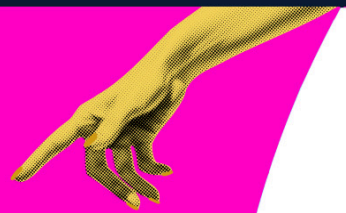
¡Believe
Présent et Futur de l'IBM i

Retour d'Expérience



IBMi
Le meilleur
est "Avenir"



SINAPSE

Db2 for i : un héritage puissant qui ne demande qu'à évoluer

Andy MALO · Ingénieur de données · 15 octobre 2026

IBM DB2 **Db2 for i**

avec itheis

Qui suis-je ?

Andy MALO

Ingénieur de données · Sinapse

- iBelieve 2021 : **500+ tables DDS → SQL**
- **Fan assumé de l'IBM i**, depuis mes années DBA
- **Le passeur** : l'existant IBM i ↔ la data moderne

avant 2016

Bureautique



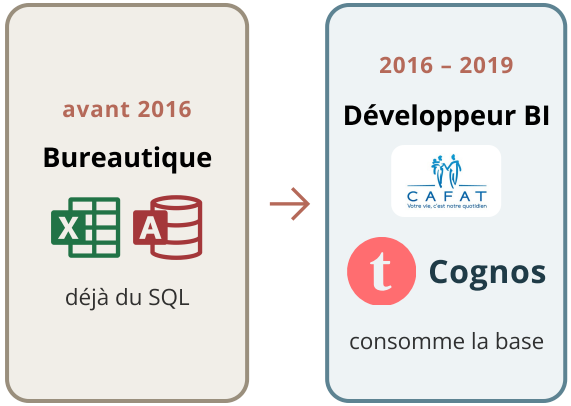
déjà du SQL

Qui suis-je ?

Andy MALO

Ingénieur de données · Sinapse

- iBelieve 2021 : **500+ tables DDS → SQL**
- **Fan assumé de l'IBM i**, depuis mes années DBA
- **Le passeur** : l'existant IBM i ↔ la data moderne

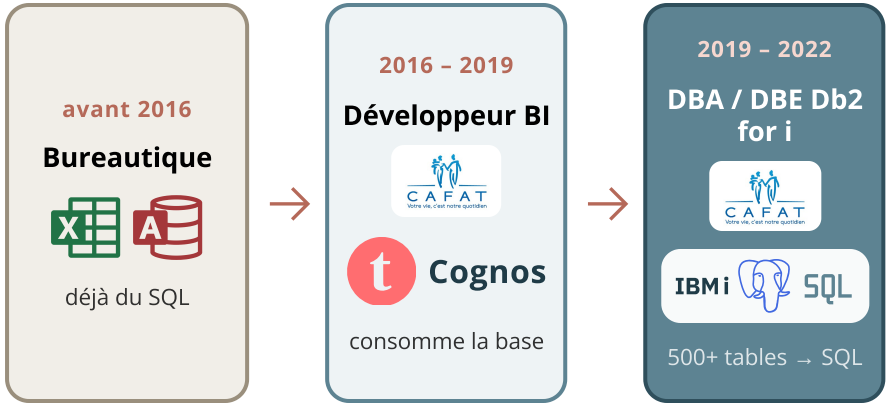


Qui suis-je ?

Andy MALO

Ingénieur de données · Sinapse

- iBelieve 2021 : **500+ tables DDS → SQL**
- **Fan assumé de l'IBM i**, depuis mes années DBA
- **Le passeur** : l'existant IBM i ↔ la data moderne

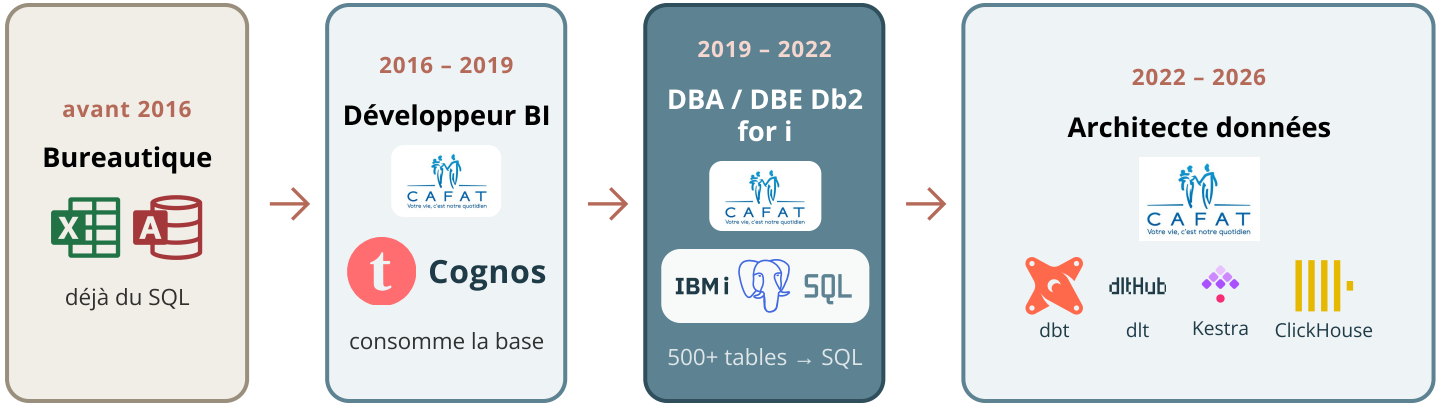


Qui suis-je ?

Andy MALO

Ingénieur de données · Sinapse

- iBelieve 2021 : **500+ tables DDS → SQL**
- **Fan assumé de l'IBM i**, depuis mes années DBA
- **Le passeur** : l'existant IBM i ↔ la data moderne

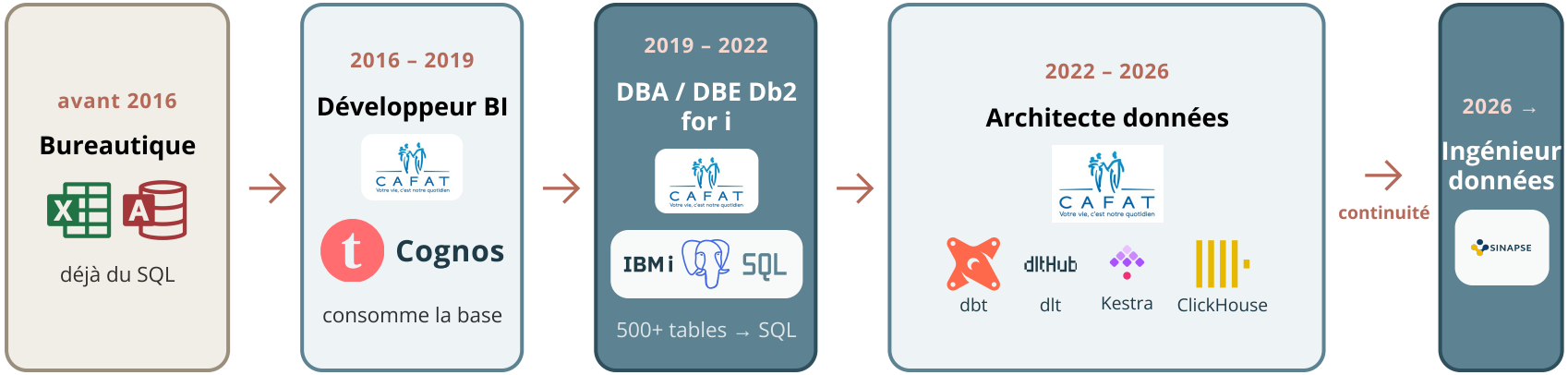


Qui suis-je ?

