I. Contrainte d'intégrité *(Rappel j'ai envie de dire)*

==Garantir la cohérence de la BD

- Doivent pouvoir être vérifiées automatiquement à chaque Insert°/Modificat°/Suppress° des données.

➢ Contrainte d'unicité[1] *(UNIQUE)*

➔ Mis en œuvre par l'identifiant qui doit donc posséder des valeurs distinctes.

- ✓ À tout instant, les lignes d'une table ont des valeurs distinctes de la PK (PK == NN).
- ✓ Tout tuple doit pouvoir être identifié de manière unique.

⇨ Même principe pour les clés primaires composites

Implémentation par le SGBD :

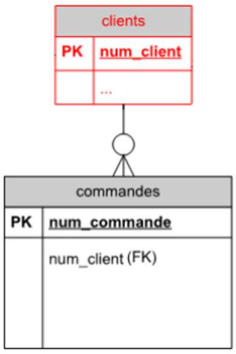
- Insertion/Modification d'un tuple → Uniquement autorisée si l'identifiant n'existe pas encore
- Suppression d'un tuple→ pas de contrainte

➢ Contrainte d'intégrité référentielles (FK)

➔Mis en œuvre par le mécanisme de clé étrangère :

- ✓ La valeur de la FK == une valeur de PK à laquelle elle se réfère
- ✓ Composée du même nbre de colonne, de même types et placées dans le même ordre.
- ✓ Le nom de la FK peut être ≠ de la PK

Implémentation par le SGBD :

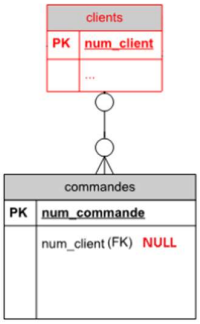


- Insertion d'un tuple dans *commande* → Uniquement autorisée si le client existe dans *clients*
- Supprimer dans *commande* → Pas de contrainte
- Modificat° de *num_client* dans *commande* → Uniquement autorisée si le nouveau num_client existe dans *clients*

- Insertion dans *clients* → Pas de contrainte
- Supprimer dans *clients* → Si ~~commandes~~ : pas de contrainte
- Supprimer dans *clients* → Si commandes : Soit refus, Soit suppression commandes associées
- Modificat° de *num_client* dans *client* → Si ~~commandes~~ : pas de contrainte
- Modificat° de *num_client* dans *client* → Si commandes : Soit refus, Soit modificat° des num_clients dans les tuples de commandes associées

Cas particulier : FK facultative

- Suppression/Modificat° dans *clients* → On mettra la FK de command NULL pour les tuples qui référençaient ce client
-



[1] Contrainte d'intégrité de base

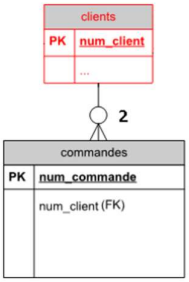
➢ **Contrainte de colonne obligatoire (NOT NULL)**

- ✓ Pas besoin d'explication, tmtc

Implémentation par le SGBD :

- Insertion/Modification d'un tuple ne contenant pas de valeur dans une colonne obligatoire est immédiatement bloquée.

- **Autre contrainte d'intégrité** – ne peuvent pas tjs être implémentées par la DB
  - ✓ Valeurs permises
  - ✓ Contraintes de cardinalité : pas possible d'implémenter le 2
  - ✓ Règles – métier : Toutes contraintes que les attributs doivent respecter (ensemble de valeurs permises, format….)



II. Identifiant (PK)

== attribut/ensemble d'attribut qui permet de distinguer de manière unique une entité. Doit être constitué de colonne obligatoire

Ex : numéro, année == identifiant / numéro, année, difficultés == identifiant / numéro, année, difficultés, énoncé == identifiant, ...

Numéro	Année	Difficultés	Enoncé	Sujet	Nbre tables
1	2010	DSD	Organisation des ventes de la société Y	Vente	12
2	2010	DSD	Euroclean	Firme nettoyage - gestion circuits aller-retour panier	6
3	2010	DSD	Hopital St Lambert	Complexe !	16
1	2012	DSD	ToutEnbois	Tarification et ensuite commandes	9+7
2	2012	DSD	RépertoireBibliographique	Cumul de petites difficultés en 1 DSD.	18
1	2013	Appro	Offres & devis	Relation 1-1	n/a

Si y a plusieurs identifiant possible (ex : pour une voiture, n° de plaque/n° de châssis unique), le plus représentatif ➔ PK.

