

# Réplication Oracle avec Streams

Débutant Avancé Expert



Complexité

par fadace ([Fabien Celaia](#))

Date de publication : 7.7.2007

Dernière mise à jour : 8.5.2008

Cet article à pour but de vous démontrer les fonctionnalités basiques de Streams, le système de réplication transactionnel d'Oracle, au travers d'une installation particulipre qu'est DownStreams.

I - Introduction.....	3
II - Logging supplémentaire.....	5
III - Configuration de l'instance source.....	6
IV - Configuration de l'instance cible.....	6
V - Heartbeat : la première table répliquée à mettre en place.....	8
VI - Contrôle de l'état de Streams.....	9
VI - Nettoyage de Streams.....	9
VIII - Conséquences d'un cluster.....	10

## I - Introduction

Par le passé, les systèmes de réplication qu'offraient Oracle n'ont jamais fait l'unanimité. Basés sur des snapshots de tables, peu performants, ils ont souvent été ignorés au profit d'autres systèmes de réplication hétérogènes d'autres éditeurs.

Dès la version 9i, Oracle a décidé de palier à cette carence en réécrivant totalement sa réplication, se basant sur l'expérience de transfuges d'autres éditeurs. C'est ainsi que Streams a vu le jour.

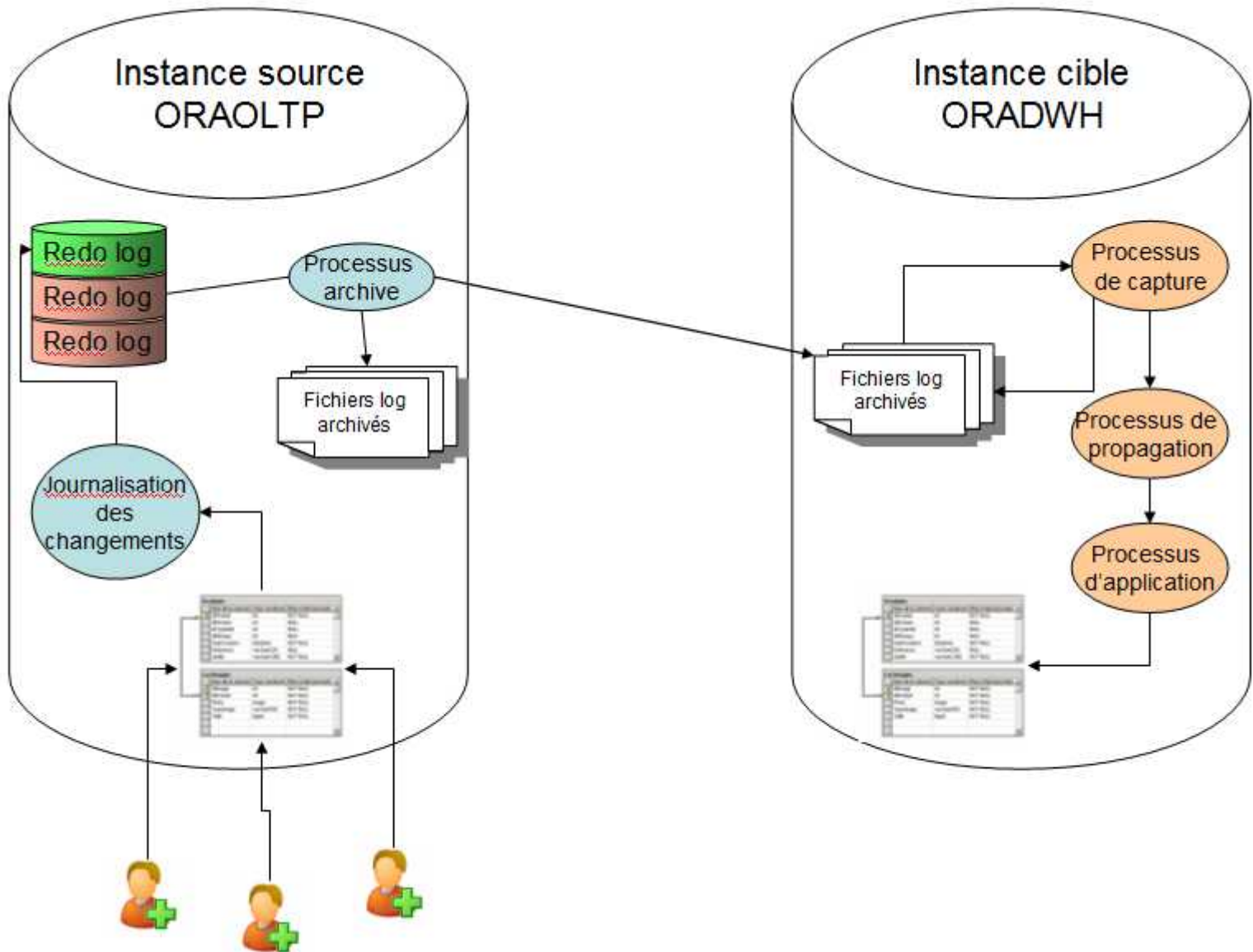
les débuts de Streams ont été douloureux, et il n'est d'ailleurs pas recommandé de l'utiliser en production avec une masse importante de données à répliquer en version pré-10.2. Dès la version 10.2, Streams commence par contre à devenir un outil de réplication transactionnel intéressant.

