Synthèse De BD1

Partie de SQL

Fait par Van Bellinghen Brandon

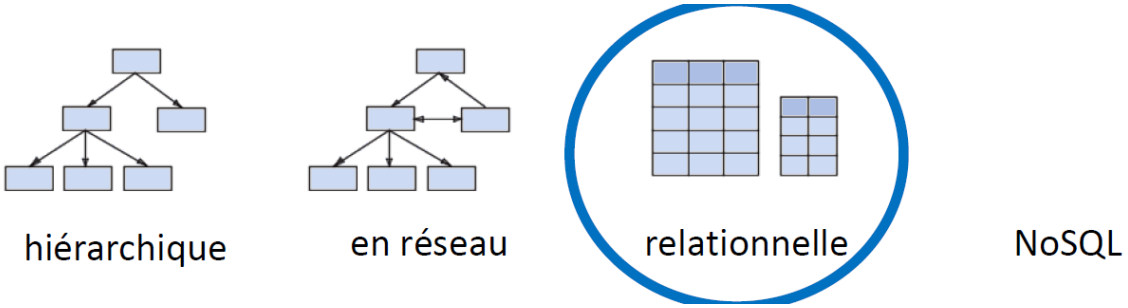
Cours 2021/2022

Partie SQL ;

Cours 1 ;

Base de données : - Gérer des grandes quantités d'informations comme clients, fournisseurs et etc.

Il existe différents types de bases de données :



SGBD : Système de Gestion de Bases de Données (en anglais : DBMS) , la gestion se fait via **SQL :** Structured Query Language.

- Permet l'accès aux données de façons souple.
- Autorise un accès aux informations à plusieurs utilisateurs.
- Permet de manipuler les données présentes dans la base de données : insertion, suppression et etc.

Bases de données relationnelles est composé de colonnes : **attributs et champs,** et aussi de lignes : **tuples, enregistrements ou records.**

Une création d'une table se fait avec la commande **CREATE :**

```
CREATE TABLE clients
(num          integer          NOT NULL,
nom           varchar(25)      NOT NULL,
prenom        varchar(15)      NOT NULL,
adresse       varchar(30)      NOT NULL,
cp            integer          NOT NULL,
ville         varchar(20)      NOT NULL,
tel           varchar(11)      NULL,
numtva        varchar(10)      NULL)
```

Comme en java, il existe plusieurs types de données :

bigint, bit, bit varying, boolean, char, character varying, character, varchar, date, double precision, integer, interval, numeric, decimal, real, smallint, time (avec et sans fuseau horaire), timestamp (avec et sans fuseau horaire), xml.

NULL : la valeur peut être omise.

NOT NULL : la valeur doit être spécifié.

⚠ **NULL n'est pas 0 ni une chaine vide !!!!!**

Conventions ;

- Mots réservés de l'instructions SQL : en majuscules.
- Tous les autres mots en minuscules
- Pas de caractères spéciaux, ni accentués, ni d'espaces et etc.
  ceci_est_un_nom_coorect.

La clé primaire ou Primary Key (PK) est un identifiant unique pour un enregistrements.

La commande pour l'insertion est **INSERT :**

```
INSERT INTO nom_table
[(nom_colonne {, nom_colonne})]
VALUES (valeur {, valeur})
```

Conventions ;

- Mot en majuscules = classe ou l'élément de l'instruction (mots réservé)
- [ ] -> élément facultatif.
- {…} -> élément facultatif qui peut être répété.
- | -> désigne une alternative (ou exclusif).

