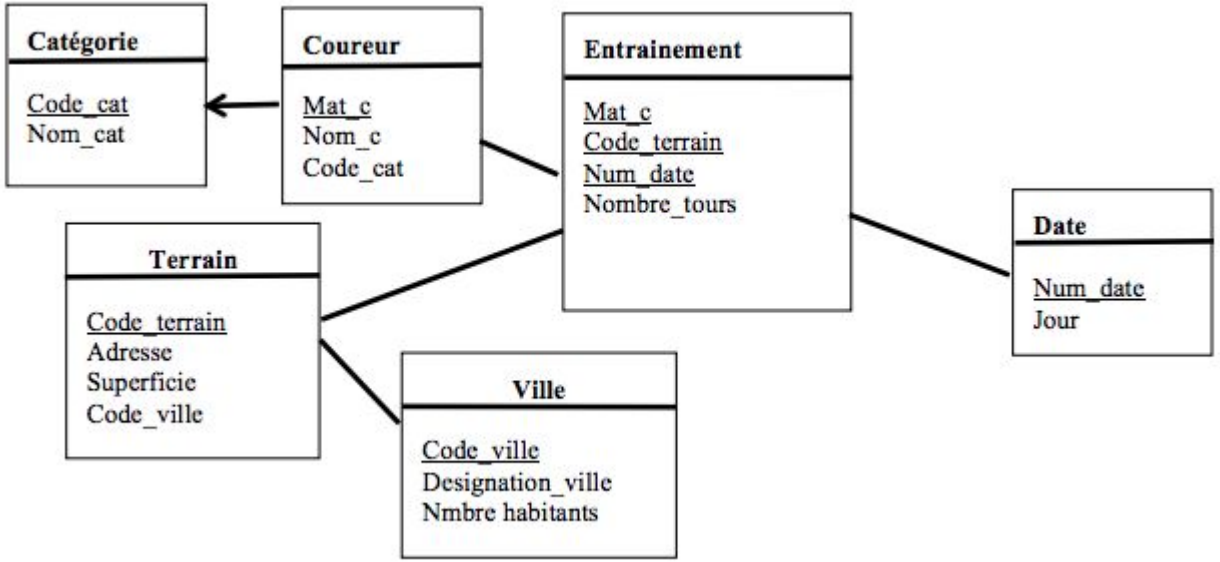
Travaux dirigés en Entrepôts de données

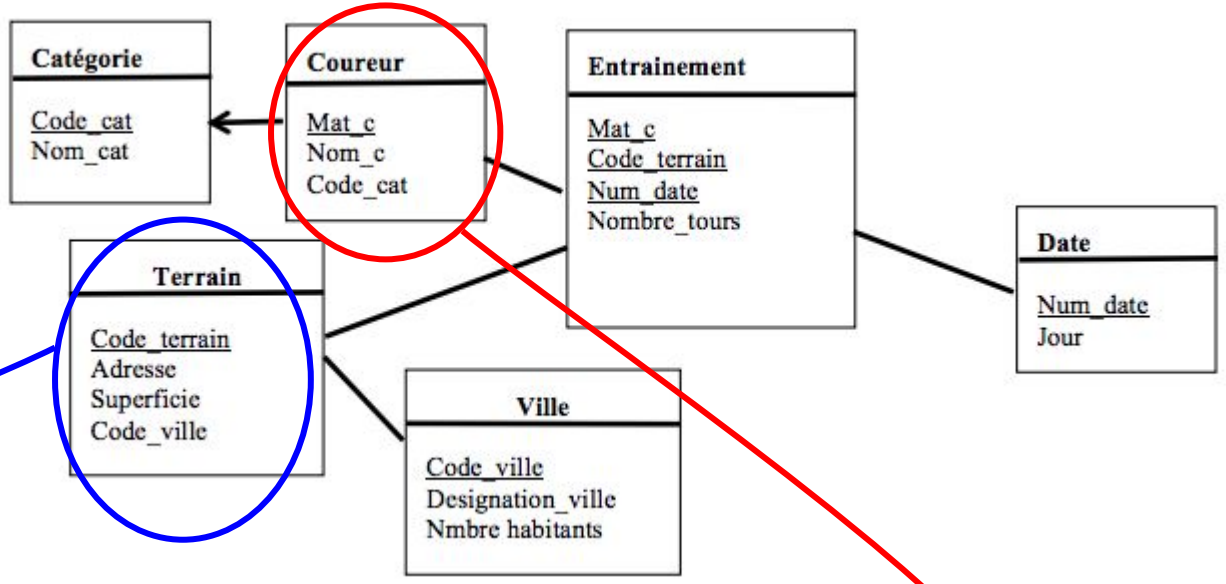
TD 2: Processus ETC

Exo 1. 1. Mapping



(a) Coureur_junior (matricule, nom, prenom, age)
(b) Coureur_espoir (mat, nom, prenom, annee_naissance)
(c) Coureur_senior (num_courreur, nom, date de naissance)
(d) Adresses (num_terrain, adresse, numero_ville)
(f) Superficie (num_terrain, superficie, num_ville)
(g) Ville (num_ville, nom_ville, nombre habitants)

Exo 1. 1. Mapping



(a) Coureur_junior (matricule, nom, prenom, age)
(b) Coureur_espoir (mat, nom, prenom, annee_naissance)
(c) Coureur_senior (num_courreur, nom, date de naissance)
(d) Adresses (num_terrain, adresse, numero_ville)
(f) Superficie (num_terrain, superficie, num_ville)
(g) Ville (num_ville, nom_ville, nombre habitants)

Exo 1. 1. Mapping

Table ED	Colonne ED	Table Source	Colonne Source	Observation
Coureur	Mat_c	Coureur junior Coureur espoir Coureur senior	Matricule Mat Num_courreur	
	Nom_c	Coureur junior Coureur espoir Coureur senior	Nom, prenom Nom, prenom Nom	A concaténer A concaténer