Cet article à pour but de vous démontrer les fonctionnalités basiques de Streams. Afin de ne pas sombrer dans la traduction stupide d'articles Oracle, je me propose de vous démontrer la mise en place d'une configuration Streams particulière : Downstreams (Archived-Log Downstreams Capture).

### Principaux traits de Downstreams


- Quasi aucun impact de performance sur la base source, la lecture se faisant via des fichier d'archive logs, et tous les processus de Streams étant localisés sur l'instance cible
- Quasi aucun impact sur la source si la cible vient à tomber en panne
- Réplication à flux semi tendu puisque les données n'arrivent qu'au rythme des archive logs (il faut donc attendre ou générer des switch logs)
- Architecture intéressante pour une réplication d'un environnement OLTP à l'ODS d'un environnement décisionnel
- Possibilité de modifier les règles de réplication pour, par exemple, transférer les données d'une table source dans un schéma et une table dont les noms sont différents sur la cible
- Possibilité (dès 10.2) de déterminer des règles négatives afin d'améliorer les performances du processus capture.

Nous partirons du postulat que nous allons répliquer des données d'un serveur source nommé ORAOLTP à un serveur cible nommé ORADSS.



En résumé, voici ce qui va se passer

- 1 Des utilisateurs modifient des enregistrements sur ORAOLTP
- 2 Les blocs modifiés passent au travers du redo log
- 3 Lorsque le redo log est plein ou qu'un switch redo log est activé, le redo suivant est activé.
- 4 Le redo traité est journalisé, soit donc copié comme archivelog par l'archiver
- 5 Downstreams oblige en plus l'archiver à copier une copie de ce fichier sur le site distant de ORADWH
- 6 Dès ce point s'arrête la charge sur ORAOLTP, et commence le travail de ORADSS
- 7 Le processus de capture d'ORADWH détecte l'arrivée d'un nouvel archivelog distant
- 8 Il le lit intégralement et en retire les modifications apportées sur les tables qui sont marquées comme répliquées
- 9 Il crée un enregistrement dans une file d'attente et l'envoie au propagateur
- 10 Le propagateur applique au besoin des transformations et l'envoie à l'apporteur (apply)
- 11 L'apporteur exécute la modification sur la base ORADWH Dans notre cas de figure (DownStreams), le propagateur n'est pas nécessaire puisque capture et applyer sont sur le même serveur de données. Nous redirigerons donc directement l'apporteur sur la file d'attente du capture

 Dans le cas de Downstreams, le propagateur n'a pas de raison d'être puisque capture et applyer sont sur le même serveur de données. Nous redirigerons donc l'entrée de l'applyer sur la file du capture.

## II - Logging supplémentaire

Streams nécessite des informations dans les redo supplémentaires. Pour obtenir lesdites informations, les tables sources doivent être en mode *supplemental logging*. Ce faisant, la taille des redo logs va augmenter, ainsi que celles des archivlogs. Cette modification impacte donc les performances globales de l'instance source, qu'il s'agisse de Streams ou de Downstreams.

Logminer est lui aussi dépendant de ces informations. Dès Oracle 10.2, la base de données doit avoir un niveau minimum de supplemental logging pour permettre l'application au niveau table.

Pour contrôler le niveau au niveau de la base

```

SELECT supplemental_log_data_min,
       supplemental_log_data_pk,
       supplemental_log_data_ui,
       supplemental_log_data_fk,
       supplemental_log_data_all
FROM v$database;
  
```

Le supplemental\_log\_data\_min doit être activé. Pour l'activer s'il ne l'est pas :

```
alter database add supplemental log data;
```

Au niveau des tables, le choix entre les divers modes de supplemental logging est possible (UNIQUE, PRIMARY KEY, FOREIGN KEY, ALL), mais a son incidence sur les performances et la façon plus ou moins stricte avec laquelle Streams gèrera certaines erreurs.

