

AgensSQL for Postgres Manual



Copyright Notice

Copyright © 2023, Bitnine Inc. All Rights Reserved.

Restricted Rights Legend

PostgreSQL is Copyright © 1996-2023 by the PostgreSQL Global Development Group.

Postgres95 is Copyright © 1994-5 by the Regents of the University of California.

AgensSQL for Postgres © is Copyright © 2023 by Bitnine Inc.

소스코드와 바이너리 형태의 재배포와 사용은 수정 여부와 관계없이 다음 조건을 충족할 때 가능하다.

소스코드를 재배포하기 위해서는 반드시 위의 저작권 표시, 지금 보이는 조건들과 다음과 같은 면책조항을 유지하여야만 한다. 바이너리 형태의 재배포는 배포판과 함께 제공되는 문서 또는 다른 형태로 위의 저작권 표시, 지금 보이는 조건들과 다음과 같은 면책조항을 명시해야 한다. 사전 서면 승인 없이는 저자의 이름이나 기여자들의 이름을 이 소프트웨어로부터 파생된 제품을 보증하거나 홍보할 목적으로 사용할 수 없다. 본 SW는 저작권자와 기여자들에 의해 “있는 그대로” 제공될 뿐이며, 상품가치나 특정한 목적에 부합하는 묵시적 보증을 포함하여(단, 이에 제한되지 않음), 어떠한 형태의 보증도 하지 않는다.

어떠한 경우에도 재단과 기여자들은 제품이나 서비스의 대체 조달, 또는 데이터, 이윤, 사용상의 손해, 업무의 중단 등을 포함하여(단, 이에 제한되지 않음), 본 소프트웨어를 사용함으로써 발생한 직접적이거나, 간접적 또는 우연적이거나, 특수하거나, 전형적이거나, 결과적인 피해에 대해, 계약에 의한 것이든, 엄격한 책임, 불법행위 (또는 과실 및 기타 행위를 포함)에 의한 것이든, 이와 여타 책임 소재에 상관없이, 또한 그러한 손해의 가능성이 예견되어 있었다 하더라도 전혀 책임을 지지 않는다.

Trademarks

AgensSQL for Postgres Manual는 **Bitnine Global Inc.**의 등록 상표입니다. 기타 모든 제품들과 회사 이름은 각각 해당 소유주의 상표로서 참조용으로만 사용됩니다.

Open Source Software Notice

본 제품의 일부 파일 또는 모듈은 다음의 라이선스를 준수합니다. : **OpenSSL, RSA Data Security, Inc., AGPL License, Jean-loup Gailly and Mark Adler, Paul Hsieh's hash**

기술문서 정보

제목 : AgensSQL for Postgres Manual

발행일 : 2023년 07월 12일

소프트웨어 버전 : AgensSQL for Postgres v2.14.5.0

기술문서 버전 : v1.0

목차

I. AgensSQL 소개

1장 AgensSQL for Postgres 이란?

- 1.1 확장기능
- 1.2 확장모듈
- 1.3 PostgreSQL이란?

II. AgensSQL 시작

2장 AgensSQL 설치

- 2.1 최소 사양
- 2.2 OS 환경 설정(Linux 기준)
- 2.3 AgensGraph 설치 및 초기화
- 2.4 Configuration 파일설정
- 2.5 AgensSQL 기동
- 2.6 AgensSQL 접속

III. AgensSQL 관리

3장 AgensSQL 아키텍처

- 3.1 Process 아키텍처
- 3.2 Memory 아키텍처
- 3.3 Database Cluster 구조

4장 AgensSQL의 동시성 제어

- 4.1 ACID 지원
- 4.2 AgensSQL의 MVCC

5장 DB 환경 구성 파일 관리

- 5.1 postgresql.conf 파일
- 5.2 pg_hba.conf 파일
- 5.3 pg_ident.conf 파일

6장 백업과 복구

- 6.1 Online Backup의 유형
- 6.2 백업
- 6.3 복구

7장 Database Replication

- 7.1 논리적(Logical) 데이터 복제 기능
- 7.2 물리적(Physical) 데이터 복제 기능

8장 Database Role 관리

- 8.1 Role 속성
- 8.2 Role 멤버십
- 8.3 Role 조회
- 8.4 Role 생성
- 8.5 Role 삭제
- 8.6 Role 변경

9장 Database 관리

- 9.1 Database 조회
- 9.2 Database 생성
- 9.3 Database 삭제
- 9.4 Database 변경

10장 Tablespace 관리

- 10.1 Tablespace 조회
- 10.2 Tablespace 생성
- 10.3 Tablespace 삭제
- 10.4 Tablespace 변경

11장 Schema 관리

- 11.1 Schema 생성
- 11.2 Schema 삭제
- 11.3 Public SCHEMA
- 11.4 SCHEMA 검색 경로
- 11.5 SCHEMA 권한
- 11.6 System Catalog SCHEMA

IV. SQL 기본

12장 데이터 정의어(DDL)

- 12.1 CREATE TABLE
- 12.2 ALTER TABLE
- 12.3 DROP TABLE
- 12.4 TRUNCATE TABLE
- 12.5 주요 오브젝트 DDL

13장 데이터 제어어(DCL)

- 13.1 COMMIT
- 13.2 ROLLBACK
- 13.3 GRANT
- 13.4 REVOKE

14장 데이터 조작어(DML)

- 14.1 INSERT
- 14.2 UPDATE
- 14.3 DELETE
- 14.4 SELECT

V. 성능 관리

15장 대량 데이터 처리 및 관리

- 15.1 Table Partitioning
- 15.2 병렬처리
- 15.3 SQL Hint 기능(pg_hint_plan)
- 15.4 Partition Pruning 기능
- 15.5 다양한 Index 활용

VI. 보안

16장 Password Profile

- 16.1 Create Password Profile View
- 16.2 Create Password Profile
- 16.3 Create Password Profile Function
- 16.4 How to Alter a Password Profile
- 16.5 How to Delete a Password Profile
- 16.6 How to associating a Profile with an Existing Role/USER
- 16.7 How to Unlock a Locked Account
- 16.8 How to Create a New Role Associated with a Profile

17장 Data Redaction

- 17.1 CREATE REDACTION POLICY
- 17.2 ALTER REDACTION POLICY
- 17.3 DROP REDACTION POLICY

18장 감사기능 (Audit)

- 18.1 개요
- 18.2 Parameters
- 18.3 Audit Log File

VII. 확장모듈

19장 AHM

- 19.1 제품 소개
- 19.2 AHM Architecture
- 19.3 AHM 설치
- 19.4 AHM 구성
- 19.5 AHM 실행

19.6 AHM 유틸리티 (Client Interface)

19.7 지원하는 Failover 시나리오

19.8 시나리오 Example

20장 AEM

20.1 Agens Enterprise Manager이란?

20.2 Agens Enterprise Manager 설치

20.3 환경 설정 및 Database 추가

20.4 CREATE

20.5 Tools

20.6 Password Profile

20.7 Data Redaction

20.8 Auditing Manager

20.9 Agens HA Manager

20.10 Agens Enterprise Manager 제거

Preface

이 매뉴얼은 AgensSQL for Postgres 공식 문서입니다.
(이하 설명부터는 AgensSQL for Postgres를 AgensSQL로 표현합니다)

본서는 아래와 같이 구성되어 있습니다.

Part I. AgensSQL 소개

- 1장은 AgensSQL에 대한 기본적인 이해와 주요 기능에 대해서 설명합니다.

Part II. AgensSQL 시작

- 2장은 AgensSQL의 설치 / OS환경설정 / 기동 및 종료 / 접속 방법 등에 대해 설명합니다.

Part III. AgensSQL 관리

- 3장은 주요 Process / Memory / Data Cluster 등 주요 아키텍처와 구조에 대해 설명합니다.
- 4장은 ACID를 구현하는 메커니즘과 처리 및 관리 방법에 대해 설명합니다.
- 5장은 DB환경 구성 및 파일을 관리하는 방법에 대해 설명합니다.
- 6장은 백업과 복구에 대한 주요 방법을 설명합니다.
- 7장은 가용성 및 안정성을 위한 HA 구성 방안인 Replication에 대해 설명합니다.
- 8장은 DB 접근 관리를 위한 Role에 대해 설명합니다.
- 9장은 Database의 생성, 변경, 삭제 방법 등을 설명합니다.
- 10장은 Tablespace의 생성, 변경, 삭제 방법 등을 설명합니다.
- 11장은 Schema의 생성, 변경, 삭제, 기본기능과 권한, 제약조건 등에 대해 설명합니다.

Part IV. SQL 기본

- 12장은 데이터 정의어 DDL을 설명합니다.
- 13장은 데이터 제어어 DCL을 설명합니다.
- 14장은 데이터 조작어 DML을 설명합니다.

Part V. 성능 관리

- 15장은 대량 데이터를 처리, 관리하는데 필요한 파티셔닝, 병렬 처리, SQL Hint, Partition Pruning, 인덱스에 대해 설명합니다.

Part VI. 보안

- 16장은 사용자 접속 보안을 위한 Password Profile에 대해 설명합니다.
- 17장은 사용자 데이터 보안을 위한 Data Redaction에 대해 설명합니다.
- 18장은 로그 감사를 위한 Audit에 대해 설명합니다.

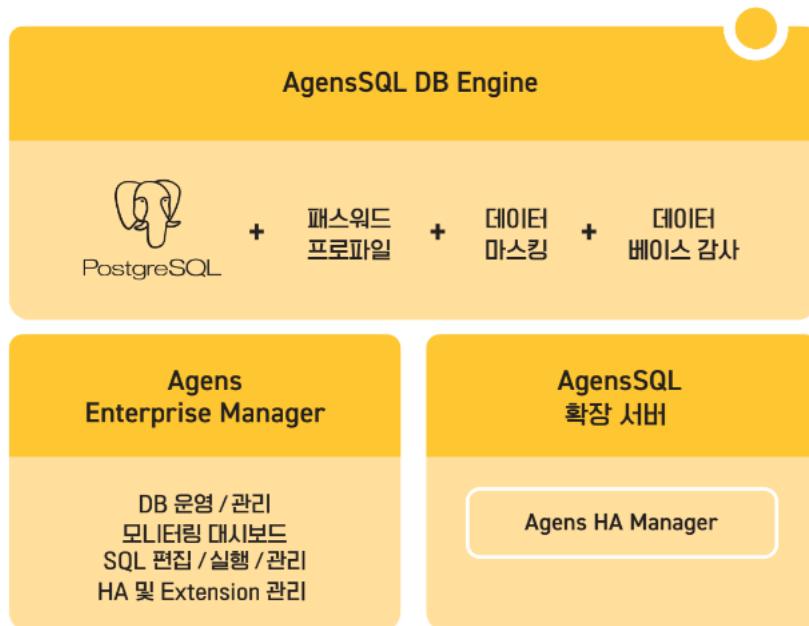
Part VII. 확장 모듈

- 19장은 고가용성을 위한 SW인 AHM(Agens HA Manager)에 대해 설명합니다.
- 20장은 AgensSQL 관리도구인 AEM(Agens Enterprise Manager)에 대해 설명합니다.

I. AgensSQL 소개

1장. AgensSQL for Postgres 이란?

AgensSQL for Postgres은 오픈소스 DBMS인 PostgreSQL과 그 확장 모듈을 통합한 데이터베이스 패키지입니다. 확장 기능으로는 Password Profile, Data Redaction, Audit이 있고, 확장 모듈인 AHM, AEM이 있습니다. (이하 설명부터는 AgensSQL for Postgres를 AgensSQL로 표현합니다.)



1.1 확장 기능

데이터 보안성을 위한 Password Profile, Data Masking, Audit 기능과 안정성 향상을 위한 AHM(Agens HA Manager), 사용자 편의를 위한 AEM(Agens Enterprise Manager)를 제공합니다.

1.1.1 Password Profile

사용자에게 엄격한 비밀번호 정책을 적용하여 Database 로그인 보안 강화 기능.
(세부적인 설명은 16장을 참고바랍니다.)

1.1.2 Data Redaction

특정 데이터를 권한에 따라 Masking 처리하여 사용자 데이터 보안 강화 기능
(세부적인 설명은 17장을 참고바랍니다.)

1.1.3 Audit

데이터베이스의 사용자 활동을 모니터링하거나 추적하고 로그를 기록하는 감사 기능
(세부적인 설명은 18장을 참고바랍니다.)

1.2 확장 모듈

1.2.1 AHM(Agens HA Manager)

AgensSQL 서버의 고가용성을 관리하는 시스템입니다. AHM은 메인 서버 장애 시 대기서버를
메인서버로 신속하게 전환하여 서비스 다운타임을 최소화 합니다.
(세부적인 설명은 19장을 참고바랍니다.)

1.2.2 AEM(Agens Enterprise Manager)

AgensSQL의 관리도구인 AEM은 사용자 친화적인 형태로 AgensSQL의 운영 및 관리를 지원 합니다.
(세부적인 설명은 20장을 참고바랍니다.)

1.3 PostgreSQL이란?

PostgreSQL은 오픈 소스 객체 관계형 데이터베이스 시스템입니다. PostgreSQL은 모든 주요 운영
체제에서 실행되고, 트랜잭션, 하위 선택, 트리거, 뷰, 외래 키 참조 무결성 및 정교한 잠금과 같은
대규모 독점 DBMS에 있는 대부분의 기능을 가지고 있습니다. 또한 PostGIS를 포함하여 추가
기능을 제공하는 많은 확장 기능을 활용할 수 있습니다.