	Code_cat	Coureur junior Coureur espoir Coureur senior	/ / /	A déduire “C1” A déduire “C2” A déduire “C3”
Terrain	Code_terrain	Adresses Superficie	Num_terrain Num_terrain	
	Adresse	Adresses	Adresse	
	Superficie	Superficie	Superficie	
	Code_ville	Adresses Superficie	Numero_ville num_ville	

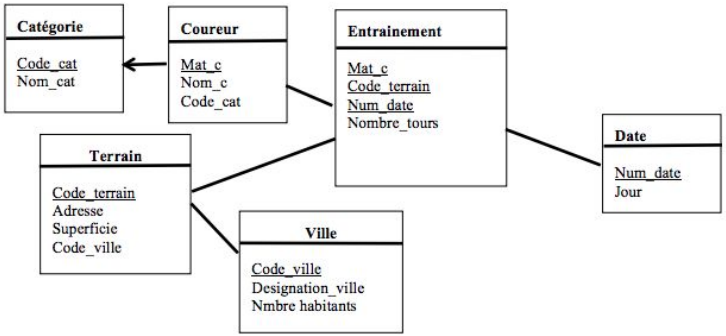
Exo 1. 2. Algèbre relationnelle

Table Coureur

Puisque la table coureur contient les trois types de coureurs qui sont de catégories différentes, on doit les UNIR. Puisque les schémas ne sont pas les mêmes, on doit d'abord unifier le schéma en écartant les attributs age, année_naissance, et date naissance.

On doit aussi ajouter le code de catégorie de chaque coureur avant d'insérer les données.

Enfin, on doit concaténer les noms et prénoms des juniors et espoirs.



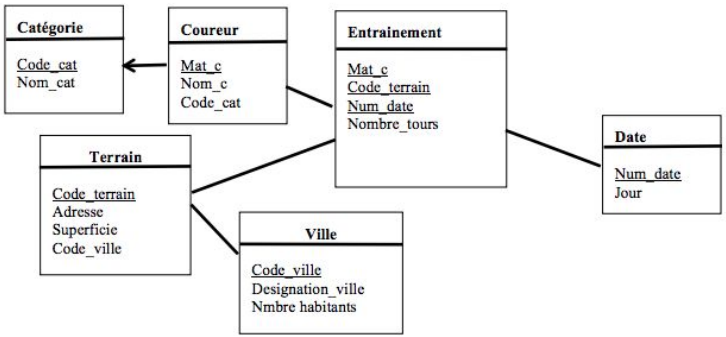
(a) Coureur_junior (matricule, nom, prenom, age)
(b) Coureur_espoir (mat, nom, prenom, annee_naissance)
(c) Coureur_senior (num_courreur, nom, date de naissance)
(d) Adresses (num_terrain, adresse, numero_ville)
(f) Superficie (num_terrain, superficie, num_ville)
(g) Ville (num_ville, nom_ville, nombre habitants)

