heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Contraintes d'intégrité

Problème

Conception

Requêtes

Préparé par Gabor Maksay - 2017

Concepts clés

Termes introduits : contraintes d'intégrité obligatoire ou facultative, d'existence, d'inclusion, d'exclusion, cyclique ou de valeur.

IN

NOT IN

Introduction

Dans les chapitres précédents, nous avons appris à dessiner un schéma relationnel composé de tables, de leurs attributs et de leur clé primaire et leurs éventuelles clés étrangères.

Mais un tel schéma peut-il tenir compte de tous les paramètres communiqués par un mandant ? Par exemple du fait que seul un collaborateur membre d'une équipe peut en être le responsable ?

Malgré la richesse d'un schéma relationnel, il existe des règles qui ne peuvent être exprimées graphiquement. Ces contraintes sont rédigées sous la forme de compléments au schéma relationnel.

Le but de ce chapitre est de rendre le lecteur attentif à la nécessité de compléter un schéma relationnel avec toutes les règles de gestion non encore décrites.

Des requêtes SQL négatives permettent de vérifier si le contenu de la base de données est cohérent avec les contraintes d'intégrité spécifiées. En effet, certaines contraintes ne peuvent être ajoutées directement dans un SGBD.

Problème

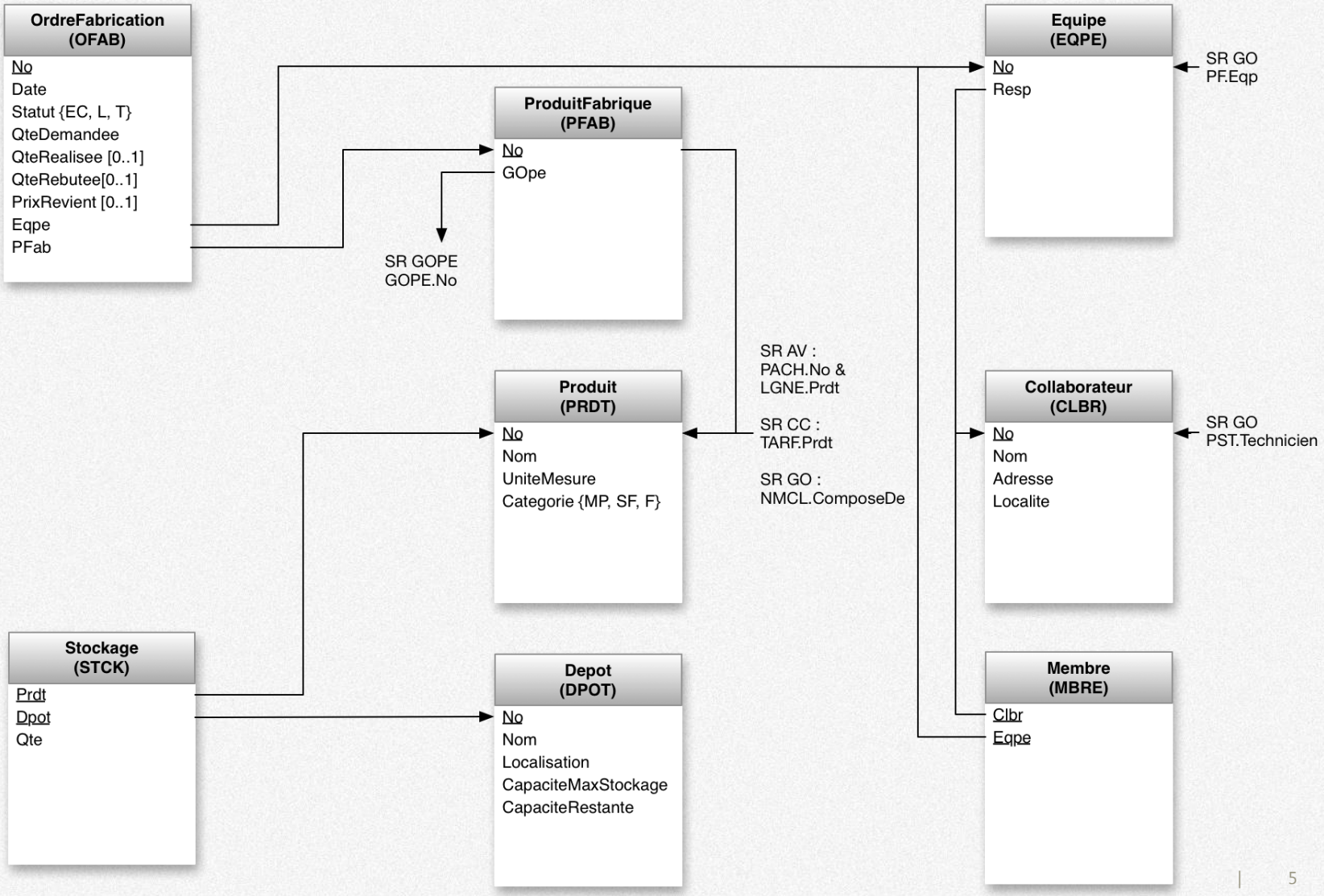
Le problème suivant constitue une extension de celui du chapitre précédent. Le mandant a en effet précisé quelques règles de gestion en vigueur dans son entreprise.