Andy MALO

Ingénieur de données · Sinapse

- iBelieve 2021 : **500+ tables DDS → SQL**
- **Fan assumé de l'IBM i**, depuis mes années DBA
- **Le passeur** : l'existant IBM i ↔ la data moderne



De la CAFAT à Sinapse : une suite logique

GIE · 2024
CAFAT · Gouvernement · FSH

265 000
Calédoniens concernés

Étape par étape
le recouvrement d'abord

Membres fondateurs

CAFAT — Votre vie, c'est notre quotidien
Gouvernement de la Nouvelle-Calédonie
fonds social de l'habitat — Bien plus qu'un toit

→ SINAPSE (étape par étape) → **SI social modernisé**

Ce que j'ai appris à la CAFAT, je le mets au service du **SI de demain**, avec l'équipe Sinapse.

sinapse.nc — Qui sommes-nous ? · sinapse.nc — Projet · sinapse.nc

Faire évoluer son IBM i : par où commencer ?

40 ans d'applicatif, des règles dans les programmes, peu de documentation



Avec un outil

jusqu'où aident-ils ?



Sans outil

comment avancer quand même ?

IBM i

Avec le natif seul

que permet le SQL ?

Le fil de la réponse : le SQL, **bien plus que des transactions**



moderniser



administrer



exploiter



ouvrir les données

1. Le chantier 2019-2022

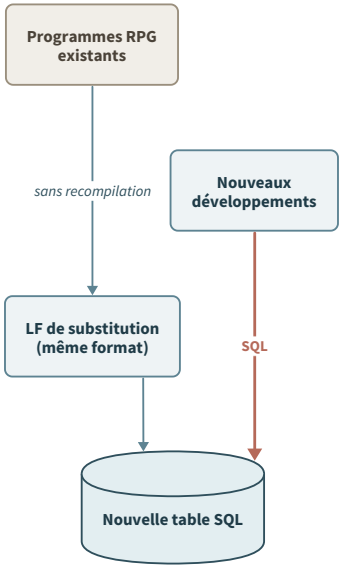
500+

tables DDS → SQL, **sans recompiler**

itheis GAIA resolution XCASE

ET APRÈS

contraintes **temporel** **RCAC**



❤ Merci à **E. Sadiman**, qui m'a confié ce chantier — et m'a rendu fan de l'IBM i

> iBelieve 2021 : **migrer des tables**. iBelieve 2026 : **reprendre la maîtrise**, en cartographiant.

Source : Itheis — méthode surrogate

2. Moderniser progressivement

On ne réécrit pas 40 ans d'applicatif d'un coup

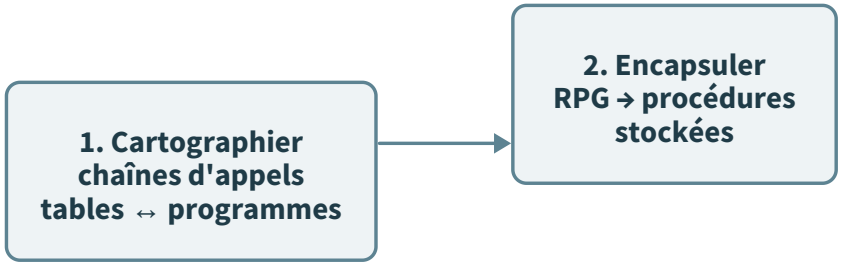
1. Cartographier chaînes d'appels tables ↔ programmes

Le SQL est l'outil de chacune de ces étapes.

Principes : SG24-8185 · Redbook 2026 MD260020 · REDP-5095

2. Moderniser progressivement

On ne réécrit pas 40 ans d'applicatif d'un coup

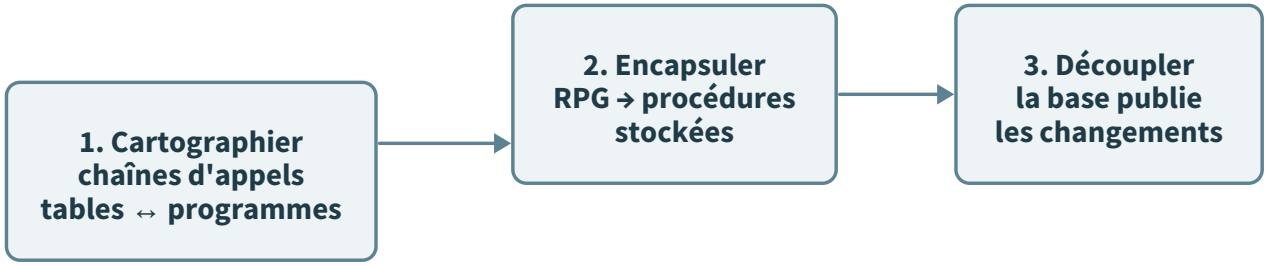


Le SQL est l'outil de chacune de ces étapes.

Principes : SG24-8185 · Redbook 2026 MD260020 · REDP-5095

2. Moderniser progressivement

On ne réécrit pas 40 ans d'applicatif d'un coup

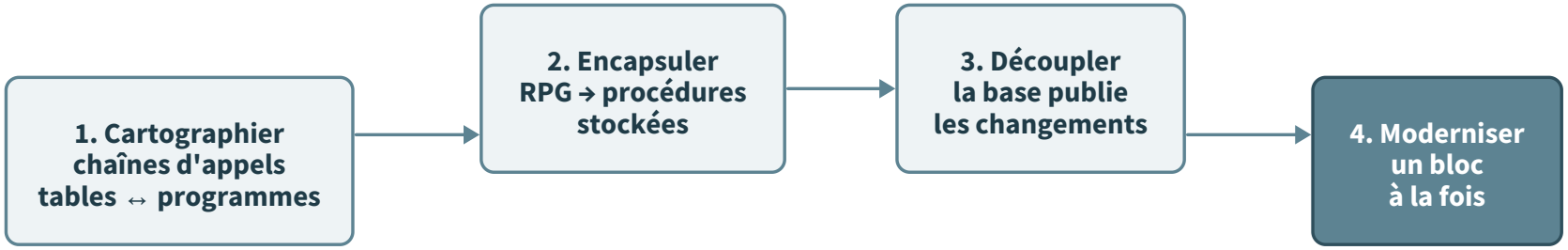


Le SQL est l'outil de chacune de ces étapes.

Principes : SG24-8185 · Redbook 2026 MD260020 · REDP-5095

2. Moderniser progressivement

On ne réécrit pas 40 ans d'applicatif d'un coup

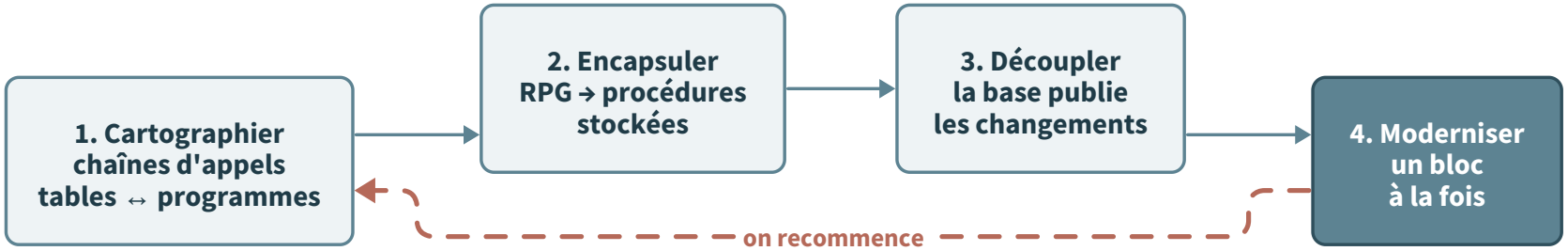


Le SQL est l'outil de chacune de ces étapes.

