HEIG VD | MEI
Media Engineering Institute

04 - Normalisation

Infrastructure de données 1

Noemi Romano | 20 mars 2025

Semaine passée

1. Créer une base de données
2. Lire les données
3. Mettre à jour les données
4. Supprimer des données
5. Filtrer les données
6. Agréger les données
7. Trier les données
8. Mettre en relation les données

Normalisation

Normalisation

*“La normalisation consiste à **restructurer** une base de données pour respecter certaines formes normales, afin d'éviter la **redondance** des données (des données apparaissent plusieurs fois) et d'assurer l'**intégrité** des données.” Wikipedia*

- Meilleure **organisation** de la base de données
- Réduit la **redondance**
- Améliore la **cohérence** et l’**intégrité**

Prémière Forme Normale (1NF)

Éliminer les valeurs multiples et assurer des colonnes atomiques

Une table non normalisée contient plusieurs valeurs dans une seule colonne

commande_id	client_nom	produits	quantités
1	Alice	Pomme, Banane	3, 2
2	Bob	Pomme	5

Problème 

Prémière Forme Normale (1NF)

Éliminer les valeurs multiples et assurer des colonnes atomiques

Une table non normalisée contient plusieurs valeurs dans une seule colonne

commande_id	client_nom	produits	quantités
1	Alice	Pomme, Banane	3, 2
2	Bob	Pomme	5

Problème

- *Impossible de filtrer ou mettre à jour un seul produit*
- *Aucune atomicité dans la structure*

Prémière Forme Normale (1NF)

Éliminer les valeurs multiples et assurer des colonnes atomiques

Une table non normalisée contient plusieurs valeurs dans une seule colonne

commande_id	client_nom	produits	quantités
1	Alice	Pomme, Banane	3, 2
2	Bob	Pomme	5

Solution ??

Prémière Forme Normale (1NF)

commande_id	client_nom	produits	quantités
1	Alice	Pomme, Banane	3, 2
2	Bob	Pomme	5

commande_id	client_nom	produits	quantités
1	Alice	Pomme	3
2	Bob	Pomme	5
3	Alice	Banane	2

Deuxième Forme Normale (2NF)

Une colonne ne doit pas dépendre d'une partie de la clé primaire

commande_id	client_id	client_nom	produit	quantité
1	101	Alice	Pomme	3
2	101	Alice	Banane	2

Problème

- *Duplication des informations clients*
- *La clé primaire commande_id n'identifie pas client_nom*

Deuxième Forme Normale (2NF)

Une colonne ne doit pas dépendre d'une partie de la clé primaire

commande_id	client_id	client_nom	produit	quantité
1	101	Alice	Pomme	3
2	101	Alice	Banane	2

Solution

Deuxième Forme Normale (2NF)

commande_id	client_id	client_nom	produit	quantité
1	101	Alice	Pomme	3
2	101	Alice	Banane	2

commandes

commande_id	client_id	produit	quantité
1	101	Pomme	3
2	101	Banane	2

clients

client_id	client_nom
101	Alice

Troisième Forme Normale (3NF)

Une colonne ne doit pas dépendre d’une autre colonne qui n’est pas la clé primaire.

employe_id	employe_nom	departement	directeur
1	Alice	Comptabilité	Mr. Dupont
2	Bob	RH	Mme. Martin
3	Clara	Comptabilité	Mr. Dupont

Problème 

Troisième Forme Normale (3NF)

Une colonne ne doit pas dépendre d'une autre colonne qui n'est pas la clé primaire.

employe_id	employe_nom	departement	directeur
1	Alice	Comptabilité	Mr. Dupont
2	Bob	RH	Mme. Martin
3	Clara	Comptabilité	Mr. Dupont

Problème

- *Le directeur dépend du département, pas de l'employé.*
- *Si le directeur change, il faut modifier plusieurs lignes*

Troisième Forme Normale (3NF)

