DON

INTRODUCTION

Ne jamais stocker une valeur qui peut être rapidement calculé.

Base commande contient : IdCommande, IdClient, Date. C'est le schéma.

Un null n'est pas égal à un null car c'est indéterminé, ça n'a pas de valeur. Représenté par <null> ou par rien.
Problème : plusieurs interprétations possibles :

- Information pertinente mais inexistante pour l'entité.
- Information non pertinente pour cette entité.
- Information existante mais actuellement inconnue.

Utiliser des valeurs sentinelles plutôt que des null.

Un identifiant c'est, dans une table, les champs qui ne peuvent pas être identique (pas primaire, aurait pu être primaire mais on ne l'a pas choisi.). Par exemple la plaque de voiture si le n° de châssis est en primaire, mais ça peut être plusieurs « colonne » par exemple plaque voiture et n°client. Détermine toutes les colonnes de la ligne d'une table.
Un identifiant est un déterminant qui détermine l'ensemble des attributs d'une table. (Aurait pu être identifiant primaire).
Identifiant c'est un ensemble d'attribut qui va déterminer tous les attributs de la table.

Identifiant minimal, si dans l'identifiant, lorsque l'on retire 1 colonne, ça n'est plus un identifiant.

Identifiant primaire, c'est un identifiant que l'on choisit. Par défaut, c'est toute la ligne, problème par exemple si ça sert pour se loguer il faudra TOUT entrer (nom, prénom, …). Peut pas être null, on doit lui attribuer une valeur. Permet de retrouver 1 ligne précise d'une table.

Identifiants secondaires = tous les identifiants pas primaires.

Clé étrangère, c'est l'identifiant primaire d'une autre table. Une clé étrangère référence l'identifiant primaire.

Identifiant ≠ clé. (La seul clé qu'on peut utiliser est étrangère.)

Contrainte référentielle : la clé étrangère doit exister dans une autre table. Cad que si dans la table client l'IdClient est utilisé comme clé étrangère dans la table commande, dans la table commande il ne peut pas y avoir un IdClient qui n'existe pas dans la table client.

Si une des colonnes d'une clé étrangère est facultative, il est recommandé de les rendre toutes facultatives.

Il se peut qu'une clé étrangère soit également un identifiant.

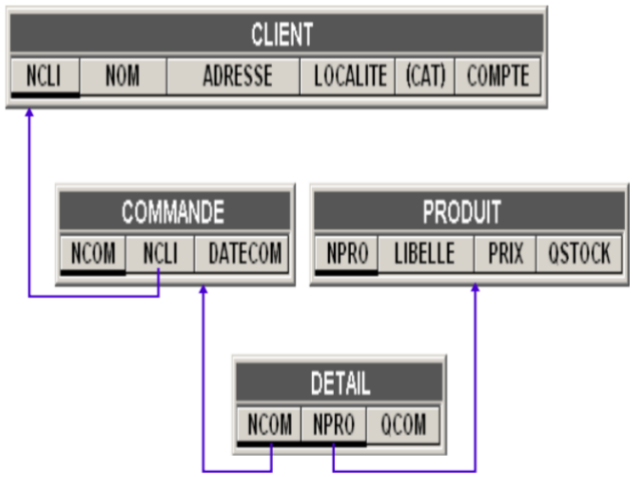
DON

Il se peut que les colonnes d'une clé étrangère appartiennent, en tout ou en partie, à un identifiant.

Un identifiant minimal est aussi appelé clé candidate (candidate key).
Un identifiant primaire s'appelle aussi clé primaire (primary key).
Il n'existe pas d'autre terme pour désigner les identifiants secondaires.
Clé étrangère = foreign key

Le contenu d'une table est sujet à de fréquentes modifications. Le schéma d'une table peut évoluer mais moins fréquemment (difficile).

Exemple BDD :



Ne pas mettre dans une table des informations dérivables d'autres infos déjà présente dans la table, cad que dans la table commande, on met le IdClient, on ne va pas ajouter aussi le nom du client.

Un SGBD s'occupe d'organiser la BDD. Il vérifie tout à notre place.

