

# JUSQU'ICI...

- **On a reçu une base de données**
  - avec le schéma déjà construit
  - avec les données déjà entrées
- **On a seulement fait des consultations**

# CRÉER SA PROPRE BASE DE DONNÉES

## Consiste à créer les tables

- Avec leurs relations
- Leurs contraintes d'intégrité
  - c-a-d les tables n'acceptent les données que si les contraintes d'intégrité sont respectées

# CREATE TABLE

```
CREATE TABLE nom_table ( [  
    { nom_colonne type_donnees  
        [ DEFAULT default_expr ] [ contrainte_colonne [...] ]  
    | contrainte_table }  
    [, ... ]  
]
```

# CREATE TABLE

```
CREATE TABLE nom_table ( [  
    { nom_colonne type_donnees  
        [ DEFAULT default_expr ] [ contrainte_colonne [...] ]  
    | contrainte_table }  
    [, ... ]  
]
```

# CREATE TABLE

## Types colonne :

- character [ (n) ]
- character varying [ (n) ]
- smallint
- integer
- numeric [ (p, s) ]
  - p=nombre total de chiffres significatifs, s=après la virgule
- double precision
- timestamp
- confer annexe syllabus

# EXAMPLE

```
CREATE TABLE utilisateurs (  
    u_id          INTEGER,  
    prenom        CHARACTER VARYING (50) ,  
    nom           CHARACTER VARYING (50) ,  
    email         CHARACTER VARYING (50) ,  
    naissance     DATE ,  
    taille        NUMERIC (4,1) ,  
    marie         BOOLEAN ,  
    theme_id      INTEGER  
)
```

# PK AUTO-GÉNÉRÉE: SERIAL

- **SERIAL**
  - Type permettant de générer des clés primaires automatiquement
  - Type entier mais sa valeur par défaut suivra l'ordre d'une séquence
    - Une SEQUENCE va être automatiquement créée

```
CREATE TABLE utilisateurs (  
    u_id          SERIAL,  
    ...
```

# CREATE TABLE

```
CREATE TABLE nom_table ( [  
    { nom_colonne type_donnees  
        [ DEFAULT default_expr ] [ contrainte_colonne [...] ]  
    | contrainte_table }  
    [, ... ]  
]
```



# EXAMPLE

```
CREATE TABLE utilisateurs (  
    u_id          SERIAL,  
    prenom        CHARACTER VARYING (50) ,  
    nom           CHARACTER VARYING (50) ,  
    email         CHARACTER VARYING (50) ,  
    naissance     DATE ,  
    taille        NUMERIC (4,1) ,  
    marie         BOOLEAN DEFAULT false ,  
    theme_id      INTEGER  
)
```

# CREATE TABLE

```
CREATE TABLE nom_table ( [  
    { nom_colonne type_donnees  
        [ DEFAULT default_expr ] [ contrainte_colonne [...] ]  
    | contrainte_table }  
    [, ... ]  
]
```

# CREATE TABLE

## Contraintes colonnes

- NOT NULL
- NULL
- UNIQUE
- PRIMARY KEY
- CHECK (expression)
- REFERENCES table (colonne)

# EXAMPLE

```
CREATE TABLE utilisateurs (  
    u_id          SERIAL PRIMARY KEY,  
    prenom        CHARACTER VARYING (50) NOT NULL,  
    nom           CHARACTER VARYING (50) NOT NULL,  
    email         CHARACTER VARYING (50) UNIQUE NOT NULL,  
    naissance     DATE NOT NULL CHECK(naissance<now()),  
    taille        NUMERIC (4,1) CHECK (taille>0),  
    marie         BOOLEAN DEFAULT false,  
    theme_id      INTEGER REFERENCES themes (theme_id)  
)
```

# CREATE TABLE

```
CREATE TABLE nom_table ( [  
    { nom_colonne type_donnees  
        [ DEFAULT default_expr ] [ contrainte_colonne [...] ]  
    | contrainte_table }  
    [, ... ]  
]
```

