7장. Database Replication

AgensSQL은 Database의 고효율성과 안정성을 위하여 데이터베이스를 실시간으로 복제하여 대기용 데이터베이스 서버를 구성 할수 있습니다. Database Replication은 구성방식에 따라 전체 Database의 모든 변경 사항을 실시간로 복제하는 Physical Replication과 전체 Database가 아닌 지정된 Table들만을 실시간으로 복제서버에 복제하는 Logical Replication가 있습니다.

7.1 논리적(Logical) 데이터 복제 기능

Logical Replication은 운영서버에 Publication 구성과 복제서버에 Subscription을 구성하여 해당 테이블의 변경 내용을 실시간으로 복제할 수 있습니다. 복제서버가 운영서버만을 복제하며 Read-Only로 동작하는 Physical Replication과는 다르게 복제서버도 별도로 Read-Write운영이 가능합니다. 따라서 HA 및 백업의 용도로 이용하기 보단 데이터 실시간 공유에 활용이 적합합니다. Subscription이 동작하는 Table에 복제서버에서의 변경이 가해진 경우 Error 와 데이터 불일치가 발생 할 수 있으므로 복제 서버 운영에 주의 하여야 합니다.

7.1.1 구성 과정

- 구성정보

Logical	Primary	Standby
IP	210.104.181.77	210.104.181.78
Port	5432	
AGHOME	/home/agens/AgensSQL-2.13.7.0/	
PGDATA	/home/agens/AgensSQL-2.13.7.0/db_cluster	
Replication User	repluser	

- Replication Node 구성 진행

  2.2 AgensSQL 설치 항목을 참고하여 STANDBY SERVER에 설치 진행

1) Primary 환경 설정

   Replication User 생성
   wal streaming 정보를 전달하기 위하여 replication과 login 권한을 가진 인증된 User를 생성

➔ **Primary Server**

```
[agens@localhost ~]$ asql -U agens -d postgres -p 5432

CREATE USER repluser WITH REPLICATION PASSWORD '1234' LOGIN;
GRANT ALL PRIVILEGES ON DATABASE postgres TO repluser;
```

pg_hba.conf에 replication DB에 접근권한을 추가

pg_hba.conf

```
[agens@localhost db_cluster]$ vi $PGDATA/pg_hba.conf

host    replication     repluser        0.0.0.0/0               trust
# Replication DB에 접속하는 User(repluser) 접속 권한 허용
```

Primary DB Postgresql.conf에 WAL, Replication 등 필요한 파라미터를 수정

```
[agens@localhost db_cluster]$ vi $PGDATA/postgresql.conf

# WRITE-AHEAD LOG
wal_level = logical
synchronous_commit = local
```

AgensSQL을 재기동하여 Rreplication 설정을 적용

```
[agens@localhost db_cluster]$ ag_ctl restart
waiting for server to shut down.... done
server stopped
waiting for server to start....2023-02-09 15:52:35.061 KST [4791] LOG:  redirecting log output to logging collector process
2023-02-09 15:52:35.061 KST [4791] HINT:  Future log output will appear in directory "log".
 done
server started
```

2) Publication 구성

- **Primary Server**

```
# Test Table 생성
CREATE TABLE p_test (
id int,
name text);

# Test Table Replication User에 권한 설정
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO repluser;
# Publication 구성
CREATE PUBLICATION pub_test1 FOR TABLE p_test;
```

3) Subscription 구성

- **Replication Server**

 # Test Table 생성

```
CREATE TABLE p_test (
id int,
name text);
```

 # Subscription 구성

```
CREATE SUBSCRIPTION sub_test1 CONNECTION 'host=210.104.181.77 user=repluser
password=1234 port=5432 dbname=postgres' PUBLICATION pub_test1;
```

4) Replication 확인

- Primary Server

```
[agens@localhost ~]$ asql -U agens -d postgres -p 5432

select * from pg_stat_replication;

 pid  | usesysid | usename  | application_name |  client_addr   | client_hostname |
client_port |         backend_start         | backend_xmin |   state   |  sent_lsn  | write_lsn  |
flush_lsn  | replay_l
sn | write_lag | flush_lag | replay_lag | sync_priority | sync_state |         reply_time
------+----------+----------+-----------------+---------------+----------------+-------------+-----------------------
--------+-------------+----------+-----------+-----------+-----------+--------
---+-----------+-----------+-----------+---------------+------------+-------------------------------
 4972 |    16398 | repluser | sub_test1       | 210.104.181.78 |              |      36795 |
2023-02-09 16:13:20.984184+09 |             | streaming | 0/12AFA150 | 0/12AFA150 |
0/12AFA150 | 0/12AFA1
50 |          |          |          |           0 | async      | 2023-02-09 16:34:53.197891+09
```