Principes : SG24-8185 · Redbook 2026 MD260020 · REDP-5095

2. Moderniser progressivement

On ne réécrit pas 40 ans d'applicatif d'un coup

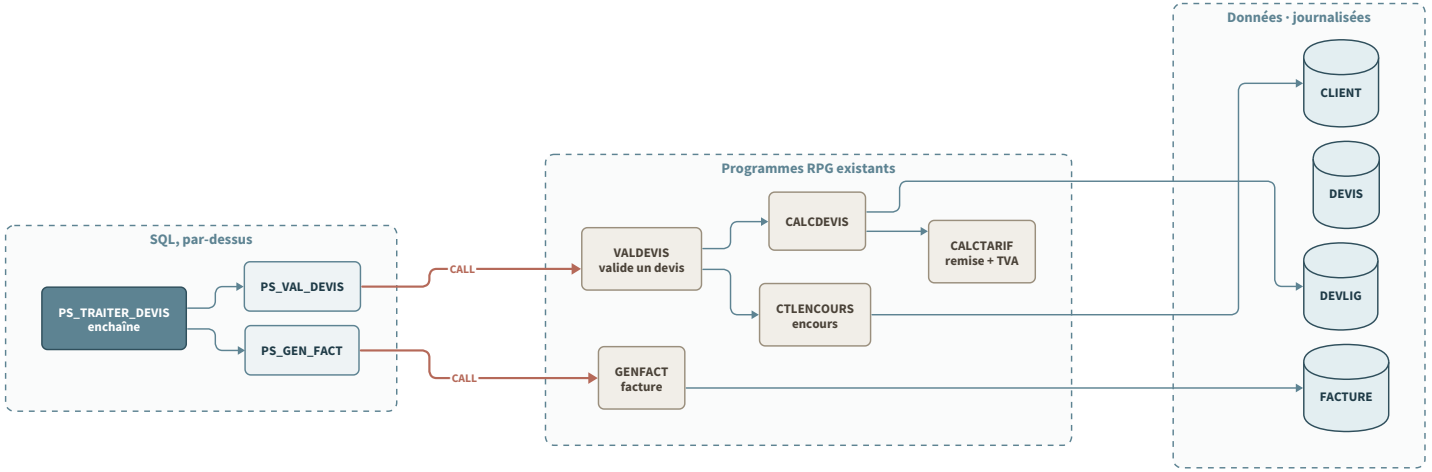


Le SQL est l'outil de chacune de ces étapes.

Principes : SG24-8185 · Redbook 2026 MD260020 · REDP-5095

Le terrain : une mini-application de devis

✓ Tout le SQL qui suit a **tourné** sur pub400 — et se rejoue avec le **kit** du dépôt



Le kit · GitHub

Kit : `kit/rpg/` — 5 tables, 7 programmes RPG compilés depuis ACS, 3 procédures SQL · sans compilateur : `kit/00-terrain.sql`

2.1 Cartographier

En clair : savoir qui appelle quoi avant de toucher à quoi que ce soit.



Avec un outil

X-Analysis : modèle déduit du code



Avec l'IA

Bob ou un autre assistant lit le RPG

IBM i

Sans outil

DSPPGMREF, piloté en SQL

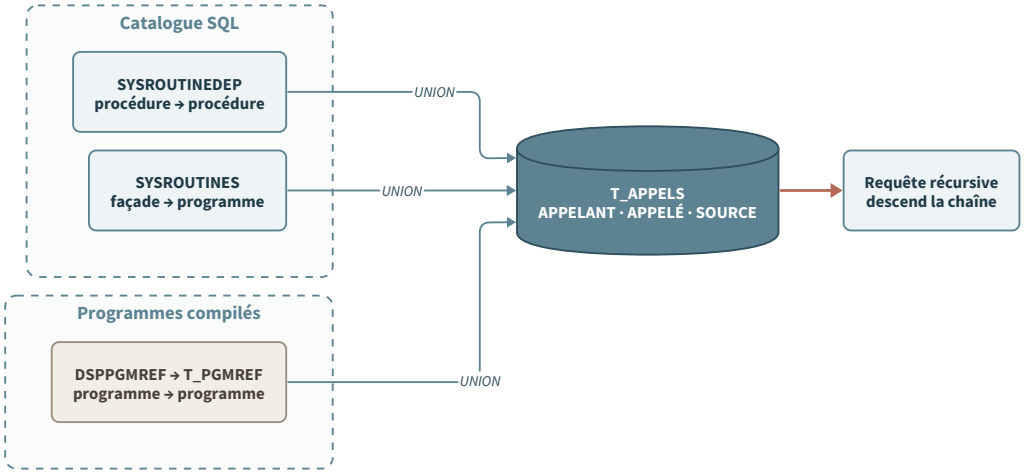
```
CALL QSYS2.QCMDEXC(
  'DSPPGMREF PGM(' CONCAT CURRENT SCHEMA CONCAT '/*ALL) OUTPUT(*OUTFILE) OBJTYPE(*PGM) '
  CONCAT 'OUTFILE(' CONCAT CURRENT SCHEMA CONCAT '/T_PGMREF) OUTMBR(*FIRST *REPLACE)');
```

8 programmes analysés → **3** appels entre eux + **9** références de tables

X-Analysis · DSPPGMREF · Format QADSPPGM · IBM Bob pour IBM i · IT Jungle — Bob 1.0 · `kit/rpg/15-cartographier-rpg.sql`

La chaîne d'appels, du SQL jusqu'au RPG

Trois sources, une table : T_APPELS



Chaque monde se lit à un endroit différent ; on matérialise, puis on interroge.

SYSROUTINEDEP · SYSROUTINES · Bob Cozzi — DSPPGMREF en UDTF · `kit/rpg/15-cartographier-rpg.sql`

La chaîne d'appels, du SQL jusqu'au RPG

Une requête récursive descend la chaîne

```
WITH CHAINE (APPELE, NIVEAU, CHEMIN, SOURCE) AS (
  SELECT APPELE, 1, CAST(APPELANT CONCAT ' > ' CONCAT APPELE AS VARCHAR(1000)), SOURCE
    FROM T_APPELS WHERE APPELANT = 'PS_TRAITER_DEVIS'
  UNION ALL
  SELECT A.APPELE, C.NIVEAU + 1, C.CHEMIN CONCAT ' > ' CONCAT A.APPELE, A.SOURCE
    FROM CHAINE C JOIN T_APPELS A ON A.APPELANT = C.APPELE
   WHERE C.NIVEAU < 15                                  -- profondeur max
     AND LOCATE(' ' CONCAT A.APPELE CONCAT ' ',
                ' ' CONCAT C.CHEMIN CONCAT ' ') = 0      -- pas de boucle
)
SELECT NIVEAU, CHEMIN, SOURCE FROM CHAINE ORDER BY CHEMIN;
```

> Le natif ne remplace pas un outil. Il permet de savoir **par où commencer** et **quoi ne pas casser**.

