# SAE3 : Réaliser un projet en BD et WEB

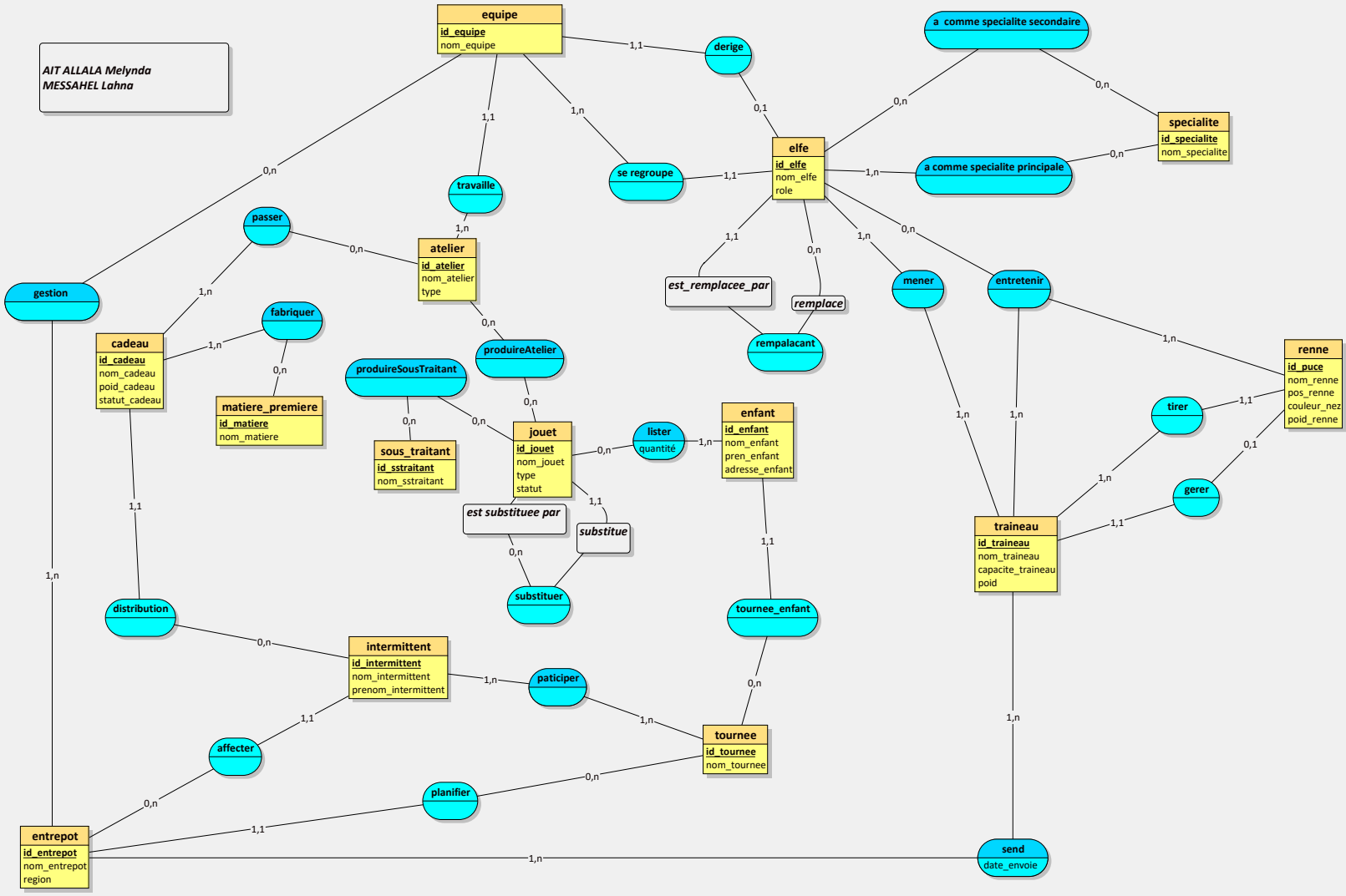
Rapport sur la Modélisation Conceptuelle de Données (MCD) et l'Implémentation des Requêtes : Justifications, Dictionnaire des Données, Requêtes SQL et Algébriques Compatibles Oracle.

