}\} \longrightarrow \{\texttt{E}\}, \{\texttt{D,F}\} \longrightarrow \{\texttt{H,A}\}\}$ un ensemble de dépendances fonctionnelles.
Est-ce que les dépendances fonctionnelles $\{\texttt{B,C}\} \longrightarrow \{\texttt{F}\}$, $\{\texttt{C,H}\} \longrightarrow \{\texttt{D}\}$, $\{\texttt{B,C}\} \longrightarrow \{\texttt{G}\}$ sont impliquées par Σ ? Autrement dit, a-t-on :

- $\Sigma \models \{\texttt{B,C}\} \longrightarrow \{\texttt{F}\}$,
- $\Sigma \models \{\texttt{C,H}\} \longrightarrow \{\texttt{D}\}$,
- $\Sigma \models \{\texttt{B,C}\} \longrightarrow \{\texttt{G}\}$?

Solution

Pour répondre aux trois questions, suffit de vérifier si `F` et/ou `G` appartiennent à $[\{\texttt{B,C}\}]^+_\Sigma$, et si D appartient à $[\{\texttt{C,H}\}]^+_\Sigma$

$$[\{\texttt{B,C}\}]^+_\Sigma = \{\texttt{B, C, D, E, F, H, A}\}$$

$$[\{\texttt{C,H}\}]^+_\Sigma = \{\texttt{C, H, B, E, D, F, A}\}$$

Les réponses sont

- $\Sigma \models \{\texttt{B,C}\} \longrightarrow \{\texttt{F}\}$,
- $\Sigma \models \{\texttt{C,H}\} \longrightarrow \{\texttt{D}\}$,
- $\Sigma \not\models \{\texttt{B,C}\} \longrightarrow \{\texttt{G}\}$?

Attention

Dans la suite, vous formulerez les requêtes dans le schéma relationnel défini par votre schéma en pattes de corbeau.
🎓 : 1 point par requête

Requête 1

Lister pour chaque fabricant, chaque gamme, le nombre de produits proposés au catalogue.

Solution

```
SELECT
  fabricant, gamme, COUNT(produit_id) AS n_produit
FROM
  produit
GROUP BY
  fabricant, gamme ;
```

Requête 2

Lister pour chaque client, la somme des montants versés par ce client.

Solution

```
SELECT
  client_id, SUM(montant) AS somme_montants
FROM
  facture
WHERE
  montant IS NOT NULL
GROUP BY
  client_id ;
```

Requête 3

On cherche à détecter s'il existe des commandes de réapprovisionnement qui concernent un même produit et dont les intervalles de temps (`[date_com, date_liv)`) se chevauchent. Écrire une requête qui liste les paire de commandes qui posent problèmes. La requête donnera les numéros de commande, le produit concerné, et les dates de commande.

Solution

```
SELECT
  r1.produit_id,
  r1.date, r1.date_liv,
  r2.date, r2.date_liv
FROM
  reappro r1 JOIN
  reappro r2 USING(produit_id)
WHERE
  r1.date < r2.date     -- les deux commandes r1 et r2 sont distinctes ...
AND
  (r1.date, r1.date_liv) OVERLAPS
  (r2.date, r2.date_liv)
;
```

Il ne faut pas oublier la condition `r1.date < r2.date` pour

Requête 4

Lister les factures pour lesquelles on ne trouve aucune ligne de facture.

Solution

```
SELECT DISTINCT
  facture_id
FROM
  facture LEFT OUTER JOIN
  ligne USING (facture_id)
WHERE
  num_ligne IS NULL;
```

Remarque

On pourrait aussi utiliser `EXCEPT` et faire la différence entre la projection de `facture` sur `facture_id` et la projection de `ligne` sur `facture_id`.
En revanche

```
SELECT
  facture_id
FROM
  ligne
GROUP BY
  facture_id
HAVING COUNT(num_ligne) == 0 ;
```

n'est pas une réponse correcte. Cette requête renvoie toujours un résultat vide.

Requête 5

Pour chaque mois, lister la catégorie de produits la plus vendue (en nombre d'articles).

Solution

```
WITH R AS (
  SELECT
    EXTRACT(MONTH FROM fa.date) AS mois, pr.cat_id,
    SUM(li.qte) AS qte_mois
  FROM
      facture fa
    JOIN
      ligne li USING (fact_id)
    JOIN
      produit pr USING (produit_id)
  GROUP BY
    EXTRACT(MONTH FROM fa.date), pr.cat_id
)

SELECT
  r1.mois, r2.cat_id
FROM
  R AS r1
WHERE
  r1.qte >= ALL (
    SELECT
      r2.qte
    FROM
      R AS r2
    WHERE
      r2.mois = r1.mois
  )
;
```

Requête 6

Lister les commandes de réapprovisionnement en cours.