1.3.1 PostgreSQL 주요 기능

- 데이터 유형
 - 프리미티브: 정수, 숫자, 문자열, 부울
 - 구조화: 날짜/시간, 배열, 범위/다중 범위, UUID
 - 문서: JSON/JSONB, XML, 키-값(Hstore)
 - 기하학: 점, 선, 원, 다각형
 - 사용자 정의: 복합, 사용자 정의 유형

- 데이터 무결성
 - NULL이 아닌 고유함
 - 기본 키
 - 외래 키
 - 제외 제약
 - 명시적 잠금, 권고 잠금
- 동시성, 성능
 - 인덱싱: B-트리, 다중 열, 표현식, 부분
 - 고급 인덱싱: GiST, SP-Gist, KNN Gist, GIN, BRIN, 커버링 인덱스, 블룸 필터
 - 정교한 쿼리 플래너/옵티마이저, 인덱스 전용 스캔, 다중 열 통계
 - 트랜잭션, 중첩 트랜잭션(저장점을 통해)
 - 다중 버전 동시성 제어(MVCC)
 - 읽기 쿼리의 병렬화 및 B-트리 인덱스 구축
 - 테이블 파티셔닝
 - 직렬화 가능을 포함하여 SQL 표준에 정의된 모든 트랜잭션 격리 수준
 - 표현식의 JIT(Just-In-Time) 컴파일
- 안정성, 재해 복구
 - 미리 쓰기 로깅(WAL)
 - 복제: 비동기식, 동기식, 논리적
 - 특정 시점 복구(PITR), 활성 대기
 - 테이블스페이스
- 보안
 - 인증: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, 인증서 등
 - 강력한 액세스 제어 시스템
 - 열 및 행 수준 보안
 - 인증서 및 추가 방법을 사용한 다단계 인증

II. AgensSQL 시작

2장. AgensSQL 설치

2023년 2월 현재 최신버전은 PostgreSQL 13.7버전을 기반으로한 AgensSQL 2.13.7.0 버전입니다.

2.1 최소 사양

CPU	1 GHz 이상 프로세서
RAM	2GB 이상
DISK	1GB 이상 여유 공간 (AgensSQL설치 파일을 제외한 DATA 공간이 필요합니다)
OS	CentOS 7 이상

2.2 OS 환경 설정(Linux 기준)

윈도우OS 계열에도 설치 및 구축 가능하나, 운영 서비스는 Linux(x86 Processor 64 bit) 계열을 권장합니다.

2.2.1 사용자 계정 생성

외부에 접근할 수 있는 서버 Daemon과 마찬가지로 AgensSQL을 별도의 사용자 계정으로 실행하는 것이 좋습니다. 이 사용자 계정은 서버가 관리하는 데이터만 소유해야 합니다. 또한 다른 Daemon과 계정을 공유하지 않는 것이 좋습니다.

시스템에 사용자 계정을 추가하려면 useradd(Redhat 계열) 또는 adduser(on Ubuntu) 명령을 사용하면 됩니다. agens 또는 postgres라는 사용자 이름이 주로 사용되며, 설령 다른 이름을 사용해도 무방합니다.

사용자 계정 생성 및 패스워드 설정 (root 또는 sudo 권한 유저 이용)
(Redhat 계열에서 수행시 예제)

```
# useradd agens
```

```
# passwd agens  
New password:  
Retype new password:  
passwd: all authentication tokens updated successfully.
```

2.2.2 환경 변수 설정 및 적용

```
$ vi ~/.bash_profile  
  
export AGHOME=/home/agens/AgensSQL-2.14.5.0  
export PGDATA=/home/agens/AgensSQL-2.14.5.0/db_cluster  
export PATH=$AGHOME/bin:$PATH  
export PS1=[`whoami`:'$PWD']#  
export LD_LIBRARY_PATH=$AGHOME/lib:/usr/lib:/lib:$LD_LIBRARY_PATH  
  
:wq
```

- AGHOME : AgensSQL Engine 설치 경로
- PGDATA : AgensSQL Data Cluster 저장경로

```
$ source ~/.bash_profile
```

2.2.3 공유 메모리 설정

OS의 공유 메모리는 프로세스 간에 데이터를 공유하고자 할 때 사용하는 메모리로 커널에 의해서 관리되어집니다. PostgreSQL은 Sub Process 모델 기반으로 각 프로세스별 데이터를 공유하기 위해서는 공유 메모리가 필요하고, 이를 Shared_buffer 파라미터로 지정합니다. 따라서 OS에서 PostgreSQL이 요구하는 Shared_buffer 메모리 값보다 적은 공유 메모리를 설정하면 PostgreSQL이 동작하지 않습니다. 따라서 물리 메모리의 크기에 따라 Shared_Buffer 값을 정하게 되면 마찬가지로 공유 메모리 설정 파라미터도 변경해줘야 합니다. 변경 방법은 아래와 같습니다.

물리 메모리가 32 GB인 경우

Shared_buffer : 8GB * 보통 물리 메모리의 25~40%로 지정

SHMMAX(최대 공유 메모리, 새 메모리 영역 할당시 사용가능한 최대 메모리양)는 최소 8GB 이상 설정

설정 단위는 bytes, 8 * 1024 * 1024 * 1024

sysctl -w kernel.shmmax=8589934592

SHMAX(모든 공유 메모리)는 28 GB 정도 설정하는 경우

설정 단위는 page(4 KB), (32 - 4 GB) * 1024 * 1024 * 1024 / 4096

sysctl -w kernel.shmall=7340032

ipcs -lm

----- Shared Memory Limits -----

max number of segments = 4096 # SHMMNI

max seg size (kbytes) = 8388608 # SHMAX

max total shared memory (kbytes) = 29360128 # SHMALL

min seg size (bytes) = 1

2.2.4 최대 프로세스 및 열기 파일 설정

아래와 같이 설정을 하면 충분하나, 메모리가 크면 클수록 높여줄 수 있습니다.

```
# vi /etc/security/limits.conf
* soft nproc 65535
* hard nproc 65535
* soft nofile 65535
* hard nofile 65535
```

2.3 AgensSQL 설치 및 초기화

AgensSQL을 설치 파일은 Bitnine 홈페이지(www.bitnine.net)에 접속하여 설치 파일을 다운로드하여 진행 하실 수 있습니다. 다운받은 설치파일을 서버에 업로드 진행 합니다.

(업로드 위치는 사용자 계정의 훔디렉토리에 주로 위치시킵니다.)

2.3.1 AgensSQL Engine 설치

```
$ tar -zvxf AgensSQL-v2.14.5.0_linux_SE.tar.gz
```

(AgensSQL 엔진 설치 디렉토리는 앞서 .bash_profile에 설정한 AGHOME 경로와 일치)

```
$ cd /home/agens/AgensSQL-2.13.7.0
```

2.3.2 Database Cluster 초기화

Database Cluster는 디스크의 데이터베이스 저장 영역을 의미하며, 실행 중인 데이터베이스 서버의 단일 인스턴스를 통해 관리되는 데이터베이스의 모음입니다. 초기화가 끝나면 기본적으로 postgres라는 Database가 포함되며, 이 Database는 유ти리티, 사용자 및 응용 프로그램에서 사용합니다. 또한 template1이라는 Database는 이후에 생성되는 다른 Database의 템플릿으로 사용됩니다. 일반적으로 아래와 같이 초기화를 진행합니다.

```
initdb -E UTF8 --locale=ko_KR.UTF-8 -U agens -D $AGDATA --wal-segsize=128
```

- -E or –encoding : 신규 Database의 기본 인코딩 지정
- –locale : 신규 Database의 기본 Locale 지정
- -U or –username : Database 슈퍼유저명
- -D or –pgdata : Database Cluster의 경로
- –wal-segsize : WAL 파일 사이즈(단위 : MB)

2.4 Configuration 파일 설정

AgensSQL을 기동 하기 전에 기본적으로 DB파라미터 설정 파일인 `postgresql.conf`와 Client 인증 설정 파일인 `pg_hba.conf` 파일을 수정합니다.

2.4.1 postgresql.conf 설정

postgresql.conf 파일은 PostgreSQL에 대한 기본 설정하는 파일로 Database Cluster가 설치된 경로에 위치합니다. 기본 변경 파라미터는 아래와 같습니다.(세부적인 파라미터 정보는 ?장을 참고 바랍니다.)

```
listen_addresses = '*'  
port = 5432  
logging_collector = on  
log_filename = 'agenssql-%Y-%m-%d.log'
```

2.4.2 pg_hba.conf 설정

pg_hba.conf 파일은 PostgreSQL에 접속하는 Client에 대한 인증 설정을 하는 파일로 Database Cluster가 설치된 경로에 위치합니다. 한 줄이 하나의 레코드로 이루어진 레코드들의 집합으로 기본 설정은 아래와 같습니다.

```
# IPv4 local connections:  
TYPE DATABASE USER ADDRESS METHOD  
host all all 0.0.0.0/0 md5
```

- TYPE : local, host, hostssl, hostnossI 중 하나를 설정
- DATABASE : 연결할 수 있는 데이터베이스 이름을 지정. 복수 지정 시 쉼표 (,)로 구분
- USER : 연결할 수 있는 역할 이름을 지정
- ADDRESS : 연결할 수 있는 호스트 이름 또는 IP 주소 범위를 지정
- METHOD : 인증 방법을 지정(세부 인증 방식은 ?장을 참고 바랍니다.)

2.5 AgensSQL 기동

AgensSQL 서비스 관리는 ag_ctl 명령어를 이용하여 관리 합니다.

2.5.1 Syntax

```
ag_ctl ( start | stop | restart | status ) [-D DATADIR]  
-D 옵션을 추가하여 PGDATA 경로를 지정할 수 있습니다.  
(미설정 시 .bash_profile에 지정된 PGDATA를 기본으로 동작 )
```

```
start : AgensSQL을 기동 합니다.  
stop : AgensSQL을 정지 합니다.  
restart : AgensSQL을 재시작 합니다.  
status : AgensSQL의 상태를 확인 합니다.
```

2.5.2 기동

```
$ ag_ctl start  
  
waiting for server to start....2023-05-11 13:52:21.679 KST [72513] LOG: starting PostgreSQL 13.7  
(AgensSQL 2.13.7.0) on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat  
4.8.5-44), 64-bit  
2023-05-11 13:52:21.681 KST [72513] LOG: listening on IPv4 address "0.0.0.0", port 5400  
2023-05-11 13:52:21.682 KST [72513] LOG: listening on IPv6 address "::", port 5400  
2023-05-11 13:52:21.708 KST [72513] LOG: listening on Unix socket "/tmp/.s.PGSQL.5400"  
2023-05-11 13:52:21.730 KST [72514] LOG: database system was shut down at 2023-05-11  
13:52:17 KST  
2023-05-11 13:52:21.743 KST [72513] LOG: database system is ready to accept connections  
done  
server started
```

2.6 AgensSQL 접속

AgensSQL이 정상적으로 기동됐다면, initdb 수행시 지정했던 superuser(미지정 시 OS 사용자 계정명이 자동적으로 superuser명이 됨)를 통해 접속이 가능합니다.

2.6.1 Syntax

```
asql [ -h | -p | -U | -d ]  
Options
```

```
-h : 접속 IP (기본값 Localhost)  
-p : 접속 Port (기본값 Default 5432)  
-U : 접속 User (기본값 OS User)  
-d : 접속 Database (기본값 OS User명과 동일)
```

2.6.2 접속

```
asql -U agens -d postgres -p 5432  
asql (13.7)  
Type "help" for help.
```

2.6.3 DB User Password 설정

```
alter user agens with password 'agens';  
ALTER ROLE
```

2.6.4 버전 확인

```
postgres=# SELECT version();  
version  
-----  
PostgreSQL 13.7 (AgensSQL 2.13.7.0) on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5  
20150623 (Red Hat 4.8.5-44), 64-bit
```

2.6.5 PSQL 명령어

psql 프로그램은 DB를 사용하기 위한 많은 내부 명령을 가지고 있습니다. 그들은 백슬래시 문자"\\"로 시작합니다. 예를 들어, 다양한 PostgreSQL SQL 명령의 구문에 대한 도움말을 다음과 같이 얻을 수 있습니다.

```
mydb=> \h
```

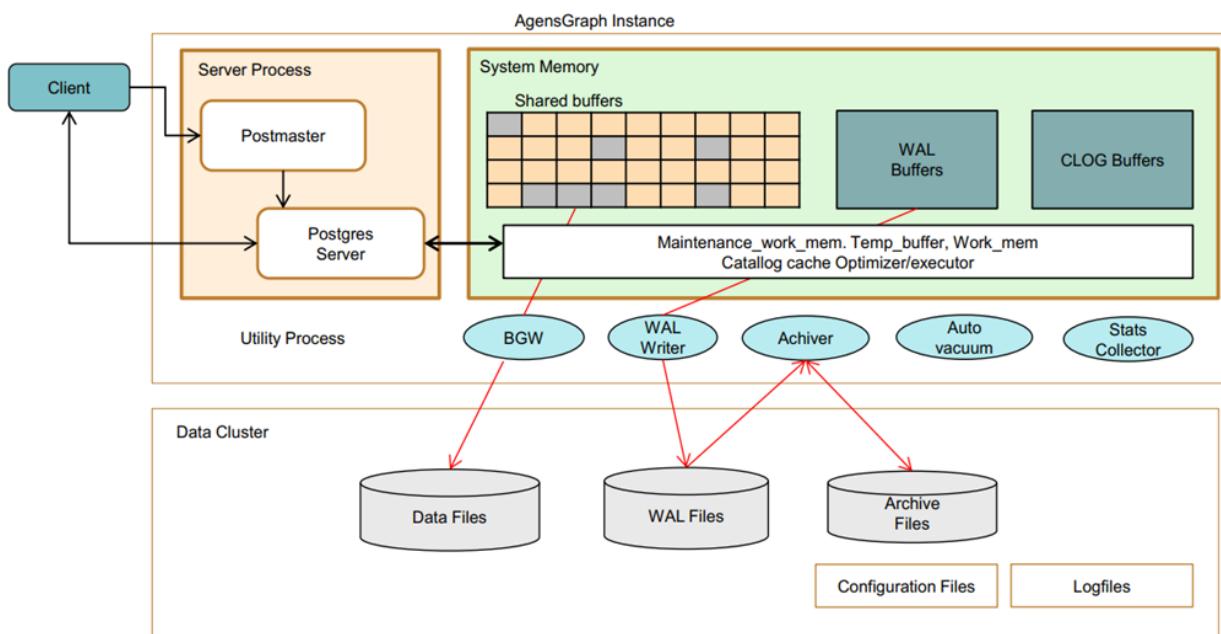
psql을 종료하려면 다음을 입력하십시오.

```
mydb=> \q
```

III. AgensSQL 관리

3장. AgensSQL 아키텍처

AgensSQL은 PostgreSQL에서 fork되었기 때문에 PostgreSQL의 아키텍처를 그대로 따릅니다.