Le mandant désire non seulement modéliser ces règles, mais s'assurer que la base de données en exploitation les respecte. Dans le cas contraire, il désire que les situations qui ne les respectent pas soient mises en évidence.

1. La catégorie d'un produit fabriqué ne peut être égale qu'à 'SF' ou 'PF',
2. Si le statut d'un ordre de fabrication est égal à 'T', alors les quantités réalisée et rebutée ainsi que le prix de revient doivent être renseignés,
3. Un produit ne peut à la fois être acheté et fabriqué,
4. Le total d'une commande est égal à la somme des totaux de ligne qui la constituent, moins un rabais éventuel,
5. Dans le cas où le total d'une commande dépasse 30'000 CHF, le rabais de cette dernière est de 10%,
6. La durée d'une gamme opératoire est égale à la somme des durées des opérations qui la séquencent,
7. Seul un collaborateur membre d'une équipe peut en être le responsable ,
8. Si une machine effectue une opération sur un poste de travail, ce dernier est adapté pour cette machine.

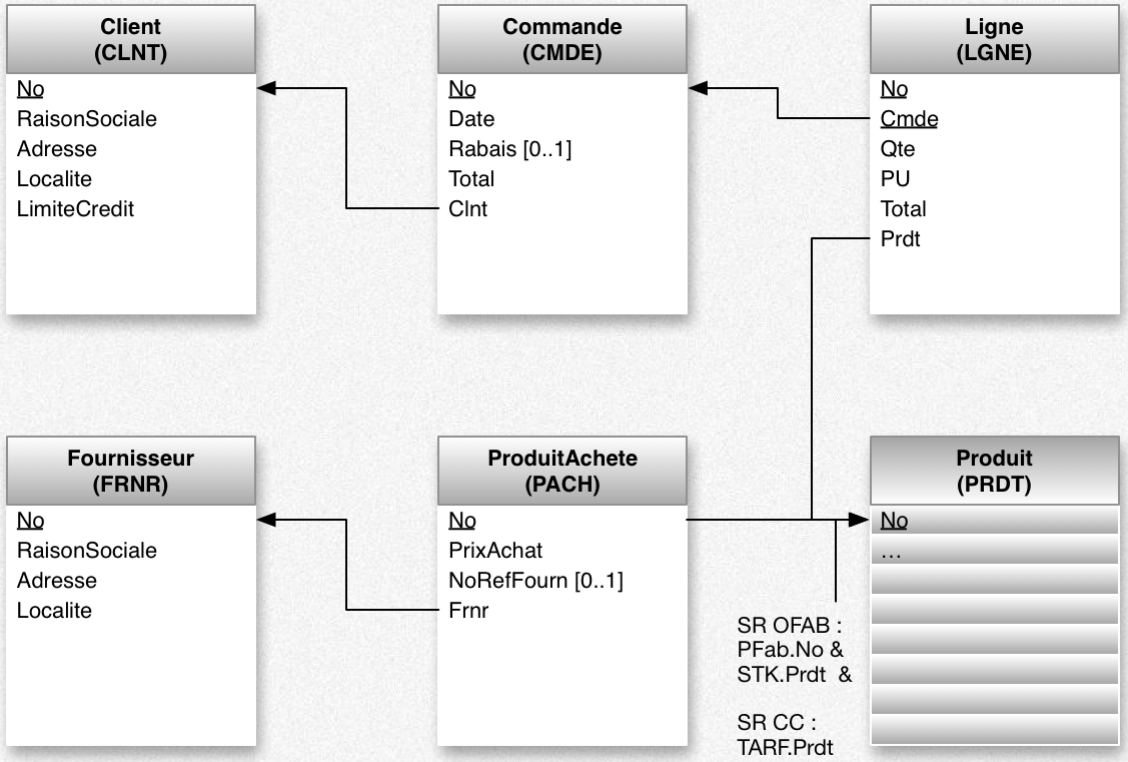
Problème

La figure ci-dessous présente le schéma relationnel correspondant à la partie *Ordre de fabrication* du problème énoncé.