# CREATE TABLE

## Contraintes table

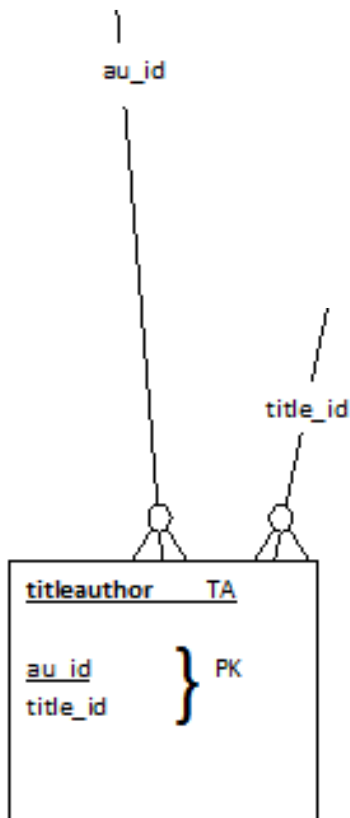
- UNIQUE (colonne1, colonne2, ...)
- PRIMARY KEY (colonne1, colonne2, ...)
- CHECK (expression)
- FOREIGN KEY (colonneA1, colonneA2, ...) REFERENCES table (colonneB1, colonneB2, ...)

# EXAMPLE

```
CREATE TABLE themes (  
    theme_id      SERIAL PRIMARY KEY,  
    couleurFond   VARCHAR(10) ,  
    couleurAvant  VARCHAR(10) ,  
    CHECK (couleurFond<>couleurAvant) ,  
    UNIQUE (couleurFond,couleurAvant)  
)
```

# AUTRE EXEMPLE

```
CREATE TABLE titleauthor (  
    au_id INTEGER NOT NULL  
        REFERENCES authors (au_id),  
    title_id INTEGER NOT NULL  
        REFERENCES titles (title_id),  
    CONSTRAINT ta_pkey PRIMARY KEY (au_id, title_id)  
)
```





# CONTRAINTES D'INTÉGRITÉ

**Vérification effectuée automatiquement.**

- Garantie ultime de l'intégrité des données.
- Travail pour le développeur :
  - Une seule fois réfléchir correctement !
  - Inutile de perdre son temps à valider ensuite.

=> Il est très intéressant d'utiliser les contraintes d'intégrité.