Exo 1. 2. Algèbre relationnelle

Table Coureur

R1 = π (matricule, concat(nom, prenom), ‘C1’ / coureur_junior)
R2 = π (mat, concat(nom, prenom), ‘C2’ / coureur_espoir)
R3 = π (num_coureur, nom, ‘C3’ / coureur_sénior)

Coureur = R1 ∪ R2 ∪ R3



(a) Coureur_junior (matricule, nom, prenom, age)
(b) Coureur_espoir (mat, nom, prenom, annee_naissance)
(c) Coureur_senior (num_courreur, nom, date de naissance)
(d) Adresses (num_terrain, adresse, numero_ville)
(f) Superficie (num_terrain, superficie, num_ville)
(g) Ville (num_ville, nom_ville, nombre habitants)

Exo 1. 2. Algèbre relationnelle

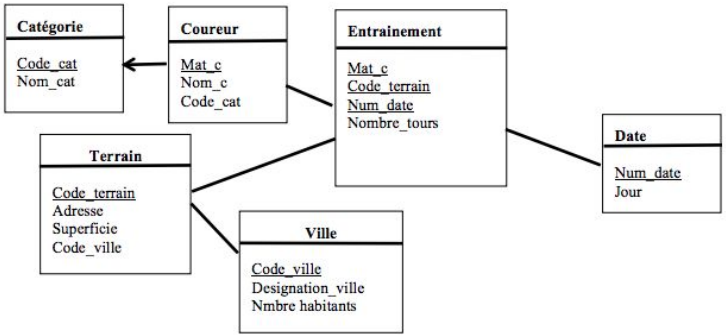
Table Terrain

Puisque les données dans la table Adresses et la table Superficie se rapportent aux mêmes terrains, dans ce cas, nous devons JOINDRE les tables

Donc:

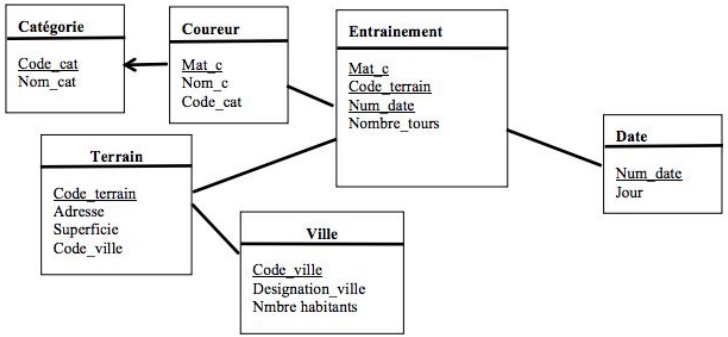
Terrain = Adresses ⋈ Superficie
(adresses.num_terrain = superficie.num_terrain

On peut ajouter une projection pour ne prendre que les colonnes nécessaires



(a) Coureur_junior (matricule, nom, prenom, age)
(b) Coureur_espoir (mat, nom, prenom, annee_naissance)
(c) Coureur_senior (num_courreur, nom, date de naissance)
(d) Adresses (num_terrain, adresse, numero_ville)
(f) Superficie (num_terrain, superficie, num_ville)
(g) Ville (num_ville, nom_ville, nombre habitants)

Exo 2. 1. Changements dans les sources



(a) Coureur_junior (matricule, nom, prenom, age)
(b) Coureur_espoir (mat, nom, prenom, annee_naissance)
(c) Coureur_senior (num_courreur, nom, date de naissance)
(d) Adresses (num_terrain, adresse, numero_ville)
(f) Superficie (num_terrain, superficie, num_ville)
(g) Ville (num_ville, nom_ville, nombre habitants)