Un attribut peut appartenir à plusieurs identifiants.
Ex : N° co, N°Ligne == identifiant / : N° co, Produit == identifiant

- ➢ Un identifiant **minimal** == identifiant dont on ne peut retirer aucun attribut. On ne choisit que des identifiants minimaux pour chaque table d'un schéma conceptuel de BD.

Ex : ici numéro, année == identifiants minimaux mais pas les autres.

Numéro	Année	Difficultés	Enoncé	Sujet	Nbre tables
1	2010	DSD	Organisation des ventes de la société Y	Vente	12
2	2010	DSD	Euroclean	Firme nettoyage - gestion circuits aller-retour panier	6
3	2010	DSD	Hopital St Lambert	Complexe !	16
1	2012	DSD	ToutEnbois	Tarification et ensuite commandes	9+7

III. Superclé – Recherche des identifiants

== groupe d'attributs qui, quand on fixe une valeur pour ce groupe, apparaît dans max. 1 tuple.
La + grande superclé = l'ensemble des attributs d'une entité.

But ? Quand on ne sait pas quel identifiant prendre, on va essayer de combiner des colonne pour que ça marche.

On ne va pas utiliser ce système, c'est juste pour nous apprendre ce que c'est une superclé.

nom commun	genre	espèce	botaniste
Alisier blanc	Sorbus	aria	L.
Châtaignier	Castanae	sativa	Miller
Cotonéaster commun	Cotoneaster	vulgaris	L.
Epine-vinette	Berberis	vulgaris	L.
Frêne commun	Fraxinus	excelsior	L.
Erable champêtre	Acer	campestre	L.
Bouleau verruqueux	Betula	verrucosa	Ehrh.
Bouleau verruqueux	Betula	pendula	Roth
Sorbier des oiseleurs	Picea	abies	(L.) Karsten
Sorbier	Sorbus	domestica	L.

- À toute première vue, on va combiner TOUTE les colonnes pour la superclé (==identifiant)
  Mais c'est logique, que c'est pas impossible de clé des PK qui reprenne l'ensemble des colonnes d'une table.
  - **nom_commun, genre, espèce, botaniste**

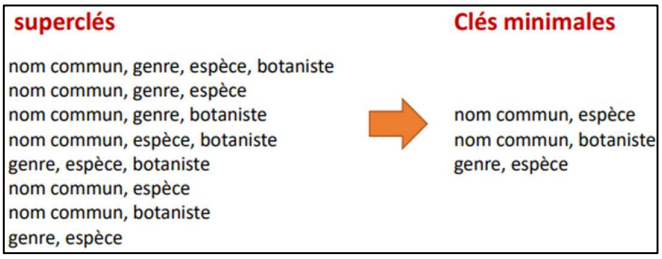
- Après on va essayer des combinaisons de 3 colonnes pour que ça ne se répète pas. On va à chaque fois regarder les combi et voir si c'est UNIQUE.
  - **nom_commun, genre, espèce**
  - **non_commun, genre, botaniste**
  - **non_commun, espèce, botaniste**
  - **genre, espèce, botaniste**

- Et puis de 2 colonnes, et là on voit certaines combi qui ne fonctionne pas. Regarde
  - non_commun, genre → NON car il sera repris 2x
  - **non_commun, espèce → OK**
  - **non_commun, botaniste → OK**
  - **genre, espèce → OK**
  - genre, botaniste → NON
  - espèce, botaniste → NON

- Et 1 seule colonne, tu te doutes, impossible, aucune précision

- **Clé candidate = identifiant minimal**

== chacun de ses attributs/colonnes est nécessaire pour garantir la contrainte d'unicité.



- On choisit ensuite **l'identifiant minimal** le + représentatif qu'on déclare PK en fonction du contexte.
- Un attribut qui ne fait pas partie de la PK = attribut non premier, quel que soit la clé primaire choisie.

