

Tutorial 4 Import Export de données via un fichier json.

Table des matières

- [Objectifs](#)
- [Ressources](#)
- [Pré-Requis](#)

Objectifs

Dans ce TP nous allons utiliser le format **json** pour échanger des données. A la différence de l'xml C'est une pratique assez peu courante. Mais ce format est beaucoup utilisé dans les webServices.

Comme pour Xml IBM se concentrant sur SQL l'a enrichi avec des instructions pour manipuler des fichiers XML provenant de l'IFS :

- [IFS_READ](#)
- [IFS_WRITE](#)
- [Working with JSON data](#)

Dans notre cas précis nous allons utiliser un script SQL

Ainsi nous allons :

- Écrire un script jsonOut.sql dans acs pour extraire les films de la table DVD dans un fichier JSON.
- Modifier les données depuis un éditeur style vs.
- Écrire un script jsonIN.sql pour extraire les données du fichier json avec sélection sur le genre.
- Intégrer ces instructions dans un programme RPG dspJson.sqlrpgle pour afficher les films.

en général suite à cette extraction nous écrivons un programme sqlRPg qui intègre les modifications dans notre modèle métier.

Ressources

- Environnement
- Temps : 45 mn.

Pré-Requis

- Accéder à internet.

Énoncé

Etape 1 Découvrons le fichier xml résultant.

1. Ouvrez le fichier [listeFilms]

```
{
  "Liste_dvd": [
    { "code": 10, "titre": "Bonjour de JSON", "genre": "S" },
    { "code": 20, "titre": "Star Wars - L'attaque des clones", "genre": "S" },
    {
      "code": 30,
      "titre": "Le seigneur des anneaux - Les deux tours",
      "genre": "F"
    },
    ...
  ],
  { "code": 430, "titre": "Pirates des caraïbes", "genre": "A" },
  { "code": 440, "titre": "Kill Bill", "genre": "A" },
  { "code": 450, "titre": "Master and Commander", "genre": "A" },
  { "code": 460, "titre": "Le dernier Samouraï", "genre": "A" }
]
```

c'est un objet "Liste_dvd". C'est objet est un tableau d'objet.

```
{ "code": 430, "titre": "Pirates des caraïbes", "genre": "A" }
```

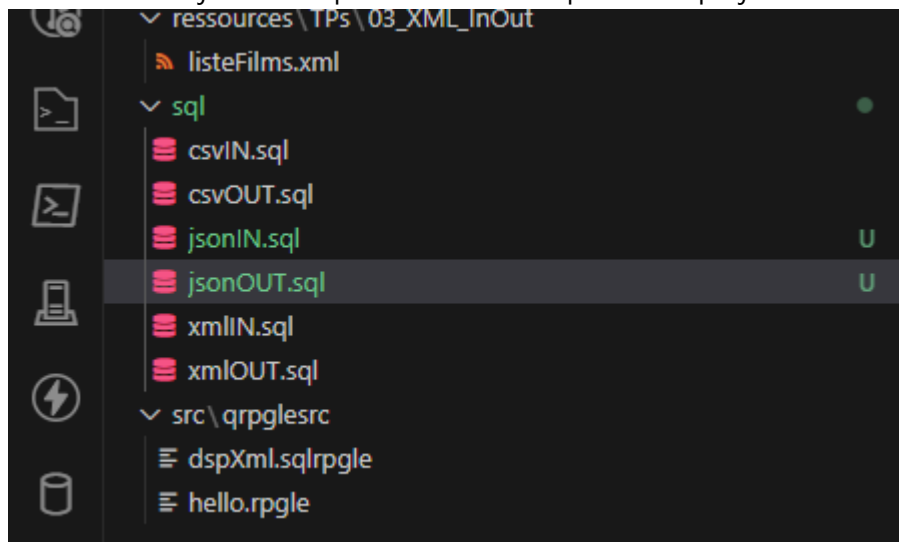
chaque item du tableau est un objet composé :

- d'un code `"code": 430`
- d'un titre `"titre": "Pirates des caraïbes"`
- d'un genre `"genre": "A"`

Json utilise la notation "clé" : valeur

Etape 2 JSON OUT

1. créer un fichier jsonOUT.sql dans le dossier sql de votre projet.



Pour arriver à notre résultat nous allons procéder par étapes :

- utiliser `json_object` pour créer un objet json représentant un dvd pour chaque ligne de la table DVD.
- utiliser `json_arrayagg` pour regrouper les objets json dvd dans un tableau.
- utiliser `json_object` pour mettre ce tableau dans un objet json notre liste de dvd.

1. création des objets DVD json

Un objet json est décrit sous la forme "`clé`":`valeur`.