**NOM : MESSAHEL**

**PRENOM : Lahna**

**NOM : AIT ALLALA**

**PRENOM : Melynda**


AIT ALLALA Melynda
MESSAHEL Lahna
equipe
id_equipe
nom_equipe
derige
a comme specialite secondaire
specialite
id_specialite
nom_specialite
elfe
id_elfe
nom_elfe
role
a comme specialite principale
travaille
se regroupe
passer
atelier
id_atelier
nom_atelier
type
gestion
est_remplacee_par
remplace
rempalacant
mener
entretenir
renne
id_puce
nom_renne
pos_renne
couleur_nez
poid_renne
tirer
gerer
cadeau
id_cadeau
nom_cadeau
poid_cadeau
statut_cadeau
fabriquer
matiere_premiere
id_matiere
nom_matiere
produireSousTraitant
produireAtelier
sous_traitant
id_sstraitant
nom_sstraitant
jouet
id_jouet
nom_jouet
type
statut
lister
quantité
enfant
id_enfant
nom_enfant
pren_enfant
adresse_enfant
est substituee par
substitue
substituer
traineau
id_traineau
nom_traineau
capacite_traineau
poid
distribution
intermittent
id_intermittent
nom_intermittent
prenom_intermittent
paticiper
tournee_enfant
affecter
tournee
id_tournee
nom_tournee
planifier
entrepot
id_entrepot
nom_entrepot
region
send
date_envoie

## Justifications du Modèle Conceptuel de Données (MCD) - SAE

## Partie BD Christmas'2025:

**1. L'association 'a_comme specialite_principale' :**

- La cardinalité (1,n) du côté 'elfe' : chaque elfe possède une ou plusieurs

spécialités principales.

- La cardinalité (0,n) du côté 'specialite' : Certaines spécialités vient d'etre crée donc elles ne peuvent pas encore être utilisées par les elfes, tandis que d'autres seront utilisées par plusieurs d'entre eux.

**2. L'association 'a_comme specialite_secondaire' :**

- La cardinalité (0,n) du côté 'elfe' : un elfe possède zéro ou plusieurs

spécialités secondaires, c'est-à-dire qu'un elfe peut ne pas posséder de

spécialité secondaire ou en avoir plusieurs.

- La cardinalité (0,n) du côté 'specialite' : certaines spécialités peuvent ne

pas être encore utilisées par des elfes donc 0 ou bien peut être utilisé par

plusieurs elfes .

**3. L'association 'remplacant' :**

- La cardinalité (1,1) pour "est_remplacee_par" : lorsqu'un elfe est absent, il est

remplacé par un seul et unique elfe.

- La cardinalité (0,n) pour "remplace" : un elfe peut ne remplacer aucun autre

elfe, comme il peut en remplacer plusieurs ayant la même spécialité.

**4. L'association 'se_regroupe' :**

- La cardinalité (1,1) du côté 'elfe' : un elfe ne peut appartenir qu'à une seule

équipe.

- La cardinalité (1,n) du côté 'equipe' : une équipe doit contenir au moins un

elfe, mais peut en contenir plusieurs.

**5. L'association 'derige' :**

- La cardinalité (0,1) du côté 'elfe' : un elfe peut ne pas être dirigeant, mais s'il

l'est, il ne peut diriger qu'une seule équipe.

- La cardinalité (1,1) du côté 'equipe' : chaque équipe doit être dirigée par un et

un seul elfe.

**6. L'association 'passer' :**

- La cardinalité (1,n) du côté 'cadeau' : les cadeaux passent d’atelier en atelier

pour assurer leur traçabilité, ils doivent donc passer au minimum par un

atelier et au maximum par plusieurs.

- La cardinalité (0,n) du côté 'atelier' : un atelier peut être nouvellement créé et

ne pas encore recevoir de cadeaux donc 0 , mais il peut ensuite en recevoir

plusieurs.

**7. L'association 'travaille' :**

- La cardinalité (1,1) du côté 'equipe' : une équipe travaille dans un seul et

unique atelier.

- La cardinalité (1,n) du côté 'atelier' : un atelier peut accueillir une ou plusieurs

équipes qui y travaillent.

**8. L'association 'lister' :**

- La cardinalité (1,n) du côté 'enfant' : chaque enfant a une liste de jouets, il

peut donc souhaiter un ou plusieurs cadeaux.