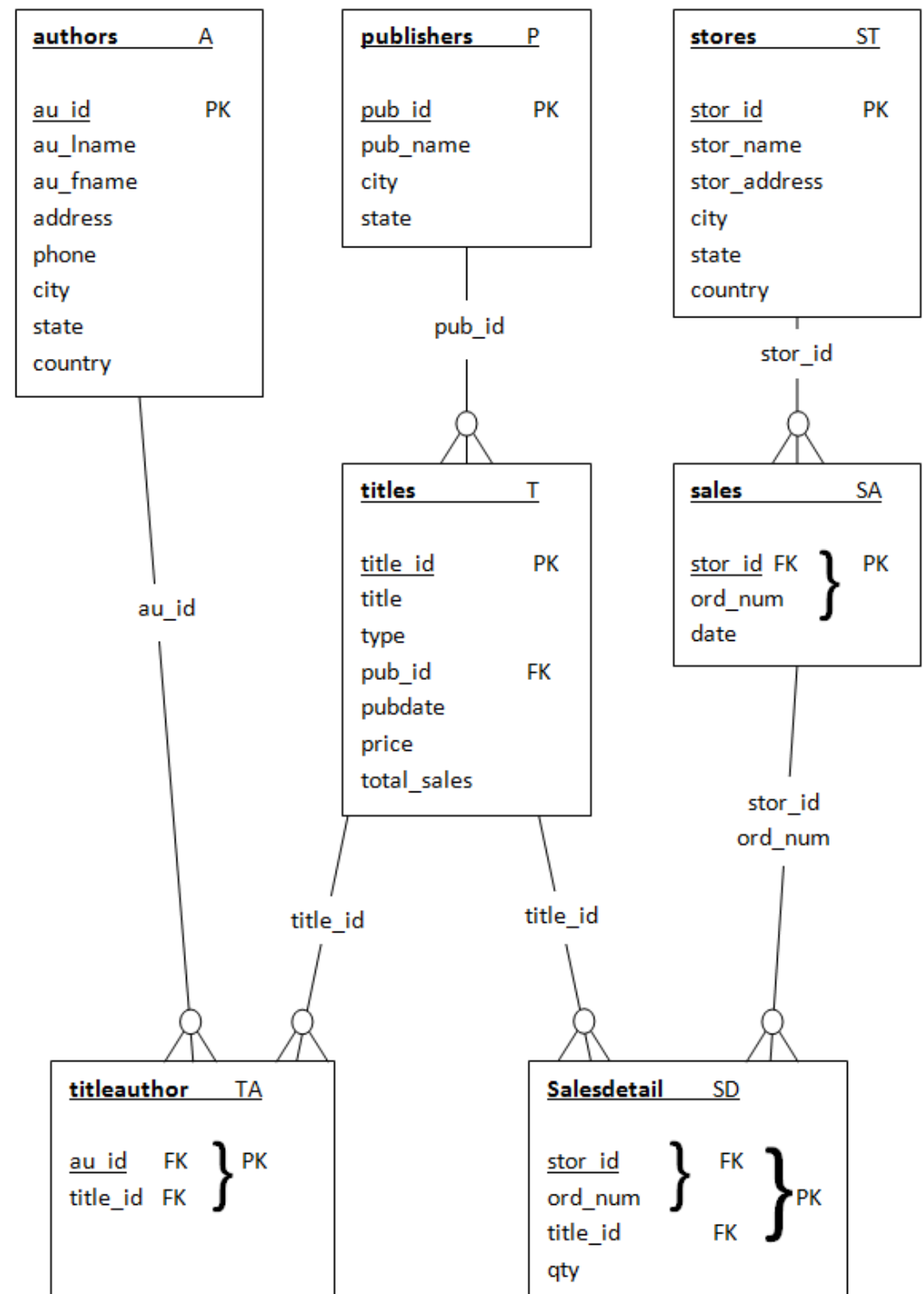
# SCHEMA

- Il est une bonne pratique de créer un schéma contenant toutes les tables
- Exemple:

```
CREATE SCHEMA projet;  
  
CREATE TABLE projet.themes (  
    theme_id      INTEGER PRIMARY KEY,  
    couleurFond   CHARACTER (10),  
    couleurAvant  CHARACTER (10),  
    CHECK (couleurFond<>couleurAvant),  
    UNIQUE (couleurFond,couleurAvant)  
)
```

# SUPPRESSION TABLE

- **DROP TABLE table**
  - Attention, à ne pas violer les contraintes d'intégrité quand on drop une table
  - En général il y a un ordre pour dropper les tables
- **Si on veut supprimer tout le schéma**
  - DROP SCHEMA nomSchema CASCADE;



# MODIFICATION D'UNE TABLE

- `ALTER TABLE nom [ * ] action [, ... ]`
  - action peut être :
    - `ADD [ COLUMN ] colonne type [ contrainte_colonne [ ... ] ]`
    - `DROP [ COLUMN ] colonne`
    - `ALTER [ COLUMN ] colonne [ SET DATA ] TYPE type [ USING expression ]`
    - `ALTER [ COLUMN ] colonne SET DEFAULT expression`
    - `ALTER [ COLUMN ] colonne DROP DEFAULT`
    - `ALTER [ COLUMN ] colonne { SET | DROP } NOT NULL`
    - `ADD contrainte_table`
    - `DROP CONSTRAINT nom_contrainte`
- Exemples
  - `ALTER TABLE publishers ADD COLUMN continent varchar(30);`
  - `ALTER TABLE publishers ALTER COLUMN continent TYPE varchar(20);`
  - `ALTER TABLE publishers DROP COLUMN continent;`