`json_object` utilise cette syntaxe

```
json_object ('clé1' value <zoneTableContenantValeurClé1>,  
            'clé2' value <zoneTableContenantValeurClé2>,  
            ....  
            'cléN' value <zoneTableContenantValeurCléN>)
```

ajoutons ce ligne dans notre script.

```
-- creation d'un objet json pour chaque ligne ==> représentant un dvd au format  
json.  
select json_object ('code' value coddvd,  
                   'titre' value titre,  
                   'genre' value genre)  
from yv.dvd;
```

The screenshot shows an IDE with a tab labeled 'jsonOUT.sql U' selected. The SQL script in the editor is:

```
sql > jsonOUT.sql  
6  -- creation d'un objet json pour chaque ligne ==> représentant un dvd au format json.  
7  select json_object ('code' value coddvd,  
8                     'titre' value titre,  
9                     'genre' value genre)  
10 from yv.dvd;  
11
```

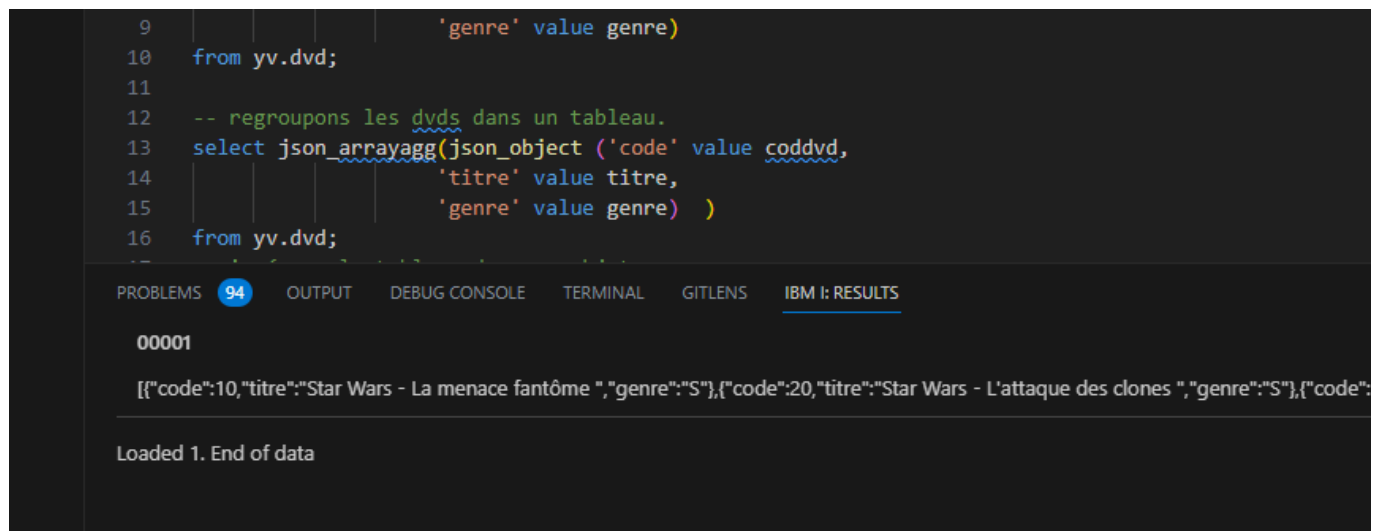
Below the editor, the 'IBM I: RESULTS' tab is active, displaying the output of the query. The output starts with '00001' and lists 10 JSON objects, each representing a DVD record with fields 'code', 'titre', and 'genre'.

```
00001  
{ "code":10,"titre":"Star Wars - La menace fantôme ","genre":"S"}  
{ "code":20,"titre":"Star Wars - L'attaque des clones ","genre":"S"}  
{ "code":30,"titre":"Le seigneur des anneaux - Les deux tours","genre":"F"}  
{ "code":40,"titre":"Matrix ","genre":"S"}  
{ "code":50,"titre":"Spiderman ","genre":"F"}  
{ "code":60,"titre":"Jurassik Park ","genre":"F"}  
{ "code":70,"titre":"Jurassik Park - Le monde perdu ","genre":"F"}  
{ "code":80,"titre":"Jurassik Park III ","genre":"F"}  
{ "code":90,"titre":"Minority Report ","genre":"S"}  
{ "code":100,"titre":"Star Wars - Un nouvel espoir ","genre":"S"}
```

1. Regroupons les objets **DVD** dans un tableau json. Le format json dérive des objets javascript utilise beaucoup les tableaux. ce qui lui permet d'être plus léger que le format XML dit plus verbeux. l'instruction **json_arrayagg** permet de consolider toutes les lignes sélectionnées par la requête SQL sous la forme d'un tableau.

ajouter cette ligne dans votre script.

```
-- regroupons les dvds dans un tableau.  
select json_arrayagg(json_object ('code' value coddvd,  
                                'titre' value titre,  
                                'genre' value genre) )  
from yv.dvd;
```



The screenshot shows an IDE with a dark theme. The editor displays the following SQL code:

```
9      'genre' value genre)  
10  from yv.dvd;  
11  
12  -- regroupons les dvds dans un tableau.  
13  select json_arrayagg(json_object ('code' value coddvd,  
14      'titre' value titre,  
15      'genre' value genre) )  
16  from yv.dvd;
```