SYSROUTINEDEP · SYSROUTINES · Bob Cozzi — DSPPGMREF en UDTF · `kit/rpg/15-cartographier-rpg.sql`

La chaîne d'appels, du SQL jusqu'au RPG

Résultat sur pub400

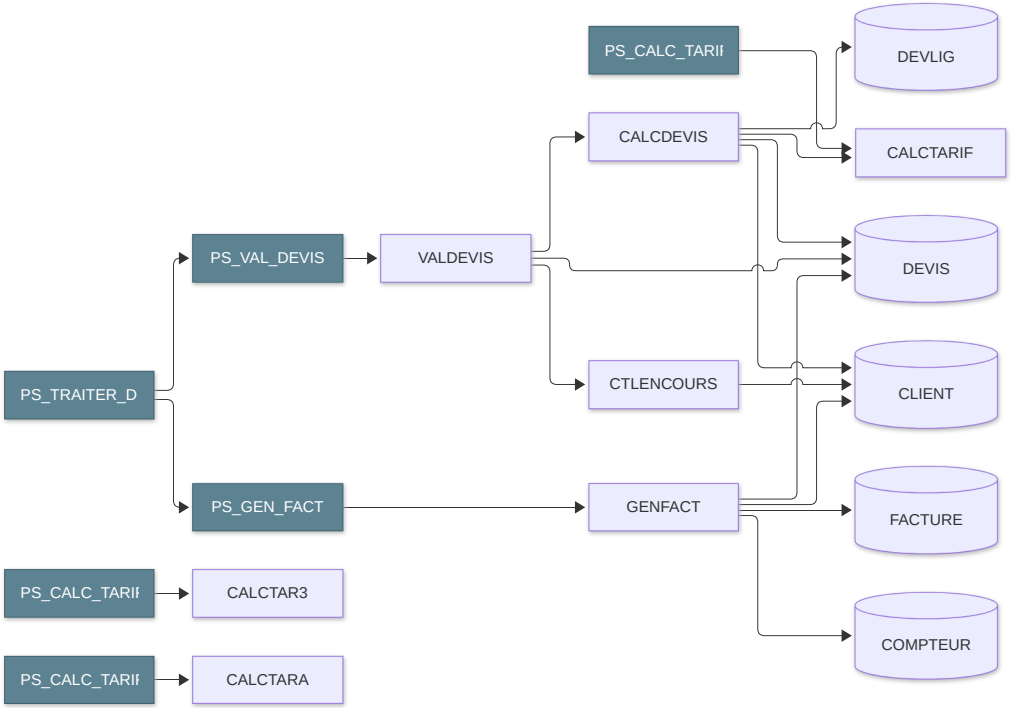
NIVEAU	CHEMIN	SOURCE
1	PS_TRAITER_DEVIS > PS_GEN_FACT	SYSROUTINEDEP
2	PS_TRAITER_DEVIS > PS_GEN_FACT > GENFACT	SYSROUTINES
1	PS_TRAITER_DEVIS > PS_VAL_DEVIS	SYSROUTINEDEP
2	PS_TRAITER_DEVIS > PS_VAL_DEVIS > VALDEVIS	SYSROUTINES
3	PS_TRAITER_DEVIS > PS_VAL_DEVIS > VALDEVIS > CALCDEVIS	DSPPGMREF
4	PS_TRAITER_DEVIS > PS_VAL_DEVIS > VALDEVIS > CALCDEVIS > CALCTARIF	DSPPGMREF
3	PS_TRAITER_DEVIS > PS_VAL_DEVIS > VALDEVIS > CTLENCOURS	DSPPGMREF

7 ligne(s)

Le natif ne remplace pas un outil. Il permet de savoir **par où commencer** et **quoi ne pas casser**.

SYSROUTINEDEP · SYSROUTINES · Bob Cozzi — DSPPGMREF en UDTF · `kit/rpg/15-cartographier-rpg.sql`

La chaîne d'appels, du SQL jusqu'au RPG



Le même graphe, écrit en Mermaid par une requête SQL (kit).

> Le natif ne remplace pas un outil. Il permet de savoir **par où commencer** et **quoi ne pas casser**.

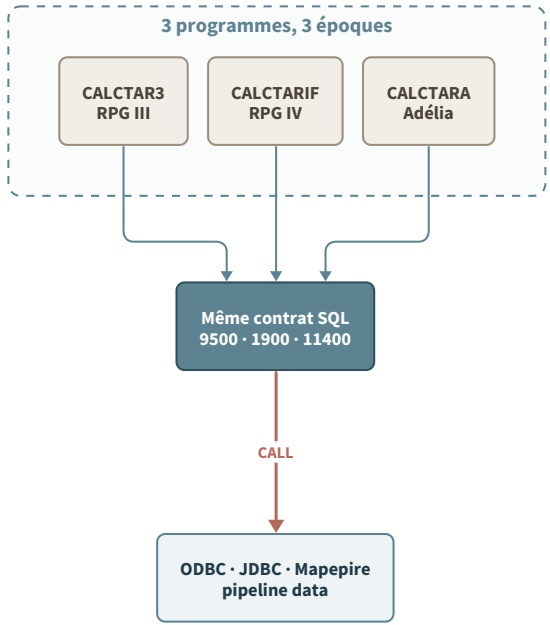
SYSROUTINEDEP · SYSROUTINES · Bob Cozzi — DSPPGMREF en UDTF · `kit/rpg/15-cartographier-rpg.sql`

2.2 Encapsuler

Un programme existant devient une **procédure stockée**. Sans toucher à son code.

```
CREATE OR REPLACE PROCEDURE PS_CALC_TARIF_RPG3 (
    IN  p_montant_ht   DECIMAL(9,2),
    IN  p_taux_remise  DECIMAL(5,2),
    OUT p_remise       DECIMAL(11,2),
    OUT p_tva          DECIMAL(11,2),
    OUT p_ttc          DECIMAL(11,2),
    OUT p_rc           CHAR(1)
)
  LANGUAGE RPG                      -- RPG, pas RPGLE : c'est de l'OPM
  SPECIFIC PS_TARIF3
  NO SQL
  EXTERNAL NAME 'VOTREBIB/CALCTAR3'
  PARAMETER STYLE GENERAL;

CALL PS_CALC_TARIF_RPG3(10000.00, 5.00, ?, ?, ?, ?);
```