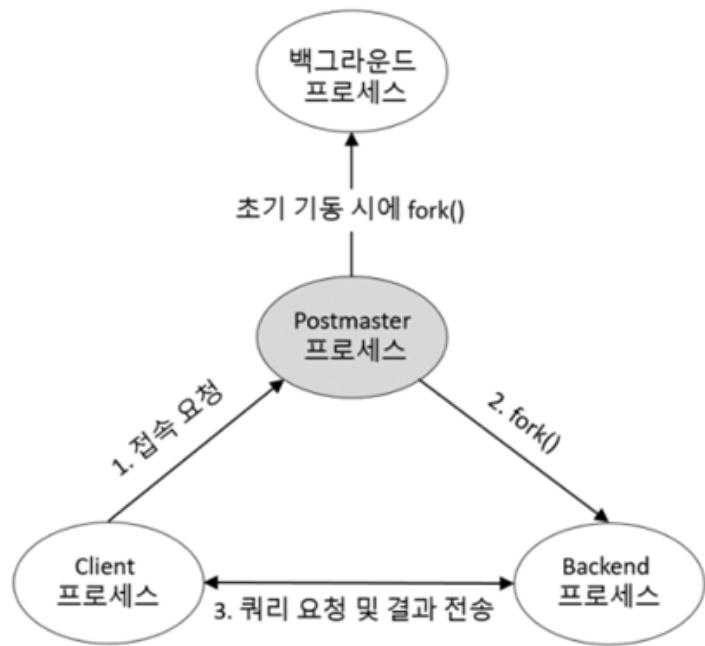


3.1 Process 아키텍처

PostgreSQL은 Client / Server 모델을 사용하며, 프로세스 기반의 DBMS입니다. PostgreSQL에서 사용하는 프로세스들은 아래와 같습니다.

3.1.1. Postmaster Process

client의 권한을 확인한 후 새로운 postgres process를 시작하여 client를 연결합니다. 1개의 Connection(=Client Process)마다 1개의 Backend Process를 생성(Postmaster Process에 의해 fork)합니다.



3.1.2. Multiple Postgres Process

프로세스명	설명
postmaster process	client의 권한을 확인한 후 새로운 postgres process를 시작하여 client를 연결합니다.
WAL writer process	client가 commit을 요청하면 WAL writer는 해당 트랜잭션이 발생하는 모든 WAL 레코드를 WAL 파일로 쓰고 flush 한다
background writer process	shared buffer를 주기적으로 점검하여 dirty page를 데이터 파일에 기록한다
checkpointer process	checkpoint 발생시 더 이상 버퍼 수정이 발생하지 않도록 합니다. background writer process가 dirty page를 flush하도록 하고, WAL Writer process가 checkpoint 레코드를 WAL파일에 기록하고 flush하도록하여 데이터베이스의 일관성을 유지합니다.

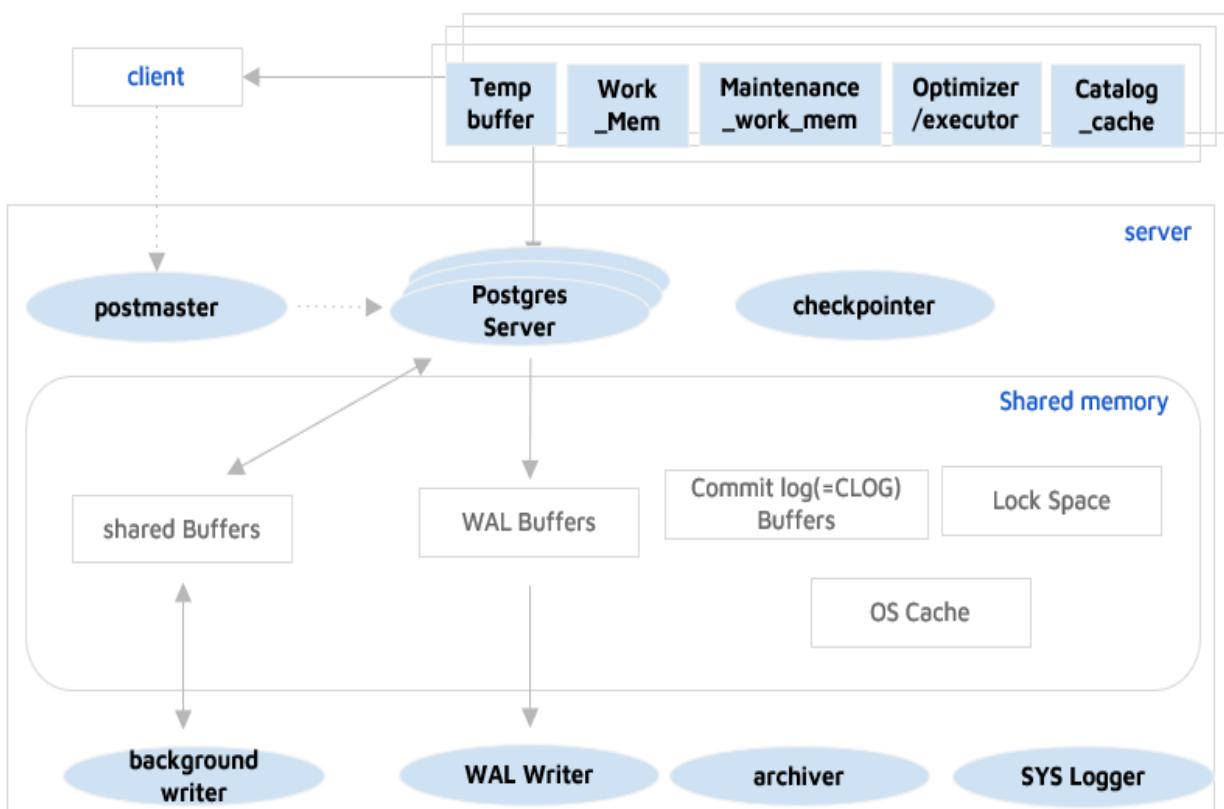
3.1.3. Optional Process

프로세스명	설명
autovacuum launcher process	하위 프로세스를 실행하여 데이터베이스에 대한 vacuum 작업을 한다
logger process	로그 파일에 log, warning, error message를 기록합니다. (WAL file과는 다른 log file임)

archiver process	WAL process에 의해 완전히 채워진 WAL 파일을 구성 가능한 위치에 복사합니다.
stats collector process	table과 index에 대한 액세스 수, table의 총 row 수에 대한 정보를 지속적으로 수집하며 VACUUM/ANALYZE 및 ANALYZE와 함께 동작합니다.
WAL sender/receiver processes	Streaming Replication 기능의 일부를 수행합니다.

3.2 Memory 아키텍처(=Shared Memory)

모든 세션들이 공유하여 사용하는 Shared Memory와 세션별로 할당되는 Backend Memory로 나뉩니다.



3.2.1. Shared Memory

Shared Memory는 모든 세션들이 공유하여 사용하는 공간으로 전역적으로 발생되는 데이터베이스의 작동과 관리에 필요한 공간이 할당됩니다.

항목	설명
Shared Buffer	데이터 영역에서 참조된 데이터를 저장하는 공간
WAL Buffer	데이터 변경시, 변경된 내역을 저장하는 공간. Crash 발생시 복구에 사용
Clog Buffer	트랜잭션 상태 정보를 저장하는 공간
Lock Space	트랜잭션 간의 Lock 정보를 저장하는 공간
Other Buffers	통계정보, two-phases-commit 등의 버퍼 공간

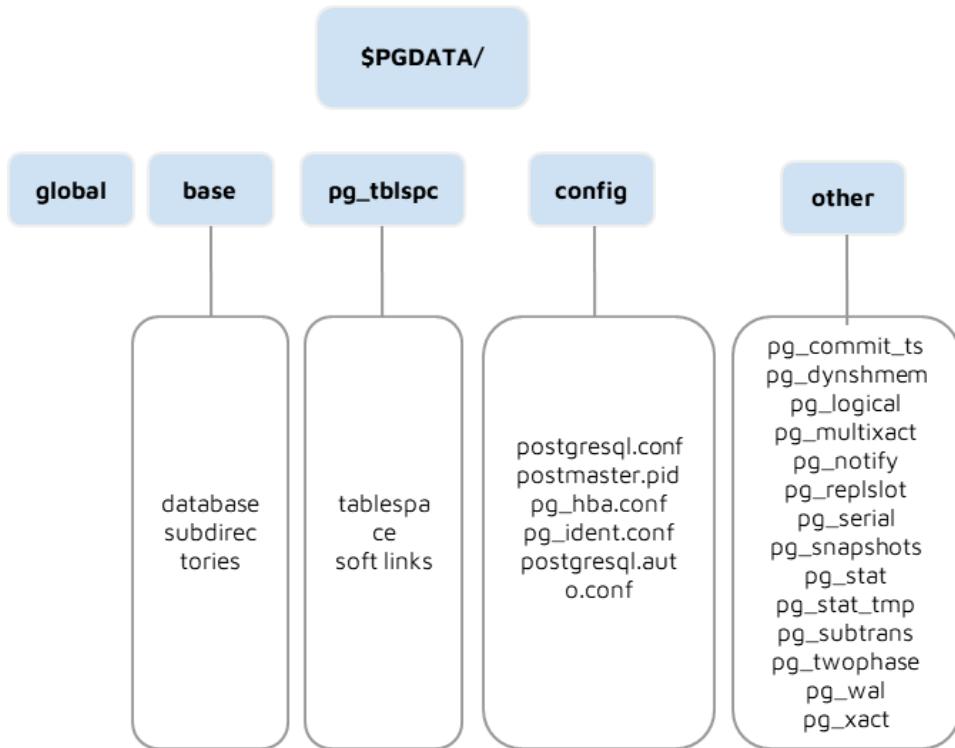
3.2.2. Backend Memory

postgres process는 각 client에 요청을 처리하는 server process에 할당되며, 각 server process에 의해 각각의 세션들에 Backend Memory가 자체적으로 할당됩니다.

항목	설명
temp_buffer	임시 테이블들을 저장하는 공간
work_mem	정렬, 조인, Hash Table 작업 등을 위해서 사용하는 공간
maintenance_work_mem	Vacuum, Create Index 등의 작업을 위해 사용하는 공간
optimizer / executor	수행할 쿼리들에 대한 최적의 실행 계획 수립 및 실행을 담당하는 영역
catalog cache	메타데이터를 조회하거나 이용하기 위한 공간

3.3 Database Cluster 구조

Database Cluster를 initdb를 통해 초기화를 진행하면, 기본적으로 아래와 같이 파일과 디렉토리가 생성됩니다.



항목	설명
PG_VERSION	PostgreSQL 의 주 버전 번호가 포함 된 파일
base	데이터베이스 별 하위 디렉터리를 포함하는 하위 디렉터리
global	pg_database 와 같은 클러스터 전체 테이블을 포함하는 하위 디렉토리
pg_commit_ts	트랜잭션 커밋 타임 스탬프 데이터가 포함 된 하위 디렉터리
pg_clog	트랜잭션 커밋 상태 데이터를 포함하는 하위 디렉터리
pg_dynshmem	동적 공유 메모리 하위 시스템에서 사용하는 파일을 포함하는 하위 디렉토리
pg_logical	논리적 디코딩을 위한 상태 데이터가 포함 된 하위 디렉토리
pg_multixact	다중 트랜잭션 상태 데이터를 포함하는 하위 디렉토리 (공유 행 잠금에 사용됨)
pg_notify	LISTEN / NOTIFY 상태 데이터를 포함하는 하위 디렉토리

pg_replslot	복제 슬롯 데이터가 포함 된 하위 디렉토리
pg_serial	커밋 된 직렬화 가능한 트랜잭션에 대한 정보가 포함 된 하위 디렉터리
pg_snapshots	내 보낸 스냅샷이 포함 된 하위 디렉토리
pg_stat	통계 하위 시스템에 대한 영구 파일이 포함 된 하위 디렉토리
pg_stat_tmp	통계 하위 시스템에 대한 임시 파일이 포함 된 하위 디렉토리
pg_subtrans	서브 트랜잭션 상태 데이터를 포함하는 서브 디렉토리
pg_tblspc	테이블 스페이스에 대한 심볼릭 링크를 포함하는 하위 디렉토리
pg_twophase	준비된 트랜잭션에 대한 상태 파일이 포함 된 하위 디렉터리
pg_xlog	WAL (Write Ahead Log) 파일이 포함 된 하위 디렉토리
postgresql.conf	기본 설정 파일
postgresql.auto.conf	ALTER SYSTEM에서 설정 한 구성 매개 변수를 저장하는 데 사용되는 파일
pg_hba.conf	PostgreSQL에 접속하는 클라이언트에 대한 인증 설정을 설명하는 파일. HBA는 호스트 기반 인증(host-based authentication)의 약어
pg_ident.conf	클라이언트의 인증 방식으로 Ident 인증을 사용하는 경우, ident 사용자 이름을 PostgreSQL의 역할 이름에 매핑하는데 사용하는 파일
current_logfiles	현재 기록되는 로그 파일명 명시
postmaster.opts	서버가 마지막으로 시작된 명령 줄 옵션을 기록하는 파일
postmaster.pid	현재 포스트 마스터 프로세스 ID (PID), 클러스터 데이터 디렉토리 경로, 포스트 마스터 시작 타임 스템프, 포트 번호, Unix 도메인 소켓 디렉토리 경로 (Windows에서는 비어 있음), 첫 번째 유효한 listen_address (IP 주소 또는 * 또는 그렇지 않은 경우 비어 있음)를 기록하는 잠금 파일 TCP에서 수신) 및 공유 메모리 세그먼트 ID (이 파일은 서버 종료 후에 존재하지 않음)

4장. AgensSQL의 동시성 제어

4.1 ACID 지원

PostgreSQL은 관계형 DBMS의 기본적인 기능인 트랜잭션과 ACID 속성을 준수합니다.

[참고] ACID

Atomicity(원자성)

Transaction은 모두 반영되거나 모두 반영되지 않아야 합니다. 여러 쿼리를 하나의 실행 단위로 반영하기 때문에 트랜잭션내의 쿼리들의 결과는 모두 반영되거나 모두 취소하여 일부만 반영되는 일이 없어야 합니다.

Consistency(일관성)

Transaction의 결과는 실행 시점에 따라 데이터의 일관성을 유지하여야 합니다. 가령 트랜잭션의 처리도중 데이터의 변경이 발생한다 하여도 트랜잭션의 결과 데이터는 처리시점을 기준으로 합니다.