Différent logiciel :

- SQLite - sgbd publique - multiplateforme
- Oracle – sgbd propriétaire – Linux, Windows, ...
- PostGreSQL – sgbd libre et open source – Unix like - Windows
- Access – sgbd propriétaire - Windows

A part SQLite, ils ont besoin d'un serveur.

DON

SCHEMA CONCEPTUALISE

Redondance internes

Problèmes de la table ci-dessous :

- Gaspillage d'espace
- Si on modifie la valeur d'un titre, il faut répercuter cette modification dans toutes les lignes similaires
- Si on supprime l'unique exemplaire d'un livre, on perd les informations sur son auteur et son titre
- Est-on certain que le titre et l'auteur ont été orthographiés exactement de la même manière pour tous les exemplaires d'un livre ?

LIVRE					
NUMERO	TITRE	AUTEUR	ISBN	DATE_ACHAT	EMPL
1029	L'humanité perdue	Finkelkraut A.	2 02 033300 7	14/10/2008	F3
1030	L'humanité perdue	Finkelkraut A.	2 02 033300 7	14/10/2008	F3
1032	Mercure	Nothomb A.	2 253 14911 X	14/10/2008	G5
1045	Eva Luna	Allende I.	2 253 05354 6	22/2/2009	F3
1067	Mercure	Nothomb A.	2 253 14911 X	24/2/2009	G5
1022	Mercure	Nothomb A.	2 253 14911 X	3/10/2008	G6

Pas faire comme le tableau du dessus car redondance. On peut rassembler les données commune (ISBN,TITRE,AUTEUR) dans une nouvelle table.

OUVRAGE		
ISBN	TITRE	AUTEUR
2 02 033300 7	L'humanité perdue	Finkelkraut A.
2 253 14911 X	Mercure	Nothomb A.
2 253 05354 6	Eva Luna	Allende I.

EXEMPLAIRE			
NUMERO	ISBN	DATE_ACHAT	EMPL
1029	2 02 033300 7	14/10/2008	F3
1030	2 02 033300 7	14/10/2008	F3
1032	2 253 14911 X	14/10/2008	G5
1045	2 253 05354 6	22/2/2009	F3
1067	2 253 14911 X	24/2/2009	G5
1022	2 253 14911 X	3/10/2008	G6

Dépendance fonctionnelle

On détecte et corrige les redondances grâce à une contrainte d'intégrité : la dépendance fonctionnelle.
Si deux lignes ont la même valeur de ISBN, alors elles ont aussi les mêmes valeurs de TITRE et d'AUTEUR. On dit que :

- Il existe une dépendance fonctionnelle de ISBN vers TITRE et AUTEUR
- ISBN détermine ou est un déterminant de TITRE et AUTEUR
- TITRE et AUTEUR dépendent de ou sont déterminés par ISBN

Ça s'écrit ISBN → TITRE, AUTEUR
Déterminant est à gauche de la flèche et à droite c'est le déterminé.

Si A défini B, B ne définit pas spécialement A. Exemple : ISBN → TITRE, AUTEUR mais TITRE, AUTEUR ⇏ ISBN car ≠ édition.
ISBN→p({TITRE,AUTEUR}) (cad ISBN → TITRE et ISBN → AUTEUR)
NUMERO → p{TITRE,AUTEUR,ISBN,DATE,EMPL}

Si A→B et que B→C => A→C

DON

Il y a redondance interne dès qu'il existe un déterminant qui n'est pas un identifiant de la table.
ISBN est un déterminant dans LIVRE mais il n'en est pas un identifiant. Il entraîne donc des redondances internes.

Une dépendance fonctionnelle dont le déterminant n'est pas un identifiant est dite anormale, non-normalisé. Genre ISBN → TITRE, si ISBN pas identifiant→anormal

Mon schéma est normalisé est-ce que l'attribut A est déterminant ? Oui car normalisé.

Dans ISBN j'ai de la redondance, est-ce qu'elle est normalisée ? Pour vérifier qu'elle est normalisée, il faut montrer qu'il y a une dépendance fonctionnelle. Il n'y a pas de dépendance fonctionnelle. ISBN pas identifiant.

