Lab 2: Pentaho - Postgres

Installation

Faites l'installation de ces 3 composants:

- Postgres SQL
- Java (le fichier jdk)
- Pentaho PDI (zip)
- PowerBI Desktop

Les fichiers d'installation sont présents dans le lien Drive suivant:

Software - Google Drive
https://drive.google.com/drive/folders/1WovG70Hyv7NYHu9Eya17KbRTV99CrJHH?usp=sharing

Pendant l'installation de Postgres assurez vous de bien noter le mote de passe de l'instance dque vous allez créer.
Vous pouvez utiliser le mot de passe des labs précédents : **cei5thi1Ahng**

Description de l'exercice

Regardez les fichiers csv inputs et le pdf de description de l'exercice (dans le pdf on décrit une base de données mySQL alors dans notre cas on v la remplacer pas Postgres)

Base de landing

Ouvrez l'intéface graphique de gestion de Postgres et commencez à créer votre base de données de **landing**

Dans cette base créez les différentes tables.

Remarque: vérifiez bien les types des différentes colonnes des types

Vous pouvez faire la création manuellement ou utiliser le script suivant

```
-- Table: public.departement

-- DROP TABLE IF EXISTS public.departement;

CREATE TABLE IF NOT EXISTS public.departement
(
    nom text COLLATE pg_catalog."default",
    numero text COLLATE pg_catalog."default",
    region text COLLATE pg_catalog."default",
    superficie bigint,
    population bigint,
    densite numeric,
    CONSTRAINT departement_pkey PRIMARY KEY (numero)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.departement
    OWNER to postgres;

-- Table: public.product

-- DROP TABLE IF EXISTS public.product;

CREATE TABLE IF NOT EXISTS public.product
(
    id bigint NOT NULL,
    name text COLLATE pg_catalog."default",
```

```
    price numeric,
    CONSTRAINT product_pkey PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.product
    OWNER to postgres;

-- Table: public.purshase

-- DROP TABLE IF EXISTS public.purshase;

CREATE TABLE IF NOT EXISTS public.purshase
(
    id bigint NOT NULL,
    product bigint,
    quantity bigint,
    requester bigint,
    status text COLLATE pg_catalog."default",
    request_date date,
    validation_time smallint,
    CONSTRAINT purshase_pkey PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.purshase
    OWNER to postgres;

-- Table: public.user

-- DROP TABLE IF EXISTS public."user";

CREATE TABLE IF NOT EXISTS public.users
(
    id bigint NOT NULL,
    first_name text COLLATE pg_catalog."default",
    last_name text COLLATE pg_catalog."default",
    departement text COLLATE pg_catalog."default",
    CONSTRAINT user_pkey PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.users
    OWNER to postgres;
```

Pipeline de landing avec PDI

Les tables de landing sont à ce niveau crée mais pas encore peuplé, pour le faire on va se baser sur Pentaho PDI:

Dans une pipeline Pentaho on se base principalement sur 2 types d'objets: la **transformation** et le **job**

La transformation permet d'enchainer des opérations sur le même flux de données.

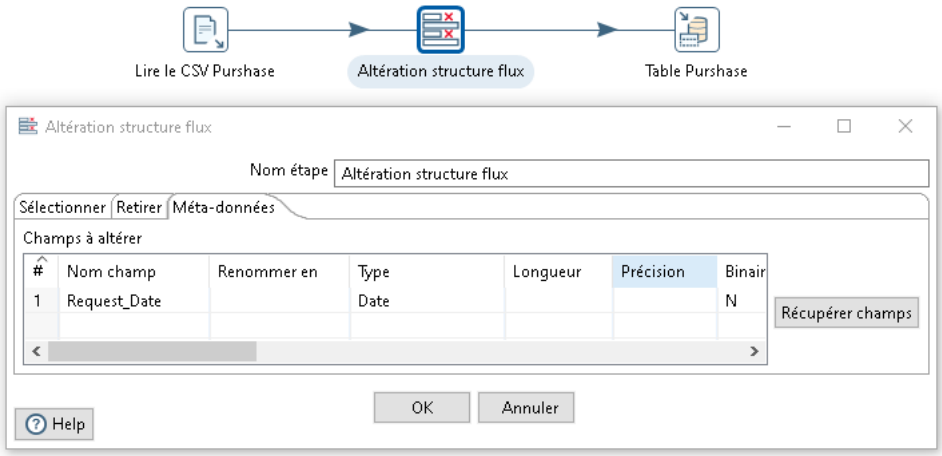
Le job permet de grouper des transformations et créer une séquence d'étapes de traitement.