Solution

```
SELECT
  produit_id, date, qte
FROM
  reappro
WHERE
  date_liv IS NULL ;
```

Requête 7

Lister les produits les plus vendus et les moins vendus dans chaque catégorie.

Solution

```
WITH R AS (
  SELECT
    pr.cat_id, pr.produit_id, SUM(li.qte) as tot_ventes
  FROM
      produit pr
    JOIN
      ligne li USING (product_id)
  GROUP BY
    cat_id, produit_id
), S AS (
  SELECT
    cat_id, MAX(tot_ventes) AS max_ventes, MIN(tot_ventes) AS min_ventes
  FROM
    R
  GROUP BY
    cat_id
)

SELECT
  r1.cat_id, r1.produit_id, r1.tot_ventes
FROM
  R r1
WHERE
  EXISTS (
    SELECT
      *
    FROM
      S s1
    WHERE
      s1.cat_id = r1.cat_id AND
      r1.tot_ventes IN (s1.max_ventes, s1.min_ventes)
  )
ORDER BY r1.cat_id ;
```

Requête 8

Lister les paires de clients qui habitent la même adresse.

Solution

```
SELECT
  c1.client_id, c2.client_id, c1.adresse
FROM
    client c1
  JOIN
    client c2 ON (c1.client_id < c2.client_id AND c1,adresse=c2.adresse)
;
```

Requête 9

Lister pour chaque fabricant, les cinq produits les vendus.

Solution

```
WITH R AS (
  SELECT
    pr.produit_id, pr.fabricant,
    RANK() OVER (PARTITION BY pr.fabricant ORDER BY SUM(li.qte) DESC) AS rnk
  FROM
    produit pr
  JOIN
    ligne li USING (produit_id)
)

SELECT
  fabriquant, produit_id, rnk
FROM
  R
WHERE
  rnk <=5
ORDER BY fabricant  ;
```

Requête 10

Lister pour chaque mois, les dix clients qui ont le plus dépensé.

Solution

```
WITH R AS (
  SELECT
    client_id,
    EXTRACT(MONTH FROM fa.date) AS mois,
    RANK() OVER (PARTITION BY EXTRACT(MONTH FROM fa.date), fa.client_id
                  ORDER BY SUM(fa.montant) DESC)  AS rnk
  FROM
    facture fa
)

SELECT
  R.mois,
  R.client_id,
  R.rnk
FROM
  R
WHERE
  R.rnk <= 10
ORDER BY
  R.mois, R.rnk
;
```

Solution

La réponse proposée au dessus est tordue. La suivante est plus pertinente.

```sql
WITH R AS (
  SELECT
    client_id,
    DATE_TRUNC('MONTH', fa.date) AS mois,
    RANK() OVER (PARTITION BY DATE_TRUNC('MONTH', fa.date), fa.client_id
                 ORDER BY SUM(fa.montant) DESC)  AS rnk
  FROM
    facture fa
)

SELECT
  R.mois,
  R.client_id,
  R.rnk
FROM
  R
WHERE
  R.rnk <= 10
ORDER BY
  R.mois, R.rnk
;
```

En PostgreSQL, pour définir un intervalle à l'aide de deux dates `debut` et `fin`, il suffit d'écrire `(début, fin)`. L'intervalle ne contient pas la date de fin.
Pour tester l'intersection/le recouvrement de deux intervalles, on utilise l'opérateur `OVERLAPS`

```
bd_2023-24=# SELECT
  ('2025-01-03'::date, '2025-01-10'::date) OVERLAPS
  ('2025-01-10'::date, '2025-01-15'::date)  ;
 overlaps
----------
 false
(1 row)

bd_2023-24=# SELECT
  ('2025-01-03'::date, '2025-01-10'::date) OVERLAPS
  ('2025-01-09'::date, '2025-01-15'::date)  ;
 overlaps
----------
 true
(1 row)
```

En PostgreSQL, pour extraire le mois d'un objet `dd` de type `date`, vous pouvez utiliser `EXTRACT(MONTH FROM dd)`. Le résultat est un entier entre 1 et 12, 1 pour janvier, …

```
postgres=# SELECT
  current_timestamp::date AS la_date,
  EXTRACT( MONTH FROM current_timestamp::date) AS le_mois ;

  la_date   | le_mois
------------+---------
 2025-01-03 |       1
```