Pour normaliser une table, il faut créer une nouvelle table avec le déterminant, tant qu'il y a une dépendance fonctionnelle. Cad, dans chaque colonne on se demande « Si 2 valeurs sont identiques, est-ce que ces 2 lignes auront d'autres valeurs identiques ? » Si oui, créer une nouvelle table avec cette colonne.
Remarques :

- Une table qui est le siège d'une dépendance fonctionnelle anormale est dite non normalisée.
- Une table sans dépendance fonctionnelle anormale est dite normalisée.
- Décomposer une table de manière à éliminer ses dépendances anormales consiste à normaliser cette table.
- Il est essentiel que toutes les tables d'une base de données soient normalisées.
- Il est possible qu'une table qui est le siège de dépendances fonctionnelles anormales ne comporte pas de redondance à certains moments. Il ne s'agirait que d'un accident statistique ! Inutile de tenter le diable !

Identifiants

Rappel : Un identifiant est un déterminant qui détermine l'ensemble des attribue d'une table. (Aurait pu être identifiant primaire).

Propriétés :

- Identifiant minimal : lorsque l'on retire 1 colonne le l'identifiant il ne l'est plus.
- Si dans un ensemble d'attributs qui sont identifiants, on rajoute 1 attribut il sera d'office identifiant. (Mais pas minimal).
- L'ensemble des attributs d'une relation/table est 1 identifiant car se détermine lui-même.
- Plusieurs identifiants minimaux peuvent coexister dans une relation/table (n° de châssis et plaque immatriculation).
- Un attribut peut appartenir à plusieurs identifiants (le n° de livre est n identifiant et le n° de livre + autre chose est aussi un identifiant).
- Il est possible de calculer automatiquement les identifiants d'une relation ()

DON

Dépendance fonctionnelle

- Contrainte d'intégrité très importante du modèle relationnel.
- Proche de l'identifiant mais plus précis.
- Á la base de la théorie de la normalisation.

On note : ACHAT:PRODUIT → PRIX car dans la table ACHAT.
PRODUIT est un identifiant de ACHAT[PRODUIT, PRIX]

Dans une relation R(A,B,C,D), il existe une dépendance fonctionnelle A→B si, à tout instant, deux lignes de R qui ont même valeur de A ont aussi même valeur de B.
Cad, si dans A on a 2 valeurs identiques, aux mêmes lignes, il y aura aussi 2 valeurs identique pour B.
Déterminant et déterminé peuvent être multicomposants B→C,D B,C→D.

Si on trouve une dépendance fonctionnelle qui détermine un sous ensemble des attribut d'une table et qui n'est pas déterminant il faut la splitter.

Relation : comme une table, un ensemble de données.

Dans une table, si ce n'est pas obligatoire on met [0-1].

SQL - SELECT SUR UNE TABLE

Le sous-langage DML (Data Manipulation Language) de SQL permet de consulter le contenu des tables et de les modifier. Il comporte 4 verbes.
Create: La requête insert insère de nouvelles lignes dans une table
Read : La requête select extrait des données des tables
Update : La requête update modifie les valeurs de colonnes de lignes existantes
Delete : La requête delete supprime des lignes d'une table

Toute requêtes select envoie un résultat sous forme de table.

Le langage SQL se base sur l'algèbre relationnelle

Algèbre relationnelle

Deux concepts de base : • le domaine de valeurs = ensemble prédéfini de valeurs simples • la relation = partie du produit cartésien de domaines

Projection

Projection : sélectionner les colonnes que l'on souhaite garder.

SQL:

```
select NCLI, NOM
from CLIENT ;
```

```
select distinct LOCALITE
from CLIENT
```

Pour avoir sans doublon. ➔ retourne un ensemble.
Après chaque requêtes SQL ➔ un ;

Relationnelle :

RELATION[liste d'attributs]
CLIENT[COMPTE]

DON

Sélection

Le where c'est pour sélectionner des lignes.