# TRANSFORMER UN PROBLÈME RÉEL EN UN SCHÉMA DE BD

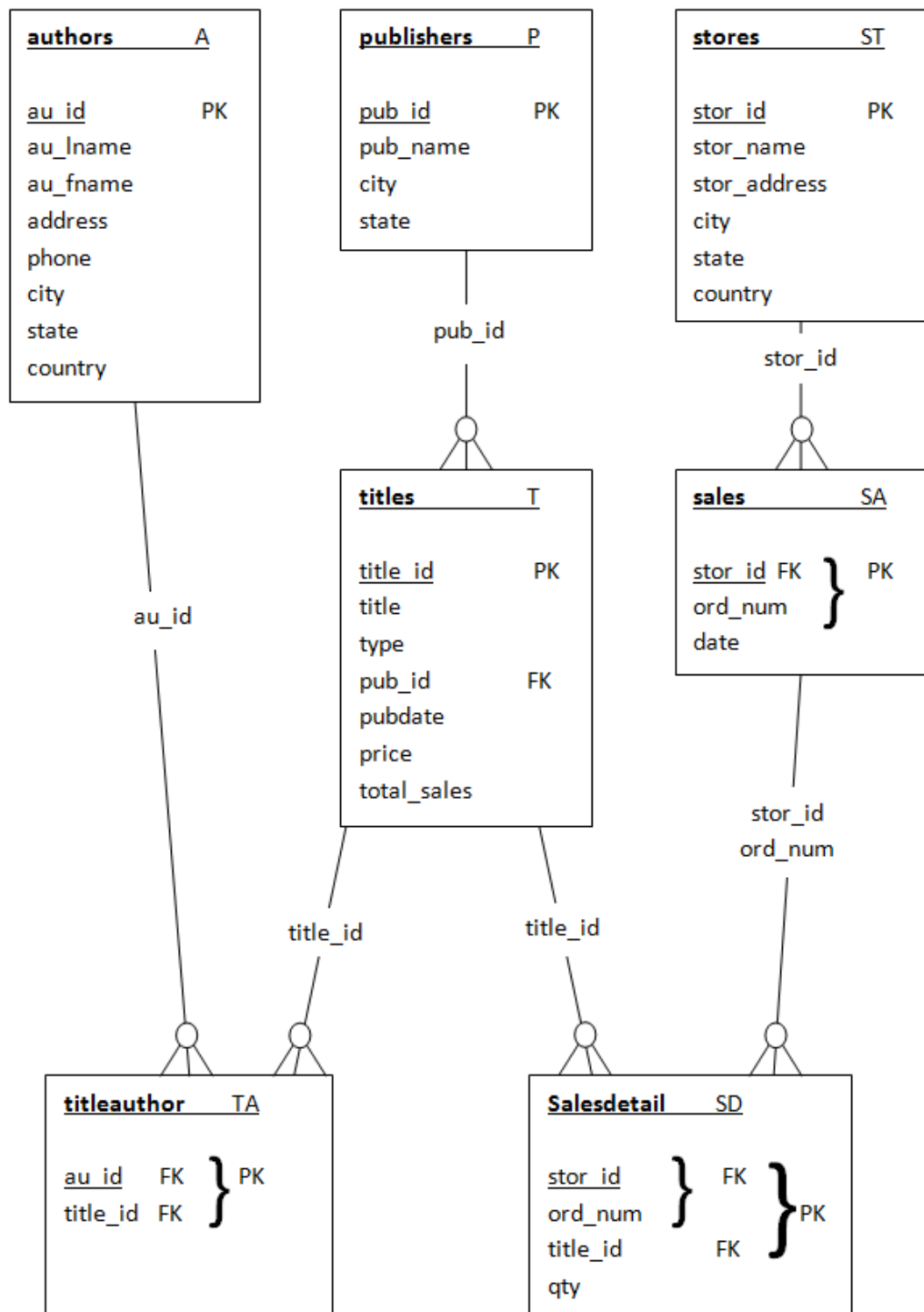
- **Si on fait n'importe quoi on passe à côté des avantages du modèle relationnel**
  - problèmes de redondance et de cohérence
  - performances médiocres de la base de données
- **Important de normaliser**
  - Voir cours de conception

# NORMALISATION

Est-ce normalisé ?