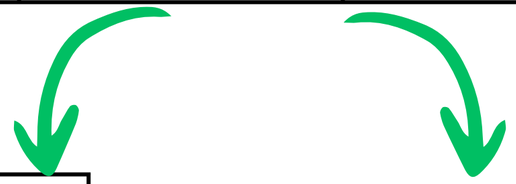
Une colonne ne doit pas dépendre d’une autre colonne qui n’est pas la clé primaire.

employe_id	employe_nom	departement	directeur
1	Alice	Comptabilité	Mr. Dupont
2	Bob	RH	Mme. Martin
3	Clara	Comptabilité	Mr. Dupont

Solution

Troisième Forme Normale (3NF)

employe_id	employe_nom	departement	directeur
1	Alice	Comptabilité	Mr. Dupont
2	Bob	RH	Mme. Martin
3	Clara	Comptabilité	Mr. Dupont



employees

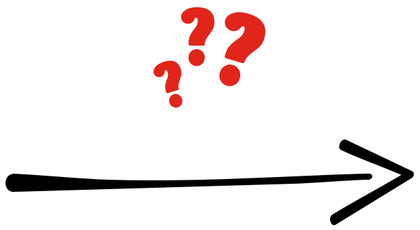
employe_id	employe_nom	departement_id
1	Alice	1
2	Bob	2
3	Clara	1

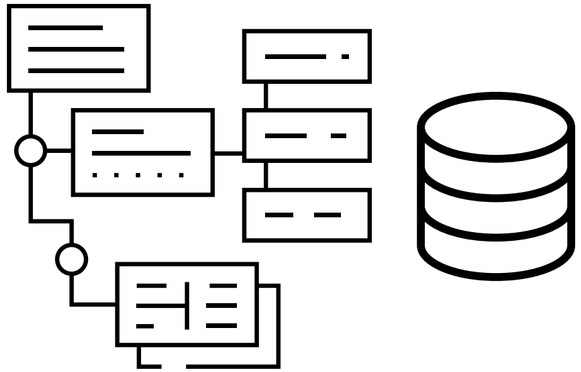
department

departement_id	nom	directeur
1	Comptabilité	Mr. Dupont
2	RH	Mme. Martin

Normaliser (SQL)

Données non normalisées





Données normalisées

Plan de normalisation

- Créer des **tables temporaires** des données **non normalisées**
- Créer **tables normalisées**
- Insérer données dans tables non normalisées
- Insérer les données dans table normalisées à partir des tables non normalisées

Tables temporaires

CREATE TEMP table_name

- Stocker des données de façon temporaire pendant la durée d'une session ou d'une transaction
- Idéales pour manipuler des données intermédiaires

```
CREATE TEMP TABLE temp_employes (
    id,
    nom VARCHAR(50),
    departement VARCHAR(50)
);
```

Insertion à partir de tables non normalisées

- ***table_name [(column1, column2, ...)]*** La table cible dans laquelle les données seront insérées et la liste des colonnes optionnelles.
- ***SELECT*** *expression1, expression2, ...* ***FROM*** *source_table [WHERE condition]* : La requête de sélection qui fournit les valeurs à insérer.
- ***ON CONFLICT*** *(conflict_target)* : La clause qui définit la ou les colonnes sur lesquelles PostgreSQL doit détecter un conflit (souvent une contrainte d'unicité).
- ***DO NOTHING*** : Option pour ignorer l'insertion si un conflit est détecté.
- ***DO UPDATE SET*** *...* : Option pour mettre à jour la ou les colonnes spécifiées en cas de conflit. La pseudo-table EXCLUDED permet de référencer les valeurs proposées pour l'insertion.

```
INSERT INTO table_name [ ( column1, column2, ... ) ]
SELECT expression1, expression2, ...
FROM source_table
[ WHERE condition ]
ON CONFLICT (conflict_target)
DO { NOTHING | UPDATE SET column1 = value1 [, column2 = value2, ...]
[ WHERE condition ] };
```

Projet

- Retour par groupe pour diagramme UML
- Création dépôt *git* pour les codes SQL (facultatif). Rendez-vous sur Github: https://github.com/
- Créer les tables normalisées et temporaires
- Créer le script d’import