- La cardinalité (0,n) du côté 'jouet' : un jouet peut ne pas être demandé par un

enfant ou être souhaité par plusieurs.

**9. L'association 'produireSousTraitant' :**

- La cardinalité (0,n) du côté 'jouet' : certains jouets ne sont pas produits par

des sous-traitants, tandis que d'autres nécessitent plusieurs sous-traitants.

- La cardinalité (0,n) du côté 'sous_traitant' : un sous-traitant peut ne pas

produire certains jouets car probablemet ils seront produit par des ateliers ou il peut produire plusieurs.

**10. L'association 'produireAtelier' :**

• La cardinalité (0,n) du côté 'jouet' : certains jouets ne sont pas produits dans les

ateliers, tandis que d'autres nécessitent plusieurs ateliers.

• La cardinalité (0,n) du côté 'atelier' : Un atelier peut ne pas produire certains jouets car ils peuvent être envoyés pour être fabriqués par des sous-traitants, ou il peut produire plusieurs autres

**11. L'association 'substituer' :**

• La cardinalité (0,n) pour "est_substituee_par" : Un jouet, lors de la première création, peut ne pas être remplacé ou être remplacé par

plusieurs autres après l’avoir créé et lui avoir ajouté sa liste de remplaçants.

• La cardinalité (1,1) pour "substitue" : Un jouet ne peut remplacer qu'un seul autre

**12. L'association 'gestion'** :

• La cardinalité (0,n) du côté 'equipe' : une équipe peut ne gérer aucun entrepôt ou en gérer plusieurs.

• La cardinalité (1,n) du côté 'entrepot' : chaque entrepôt est géré par au moins une équipe logistique comme il peut gérer plusieurs .

**13. L'association 'entretenir' :**

• La cardinalité (1,n) du côté 'renne' : un renne peut être entretenu par un ou plusieurs

elfes..

• La cardinalité (1,n) du côté 'traineau' :un traîneau peut être entretenu par un ou

plusieurs elfes et rennes.

• La cardinalité (0,n) du côté 'elfe' : un elfe peut ne pas entretenir de traîneau ou de

renne, mais peut aussi en entretenir plusieurs.

**14. L'association 'paticiper' :**

• La cardinalité (1,n) du côté 'intermittent' : un intermittent participe à au moins une

tournée, mais peut en faire plusieurs.

• La cardinalité (1,n) du côté 'tournee' : une tournée peut avoir un ou plusieurs

intermittents.

**15. L'association 'distribution' :**

• La cardinalité (0,n) du côté 'intermittent' : un intermittent peut ne distribuer aucun

cadeau ou en distribuer plusieurs.

• La cardinalité (1,1) du côté 'cadeau' : un cadeau est distribué par un seul intermittent.

**16. L'association 'affecter' :**

• La cardinalité (0,n) du côté 'entrepot' : un entrepôt peut ne pas avoir d’intermittents

affectés ou en avoir plusieurs.

• La cardinalité (1,1) du côté 'intermittent' : un intermittent est affecté à un seul

entrepôt.

**17. L'association 'planifier' :**

• La cardinalité (0,n) du côté 'tournee' : une tournée nouvellement créée peut ne pas

encore être planifiée par un entrepôt ou être planifiée par plusieurs.

• La cardinalité (1,1) du côté 'entrepot' : un entrepôt ne planifie qu'une seule tournée.

**18. L'association 'send' :**

• La cardinalité (1,n) du côté 'traineau' : un traîneau peut être envoyé à un ou plusieurs

entrepôts.

• La cardinalité (1,n) du côté 'entrepot' : un entrepôt peut recevoir un ou plusieurs

traîneaux.

**19. L'association 'fabriquer' :**

• La cardinalité (1,n) du côté 'cadeau' : un cadeau est fabriqué à partir d’au moins une

matière première, voire plusieurs.

• La cardinalité (0,n) du côté 'matiere_premiere' : certaines matières ne sont pas utilisées pour fabriquer tous les cadeaux comme d’autre sont utilisées pour fabriquer plusieurs.

**20. L'association 'mener' :**

• La cardinalité (1,n) du côté 'elfe' : un elfe peut mener un ou plusieurs traîneaux.

• La cardinalité (1,n) du côté 'traineau' : un traîneau peut être mené par un ou

plusieurs elfes.