Pour ma part, je conseille l'utilisation de PRIMARY KEY.

```
alter table MonSchema.MaTable add supplemental log data (PRIMARY KEY) columns ;
```

Si vous obtenez une erreur lors du passage de l'ordre, il se peut que cela vienne d'une ancienne configuration. Il suffit alors de supprimer le groupe log spécifique à la table. La commande se génère aisément grâce à l'ordre suivant :

```

select 'alter table '||owner||'.'||table_name||' drop supplemental log group '||log_group_name||';'
from dba_log_groups
where owner = 'MonSchema'
and table_name = ?MaTable';
  
```

Il est ensuite aisé de déterminer quel type de supplemental logging est associé à chaque table grâce à la commande suivante :

```

select table_owner, table_name, supplemental_log_data_all, supplemental_log_data_FK,
       supplemental_log_data_ui, supplemental_log_data_pk
from dba_capture_prepared_tables
where supplemental_log_data_all &lt;&lt;&lt; 'NO'
OR supplemental_log_data_PK&lt;&lt;&lt; 'NO'
OR supplemental_log_data_FK&lt;&lt;&lt; 'NO'
  
```

```
OR supplemental_log_data_UI<&&lt;&&gt;'NO' ;
```

### III - Configuration de l'instance source

Afin de permettre le bon fonctionnement de Streams un certain nombre de paramètres de l'instance source doivent être configurés avec des valeurs spécifiques :

Paramètre	Valeur
log_archive_config	DG_CONFIG=(ORAOLTP,ORADWH)
log_archive_dest_2	SERVICE=ORADWH.DEVELOPPEZ.COM ARCH OPTIONAL NOREGISTER VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) TEMPLATE=/CheminSurServeurCible/ ORAOLTP_%t_%s_%r.arc DB_UNIQUE_NAME=ORADWH
log_archive_dest_state_2	enable

Création d'un super-utilisateur STREAMSADM


```
CONNECT SYS/*****@ORAOLTP AS SYSDBA
CREATE USER STREAMSADM IDENTIFIED BY ***** ACCOUNT UNLOCK ;
GRANT DBA TO STREAMSADM ;
GRANT RESOURCES TO STREAMSADM ;
```

### IV - Configuration de l'instance cible

Afin de permettre le bon fonctionnement de Streams un certain nombre de paramètres de l'instance cible doivent être configurés avec des valeurs spécifiques :

Paramètre	Valeur
log_archive_config	DG_CONFIG=(ORAOLTP,ORADWH)

Création d'un super-utilisateur STREAMSADM et de son schéma, en tant que SYS

 *En configuration Downstreams, les mots de passe de SYS doivent être obligatoirement similaires sur ORADWH que sur ORAOLTP. Par convenance, nous agissons de même en ce qui concerne l'utilisateur STREAMADM : cela simplifie certaines tâches administratives.*

```
CONNECT SYS/*****@ORADWH AS SYSDBA
CREATE TABLESPACE STREAMS_D01 DATAFILE '/VotreLocalisation/
STREAMSADM_D01.dat' SIZE=200M AUTOEXTEND ON NEXT 100M MAXSIZE 2G ;
CREATE USER STREAMSADM IDENTIFIED BY ***** DEFAULT TABLESPACE STREAMS_D01 ACCOUNT UNLOCK;
GRANT DBA TO STREAMSADM ;
GRANT RESOURCE TO STREAMSADM ;

GRANT EXECUTE ON DBMS_FLASHBACK TO STREAMSADM ;
GRANT EXECUTE ON dbms_streams_adm TO STREAMSADM ;
GRANT EXECUTE ON dbms_apply_adm TO STREAMSADM ;
GRANT EXECUTE ON dbms_streams_adm TO STREAMSADM ;

BEGIN
  DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE(
    grantee      =&&gt; 'STREAMSADM',
    grant_privileges =&&gt; true);
END;
/
```

Bien que l'environnement est en mode Downstreams, il est quand même nécessaire d'avoir un database link entre la base de données cible et la base de données source. Ce database link est à créer sur la base de données cible. Il

permet de faciliter l'opération d'enregistrement de nouvelles tables dans le flux de réplication Streams. Le database link appartient à l'administrateur Streams (STREAMSADM).

```
CONNECT STREAMSADM/*****@ORADWH
-- création
CREATE DATABASE LINK ORAOLTP CONNECT TO CURRENT_USER USING 'ORAOLTP.DEVELOPPEZ.COM';

-- Test
SELECT * FROM DUAL@ORAOLTP ;
```