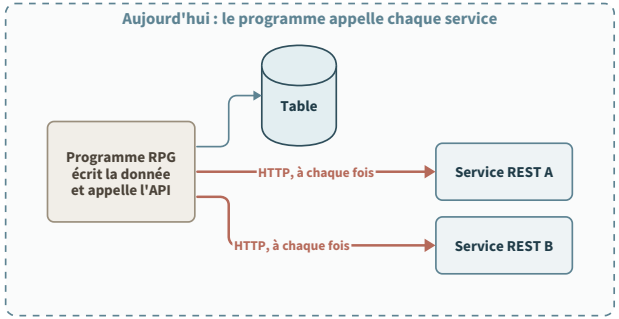


SG24-6393 — External stored procedures · `kit/rpg/13-facades.sql`

Db2 for i : un héritage puissant qui ne demande qu'à évoluer

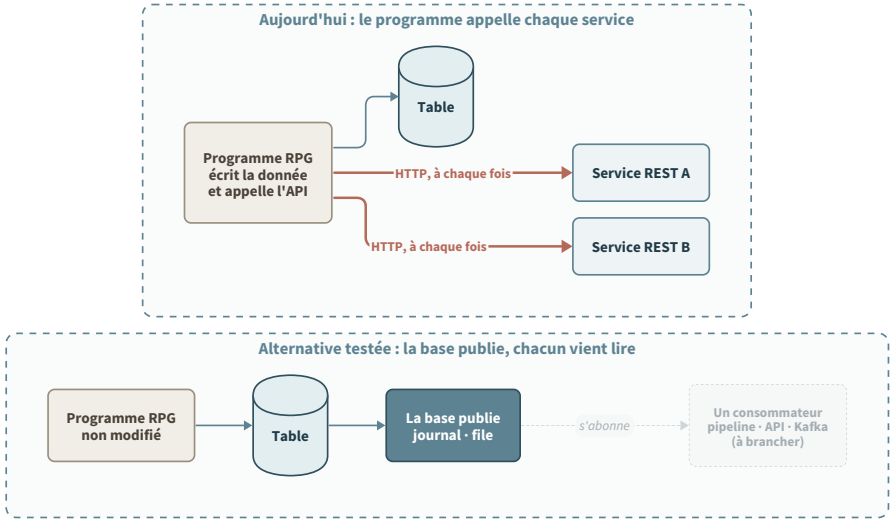
2.3 Découpler : publier sans toucher aux programmes

En clair : au lieu que le programme appelle chaque service, la base publie et chacun vient lire.



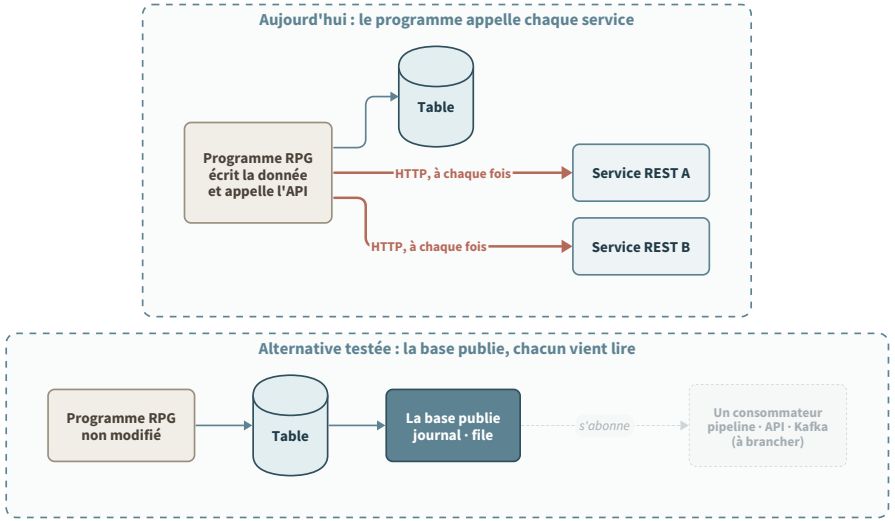
2.3 Découpler : publier sans toucher aux programmes

En clair : au lieu que le programme appelle chaque service, la base publie et chacun vient lire.



2.3 Découpler : publier sans toucher aux programmes

En clair : au lieu que le programme appelle chaque service, la base publie et chacun vient lire.



Le programme n'a pas changé, et **la base publie** ce qui change.

Testé sur pub400 : journal et file. Aucun consommateur n'est déployé.

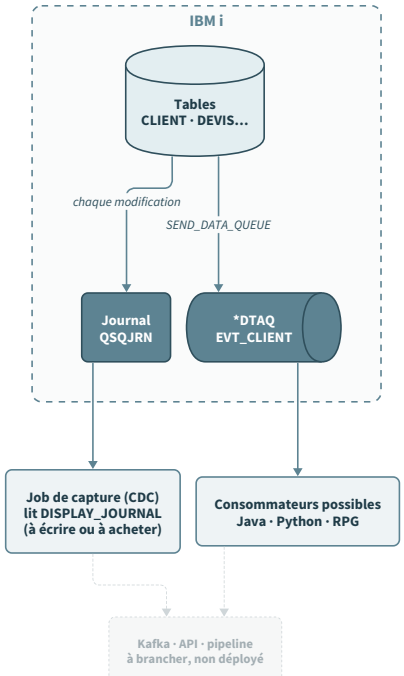
2.3 Le journal et la file : la base publie

Le journal a déjà tout vu

```
-- Le journal a déjà tout capté
SELECT ENTRY_TIMESTAMP, JOURNAL_ENTRY_TYPE,
       SUBSTR(OBJECT, 1, 10) AS TABLE_, PROGRAM_NAME
  FROM TABLE (QSYS2.DISPLAY_JOURNAL(
         JOURNAL_LIBRARY    => CURRENT SCHEMA,
         JOURNAL_NAME       => 'QSQJRN',
         STARTING_TIMESTAMP => CURRENT TIMESTAMP - 1 HOUR))
 WHERE JOURNAL_CODE = 'R';
```

```
2026-09-29 09:45:54.211616 | UP | DEVIS    | CALCDEVIS
2026-09-29 09:45:54.2132   | UP | DEVIS    | VALDEVIS
2026-09-29 09:45:54.224304 | UP | COMPTEUR | GENFACT
2026-09-29 09:45:54.23416  | PT | FACTURE  | GENFACT
2026-09-29 09:45:54.235536 | UP | CLIENT   | GENFACT
2026-09-29 09:45:54.236672 | UP | DEVIS    | GENFACT
```

Le journal sait **quel programme** a modifié quoi. Aucun n'a été modifié.



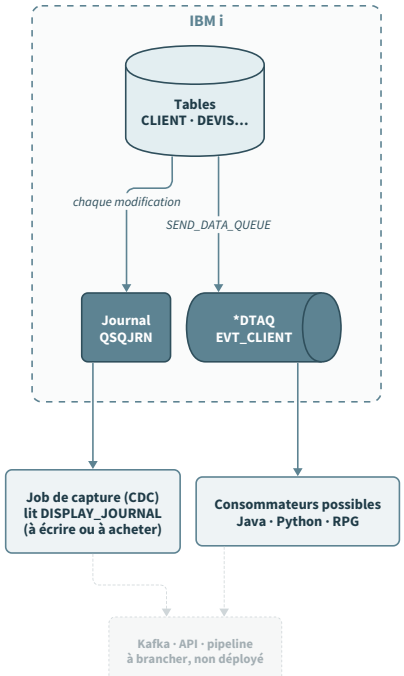
DISPLAY_JOURNAL · SEND_DATA_QUEUE · RECEIVE_DATA_QUEUE · JTOpen DataQueue · QRCVDTAQ · `kit/rpg/17-decoupler.sql`

2.3 Le journal et la file : la base publie

La base publie un JSON dans une file

```
-- Publier l'état du client en JSON
CALL QSYS2.SEND_DATA_QUEUE_UTF8(
  MESSAGE_DATA => (
    SELECT JSON_OBJECT('table'   VALUE 'CLIENT',
                       'id'      VALUE CLI_NUM,
                       'encours' VALUE CLI_ENCOURS,
                       'ts'      VALUE CURRENT TIMESTAMP)
      FROM CLIENT WHERE CLI_NUM = 1001),
  DATA_QUEUE         => 'EVT_CLIENT',
  DATA_QUEUE_LIBRARY => CURRENT SCHEMA);
```

Ici l'envoi est lancé à la main. Sur un vrai système, un trigger le fait à chaque modification.