**21. L'association 'tournee_enfant'** :

• La cardinalité (0,n) du côté 'tournee' : une tournée peut être nouvellement créée et

ne pas encore inclure d'enfants , après la création peut avoir plusieurs enfants.

• La cardinalité (1,1) du côté 'enfant' : un enfant est inscrit à une seule tournée.

**22. L'association 'tirer' :**

• La cardinalité (1,n) du côté 'traineau' : un traîneau est tiré par plusieurs renne mais au moins un renne.

• La cardinalité (1,1) du côté 'renne' : un renne ne peut tirer qu’un seul traîneau.

**23. L'association 'gerer' :**

• La cardinalité (1,1) du côté 'traineau' : un traîneau est géré par un seul renne.

• La cardinalité (0,1) du côté 'renne' : un renne peut ne pas gérer de traîneau donc 0 , sauf s'il est celui au nez rouge donc 1.

# Dictionnaire des données

| attribut | description | type | longueur | propriétés | Règle de calcul |
|---|---|---|---|---|---|
| id_specialite | Identifiant spécialité | Chaine de caractère De longueur fixe | 3 | Clé | |
| nom_specialite | Nom de la spécialité | Chaine de caractère | 50 | | |
| id_atelier | Identifiant atelier | Chaine de caractère De longueur fixe | 3 | Clé | |
| nom_atelier | Nom de l'atelier | Chaine de caractère | 50 | | |
| type | Type de l'atelier | Chaine de caractère | 50 | | |
| id_matiere | Identifiant de la matiere | Chaine de caractère de longueur fixe | 3 | Clé | |
| nom_matiere | Nom de la matiere | Chaine de caractère | 50 | | |
| id_sstraitant | Identifiant du sous-traitant | Chaine de caractère de longueur fixe | 4 | Clé | |
| nom_sstraitant | Nom du sous-traitant | Chaine de caractère | 50 | | |
| id_tournee | Identifiant de la tournée | Chaine de caractère De longueur fixe | 3 | Clé | |
| nom_tournee | Nom de la tournée | Chaine de caractère | 50 | | |

| | | | | | |
|---|---|---|---|---|---|
| id_enfant | Identifiant de l'enfant | Chaine de caractère De longueur fixe | 3 | Clé | |
| nom_enfant | Nom de l'enfant | Chaine de caractère | 50 | | |
| pren_enfant | Prénom de l'enfant | Chaine de caractère | 50 | | |
| adresse_enfant | Adresse de l'enfant | Chaine de caractère | 50 | | |
| id_entrepot | Identifiant de l'entrepôt | Chaine de caractère de longueur fixe | 4 | clé | |
| nom_entrepot | Nom de l' entrepôt | Chaine de caractère | 50 | | |
| region | La region dans laquelle il se situe l' entrepôt | Chaine de caractère | 50 | | |
| id_intermittent | Identifiant de l'intermittent | Chaine de caractère de longueur fixe | 3 | clé | |
| nom_intermittent | Nom de l'intermittent | Chaine de caractère | 50 | | |
| prenom_intermittent | Prénom de l'intermittent | Chaine de caractère Chaine de caractère | 50 | | |
| id_cadeau | Identifiant cadeau | Chaine de caractère De longueur fixe | 3 | clé | |

| | | | | | |
|---|---|---|---|---|---|
| nom_cadeau | Nom du cadeau | Chaine de caractère | 50 | | |
| poid_cadeau | Poids du cadeau | numérique | | | |
| statut_cadeau | Statut du cadeau | Chaine de caractère | 50 | | |
| id_equipe | Identifiant de l'équipe | Chaine de caractère de longueur fixe | 3 | clé | |
| nom_equipe | Nom de l'équipe | Chaine de caractère | 50 | | |
| id_elfe | Identifiant de l'elfe | Chaine de caractère de longueur fixe | 3 | clé | |
| nom_elfe | Nom de l'elfe | Chaine de caractère | 50 | | |
| role | Le role de l'elfe | Chaine de caractère | 50 | | |
| id_traineau | Identifiant du traîneau | Chaine de caractère de longueur fixe | 3 | clé | |
| nom_traineau | Nom du traîneau | Chaine de caractère | 50 | | |
| capacite_traineau | Capacité du traîneau | Numérique | | {0,1,2,3,4,5,6,7,8} | |
| poid | Poid du traîneau | numérique | | | |
| id_puce | Identifiant de renne | Chaine de caractère de longueur fixe | 3 | clé | |
| nom_renne | Nom de renne | Chaine de caractère | 50 | | |

