Steeve ASSOUS
pour

DB2 DANS ENVIRONNEMENT TRANSACTIONNEL

Introduction

- L'environnement transactionnel fait référence à un cadre où des applications COBOL interagissent avec une base de données DB2, en effectuant des transactions.
- Une transaction est un groupe d'opérations exécutées sur la base de données qui doit être traité comme une unité.
- Toutes les opérations dans une transaction doivent réussir ou échouer ensemble. Si une opération échoue, toutes les autres doivent être annulées.
- Cela garantit l'intégrité des données.
- Les transactions en DB2 doivent respecter les principes **ACID** pour garantir la fiabilité des transactions :
  - Atomicité
  - Cohérence
  - Isolation
  - Durabilité

Accès concurrents

- La notion d'**accès concurrents** fait référence à la situation où plusieurs programmes ou utilisateurs accèdent simultanément aux mêmes données dans une base de données DB2. Il est essentiel de gérer ces accès de manière efficace pour éviter les conflits, la corruption des données, ou les problèmes de performance.

- DB2 utilise des mécanismes de verrouillage pour garantir l'isolation des transactions et empêcher les conflits. Un verrou est posé sur les données lorsqu'une transaction les modifie, et ce verrou est maintenu jusqu'à ce que la transaction soit validée ou annulée.

  - Verrouillage de ligne ou de table :
    - DB2 peut verrouiller des lignes spécifiques ou des tables entières.

  - Problèmes de concurrence :
    - Les transactions simultanées peuvent provoquer des conflits, comme les lectures sales (une transaction lit des données non validées par une autre transaction).

Notion de Verrouillage

- Lorsqu'un programme accède à une base de données, DB2 utilise des **verrous** pour protéger l'intégrité des données. Il existe différents types de verrous :
  - **Verrou d'exclusivité** (Exclusive lock) : Interdit tout autre accès aux données verrouillées jusqu'à ce que le verrou soit relâché.
  - **Verrou partagé** (Shared lock) : Permet à plusieurs programmes de lire les mêmes données en même temps, mais interdit toute modification.
- Le verrouillage peut être appliqué à différents niveaux :
  - **Verrouillage au niveau de la table** : Le verrou couvre l'ensemble de la table.
  - **Verrouillage au niveau de la ligne** : Le verrou ne concerne qu'une seule ligne.
  - **Verrouillage au niveau de la page** : Une page (un groupe d'enregistrements physiques) est verrouillée.

Isolation : Modes d'accès concurrents

- L'**isolation** définit le degré de séparation entre les transactions concurrentes. En DB2, les différents niveaux d'isolation (UR, CS, RS, RR) permettent de contrôler l'accès aux données, équilibrant ainsi entre intégrité des données et performance.

  - CS (Cursor Stability)
    - Ce mode garantit qu'un verrou est appliqué uniquement sur la ligne en cours d'accès et relâché dès que le curseur passe à la ligne suivante. Ce mode minimise les conflits entre utilisateurs et est souvent utilisé en lecture.

  - RR (Repeatable Read)
    - Garantit que toutes les lignes lues par un curseur restent verrouillées jusqu'à la fin de la transaction, empêchant ainsi d'autres transactions de modifier les lignes lues.

  - UR (Uncommitted Read)
    - Permet la lecture de données qui ne sont pas encore validées par d'autres transactions. Cela peut améliorer la performance, mais présente le risque de lire des données non fiables (lectures "sales").

  - RS (Read Stability)
    - Verrouille uniquement les lignes lues, assurant que les lignes lues ne peuvent pas être modifiées par d'autres transactions pendant la durée de la transaction.

Gestion des transactions

- Une **transaction** en DB2 est une unité logique de travail composée de plusieurs opérations sur la base de données. Les transactions sont entourées par les instructions **COMMIT** et **ROLLBACK** en COBOL DB2 :

- **COMMIT** : Valide les modifications dans la base de données et relâche les verrous associés.