IV. Dépendance fonctionnelle

Pk ? Pour éviter les redondances d'information.

R(No-frn,	Nom,	Ville,	No-produit,	Prix)
1	Antoine	Bruxelles	1	3,00
2	Bernard	Anvers	5	6,20
1	Antoine	Bruxelles	5	5,95
…				

- **Problème des redondances internes**

Redondance d'information ==

Prenons un exemple : Quand on va voir le **n° du fournisseur** 1, le **Nom** et la **Ville** vont être répétée à chaque fois qu'on va voir ce fournisseur 1.

Problème :

- La modificat° des donnée du fournisseur doit être faite sur TOUT les tuples de la table.
- Les fautes de frappes/Numéro différents pour le même fournisseur.
- Le fournisseur n'est pas une entité isolée → Rompt le principe d'1 BD relationnelles (==Toute entité du domaine doit être enregistrée une seule fois).
- Peut pas définir un nouveau fournisseur s'il n'a pas fourni de produit.
- On perd les renseignements du fournisseur si on supprime les produits qu'il livre.

- **Dépendance fonctionnelle**

== Un ensemble d'attribut X détermine fonctionnellement un ensemble d'attributs Y SI Dans tout tuple les mêmes valeurs pour les attributs de X == les même valeurs pour les attributs de Y.

En BD relationnelles, il doit donc exister dans chaque table, une DF entre l'id de cette table et chacune des colonnes de celle-ci.

Ex : Si les 2 lignes ont le même **n°_fournisseur**, alr elles doivent avoir le mêmes valeurs de **Nom** et de **Ville** → N°_fournisseur détermine Nom, Ville

Notation : $X \longrightarrow Y$

X détermine Y (*X est un déterminant de Y*)

Y est déterminé par X (*Y est dépendant de X*)

- **Détection des redondances internes**

== Quand il existe un déterminant qui n'est pas un id de la relation.

- **Suppression des redondances internes par décomposition**

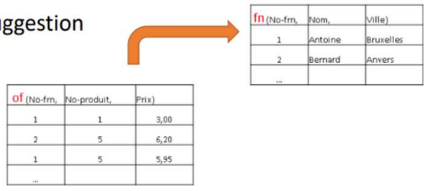
1. Redondance détectée

R(No-frn,	Nom,	Ville,	No-produit,	Prix)
1	Antoine	Bruxelles	1	3,00
2	Bernard	Anvers	5	6,20
1	Antoine	Bruxelles	5	5,95
…				

Cette table contient 2 entités :

- L'entité fournisseur (n°_founisseur qui détermine Nom et Ville)
- L'entité offre_prix (n°_fournisseur et le n°_produit détermine le prix).

2. Suggestion



- Décomposer R en 2 fragments R1 et R2.
  - ✓ R1 (déterminant_1, déterminé)
  - ✓ R2 (déterminant_2, résidu)
  - ✓ Le déterminant_1 dans R2 = FK vers R1

- Normaliser == décomposer 1 table de manière à éliminer ses dépendances anormales

- La décomposition ne sait SANS perte d'information.

V. Calcul des identifiants d'une relation

Quand ? Qd on trouve des tables intermédiaires pcq il y a des associations M à N.

Cmt ? À partir de l'ensemble des DF dont la relation est le siège.

On a 1 grande relation, on a des données. En trouvant les DF dans ces données, on va être capable de trouver l'id de ces données.

VI. Normalisation

Normaliser une relation = lui appliquer une/plusieurs décompositions préservant les données afin d'éliminer les problèmes de redondance interne dont elle est éventuellement le siège.

C'est un processus qu'on doit suivre, et on doit vrmt le suivre d'une certaine manière. On doit être très ordonnée pcq c'est comme un processus à tiroir, faut avoir terminer la 1ère activité pour faire le second, etc…

- **1 FN** *(Première Forme Normale)* **: Chaque attribut n'a que des valeurs atomiques.**

Eviter qu'il y ait des listes comme valeur dans une BD.