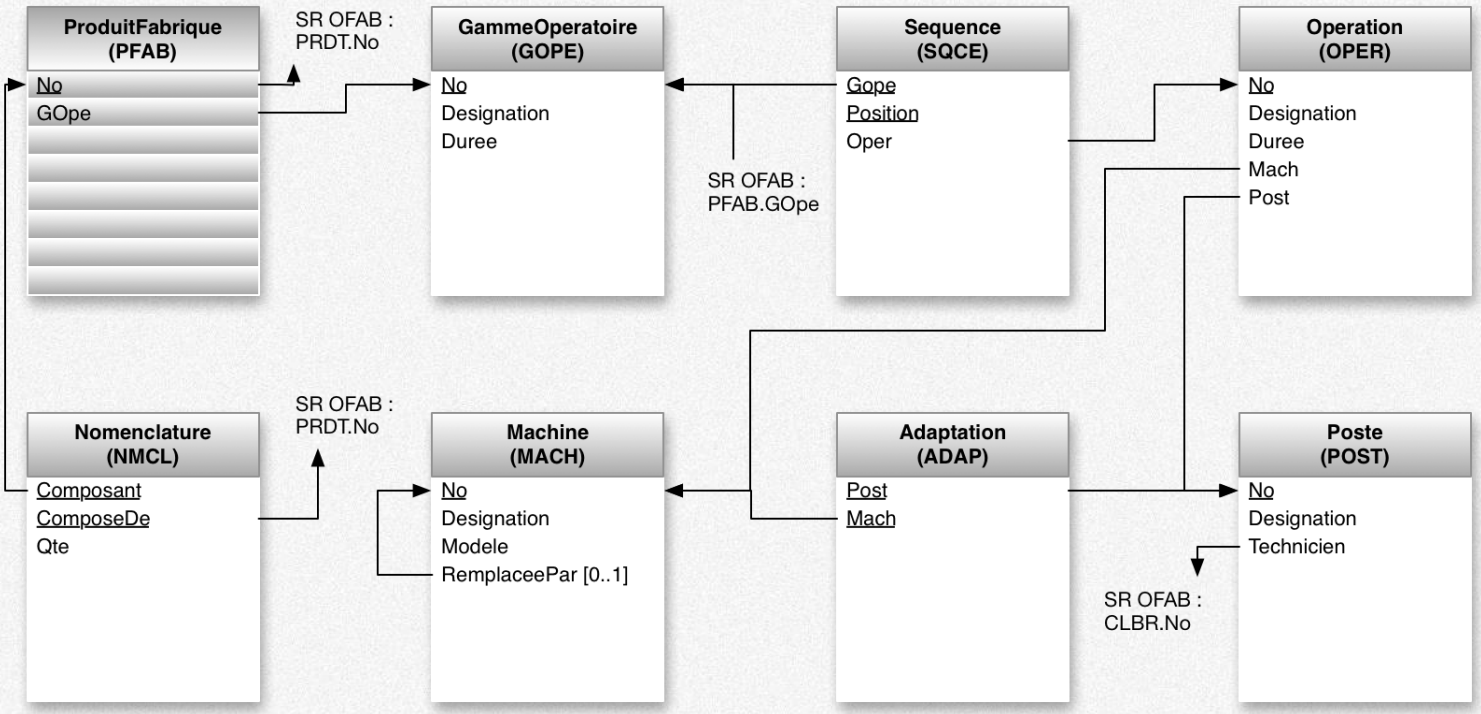
Problème

La figure ci-dessous présente le schéma relationnel correspondant à la partie *Achat-vente* du problème énoncé.



Problème

La figure ci-dessous présente le schéma relationnel correspondant à la partie *Gamme opératoire* du problème énoncé.



Conception

Un schéma relationnel ne peut représenter toutes les règles de gestion que le modèle de données doit respecter.

Une distinction s'impose donc entre un modèle de données et un schéma relationnel, ce qui est nouveau dans la démarche présentée ; en effet, le résultat d'une modélisation (= modèle de données) était jusqu'ici uniquement constitué d'un schéma relationnel.

Le but final de toute modélisation est celui de représenter une réalité donnée sous la forme d'un modèle. La modélisation de données ne déroge pas à cette règle. Un modèle comporte des éléments de base – tables, attributs, clés étrangères – ainsi que certaines restrictions traduisant les règles de gestion du monde réel.

Contraintes d'intégrité

Une contrainte d'intégrité est une règle complémentaire aux concepts de base du modèle de données qui vise à spécifier des propriétés sémantiques afin de garantir la cohérence et la validité du schéma des données. Elle est également utilisée pour vérifier cette cohérence et cette validité.

On peut déduire de cette définition qu'il s'agit d'une règle restrictive. Or, nous connaissons déjà deux éléments de base d'un modèle de données qui le restreignent :

- les clés primaires,
- les clés étrangères.

Il reste à définir les autres contraintes d'intégrité, rassemblées sous le terme de contraintes d'intégrité explicites, qui doivent être décrites explicitement.

Il est important de rappeler qu'un modèle ne peut être consistent et cohérent que s'il est composé de ses éléments de base complétés par les contraintes explicites, lesquelles font partie intégrante du modèle de données.

Les contraintes d'intégrité qui ne peuvent pas être vérifiées au niveau de la base de données doivent donc être :

- intégrées dans une démarche de construction d'une solution informatique complète (objet d'une autre étude), puis
- vérifiées a posteriori pour permettre l'élimination des erreurs décelées.

Pour vérifier une contrainte, nous devons élaborer une requête à même de détecter les instances de la base de données non conformes à cette contrainte. Cela consiste en fait à formuler une requête inversée par rapport à la contrainte.

Typologie des contraintes d’intégrité

Les concepts de clés primaires et de clés étrangères sont déjà des éléments contraignants d’un modèle et peuvent être illustrés sur le schéma relationnel Nous les regroupons dans les contraintes de type obligatoire, dans la mesure où chaque table doit avoir une clé primaire et chaque clés étrangère doit être décrite et référencer une clé primaire, que cette clé étrangère soit obligatoire ou pas.

Il existe des contraintes plus difficiles à exprimer graphiquement et que l’on range dans les contraintes de type explicite (c.-à-d. qui doivent être décrites explicitement) ; mentionnons à ce propos les contraintes :

- d'existence,
- d'inclusion,
- d'exclusion,
- d'égalité,
- cycliques,
- de valeurs et
- de séquencement.