5) Replication Test

- Primary Server

```
INSERT INTO p_test VALUES(1,'TEST');
SELECT * FROM p_test;

 id | name
----+------
  1 | TEST
(1 row)
```

- Replication Server

```
SELECT * FROM p_test;
```

```
 id | name
----+------
  1 | TEST
(1 row)
```

7.2 물리적(Physical) 데이터 복제 기능

Physical Replication은 Database의 변경이 발생할때마다 변경 내용을 wal 레코드단위로 복제서버에 전달하여 실시간으로 Database 동기화 합니다. 따라서 운영에 사용하는 DML 뿐만이 아닌 DDL 동작도 모두 반영할 수 있습니다. 추가로 복제서버는 Read-Only로 기동되어 Select 쿼리의 경우 Load Balancing으로 활용 할 수 있습니다. Database 전체를 동일하게 유지 할 수 있기 때문에 HA 및 백업 구성에 주로 활용 됩니다.

7.2.1 구성 과정

- 구성정보

Physical	Primary	Standby
IP	210.104.181.77	210.104.181.78
Port	5432	
AGHOME	/home/agens/AgensSQL-2.13.7.0/	
PGDATA	/home/agens/AgensSQL-2.13.7.0/db_cluster	
Replication User	repluser	

1) Replication User 생성 (PRIMARY DB 진행)

wal streaming 정보를 전달하기 위하여 replication과 login 권한을 가진 인증된 User를 생성

```
[agens@localhost ~]$ asql -U agens -d postgres -p 5432

CREATE USER repluser WITH REPLICATION PASSWORD '1234' LOGIN;
```

pg_hba.conf에 replication DB에 접근권한을 추가

pg_hba.conf

```
[agens@localhost db_cluster]$ vi $PGDATA/pg_hba.conf
# replication privilege
host    replication     repluser        0.0.0.0/0               trust
```

```
# Replication DB에 접속하는 User(repluser) 접속 권한 허용
```

2) Primary 환경 설정 (PRIMARY DB 진행)

Primary DB Postgresql.conf에 WAL, Replication 등 필요한 파라미터를 수정

```
[agens@localhost db_cluster]$ vi $PGDATA/postgresql.conf

# WRITE-AHEAD LOG
wal_level = replica
synchronous_commit = local

# REPLICATION
wal_keep_size = 32
synchronous_standby_names = '*'
promote_trigger_file = '/home/agens/AgensSQL-2.13.7.0/db_cluster/trigger.signal'
hot_standby = on
```

- wal_level (기본값 : replica)

  wal log에 기록되는 정보의 양을 결정, Standby 서버에서 읽기 전용 쿼리를 실행하는 것을 포함하여 wal 보관 및 복제를 지원하는 설정. Streaming Replication을 이용하기 위해선 replica 이상의 설정이 필요합니다.

- synchronous_commit (기본값 : on)

  트랜잭션 Commit 전 wal 레코드의 처리 수준, 동적으로 운영 중 변경 가능하며 세션 및 DB 단위로도 변경 가능, 설정에 따라 운영서버에 지연을 발생하기 때문에 local로 설정 후 sync가 필요한 쿼리에 대하여 부분적 설정을 권고 합니다.

  ➔ remote_apply : 대기서버가 wal 데이터를 대기서버에 적용 완료하였다는 응답을 확인하고 트랜잭션 commit 진행 합니다.
  ➔ on : 대기서버가 wal 데이터를 수신하고 Disk로 저장을 완료하였다는 응답을 확인하고 트랜잭션 commit 진행 합니다.
  ➔ remote_write : 대기서버가 wal 데이터를 수신하여 OS Cache까지 전달을 완료하였다는 응답을 확인하고 트랜잭션 commit 진행 합니다.
  ➔ local : 운영서버가 wal 데이터를 Disk에 저장을 완료한뒤 Transaction commit을 진행합니다.
  ➔ off : 쿼리 실행시 Transaction commit을 진행합니다.

- wal_keep_size (기본 값 0)

Streaming Replication의 경우 wal_keep_segments 값 설정하여 대기서버로 반영되지 못한 wal file을 남겨두어야 합니다. 그렇지 않으면 wal file의 경우 파일을 재사용하기 때문에 대기 서버의 지연이 길어지는 경우 운영서버의 wal file을 재사용되면 Standby 서버와의 동기화는 깨어지게 됩니다.