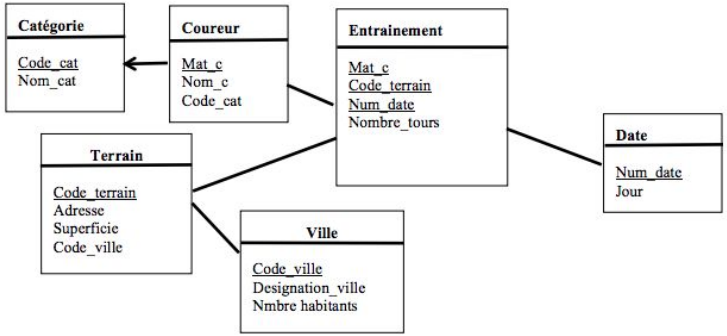
a) Ajout d'un nouveau coureur espoir. → Oui

b) Modification de l'âge d'un coureur junior → Non, vu que l'age n'est pas utilisé dans la table Coureur

c) Suppression d'un terrain (table adresses → Oui, mais on ne supprime pas, on peut ajouter un attribut dans Terrain pour savoir que le terrain n'est plus utilisé

d) Modification du nombre d'habitants d'une ville → Oui, mais on ne met pas à jour l'entrepöt, on historise l'évolution du nombre d'habitants

Exo 2. 1. Répercussion des changements



(a) Coureur_junior (matricule, nom, prenom, age)
(b) Coureur_espoir (mat, nom, prenom, annee_naissance)
(c) Coureur_senior (num_courreur, nom, date de naissance)
(d) Adresses (num_terrain, adresse, numero_ville)
(f) Superficie (num_terrain, superficie, num_ville)
(g) Ville (num_ville, nom_ville, nombre habitants)

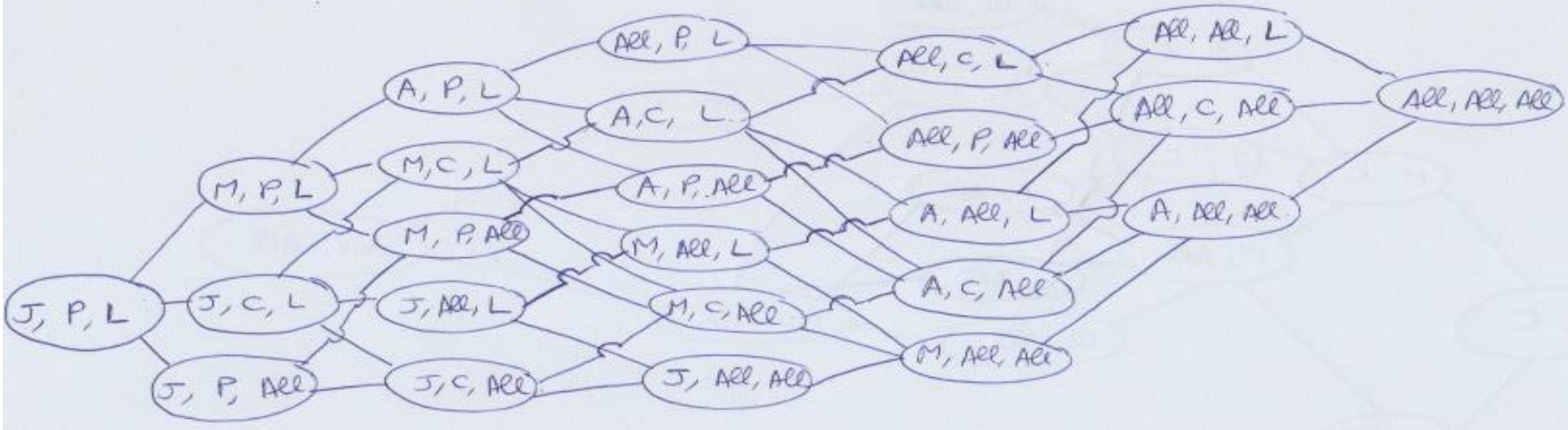
Évènement	Action
ON INSERT INTO Coureur_espoir	INSERT INTO Coureur
ON DELETE FROM adresses	UPDATE Terrain
ON UPDATE ville ...	INSERT INTO evolution_habitants

Update ville (on ajoute la dernière valeur
dans la table ville et la valeur écrasée dans la table evolution