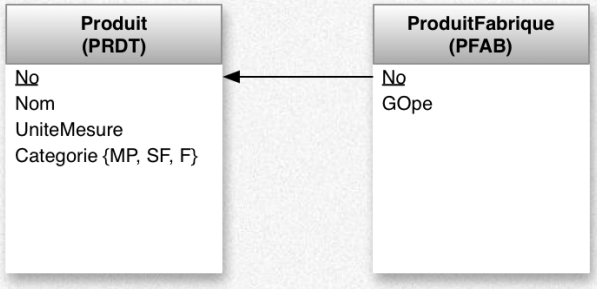
Des exemples de contraintes importantes sont décrites ci-après. Il est hors périmètre de ce cours de toutes les illustrer, que ce soit dû à leur rareté ou car l’exemple fil rouge n'est pas adapté.

Contrainte d'existence

Le mandant a communiqué une règle de gestion stipulant que dans le cas où un produit est un produit fabriqué, il doit avoir sa catégorie à *SF* (semi-fini) ou *PF* (produit fini).

Sur base de l'extrait du schéma relationnel ci-contre, nous faisons la constatation suivante : Un produit fabriqué est un produit : sa clé primaire constitue une clé étrangère sur la clé primaire de la table *Produit*.

Nous reformulons la règle de gestion afin de préciser qu'un produit, s'il est un produit fabriqué doit avoir son attribut *Categorie* égal à *SF* ou *PF*.



Contrainte d'existence

Or, nous constatons dans cet exemple qu'il est possible d'ajouter à la base de données un produit avec le no *1000* et une catégorie *autre* ainsi que de créer un produit fabriqué avec ce même no *1000*.

```
INSERT INTO PRDT VALUES (1000, 'Exemple', 'm', 'autre')
INSERT INTO PFAB VALUES (1000, 15)
```

Pour vérifier la contrainte d'existence, il convient de rédiger une requête fournissant les produits fabriqué dont la catégorie n'est pas égale à *SF* ou *PF*. Pour ce faire, il est nécessaire d'effectuer une jointure entre les tables *PRDT* et *PFAB* avant d'appliquer le filtre.

```
SELECT PFAB.No, Nom, UniteMesure, Categorie, GOpe
FROM   PRDT
INNER  JOIN PFAB
ON     PRDT.No = PFAB.No
WHERE  PRDT.Categorie <> 'SF'
AND    PRDT.Categorie <> 'PF'
```

L'exécution de cette requête permet de constater que le produit fabriqué dont le *No* vaut *1000* possède une catégorie *autre*.

No	Nom	UniteMesure	Categorie	GOpe
1000	Exemple	m	autre	15

Cette requête ne corrige pas le problème, mais permet de le mettre en évidence. Une éventuelle correction manuelle est encore nécessaire.

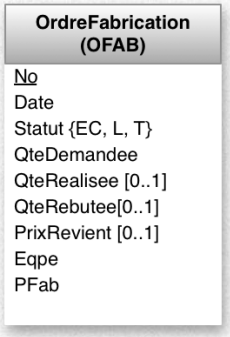
Contrainte d'inclusion

La deuxième règle de gestion exprimée concerne le fait que si le statut d'un ordre de fabrication est égal à *T*, alors les quantités réalisée et rebutée ainsi que le prix de revient doivent être renseignés.

On constate avec l'exemple ci-contre que la base de données accepte un ordre de fabrication, ici le no *1001* dont le statut est égal à *T* et dont les quantités réalisée et rebutée, ainsi que le prix de revient sont à *NULL* ce qui est en contradiction avec la contrainte énoncée ci-dessus.

Le but de la requête SQL est de détecter les ordres de fabrication dont le statut est à *T* et dont la quantité réalisée ou la quantité rebutée ou le prix de revient est à *NULL*.

Attention à l'emploi nécessaire des parenthèses entre les prédicats séparés par *OR*. En effet, sans l'emploi des parenthèses, il suffirait qu'un ordre de fabrication ait un des trois champs à *NULL*, indépendamment de la valeur de son statut pour qu'il soit retenu par la requête.




```
1  INSERT INTO OFAB VALUES
2      (1001, '2017-10-12', 'T', 1900, NULL, NULL, NULL, 6, 26)
```

```
1  SELECT *
2  FROM   OFAB
3  WHERE  Statut = 'T'
4  AND    (QteRealisee IS NULL OR
5          QteRebutee IS NULL OR
6          PrixRevient IS NULL)
```

No	Date	Statut	QteDemandee	QteRealisee	QteRebutee	PrixRevient	Eqpe	PFab
1001	2017-10-12	T	1900	NULL	NULL	NULL	6	26