Non

-> Parfois on dénormalise  
exprès pour des  
raisons de performance



# MAINTENANT QUE L'ON PEUT CRÉER UNE BD, ON PEUT Y METTRE DES DONNÉES

```
INSERT INTO table [ ( colonne [, ...] ) ]  
    {    DEFAULT VALUES |  
        VALUES ( { expression | DEFAULT } [, ...] ) }
```

```
UPDATE table  
    SET colonne = { expression | DEFAULT }  [, ...]  
    [ WHERE condition ]
```

```
DELETE FROM table [ WHERE condition ]
```

# TABLE THEMES

```
CREATE TABLE projet.themes (  
    theme_id      SERIAL PRIMARY KEY,  
    couleurFond   VARCHAR(10) DEFAULT 'blanc',  
    couleurAvant  VARCHAR(10) DEFAULT 'noir',  
    CHECK (couleurFond<>couleurAvant),  
    UNIQUE (couleurFond,couleurAvant)  
)
```



# EXEMPLES D'INSERT

```
INSERT INTO projet.themes VALUES (DEFAULT, 'rouge', 'bleu');  
INSERT INTO projet.themes VALUES (DEFAULT, 'rouge', DEFAULT);  
INSERT INTO projet.themes DEFAULT VALUES;  
INSERT INTO projet.themes (couleurfond) VALUES ('vert');  
INSERT INTO projet.themes (couleurAvant, couleurfond) VALUES  
('rouge', 'bleu');
```

# EXEMPLES D'INSERT

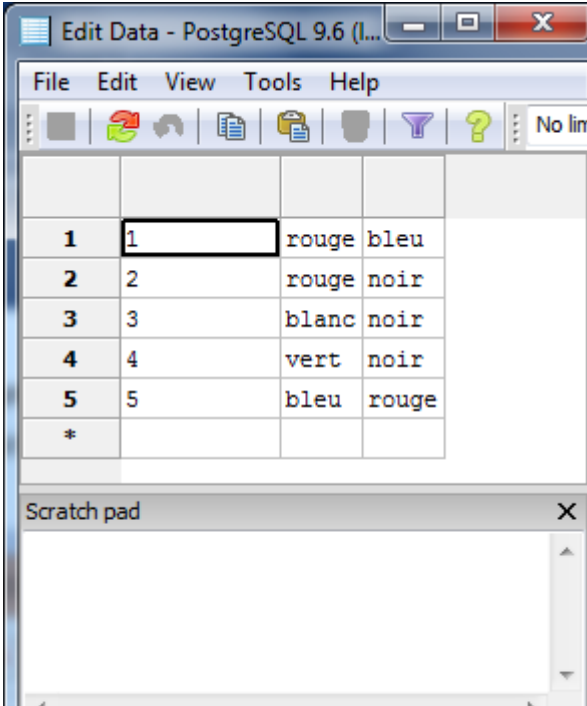
```
INSERT INTO projet.themes VALUES (DEFAULT, 'rouge', 'bleu');
```

```
INSERT INTO projet.themes VALUES (DEFAULT, 'rouge', DEFAULT);
```

```
INSERT INTO projet.themes DEFAULT VALUES;
```

```
INSERT INTO projet.themes (couleurfond) VALUES ('vert');
```

```
INSERT INTO projet.themes (couleurAvant, couleurfond) VALUES  
('rouge', 'bleu');
```



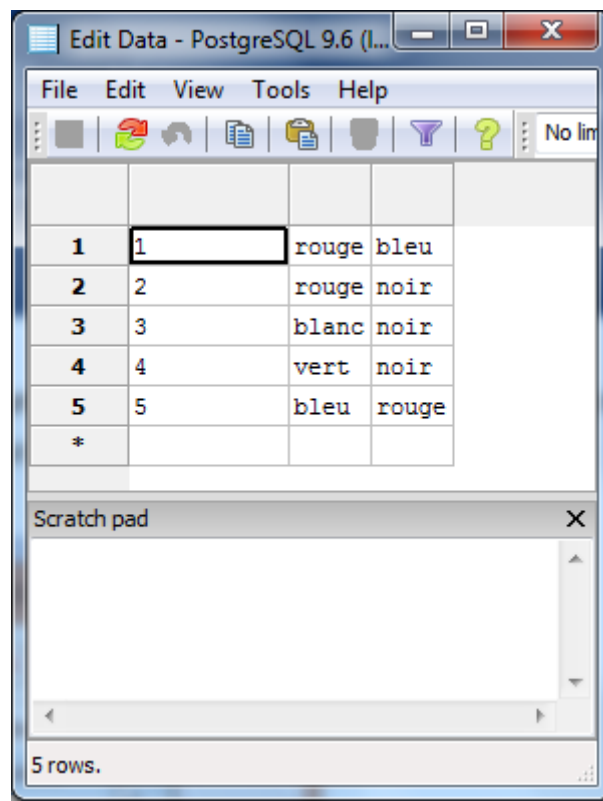
The screenshot shows the 'Edit Data - PostgreSQL 9.6' window. It displays a table with 5 rows and 4 columns. The first column contains row numbers (1-5), the second column contains IDs (1-5), and the next two columns contain color pairs. The first row is highlighted, and the first cell of the second column (ID 1) is selected.

