

# Avancement SAE

## S2.01 Conception et implémentation d'une base de données

Ibrahim BENKHERFELLAH   Axel COULET

Université Sorbonne Paris Nord  
BUT1 SD Semestre 2 BUT1 SD Semestre 2

May 23, 2025

# Table des matières

- 1 Introduction
- 2 Exploration et compréhension des données
- 3 Modélisation de la base de données
- 4 Script de création et peuplement SQL
- 5 Interrogation et visualisation des données
- 6 Conclusion

# Objectif du projet

- Comprendre et modéliser le commerce des technologies à faible émission de carbone.
- Concevoir une base de données normalisée à partir de deux jeux de données CSV.
- Interroger et visualiser les données pour produire des analyses pertinentes.

# Objectif du projet

- Comprendre et modéliser le commerce des technologies à faible émission de carbone.
- Concevoir une base de données normalisée à partir de deux jeux de données CSV.
- Interroger et visualiser les données pour produire des analyses pertinentes.

# Objectif du projet

- Comprendre et modéliser le commerce des technologies à faible émission de carbone.
- Concevoir une base de données normalisée à partir de deux jeux de données CSV.
- Interroger et visualiser les données pour produire des analyses pertinentes.

- Fichiers CSV étudiés :
  - `Trade_in_Low_Carbon_Technology_Products.csv`
  - `Bilateral_Trade_in_Low_Carbon_Technology_Products.csv`
- Exploration initiale avec Python (pandas) :  
`.head()`, `.info()`, etc.
- Identification de colonnes clés : `CTS_Code`, `Indicator`, `Trade_Flow`, etc.
- Difficultés rencontrées :
  - Colonnes de dates (comme `Year`) au mauvais format ou non reconnues comme `Date`.
  - Colonnes "`F1994`" à "`F2023`" en format long à dépivoter (via `UNPIVOT` de SQL).

- Fichiers CSV étudiés :
  - `Trade_in_Low_Carbon_Technology_Products.csv`
  - `Bilateral_Trade_in_Low_Carbon_Technology_Products.csv`
- Exploration initiale avec Python (pandas) :  
`.head()`, `.info()`, etc.
- Identification de colonnes clés : `CTS_Code`, `Indicator`, `Trade_Flow`, etc.
- Difficultés rencontrées :
  - Colonnes de dates (comme `Year`) au mauvais format ou non reconnues comme `Date`.
  - Colonnes "`F1994`" à "`F2023`" en format long à dépivoter (via `UNPIVOT` de SQL).

- Fichiers CSV étudiés :
  - `Trade_in_Low_Carbon_Technology_Products.csv`
  - `Bilateral_Trade_in_Low_Carbon_Technology_Products.csv`
- Exploration initiale avec Python (pandas) :  
`.head()`, `.info()`, etc.
- Identification de colonnes clés : `CTS_Code`, `Indicator`, `Trade_Flow`, etc.
- Difficultés rencontrées :
  - Colonnes de dates (comme `Year`) au mauvais format ou non reconnues comme `Date`.
  - Colonnes "`F1994`" à "`F2023`" en format long à dépivoter (via `UNPIVOT` de SQL).



- Fichiers CSV étudiés :
  - `Trade_in_Low_Carbon_Technology_Products.csv`
  - `Bilateral_Trade_in_Low_Carbon_Technology_Products.csv`
- Exploration initiale avec Python (pandas) :  
`.head()`, `.info()`, etc.
- Identification de colonnes clés : `CTS_Code`, `Indicator`, `Trade_Flow`, etc.
- Difficultés rencontrées :
  - Colonnes de dates (comme `Year`) au mauvais format ou non reconnues comme `Date`.
  - Colonnes "`F1994`" à "`F2023`" en format long à dépivoter (via `UNPIVOT` de SQL).

- Données structurées en colonnes d'années : F1994 à F2023.
- Présence de colonnes de description redondantes (CTS\_Code, CTS\_Name, CTS\_Full\_Descriptor).
- Première étape : lecture manuelle pour repérage des redondances et structures tabulaires.