Below the editor, there is a panel with tabs: PROBLEMS (94), OUTPUT, DEBUG CONSOLE, TERMINAL, GITLENS, and IBM I: RESULTS. The 'IBM I: RESULTS' tab is active, showing the following output:

```
00001  
[{"code":10,"titre":"Star Wars - La menace fantôme ","genre":"S"}, {"code":20,"titre":"Star Wars - L'attaque des clones ","genre":"S"}, {"code":30,"titre":"Star Wars - Le retour du Jedi ","genre":"S"}, {"code":40,"titre":"Star Wars - Les Jedi tombent ","genre":"S"}, {"code":50,"titre":"Star Wars - Le réveil de la Force ","genre":"S"}, {"code":60,"titre":"Star Wars - Le dernier Jedi ","genre":"S"}, {"code":70,"titre":"Star Wars - Le monde du Jedi ","genre":"S"}, {"code":80,"titre":"Star Wars - Le monde du Jedi 2 ","genre":"S"}, {"code":90,"titre":"Star Wars - Le monde du Jedi 3 ","genre":"S"}, {"code":100,"titre":"Star Wars - Le monde du Jedi 4 ","genre":"S"}]  
Loaded 1. End of data
```

1. enfin plaçons notre tableau dans un objet json
ajouter cette ligne dans votre script.

```
--- insérons le tableau dans un objet.  
select json_object ('Liste_dvd' value  
                  json_arrayagg(json_object ('code' value coddvd,  
                                              'titre' value trim(titre),  
                                              'genre' value genre)) )  
from yv.dvd;
```

```

16  from yv.dvd;
17  -- insérons le tableau dans un objet.
18  select json_object('Liste_dvd' value
19  ..... json_arrayagg(json_object('code' value coddvd,
20  ..... 'titre' value trim(titre),
21  ..... 'genre' value genre)) )
22  from yv.dvd;
23  -- insérons ce document dans un fichier de l'IFS

```

PROBLEMS 94 OUTPUT DEBUG CONSOLE TERMINAL GITLENS IBM I: RESULTS

00001

```

{"Liste_dvd":[{"code":10,"titre":"Star Wars - La menace fantôme","genre":"S"},{"code":20,"titre":"Star Wars - L'att

```

Loaded 1. End of data

1. creation du fichier json dans l'IFS

afin de créer directement via sql notre fichier IFS avec le contenu du résultat de notre requête précédente, nous allons utiliser la procédure SQL [IFS_WRITE](#)

Pour ce faire nous devons préciser :

- **PATH_NAME** => le chemin complet du fichier IFS
- **overwrite** => le mode de gestion ici nous serons en **overwrite** cad annule et remplace du contenu.
- **line** => les lignes à écrire.
- **FILE_CCSID** => le CCSID du fichier résultant, nous choisissons **1208** pour l'UTF-8.

- ajouter cette ligne dans votre script.

⚠️ veuillez à modifier le contenu de la ligne pour préciser le chemin de votre home sur l'IFS.

```

-- insérons ce document dans un fichier de l'IFS.
CALL QSYS2.IFS_WRITE(
  PATH_NAME => '/home/YV/jsonOUT.json',
  overwrite => 'REPLACE',
  LINE => (
select json_object ('Liste_dvd' value
  json_arrayagg(json_object ('code' value coddvd,
    'titre' value trim(titre),
    'genre' value genre)) )
from yv.dvd),
  FILE_CCSID => 1208);

```

au lancement , nous obtenons .

```
21 |         'genre' value genre)) )
22 | from yv.dvd;
23 | -- insérons ce document dans un fichier de l'IFS.
24 | CALL QSYS2.IFS_WRITE(.....
25 |     PATH_NAME => '/home/YV/jsonOUT.json',.....
26 |     overwrite => 'REPLACE',.....
27 |     LINE => (.....
28 | select json_object ('Liste_dvd' value .....
29 |     json_arrayagg(json_object ('code' value coddvd, .....
30 |         'titre' value trim(titre), .....
31 |         'genre' value genre)).....
32 | from yv.dvd),.....
33 |     FILE_CCSID => 1208);
```