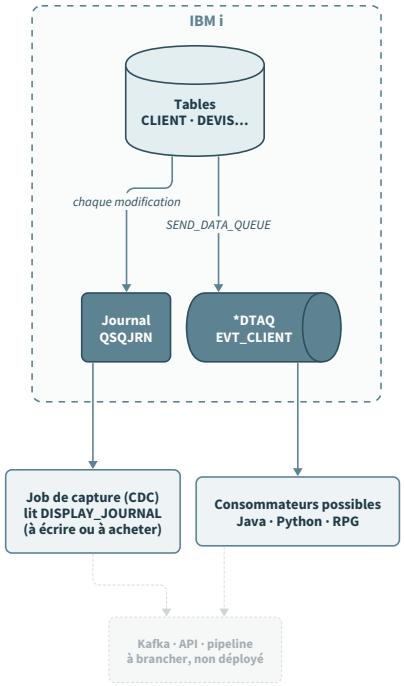
DISPLAY_JOURNAL · SEND_DATA_QUEUE · RECEIVE_DATA_QUEUE · JTOpen DataQueue · QRCVDTAQ · `kit/rpg/17-decoupler.sql`

2.3 Le journal et la file : la base publie

Le consommateur lit, y compris par ODBC

```
-- Côté consommateur, y compris via ODBC
SELECT CAST(MESSAGE_DATA_UTF8 AS VARCHAR(1000)) AS MESSAGE
  FROM TABLE (QSYS2.RECEIVE_DATA_QUEUE(
         DATA_QUEUE         => 'EVT_CLIENT',
         DATA_QUEUE_LIBRARY => CURRENT SCHEMA));
```

```
{"table":"CLIENT","id":1001,"encours":2910.00,
 "ts":"2026-09-29-09.45.57.398581"}
```

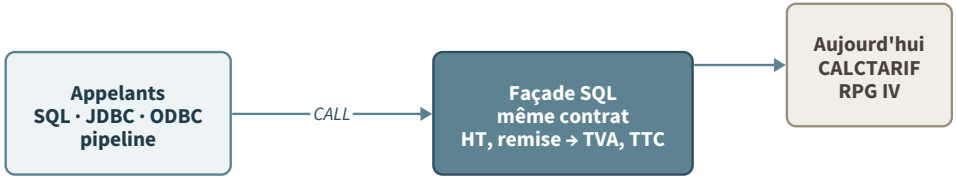


DISPLAY_JOURNAL · SEND_DATA_QUEUE · RECEIVE_DATA_QUEUE · JTOpen DataQueue · QRCVDTAQ · `kit/rpg/17-decoupler.sql`

Db2 for i : un héritage puissant qui ne demande qu'à évoluer

2.4 Moderniser, un bloc à la fois

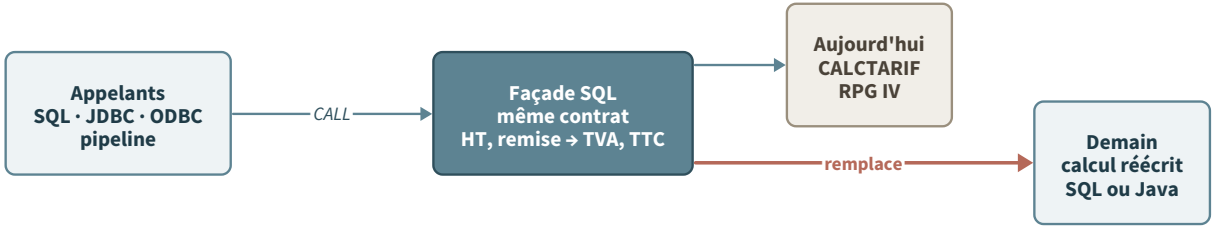
En clair : on remplace ce qu'il y a derrière la façade, les appelants ne voient rien.



Principe, non exécuté : le kit montre trois implémentations derrière un même contrat (2.2), pas encore une réécriture.

2.4 Moderniser, un bloc à la fois

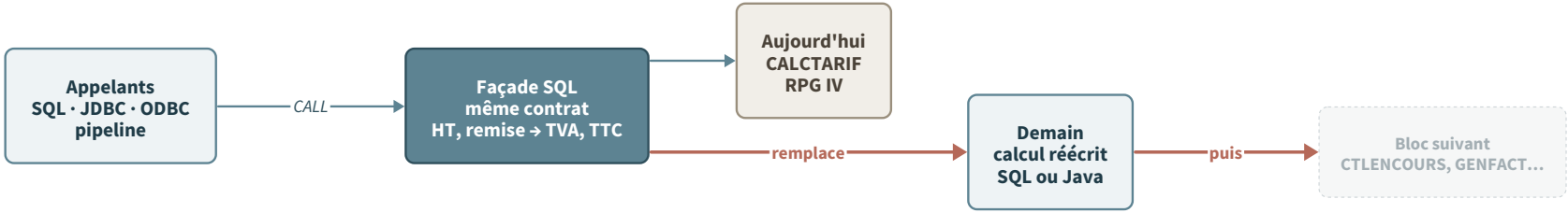
En clair : on remplace ce qu'il y a derrière la façade, les appelants ne voient rien.



Principe, non exécuté : le kit montre trois implémentations derrière un même contrat (2.2), pas encore une réécriture.

2.4 Moderniser, un bloc à la fois

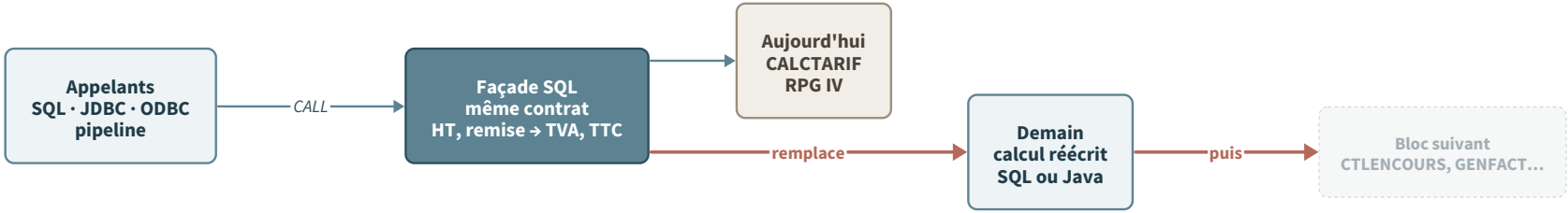
En clair : on remplace ce qu'il y a derrière la façade, les appelants ne voient rien.



Principe, non exécuté : le kit montre trois implémentations derrière un même contrat (2.2), pas encore une réécriture.

2.4 Moderniser, un bloc à la fois

En clair : on remplace ce qu'il y a derrière la façade, les appelants ne voient rien.



Le contrat ne change pas, **l'implémentation si.**

Principe, non exécuté : le kit montre trois implémentations derrière un même contrat (2.2), pas encore une réécriture.