- Données structurées en colonnes d'années : F1994 à F2023.
- Présence de colonnes de description redondantes (CTS\_Code, CTS\_Name, CTS\_Full\_Descriptor).
- Première étape : lecture manuelle pour repérage des redondances et structures tabulaires.

- Données structurées en colonnes d'années : F1994 à F2023.
- Présence de colonnes de description redondantes (CTS\_Code, CTS\_Name, CTS\_Full\_Descriptor).
- Première étape : lecture manuelle pour repérage des redondances et structures tabulaires.

- Conversion des colonnes FXXXX en format court (année, valeur).
- Normalisation des années (F1994 à F2023) :
  - Chaque colonne FXXXX devient une ligne avec une valeur d'année.
  - Transformation effectuée via `UNION ALL` dans SQL pour correspondre à la logique de `stack()` en Python.
- Problèmes rencontrés :
  - Ambiguïté sur le pays origine.
  - Duplication de lignes si filtrage incorrect.

- Conversion des colonnes FXXXX en format court (année, valeur).
- Normalisation des années (F1994 à F2023) :
  - Chaque colonne FXXXX devient une ligne avec une valeur d'année.
  - Transformation effectuée via `UNION ALL` dans SQL pour correspondre à la logique de `stack()` en Python.
- Problèmes rencontrés :
  - Ambiguïté sur le pays origine.
  - Duplication de lignes si filtrage incorrect.

- Conversion des colonnes FXXXX en format court (année, valeur).
- Normalisation des années (F1994 à F2023) :
  - Chaque colonne FXXXX devient une ligne avec une valeur d'année.
  - Transformation effectuée via `UNION ALL` dans SQL pour correspondre à la logique de `stack()` en Python.
- Problèmes rencontrés :
  - Ambiguïté sur le pays origine.
  - Duplication de lignes si filtrage incorrect.

# Transformation des données : format large vers format long

## Format large (wide) :

Chaque année est une colonne

...	Scale	F1994	F1995	F1996
...	Units	190885.0	2433262.0	17679060.0
...	Units	0.0004146	0.0053298	0.0347023
...	Units	668268200.0	612865500.0	493126000.0



## Format long (tidy) :

Chaque ligne correspond à une observation unique (année, valeur).

...	Scale	Year	Value
...	Units	F1994	190885.0
...	Units	F1994	0.0004146
...	Units	F1994	668268200.0



- Création d'une table `Bilateral_Trade` pour les données bilatérales :
  - Relation réflexive entre deux pays (deux clés étrangères vers `country`).
- Table `Trade` pour les données nationales.
- Lien systématique aux dimensions : pays, indicateur, catégorie, année.

- Création d'une table `Bilateral_Trade` pour les données bilatérales :
  - Relation réflexive entre deux pays (deux clés étrangères vers `country`).
- Table `Trade` pour les données nationales.
- Lien systématique aux dimensions : pays, indicateur, catégorie, année.

- Création d'une table `Bilateral_Trade` pour les données bilatérales :
  - Relation réflexive entre deux pays (deux clés étrangères vers `country`).
- Table `Trade` pour les données nationales.
- Lien systématique aux dimensions : pays, indicateur, catégorie, année.