SQL :

```
select *
from CLIENT
where CAT = 'B2';
```

Relationnelle :

Relation(condition)
CLIENT(CAT='B2')
Si on veut mettre les 2 ensemble, d'abord () sélection puis [] projection.

Null

Null n'est pas égal à un nul, pour chercher un null il faut faire « CAT is null ; » tandis que pour savoir si ce n'est pas null « CAT is not null ; »

In - Ont créé une liste, ce qu'il y a entre parenthèse.

Where CAT in ('C1','C2','C3') ; s'il est dedans.
where LOCALITE not in ('Toulouse','Breda'); s'il n'est pas dedans.

Between

where COMPTE between 1000 and 4000; ➔ compris

Masque

_ ➔ remplace 1 caractère seulement.
where CAT like 'B_'; ➔ Bx, BI, …

% ➔ remplace plusieurs caractères.
where LIBELLE like '%SAPIN%'; blablaSAPINblabla

_%SAPIN% prend tous les mots SAPIN qui ne commence pas par SAPIN.
like'%Z%' and not like'Z%'; qui ne commence pas par Z mais en contient un.
Opérateur logique : or, and

Données extraites et données dérivées

Extraite : les données disponible dans la table.
Dérivée : données qui se base sur les données de la table.

select 'TVA de ', NPRO, ' = ' , 0.21*PRIX*QSTOCK
La virgule c'est pour la concaténation.

TVA de	NPRO	=	0,21*PRIX*QSTOCK
TVA de	CS264	=	67788
TVA de	PA45	=	12789
TVA de	PH222	=	37770.6
TVA de	PS222	=	47397

Création d'un alias :

```
select NPRO as Produit, 0.21*PRIX*QSTOCK as Valeur_TVA
  from PRODUIT
  where QSTOCK > 500;
```

Avantage d'utiliser un as : pour simplifier les requêtes

DON

Les fonctions agrégatives

COUNT, SUM, MIN, MAX, AVG(average)
Count : compte le nbre de ligne.
Sum : fais la somme.
Avg : la moyenne

Les valeurs nulle ne sont pas reprises par les fonctions agrégatives

Select ncli,count(*) from CLIENT ➔ c'est faux car il affiche ncli et il en prend un au hasard, normalement réponse de cette requête = erreur.

Le 1er n'a pas de sens. Le 2ème affiche le nbre de client qui ont commandé.

```
select distinct count(NCLI)
from   COMMANDE;
```

count(NCLI)
7

```
select count(distinct NCLI)
from   COMMANDE;
```

count(NCLI)
5

select count(*) from client ➔ si on ne sait vraiment pas quel est l'identifiant primaire.

JOINTURE ET DONNÉES GROUPÉES

R1	A	B
	A1	B1
	A2	B2

R2	C	D
	B1	D1
	B1	D2

R1XR2	A	B	C	D
	A1	B1	B1	D1
	A1	B1	B1	D2
	A2	B2	B1	D1
	A2	B2	B1	D2

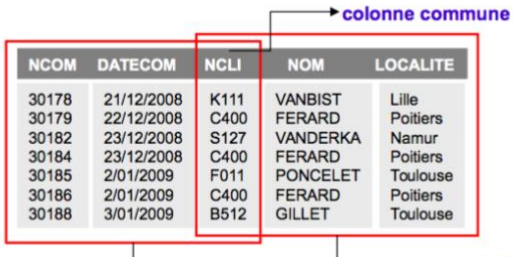
Une jointure c'est un peu +, on veut sélectionner 2 ligne =.
Si on a 2 ligne, leur produit cartésien fais 2x2.

Jointure = de 1 le produit cartésien et de 2 la projection.
SELECT * from R1,R2 ; ➔ c'est JUSTE le produit cartésien.
SELECT * from R1,R2 where B=C; prob : si 1000 lignes, il fait le prod cartésien.

SELECT * from R1 join R2 on B=C where D=d1 ; ➔ ne fournit pas le produit cartésien, avec join vérifie avant de produire.

La jointure permet de produire une nouvelle table constituée de données extraites de plusieurs tables.