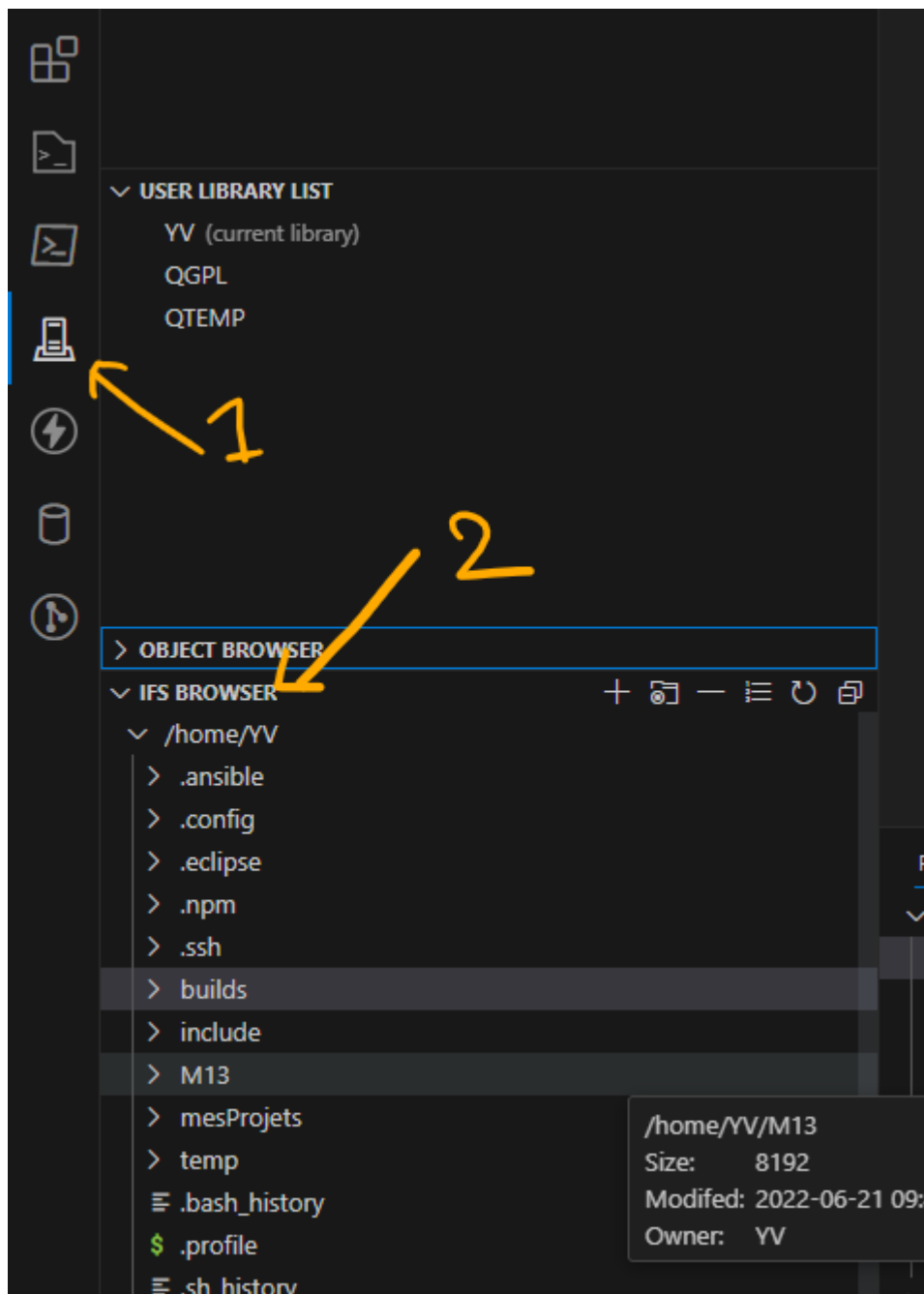
PROBLEMS 94 OUTPUT DEBUG CONSOLE TERMINAL GITLENS IBM I: RESULTS

Query executed with no data returned.