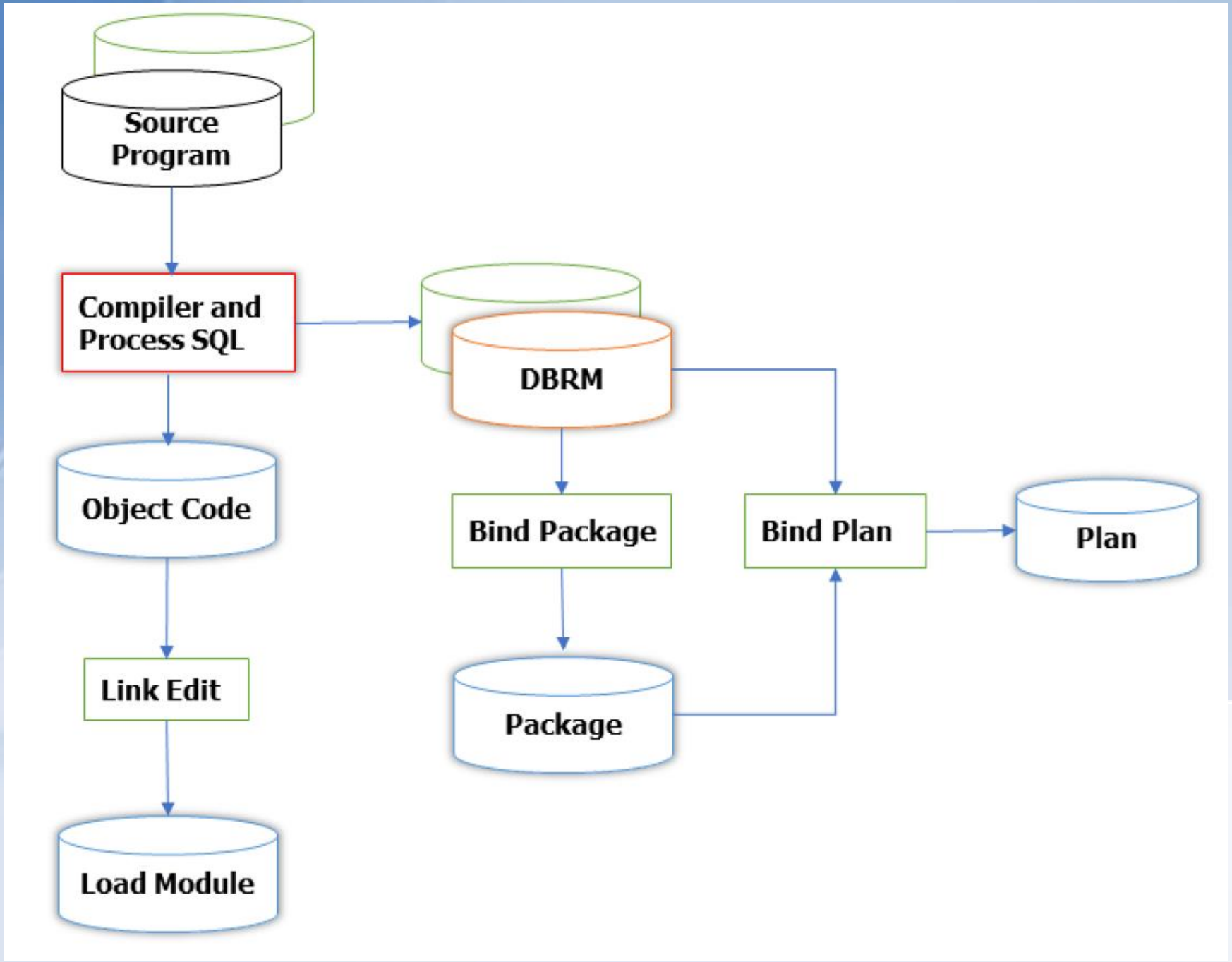
```
EXEC SQL
     COMMIT
END-EXEC
```

- **ROLLBACK** : Annule toutes les modifications faites depuis le dernier COMMIT et relâche les verrous.

```
EXEC SQL
     ROLLBACK
END-EXEC
```

- La gestion des accès concurrents en DB2 COBOL est cruciale pour assurer la cohérence et la performance des bases de données partagées.