# Schéma Entité-Association

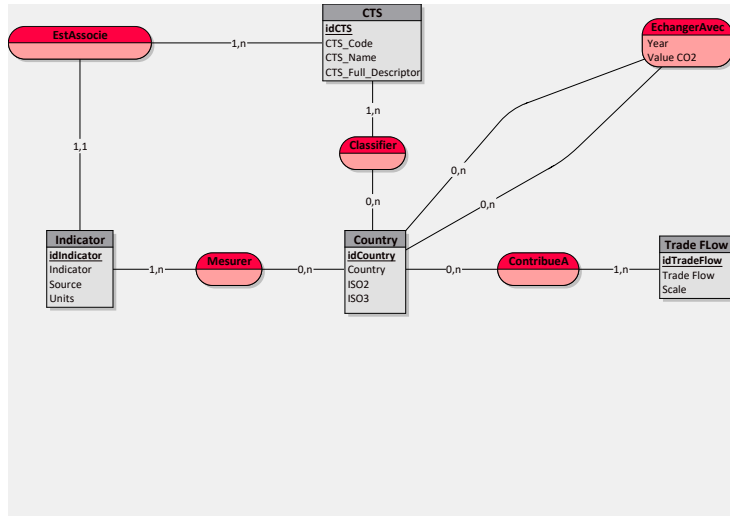


Figure: Schéma Entité-Association Initial

# Schéma Entité-Association

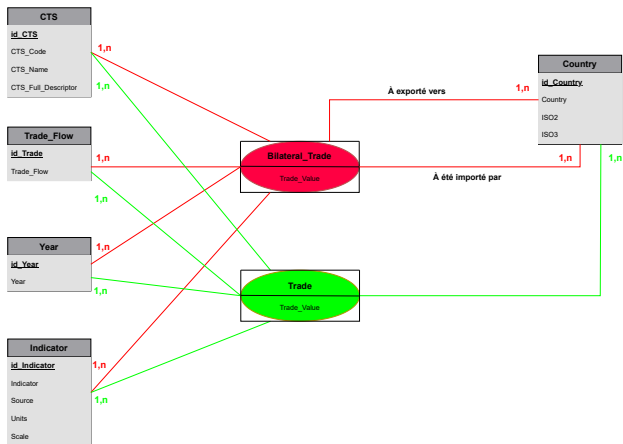


Figure: Schéma Entité-Association Final

- Type Entités : Country, Indicator, CTS, Trade\_Flow, Year
- Type Associations :
  - Bilateral\_Trade (réflexive sur Country)
  - Trade (relation simple avec agrégation)
- Justification des cardinalités et des relations

# Modèle conceptuel (EA)

- Type Entités : Country, Indicator, CTS, Trade\_Flow, Year
- Type Associations :
  - Bilateral\_Trade (réflexive sur Country)
  - Trade (relation simple avec agrégation)
- Justification des cardinalités et des relations

- Type Entités : Country, Indicator, CTS, Trade\_Flow, Year
- Type Associations :
  - Bilateral\_Trade (réflexive sur Country)
  - Trade (relation simple avec agrégation)
- Justification des cardinalités et des relations



- Schéma relationnel résultant de la modélisation EA :
  - Country(idCountry, Country, IS02, IS03)
  - CTS(idCTS, CTS\_Code, CTS\_Name, CTS\_Full\_Descriptor)
  - Trade\_Flow(id\_Trade\_Flow, Trade\_Flow, Scale)
  - Indicator(idIndicator, Indicator\_, Source, Units)
  - Year(id\_Year, Year)
  - Bilateral\_Trade(id\_country, id\_counterpart, id\_indicator, id\_cts, id\_Trade\_Flow, id\_year, trade\_value)
  - Trade(id\_country, id\_indicator, id\_cts, id\_Trade\_Flow, id\_year, trade\_value)

# Forme normale : 1NF

- Toutes les valeurs sont atomiques : aucun champ multivalué ou composé.
- Transformation des colonnes F1994 à F2023 en une colonne `id_year` via pivot SQL.
- Tables relationnelles sans redondance horizontale.
- **Conclusion** : le schéma respecte la **première forme normale (1NF)**.

- Le schéma est en 1NF.
- Aucune dépendance fonctionnelle partielle dans les tables à clés composites.
- Exemple : les descripteurs CTS ont été extraits dans une table distincte, reliée par CTS\_Code.
- **Conclusion** : toutes les dépendances fonctionnelles concernent la clé entière → **2NF validée**.

- Le schéma est en 2NF.
- Suppression des dépendances transitives.
- Exemple : l'unité d'un indicateur dépend uniquement de l'indicateur, pas d'un pays ou autre entité.
- **Conclusion** : toutes les colonnes non-clés dépendent uniquement de la clé primaire → **schéma en 3NF**.