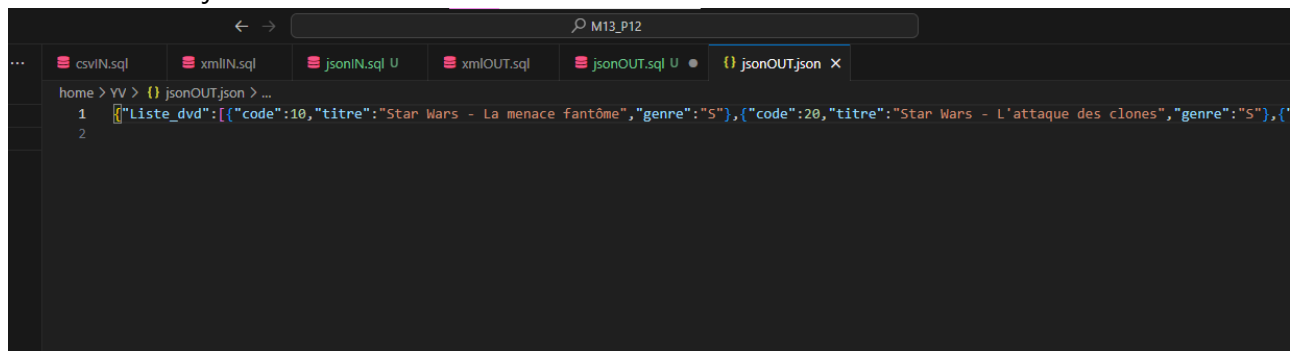
Etape 2 modifier les données

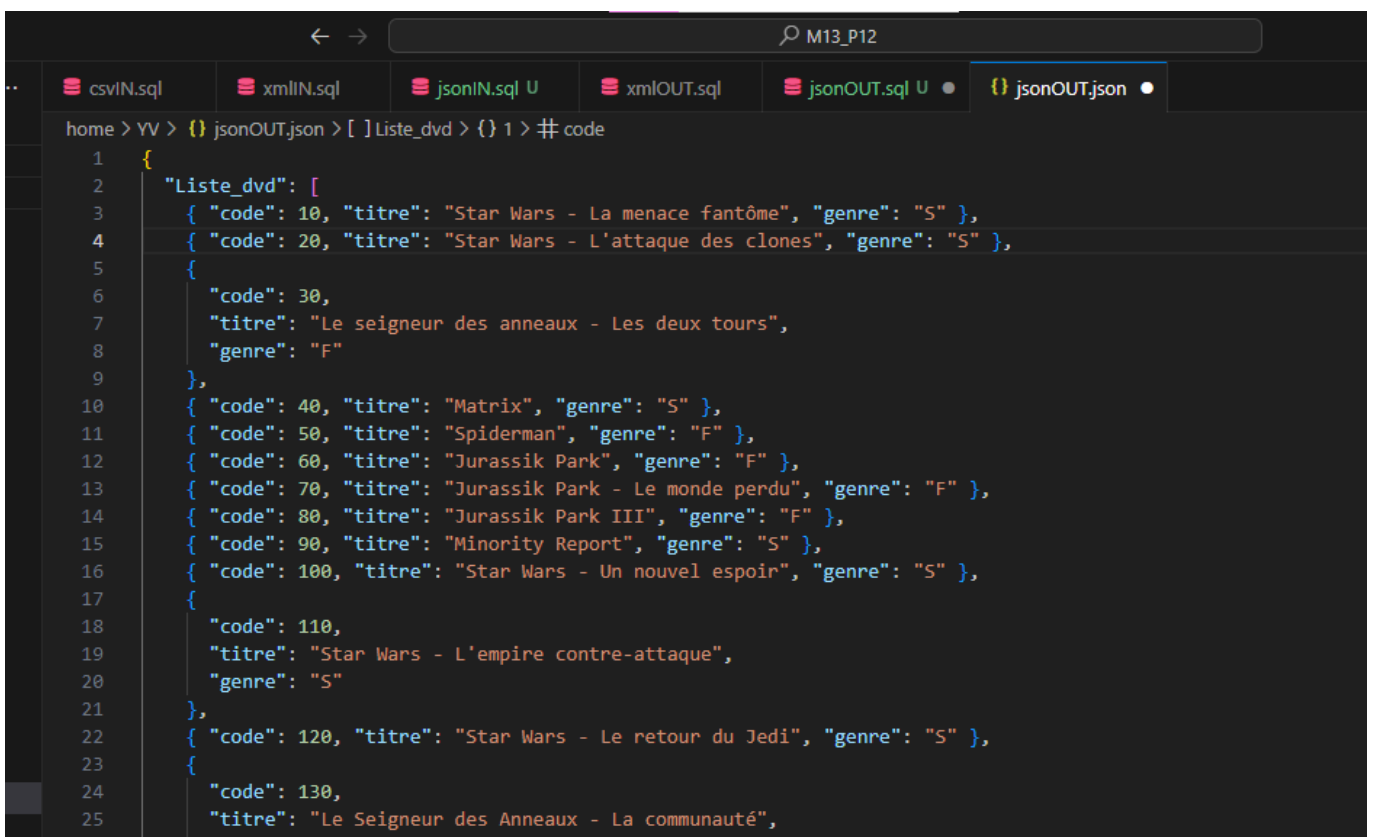
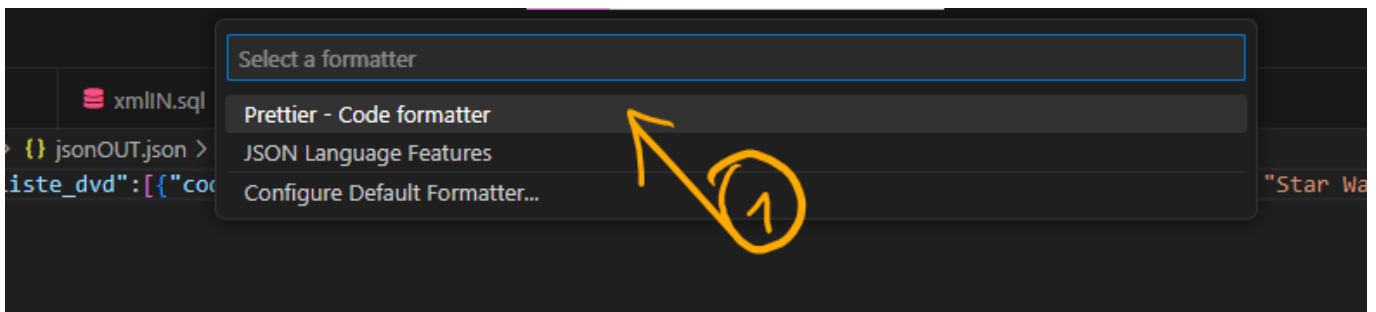
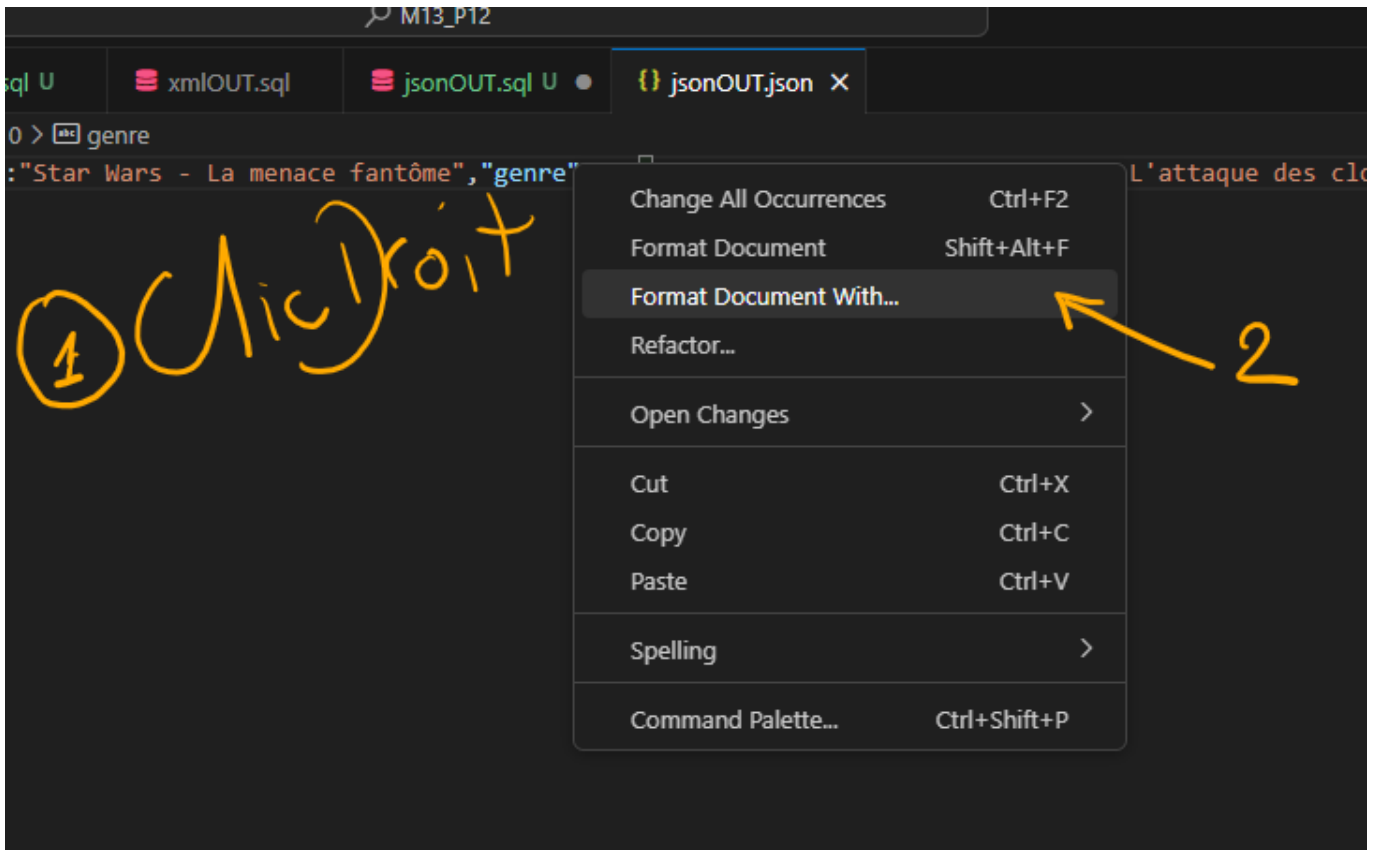
1. visualiser le fichier json dans VSC