Dans notre cas, on va crée 4 transformations et un job qui enchaine ces transformations l'une après l'autre. Chaque transformation va peupler l'une des 4 tables à partir du csv correspondant.

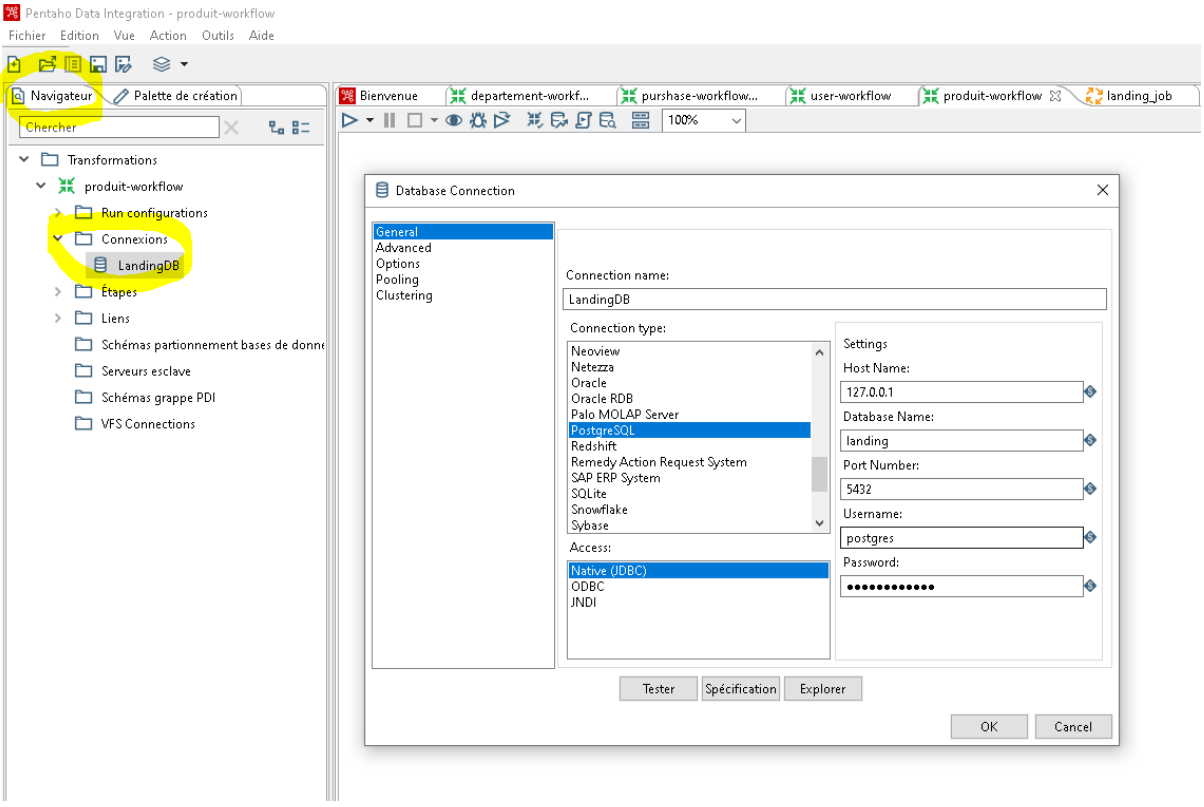
Remarque: On peut avoir une seule transformation qui fait les 4 tables au même temps en pratique, la séparation permet par contre une meilleure organisation et debuggage des flux plus facile.