Exemple :
select NCOM, DATECOM, CLIENT.NCLI, NOM, LOCALITE (car NCLI dans les 2)
from COMMANDE join CLIENT on COMMANDE.NCLI = CLIENT.NCLI;



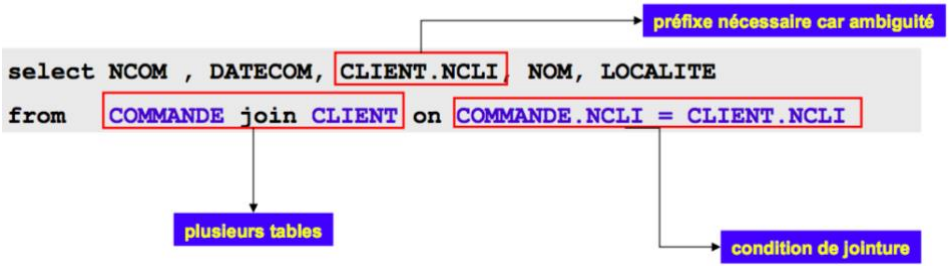
DON

Algèbre relationnelle

Produit cartésien : R1 X R2
Jointure : Join(R1, R2 ; condition)

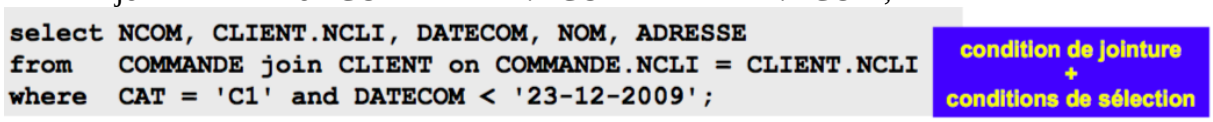
SQL

Select A,D from R1 join R2 on R1.B=R2.B where D=d1 ;
Relationnelle : Join(R,R2(D=d1), R1.B=R2.B)[A,D]
D'abord on calcul D=d1 puis on fais la jointure.



Infos

On peut faire une jointure de 3 table.
CLIENT join COMMANDE on CLIENT.NCLI = COMMANDE.NCLI
join DETAIL on COMMANDE.NCOM = DETAIL.NCOM;



La requête :
select NCOM, CLIENT.NCLI, DATECOM, NOM, LOCALITE
from COMMANDE join CLIENT on COMMANDE.NCLI = CLIENT.NCLI;
Ignore les lignes de CLIENT qui n'ont pas de lignes correspondantes dans COMMANDE.
Ces lignes de CLIENT sont dites célibataires.

from DETAIL D join PRODUIT P on D.NPRO = P.NPRO D et P sont alias.

On peut faire une jointure de 2 tables qui n'ont pas de clé étrangère.

Les données groupées

Principe

```
select LOCALITE,
      count(*) as NOMBRE_CLIENTS,
      avg(COMPTE) as MOYENNE_COMPTE
from   CLIENT
group by LOCALITE;
```

LOCALITE	NOMBRE_CLIENTS	MOYENNE_COMPTE
Bruxelles	1	0.00
Geneve	1	0.00
Lille	1	720.00
Namur	4	-2520.00
Paris	1	0.00
Poitiers	3	533.33
Toulouse	5	-2530.00

le groupe des clients de Genève
le groupe des clients de Namur
le groupe des clients de Poitiers

On s'intéresse aux localités et non plus aux clients

DON

Algèbre relationnel

Agrégation = max, count, avg, …

Agregat(Relation ;
liste d'attributs pour le groupement ;
liste de fonctions/attributs affichés)

S'utilise sur une liste.

Interdit de mettre un champ dont on a pas fait d'agrégat.
On ne peut jamais mettre 2 même valeurs dans un champ.
Les autres sont les listes.

Agregat (R; y; y, t, Count(*))
Erreur de syntaxe !!!!!!
et même de sens!
car on a fait agrégation sur y et on veut afficher t, t ne fais pas de l'agrégation donc c'est une liste.