- Ouvrir la vue **IFS BROWSER** dans C4i.



- Ouvrir le fichier jsonOUT.xml





- Modifier la première ligne

```

1  {
2    "Liste_dvd": [
3      { "code": 10, "titre": "Bonjour de JSON", "genre": "S" },
4      { "code": 20, "titre": "Star Wars - L'attaque des clones", "genre": "S" },
5      {
6        "code": 30,
7        "titre": "Le seigneur des anneaux - Les deux tours",
8        "genre": "F"
9      },
10     { "code": 40, "titre": "Matrix", "genre": "S" },
11     { "code": 50, "titre": "Spiderman", "genre": "F" },
12     { "code": 60, "titre": "Jurassik Park", "genre": "F" },

```

Etape 3 JSON IN

1. créer un script sql jsonIN.sql

2. visualiser le contenu du fichier sur l'IFS.

Nous allons utiliser la procédure [IFS_READ](#) pour afficher le contenu du fichier json directement dans une pseudo table sql.

- ajouter cette ligne dans votre script.

```

-- lecture du fichier xml sur l'IFS.
SELECT * FROM TABLE(QSYS2.IFS_READ('/home/YV/jsonOUT.json'));

```

```

4  delete from yv.dvdin;
5
6  -- lecture du fichier xml sur l'IFS.
7  SELECT * FROM TABLE(QSYS2.IFS_READ('/home/YV/jsonOUT.json'));
8

```

LINE_NUMBER	LINE
1	{
2	"Liste_dvd": [
3	{ "code": 10, "titre": "Bonjour de JSON", "genre": "S" },
4	{ "code": 20, "titre": "Star Wars - L'attaque des clones", "genre": "S" },
5	{
6	"code": 30,
7	"titre": "Le seigneur des anneaux - Les deux tours",
8	"genre": "F"
9	},
10	{ "code": 40, "titre": "Matrix", "genre": "S" },

Chaque ligne de notre fichier donne lieu à une ligne dans notre table.

1. intégrer les données dans la table de travail.

Il nous reste à décoder le contenu de la table json restituée pour en extraire le contenu des balises xml comme des zones d'une table sql.

Pour ce faire nous utilisons ,l'instruction [JSON_TABLE](#).