Isolation(격리성)

각 Transaction은 수행될 때 서로의 연산에 간섭할 수 없습니다. 즉 한 Transaction의 중간 결과가 다른 트랜잭션에 영향을 주어서는 안됩니다. 격리성을 보장할 수 있는 가장 좋은 방법은 각각의 Transaction을 순차적으로 수행하는 것입니다.

Durability(지속성)

완료된 Transaction의 결과는 데이터베이스에 저장하여 이후 어떤 소프트웨어나 하드웨어 장애에도 데이터를 보존하여야 합니다.

4.2 AgensSQL의 MVCC

RDBMS는 데이터의 일관성을 확보하기 위해 데이터에 대한 버전관리를 통해 이를 해결하는데 이를 MVCC (Multi Version Concurrency Control)라 말합니다. AgensSQL에서는 각 데이터별로 4Byte의 버전정보(XID)를 두어 시점을 식별 합니다. 쿼리 수행 시점의 버전 정보와 데이터의 버전 정보의 비교를 통해 MVCC를 구현합니다.

이전 버전의 데이터를 테이블 블록 내에 저장함으로써, MVCC를 매우 단순하게 구현할 수 있다는 장점이 있는 반면, 불필요한 데이터로 인해 테이블의 공간 사용 효율이 떨어진다는 단점이 있습니다. 이를 해결하기 위해 Vacuum 작업을 자동 및 수동적으로 수행해야 합니다.

4.2.1 MVCC 작동 예제

- **Transaction A (Terminal 1)**

```
# 접속  
asql -U agens -d postgres
```

```
# Test Table 생성
```

```
CREATE TABLE test (a int);
```

```
# Data Insert 진행
```

```
BEGIN;  
INSERT INTO test VALUES (1);
```

- **Transaction B (Terminal 2)**

```
# 접속  
asql -U agens -d postgres
```

```
# Data Insert 진행
```

```
BEGIN;  
INSERT INTO test VALUES (2);
```

- **Transaction C (Terminal 3)**

```
# 접속  
asql -U agens -d postgres
```

```
# pg_stat_activity View를 이용한 XID 확인
```

```
SELECT backend_xid,backend_xmin,query FROM pg_stat_activity;
```

```

backend_xid | backend_xmin |
-----+-----+
 495 |           | INSERT INTO test VALUES (1);
      |           | 495 | SELECT usename,backend_xid,b
 496 |           | insert into test VALUES(2);

```

쿼리 별 XID, XMIN을 이용하여 Data Version 관리

4.2.2 세션별 데이터 확인 예제

transaction_isolation이 default 값인 'read committed' 인 경우 MVCC 작동 방식

Session A (asql -U agens -d postgres)		Session B (asql -U agens -d postgres)	
①	create table test (id numeric, name varchar(20)); insert into test select 1, 'session1'; insert into test select 2, 'session2';		
②	begin; update test set name='session10' where id=1;		
		③	select name from test where id=1; => session1
		④	update test set name='session20' where id=1; => LOCKED WAITING...
⑤	commit;		
			UPDATED 1 => success updated
		⑥	select name from test where id=1; => session20

5장. DB 환경 구성 파일 관리

총 4개의 환경 구성 파일이 존재합니다.

파일명	설명
postgresql.conf	기본 설정 파일(pg_settings Catalog View를 통해서도 확인 가능)
postgresql.auto.conf	ALTER SYSTEM에서 설정 한 구성 매개 변수를 저장하는 데 사용되며, Postgresql.conf 보다 postgresql.auto.conf에 적용된 파라미터가 서버 재 시작 시 우선 적용됨됨
pg_hba.conf	PostgreSQL에 접속하는 클라이언트에 대한 인증 설정을 설명. HBA는 호스트 기반 인증(host-based authentication)의 약어
pg_ident.conf	클라이언트의 인증 방식으로 Ident 인증을 사용하는 경우, ident 사용자 이름을 PostgreSQL의 역할 이름에 매핑하는데 사용

5.1 postgresql.conf

주요 항목별 파라미터 설명은 아래와 같습니다.

5.1.1 Connection 설정

파라미터명	기본값	권장값	적용 유형	설명
listen_addresses	localhost	*	정적	접속 허용 address. '*'는 모든 IP 접속 허용. IP 접근 제어는 pg_hba.conf에서 적용 권장
port	5432	-	정적	기본적으로 서버가 listen하는 TCP 포트
max_connections	200	-	정적	동시에 접속할 수 있는 최대 수를 설정

5.1.2 Memory 설정

파라미터명	기본값	권장값	적용 유형	설명
shared_buffers	128MB	25~40%	정적	서버에서 사용할 공유 메모리 버퍼의 수를 설정합니다.
temp_buffers	8 MB		동적	각 세션에서 사용하는 임시 버퍼의 최대 수를

				설정합니다.
maintenance_work_mem	64 MB	Available Memory / 8	동적	유지 관리 작업에 사용할 최대 메모리를 설정합니다. vacuum, index 생성 등에서 일시적으로 사용됨
work_mem	4MB	(OS cache memory / connections) * 0.5	동적	쿼리 작업 공간에 사용할 최대 메모리를 설정합니다. 내부 정렬 및 해시 테이블에 사용할 메모리 양 설정. 질의 수행 시에 사용하는 메모리. 정렬 속도 등에 영향을 줌.
effective_cache_size	-	Available Memory의 75%	동적	디스크 캐시 크기에 대한 planner가 작업 할 메모리 설정

5.1.3 Log File 설정 및 관리

파라미터명	기본값	권장값	적용 유형	설명
log_directory	log	-	정적	로그 경로 지정
logging_collector	-	on	정적	stderr를 log파일로 저장
log_destination	stderr	-	정적	stderr 및 csvlog, syslog를 비롯한 서버 메시지를 로깅하는 몇 가지 메서드를 지원
log_filename	-	agenssql-%Y-%m-%d_%H%M%S.log	정적	로그 파일의 이름 패턴을 설정합니다.
log_file_mode	600		정적	Unix 시스템에서 이 매개 변수는 logging_collector 가 사용 가능할 때 로그 파일에 대한 사용 권한 설정

5.1.4 Log File 내용 제어

파라미터명	기본값	권장값	적용 유형	설명
log_min_error_statement	error		동적	<p>오류 조건을 발생시키는 SQL 문을 서버 로그에 기록</p> <ul style="list-style-type: none"> - DEBUG[1-5] 개발자를 위한 상세 정보 제공 - INFO 사용자가 암시적으로 요청한 정보 제공 - NOTICE user에게 도움이 될 수 있는 정보 전달 제공 - WARNING 경고에 해당하는 정보 제공

				<ul style="list-style-type: none"> - ERROR 현재 명령 중단의 원인이 되는 에러 정보 제공 - LOG 체크 포인트 활동과 같은 관리자에게 필요한 정보 제공 - FATAL 현재 세션을 중단의 원인이 되는 에러 정보 제공 - PANIC 모든 Database 세션 중단의 원인이 되는 에러 정보 제공
log_min_duration_statement	-1	동적		<p>완료된 각 명령문의 duration 시간을 기록 (default : -1)</p> <ul style="list-style-type: none"> • -1 : 비활성화 • 0 : 모든 명령문 지속 시간 기록 • 250ms : 250ms 이상 실행되는 모든 SQL 문 기록(파라미터 설정 값보다 오래 실행되는 쿼리에 대해 log file에 기록)
log_connections	off	정적		서버로의 연결 시도 및 접속 성공에 대한 정보
log_disconnections	off	정적		접속해제 로그
client_min_messages	notice	동적		클라이언트에 보낼 메시지의 level 설정
log_min_messages	warning	동적		서버 로그에 기록되는 메시지의 level 설정
log_timezone		Asia/Southeast Asia/Seoul		
log_statement	none	all	동적	<p>logging 할 SQL문 제어 (default : none)</p> <ul style="list-style-type: none"> • none(off) • ddl : CREATE, ALTER, DROP • mod : ddl 및 INSERT, UPDATE, TRUNCATE, COPY FROM 등 • all : ALL
log_line_prefix	'<%m-%c(%p):%x>'	정적		<p>로그 Prefix패턴</p> <ul style="list-style-type: none"> • %u : 접속유저명 • %d : 디비명 • %p : 프로세스 ID • %i : command tag • %c : session id • %l : session line number • %s : session start timestamp • %x : transaction id • %q : stop here in non-session processes • %% : '%' 입력시 • %r : 리모트접속자의 HostName (Resolve) 또는 IP와 Port • %t : UnixTimeStamp (일반적으로 사람이 보기 쉬운(흔한) 형태의 시간

5.1.5 WAL 설정 관련

파라미터명	기본값	권장값	적용유형	설명
wal_buffers		-1	정적	<p>WAL 기능을 위해 공유 메모리에서 사용할 디스크 페이지 버퍼 수를 설정합니다.</p> <p>(-1은 shared_buffers에 기반하여 설정됨)</p>
wal_level	minimal		정적	<p>WAL파일에 기록 할 데이터 양. Slave서버에서 읽기전용쿼리에 필요한 정보 추가함</p> <ul style="list-style-type: none"> - minimal : 기본값 - archive : WAL 아카이브에 필요한 로깅만 추가 - hot_standby : 대기 서버에서 읽기 전용 쿼리에 필요한 정보 추가 - replica : PG 9.6 이상부터 archive와 hot_standby 모드가 합쳐진 모드, 하위 호환을 위해 archive/hot_standby로 설정시 본 모드로 동작
wal_writer_delay			동적	WAL 플러시 사이에 WAL 작성기의 절전 시간을 지정합니다.
max_wal_size	1GB		정적	검사 점을 트리거하는 WAL 크기를 설정합니다. PostgreSQL 버전 9.6 이하의 경우 max_wal_size는 16MB 단위입니다. PostgreSQL 버전 10 이상의 경우 max_wal_size는 1MB 단위입니다.
min_wal_size			정적	WAL을 축소할 최소 크기를 설정합니다. PostgreSQL 버전 9.6 이하의 경우 min_wal_size는 16MB 단위입니다. PostgreSQL 버전 10 이상의 경우 min_wal_size는 1MB 단위입니다.
checkpoint_timeout	5min	20min	동적	자동 WAL 체크포인트 사이의 최대 시간을 설정합니다.
checkpoint_completion_target	0.5	0.9	동적	체크포인트 도중 변경된 버퍼 플러시에 사용된 시간으로 체크포인트 간격의 분수 값입니다.

5.1.6 Archive 설정

파라미터명	기본값	권장값	적용유형	설명
archive_mode	off	on	정적	archive_command 설정에 의해 아카이브 저장소로 전달 되도록 설정. wal_level이 minimal로 설정 된 경우 archive_mode를 사용할 수 없다
archive_command		test ! -f /archive/%f && cp %p /archive/%f		완료된 WAL 파일 세그먼트를 아카이브하기 위해 실행하는 로컬 쉘 명령 - string : %p 아카이브할 파일의 경로명으로 대체. %f 파일명으로만 대체
archive_timeout		1~2min		archive_command는 완료된 WAL 세그먼트를 호출 하므로 WAL 트래픽이 발생되지 않아 트랜잭션이 완료되는 시간과 아카이브 저장소에서 안전하게 기록되는 사이에 긴 지연시간이 발생할 수 있음. 데이터가 아카이브되지 않은 채로 방치되지 않기 주기적으로 전환되도록 archive_timeout을 설정

5.1.7 autovacuum 관련

파라미터명	기본값	권장값	적용유형	설명
autovacuum	on		동적	autovacuum 프로세스 사용 여부
autovacuum_freeze_max_age	2억		정적	한 테이블에서 마지막 VACUUM 작업으로 pg_class.relfrozenid 값이 지정된 뒤로, 이 설정값 만큼 더 커지면 VACUUM 작업을 합니다. 트랜잭션 ID 겹치는 문제를 막기위한 것입니다. 알아야할 점은 어떤 다른 상황에서 autovacuum 작업을 무시하고 있어도, 이 작업은 진행됩니다. 초기값은 2억입니다. 이 값은 서버가 시작될 때만 지정할 수 있지만, 각 테이블 단위로 pg_autovacuum 시스템 테이블의 값을 지정함으로 줄일 수는 있습니다.
autovacuum_max_workers			정적	autovacuum 작업자 프로세스를 동시에 실행할 수 있는 최대 수를 설정합니다.
autovacuum_	0.1		동적	한 테이블에서 autovacuum_analyze_threshold 값이

analyze_scale_factor				초과되어 ANALYZE 작업을 시작하려고 할 때, 추가적으로 테이블 크기의 변화량도 함께 고려합니다. 여기서 지정한 만큼 크기가 커졌을 때, 실제 작업을 진행합니다. 초기값은 0.1(테이블 크기의 10%)입니다.
autovacuum_analyze_threshold	250		동적	ANALYZE 작업을 시작할 최소 자료 작업 변경 수, 초기값은 250입니다. 즉, 하나의 테이블에서 250개의 튜플이 추가되었거나, 갱신되었거나, 삭제되었다면, ANALYZE 작업을 합니다.
autovacuum_naptime	1min		동적	autovacuum 데몬이 한번 작업하고, 쉬는 시간을 지정합니다. 이 시간을 경과하면, autovacuum 데몬은 서버가 관리하는 데이터베이스의 모든 테이블을 하나씩 VACUUM 명령과 ANALYZE 명령을 실행할지 안 할지를 검사합니다. 이 쉬는 시간은 초단위로 지정할 수 있습니다. 초기값은 1분이다(1m).
autovacuum_vacuum_cost_delay	-1		동적	autovacuum 데몬의 초과 비용 사용시 멈추는 최대 시간(밀리세컨드)입니다. 이 값이 -1(초기값)이면, VACUUM 작업을 멈추지 않는다. 특별히 지정하지 않으면, vacuum_cost_delay 설정값이 사용될 것입니다.
autovacuum_vacuum_cost_limit	-1		동적	autovacuum 데몬이 사용할 수 있는 최대 비용을 지정합니다. 이 값이 -1(초기값)이면, 작업비용에 제한을 두지 않는다. 특별히 지정하지 않으면, vacuum_cost_limit 설정값이 사용될 것입니다.
autovacuum_vacuum_scale_factor	0.2		동적	한 테이블에서 autovacuum_vacuum_threshold 값이 초과되어 VACUUM 작업을 시작하려고 할 때, 추가적으로 테이블 크기의 변화량도 함께 고려합니다. 여기서 지정한 만큼 크기가 커졌을 때, 실제 작업을 진행합니다. 초기값은 0.2(테이블 크기의 20%)입니다.
autovacuum_vacuum_threshold	500		동적	VACUUM 작업을 시작할 최소 자료 작업 변경 수, 초기값은 500입니다. 즉, 하나의 테이블에서 500개의 튜플이 갱신되었거나, 삭제되었다면, VACUUM 작업을 합니다.