num	nom	prenom	adresse	cp	ville	tel	numtva

```
INSERT INTO clients (num, nom, prenom, adresse, cp, ville)
VALUES (18, 'Van Moer', 'Willy', 'rue Sainte Anne, 6', 1300, 'Wavre')

INSERT INTO clients (nom, prenom, numtva, num, tel, adresse, cp, ville)
VALUES ('Dupuis', 'Luc', NULL, 3, '010/34.65.67', 'rue Lambert Fortune,
2', 1300, 'Wavre')

INSERT INTO clients
VALUES (1, 'Dupuis', 'Benoit', 'clos Chapelle aux Champs, 43', 1200,
'Bruxelles', '02/764.46.46', NULL)
```

num	nom	prenom	adresse	cp	ville	tel	numtva
18	Van Moer	Willy	Rue Sainte Anne, 6	1300	Wavre		
3	Dupuis	Luc	Rue Lambert Fortune, 2	1300	Wavre	010/34.65.67	
1	Dupuis	Benoit	Clos Chapelle aux Champs, 43	1200	Bruxelles	0499123456	

La commande pour la consultation de données est **SELECT :**

```
SELECT * | nom_colonne {, nom_colonne…}
FROM nom_table {, nom_table …}
WHERE nom_condition
```

Conventions ;

- Mot en majuscules = classe ou l'élément de l'instruction (mots réservé)
- [ ] -> élément facultatif.
- {…} -> élément facultatif qui peut être répété.
- | -> désigne une alternative (ou exclusif).

Le mot clé **DISTINCT** permet d'éviter qu'un résultat se répète, **DISTINCT** se porte sur toutes les colonnes du **SELECT.**

Les conditions simples :

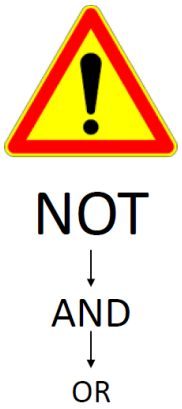
```
WHERE nom_colonne opérateur value
```

- Opérateurs : = | < | > | <> | != | IS | IS NOT

Conditions générales :

- Opérateurs : **AND | OR | NOT**
- **OR** est un ou non exclusif

Priorité des opérateurs ;



IS (NOT) NULL :

NULL ne peut jamais vérifier une égalité !!!

- On utilisera les mots clés **IS et IS NOT.**

Cours 2 ;

Condition LIKE :

```
SELECT [DISTINCT] * | nom_colonne {, nom_colonne…}
FROM clients
WHERE chaîne [NOT] LIKE motif
```

Motif ;

- Joker'%' -> remplace toute chaine de caractères.
- Joker'_' -> remplace un seul caractère.

Fonctions :

- **lower()** -> place le paramètre en minuscule.
- **upper()** -> place le paramètre en majuscule.

titre
La marche de l'Empereur
EMPEREURS, où êtes-vous ?
La vie secrète de l'empereur
Empereurs et impératrices

```
SELECT DISTINCT titre
FROM bd1.albums
WHERE lower(titre) LIKE
'%empereur%'
```

La marche de l'Empereur
EMPEREURS, où êtes-vous ?
La vie secrète de l'empereur
Empereurs et impératrices

Clause de tri: ORDER BY;

- **ORDER BY** définit l'ordre de présentation des résultats de l'instruction **SELECT.**

```
SELECT [DISTINCT]  * | [nom_table.] nom_colonne
        {,[nom_table.] nom_colonne…}
FROM      nom_table {, nom_table…}
WHERE     nom_condition
ORDER BY  nom_colonne {,nom_colonne…}
```

- L'attribut (ou la colonne) utilisé pour trier est la clé de tri.
- ORDER BY peut être utiliser avec **ASC (trier en ordre croissant) ou DESC (trier en ordre décroissant).**

```
SELECT * FROM clients ORDER BY nom ASC
```

num	nom	prenom	adresse	cp	ville	tel	numtva
1	Dupuis	Benoit	Clos Chapelle Champs, 43	1200	Bruxelles	02/764.46.46	BE413770425
3	Dupuis	Luc	Rue Lambert Fortune, 2	1300	Wavre	010/34.65.67	
18	Van Moer	Willy	Rue Sainte Anne, 6	1300	Wavre		

```
SELECT * FROM clients ORDER BY prenom DESC
```

num	nom	prenom	adresse	cp	ville	tel	numtva
18	Van Moer	Willy	Rue Sainte Anne, 6	1300	Wavre		
3	Dupuis	Luc	Rue Lambert Fortune, 2	1300	Wavre	010/34.65.67	
1	Dupuis	Benoit	Clos Chapelle Champs, 43	1200	Bruxelles	02/764.46.46	BE413770425

Fonctions agrégées ;

= fonctions portant sur la valeur globale résultat d'un calcul ou d'une comparaison.