Tout ce qui est à droite doit être à gauche mais l'inverse non.

SQL

Il est interdit d'avoir qqchsose dans group by et qui n'est pas dans select on ne peut pas avoir qqchose qui n'est pas dans group by et qui est dans select.

```
select LOCALITE, count(*), avg(COMPTE)
from   CLIENT
where
group by LOCALITE
having count(*) >= 3;
```

Ce qui peut se faire dans le where et ce qui peut se faire dans le having.
Parfois si dans where ou dans having ➔ même réponse.

Le having c'est une sélection mais post agrégation.

Where c'est une sélection des lignes.
Having est une sélection des groupes.

Group by on peut faire sur plusieurs attribut, on peut aussi fais sur un élément dérivé.

Si on peut faire une sélection dans le where ou dans le having (après group by), on doit faire avec where.

SQL ordre :

- SELECT
- FROM
- Join
- WHERE
- GROUP BY
- HAVING

DON

Exercice

Algèbre relationelle :

agregat(client(compte>1000);localité;localité, count(*)c)(c>=2)[localité]

En français :

Affiche la localité si minimum 2 compte sont > 1000

SQL :

```
Select localité
from client
where compte > 1000
group by localité
having count(*)>=2
```

Avoir un where avant de faire un join

```
SELECT M.ncli, Count(*)
FROM(select * from detail where npro = 'PA45')D
Join commande M ON M.ncom = D.ncom
GROUP BY m.ncli
HAVING count(*)≥2;
```

Ordonné

order by LOCALITE, CAT;
ordonné par localité puis par cat

DON

LES SOUS REQUÊTES

Liste : 1 attribut avec plusieurs valeurs.

Condition d'association

On préfère imbriquer que de faire un JOIN.
Les numéros de commandes des clients habitants à Namur :

```
SELECT ncom, datecom
FROM   commande
WHERE  ncli   IN
           (SELECT  ncli
            FROM    client
            WHERE   localité = 'Namur');
```

On peut mettre une infinité d'imbrications.

```
SELECT *
FROM   produit
WHERE  npro IN
       (SELECT npro
        FROM   detail
        WHERE  ncom IN
               (SELECT ncom
                FROM   commande
                WHERE  ncli IN
                       (SELECT ncli
                        FROM   client
                        WHERE  localite = 'Namur')));
```

→ Clients de Namur
→ Commandes des clients de Namur
→ Détails des commandes des clients de Namur
→ Les produits référencés par les détails des commandes des clients de Namur

Une condition IN (sous-requête) correspond le plus souvent à une condition d'association.

Exemple :

Quelles sont les clients associés à une commande passée le 12-9-2017 ?

```
SELECT *
FROM   client
WHERE  ncli IN (SELECT ncli
                FROM   commande
                WHERE  datecom = '12-9-2017');
```

Quelles sont les commandes qui sont associées à un client habitant à Namur ?

```
SELECT *
FROM   commande
WHERE  ncli IN (SELECT ncli
                FROM   client
                WHERE  localite = 'Namur');
```

DON

Référence multiples

Exemple :

```
SELECT ncli, nom, localite, compte
FROM   client AS c  = client c  le as n'est pas obligatoire, mais + propre
WHERE  compte > (SELECT AVG(compte)
                 FROM   client
                 WHERE  localite = c.localite);
```

Affiche les info des client qui on + sur le compte que la moyenne des gens de leur localité.

```
SELECT  ncom, datecom, ncli
FROM    commande AS c
WHERE   (SELECT COUNT(*)
         FROM   detail
         WHERE  ncom = c.ncom) >= 3;
```

Les commande pour lequel + de 3 produits ont été commandé.