5.1.8 Extension Parameters 설정

파라미터명	기본값	권장값	적용 유형	설명
shared_preload_libraries	none	'pg_stat_statements,pg_hint_plan,pg_prewarm'		서버 시작 시에 사전 로드할 하나 이상의 공유 라이브러리를 쉼표를 사용하여 지정합니다. 이 매개변수는 서버 시작 시에만 설정 가능합니다. 지정된

				라이브러리를 찾지 못하면 서버 시작이 실패합니다.
pg_stat_statements.max	1000		정적	처리할 수 있는 최대 쿼리 수를 지정. pg_stat_statements 뷰에 보일 최대 로우 수 pg_stat_statements.max * track_activity_query_size 바이트만큼의 공유 메모리를 추가로 필요
pg_stat_statements.track	top		정적	수집 할 쿼리문의 사용빈도에 따른 쿼리문 사용 통계 정보 수집 범위를 지정. 슈퍼유저만 바꿀 수 있음 top : 자주 사용하는 쿼리 대상 all : 모든 쿼리 대상 none : 아무 쿼리도 수집하지 않음.
pg_stat_statements.track_utility (boolean)	on		정적	SELECT, INSERT, UPDATE, DELETE 구문 외 다른 구문들도 수집 대상으로 할 것인지를 지정 (on / off)
pg_stat_statements.save (boolean)	on		정적	서버가 중지 되고, 재실행 되었을 때, 마지막 중지 시점의 쿼리문 통계 정보를 저장할 것인지를 지정
pg_hint_plan.enable_hint	on			True enables pg_hint_plan.
pg_hint_plan.enable_hint_table	on			True enables hinting by table.
pg_hint_plan.parse_messages	INFO			Specifies the log level of hint parse error. Valid values are error, warning, notice, info, log, debug.
pg_hint_plan.debug_print	off			Controls debug print and verbosity. Valid values are off, on, detailed, and verbose.
pg_hint_plan.message_level	INFO			Specifies message level of debug print. Valid values are error, warning, notice, info, log, debug.

5.1.9 통계정보 관련

파라미터명	기본값	권장값	적용 유형	설명
track_activity_query_size	1024		정적	pg_stat_activity.current_query에 예약되는 크기(바이트)를 설정합니다.

track_activities	on		동적	각 세션에서 현재 실행 중인 명령의 실행이 시작될 때 해당 명령에 대한 정보 수집을 활성화
track_counts	on		동적	데이터베이스 작업에 대한 통계 수집을 활성화
track_functions	none		동적	함수 호출 횟수 및 사용된 시간의 추적을 활성화합니다. 프로시저 언어 함수 all만 추적하기 위해 pl을 지정하면 SQL 및 C 언어 함수도 추적
track_io_timing	off		동적	데이터베이스 I/O 호출의 타이밍을 활성화합니다. 운영 체제에 현재 시간을 반복해서 쿼리함으로써 일부 플랫폼에서는 상당한 오버헤드가 발생되므로 이 매개변수는 기본적으로 off이다
stats_temp_directory (string)	pg_stat_tm p		정적	임시 통계 데이터를 저장할 디렉터리를 설정합니다. 디렉터리에 대한 상대 경로이거나 절대 경로일 수 있습니다.
log_planner_stats (boolean)			동적	planner 성능 통계를 서버 로그에 기록합니다.

5.1.10 SQL 실행 계획 관련

파라미터명	기본값	권장값	적용 유형	설명
auto_explain.log_min_duration			동적	5초 이상 소요되는 sql에 대해 쿼리 실행 계획을 로그에 기록
auto_explain.log_analyze			동적	쿼리 실행 계획을 로그에 기록. 그런데 EXPLAIN ANALYZE 명령문을 기반으로 로그를 남기기 때문에, 롱쿼리를 다시 실행시킨다는 리스크가 있습니다.(만약 롱쿼리가 갑자기 몰리는 상황에서, 로그를 남기기 위해 EXPLAIN ANALYZE 쿼리가 날라간다면, 롱쿼리가 2배로 실행되어 문제를 더 키울 수 있습니다.) 그리고 단일 롱쿼리만 파악할 수 있기 때문에, 짧지만 여러번 호출되어 문제를 일으키는 쿼리를 알 수 없단 단점이 있습니다.

debug.pretty_print			동적	구문과 실행 계획 트리를 들여쓰기 하여 표시합니다.
debug.print_parse			동적	각 쿼리의 구문 분석 트리를 기록합니다.
debug.print_plan			동적	각 쿼리의 실행 계획을 기록합니다.
debug.print_rewritten			동적	각 쿼리에서 재작성된 구문 분석 트리를 기록합니다.
default_statistics_target			동적	기본 통계 대상을 설정합니다.
enable_bitmapscan			동적	planner가 비트맵 스캔 계획을 사용할 수 있도록 활성화합니다.
enable_hashagg			동적	planner가 해시된 집계 계획을 사용할 수 있도록 활성화합니다.
enable_hashjoin			동적	planner가 해시 조인 계획을 사용할 수 있도록 활성화합니다.
enable_indexscan			동적	planner가 인덱스 스캔 계획을 사용할 수 있도록 활성화합니다.
enable_material			동적	planner가 구체화를 사용할 수 있도록 활성화합니다.
enable_mergejoin			동적	planner가 병합 조인 계획을 사용할 수 있도록 활성화합니다.
enable_nestloop			동적	planner가 중첩 루프 조인 계획을 사용할 수 있도록 활성화합니다.
enable_seqscan			동적	planner가 순차적 스캔 계획을 사용할 수 있도록 활성화합니다.
enable_sort			동적	planner가 명시적 정렬 단계를 사용할 수 있도록 활성화합니다.
enable_tidscan			동적	planner가 TID 스캔 계획을 사용할 수 있도록 활성화합니다.
constraint_exclusion			동적	planner가 제약 조건을 사용하여 쿼리를 최적화하도록 활성화합니다.
cpu_index_tuple_cost			동적	인덱스 스캔 중 각 인덱스 항목을 처리하는 데 따른 planner의 예상 코스트를 설정합니다.
cpu_operator_cost			동적	각 연산자 또는 함수 호출을 처리하는 데 따른 planner의 예상 코스트를 설정합니다.
cpu_tuple_cost			동적	각 튜플(행)을 처리하는 데 따른 planner의 예상 코스트를 설정합니다.
cursor_tuple_fraction			동적	planner가 예상하는 검색할 거서 행의 분수 값을 설정합니다.

random_page_cost			동적	비순차적으로 가져온 디스크 페이지에 대한 planner의 예상 코스트를 설정합니다.
------------------	--	--	----	--

5.2 pg_hba.conf 파일

클라이언트 인증 설정 파일인 `pg_hba.conf` 안의 각 레코드는 연결 형식, 클라이언트 IP 주소 범위, 데이터베이스 이름, 사용자 이름 및 이러한 매개 변수와 일치하는 연결에 사용되는 인증 방법을 지정합니다. 어떤 레코드도 일치하지 않으면 액세스가 거부됩니다.

레코드는 다음 7가지 형식 중 하나입니다.

```
local  database user auth-method [auth-options]
host   database user address auth-method [auth-options]
hostssl database user address auth-method [auth-options]
hostnossal database user address auth-method [auth-options]
host   database user IP-address IP-mask auth-method [auth-options]
hostssl database user IP-address IP-mask auth-method [auth-options]
hostnossal database user IP-address IP-mask auth-method [auth-options]
```

필드의 의미는 다음과 같습니다.

- **local**

이 레코드는 Unix 도메인 소켓을 사용한 연결에 해당합니다. 유형의 레코드가 없으면 Unix 도메인 소켓 연결은 불가능합니다.

- **host**

이 레코드는 TCP/IP를 사용한 연결에 해당합니다. host 레코드는 SSL 연결 시도 혹은 비 SSL 연결 시도와 일치합니다.

- **hostssl**

이 레코드는 TCP/IP를 사용한 연결 시도와 일치하지만, SSL 암호화를 사용한 연결에만 해당됩니다. 이 옵션을 사용하려면 서버는 SSL 지원이 내장되어 있어야 합니다. 또한 SSL은 ssl 환경 설정 매개 변수를 설정함으로써 서버 시작 시에 활성화되어야 합니다.

- **hostnossal**

이 레코드 유형은 hostssl과는 반대로 동작합니다. SSL을 사용하지 않는 TCP/IP 상의 연결 시도에 대해서만 일치합니다.

- **database**

이 레코드는 데이터베이스 이름을 지정합니다.

- all 값은 모든 데이터베이스와 일치하도록 지정합니다.
- sameuser 값은 요청된 데이터베이스가 요청된 사용자와 이름이 동일한 경우에 레코드가 일치하도록 지정합니다.
- samerole 값은 요청된 사용자가 요청된 데이터베이스와 이름이 동일한 role의 멤버여야 하는지 지정합니다. superuser는 직접 혹은 간접적으로 role의 명시적인 멤버가 아닐 경우, 단지 superuser라는 이유로 samerole에 대한 role의 멤버로 간주되지 않습니다.
- replication 값은 복제 연결이 요청되는 경우 레코드가 일치하도록 지정합니다.(복제 연결은 특정 데이터베이스를 지정하지는 않습니다). 이 경우가 아니라면 특정 AgensSQL 데이터베이스의 이름으로 사용됩니다. 쉼표로 구분해서 데이터베이스 이름을 여러 개 쓸 수 있습니다. 데이터베이스 이름이 포함된 파일은 파일 이름 앞에 @를 붙여서 지정 가능합니다.

- **user**

이 레코드와 일치하는 데이터베이스 사용자 이름을 지정합니다. all 값은 모든 사용자와 일치하도록 지정합니다. 이 외에는, 특정한 데이터베이스 사용자의 이름이거나 앞에 +를 붙인 그룹 이름입니다. 이러한 이유로, superuser는 단지 superuser라는 이유 때문이 아니라, 직접 혹은 간접적으로 role의 명시적 멤버인 경우에만 role 멤버로 간주됩니다. 쉼표로 구분해서 사용자 이름을 여러 개 쓸 수 있습니다. 사용자 이름이 포함된 파일은 파일 이름 앞에 @를 붙여서 지정 가능합니다.

- **address**

이 레코드와 일치하는 클라이언트 머신 주소를 지정합니다. 이 필드는 호스트 이름, IP 주소 범위 또는 아래 설명된 특수 키워드 중 하나를 포함할 수 있습니다. IP 주소는 CIDR 마스크 길이의, 점으로 구분된 십진수(dotted decimal) 표준 표기법으로 지정됩니다. 마스크 길이는 일치해야 하는 클라이언트 IP 주소의 상위 비트 수를 나타냅니다. 이것의 오른쪽에 있는 비트는 주어진 IP 주소에서 0이어야 합니다. IP 주소 및 /, CIDR 마스크 길이 사이에 공백이 있으면 안됩니다.

이러한 방법으로 지정된 IP 주소 범위의 전형적인 예시는 단일 호스트의 경우 172.20.143.89/32, 소규모 네트워크의 경우 172.20.143.0/24, 대규모 네트워크의 경우 10.6.0.0/16일 수 있습니다. 0.0.0.0/0은 모든 IPv4 주소를 나타내며 ::/0은 모든 IPv6 주소를

나타냅니다. 단일 호스트를 지정하려면 IPv4의 경우 CIDR 마스크 32를 사용하고 IPv6의 경우 128을 사용해야 합니다. 네트워크 주소 끝에 0을 빠트리면 안 됩니다.

사용자는 아무 IP 주소나 일치하도록 `all`을 쓸 수도 있고, 서버의 자체 IP 주소만을 일치하도록 `samehost`를 쓸 수도 있고, 서버가 직접 연결되는 서브넷의 모든 주소와 일치하도록 `samenet`을 쓸 수도 있습니다.

- **IP-address, IP-mask**

이 필드는 CIDR-address 표기의 대안으로 사용될 수 있습니다. 마스크 길이를 지정하는 대신 실제 마스크가 쉼표로 구분하여 지정됩니다. 예를 들면, 255.0.0.0은 IPv4 CIDR 마스크 길이 8을 나타내고, 255.255.255.255는 CIDR 마스크 길이 32를 나타냅니다. 이 필드는 `host` 및 `hostssl`, `hostnossl` 레코드에 적용됩니다.

- **auth-method**

연결이 이 레코드와 일치할 때 사용하는 인증 방법을 지정합니다. 가능한 선택안은 다음과 같습니다.

- **trust**

무조건 연결을 허용합니다. 이 방법은 패스워드나 다른 인증 없이 임의의 데이터베이스 사용자로 로그인하여 누구나 데이터베이스 서버에 연결할 수 있습니다.

- **reject**

무조건 연결을 거부합니다. 이것은 그룹에서 특정 호스트를 “필터링”할 때 유용합니다. 예를 들면, `reject` 줄은 특정 호스트의 연결을 차단하고, 그 이후의 줄은 특정 네트워크의 남은 호스트들과의 연결을 허용합니다.

- **md5**

클라이언트가 인증을 위해 double-MD5-hashed 패스워드를 제공해야 합니다.

- **password**

클라이언트가 인증을 위해 암호화되지 않은 패스워드를 제공해야 합니다. 패스워드는 네트워크 상에서 일반 텍스트로 전송되므로 신뢰하지 않는 네트워크에서 이것을 사용하면 안 됩니다.

- **gss**

GSSAPI를 사용하여 사용자를 인증합니다. 이것은 TCP/IP 연결에서만 사용할 수 있습니다.

- **sspi**

SSPI를 사용하여 사용자를 인증합니다. 이것은 Windows에서만 사용할 수 있습니다.

- **ident**

클라이언트의 ident 서버에 접속함으로써 클라이언트의 운영 체제 사용자 이름을 획득하고, 요청된 데이터베이스 사용자 이름과 일치하는지 확인합니다. Ident 인증은 TCP/IP 연결에서만 사용할 수 있습니다. 로컬 연결에 대해 지정하는 경우 피어(peer) 인증이 대신 사용됩니다.