- Elle repose sur des mécanismes de verrouillage, des niveaux d'isolation, et la gestion appropriée des transactions pour éviter les conflits tels que les deadlocks et les timeouts.


Source Program
Compiler and Process SQL
DBRM
Object Code
Bind Package
Bind Plan
Plan
Link Edit
Package
Load Module

Etapes

- La Pré-Compilation DB2 donne :
  - un programme source modifié, destiné à être compilé et linkédité, où les ordres SQL sont transformés en CALL à des routines d'accès.
  - Un module source DBRM des ordres SQL, destiné au BIND.

- La définition des modes d'accès concurrent en DB2 COBOL se fait soit au niveau du plan ou package lors du BIND, soit directement dans le code à travers des clauses SQL spécifiques.

- Ces modes (CS, RR, RS, UR) permettent de contrôler le degré de verrouillage et d'isolation des transactions, en fonction des besoins en termes de performance ou de cohérence des données.

- **Compilation des instructions SQL**
  - Lorsqu'un programme COBOL contenant des instructions SQL est compilé, il génère du **code source DBRM** (Database Request Module). Ce code contient les instructions SQL extraites du programme et est nécessaire pour la phase de BIND.

- **Optimisation**
  - Durant le BIND, DB2 analyse les instructions SQL pour optimiser leur exécution. Cela inclut le choix des meilleurs chemins d'accès aux données (plans d'exécution) en fonction de facteurs comme l'indexation des tables et la distribution des données.

- **Liens avec les objets DB2**
  - Le BIND lie les instructions SQL à des objets spécifiques de la base de données, tels que les tables, les vues, et les index.

BIND

- Le processus **BIND** fait référence à l'étape où les instructions SQL dans un programme COBOL (ou tout autre langage supportant DB2) sont "liées" à la base de données DB2.

- En d'autres termes, c'est une phase de préparation où DB2 analyse et optimise les requêtes SQL avant l'exécution réelle du programme.

- Il décide du chemin d'accès pour chaque ordre SQL. Cette décision tient compte de plusieurs informations contenues dans le catalogue : taille des tables, définition des vues, existence ou non d'index, degré de désorganisation de chaque index etc ….

- Rappelons que la décision d'utiliser ou non un index est du ressort exclusif du BIND.

- Enfin, bien entendu, construit le plan qui est un LOAD MODULE stocké dans le DIRECTORY de DB2. On peut voir un plan comme un ensemble de paramètres fournies aux routines d'accès résultants des ordres SQL.

PLAN

- Il s'agit d'un ensemble d'instructions d'exécution résultant du processus de BIND. C'est un objet DB2 qui contient les chemins d'accès optimisés (ou **plans d'exécution**) pour chaque instruction SQL présente dans un programme COBOL.

- Le **PLAN** est nécessaire à l'exécution du programme : Quand un programme COBOL est exécuté, il se réfère au PLAN pour savoir comment accéder aux données de manière optimale.

- Un PLAN peut être partagé par plusieurs programmes COBOL, mais il est spécifique à un ensemble d'instructions SQL compilées.

- Le PLAN est stocké dans le catalogue DB2 et est géré par le système DB2.

Exemple de définition

- Le **BIND** est l'opération où le programme est lié à DB2, et cela permet de créer un **PLAN**. Le **PLAN** contient des informations sur les chemins d'accès optimisés et les stratégies que DB2 utilisera pour exécuter les instructions SQL dans le programme.

```
BIND PLAN(plan_name)
MEMBER(dbrm_name)
ISOLATION(CS)
VALIDATE(BIND)
ACQUIRE(USE)
RELEASE(COMMIT)
```

- **BIND PLAN(plan_name) :** Cette partie indique à DB2 de créer un PLAN nommé plan_name.
- **MEMBER(dbrm_name) :** Spécifie le nom du fichier DBRM (généré lors de la compilation) qui contient les instructions SQL à lier.
- **ISOLATION(CS) :** Définit le niveau d'isolation des transactions, ici Cursor Stability (CS), ce qui signifie que les verrous sont relâchés après chaque ligne lue.
- **VALIDATE(BIND) :** Indique que la validation des objets de base de données se fait au moment du BIND.
- **ACQUIRE(USE) :** Indique que les ressources seront acquises au moment de leur utilisation, et non au démarrage de l'exécution.
- **RELEASE(COMMIT) :** Les verrous seront relâchés lorsque le COMMIT est effectué.