Afin de permettre la configuration des processus de capture et d'application il est nécessaire de créer 2 queues Oracle de type Anydata sur la base de données cible. Cette création s'effectue au moyen de la procédure DBMS\_STREAMS\_ADM.SET\_UP\_QUEUE

```
PROMPT Création de la queue de capture
begin
  dbms_streams_adm.set_up_queue(
    queue_table =& ; 'streamsadm.stream_queue_cpt',
    queue_name  =& ; 'stream_queue_cpt',
    queue_user  =& ; 'streamsadm');
end;
/

-- Test
SELECT * FROM streamsadm.stream_queue_cpt ;
```

Selon la même méthode, création d'une queue d'application nommée stream\_queue\_appl et appartenant à l'administrateur Streams (STREAMSADM).

```
PROMPT Création de la queue d'application

begin
  dbms_streams_adm.set_up_queue(
    queue_table =& ; 'streamsadm.stream_queue_appl',
    queue_name  =& ; 'stream_queue_appl',
    queue_user  =& ; 'streamsadm');
end;
/

-- Test
SELECT * FROM streamsadm.stream_queue_appl ;
```

La dernière étape à réaliser avant de pouvoir ajouter une table dans le processus de réplication Streams et la création du processus de Capture sur la base cible. Cette création s'effectue au moyen du package DBMS\_CAPTURE\_ADM.CREATE\_CAPTURE.

```
BEGIN
  DBMS_CAPTURE_ADM.CREATE_CAPTURE(
    queue_name          =& ; 'streamsadm.stream_queue_cpt',
    capture_name       =& ; 'strm_capture',
    rule_set_name      =& ; NULL,
    start_scn          =& ; NULL,
    source_database     =& ; 'ORAOLTP',
    use_database_link  =& ; true,
    first_scn          =& ; NULL,
    logfile_assignment =& ; 'implicit');
END;
/

-- Test
```

```
SELECT CAPTURE_NAME, STATUS, STATUS_CHANGE_TIME FROM DBA_CAPTURE ;
```

## V - Heartbeat : la première table répliquée à mettre en place

Afin d'être certain que notre système de réplication fonctionne, il est d'usage de créer une table de test sur laquelle on opère un update chaque 5 minutes, via un cron ou un job Oracle. Cette table permet alors de calculer la latence de la réplication.

### Première table à répliquer

```
DECLARE
  cStreamUser CONSTANT VARCHAR2(30) := 'STREAMSADM';
  cStreamPropName CONSTANT VARCHAR2(30) := 'STRM_PROP';
  cStreamApplyName CONSTANT VARCHAR2(30) := 'STRM_APPLY';
  cStreamCaptureName CONSTANT VARCHAR2(30) := 'STRM_CAPTURE';
  cSourceQueueName CONSTANT VARCHAR2(30) := cStreamUser || '.' || 'STREAM_QUEUE_CPT';
  cDestinationQueueName CONSTANT VARCHAR2(30) := cStreamUser || '.' || 'STREAM_QUEUE_APPL';

  pOwner VARCHAR2(30) := 'STREAM';
  pTableName VARCHAR2(30) := 'STREAM_LASTCOMMIT';
  pSourceInstance VARCHAR2(30) := 'ORAOLTP.DEVELOPPEZ.COM';
  iScn NUMBER := 0;
  vSql varchar2(500);

BEGIN

DBMS_STREAMS_ADM.ADD_TABLE_RULES(
  table_name =& ; Upper(pOwner) || '.' || Upper(pTableName),
  streams_type =& ; 'capture',
  streams_name =& ; cStreamCaptureName,
  queue_name =& ; cSourceQueueName,
  include_dml =& ; true,
  include_ddl =& ; false,
  include_tagged_lcr =& ; false,
  source_database =& ; pSourceInstance,
  inclusion_rule =& ; true);

dbms_streams_adm.add_table_rules(
  table_name =& ; pOwner || '.' || pTableName,
  streams_type =& ; 'APPLY',
  streams_name =& ; cStreamApplyName,
  queue_name =& ; cSourceQueueName,
  include_dml =& ; true,
  include_ddl =& ; FALSE,
  include_tagged_lcr =& ; false,
  inclusion_rule =& ; true,
  source_database =& ; pSourceInstance);

vSql := 'SELECT DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER() into iSCN FROM DUAL@' || pSourceInstance ;
execute immediate (vSql) ;

vSql := 'truncate table ' || pOwner || '.' || pTableName ;
execute immediate (vSql) ;

vSql := 'insert into ' || pOwner || '.' || pTableName
        || ' select * from ' || pOwner || '.' || pTableName || '@' || pSourceInstance ;
execute immediate (vSql) ;

dbms_apply_adm.set_table_instantiation_scn (
  source_object_name =& ; Upper(pOwner) || '.' || Upper(pTableName),
  source_database_name =& ; pSourceInstance,
  instantiation_scn =& ; iSCN
) ;

END;
```