Pour ce faire nous devons préciser :

- notre document JSON ici nous utiliserons [get_clob_from_file](#) pour retourner le contenu du fichier IFS sous la forme d'un CLOB format interprétable comme un document json.

- `'$.Liste_dvd'` précise l'élément de référence pour une ligne de la table dans le document json cad un poste du tableau json.

```

home > YV > {} jsonOUT.json > [ ] Liste_dvd
1  {
2    "Liste_dvd": [
3      { "code": 10, "titre": "Bonjour de JSON", "genre": "S" },
4      { "code": 20, "titre": "Star Wars - L'attaque des clones", "genre": "S" },
5      {
6        "code": 30,
7        "titre": "Le seigneur des anneaux - Les deux tours",
8        "genre": "F"
9      },
10     { "code": 40, "titre": "Matrix", "genre": "S" },
11     { "code": 50, "titre": "Spiderman", "genre": "F" },
12     { "code": 60, "titre": "Jurassik Park", "genre": "F" },
13     { "code": 70, "titre": "Jurassik Park - Le monde perdu", "genre": "F" },

```

- **COLUMNS** reste à indiquer nos colonnes.
- **CODEDVD INTEGER PATH '\$.code'**, ma table sql aura une zone `code_dvd` qui sera un entier son contenu sera celui de l'attribut json `code` ou plus précisément `/Liste_dvd/code`

```

-- transformons le document xml de l'ifs en table sql ?
SELECT CODEDVD,TITRE,GENRE
FROM JSON_TABLE(
  get_clob_from_file('/home/YV/jsonOUT.json'),
  '$.Liste_dvd'
  COLUMNS(
    CODEDVD INTEGER PATH '$.code',
    TITRE varchar(52) PATH '$.titre',
    GENRE CHAR(1) PATH '$.genre'
  )
) AS a WITH CS;

```

```

9
10 -- transformons le document xml de l'ifs en table sql ?
11 SELECT CODEDVD,TITRE,GENRE
12 FROM JSON_TABLE(
13   get_clob_from_file('/home/YV/jsonOUT.json'),
14   '$.Liste_dvd'
15   COLUMNS(
16     CODEDVD INTEGER PATH '$.code',
17     TITRE varchar(52) PATH '$.titre',
18     GENRE CHAR(1) PATH '$.genre'
19   )
20 ) AS a WITH CS;
21

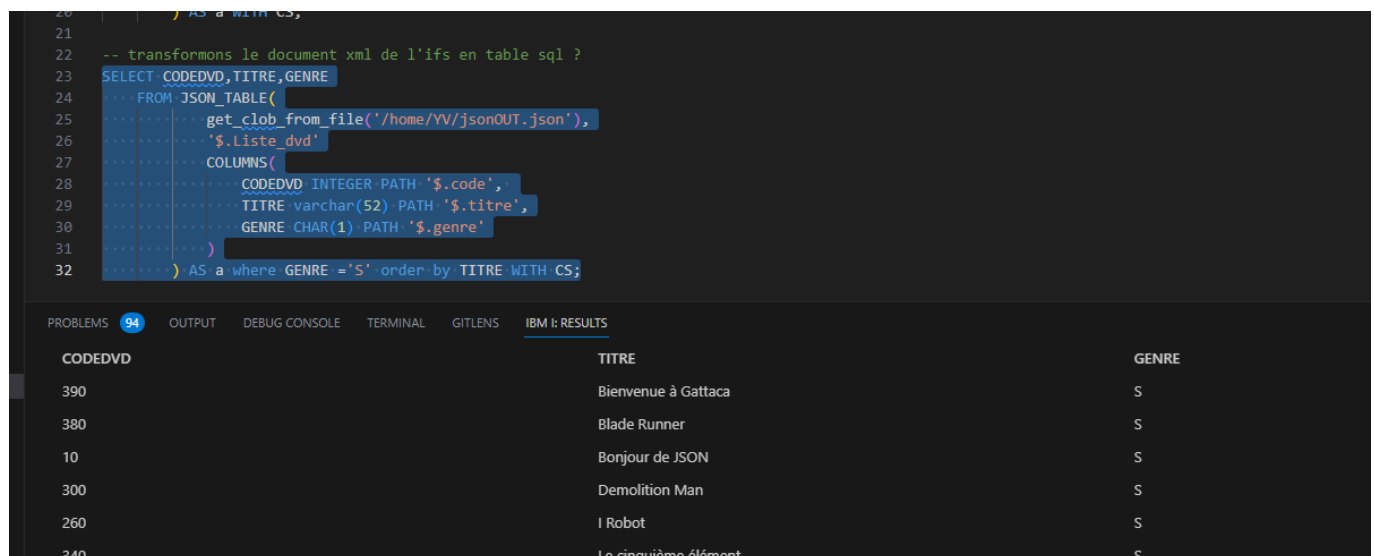
```

CODEDVD	TITRE	GENRE
10	Bonjour de JSON	S
20	Star Wars - L'attaque des clones	S
30	Le seigneur des anneaux - Les deux tours	F
40	Matrix	S
50	Spiderman	F