Travaux dirigés en Entrepôts de données

TD 3: OLAP

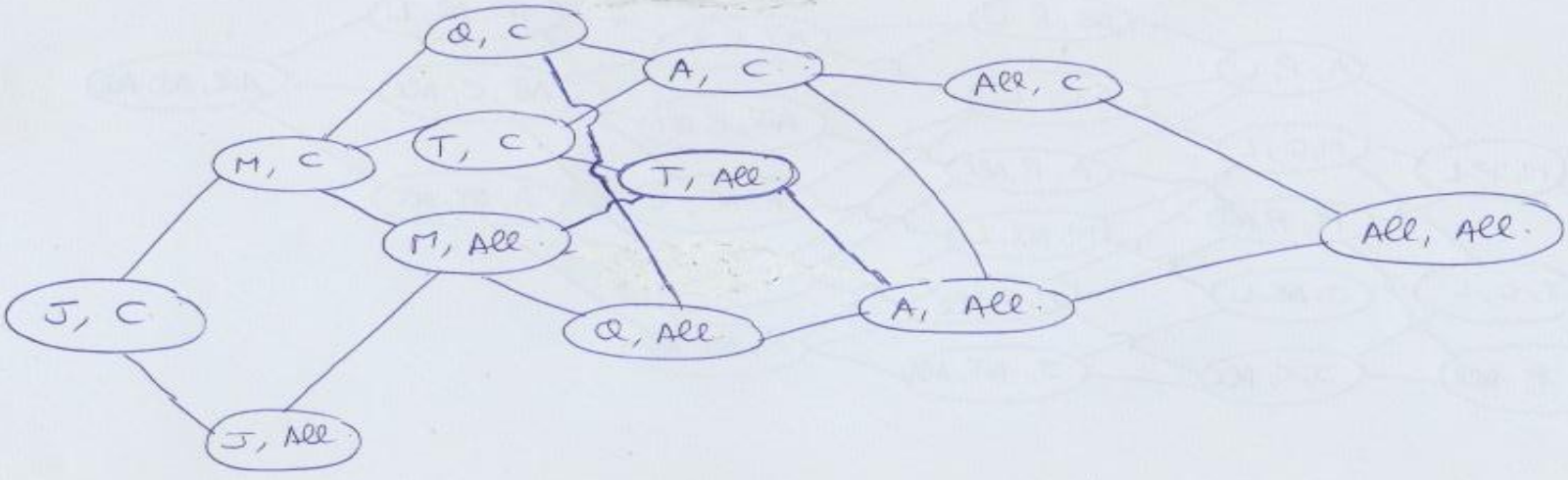
Exo 1 1) Treillis des Cuboids [J → Jour / M → Mois / A → Année / P → Progiciel / L → client / C → catégorie]



Le nombre de noeuds est 24

$$24 = \underbrace{J, M, Année, All}_{4} \times \underbrace{Progiciel, catégorie, All}_{3} \times \underbrace{client, All}_{2}$$

Exo 1 / 2 M → Machine / J → Jour / T : Trimestre / Q : Quadrimestre / M → Mois / A : Année



Exo2

- nombre d'appels / Mois / client / catégorie

		cat1					cat2						
		C_1	C_2	C_3	C_4	C_5	C_1	C_2	C_3	C_5	C_6	C_7	C_8
2010	Jan	21	57	33	36	108	24	68	43	85	24	35	47
	Fev	22	62	17	47	40	43	78	100	32	28	26	38
	Mar	15	20	9	14	17	20	22	21	6	8	24	10
2011	Jan	1	12	5	65	40	50	15	72	55	43	19	10
	Fev	10	16	60	12	30	15	10	20	23	25	10	35
	Mar	88	111	31	47	99	82	58	67	59	134	136	20
	AVR	10	15	62	32	65	11	2	44	45	12	44	30

- nbr app / Année, client, catégorie ⇒ ... group by Année, client, catégorie

{ Jointe : les 7 tables }

- // // / // , catégorie ⇒ ... group by Année, catégorie

{ Jointe Appel, Jour, Mois, Année, Progiciel, Catégorie }

- // // / catégorie ⇒ ... group by catégorie

{ Jointe Appel, Progiciel, catégorie }

- // // / progiciel ⇒ ... group by progiciel

{ Jointe Appel, progiciel }

- // // / client ⇒ ... group by client

{ Jointe Appel et client }

SQL

```
Select SUM(nbr_appel), M.Mois, C.cat_name, L.client
FROM Jour J, Mois, M, client L,
     Appel A, Progiciel P, Catégorie C
where A.code_P = P.code_P AND
P.code_cat = C.code_cat AND
  A.num_Jour = J.num_JOUR
  AND J.num_mois = M.num_Mois
  AND A.num_client = L.num_client
Group by M.MOIS, C.Catégorie,
  L.client
```

nbre d'appel des clients C1 et C3 durant le mois de Janvier 2010 du progiciel

⇒ ... group by client, progiciel, Mois avec Condition where / Mois et progiciel et clients