1	1	rouge	bleu
2	2	rouge	noir
3	3	blanc	noir
4	4	vert	noir
5	5	bleu	rouge
*			

Below the table is a 'Scratch pad' area.

# EXEMPLE D'UPDATE

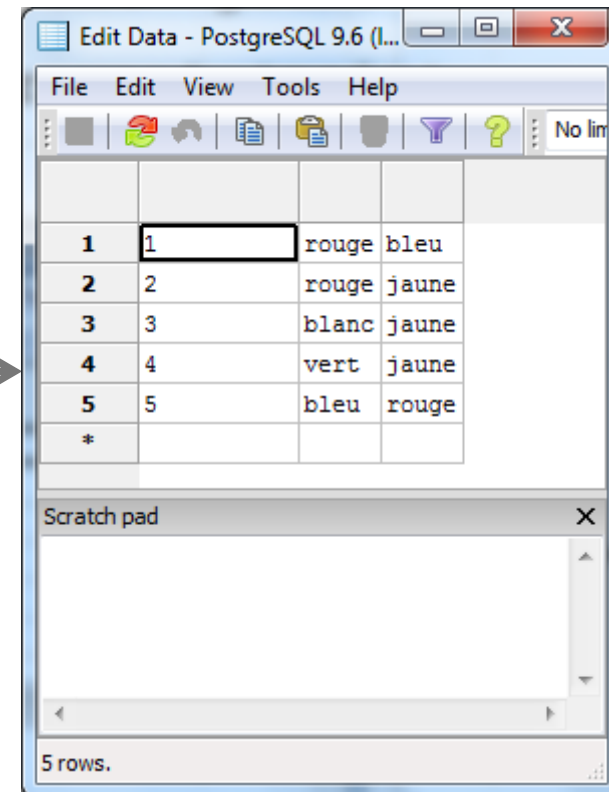
```
UPDATE projet.themes  
SET couleuravant='jaune'  
WHERE couleuravant='noir'
```



1	1	rouge	bleu
2	2	rouge	noir
3	3	blanc	noir
4	4	vert	noir
5	5	bleu	rouge
*			

Scratch pad

5 rows.



1	1	rouge	bleu
2	2	rouge	jaune
3	3	blanc	jaune
4	4	vert	jaune
5	5	bleu	rouge
*			

Scratch pad

5 rows.

# EXEMPLE DE DELETE

```
DELETE FROM projet.themes
```

```
WHERE couleurfond='vert '
```

**Attention aux contraintes d'intégrité !**

Edit Data - PostgreSQL 9.6 (l...)

1	1	rouge	bleu
2	2	rouge	jaune
3	3	blanc	jaune
4	4	vert	jaune
5	5	bleu	rouge
*			

Scratch pad

5 rows.



Edit Data - PostgreSQL 9.6 (l...)

1	1	rouge	bleu
2	2	rouge	jaune
3	3	blanc	jaune
4	5	bleu	rouge
*			

Scratch pad

4 rows.

# EXEMPLES DE DELETE

## **Pour effacer un tuple en particulier : PK**

```
DELETE FROM themes  
WHERE theme_id=1
```

## **Pour effacer tous les tuples :**

```
DELETE FROM themes
```