- synchronous_standby_names (기본값 '')

  Streaming Replication이 진행 될 Standby Server 목록을 지정, 경우에 따라서 Standby Server별 동기화 수준 설정 가능, *로 지정할경우 모든 Standby Server 동기화 합니다.

- promote_trigger_file

  Standby Server의 Recovery Mode(Read-Only) 종료 Trigger File 위치, HA Cluster에서 FailOver 동작시 해당 Trigger File 생성

- hot_standby

  Standby Server에 Replication 설정이 되어 있는 경우 읽기 전용 쿼리를 가능하도록 합니다.

3) AgensSQL 재기동 진행

AgensSQL을 재기동하여 Rreplication 설정을 적용

```
[agens@localhost db_cluster]$ ag_ctl restart
waiting for server to shut down.... done
server stopped
waiting for server to start....2023-02-09 17:05:29.786 KST [5533] LOG:  redirecting log output to logging collector process
2023-02-09 17:05:29.786 KST [5533] HINT:  Future log output will appear in directory "log".
 done
server started
```

4) Standby Replication 구성 (STANDBY 서버 진행)

Standby Server에 AgensSQL 엔진 설치 진행 후 pg_baseBackup을 이용하여 Standby Database 구성

```
# pg_basebackup 전 DATA경로 제거
[agens:/home/agens]#rm -rf $PGDATA

# pg_basebackup 진행
[agens@localhost ~]$ pg_basebackup -h 210.104.181.77 -D $PGDATA -U repluser -p 5432 -v -P
```

```
-R --wal-method=stream

pg_basebackup: initiating base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 0/2000060 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: created temporary replication slot "pg_basebackup_5549"
24725/24725 kB (100%), 1/1 tablespace
pg_basebackup: write-ahead log end point: 0/2000138
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: renaming backup_manifest.tmp to backup_manifest
pg_basebackup: base backup completed
```

pg_basebackup 옵션 설명

-h Primary Database IP

-D Database Cluster 경로

-U Replication User

-p Primary Database Port

-v 작업 상세 정보 표시

-P 진행률 표시

-R Primary postgresql.conf 설정파일 복제

--wal-method 백업 수행 중 wal 데이터 Steaming 설정

5) Standby Database 기동

pg_basebackup 완료 후 Standby Mode로 AgensSQL을 기동 합니다.

pg_is_in_recovery() 함수를 이용하여 recovery 상태 확인 합니다.

Primary DB에서 pg_stat_replication View를 이용하여 Replication 상태를 확인 합니다.

- **Standby Server**

```
[agens@localhost db_cluster]$ ag_ctl start
waiting for server to start....2023-01-16 16:57:19.028 KST [2003] LOG:  redirecting log output to logging collector process
2023-01-16 16:57:19.028 KST [2003] HINT:  Future log output will appear in directory "log".
.... done
server started

# Standby DB Recover 상태 확인
[agens@localhost db_cluster]$ asql -U agens -d postgres -p 5432

select * from pg_is_in_recovery();

 pg_is_in_recovery
-------------------
 t
```

```
(1 row)

    - Primary Server
# Primary DB로 Replication 연결 확인
[agens@localhost ~]$ asql -U agens -d postgres -p 5432
select * from pg_stat_replication;

 pid  | usesysid | usename  | application_name |  client_addr   | client_hostname |
client_port |         backend_start         | backend_xmin |   state   | sent_lsn  | write_lsn |
flush_lsn | replay_lsn
| write_lag | flush_lag | replay_lag | sync_priority | sync_state |         reply_time
------+----------+----------+------------------+----------------+-----------------+----------
 5556 |    16384 | repluser | walreceiver      | 210.104.181.78 |                 |      36884 |
2023-02-09 17:07:17.024833+09 |               | streaming | 0/3000060 | 0/3000060 | 0/3000060
| 0/3000060
|           |           |           |           1 | sync       | 2023-02-09 17:07:37.512722+09
(1 row)
```

6) Replication Test

```
    ➔ Primary Server
CREATE TABLE p_test (
id int,
name text);
INSERT INTO p_test VALUES(1,'TEST');
SELECT * FROM p_test;

 id | name
----+------
  1 | TEST
(1 row)
    - Standby Server
SELECT * FROM p_test;

 id | name
----+------
  1 | TEST
(1 row)
```