
✅ COMPTE RENDU DU PROGRAMME COBOL DB2PART1

🎯 Objectif du programme

Le programme **DB2PART1** a un rôle simple mais crucial :

👉 **Lire un fichier séquentiel OUTPRODS** contenant des produits déjà nettoyés
👉 **Convertir les champs texte (prix, stock)** en données numériques
👉 **Insérer chaque produit dans la table DB2** `API7.PRODUCTS`
👉 **Gérer toutes les erreurs SQL** (doublons, warnings, anomalies, échecs) et journaliser si besoin

Il s'agit d'un **programme d'alimentation de base DB2**, typique d'un traitement batch.

🗂️ Structure générale

1. ENVIRONMENT DIVISION

Déclare le fichier d'entrée :

Fichier	Type	Rôle
OUTPRODS	Séquentiel	Contient les produits à insérer dans DB2

---

2. DATA DIVISION

● FILE SECTION

Structure du fichier OUTPRODS :

Champ	Taille	Description
CODE-OP	X(3)	Code produit
DESC-OP	X(30)	Description
PU-OP	X(6)	Prix en texte
DE-OP	X(3)	Devise (non utilisée ici)
ST-OP	X(3)	Stock en texte

• WORKING-STORAGE

Inclut :

- **SQLCA** (zone de communication SQL)
- Variables du DCLGEN `PRODUCTS`
- Variables de conversion (prix, stock)
- Variables de LOG (WS-MSG, WS-USER, WS-CONTEXT…)
- Flag de fin de fichier
- File status du fichier d'entrée

🔄 3. PROCEDURE DIVISION

➤ OUVERTURE-FICHIERS

- Ouvre OUTPRODS en lecture
- Contrôle du file status
- Affichage d'erreur si ouverture KO

➤ LECTURE-OP (boucle principale)

Pour chaque enregistrement lu :

1. **Si fin de fichier** → FIN-PROD = "O"
2. **Sinon** :
    - Conversion des champs texte → numérique (`CONVERSIONS-VAR`)
    - Alimentation des variables hôtes DB2 (`TRANSFERT`)
    - Insertion dans DB2 (`AJOUT-OP`)

La boucle continue jusqu'à FIN-PROD = "O".

⚙️ 4. Modules principaux

🧮 CONVERSIONS-VAR

But : convertir les données du fichier (texte) en valeurs numériques utilisables par DB2.

🔸 **Prix**

- Copie PU-OP → WS-PU-STR
- Remplacement "." → "," (à cause du DECIMAL-POINT IS COMMA)
- Conversion via `FUNCTION NUMVAL`
- Stockage dans WS-PU-NUM

🔸 **Stock**

- Copie ST-OP → WS-STOCK-STR
- Conversion via `NUMVAL` → WS-STOCK-NUM

🔄 TRANSFERT

Transfert vers les variables du DCLGEN :

- CODE-OP → PRO-P-NO
- DESC-OP → PRO-DESCRIPTION + PRO-DESCRIPTION-LEN
- WS-PU-NUM → PRO-PRICE
- WS-STOCK-NUM → PRO-STOCK

Ces variables sont ensuite utilisées dans l'insertion SQL.

🏛️ AJOUT-OP (opération SQL principale)

🔹 Insertion :

```
INSERT INTO API7.PRODUCTS (P_NO, DESCRIPTION, PRICE, STOCK)
VALUES (:PRO-P-NO, :PRO-DESCRIPTION, :PRO-PRICE, :PRO-STOCK)
```

🔹 Gestion des SQLCODE

L'EVALUATE gère tous les cas :

✔️ **SQLCODE = 0**

→ Produit inséré correctement
→ DISPLAY "AJOUTE"

❗ **SQLCODE = -803 (doublon)**

→ Le produit existe déjà
→ Construction d'un message détaillé dans WS-MSG
→ Appel du sous-programme `LOGDATA` pour journalisation

⚠️ **SQLCODE > 0 (warnings)**

→ Log d'un message d'avertissement
→ Info enregistrée dans le log via LOGDATA

💥 **SQLCODE < 0 (erreur grave, sauf -803)**

→ ROLLBACK
→ Appel TEST-SQLCODE
→ Journalisation

🧯 **WHEN OTHER**

→ Sécurité : tous les cas imprévus sont traités comme anomalies SQL

📄 5. TEST-SQLCODE

Petit module dédié :

- Si SQLCODE = 0 → CONTINUE
- Si > 0 → "WARNING"
- Si erreur négative → "ANOMALIE" + appel ABEND-PROG

💣 6. ABEND-PROG

Forcer un ABEND (division par zéro) pour arrêter le traitement en cas d'erreur grave.

📝 7. LOG

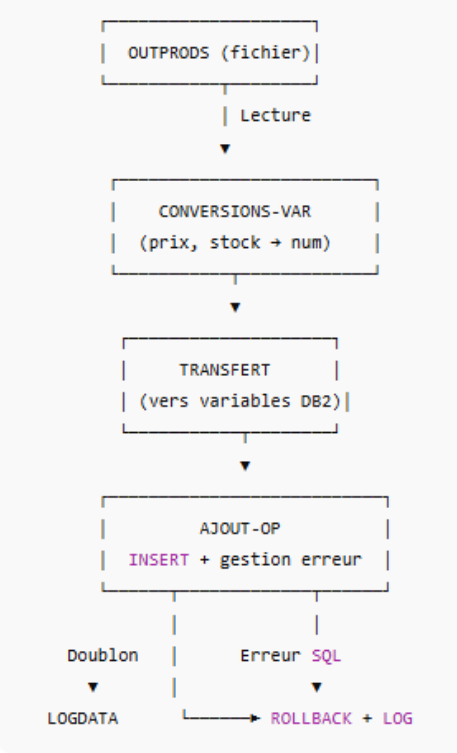
Gère la journalisation :

Récupère l'utilisateur courant via :

SELECT USER INTO WS-USER FROM SYSIBM.SYSDUMMY1

1.
2. Appelle le sous-programme **LOGDATA**, qui enregistre :
    - Le message
    - Le nom du programme
    - L'utilisateur

📊 SCHÉMA GÉNÉRAL DU TRAITEMENT



🏁 Résumé oral possible en 20 secondes

« Le programme DB2PART1 lit un fichier produit OUTPRODS, convertit le prix et le stock en numérique, prépare les variables hôtes DB2 et insère chaque produit dans la table API7.PRODUCTS. Il gère finement tous les SQLCODE : insertion correcte, doublons, warnings ou erreurs graves avec rollback. En cas d’anomalie ou de doublon, un message structuré est envoyé au sous-programme de journalisation LOGDATA. »