Pour les transformations de landing, on aura besoin de 3 types de composants:

- Extraction depuis un fichier CSV
- Insertion/Mise à jour Table
- Alteration Structure Flux - Seulement pour "Purchase" pour convertir la colonne de date



Important - Connexion aux bases de données : Pour créer une connexion à la base de données de landing, il faudra le faire au niveau de l'une des transformations



Une fois la connexion crée, on la partage pour tous les autres transformation en faisant un **click-droit** sur la connexion dans le navigateur puis **partager**

De cette façon on n'est pas obligé de reconfigurer la même connexion pour chaque transformation

Une fois toutes les transformations faites on peut les englober dans un seul job

Remarques:

Pentaho stocke chaque transformation et job dans un fichier à part, essayez de bien renommer ces fichiers pour bien retrouver les flux

Datawarehouse

Nous avons à ce stade peuplé la base de landing et on va construire notre Datawarehouse

On commence par créer une nouvelle base nommée Datawarehouse dans note instance Postgres et on crée les tables de cette base.

On remarque que la table departement va disparaitre car on va l'inclure dans la table users pour converger vers un schéma en étoiles

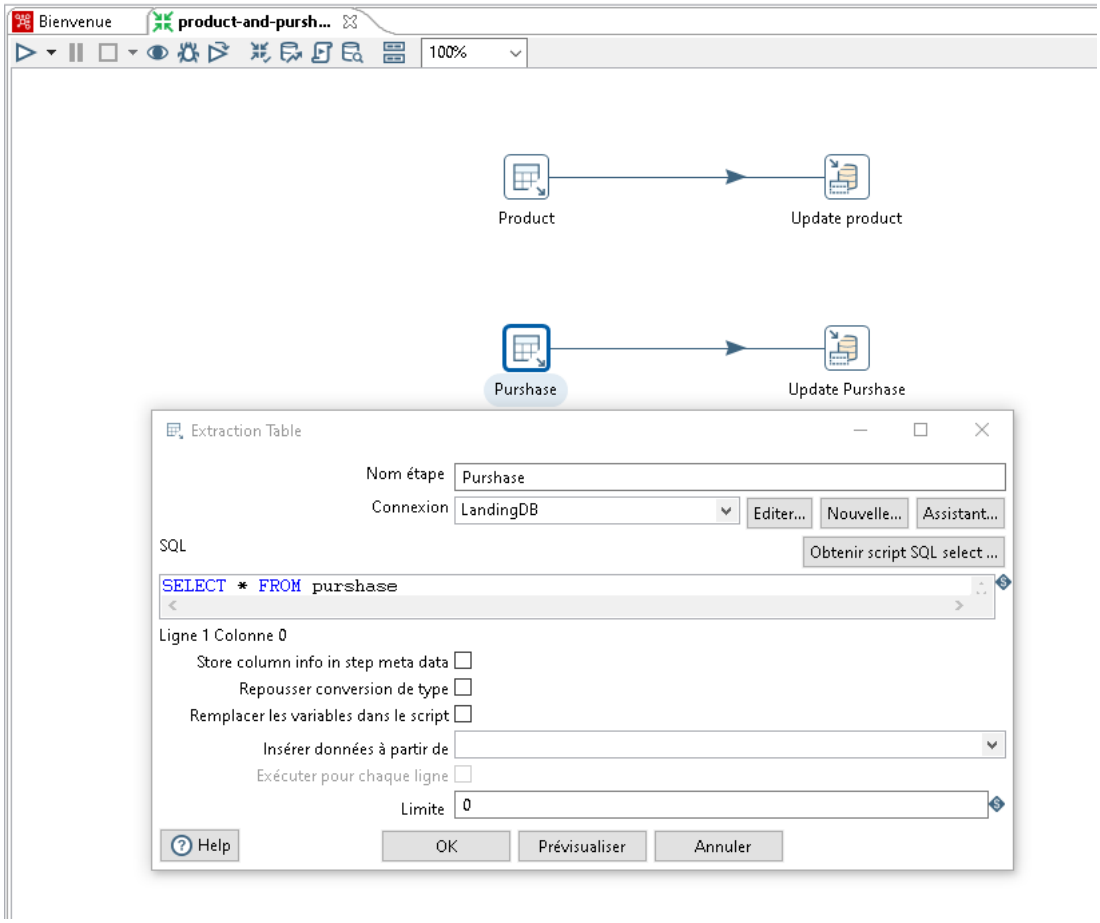
```
-- Table: public.product

-- DROP TABLE IF EXISTS public.product;

CREATE TABLE IF NOT EXISTS public.product
(
    id bigint NOT NULL,
    name text COLLATE pg_catalog."default",
    price numeric,
    CONSTRAINT product_pkey PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.product
    OWNER to postgres;

-- Table: public.purshase

-- DROP TABLE IF EXISTS public.purshase;

CREATE TABLE IF NOT EXISTS public.purshase
(
    id bigint NOT NULL,
    product bigint,
    quantity bigint,
    requester bigint,
    status text COLLATE pg_catalog."default",
    request_date date,
    validation_time smallint,
    CONSTRAINT purshase_pkey PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.purshase
    OWNER to postgres;

-- Table: public.user

-- DROP TABLE IF EXISTS public."user";

CREATE TABLE IF NOT EXISTS public.users
(
    id bigint NOT NULL,
    first_name text COLLATE pg_catalog."default",
    last_name text COLLATE pg_catalog."default",
    departement text COLLATE pg_catalog."default",
    nom_departement text COLLATE pg_catalog."default",
    region_departement text COLLATE pg_catalog."default",
    superficie_departement bigint,
    population_departement bigint,
    densite_departement numeric,
    CONSTRAINT user_pkey PRIMARY KEY (id)
```

```
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.users
    OWNER to postgres;
```