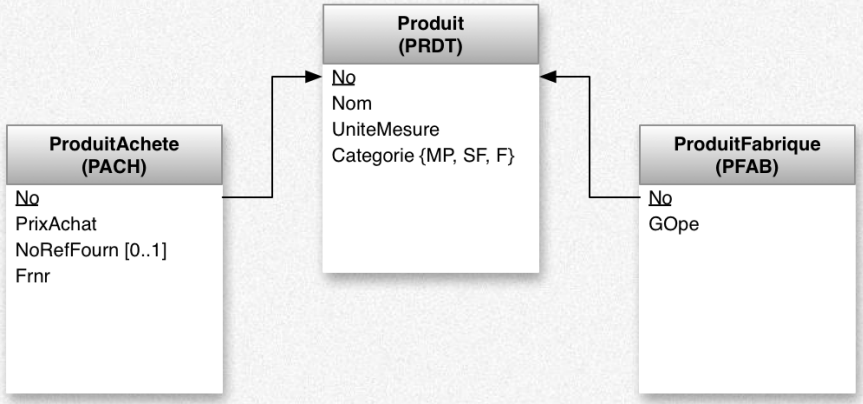
Contrainte d'exclusion

La troisième règle de gestion stipule qu'un produit ne peut à la fois être acheté et fabriqué.

Les tables utilisées pour la vérification sont *PRDT (Produit)*, *PACH* (*Produit acheté*) et *PFAB* (*Produit fabriqué*) .

Dans notre exemple, le produit *1001*, de nom *Produit1*, d'unité de mesure *m* et de catégorie *F* est non seulement un produit acheté mais également un produit fabriqué. En d'autres termes, la clé primaire du produit acheté et du produit fabriqué ont la même valeur et pointent, chacune, en tant que clé étrangère, sur la clé primaire de *PRDT*.

Nous constatons que cette situation ne pose pas de problème au niveau de la base de données. Pour respecter la contrainte référentielle, il suffit qu'il existe un produit acheté, respectivement fabriqué, qui puisse être référencé au travers de sa clé étrangère – qui est aussi sa clé primaire dans ce cas précis – et pointant vers la clé primaire de la *PRDT*.




```
1 INSERT INTO PRDT VALUES (1001, 'Produit1', 'm', 'F')
2 INSERT INTO PACH VALUES (1001, 100, NULL, 1)
3 INSERT INTO PFAB VALUES (1001, 1)
```

Contrainte d'exclusion

Dans notre exemple, la requête peut se formuler de la manière suivante : rechercher tous les produits qui sont à la fois un produit acheté et un produit fabriqué. Si cette requête fournit un résultat différent de l'ensemble vide, nous avons la preuve que la contrainte d'exclusion n'est pas respectée.

Les erreurs éventuelles sont identifiées au moyen de la requête SQL ci-contre.

Cette requête met en liaison, pour chaque produit de la table *PRDT*, les éventuelles clés étrangères correspondantes dans les tables *PACH* et *PRDT*. Si une ou plusieurs clés étrangères existent dans les deux tables à la fois, cette requête affiche les produits concernés.

Nous arrivons à la conclusion que le produit dont le *No* est égal à *1001* ne respecte pas la contrainte d'exclusion.

```
1  SELECT PRDT.No,Nom, UniteMesure, Categorie
2  FROM   PRDT
3  INNER  JOIN PACH
4  ON     PRDT.No = PACH.No
5  INNER  JOIN PFAB
6  ON     PRDT.No = PFAB.No
```

No	Nom	UniteMesure	Categorie
1001	Produit 1	m	F

Contrainte de valeur

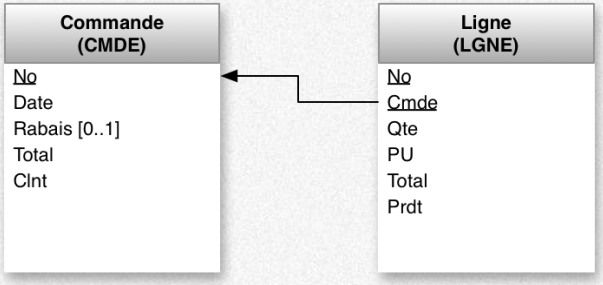
Le quatrième type de contrainte présenté permet de restreindre l'ensemble des valeurs que peut prendre l'attribut d'une table ou de contrôler la cohérence entre deux champs d'une ligne d'une ou plusieurs tables

Le mandant demande d'intégrer la règle de gestion permettant de vérifier que le total d'une commande est égal à la somme des totaux de ligne qui la constituent, moins un rabais éventuel.

Une fois de plus, l'ajout des lignes ci-contre est autorisé par la base de données et la contrainte n'est plus garantie.

Les tables concernées par cette vérification sont *CMDE* (*Commande*) et *LGNE* (*Ligne Commande*) .

On constate que le total des lignes de commande fait *39000*, alors que le total de commande additionné à son rabais font *40000.*




```
1 INSERT INTO CMDE VALUES (1001, '2017-08-28', 3900, 36100, 1)
2 INSERT INTO LGNE VALUES (1, 1001, 1000, 12, 12000, 26)
3 INSERT INTO LGNE VALUES (2, 1001, 2000, 6, 12000, 25)
4 INSERT INTO LGNE VALUES (3, 1001, 1500, 10, 15000, 14)
```


.

Contrainte de valeur

.

Le but de cette requête consiste à regrouper les commandes entre elles (ligne 6), de calculer les sommes des lignes concernant ces commandes (ligne 1) et de mettre en évidence celles dont la somme des lignes est différente du total de la commande additionné de son rabais (ligne 7).