| | | | | | |
|---|---|---|---|---|---|
| pos_renne | Position de renne | Chaine de caractère | 50 | | |
| couleur_nez | Couleur du nez de renne | Chaine de caractère | 50 | | |
| poid_renne | Poids du renne | numérique | | | |
| qte | Quantité de jouets que l'enfant a choisit | numérique | | | |
| date_envoie | la date d'envoie du traîneau vers un entrepôt | date | | | |
| id_jouet | Identifiant du jouet | Chaine de caractère de longueur fixe | 3 | clé | |
| nom_jouet | Nom du jouet | Chaine de caractère | 50 | | |
| type | Type de jouet | Chaine de caractère | 50 | | |
| statut | Statut du jouet | Chaine de caractère | 50 | | |

# Les requêtes compatibles Oracle sous formes SQL et formes algébriques :

**Requête 01** : Afficher les noms des cadeaux disponibles et leur poids, triés par poids décroissant

Forme SQL : SELECT nom_cadeau, poid_cadeau FROM cadeau WHERE statut_cadeau= 'En livraison' ORDER BY poid_cadeau DESC;

Forme Algébrique : π nom_cadeau, poid_cadeau (σ statut_cadeau= 'En livraison' (cadeau))

Remarque : La forme algébrique affiche bien le nom et le poids des cadeaux disponibles mais pas trié en ordre décroissant car la notion de ORDER BY n'existe pas en algèbre relationnelle

**Requête 02** : Lister les noms et prénoms des enfants qui ont choisi une poupée comme jouet

Forme SQL : select distinct nom_enfant,pren_enfant from enfant e , lister l , jouet j where e.id_enfant=l.id_enfant and l.id_jouet=j.id_jouet and nom_jouet='Poupée';

Forme algébrique : π nom_enfant, pren_enfant (enfant ⋈ lister ⋈ σ nom_jouet='Poupée'(jouet))

**Requête 03** : lister toutes les informations concernant les enfants
Forme SQL : select * from enfant;
Forme algébrique : π id_enfant,nom_enfant,pren_enfant,adresse_enfant,id_tournee (enfant )

**Requête 04** : Lister les noms des cadeaux fabriqués à partir de la matière "bois"

Forme SQL : select nom_cadeau from cadeau c , fabriquer f ,matiere_premiere m where c.id_cadeau=f.id_cadeau and f.id_matiere=m.id_matiere and lower(nom_matiere)='bois';

Forme algébrique : π nom_cadeau (cadeau ⋈ fabriquer ⋈ σ nom_jouet='bois'(matiere_matiere))

Remarque : UPPER et LOWER n'existe pas en algèbre relationnel

**Requête 05** : Lister les noms des équipes qui regroupent des elfes ayant un rôle de Menuisier et de Logistique
Forme SQL : select nom_equipe from equipe eq , elfe el where eq.id_equipe=el.id_equipe_se_regroupe and role = 'Menuisier' intersect select nom_equipe from equipe eq , elfe el where eq.id_equipe=el.id_equipe_se_regroupe and role =

'Logistique' ;

Forme algébrique : π nom_equipe (equipe ⋈ σ role= 'Menuisier' (elfe)) ∩ π nom_equipe (equipe ⋈ σ role= 'Logistique' (elfe)) ;

**Requête 06** :Lister les noms des  entrepôts qui ont reçu un traîneau à la date du 25/12/01 ou à la date du 2025/12/03

Forme SQL : select nom_entrepot from entrepot e , send s where e.id_entrepot=s.id_entrepot and date_envoie=TO_DATE('2025-12-01', 'YYYY-MM-DD') union select nom_entrepot from entrepot e , send s where e.id_entrepot=s.id_entrepot and date_envoie=TO_DATE('2025-12-03', 'YYYY-MM-DD');

Forme algébrique :  π nom_entrepot (entrepot ⋈ σ date_envoie = '2025-12-01'(send)) ∪ π nom_entrepot (entrepot ⋈ σ date_envoie = '2025-12-03' (send)) ;