- **peer**

클라이언트의 운영 체제 사용자 이름을 운영 체제에서 획득하고, 요청된 데이터베이스 사용자 이름과 일치하는지 확인합니다. 이것은 로컬 연결에서만 사용할 수 있습니다.

- **ldap**

LDAP 서버를 사용하여 인증합니다.

- **radius**

RADIUS 서버를 사용하여 인증합니다.

- **cert**

SSL 클라이언트 인증을 사용하여 인증합니다.

- **pam**

운영 체제에서 제공하는 PAM(Pluggable Authentication Modules)을 사용하여 인증합니다.

- **auth-options**

auth-method 필드 이후에 인증 방법에 대한 옵션을 지정하는 name=value 형식의 필드가 있을 수 있습니다.

@ 구문이 포함된 파일은, 공백 또는 쉼표로 구분된 이름 목록으로 읽습니다. pg_hba.conf처럼 #으로 표시된 주석 및 중첩된 @ 구문이 허용됩니다. 파일 이름 뒤에 @가 나오는 것이 절대 경로가 아니면 참조 파일이 있는 디렉토리의 상대 경로로 취급됩니다.

pg_hba.conf 레코드는 각 연결 시도에 대해 순차적으로 검사되므로 레코드의 순서는 중요합니다. 일반적으로 초기 레코드는 연결 일치 매개 변수는 치밀하고, 인증 방법은 느슨한 반면, 후기 레코드는 일치 매개 변수는 느슨하고 인증 방법은 강력합니다. 예를 들면, 로컬 TCP/IP 연결에 대한 trust 인증을 사용하려고 하면서 원격 TCP/IP 연결을 할 수도 있습니다. 이런 경우 127.0.0.1로부터 연결을 위한 trust 인증을 지정한 레코드는 다양한 허용 클라이언트 IP 주소에 대해 패스워드 인증을 지원하는 레코드 이전에 나타납니다.

5.3 pg_ident.conf 파일

Ident 또는 GSSAPI 같은 외부 인증 시스템을 사용하는 경우, 연결을 시작하는 운영 체제 사용자의 이름은 연결해야 하는 데이터베이스 사용자 이름과 다를 수 있습니다. 이런 경우 사용자 이름 맵을 사용하여 운영 체제 사용자 이름과 데이터베이스 사용자 이름을 맵핑할 수 있습니다. 사용자 이름 맵핑을 사용하려면 pg_hba.conf 옵션 필드에서 map=map-name을 지정해야 합니다. 이 옵션은 외부 사용자 이름을 수신하는 모든 인증 방법에서 지원됩니다. 서로 다른 연결에 서로 다른 맵핑이 필요할 수 있으므로 연결별로 사용할 맵을 지정하기 위해 사용할 맵의 이름은 pg_hba.conf의 map-name 매개 변수에서 지정됩니다.

사용자 이름 맵은 ident 맵 파일에서 정의되며, 기본적으로 이름은 pg_ident.conf이며 데이터 디렉토리에 저장됩니다.

```
map-name system-username database-username
```

주석 및 공백은 pg_hba.conf에서 와 동일하게 처리됩니다. map-name은 pg_hba.conf에서 이 맵핑을 참고하기 위해 사용되는 임의의 이름입니다. 나머지 2개의 필드는 운영 체제 사용자 이름과 데이터베이스 사용자 이름을 지정합니다. 동일한 map-name을 여러 번 사용해서 단일 맵 내에서 여러 사용자 맵핑을 지정할 수 있습니다.

주어진 한 명의 운영 체제 사용자가 몇 명의 데이터베이스 사용자에 대응하는지에 대해서는 아무런 제한이 없습니다.(그 반대도 마찬가지). 따라서, 맵의 항목은 사용자가 동일함을 의미한다기보다 “이 운영 체제 사용자는 이 데이터베이스 사용자로서 연결이 허용된다”로 생각되어야 합니다. 사용자가 연결 요청을 한 데이터베이스 사용자 이름을 사용하여 외부 인증 시스템에서 획득한 사용자 이름과 쌍을 이루는 맵 항목이 있을 경우 연결이 허용됩니다.

system-username 필드가 슬래시(/)로 시작되는 경우 필드의 나머지는 정규식으로 처리됩니다. 정규식은 단일 캡처 또는 괄호 표현식을 포함할 수 있으며, \1(역슬래시)로 database-username 필드에서 참조가 가능합니다. 이것은 한 줄로 된 여러 사용자 이름을 맵핑할 수 있으며, 단순 구문 대체 시 특히 유용합니다. 예를 들면, 다음과 같습니다.

```
mymap /(.*)@mydomain\.com$\1
mymap /(.*)@otherdomain\.com$ guest
```

이 항목은 @mydomain.com로 끝나는 시스템 사용자 이름을 사용하여 사용자에 대한 도메인 부분을 삭제하고, 시스템 이름이 @otherdomain.com로 끝나는 모든 사용자가 guest로 로그인하는 것을 허용합니다.

6장. 백업과 복구

서비스 운영 중 시스템 장애, 사용자의 실수 등 예상치 못한 상황으로부터 데이터 보호를 위해 데이터베이스 백업은 매우 중요합니다. 데이터베이스 백업은 DownTime이 필요한 COLD BACKUP과 DownTime 없이 가능한 다양한 Online Hot Backup이 있습니다. AgensSQL에서 Online Backup은 Archive Log를 이용하여 전체 데이터베이스를 완전 복구 및 시점 복구가 가능한 물리적 백업인 Directory Copy를 이용한 Low Level API 백업과 백업 시점 데이터를 부분적 복구가 가능한 논리적 백업인 pg_dump가 있습니다.

6.1 Online Backup의 유형

6.1.1 Archive Log Backup

Archive Log란 AgensSQL의 Wal Log를 Log Switch 발생시 File형태로 보관소에 저장한 Log입니다. 데이터베이스의 모든 변경이력을 가지고 있어 복구시 해당 파일을 이용하여 데이터베이스 완전복구 및 시점복구에 사용되기 때문에 해당 Log File의 백업이 중요합니다. Archive Log File은 로그 데이터로 백업 후에는 삭제해도 무관합니다.

6.1.2 Low Level Backup

Physical Backup인 Low Level Backup은 Database에 백업 시점 Label을 설정 한뒤 Data Cluster를 전체를 OS 명령어를 이용하여 백업하여 비교적 단순하게 백업이 가능합니다. 복구시에 백업한 Data Cluster를 제자리에 위치한 뒤 Archive Log Backup을 이용하여 복구를 진행 합니다. 따라서 백업을 진행하기 위해선 Archive Log Mode에서 가능 합니다.

6.1.3 pg_dump

Logical Backup인 pg_dump Backup은 데이터베이스의 모든 변경기록을 보유하고 있진 않기 때문에 백업 시점의 해당 데이터베이스 데이터만을 복구 할 수 있습니다. 하지만 Logical Backup이기 때문에 복구가 필요한 Data만을 추출하여 빠르게 복구 할 수 있는 장점 있습니다.

6.2 백업

6.2.1 Archive Log 설정 및 Backup

- 1) Archive 백업 지점 생성 및 권한 설정 (백업 지점 : /home/agens/backup)

```
mkdir -p /home/agens/backup/archive  
chown -R agens:agens /home/agens/backup
```

- 2) 환경설정 파라미터 수정 (파라미터 파일 : \$PGDATA/postgresql.conf)

```
vi $PGDATA/postgresql.conf
```

```
archive_mode = on  
archive_command = 'test ! -f /home/agens/backup/archive/%f && cp %p  
/home/agens/backup/archive/%f'
```

- 3) AgensSQL 재기동

```
ag_ctl restart
```

- 4) Log Switch 실행 (pg_switch_wal를 이용하여 로그 Switch)

```
asql -U agens -d postgres  
select pg_switch_wal();
```

- 5) Archive Log 확인

```
ls /home/agens/backup/archive
```

6.2.2 Low Level Backup

- 1) Backup Start Function

Backup Start Function : **select pg_start_backup('Label');**
시작 시 Backup 시작 시점 WAL Segments를 기록합니다.

2) Backup End Function

Backup Stop Function : **select pg_stop_backup();**

종료 시 Backup 종료 시점 WAL Segments를 기록합니다.

3) Examples

pg_start_backup을 이용하여 백업 Start

```
asql -U agens -d postgres -c "select pg_start_backup('Hot Backup');"
```

tar를 이용하여 Data Cluster Directory Backup 진행

```
tar -cvf /home/agens/backup/hotbackup.tar $PGDATA
```

pg_stop_backup을 이용하여 백업 End

```
asql -U agens -d postgres -c "select pg_stop_backup();"
```

pg_switch_wal을 이용한 Log switch 진행

```
asql -U agens -d postgres -c "select pg_switch_wal();"
```

6.2.3 pg_dump

1) pg_dump Options

F(format) : 백업 포맷 선택

p(plain) : SQL script로 백업

c(custom) : custom-format으로 백업. 기본적으로 압축.

d(directory) : directory-format으로 백업. 기본적으로 압축.

t(tar) : tar-format으로 백업. 기본적으로 미압축

f(file) : 백업 파일명 지정

n(schema) : Schema 지정

ex1) -n schema1 -n schema2 -n schema3

ex2) -n 'schema*'

N(exclude-schema) : Schema 제외

t(table) : 테이블 지정 (-t 옵션 사용 시 -n 옵션은 적용되지 않음)

ex1) -t table1 -t table2 -t table3

ex2) -t 'table*'

N(exclude-table) : 테이블 제외

d(dbname) : DB 지정

U(username) : 백업 시 접속할 사용자명

```
v(verbose) : 진행 상황 확인
```

2) Examples

```
# Backup 진행
```

```
pg_dump -Fd -d postgres -f /home/agens/backup/logical_backup -U agens -p 5432 -v
```

6.3 복구

6.3.1 Physical Recovery

Physical Recovery의 경우 시스템 장애 및 데이터의 손상으로 인한 Data의 물리적 장애 상황에서 데이터 복구를 위해 주로 사용됩니다. 운영 서비스의 Disk에 여유가 있다면 과거 Data Cluster를 임시 경로로 이동 시킨 뒤 백업해둔 Physical Backup Data를 Data Cluster로 위치한 뒤 백업 이후의 Archive Log File들을 이용하여 Archive에 존재하는 최신의 시점까지 Database를 복구 합니다.

Examples

```
# Database 손실 상황 가정 DB Down 후 Data Cluster 제거
```

```
ag_ctl stop  
rm -rf $PGDATA
```

```
# Hot Backup Data 압축 해제후 기존 Data Cluster로 변경
```

```
tar -xvf hotbackup.tar  
cd home/agens/AgensSQL-2.13.7.0/  
mv -f db_cluster/ $PGDATA
```

```
# Database Start 진행 (Database Auto Recovery를 이용하여 복구)
```

```
ag_ctl start
```

6.3.2 Logical Recovery

Logical Recovery의 경우 주로 사용자의 실수 등의 예상치 못한 Data의 손실로 인한 논리적 장애 상황에서 데이터 복구를 위해 주로 사용됩니다. Database 자체의 장애는 있지 않는 상황이라 전체 복구에 이용하지 않고 일부 데이터의 복구에 많이 사용하며 완벽한 복구를 위해선

추가적인 운영자의 확인 작업이 필요하기도 합니다. Logical Recovery의 경우 pg_restore를 이용하여 복구가 가능하며 복구 대상 Database는 존재해야 합니다.

1) pg_restore Options

```
F(format) : 백업 포맷 선택
    p(plain) : SQL script로 백업 받은 파일
    c(custom) : custom-format으로 백업 받은 파일
    d(directory) : directory-format으로 백업 받은 백업 경로
    t(tar) : tar-format으로 백업 받은 파일

f(file) : 백업 파일명
n(schema) : Schema 지정
    ex1) -n schema1 -n schema2 -n schema3
    ex2) -n 'schema*'

N(exclude-schema) : Schema 제외
t(table) : 테이블 지정 (-t 옵션 사용 시 -n 옵션은 적용되지 않음)
    ex1) -t table1 -t table2 -t table3
    ex2) -t 'table*'

N(exclude-table) : 테이블 제외
d(dbname) : DB 지정
U(username) : 백업 시 접속할 사용자명
v(verbose) : 진행 상황 확인
```

2) Examples

```
# Recovery Database(testdb) 생성
createdb testdb

# Recovery Database 진행 (p_test Table 복구)
pg_restore -d testdb -U agens -t p_test -p 5432 -Fd /home/agens/backup/logical_backup -v
```

7장. Database Replication

AgensSQL은 Database의 고효율성과 안정성을 위하여 데이터베이스를 실시간으로 복제하여 대기용 데이터베이스 서버를 구성 할 수 있습니다. Database Replication은 구성방식에 따라 전체 Database의 모든 변경 사항을 실시간으로 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-Write 운영이 가능합니다. 따라서 HA 및 백업의 용도로 이용하기 보단 데이터 실시간 공유에 활용이 적합합니다. Subscription이 동작하는 Table에 복제서버에서의 변경이 가해진 경우 Error와 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

- 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 Replication 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 | username | 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 레코드 단위로 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 완료한 뒤 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 Replication 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 설정 파일 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	username	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			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)

8장. Database Role 관리

PostgreSQL은 **Role**이라는 개념을 사용하여 데이터베이스 액세스 권한을 관리합니다. **Role**은 데이터베이스가 설정된 방법에 따라 데이터베이스 사용자 또는 데이터베이스 사용자 그룹으로 생각할 수 있습니다. **Role**의 개념은 "사용자" 및 "그룹"의 개념을 포함합니다. PostgreSQL 8.1 이전 버전에서 사용자와 그룹은 별개의 엔티티였지만 이제는 **Role**만 있습니다. 모든 **Role**은 사용자, 그룹 또는 양쪽 모두로 작용할 수 있습니다. **LOGIN** 속성이 있는 **Role**은 "Database User"와 동일합니다.

8.1 Role 속성

데이터베이스 **role**은 권한을 정의하는 속성이 다수 있으며, 클라이언트 인증 시스템과 인터랙션합니다.