D(Nom,	Ville,	Chef,	Employés)
Compta	Bruxelles	Jean	~~René, Paul, Marie~~
Marketing	Namur	Paul	~~Pierre, Jules~~

D(Nom,	Ville,	Chef,	Employé1,	Employé2,	Employé3)
Compta	Bruxelles	Jean	René	Paul	Marie
Marketing	Namur	Paul	Pierre	Jules	
…					

On ne rajoute pas des colonnes !!

On rajoute des lignes.

On va créer autant de ligne qu'il n'y a d'éléments dans la liste.

D(Nom,	Ville,	Chef,	Employés)
Compta	Bruxelles	Jean	René, Paul, Marie
Marketing	Namur	Paul	Pierre, Jules
…			

1FN

D(Nom,	Ville,	Chef,	Employé)
Compta	Bruxelles	Jean	René
Compta	Bruxelles	Jean	Paul
Compta	Bruxelles	Jean	Marie
Marketing	Namur	Paul	Pierre
Marketing	Namur	Paul	Jules
…			

Remarque :

Les propriétés composées qui ont du sens (ex : adresse (rue, n°, code_postale,ville)), l'éclatement des valeurs concaténées peut être traités après toutes les étapes de normalisat°.

- **2 FN** *(Deuxième Forme Normale)* **:**

Condition :

- Faut qu'elle soit d'abord en 1FN
- Ne pas contenir de DF à une partie stricte de la clé == A est une partie stricte de B
  - ☞ Si A est incluse dans B
  - ☞ A est ≠ de B
  - ☞ Ce qui implique que A est nécessairement + petit que B.

Si j'ai 1 clé qui est composé d'1 seul champs → Pas de partie stricte pcq **clé < partie stricte.**

Si j'ai par ex. clé = no_commande, no_produit, alr la partie strictre : Le no_commande ET no_produit. Ça doit être une colonne incluse dans la clé MAIS ça ne peut pas être la clé composée entièrement car **clé < partie stricte**. clé < partie

Clé	Partie stricte
No_client	--
No_commande, no_produit	No_commande et no_produit

Autre exemple : Clé : A,B,C

Partie strictes :

- A, B - B, C - A, C
- A - B - C

R(produit, fournisseur, nom, ville, prix) pas en 2FN ⟶ Car il y a des DF (fournisseur -> nom, ville).

2FN

Comment décomper ?

1. Isoler la 1ère relation le fournisseur et tout ce qu'il détermine.
2. Garder dans la 2ème relation, la clé de base et les résidus.
3. Qd on a finit de décomposer, faut checker pour voir si chacune des relation est bien en 2FN sinon on continue à décomp

R (produit, description, fournisseur, nom, v

Décomposition :

R1(fournisseur, nom, ville) **en 2FN**

R2(produit, fournisseur, prix) **en 2FN**

- **3 FN** *(Troisième Forme Normale)* **:**

Condition :

- Être en 2 FN
- Ne doit pas contenir de DF **transitive** ==
  - ☞ Il ne peut pas y avoir d'attribut, un champ/colonne qui soit non premier[2] et qui dépendent d'un autre attribut non premier
    ➔En gros tous les champs doivent être dépendant de la clé.
  - ☞ Chaque déterminant est in identifiant

Prenons un exemple :

R (produit, description, fournisseur, nom, ville, prix) ⟵ DF transitives car :

nom et ville dépend de fournisseur

fournisseur dépend de produit

R (produit, description, fournisseur, nom, ville, prix) en 2FN

3FN

Comment décomper ?

1. Isoler la 1ère relation le fournisseur et tout ce qu'il détermine.
2. Garder dans la 2ème relation, la clé de base et tous les champs que produit définit directement.
3. Fournisseur devient la FK vers la R1.

- R1 (fournisseur, nom, ville)
- R2 (produit, description, fournisseur, prix)

[2] Non premier == qui ne fait pas partie de la clé

Donc au lieu de partir d'un énoncé dans lequel on nous donne des types d'entité où on doit trouver les propriétés, où on doit mettre des identifiants, où on doit faire des association,..

Nous on part d'un espèce de grand SELECT, de jointure de toutes les tables et on va essayer de reconstruire au départ des données, une grande BD.

- **Implications**

Si la BD est normalisée :

- Un id doit déterminer toutes les colonnes de la table.
- Tout déterminant doit être un id d'une table.