- Toutes les dépendances fonctionnelles ont un antécédent qui est une super-clé.
- Exemple : dans `Bilateral_Trade`, seule la combinaison complète des identifiants détermine `trade_value`.
- Il n'existe pas de dépendance fonctionnelle violant cette condition.
- **Conclusion** : le schéma respecte la forme normale de **Boyce-Codd (BCNF)**.

# Création de la table Country

```
CREATE TABLE Country(  
    id_Country SMALLINT PRIMARY KEY,  
    Country VARCHAR(50),  
    ISO2 CHAR(2),  
    ISO3 CHAR(3)  
);
```

# Création de la table Indicator

```
CREATE TABLE Indicator(  
    id_Indicator SMALLINT PRIMARY KEY,  
    Indicator VARCHAR(80),  
    Source VARCHAR,  
    Units VARCHAR(50),  
    Scale VARCHAR(5)  
);
```

# Création de la table CTS

```
CREATE TABLE CTS(  
    id_CTS SMALLINT PRIMARY KEY,  
    CTS_Code VARCHAR(6),  
    CTS_Name VARCHAR(100),  
    CTS_Full_Descriptor VARCHAR(150)  
);
```



# Création de la table Trade\_Flow

```
CREATE TABLE Trade_Flow(  
    id_Trade_Flow SMALLINT PRIMARY KEY,  
    Trade_Flow VARCHAR(20)  
);
```

# Création de la table Year

```
CREATE TABLE Year(  
    id_Year SMALLINT PRIMARY KEY,  
    Year DATE  
);
```

# Création de la table Bilateral\_Trade

```
CREATE TABLE Bilateral_Trade (  
    id_Country SMALLINT REFERENCES Country(id_Country),  
    id_Counterpart SMALLINT REFERENCES Country(id_Country),  
    id_Indicator SMALLINT REFERENCES Indicator(id_Indicator),  
    id_CTS SMALLINT REFERENCES CTS(id_CTS),  
    id_Trade_Flow SMALLINT REFERENCES Trade_Flow(...),  
    id_Year SMALLINT REFERENCES Year(id_Year),  
    trade_value DOUBLE PRECISION,  
    PRIMARY KEY(id_Country, id_Counterpart_Country,  
                id_Indicator, id_CTS, id_Trade_Flow, id_Year)  
);
```

# Création de la table Trade

```
CREATE TABLE Trade (  
    id_Country SMALLINT REFERENCES Country(id_Country),  
    id_Indicator SMALLINT REFERENCES Indicator(id_Indicator),  
    id_CTS SMALLINT REFERENCES CTS(id_CTS),  
    id_Trade_Flow SMALLINT REFERENCES Trade_Flow(...),  
    id_Year SMALLINT REFERENCES Year(id_Year),  
    trade_value DOUBLE PRECISION,  
    PRIMARY KEY(id_Country, id_Indicator, id_CTS,  
    id_Trade_Flow, id_Year)  
);
```

- Utilisation de COPY, INSERT INTO avec jointures.
- Aligement avec les traitements Python (Avec `.stack()`).
- Cas particuliers traités :
  - Échanges d'un pays avec lui-même
  - Lignes NULL / Trade Flow = Not Applicable
- Difficultés rencontrées :
  - Contraintes techniques : NULL, unicité, jointures complexes.
  - Problèmes de données : fusion sans doublons, nettoyage, volume élevé.

- Utilisation de COPY, INSERT INTO avec jointures.
- Alignement avec les traitements Python (Avec `.stack()`).
- Cas particuliers traités :
  - Échanges d'un pays avec lui-même
  - Lignes NULL / Trade Flow = Not Applicable
- Difficultés rencontrées :
  - Contraintes techniques : NULL, unicité, jointures complexes.
  - Problèmes de données : fusion sans doublons, nettoyage, volume élevé.