Fonctions agrégées	
MIN( ) – MAX ( )	Minimum – maximum
COUNT ( )	Comptage
SUM ( )	Somme
AVG ( )	Moyenne (average in English)

- Renvoie toujours **une seule valeur.**

Importance du paramètre ;

```
SELECT count(*)
FROM bd1.albums
```

Comptage de :
- tous les tuples

```
SELECT count(serie)
FROM bd1.albums
```

- tous les tuples qui ont une série

=

```
SELECT
COUNT(*)
WHERE serie
IS NOT NULL
```

```
SELECT count(DISTINCT serie)
FROM bd1.albums
```

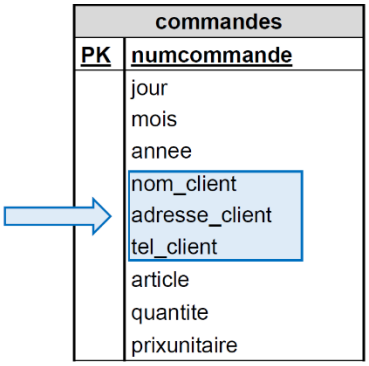
- toutes les séries différentes, **sauf valeur NULL**

Cours 3 ;

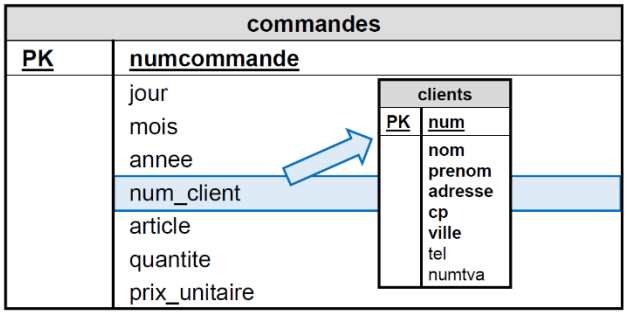
Relation 1 à N :

Contenu de la table commandes :

Ces 3 informations devront être répétées à chaque commande

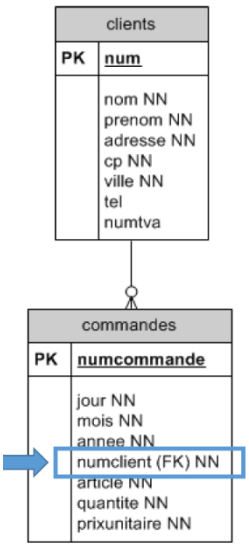


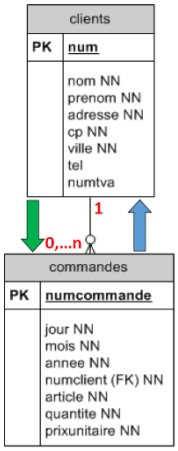
On pourrait extraire cette information et la stocker dans une autre table :



Clé étrangère ou Foreign Key (FK) :

- Identifie une colonne ou un ensemble de colonnes d'une table comme référençant une colonne ou un ensemble de colonnes d'une autre table.
- La FK garantit l'intégrité référentielle entre 2 tables.
- = un numclient dans la table commandes fera **TOUJOURS** référence à un num existant dans la table client.





UN client peut faire 0,1 ou plusieurs commandes.

UNE commande appartient à un seul client.