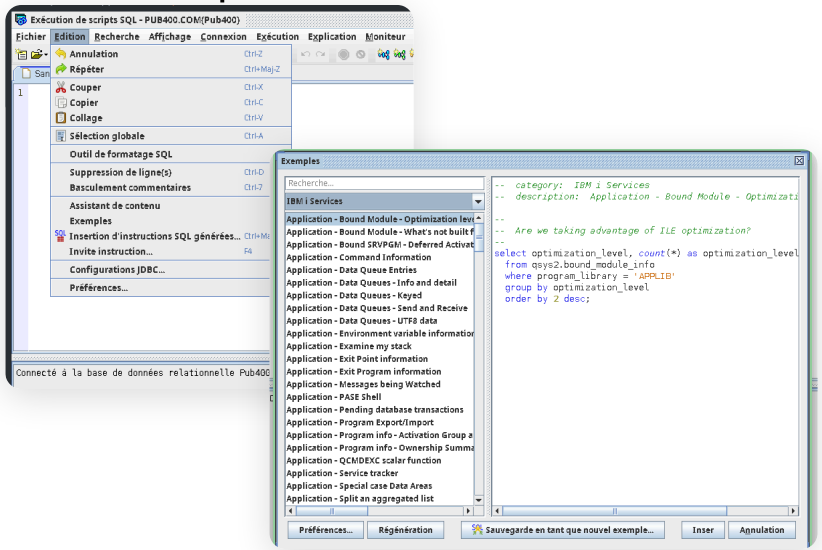
3. Administrer et exploiter en SQL

En clair : le système se consulte comme une table, sans connaître les commandes.

Une commande CL, un service SQL

Commande CL	Service SQL
`WRKACTJOB`	`QSYS2.ACTIVE_JOB_INFO`
`DSPLOG`	`QSYS2.HISTORY_LOG_INFO`
`DSPJRN`	`QSYS2.DISPLAY_JOURNAL`
`WRKOBJOWN`	`QSYS2.OBJECT_OWNERSHIP`
`WRKSYSVAL`	`QSYS2.SYSTEM_VALUE_INFO`

Les exemples sont livrés



IBM i SQL Services · Db2 for i Technology Updates · Volubis (FR)

Sécurité · exploitation · audit

Sécurité : les valeurs système qui comptent

```
SELECT SYSTEM_VALUE_NAME, CURRENT_CHARACTER_VALUE
  FROM QSYS2.SYSTEM_VALUE_INFO
 WHERE SYSTEM_VALUE_NAME IN ('QSECURITY', 'QMAXSIGN', 'QPWDEXPITV');
```

SYSTEM_VALUE_NAME	CURRENT_CHARACTER_VALUE
QMAXSIGN	000005
QSECURITY	50
QPWDEXPITV	*NOMAX

3 ligne(s)

SYSTEM_VALUE_INFO · HISTORY_LOG_INFO · OBJECT_PRIVILEGES · Syslog · Graylog · ELK · ClickStack

Sécurité · exploitation · audit

⚠ Exploitation : messages graves de la dernière heure

```
SELECT MESSAGE_TIMESTAMP, MESSAGE_ID, SEVERITY
  FROM TABLE (QSYS2.HISTORY_LOG_INFO(
         START_TIME => CURRENT TIMESTAMP - 1 HOUR)) X
 WHERE SEVERITY >= 40;
```

MESSAGE_TIMESTAMP	MESSAGE_ID	SEVERITY
2026-09-29 07:29:06.25702	CPF590A	40
2026-09-29 07:29:06.706073	CPF590A	40
2026-09-29 07:29:06.713749	CPF590A	40
2026-09-29 07:29:11.696488	CPF590A	40
2026-09-29 07:29:12.504385	CPI2417	40

… 361 ligne(s)

SYSTEM_VALUE_INFO · HISTORY_LOG_INFO · OBJECT_PRIVILEGES · Syslog · Graylog · ELK · ClickStack

Sécurité · exploitation · audit

Audit : qui a quels droits sur CLIENT ?

```
SELECT AUTHORIZATION_NAME, OBJECT_AUTHORITY
  FROM QSYS2.OBJECT_PRIVILEGES
 WHERE SYSTEM_OBJECT_SCHEMA = CURRENT SCHEMA
   AND SYSTEM_OBJECT_NAME   = 'CLIENT';
```

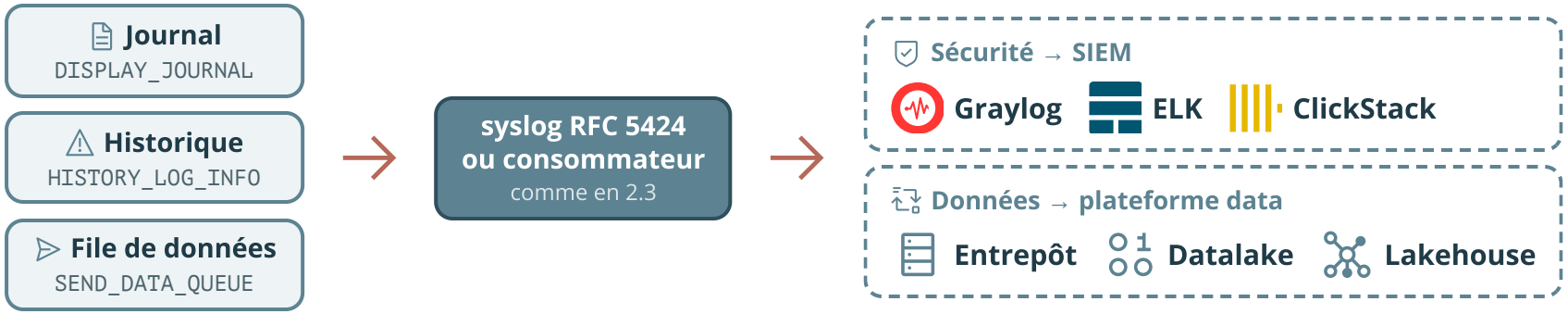
AUTHORIZATION_NAME	OBJECT_AUTHORITY
*PUBLIC	*EXCLUDE
‹propriétaire›	*ALL

2 ligne(s)

SYSTEM_VALUE_INFO · HISTORY_LOG_INFO · OBJECT_PRIVILEGES · Syslog · Graylog · ELK · ClickStack

Sécurité · exploitation · audit

Tout peut partir vers un SIEM ou une plateforme data, par le chemin de 2.3



Un même chemin : la sécurité rejoint **la supervision**, les données rejoignent **la plateforme data**.

Principe, non exécuté : ni SIEM ni entrepôt branché sur pub400.

SYSTEM_VALUE_INFO · HISTORY_LOG_INFO · OBJECT_PRIVILEGES · Syslog · Graylog · ELK · ClickStack

4. L'IBM i, source de données de premier plan

4.1 Se connecter

En clair : les données de l'IBM i alimentent les outils modernes, sans recopie à la main.



ODBC
Python · BI · DuckDB



JDBC
Java · Talend



Mapepire
un seul port chiffré



HTTP
l'IBM i appelle des API



MCP
assistants IA

Les procédures de l'étape 2.2 passent **par les mêmes portes**

ODBC ACS · ODBC Linux · JTOpen · Mapepire · MCP

4.2 ODBC → DuckDB, avec ou sans Arrow

✓ Exécuté sur pub400 le 30/09, **en connexion directe** : le pilote IBM i Access ODBC et un DSN.