- **로그인 권한**

데이터베이스 연결을 위한 초기 **role** 이름으로 **LOGIN** 속성이 있는 **role**만 사용할 수 있습니다. **LOGIN** 속성이 있는 **role**은 “데이터베이스 사용자”와 동일한 것으로 간주될 수 있다. 로그인 권한이 있는 **role**을 생성하려면 다음 중 하나를 사용해야 합니다.

```
db=# CREATE ROLE name LOGIN;  
db=# CREATE USER name;
```

(**CREATE USER**는 디폴트로 **LOGIN** 권한을 준다는 점에서 **CREATE ROLE**과 다릅니다.)

- **Superuser 상태**

데이터베이스 **superuser**는 로그인 권한을 제외한 모든 권한 검사를 건너뜁니다. 이 권한은 위험하며, 무심코 사용해서는 안 됩니다. 작업 대부분은 **superuser** 이외의 다른 **role**로 수행하는 것이 좋습니다. 새 데이터베이스 **superuser**를 생성하려면 **CREATE ROLE name SUPERUSER**를 사용해야 합니다. **superuser**인 **role**로 이것을 수행해야 합니다.

- **데이터베이스 생성**

데이터베이스를 생성하려면 권한이 명시적으로 **role**에 주어져야 합니다(모든 권한

검사를 건너 뛰는 superuser인 경우 제외). 이러한 role을 생성하려면 CREATE ROLE name CREATEDB를 사용해야 합니다.

- **role** 생성

role을 추가적으로 생성하려면 권한이 명시적으로 role에 주어져야 합니다(모든 권한 검사를 건너 뛰는 superuser인 경우 제외). 이러한 role을 생성하려면 CREATE ROLE name CREATEROLE을 사용해야 합니다. CREATEROLE 권한이 있는 role은 다른 role을 변경 및 삭제할 수 있으며, 멤버십을 부여 또는 취소할 수도 있습니다. 단, superuser role의 멤버십을 생성, 변경(alter), 삭제 또는 변경(change)하려면 superuser 상태가 필요합니다. CREATEROLE로는 부족합니다.

- 복제 초기화

streaming replication을 초기화하려면 권한이 명시적으로 role에 주어져야 합니다(모든 권한 검사를 건너 뛰는 superuser인 경우 제외). streaming replication에 사용되는 role은 항상 LOGIN 권한이 있어야 합니다. 이러한 role을 생성하려면 CREATE ROLE name REPLICATION LOGIN을 사용해야 합니다.

- 패스워드

패스워드는 데이터베이스에 연결할 때 사용자가 패스워드를 입력해야 하는 클라이언트 인증 방법인 경우에만 중요합니다. password 및 md5 인증 방법은 패스워드를 이용합니다. 데이터베이스 패스워드는 운영 체제 사용자 패스워드와 구분됩니다. role 생성 시 패스워드 지정은 CREATE ROLE name PASSWORD 'string'을 사용해야 합니다.

8.2 Role 멤버십

권한 관리의 편의상 사용자를 그룹으로 묶는 것이 편리할 수 있습니다. 이렇게 하면 권한을 그룹 단위로 부여하거나 취소할 수 있습니다. 그룹을 나타내는 role을 생성한 다음, 그룹 role의 멤버십을 개별 사용자 role에 부여하면 됩니다. 그룹 role을 설정하려면 먼저 role을 생성해야 합니다. 일반적으로 그룹으로 사용되는 role은 LOGIN 속성이 없으며, 원하면 설정은 할 수 있습니다.

그룹 role이 존재하는 경우 GRANT 및 REVOKE 명령을 사용하여 멤버를 추가 및 삭제할 수 있습니다.

```
db=# GRANT group_role TO role1, ... ;
db=# REVOKE group_role FROM role1, ... ;
```

다른 그룹 role에도 멤버십을 부여할 수 있습니다.(그룹 role과 비 그룹 role 사이에 실제로는 구분이 없음). 데이터베이스는 순환식 멤버십 루프의 설정을 허용하지 않습니다. 또한, role의 멤버십을 PUBLIC에 부여하는 것도 허용하지 않습니다.

그룹 role의 멤버는 두 가지 방법으로 role의 권한을 사용할 수 있습니다.

첫째, 그룹의 모든 멤버는 명시적으로 SET ROLE을 수행하여 일시적으로 그룹 role이 “됩니다”. 이 상태에서 데이터베이스 세션은 원래의 로그인 role이 아닌 그룹 role에 대한 액세스 권한을 가지며, 생성된 데이터베이스 오브젝트는 로그인 role이 아닌 그룹 role이 소유하는 것으로 간주됩니다.

둘째, INHERIT 속성이 있는 멤버 role은 해당 role에서 상속된 모든 권한을 비롯하여 멤버로서 role의 권한을 자동으로 갖습니다. 예를 들면, 다음을 실행했다고 가정하면,

```
db=# CREATE ROLE joe LOGIN INHERIT;
db=# CREATE ROLE admin NOINHERIT;
db=# CREATE ROLE wheel NOINHERIT;
db=# GRANT admin TO joe;
db=# GRANT wheel TO admin;
```

joe role로 연결한 직후에 데이터베이스 세션은 joe에 직접 부여된 권한 외에도, joe는 admin의 권한을 “상속 받기” 때문에 admin에 부여된 권한도 사용합니다. 그러나, joe가 간접적으로 wheel의 멤버지만 이 멤버십은 NOINHERIT 속성을 갖는 admin을 통한 것이므로 wheel에 부여된 권한은 사용할 수 없습니다.

```
db=# SET ROLE admin;
```

위 명령을 실행하면, 세션은 admin에 부여된 이러한 권한만 사용하고 joe에 부여된 권한은 사용하지 않습니다.

```
db=# SET ROLE wheel;
```

위 명령을 실행하면, 세션은 wheel에 부여된 이러한 권한만 사용하고 joe 또는 admin에 부여된 권한은 사용하지 않습니다. 원래의 권한 상태는 다음 중 하나를 사용하면 복원됩니다.

```
db=# SET ROLE joe;
db=# SET ROLE NONE;
db=# RESET ROLE;
```

role 속성 LOGIN 및 SUPERUSER, CREATEDB, CREATEROLE은 특수한 권한으로 생각될 수 있지만 데이터베이스 오브젝트의 일상적인 권한으로 상속되지 않습니다. 속성을 사용하려면 이러한 속성 중 하나를 보유한 특정 role에 실제로 SET ROLE을 해야 합니다. 위의 예시에 이어서, CREATEDB 및 CREATEROLE을 admin role에 부여할 수도 있습니다. 그러면, joe role로 연결하는 세션은 이러한 권한을 즉각 갖지는 못하며, SET ROLE admin을 수행한 이후에만 권한이 부여됩니다.

그룹 role을 소멸하려면 DROP ROLE을 사용해야 합니다.

```
db=# DROP ROLE name;
```

그룹 role의 멤버십이 자동 취소됩니다(단, 멤버 role은 영향을 받지 않음). 그룹 role이 소유한 오브젝트를 먼저 삭제하거나 다른 소유자에게 재할당해야 하며, 그룹 role에 부여된 권한은 취소해야 한다는 점을 유의해야 합니다.

8.3 Role 조회

PSQL에서 아래의 명령어로 Role 조회가 가능합니다.

```
postgres=# \du

          List of roles
Role name |          Attributes          | Member of
-----+-----+-----+
postgres | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
```

아래의 Catalog View를 통해 Role 조회가 가능합니다.

```
select * from pg_user ;
select * from pg_shadow ;
Select * from pg_roles;
```

8.4 Role 생성

8.4.1 Syntax

```
CREATE USER username [[ WITH ] option [ ... ]]
```

SUPERUSER | NOSUPERUSER : SUPERUSER 권한 (Default : NOSUPERUSER)

CREATEDB | NOCREATEDB : DATABASE 생성 권한 (Default: NOCREATEDB)

CREATEUSER | NOCREATEUSER : 새로운 User를 생성하는 권한 (Default: NOCREATEUSER)

INHERIT | NOINHERIT : Database의 권한을 다른 구성원들에게 상속하는 역할 (Default: INHERIT)

LOGIN | NOLOGIN : LOGIN 역할 부여 (Default: NOLOGIN)

[ENCRYPTED | UNCRYPTED] PASSWORD 'password' : 'password'를 입력하고 인증이 필요 없는 경우
옵션 생략 가능

8.4.2 Example

```
CREATE USER test1; (동일) CREATE ROLE test1 LOGIN;
```

```
CREATE USER test2 with PASSWORD 'test2';
```

```
CREATE USER test3 with PASSWORD 'test3' SUPERUSER;
```

8.5 Role 삭제

8.5.1 Syntax

```
CREATE USER username [[ WITH ] option [ ... ]]
```

8.5.2 Example

```
Drop user test1; (동일) Drop role test1;
```

```
Drop role test2, test3;
```

```
Drop role if exists test3;
```

8.6 Role 변경

8.6.1 Syntax

```
ALTER ROLE role_specification [ WITH ] option [ ... ]
```

8.6.2 Example

```
# Change a role's password
ALTER ROLE davide WITH PASSWORD 'hu8jmn3';

# Remove a role's password
ALTER ROLE davide WITH PASSWORD NULL;

# Change a password expiration date
ALTER ROLE chris VALID UNTIL 'May 4 12:00:00 2015 +1';

# Make a password valid forever
ALTER ROLE fred VALID UNTIL 'infinity';

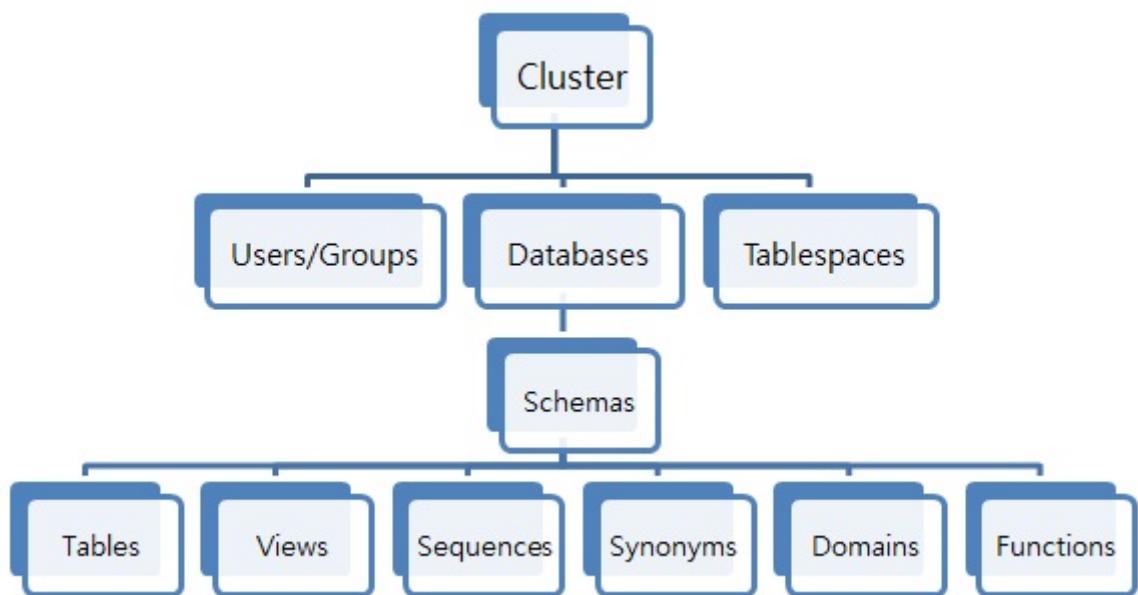
# Give a role a non-default setting
ALTER ROLE worker_bee SET maintenance_work_mem = 100000 ;

# Give a role a default setting
ALTER ROLE worker_bee RESET maintenance_work_mem ;

# Give a role a non-default, database-specific setting
ALTER ROLE fred IN DATABASE devel SET client_min_messages = DEBUG;
ALTER ROLE fred IN DATABASE devel SET client_min_messages FROM CURRENT ;
ALTER ROLE fred IN DATABASE devel RESET ALL ;
ALTER ROLE name RENAME TO new_name ;
```

9장. Database 관리

하나의 Database Cluster에는 복수 개의 Database를 생성할 수 있으며, 하나의 Database 구조는 여러 개의 Schema와 오브젝트를 소유하고 있습니다.



9.1 Database 조회

PSQL에서 아래의 명령어로 Database 조회가 가능합니다.

```

postgres=# \l
      List of databases
   Name | Owner | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+
mydb  | postgres | UTF8  | ko_KR.UTF-8 | ko_KR.UTF-8 |
postgres | postgres | UTF8  | ko_KR.UTF-8 | ko_KR.UTF-8 |
template0 | postgres | UTF8  | ko_KR.UTF-8 | ko_KR.UTF-8 | =c/postgres      +
|       |       |       |       | postgres=CTc/postgres
template1 | postgres | UTF8  | ko_KR.UTF-8 | ko_KR.UTF-8 | =c/postgres      +
|       |       |       |       | postgres=CTc/postgres

```

아래의 Catalog View를 통해 Role 조회가 가능합니다.

```
select * from pg_database ;
```

9.2 Database 생성

Database를 생성하기 위해서는 SUPERUSER이거나 CREATEDB 권한이 있는 USER이어야 합니다. SUPERUSER는 다른 USER의 DATABASE를 소유할 수 있으며, 일반 USER들은 해당 USER가 생성한 DATABASE만 소유할 수 있습니다.