CREATE TABLE commandes :

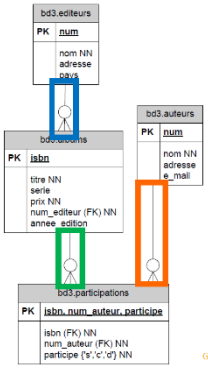
```
CREATE TABLE commandes
(numcommande     integer PRIMARY KEY,
jour             integer NOT NULL,
mois             integer NOT NULL,
annee            integer NOT NULL,
numclient        integer NOT NULL
                 REFERENCES clients(num),
article          varchar(30) NOT NULL,
quantite         integer NOT NULL,
prixunitaire     integer NOT NULL)
```

On ne pourra jamais créer de commande pour un client qui n'existe pas

Jointures :

```
SELECT commandes.article, commandes.quantite,
       clients.nom, clients.prenom
FROM commandes, clients
WHERE commandes.numclient = clients.num
```

article	quantite	nom	prenom
stylo M3	20	Dupuis	Benoit
styloM3	80	Van Moer	Willy
cartouche	160	Van Moer	Willy
classeurL	10	Dupuis	Luc
stylo M3	30	Dupuis	Benoit




```
SELECT *
FROM bd3.editeurs ed, bd3.albums
al, bd3.auteurs au, bd3.partipations
pa
WHERE ed.num = al.num_editeur
AND al.isbn = pa.isbn
AND pa.num_auteur = au.num
```


Alias d'une table :

- Un alias sur une table permet d'éviter de devoir réécrire le nom complet de la table à chaque fois.
- Par convention, on utilisera toujours des alias pour les noms de tables.

```
SELECT commandes.article, commandes.quantite,
       clients.nom, clients.prenom
FROM commandes, clients
WHERE commandes.numclient = clients.num
```

```
SELECT co.article, co.quantite, cl.nom, cl.prenom
FROM commandes co, clients cl
WHERE co.numclient = cl.num
```

Nom de table dans le SELECT ;

- Lorsque 2 attributs de 2 tables différent ont le même nom, il faut préciser le nom de la table.
- Imaginons que les tables commandes et clients aient toutes les 2 un attributs ‘’num’’pour designer le numéro clients.

```
SELECT article, quantite, nom, prenom
FROM commandes, clients
WHERE num = num
```

```
SELECT co.article, co.quantite, cl.nom, cl.prenom
FROM commandes co, clients cl
WHERE co.num = cl.num
```

Alias sur une colonne : AS ;

- On utilise AS pour définir le nom de la colonne de fonction agrégée.

```
SELECT count(co.numcommande),
       sum(co.prixunitaire*co.quantite)*0.95
FROM commandes co, clients cl
WHERE co.numclient = cl.num
AND cl.ville = 'Wavre'
```

count	?Column?
3	11780

```
SELECT count(co.numcommande) AS "nombre de commandes",
       sum(co.prixunitaire*co.quantite)*0.95 AS prix_total
FROM commandes co, clients cl
WHERE co.numclient = cl.num
AND cl.ville = 'Wavre'
```

nombre de commandes	prix_total
3	11780

Produit cartésien :

- Existe-t-il un client qui porte le même nom qu'un article ? Donnez son numéro et son nom

```
SELECT DISTINCT cl.num, cl.nom
FROM commandes co, clients cl
WHERE co.article = cl.nom
```

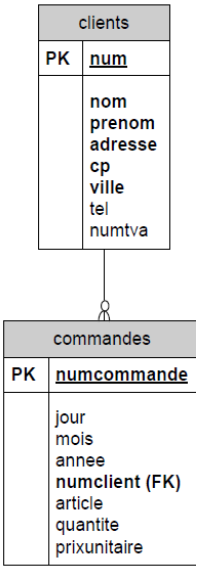
- On l'utilise assez rarement …

Cours 4 ;