- Évolution annuelle des importations mondiales de technologies bas-carbone.
- Top 5 technologies les plus échangées par type de flux
- Top pays exportateurs par valeur totale échangée (3 dernières années)
- Comparaison des échanges bilatéraux vs nationaux sur une année donnée

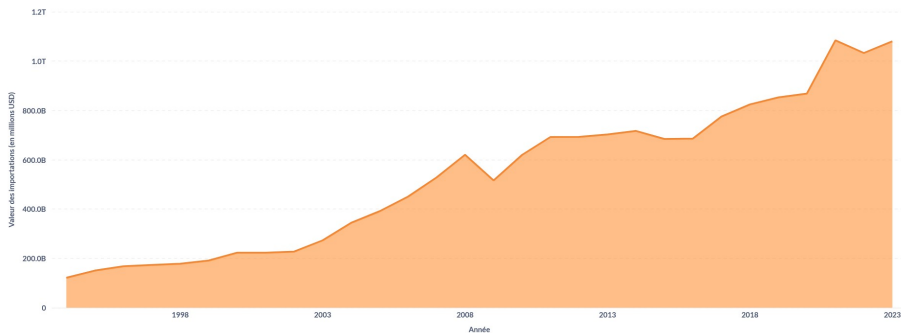
- Utilisation de Metabase pour interroger la BD
- Utilisation de Tableau pour valider les valeurs issues du CSV
- Comparaison SQL / Python → vérification de l'intégrité
- Exemples visuels :



# Requête SQL – Importations bas carbone

```
SELECT year, SUM(trade_value) AS trade_value
FROM Trade
JOIN year USING(id_year)
JOIN indicator USING(id_indicator)
JOIN trade_flow USING(id_trade_flow)
JOIN country USING(id_country)
WHERE trade_flow = 'Imports'
      AND indicator = 'Imports_of_low_carbon_technology_products'
      AND units ~* '.*dollar'
      AND country.iso2 IS NOT NULL
GROUP BY year;
```

Listing 1: Requête SQL pour les importations de technologies bas carbone



**Figure:** Évolution annuelle des importations mondiales de technologies bas-carbone

# Relation Tableau – Top 5 technologies

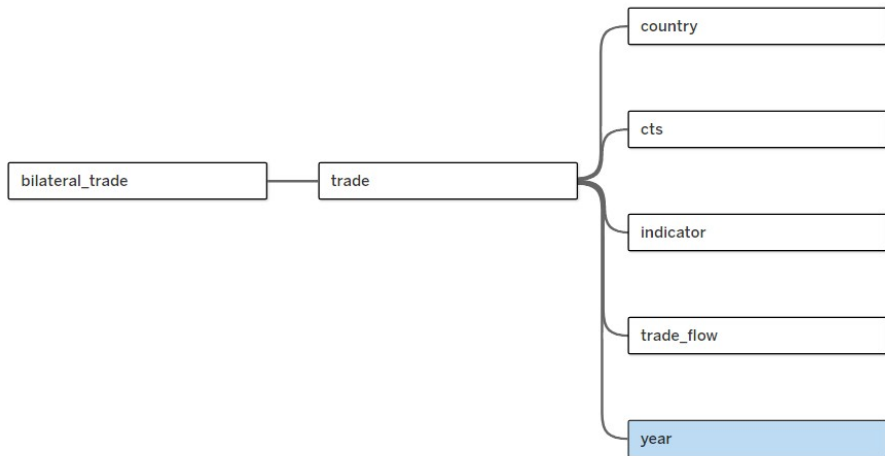


Figure: Relations établie sur Tableau Desktop

# Relation Tableau – Top 5 technologies

## Top 5 technologies les plus échangées par type de flux

Trade Flow	Cts Name	
Not Applicable	Trade in Low Carbon Technology Products	66540287104001
	Trade in Low Carbon Technology Products; Trade Balance	802826776438
	Trade in Low Carbon Technology Products; Comparative Advantage	1680
Exports	Trade in Low Carbon Technology Products; Exports	33671741958940
Imports	Trade in Low Carbon Technology Products; Imports	32886089437153