Bien que cette requête soit présentée dans ce chapitre, les notions nécessaires pour résoudre cette dernière nécessitent des connaissances qui seront abordées lors du chapitre intitulé *Entrepôt de données*.

L'exécution de la requête permet de constater que le total des lignes de commande fait *39000*, alors que le total de commande additionné à son rabais font *40000*.

```
1  SELECT CMDE.No, CMDE.Total, CMDE.Rabais, SUM(LGNE.Total) AS TotalLignes,
2         SUM(LGNE.Total) - (CMDE.Total + Rabais) AS Difference
3  FROM   CMDE
4  INNER  JOIN LGNE
5  ON     CMDE.No = Cmde
6  GROUP  BY CMDE.No, CMDE.Total, CMDE.Rabais
7  HAVING SUM(LGNE.Total) - (CMDE.Total + Rabais) <> 0
```

No	Total	Rabais	TotalLignes	Difference
1001	36100.00	3900.00	39000.00	-1000.00

Contrainte de valeur
2e exemple

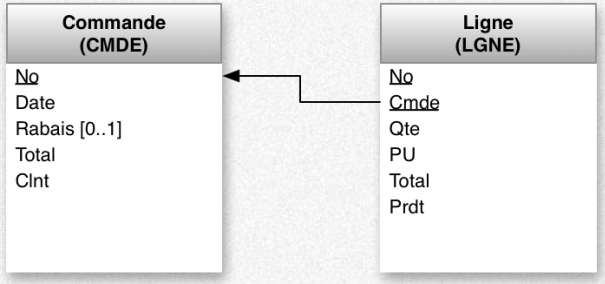
Le cinquième contrainte est également une contrainte de valeur.

Le mandant demande de tenir compte du fait que, dans le cas où le total d'une commande dépasse 30'000 CHF, le rabais de cette dernière est de 10%.

Les tables concernées par cette vérification sont *CMDE* (*Commande*) et *LGNE* (*Ligne Commande*).

L'insertion des lignes ci-contre étant permise par la base de données, la contrainte évoquée ci-dessus n'est plus garantie.

On constate que le total des lignes de commande fait *39000*. Le rabais devrait être de *3900* alors qu'il se monte effectivement à *2900*.




```
1 INSERT INTO CMDE VALUES (1001, '2017-08-28', 2900, 35100, 1);
2 INSERT INTO LGNE VALUES (1, 1001, 1000, 12, 12000, 26);
3 INSERT INTO LGNE VALUES (2, 1001, 2000, 6, 12000, 25);
4 INSERT INTO LGNE VALUES (3, 1001, 1500, 10, 15000, 14);
```


.

Contrainte de valeur
2e exemple

.

Le but de la requête ci-contre consiste à regrouper les commandes entre elles (ligne 5), de calculer les sommes des lignes concernant ces commandes (ligne 1) et de mettre en évidence celles dont la somme des lignes est supérieure à *30000* (ligne 6) et dont le 10% de cette somme est différente au rabais effectif de la commande (ligne 7).

```
1  SELECT CMDE.No, CMDE.Total, CMDE.Rabais, SUM(LGNE.Total) AS TotalLignes
2  FROM   CMDE
3  INNER  JOIN LGNE
4  ON     CMDE.No = Cmde
5  GROUP  BY CMDE.No, CMDE.Total, CMDE.Rabais
6  HAVING SUM(LGNE.Total) > 30000
7  AND    SUM(LGNE.Total) / 10 <> CMDE.Rabais
```

Bien que cette requête soit présentée dans ce chapitre, les notions nécessaires pour résoudre cette dernière nécessitent des connaissances qui seront abordées lors du chapitre intitulé *Entrepôt de données*.

L'exécution de la requête permet de constater que le total des lignes de commande fait *39000*, alors que le rabais effectif se mont à *2900*.

No	Total	Rabais	TotalLignes
1001	35100.00	2900.00	39000.00

Contrainte de valeur
3e exemple

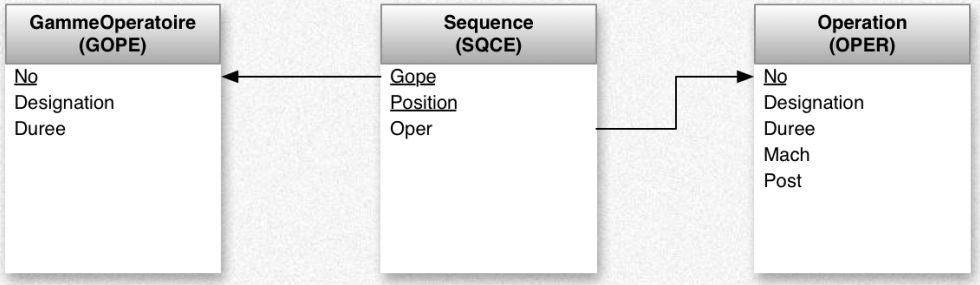
Le sixième contrainte est également une contrainte de valeur.

Le mandant demande de vérifier que la durée d'une gamme opératoire soit à chaque fois égale à la somme des durées des opérations qui la séquencent.

Les tables concernées par cette vérification sont *GOPE* (*Gamme opératoire*) et *SQCE* (*Sequence*) .