Les quantificateurs ensemblistes

Exists & not exists

Quels sont les produits pour lesquels il existe au moins un détail ?

```
SELECT nrpo, libelle
FROM   produit AS p
WHERE  EXISTS(SELECT *
              FROM   detail
              WHERE  nrpo = p.nrpo);
```

ALL&ANY

Le n° de commande pour laquelle la quantité de PA60 est le minimum.
Pour la quantité max : qcom ≥ ALL.

```
SELECT DISTINCT ncom
FROM   detail
WHERE  qcom <= ALL (SELECT qcom
                    FROM   detail
                    WHERE  nrpo = 'PA60')
       AND     nrpo = 'PA60';
```

OU

```
SELECT DISTINCT ncom
FROM   detail
WHERE  qcom = (SELECT MIN(qcom)
               FROM   detail
               WHERE  nrpo = 'PA60')
       AND     nrpo = 'PA60';
```

Le distinct est utile pour éviter qu'il y ai 2 fois le même n° de commande.
C'est un ET sur chaque élément de la liste. (qcom ≤ 20 ET 30 ET …, si tout vrai, alors la condition est vraie.)

Quelles sont les commandes qui ne spécifient pas la plus petite quantité de PA60 ?

```
SELECT DISTINCT ncom
FROM   detail
WHERE  qcom > ANY (SELECT qcom
                   FROM   detail
                   WHERE  nrpo = 'PA60')
       AND     nrpo = 'PA60';
```

```
SELECT DISTINCT ncom
FROM   detail
WHERE  qcom > (SELECT MIN(qcom)
               FROM   detail
               WHERE  nrpo = 'PA60')
       AND     nrpo = 'PA60';
```

C'est un OU sur chaque élément de la liste. (qcom > 20 OU 30 OU … si 1 vrai, alors la condition est vraie)

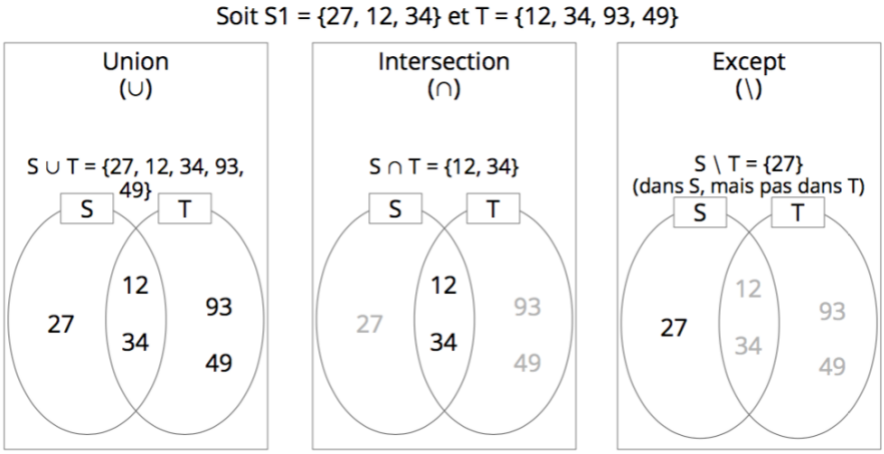
Table courante

Quelle est la nature du retour d'une requête ? C'est une table temporaire, on peut l'utiliser n'importe où (where, from).

```
SELECT * from commande
JOIN (SELECT *
      FROM CLIENT
      WHERE localite = 'Namur') AS c
     ON commande.ncli = c.ncli;
```

OPERATIONS ENSEMBLISTES ET STRUCTURES CYCLIQUES

Les opérateurs ensemblistes



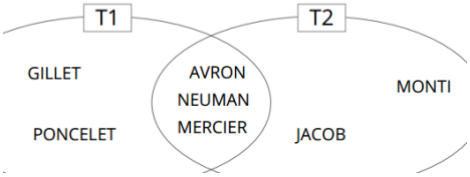
On peut changer l'ordre dans union et intersection mais pas dans Except.
Lorsque l'on fait un UNION ou une INTERSECT, **même s'il y a des doublons, il ne l'affiche qu'une fois SAUF si on met un ALL, alors ça affiche les doublons.**
Lorsque l'on fait un EXCEPT, même si dans T1 on a deux 4 et que dans T2 on a un 4, dans le EXCEPT il n'y aura **AUCUN 4 sauf si on met un ALL, alors il en affichera 1 dans l'exemple du dessus. Cad garde les doublons.**
Pour utiliser op. ensembliste : avoir le même nbre d'attributs.