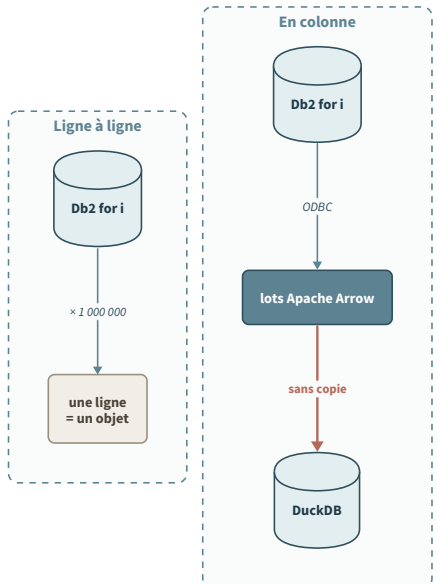
Le plus simple : DuckDB interroge l'IBM i

```
-- Une connexion, puis une requête exécutée par l'IBM i
SET VARIABLE ibmi = odbc_connect('DSN=MONIBMI');
CREATE TABLE client AS
  FROM odbc_query(getvariable('ibmi'),
                  'SELECT * FROM VOTREBIB.CLIENT');
```

Dans les deux cas, dans DuckDB :

clients	encours_total
3	48000.00

1 ligne(s)



DuckDB — extension ODBC · arrow-odbc-py · arrow-odbc (Rust) · `kit/exemples/duckdb-odbc.sql` · `arrow_odbc_duckdb.py`

Db2 for i : un héritage puissant qui ne demande qu'à évoluer

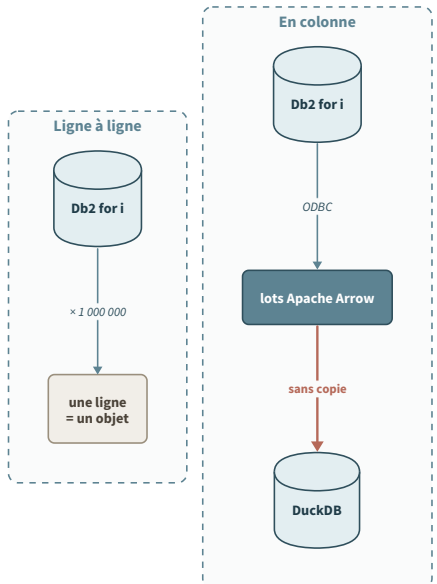
4.2 ODBC → DuckDB, avec ou sans Arrow

✓ Exécuté sur pub400 le 30/09, **en connexion directe** : le pilote IBM i Access ODBC et un DSN.

Pour le volume : des lots Arrow, en colonne

```
reader = read_arrow_batches_from_odbc(
    query="SELECT * FROM VOTREBIB.CLIENT",
    connection_string="DSN=MONIBMI",
    batch_size=50_000,
)
clients = pa.Table.from_batches(
    reader, schema=reader.schema)

duckdb.sql(
  "CREATE TABLE client AS SELECT * FROM clients")
```



Dans les deux cas, dans DuckDB :

clients	encours_total
3	48000.00

1 ligne(s)

DuckDB — extension ODBC · arrow-odbc-py · arrow-odbc (Rust) · `kit/exemples/duckdb-odbc.sql` · `arrow_odbc_duckdb.py`

4.3 Industrialiser l'extraction

dlt + pyarrow

vers Parquet : 20 à 30× plus rapide

Orchestrer

planifier · rejouer · historiser

GARDE-FOUS

- **lecture seule**
- **TLS**
- **typage**
- **hors pics**

Paralléliser par RRN

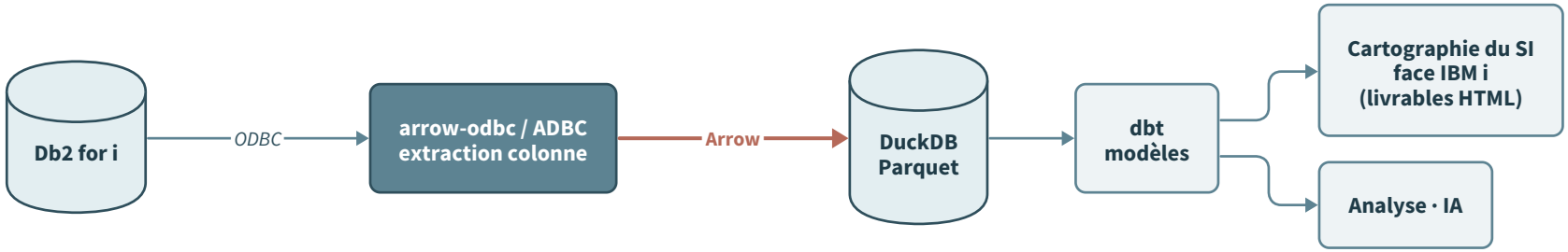
```
-- Taille connue du catalogue
SELECT NUMBER_ROWS
  FROM QSYS2.SYSPARTITIONSTAT
 WHERE TABLE_SCHEMA = CURRENT SCHEMA
   AND TABLE_NAME   = 'DEVLIG';

-- Une tranche parmi N
SELECT * FROM DEVLIG D
 WHERE RRN(D) BETWEEN 1 AND 2;
```

Sur le kit : **4** lignes au catalogue → 2 tranches de 2.

dlt sql_database · ODBC Linux + TLS · Query Supervisor

4.4 En vrai : Sinapse construit l'outil qui manque



Le besoin

préparer l'intégration du nouveau SI, anticiper les régressions

L'existant

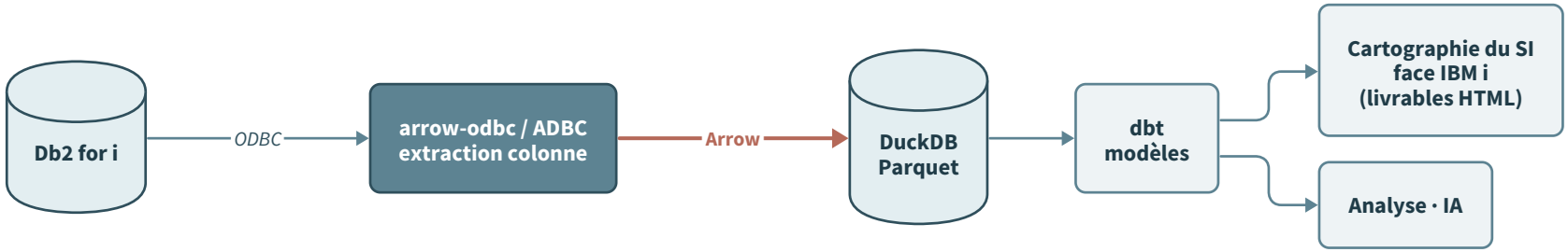
ODBC, fiable mais ligne à ligne

Construit

Arrow · ADBC · piste Flight SQL

Apache Arrow ADBC

4.4 En vrai : Sinapse construit l'outil qui manque



Le besoin

préparer l'intégration du nouveau SI, anticiper les régressions

L'existant

ODBC, fiable mais ligne à ligne

Construit

Arrow · ADBC · piste Flight SQL

Résultat

les chiffres, de vive voix

Apache Arrow ADBC

À retenir



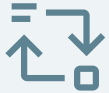
Pas à pas

cartographier · encapsuler · découpler



Le SQL intègre

administre · exploite · publie



Une vraie source

ODBC · Arrow · DuckDB



Un tremplin

l'existant construit la suite

LES TROIS QUESTIONS DU DÉBUT



Avec un outil : il accélère, l'IA aussi



Sans outil : des standards ouverts

IBM i

Avec le natif seul : le SQL va très loin

↗ Un moteur qui **évolue** : **17 nouveaux services** à l'été 2026

IT Jungle — Summer TRs 2026 · 7.6 TR2 · poweribmi.fr

L’héritage est là.

La seule question, c’est ce qu’on en fait.

Merci — questions ?

Le kit et le deck · GitHub

SG24-8185 · MD260020 · REDP-5095 · Gists Scott Forstie · Db2 for i Technology Updates
ODBC Linux · arrow-odbc-py · Apache Arrow ADBC · Mapepire · DuckDB ODBC