Pipeline de transformation

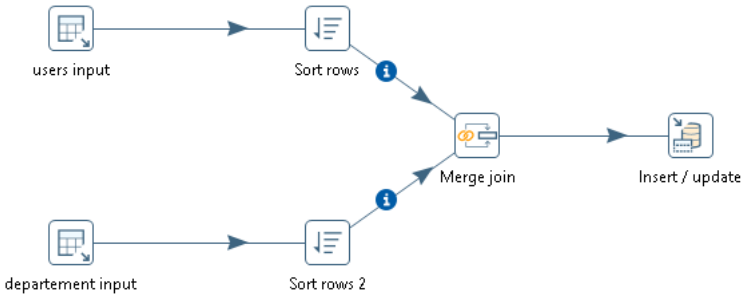
Dans cette nouvelle pipeline on va principalement bouger la données de la base de landing vers notre datawarehouse et faire un merge entre les tables users et departement

Concerant les tables purchase et product il s'agit d'un simple transfert de données



Dans le neoud extraction une requête SQL permet de spécifier la nature d'extraction, on va simplement copier toutes la données dans notre cas

Pour les deux tables restantes on fait in join pour mettre toute l'information dans la table users du Datawrehouse



Ici on ajoute deux autres types de noeuds: Sort rows et merge join.

Une spécificité de Pentaho fait qu'on a besoin de faire le tri explicitement sur les clés de jointure des deux tables avant de faire la jointure pour garantir un résultat fiable à la sortie

Date dimension

Dans la base de données Datawarehouse créez une nouvelle table de dates

```
CREATE TABLE dates AS (
    SELECT
        d::date AS "Date",
        EXTRACT(YEAR FROM d)::text AS "Year",
        EXTRACT(MONTH FROM d)::text AS "Month Number",
        'M' || EXTRACT(MONTH FROM d)::text AS "Month",
        EXTRACT(DAY FROM d)::text AS "Day",
        TO_CHAR(d, 'YYYYMM') AS "Year Month",
        TO_CHAR(d, 'Month') AS "Month Name",
        (EXTRACT(DOW FROM d)::integer + 1)::text AS "Day Number", -- PostgreSQL counts Sunday as 0, so we add 1 to match the DAX versi
        TO_CHAR(d, 'Day') AS "Day Name"
    FROM generate_series(
        '2019-01-01'::date,
        '2021-12-31'::date,
        '1 day'::interval
    ) AS d
);
```

Dataviz avec PowerBI

Créez un nouveau projet PowerBI et ajoutez les tables de votre datawarehouse en mode import

1. Ajoutez/corrigez les relations entre les tables Dans le schéma en étoile
2. Dans la table purchase ajoutez une nouvelle colonne de montant:

```
Amount = [quantity] * RELATED('public product'[price])
```

3. Représentez les indicateurs suivants

- Le chiffre d'affaires par produit par année dans un "clustered bar chart"
- Le nombre d'achats par année dans un tableau
- Les quantités d'achats par année dans un tableau
- L'évolution des quantités par mois (YYYYMM) dans un "line chart"