Figure: Top 5 technologies les plus échangées par type de flux

# Requête SQL – Top 10 pays exportateurs

```
SELECT
    country AS pays_exportateur ,
    100.0 * SUM(trade_value) / SUM(SUM(trade_value))
        OVER () AS total_exports
FROM Trade
JOIN country USING(id_country)
JOIN trade_flow USING(id_trade_flow)
JOIN year USING(id_year)
JOIN indicator USING(id_indicator)
WHERE trade_flow = 'Exports'
    AND units ~* '[*dollar]'
    AND iso2 IS NOT NULL
    AND EXTRACT(YEAR FROM year) >= 2021
GROUP BY country
ORDER BY total_exports DESC
LIMIT 10;
```

Listing 2: Requête SQL pour Top 10 pays exportateurs

# Visualisation – Top 10 pays exportateurs

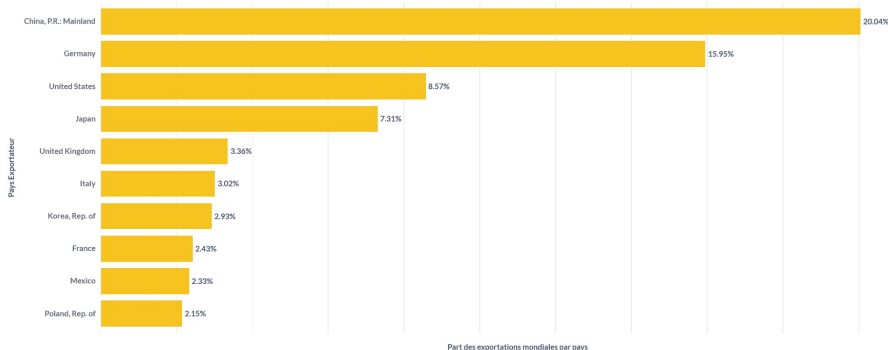


Figure: Top 10 pays exportateurs par valeur totale échangée (3 dernières années)

# Requête SQL – Échanges Bilatéraux VS Nationaux

```
WITH totals AS (  
    SELECT  
        y.year,  
        SUM(t.trade_value) AS total_trade,  
        SUM(bt.trade_value) AS total_bilateral  
    FROM year AS y  
    LEFT JOIN trade AS t ON t.id_year = y.id_year  
        AND t.id_indicator IN (  
            SELECT id_indicator FROM indicator WHERE units  
                ILIKE '%dollar%'  
        )  
    LEFT JOIN bilateral_trade AS bt ON bt.id_year = y.  
        id_year  
        AND bt.id_indicator IN (  
            SELECT id_indicator FROM indicator WHERE units  
                ILIKE '%dollar%'  
        )  
    ...  
)
```

Listing 3: Requête SQL pour Échanges Bilatéraux VS Nationaux

# Relation Tableau – Top 5 technologies

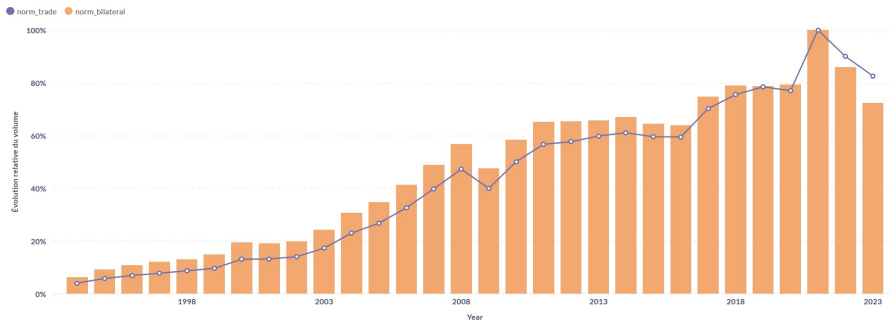


Figure: Comparaison des échanges Bilatéraux VS Nationaux



- Base de données fonctionnelle et fidèle au CSV (dans nos rêves)
- Respect des étapes du processus de modélisation
- Difficultés sur l'import (pivot, valeurs nulles, sens du flux)
- Vérification croisée avec Python
- Améliorations possibles :

Merci pour votre attention

**Questions ?**