L'insertion des lignes ci-contre ne posant pas de problème particulier à la base de données, la contrainte évoquée ci-dessus ne peut être garantie.

On constate que la durée de la gamme est de *340* alors que les opérations *1* et *2* ont une durée de *100*, chacune, et l'opération *3* de *40*.




```
INSERT INTO GOPE VALUES
    (1001,
    'Carton 100 enveloppes - blanche - C4 - 80 g/m2 - sans fenêtre - sans soufflet - sans patte autocollante',
    '340');
INSERT INTO SQCE VALUES (1001, 1, 1);
INSERT INTO SQCE VALUES (1001, 2, 2);
INSERT INTO SQCE VALUES (1001, 3, 3);
```


.

Contrainte de valeur
3e exemple

.

Le but de cette requête consiste à regrouper les gammes opératoires entre elles (ligne 9), de calculer la sommes des durées des opérations qui la constituent (ligne 3) et de mettre en évidence celles dont la somme des durées des opérations diffèrent de la durée de la gamme opératoire considérée (ligne 10).

Bien que cette requête soit présentée dans ce chapitre, les notions nécessaires pour résoudre cette dernière nécessitent des connaissances qui seront abordées lors du chapitre intitulé *Entrepôt de données*.

L'exécution de la requête permet de constater que la durée de la gamme opératoire est de *340*, alors que la somme des opérations qui la constituent fait *240*.

```
1  SELECT GOpe.No, GOpe.Designation,
2         GOpe.Duree AS "Durée gamme opératoire",
3         SUM(Oper.Duree) AS "Somme durées opérations"
4  FROM   GOpe
5  INNER  JOIN SQCE
6  ON     GOpe.No = GOpe
7  INNER  JOIN OPER
8  ON     Oper.No = Oper
9  GROUP  BY GOpe.No, GOpe.Designation, GOpe.Duree
10 HAVING GOpe.Duree <> SUM(Oper.Duree)
```

No	Designation	Durée gamme opératoire	Somme durées opérations
1001	Carton 100 enveloppes - blanche - C4 - 80 g/m2 -...	340	240

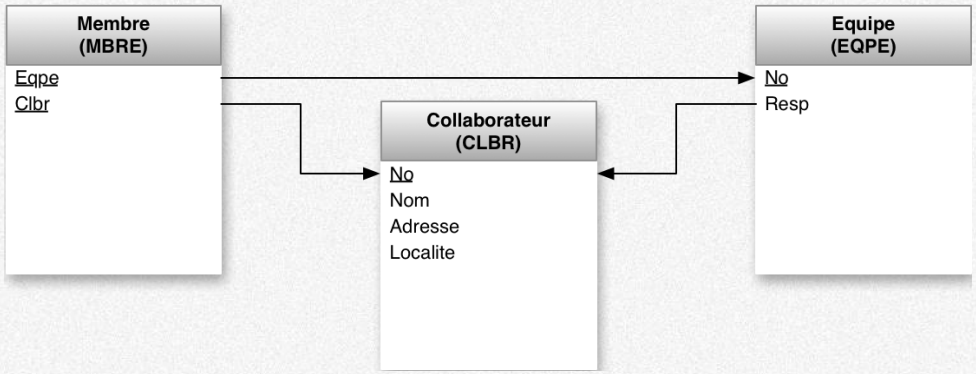
Contrainte cyclique

Un schéma relationnel fait l'objet de cycles, dans le sens où il est possible de naviguer dans le schéma en partant et en arrivant à la même ligne d'une table. En d'autres termes, une ligne d'une table présente la source et la destination de l'expression d'une contrainte. Ces cycles sont souvent source d'ambiguïté.

Dans notre exemple, le mandant demande de tenir compte de la règle de gestion selon laquelle seul un collaborateur membre d'une équipe peut en être le responsable .

À défaut de contrainte spécifique, le schéma relationnel permettrait de représenter la situation suivante, plutôt surprenante : le collaborateur identifié par le numéro *1004* est responsable de l'équipe *752* alors que les seuls membres de l'équipe *752* portent les numéros *1001*, *1002* et *1003*.

Le non-respect de la contrainte cyclique constitue probablement le risque le plus élevé d'incohérence dans la base de données. En effet, les contraintes cycliques sont rarement exprimées par les utilisateurs sous forme de règles de gestion et aucun mécanisme de contrôle de la base de données ne peut vérifier ce type de contraintes.




```
1  INSERT INTO CLBR VALUES (1001, 'Coll1', 'Adresse1', 'Localité1');
2  INSERT INTO CLBR VALUES (1002, 'Coll2', 'Adresse2', 'Localité2');
3  INSERT INTO CLBR VALUES (1003, 'Coll3', 'Adresse3', 'Localité3');
4  INSERT INTO CLBR VALUES (1004, 'Coll4', 'Adresse4', 'Localité4');
5  INSERT INTO EQPE VALUES (752, 1004);
6  INSERT INTO MBRE VALUES (1001, 752);
7  INSERT INTO MBRE VALUES (1002, 752);
8  INSERT INTO MBRE VALUES (1003, 752);
```

Contrainte cyclique

Le but de la requête SQL consiste à trouver tout collaborateur responsable d'une équipe mais qui n'en serait pas membre.