Noter que nous obtenons une véritable table sql et qu'à ce titre onus pouvons utiliser toute la puissance du sql pour filter,ordonner, modifier les données du fichier xml (enfin sa représentation) . taper la ligne)

```
-- transformons le document xml de l'ifs en table sql ?
SELECT CODEDVD,TITRE,GENRE
  FROM JSON_TABLE(
    get_clob_from_file('/home/YV/jsonOUT.json'),
    '$.Liste_dvd'
    COLUMNS(
      CODEDVD INTEGER PATH '$.code',
      TITRE varchar(52) PATH '$.titre',
      GENRE CHAR(1) PATH '$.genre'
    )
  ) AS a where GENRE = 'S' order by TITRE WITH CS;
```

vous obtenez



The screenshot shows an IDE with a dark theme. The top part displays the same SQL query as in the previous block. Below the editor, there is a panel titled 'IBM I: RESULTS' which contains a table with the query's output.

CODEDVD	TITRE	GENRE
390	Bienvenue à Gattaca	S
380	Blade Runner	S
10	Bonjour de JSON	S
300	Demolition Man	S
260	I Robot	S
340	Le cinquième élément	S

Intégrer la lecture du documentXML dans un programme RPG.

1. copier le source du programme dspJSON.sqlrpgle dans dspXML.sqlrpgle via ctrl-c puis ctrl-v du contenu (c'est plus simple..)
2. changer la requête du curseur.

```
exec sql
  declare listeFilms cursor for
    SELECT TITRE
  FROM XMLTABLE('$result/liste_Films/dvd'
    PASSING XMLPARSE(
      DOCUMENT
      GET_XML_FILE('/home/YV/xmlOUT.xml')
    ) as "result"
    COLUMNS
      TITRE varchar(52) PATH 'titre'
```

```
) AS TABLEXML order by titre
for fetch only;
```

devient

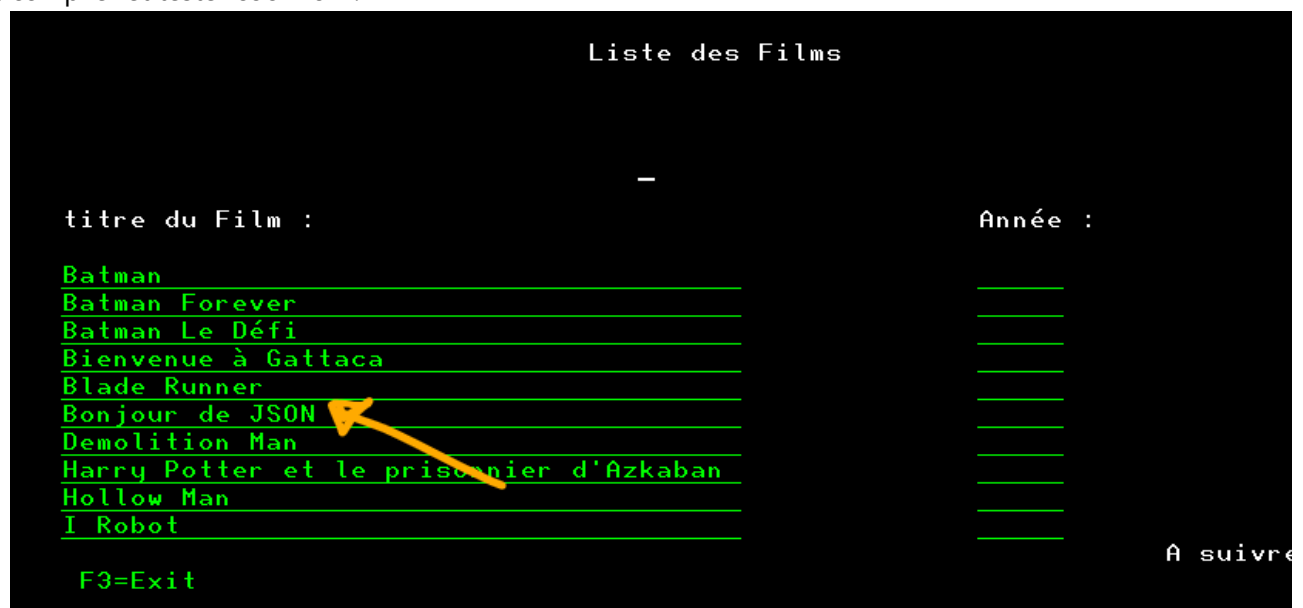
```
exec sql
declare listeFilms cursor for
SELECT TITRE
FROM JSON_TABLE(
    get_clob_from_file('/home/YV/jsonOUT.json'),
    '$.Liste_dvd'
    COLUMNS(
        CODEDVD INTEGER PATH '$.code',
        TITRE varchar(52) PATH '$.titre',
        GENRE CHAR(1) PATH '$.genre'
    )
) AS a order by titre
for fetch only;
```

penser à modifier le code de la boucle

```
DoU 1 = 0;
exec sql
fetch from listeFilms into :titre;
if sqlcode < *zeros or sqlcode = 100;
    leave;
endif;
```

Nous avons enlevé l'année. 🤪 j'ai oublié de la mettre dans le document xml mais le principe est là.

1. compiler et tester cool non ?



Conclusion et feed-back

Correction

💡💡💡💡 Idées

-