**Requête 07** :  Lister les  noms des traîneaux dont le poids est compris entre 400 et 800

Forme SQL : select nom_traineau from traineau where poid <=800 and poid >=400;
autre méthode : select nom_traineau from traineau where poid  BETWEEN 400 and 800;

Forme algébrique :  π nom_traineau (σ(poid <= 800) ∧ (poids >= 400)(traineau)) ;

**Requête 08** : Lister les identifiants des entrepôts situés dans les régions Europe, Asie et Afrique

Forme SQL :SELECT id_entrepot  FROM entrepot WHERE region IN ('Europe', 'Asie','Afrique');

autre méthode :
SELECT id_entrepot  FROM entrepot WHERE region='Europe' OR region='Asie' OR region='Afrique';

Forme algébrique :

π id_entrepot (σ(region = 'Europe') ∨ (region = 'Asie') ∨ (region = 'Afrique) (entrepot))

**Requête 09** : Lister les noms des équipes  qui ne regroupent pas des elfes ayant un rôle d Emballage

Forme SQL : select nom_equipe from equipe MINUS  select nom_equipe from equipe ,elfe where equipe.id_equipe=elfe.id_equipe_se_regroupe and role='Emballage';

Forme algébrique : π nom_equipe (equipe) - π nom_equipe (equipe ⋈ σ role = 'Emballage'(elfe));

**Requête 10** : Lister les identifiants des enfants dont le nom commence par "D" et dont le prénom se termine par "l"

Forme SQL : select id_enfant from enfant where nom_enfant LIKE 'D%' AND pren_enfant LIKE '%l';

Forme algébrique : π id_enfant (σ nom_enfant ≈ 'D*' ∧ pren_enfant ≈ '*l' (enfant))

**Requête 11** : Lister les noms des enfants dont le prénom ne contient pas de voyelle en deuxième position

Forme SQL : select Distinct nom_enfant from enfant e where pren_enfant NOT LIKE '_a%' AND pren_enfant NOT LIKE '_e%' AND pren_enfant NOT LIKE '_u%' AND pren_enfant NOT LIKE '_i%' AND pren_enfant NOT LIKE '_o%';
autre méthode : select distinct nom_enfant from enfant MINUS (select distinct nom_enfant from enfant where pren_enfant LIKE '_a%' or pren_enfant LIKE '_e%' or pren_enfant LIKE '_u%' or pren_enfant LIKE '_i%' or pren_enfant LIKE '_o%'

Forme algébrique : π nom_enfant - π nom_enfant (σ pren_enfant ≈ '_a*' ∧ pren_enfant ≈ '_e*' ∧ pren_enfant ≈ '_u*' ∧ pren_enfant ≈ '_i*' ∧ pren_enfant ≈ '_o*' (enfant))

**Requête 12** : Lister toutes les informations sur les entrepôts et les intermittents associés, où l identifiant de l entrepôt correspond à celui de l intermittent

Forme SQL : select * from entrepot e ,intermittent i where e.id_entrepot=i.id_entrepot;
Forme algébrique : entrepot ⋈ intermittent

**Requête 13** : Lister les noms des intermittents qui sont affectés à un entrepôt situé en Europe

Forme SQL : select DISTINCT nom_intermittent from intermittent i where id_entrepot IN (select id_entrepot from entrepot where region='Europe');
autre méthode : select DISTINCT nom_intermittent from intermittent i , entrepot e where e.id_entrepot=i.id_entrepot and region='Europe';

Forme algébrique : π nom_intermittent (intermittent ⋈ σ region= 'Europe'(entrepot));

**Requête 14** : Lister les noms des rennes qui tirent un traineau de poid 500

Forme SQL : select nom_renne from renne r , traineau t where r.id_traineau_tirer=t.id_traineau and poid=500;

Forme algébrique : π nom_renne (renne ⋈ σ poid = 500 (traineau))

**Requête 15** : Lister les noms des enfants dont la taille de l dresse ne dépasse pas 30 caractères

Forme SQL : SELECT nom_enfant FROM enfant WHERE LENGTH(adresse_enfant) <=30;

Forme algébrique : En algèbre relationnelle, la fonction `LENGTH` n'existe pas.