1) Operateurs OLAP

en partant du cube de base $C_1 = (J, P, L)$

1) $C_2 = \text{RollUp}(C_1 / J)$, $C_3 = RU(C_2 / \text{progiciel}) \Rightarrow (M, C, L)$

2) $C_4 = RU(C_3 / \text{Mois}) \Rightarrow (A, C, L)$

3) $C_5 = RU(C_4 / \text{client}) \Rightarrow (A, C, All)$

4) $C_6 = RU(C_5 / \text{Annee}) \Rightarrow (All, C, All)$

5) $C_7 = DD(C_6 / \text{categorie}) \Rightarrow (All, P, All)$

6) $C_8 = RU(C_7 / \text{progiciel}) \Rightarrow (All, C, All) \Rightarrow C_9 = RU(C_8 / \text{categorie}) = (All, All, All)$

$C_{10} = DD(C_9 / \text{client}) = (All, All, L)$

7) $C_{11} = DD(C_{10} / \text{All Annee}) = (A, All, L)$

$C_{12} = DD(C_{11} / \text{Annee}) = (M, All, L)$

$C_{13} = DD(C_{12} / \text{All cat}) = (M, cat, L)$

8) $C_{14} = DD(C_{13} / cat) = (M, Pro, L)$

$C_{15} = \text{Slice \& Dice}\ (C_{14}, \text{LIN}(C_1, C_7)\ \&$
$\text{Mois} = 01/2010\ \&$
$\text{Progiciel} = \text{'Progiciel 1'})$

2) Soit C'_1 le cube présenté dans la série :

1) on fait une rotation [Rotate (C'_1 / Temps)]

2) on fait une rotation de C'_1 [Rotate (C'_1 / client)]

3) on fait un "Push" de date dans les mesures

Travaux dirigés en Entrepôts de données

TD 4: Administration d’un ED

Série n° 4 en ED

Exo 1

1)

	R_1	R_2	R_3	R_4	R_5	R_6
R_1	/	$\not\subset$	$\not\subset$	$\not\subset$	$\not\subset$	$\not\subset$
R_2	$\subset$	/	$\not\subset$	$\not\subset$	$\not\subset$	$\not\subset$
R_3	$\subset$	$\not\subset$	/	$\not\subset$	$\not\subset$	$\not\subset$
R_4	$\not\subset$	$\not\subset$	$\not\subset$	/	$\not\subset$	$\not\subset$
R_5	$\not\subset$	$\not\subset$	$\tilde{\subset}$	$\not\subset$	/	$\not\subset$
R_6	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	/

2) L'ensemble minimal : les requêtes non incluses dans aucune requête

R_1, [illegible] R_4, R_5

EX02

1) Index BITMAP / Prix

	500	4000	30000	3000
P_1	1	0	0	0
P_2	0	1	0	0
P_3	0	1	0	0
P_4	1	0	0	0
P_5	0	0	1	0
P_6	0	0	0	1

Index BITMAP / Cat-ID

	C_1	C_2	C_3
P_1	1	0	0
P_2	0	1	0
P_3	0	1	
P_4	0	0	1
P_5	0	0	1
P_6	0	0	1

Pour trouver le nombre de produits de la deuxième catégorie dont le prix est ≠ 500 DA

on fait la somme de la colonne resultant du ET logique entre $\overline{500}$ et C_2. ⟹ on trouve "2"

2) Index de jointure sur la table Date pour exemple regrouper les données par mois

- Index de jointure sur la table Produit pour regrouper les donnée par Cat-ID

3) on peut créé un index de jointure sur la colonne "Prix" dans la table "Vente" et on fait la somme de valeur de mesure qui correspondent "1" dans la colonne "30000"

qui est la dernière colonne si l'index est ordonné