La sous-requête recherche les responsables d'équipe qui sont membres de leur équipe. La requête principale affiche les collaborateurs des équipes qui sont responsable d'une équipe et qui ne figurent pas dans la sous-requête, à savoir les responsables d'équipes qui ne sont pas membres de l'équipe dont ils sont responsables.

L'exécution de la requête indique que le collaborateur dont le *No* vaut *1004* est dans ce cas.

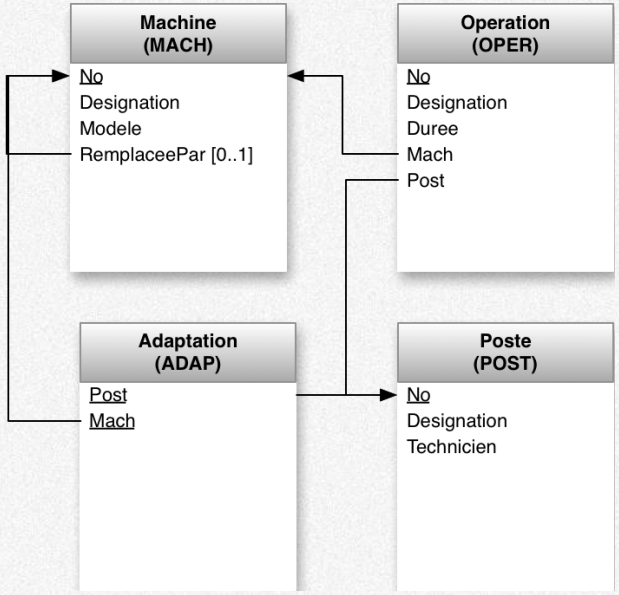
```sql
SELECT CLBR.No, Nom, Adresse, Localite
FROM   CLBR
INNER  JOIN EQPE
ON     CLBR.No = Resp
WHERE  Resp NOT IN (
    SELECT Resp
    FROM   EQPE
    INNER  JOIN MBRE
    ON     EQPE.No = MBRE.Eqpe
    WHERE  Resp = MBRE.Clbr)
```

No	Nom	Adresse	Localite
1004	Coll4	Adresse4	Localité4

Contrainte cyclique
2e exemple

Le dernier exemple de ce chapitre est également à classer dans la famille des contraintes cycliques. Elle stipule que si une machine effectue une opération sur un poste de travail, ce dernier est adapté pour cette machine.

Comme pour les autres exemples, la base de données ne s'oppose pas à l'insertion de la ligne ci-contre : une nouvelle opération avec le numéro *1001*, dont la désignation est *Découpe* est censée être effectuée par la machine *4* sur le poste *1*. Or, il n'existe pas de correspondance entre le poste *1* et la machine *4* dans la table *ADAP*.




```
1 INSERT INTO OPER VALUES (1001, 'Découpe', 100, 4, 1);
```

Contrainte cyclique
2e exemple

Le but de la requête SQL consiste à trouver toute machine effectuant des opérations sur un poste de travail, machine n'étant pas adaptée pour ce poste.

La sous-requête recherche les machines qui sont adaptées sur quel poste et la requête principale affiche les machines effectuant des opérations par des machines sur un poste de travail, machines qui ne se retrouveraient pas dans le résultat de la sous-requête..

L'exécution de la requête indique que la machine *4* se trouve dans le cas précité.

```
1  SELECT MACH.*
2  FROM   MACH
3  INNER  JOIN OPER
4  ON     MACH.No = Mach
5  INNER  JOIN POST
6  ON     POST.No = Post
7  WHERE  MACH.No NOT IN (
8      SELECT MACH.No
9      FROM   MACH
10     INNER  JOIN ADAP
11     ON     MACH.No = Mach
12     INNER  JOIN POST
13     ON     POST.No = Post)
```

No	Designation	Modele	RemplaceePar
4	Unité d'impression	W&D 22	NULL

Synthèse

Dans ce chapitre, nous avons vu que certaines règles de gestion ne peuvent pas être modélisées directement dans un schéma relationnel. Néanmoins, ces règles sont très importantes pour maintenir la cohérence des informations stockées dans une base de données.

Nous avons défini le terme de contrainte d'intégrité comme une règle restrictive qui complète un modèle de données. Nous avons mis en évidence deux familles de contraintes : les contraintes obligatoires (obligatoirement présentes dans un schéma relationnel) et les contraintes explicites (qui doivent être rédigées de manière explicite).

Nous avons présenté une typologie des contraintes explicites.

Nous avons également constaté que tous les SGBD relationnels du marché n'offrent pas encore une vérification systématique de tous les types de contraintes et donc que les règles de gestion sont le plus souvent implantées dans une solution informatique complète.

Néanmoins, nous avons constaté l'utilité de transformer ces contraintes en requêtes SQL afin de pouvoir vérifier a posteriori que les informations stockées dans une base de données respectent les règles de gestion imposées par les mandants.

Pour effectuer ces contrôles, nous avons utilisé, entre autres, des requêtes négatives incluant des sous-requêtes afin de mettre en évidence les cas dérogeant aux règles édictées.