**Requête 16** : Lister tous les rennes (identifiant et nom) avec, s'ils existent, les traîneaux qui les concernent (qu'ils tirent)

Forme SQL : SELECT id_puce , nom_renne , id_traineau FROM renne LEFT OUTER JOIN traineau ON traineau.id_traineau=renne.id_traineau_tirer;
autre méthode :

SELECT id_puce, nom_renne , id_traineau FROM renne, traineau WHERE renne.id_traineau_tirer = traineau.id_traineau(+);

Forme algébrique : π id_puce, nom_renne , id_traineau (renne |>< traineau);

**Requête 17** : Quels sont les identifiants des intermittents qui ont participé à toutes les tournées

Forme SQL : select distinct id_intermittent from paticiper p1 where not exists ( select id_tournee from tournee minus ( select id_tournee from paticiper p2 where p2.id_intermittent=p1.id_intermittent )) ;

--autre méthode :

select distinct id_intermittent from paticiper p1 where not exists ( select * from tournee t where not exists ( select * from paticiper p2 where p2.id_tournee=t.id_tournee and p2.id_intermittent=p1.id_intermittent )) ;

--autre méthode : select id_intermittent from paticiper GROUP BY id_intermittent HAVING COUNT(distinct id_tournee)=(select count(*) from tournee);

Forme algébrique : ( π id_intermittent , id_tournee (paticiper)) / (π id_tournee(tournee))

**Requête 18 :** Quels sont les identifiants des intermittents qui ont participé à toutes les tournées d'Europe

Forme SQL : select distinct id_intermittent from paticiper p1 where not exists ( select id_tournee from tournee t where nom_tournee='Tournee Europe' minus ( select id_tournee from paticiper p2 where p2.id_intermittent=p1.id_intermittent )) ;

--autre méthode :

select distinct id_intermittent from paticiper p1 where not exists ( select * from tournee t where nom_tournee='Tournee Europe' and not exists ( select * from paticiper p2 where p2.id_tournee=t.id_tournee and p2.id_intermittent=p1.id_intermittent )) ;

--autre méthode :

select id_intermittent from paticiper where id_tournee in ( select id_tournee from tournee where nom_tournee='Tournee Europe') GROUP BY id_intermittent HAVING COUNT(distinct id_tournee)=(select count(*) from tournee where nom_tournee='Tournee Europe');

Forme algébrique : ( π id_intermittent , id_tournee (paticiper)) / (π id_tournee(σ nom_tournee='Tournee Europe' (tournee))

**Requête 19** : quel est le poids total des traineaux
Forme SQL : SELECT SUM(poid) FROM traineau ;
Forme algébrique : somme(traineau,poid)

**Requête 20** : Quelle est la somme du poids des traîneaux ayant une capacité de 8
Forme SQL : select SUM(poid) from traineau where capacite_traineau=8;

Forme algébrique : somme(σ capacite_traineau=8(traineau),poid)

**Requête 21** : Quel est le total du poids des traîneaux envoyés à chaque entrepôt
Forme SQL : SELECT s.id_entrepot, SUM(poid) FROM send s, traineau t WHERE s.id_traineau = t.id_traineau GROUP BY s.id_entrepot;

Forme algébrique : somme id_entrepot (traineau >< send , poid )

**Requête 22** : Quels sont les identifiants des traîneaux envoyés par un seul entrepôt

Forme SQL : SELECT id_traineau FROM send GROUP BY id_traineau HAVING COUNT(DISTINCT id_entrepot) = 1;

Forme algébrique : π id_traineau ( σ compte = 1 ( compte id_traineau (π id_entrepot,id_traineau(send))))

**Requête 23** : Quel est le nombre de traîneaux ayant une capacité de 8

Forme SQL : SELECT COUNT(id_traineau) FROM traineau WHERE capacite_traineau = 8;
Forme algébrique : compte id_traineau (σ capacite_traineau = 8 (traineau))

**Requête 24** : Quel est le poids minimal parmi les traîneaux ayant une capacité de 8

Forme SQL : SELECT MIN(poid) FROM traineau WHERE capacite_traineau = 8;
Forme algébrique : MIN poid (σ capacite_traineau = 8 (traineau))

**Requête 25** : Quel est le poids maximal parmi les traîneaux ayant une capacité de 8
Forme SQL : SELECT MAX(poid) FROM traineau WHERE capacite_traineau = 8;