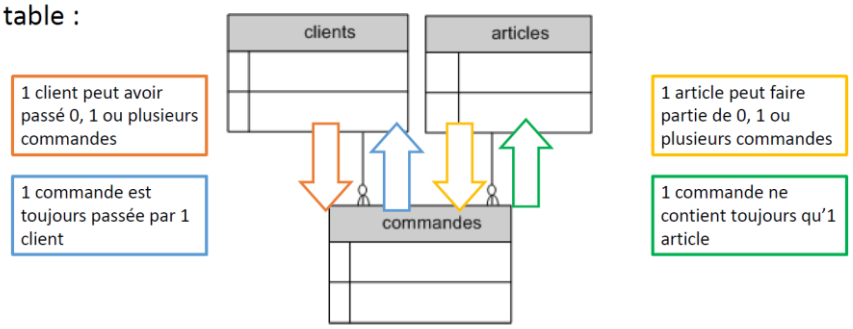
Relation M à N :

numcommande	jour	mois	année	numclient	article	quantite	prixunitaire
1	5	9	2000	1	stylo M3	20	150
2	6	9	2000	18	styloM3	80	150
3	6	9	2000	18	cartouche	160	20
5	8	9	2000	3	classeurL	10	80
8	1	10	2000	1	stylo M3	30	150

Redondance → prévoir un CATALOGUE D'ARTICLES



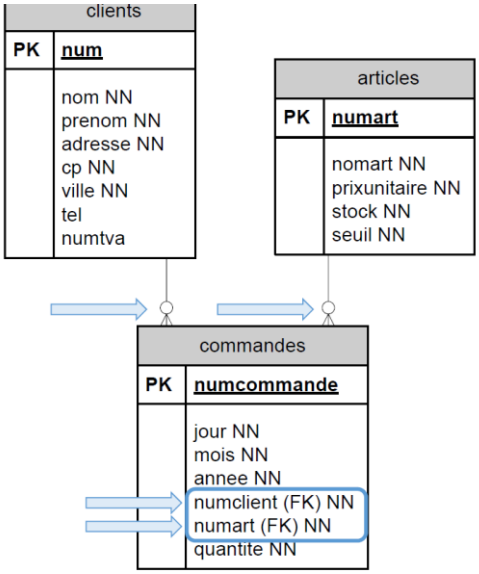
On pourrait extraire cette information et la stocker dans une autre table :



2 relations de 1 à N -> 2

FK

CREATE TABLE :

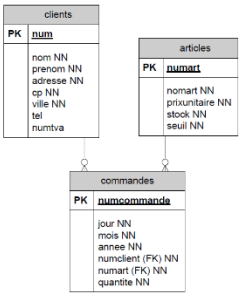


Haute École Léonard de Vinci
SCIENCES ET TECHNIQUES

CREATE TABLE (p. 20)

```
CREATE TABLE articles
(numart       int           PRIMARY KEY,
nomart        varchar(30)   NOT NULL,
prixunitaire  int           NOT NULL,
stock         int           NOT NULL,
seuil         int           NOT NULL);

CREATE TABLE commandes
(numcommande  int           PRIMARY KEY,
jour          int           NOT NULL,
mois          int           NOT NULL,
annee         int           NOT NULL,
numclient     int           NOT NULL      REFERENCES clients(num),
numart        int           NOT NULL      REFERENCES articles(numart),
quantite      int           NOT NULL);
```



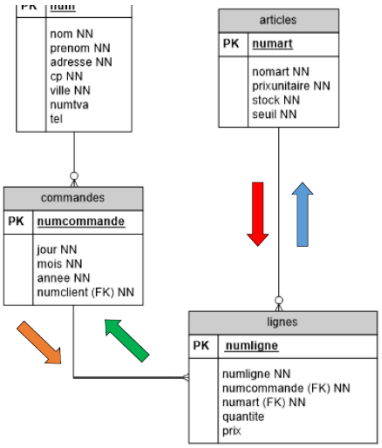
Problème est que toujours 1 seule article par commandes donc faudrait encore modifier quelques choses dans cette base de données.