Exemple Sans ALL:

T1
Nom
GILLET
AVRON
MERCIER
PONCELET
NEUMAN

T2
Nom
MONTI
NEUMAN
JACOB
MERCIER
AVRON

Soit sur les ensembles equivalents suivants :



T1 ∪ T2

```
select nom
from   T1
UNION
select nom
from   T2
```

nom
GILLET
AVRON
MERCIER
PONCELET
NEUMAN
MONTI
JAVOB

T1 \ T2

```
select nom
from   T1
EXCEPT
select nom
from   T2
```

nom
GILLET
PONCELET

T1 ∩ T2

```
select nom
from   T1
INTERSECT
select nom
from   T2
```

nom
AVRON
NEUMAN
MERCIER

DON

Exemple avec ALL

T1
num
4
4
5
7
1

T2
num
4
5
2

```
SELECT num
FROM   t1
INTERSECT
ALL
SELECT num
FROM   t2
```

num
4
5

```
SELECT num
FROM   t1
EXCEPT ALL
SELECT num
FROM   t2
```

num
7
4
1

```
SELECT num
FROM   t1
UNION ALL
SELECT num
FROM   t2
```

num
4
4
4
5
5
7
1
2

Remarque :

Pour pouvoir utiliser une requête ensembliste entre deux requêtes, il est nécessaire et suffisant que celles-ci renvoient exactement le même nombre de champs (colonnes). Le type de ceux-ci n'a pas d'importance.

Les opérateurs ensembliste peuvent être combinés.
Pour la différence symétrique par exemple, c'est pour avoir T1 et T2 – leur intersections.

```
(SELECT nom FROM T1
 EXCEPT
 SELECT nom FROM T2)
UNION
(SELECT nom FROM T2
 EXCEPT
 SELECT nom FROM T1)
```

Ou même, pour aussi avoir les éléments qui sont null.

```
SELECT ncom, client.ncli, datecom, nom, localite
FROM   commande, client
WHERE  commande.ncli = client.ncli

UNION

SELECT NULL AS ncom, ncli, NULL AS datecom, nom, localite
FROM   client
WHERE  NOT EXISTS
       (SELECT * FROM commande WHERE ncli = client.ncli);
```

Les structures cycliques

Table sur laquelle on peut faire une jointure sur elle-même pour avoir d'autre informations.

Exemple : Pas de pluriels pour les noms des tables

PRODUIT			
NPRO	LIBELLE	(PRIX_U)	(POIDS_U)
p1	A-200	--	--
p2	A-056	--	--
p3	B-661	--	--
p4	B-122	--	--
p5	B-326	--	--
p6	D-822	3.5	0.70
p7	D-507	8.0	0.25
p8	G-993	5.0	1.15
p9	F-016	--	--
p10	J-500	--	--
p11	J-544	0.5	0.90
p12	L-009	1.7	2.30

COMPOSITION		
COMPOSE	COMPOSANT	QTE
p1	p2	2
p1	p3	1
p1	p4	2
p2	p7	8
p2	p8	2
p3	p8	5
p4	p8	4
p4	p9	5
p4	p10	5
p5	p4	2
p5	p6	7
p9	p11	2
p10	p11	4
p10	p12	3

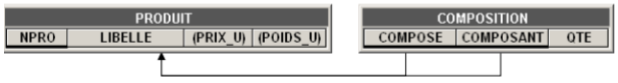
Quelle est la composition du produit 'p4' ?

```
SELECT h.npro, h.libelle, c.qte, b.npro, b.libelle
FROM   produit h
JOIN   composition c  ON c.compose = h.npro
JOIN   produit b      ON c.composant = b.npro
WHERE  h.npro = 'p4';
```

h.npro	h.libelle	c.qte	b.npro	b.libelle
p4	B-122	4	p8	G-993
p4	B-122	5	p9	F-016
p4	B-122	5	p10	J-500

Produit composé (haut)

Produit composant (bas)



Pour pouvoir représenter le composant et représenter le composé.