Forme algébrique : MAX poid (σ capacite_traineau = 8 (traineau))

**Requête 26** : Quelles sont les spécialités ayant moin de 2 elfes affecté à une spécialité secondaire
Forme SQL :SELECT id_specialite FROM a_comme_specialite_secondaire GROUP BY id_specialite HAVING COUNT(id_elfe) < 2;

Forme algébrique : π id_ specialite ( σ compte < 2 ( compte id_ specialite (π id_elfe,id_ specialite (a_comme_specialite_secondaire))))

**Requête 27 :** creer une table provisoir contenat la liste des enfants avec le total des jouets qu'ils ont commandés
Forme SQL create view jouenfant as select e.id_enfant, e.nom_enfant, sum(l.qte) as somme
from enfant e , lister l where e.id_enfant = l.id_enfant
group by e.id_enfant, e.nom_enfant ;

**Requête 28** : calculer la charge de chaque traîneau afin de s'assurer qu'ils soient dans la capacité définie pour le traîneau (somme des capacités de chaque renne) ;

Forme SQL : SELECT r.id_traineau_tirer AS id_traineau, t.nom_traineau, SUM(capacite_traineau) AS charge_totale FROM renne r , traineau t where r.id_traineau_tirer = t.id_traineau GROUP BY r.id_traineau_tirer, t.nom_traineau;

Forme algébrique : π id_traineau_tirer, nom_traineau, somme(traineau,capacite_traineau) (renne ⋈ traineau)

**Requête 29 :** quels elfes s'occupent de quel traîneau/renne
Forme SQL : SELECT nom_elfe,nom_traineau,nom_renne,pos_renne FROM elfe e , entretenir en, renne r, traineau t where e.id_elfe = en.id_elfe and en.id_puce = r.id_puce and r.id_traineau_tirer = t.id_traineau;

Forme algébrique : π nom_elfe,nom_traineau,nom_renne,pos_renne ( traineau ⋈ renne ⋈ entretenir ⋈ elfe)

**Requête 30 :** quel est le joué le plus produit
Forme SQL : SELECT id_jouet, nom_jouet, COUNT(*) AS nombre_productions FROM produireAtelier pa, jouet j where pa.id_jouet = j.id_jouet GROUP BY pa.id_jouet, j.nom_jouet HAVING COUNT(*) = (SELECT MAX(COUNT(*)) FROM produireAtelier GROUP BY id_jouet) ORDER BY nombre_productions DESC;

Forme algébrique : ORDER BY n'existe pas en algèbre relationnel

**Requête 31 :** Lors d'un problème avec un jouet, il devra être possible de remonter toute la chaîne permettant
ainsi de vérifier les autres jouets issus de cette chaîne

Forme SQL :

```
SELECT
    id_jouet,
    nom_jouet,
    id_atelier,
    nom_atelier,
    nom_sstraitant,
    nom_matiere,
    nom_elfe
FROM
    produireAtelier pa
JOIN
    jouet j ON pa.id_jouet = j.id_jouet
JOIN
    atelier a ON pa.id_atelier = a.id_atelier
LEFT JOIN
    produireSousTraitant pst ON j.id_jouet = pst.id_jouet
LEFT JOIN
    sous_traitant st ON pst.id_sstraitant = st.id_sstraitant
LEFT JOIN
```

```
    fabriquer f ON j.id_jouet = f.id_cadeau
LEFT JOIN
    matiere_premiere mp ON f.id_matiere = mp.id_matiere
LEFT JOIN
    elfe e ON pa.id_atelier = e.id_equipe_se_regroupe
ORDER BY
    pa.id_atelier, j.id_jouet;
```

Forme algébrique : π id_jouet, nom_jouet, id_atelier, nom_atelier, nom_sstraitant, nom_matiere, nom_elfe ( (produireAtelier ⋈ jouet) ⋉ atelier ⋉ sous_traitant ⋉ produireSousTraitant ⋉ joue ⋉ fabriquer ⋉ matiere_premiere ⋉ jouet ⋉ elfe)

ORDER BY n'existe pas en algèbre relationnel donc on ne peut pas afficher dans l'ordre id_atelier , id_jouet

Remarque : ⋉ : left join
⋈ : join simple