Solution :

Solution

Table lignes de commandes :

- Une commande a 1 ou plusieurs lignes
- Une ligne fait partie d'une seule commande
- Un article peut être commandé 0, 1 ou plusieurs fois
- Une ligne de commande porte sur un seul article



Le choix de la PK :

1ere solution :

Une seule numérotation des lignes.

num ligne	num commande	numart	quantite	prix
1	1	3	4	5
2	1	4	1	4
3	2	3	2	5
...				

2eme solution :

Une numérotation des lignes par commandes.

num ligne	num commande	numart	quantite	prix
1	1	3	4	5
2	1	4	1	4
1	2	3	2	5
...				

3eme solution :

Un article ne peut être commandé qu'seule fois par commandes donc CONTRAIGNANT !!!

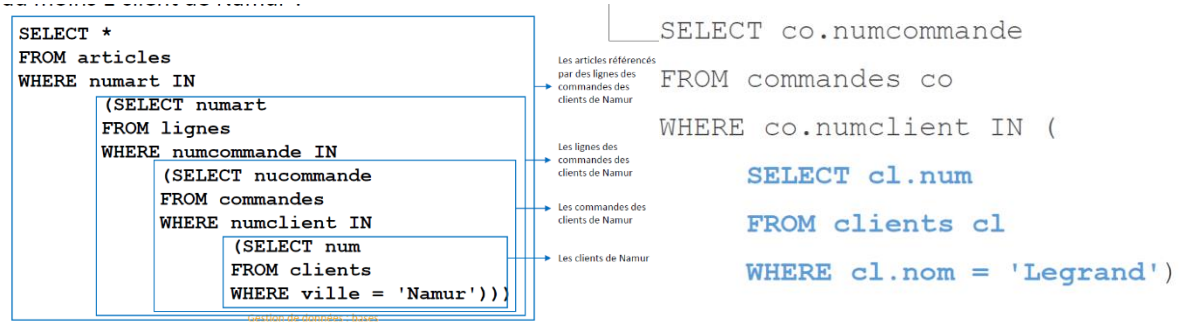
num commande	numart	quantite	prix
1	3	4	5
1	4	1	4
1	5	2	5
1	3	1	5

CREATE de la table ligne :

```
CREATE TABLE lignes
(numligne     int    NOT NULL,
numcommande   int    NOT NULL     REFERENCES commandes(numcommande)
numart        int    NOT NULL     REFERENCES articles(numart),
quantite      int,
prix          int,
PRIMARY KEY (numcommande, numligne))
```

Sub-SELECT ou sous requêtes :

- Une sous-requête consiste à exécuter une requête à l'intérieur d'une autre requête.
- Une sous-requête est aussi appelée une requête imbriquée, une requête cascade ou un sub-SELECT.
- Une sous-requête peut elle-même contenir une autre sous-requête.



Condition d'association négative :

- Permet de retenir les lignes qui ne sont pas associées aux éléments d'un ensemble déterminée de lignes.

```
SELECT *
FROM commandes
WHERE numcommande NOT IN
            (SELECT numcommande
             FROM lignes
             WHERE numarticle = 4)
```

GROUP BY :

- Regroupement des tuples selon un critère.
- Compte le nombre de lignes de commandes par commandes.

```
SELECT numcommande, count(*) AS nombre_lignes
FROM lignes
GROUP BY numcommande
```

numcommande	nombre_lignes
105	10
106	7
108	2

Fonction agrégée et GROUP BY :

- Dans le SELECT, on ne peut faire appel à une fonction agrégée que si
- Toutes les autres colonnes résultent elles aussi d'une fonction agrégée (-> résultat = 1 seule lignes).

OU

- Toutes les autres colonnes sont présentes dans la clause GROUP BY.

HAVING :

- **SELECTION DE CERTAINES TUPLES DU RESULTAT.**
- Ajouter un critère de sélection.

```
SELECT numcommande, count(*) AS nombre_lignes
FROM lignes
GROUP BY numcommande
HAVING count(*) > 5
```

numcommande	nombre_lignes
105	10
106	7

Ordre des instructions :

```
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
```

Facultatifs, dépend de ce qui est demandé dans la requête