## Première table à répliquer

/

## VI - Contrôle de l'état de Streams

Outre la Grid Control qui permet une administration assez aisée de Streams, voici quelques requêtes intéressantes à encapsuler dans vos scripts de contrôle.

## Statuts des divers processus

```
SELECT 'CAPTURE', C.CAPTURE_NAME, C.STATUS, C.STATUS_CHANGE_TIME
FROM DBA_CAPTURE C
UNION ALL
SELECT 'APPLY', a.APPLY_NAME, a.STATUS, a.STATUS_CHANGE_TIME
from DBA_APPLY a
```

## Erreurs apparaissant lors de l'application

```
SELECT * from DBA_APPLY_ERROR ;
```

Les transactions bloquantes sont stockées comme des messages. Il n'est donc pas aisé, en cas de blocage, de détecter quelle transaction bloque et pourquoi. Oracle propose à cet effet la fonction *print\_transaction()* qui affiche, pour un no de transaction donné par la table *DBA\_APPLY\_ERROR*.

Je vous conseille donc vivement de suivre la  [documentation Oracle](#) pour créer cette fonction fort utile.

## Utilisation de la fonction print\_transaction()

```
SET SERVEROUTPUT ON SIZE 1000000
EXEC print_transaction('1.12.3455')
```

Pas de SQL dans ce que vous obtiendrez : on décortique assez aisément le message en lisant les nouvelles et anciennes valeurs des objets. Relevez rapidement le nombre de modifications par message et le type du traitement apparaissant dans l'entête.

## Monitoring des tables répliquées

```
select
  substr(rule_condition, instr(rule_condition, '')+1, instr(substr(rule_condition, instr(rule_condition, '')+1),
    SCHEMAS,
      substr(rule_condition, instr(rule_condition, 'get_object_name()')+21,
        instr(rule_condition, ') and :dml.is_null_tag()') - instr(rule_condition, 'get_object_name()')-22) TABLES,
    source_database
from sys.STREAMS$_RULES
where streams_name='STRM_CAPTURE';
```

## VI - Nettoyage de Streams

Il peut être nécessaire parfois de redémarrer complètement l'environnement Streams depuis zéro. Pour ce faire il est nécessaire de supprimer toutes les tables du flux de réplication Streams.

Selon la documentation Oracle le nettoyage de l'environnement Streams peut être fait avec l'exécution de la procédure *DBMS\_STREAMS\_ADM.REMOVE\_STREAMS\_CONFIGURATION*. Malheureusement ce n'est pas le

cas, en plus de ce package il est nécessaire de nettoyer les règles créées lors de l'ajout de table et de supprimer les queues d'Apply et de Capture.

```
BEGIN
DBMS_STREAMS_ADM.REMOVE_STREAMS_CONFIGURATION ;

FOR maqueue IN (select name from dba_queues where owner = 'STREAMSADM')
LOOP
    DBMS_STREAMS_ADM.REMOVE_QUEUE(maqueue.name, TRUE) ;
END LOOP;

FOR maregle IN (select rule_set_name from dba_rule_sets where rule_set_owner = 'STREAMSADM')
LOOP
    dbms_rule_adm.drop_rule_set(maregle.rule_set_name, TRUE) ;
END LOOP;
END ;
/
```

## VIII - Conséquences d'un cluster

Dans le cas d'un serveur source en cluster, soit donc avec plusieurs instances gérant une même base, la logique est la même, bien que chacune des instances gère ses propres archivelogs.

Les archivelogs sont gérées par chacune des instances et copiées de manière asynchrone sur la cible.

Le seul impact que le RAC génère est la latence de la réplication. Quel que soit l'ordre d'arrivée des archivelogs, c'est la séquence du SCN, global à la base et à toutes les instances sources qui déterminera l'ordre de lecture des archivelogs par le processus de capture.

Il convient donc, si les quelques instances sources n'ont pas le même rythme de génération des archivelogs (souvent le cas lorsqu'une instance est plus fortement sollicitée), de forcer via job Oracle un switch logfile à intervalle régulier sur les 2 instances.