BIND : Options

- VALIDATE : différer un non le contrôles des autorisations.
  - VALIDATE (BIND) : exiger tout de suite toutes les autorisations nécessaires
  - VALIDATE (RUN) : le contrôle des autorisations se fera à l'exécution du programme.
- ACQUIRE : précise le moment d'acquisition des ressources et des verrouillages nécessaires
  - ACQUIRE (ALLOCATE) : toutes les ressources avant l'exécution
  - ACQUIRE (USE) : les ressources sont acquises, et les verrouillages effectués, au fur et à mesure des besoins.
- En particulier, si une table est référencée dans une partie du programme qui n'est pas exécutée suite à un test, cette table ne sera pas verrouillée.
- RELEASE : précise le moment de libération des ressources acquises
  - RELEASE (DEALLOCATE) : libération à la fin du programme
  - RELEASE (COMMIT) : libération de toutes les ressources au prochain point de synchronisation. IL est à noter que les demandes de LOCK explicites (LOCK TABLE ..) sont aussi désactivées.
- EXPLAIN : n'a pas d'argument YES or NO.
  - L'existence du mot EXPLAIN déclenche l'exécution de l'ordre EXPLAIN qui renseigne la table USER.PLAN_TABLE; Celle-ci peut être consultée pour connaître les chemins d'accès des ordres SQL.

COMPILATION

```
//JOBDB2    JOB NOTIFY=&SYSUID,CLASS=A,MSGCLASS=H,
//      TIME=(0,30),MSGLEVEL=(1,1)
//PROCLIB  JCLLIB ORDER=SDJ.FORM.PROCLIB
//*
//       SET SYSUID=APIX,
//          NOMPGM=EXO01
//*
//APPROC   EXEC COMPDB2
//STEPDB2.SYSLIB   DD DSN=&SYSUID..SOURCE.DCLGEN,DISP=SHR
//*            DD DSN=&SYSUID..SOURCE.COPY,DISP=SHR
//STEPDB2.SYSIN    DD DSN=&SYSUID..SOURCE.DB2(&NOMPGM),DISP=SHR
//STEPDB2.DBRMLIB  DD DSN=&SYSUID..SOURCE.DBRMLIB(&NOMPGM),DISP=SHR
//STEPLNK.SYSLMOD  DD DSN=&SYSUID..SOURCE.PGMLIB(&NOMPGM),DISP=SHR
//*TEPCOB.SYSLIB   DD DSN=&SYSUID..DB2FILES.COPY,DISP=SHR
```

JCL – BIND PLAN

```
//*********************************************
//*                ETAPE DE BIND
//*********************************************
//BIND     EXEC PGM=IKJEFT01,COND=(4,LT)
//DBRMLIB  DD  DSN=&SYSUID..SOURCE.DBRMLIB,
//         DISP=SHR
//SYSTSPRT DD  SYSOUT=*,OUTLIM=25000
//SYSTSIN  DD  *
  DSN SYSTEM (DSN1)
  BIND PLAN      (EXO01) -
       QUALIFIER (APIX)    -
       ACTION    (REPLACE) -
       MEMBER    (EXO01) -
       VALIDATE  (BIND)    -
       ISOLATION (CS)      -
       ACQUIRE   (USE)     -
       RELEASE   (COMMIT)  -
       EXPLAIN   (NO)
/*
```

```
//*********************************************
//*                EXECUTION
//*********************************************
//STEPRUN  EXEC PGM=IKJEFT01,COND=(4,LT)
//STEPLIB  DD
DSN=&SYSUID..SOURCE.PGMLIB,DISP=SHR
//SYSTSPRT DD  SYSOUT=*,OUTLIM=2500
//SYSOUT   DD  SYSOUT=*,OUTLIM=1000
//SYSTSIN  DD  *
  DSN SYSTEM (DSN1)
  RUN PROGRAM(EXO01) PLAN(EXO01)
/*
```