9.2.1 Syntax

```

CREATE DATABASE name
[ WITH ] [ OWNER [=] user_name ]
[ TEMPLATE [=] template ]
[ ENCODING [=] encoding ]
[ STRATEGY [=] strategy ]
[ LOCALE [=] locale ]
[ LC_COLLATE [=] lc_collate ]
[ LC_CTYPE [=] lc_ctype ]
[ ICU_LOCALE [=] icu_locale ]
[ LOCALE_PROVIDER [=] locale_provider ]
[ COLLATION_VERSION = collation_version ]
[ TABLESPACE [=] tablespace_name ]
[ ALLOW_CONNECTIONS [=] allowconn ]
[ CONNECTION LIMIT [=] connlimit ]
[ IS_TEMPLATE [=] istemplate ]
[ OID [=] oid ]

```

- [[WITH] [OWNER [=] user_name] DATABASE를 소유하는 USER 지정

- [TEMPLATE [=] template] 지정한 TEMPLATE을 복제하여 미리 정의된 표준객체를 포함한 DATABASE 생성
- [ENCODING [=] encoding] DATABASE에서 사용할 ENCODING 설정 (ex : SQL_ASCII)
- [LC_COLLATE [=] lc_collate] column, index, order by 절에서 문자열에 적용되는 정렬 순서를 설정
- [LC_CTYPE [=] lc_ctype] 문자의 분류로 대/소문자, 숫자 분류를 설정
- [TABLESPACE [=] tablespace] 생성될 Database의 Default Tablespace를 지정
- [CONNECTION LIMIT [=] connlimit] Database에 동시 접속을 제한 (Default : -1(제한없음))

9.2.2 Example

```
CREATE DATABASE name;
CREATE DATABASE T_D2 OWNER BITNINE;
```

9.3 Database 삭제

9.3.1 Syntax

```
DROP DATABASE [ IF EXISTS ] name [ [ WITH ] ( option [ , ... ] ) ]
where option can be:
    FORCE
```

9.3.2 Example

```
DROP DATABASE T_D2 ;
```

9.4 Database 변경

9.4.1 Syntax

```
ALTER DATABASE name [ [ WITH ] option [ ... ] ]
where option can be:
    ALLOW_CONNECTIONS allowconn
    CONNECTION LIMIT connlimit
    IS_TEMPLATE istemplate
```

9.4.2 Example

```
# 테이블스페이스 지정(default tablespace는 pg_default)
ALTER DATABASE T_D SET TABLESPACE TBS01 ;

# DB 이름 변경
ALTER DATABASE T_D RENAME TO T_D3 ;

# DB Owner 변경
ALTER DATABASE T_D2 OWNER TO TEST4 ;

# DB Index Scan 사용 변경
ALTER DATABASE T_D3 SET ENABLE_INDEXSCAN TO OFF ;
ALTER DATABASE T_D SET ENABLE_INDEXSCAN FROM CURRENT ;

# DB Index Scan 사용 변경 RESET
ALTER DATABASE T_D RESET ENABLE_INDEXSCAN ;
ALTER DATABASE T_D RESET ALL ;
```

10장. Tablespace 관리

Tablespace를 사용하면 데이터베이스 오브젝트를 나타내는 파일을 저장할 수 있는 파일 시스템의 위치를 정의할 수 있습니다. 이를 통해 Database Cluster를 초기화한 파티션 또는 볼륨의 용량이 부족하여 확장이 불가능한 경우, 시스템을 재구성할 때까지 다른 파티션에 Tablespace를 작성하여 이를 사용할 수 있습니다. 또한 오브젝트의 사용 패턴에 따라 데이터 저장 위치를 조정하여 성능 향상을 꾀할 수 있습니다. 기본적으로 DB로 지정된 디렉토리 전체가 하나의 기본 테이블스페이스로 인식되며, 하위 디렉토리에 오브젝트 파일이 생성됩니다. 해당 경로가 pg_tblspc 디렉토리 밑에 심볼릭 링크로 걸리게 되고, 이 링크를 통해 디렉토리 파일들을 조회가 됩니다.(단 'pg_'로 시작하는 이름은 불가능)

주의할 점은 Tablespace도 Database Cluster의 필수적인 부분이기 때문에 파티션 위치가 다르더라도 백업이나 이관시 함께 수행해야 하며, HA 구성 시에도 양쪽 서버에 동일한 디렉토리를 생성 후 작업해야 합니다.

10.1 Tablespace 조회

PSQL에서 아래의 명령어로 Tablespace 조회가 가능합니다.

```
postgres=# \db+
      List of tablespaces
   Name  |  Owner  | Location | Access privileges | Options | Size | Description
-----+-----+-----+-----+-----+-----+
pg_default | postgres |          |          |          | 31 MB |
```

pg_global	postgres			636 kB
-----------	----------	--	--	--------

- pg_default : \$PGDATA/base
- pg_global : \$PGDATA/global

아래의 Catalog View를 통해 Role 조회가 가능합니다.

```
select * from pg_tablespace ;
```

10.2 Tablespace 생성

10.2.1 Syntax

```
CREATE TABLESPACE tablespace_name  
[ OWNER { new_owner | CURRENT_USER | SESSION_USER } ]  
LOCATION 'directory'  
[ WITH ( tablespace_option = value [, ... ] ) ]
```

10.2.2 Example

```
$ mkdir -p /data/TESTTBS  
  
$ chown agens:agens /data/TESTTBS  
  
# create tablespace testtbs01 owner bitnine location '/data/TESTTBS';  
  
# select oid, spcname from pg_tablespace where spcname='testtbs01' ;  
oid | spcname  
-----+-----  
27837 | testtbs01  
  
agens@TAR01 PG_10_201707211]$ pwd  
/data/TESTTBS/PG_10_201707211  
[agens@TAR01 PG_10_201707211]$ ls -IR  
.:  
total 0  
drwx-----. 2 postgres postgres 45 Dec 10 13:48 13280  
.13280:  
total 8  
-rw----- 1 postgres postgres 0 Dec 10 13:48 27838  
-rw----- 1 postgres postgres 0 Dec 10 13:48 27841  
-rw----- 1 postgres postgres 8192 Dec 10 13:48 27843
```

```
$ pwd  
$PGDATA/pg_tblspc  
  
$ ls -alR  
.:  
total 8  
drwx----- 2 postgres postgres 4096 Dec 10 13:45 .  
drwx----- 20 postgres postgres 4096 Dec 10 10:36 ..  
lrwxrwxrwx. 1 postgres postgres 11 Dec 10 13:45 27837 -> /data/TESTTBS
```

10.3 Tablespace 삭제

10.3.1 Syntax

```
DROP TABLESPACE [ IF EXISTS ] name
```

10.3.2 Example

```
DROP TABLESPACE testtbs01 ;
```

10.4 Tablespace 변경

10.4.1 Syntax

```
ALTER TABLESPACE name RENAME TO new_name  
ALTER TABLESPACE name OWNER TO { new_owner | CURRENT_USER | SESSION_USER }  
ALTER TABLESPACE name SET ( tablespace_option = value [, ...] )  
ALTER TABLESPACE name RESET ( tablespace_option [, ...] )
```

10.4.2 Example

```
# Tablespace owner 변경  
ALTER TABLESPACE tbs_ldbc OWNER TO agens ;  
  
# Tablespace 이름 변경  
ALTER TABLESPACE testtbs01 RENAME TO tbs_ldbc2 ;  
  
# Tablespace 옵션 변경 및 해제  
ALTER TABLESPACE testtbs01 SET effective_io_concurrency=1;  
ALTER TABLESPACE testtbs01 RESET effective_io_concurrency ;
```

11장. Schema 관리

데이터베이스 클러스터에는 하나 이상의 데이터베이스가 포함됩니다. Role 및 일부 다른 유형의 오브젝트는 클러스터 전체에서 공유됩니다. 서버에 연결된 클라이언트 연결은 단일 데이터베이스, 즉 연결 요청에 지정된 데이터베이스의 데이터에만 액세스 할 수 있습니다. 데이터베이스에는 하나 이상의 Schema가 포함되고, Schema에는 테이블이 포함됩니다. Schema에는 데이터 형식, 함수 및 연산자와 같은 다른 오브젝트들도 포함됩니다. 동일한 오브젝트명을 다른 Schema에서 사용해도 충돌이 발생하지 않습니다. 예를 들어, schema1과 myschema Schema 모두에 mytable이라는 테이블이 포함될 수 있습니다. Schema는 데이터베이스와 달리 엄격하게 분리되어 있지 않으므로 사용자는 권한이 있으면 연결된 데이터베이스의 모든 Schema 오브젝트에 액세스 할 수 있습니다.

Schema 사용을 선호하는 몇 가지 이유가 있습니다.

- 하나의 데이터베이스를 여러 사용자가 서로 간섭하지 않고 사용할 수 있도록 합니다.
- 관리하기 쉽도록 데이터베이스 오브젝트를 논리 그룹으로 구성합니다.
- 타사 응용 프로그램을 별도의 Schema에 넣어 다른 오브젝트명과의 충돌을 방지합니다.

Schema는 중첩할 수 없다는 점을 제외하고 운영 체제 디렉토리와 유사합니다.

11.1 Schema 생성

Schema를 작성하려면 CREATE SCHEMA 명령을 사용하십시오. Schema에 자유롭게 이름을 지정합니다. 예시는 다음과 같습니다.

```
CREATE SCHEMA myschema;
```

Schema에 오브젝트를 작성하거나 액세스하려면 Schema 이름과 테이블 이름을 점으로 구분된 이름을 작성하십시오.

```
schema.table
```

Schema에 테이블을 만드는 구문은 다음과 같습니다.

```
CREATE TABLE myshema.mytable(  
...);
```

다른 사용자가 소유하는 Schema를 만들고 싶을 수 있습니다. 이를 위한 구문은 다음과 같습니다.

```
CREATE SCHEMA schema_name AUTHORIZATION user_name;
```

Schema 이름은 생략할 수도 있으며, 이 경우 Schema 이름은 사용자 이름과 동일합니다. pg_로 시작하는 Schema 이름은 시스템에서 사용하기 위해 예약되어 있으며 사용자가 만들 수 없습니다.

11.2 Schema 삭제

빈 Schema(모든 객체가 삭제된 Schema)를 삭제하려면 다음을 수행합니다.

```
DROP SCHEMA myschema;
```

Schema의 모든 오브젝트를 포함하여 Schema를 삭제하는 구문은 다음과 같습니다.

```
DROP SCHEMA myschema CASCADE;
```

11.3 public SCHEMA

앞에서 Schema 이름을 지정하지 않고 테이블을 만들었습니다. 기본적으로 이런 테이블(및 다른 오브젝트)은 자동으로 “public”이라는 Schema에 배치됩니다. 모든 새 데이터베이스에는 이러한 Schema가 포함되어 있습니다. 따라서 다음 두 구문은 동일합니다.

```
CREATE TABLE products(...);
```

또는

```
CREATE TABLE public.products(...)
```

11.4 Schema 검색 경로

검색 경로의 첫 번째 열거된 Schema를 현재 Schema라고 합니다. 현재 Schema는 검색되는 첫 번째 Schema이며 Schema 이름을 지정하지 않고 CREATE TABLE 명령으로 테이블을 작성한 경우 새 테이블이 작성되는 Schema입니다.

현재 검색 경로를 표시하는 구문은 다음과 같습니다.

```
SHOW search_path
```

```
search path
```

```
-----
```

```
“$user”, public
```

첫 번째 요소는 현재 사용자와 이름이 같은 Schema를 검색하도록 지정합니다. 이러한 Schema가 없으면 이 항목은 무시됩니다. 두 번째 요소는 앞에서 설명한 공용 Schema를 참조합니다. 존재하는 Schema 중 검색 경로에서 처음 나타나는 Schema는 새 오브젝트가 생성되는 기본 위치입니다. 이것이 디폴트로 오브젝트가 public schema에 작성되는 이유입니다. 오브젝트가 Schema 규정 없이 다른 컨텍스트에서 참조되는 경우(테이블 변경, 데이터 변경 또는 조회 명령 등), 일치하는 오브젝트가 발견될 때까지 검색경로에서 탐색됩니다. 따라서 기본 구성에서는 정규화 되지 않은 액세스는 공용 Schema만 참조할 수 있습니다.

새 Schema를 경로에 추가하려면 다음을 수행합니다.

```
SET search_path TO myschema.public
```

그런 다음 Schema 규정 없이 테이블에 액세스합니다.

```
DROP TABLE mytable;
```

또, myschema 는 패스 내의 최초의 요소이므로, 새로운 오브젝트는 디폴트로 여기에 작성됩니다.

다음과 같이 쓸 수도 있습니다.

```
SET search_path TO myschema;
```

이렇게 하면 앞으로는 규정된 이름없이 public Schema에 액세스 할 수 없게 됩니다. public Schema는 기본적으로 존재한다는 것 외에 특별한 의미는 없습니다. 다른 Schema와 마찬가지로 삭제할 수도 있습니다.

11.5 Schema 권한

사용자는 기본적으로 소유하지 않은 Schema의 객체에 접근할 수 없습니다. 접근하려면 Schema 소유자에게 Schema USAGE권한을 부여해야 합니다. Schema의 오브젝트에 대해 작업을 수행하려면 해당 오브젝트에 따라 추가 권한이 필요할 수 있습니다.

다른 사용자의 Schema에서 오브젝트를 만들 수 있습니다. 이를 위해서는 Schema에서 CREATE권한이 부여되어야 합니다. 기본적으로 공용 Schema의 경우 모든 사용자에게 CREATE 및 USAGE 권한이 있습니다. 즉, 모든 사용자는 사용자가 연결할 수 있는 모든 데이터베이스의 공용 Schema에 오브젝트를 만들 수 있습니다.

```
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
```

첫 번째 'public'은 Schema입니다. 두 번째 'public'은 모든 사용자를 의미합니다.

11.6 System Catalog SCHEMA

각 데이터베이스에는 public 및 사용자가 정의한 Schema 외에 pg_catalog Schema가 포함되어 있습니다. 이 Schema에는 시스템 테이블과 모든 내장 데이터 유형, 함수 및 연산자가 포함됩니다. pg_catalog는 항상 검색 경로에 포함되어 있습니다. 경로에 명시적으로 나열되지 않은 경우 경로Schema를 검색하기 전에 암시적으로 검색됩니다. 이렇게 하면 기본 제공 이름을 항상 검색할 수 있습니다. 그러나 사용자 정의 이름으로 내장 이름을 덮어 쓰는 경우 pg_catalog를 경로의 끝에 명시적으로 넣을 수 있습니다. 시스템 카탈로그의 이름은 pg_로 시작하므로 이러한 이름은 사용하지 않는 것이 좋습니다.

IV. SQL 기본

12장. 데이터 정의어(DDL)

DDL은 Data Definition Language의 약자로 데이터 정의어를 말합니다. 즉, 데이터베이스를 정의하는 언어를 말하며 데이터를 생성하거나 수정, 삭제 등 데이터의 전체 골격을 결정하는 역할을 맡고 있습니다. 데이터 정의어에는 CREATE, ALTER, DROP, TRUNCATE가 있습니다. DDL의 대상은 테이블 뿐만 아니라, 데이터베이스(또는 스키마), 인덱스, 뷰 등이 될 수 있지만 관계형 데이터베이스에 데이터가 저장되는 기본 구조인 테이블 기준으로 설명드립니다.

12.1 CREATE TABLE

12.1.1 Syntax

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ]
table_name ( [
{ column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
| table_constraint
| LIKE source_table [ like_option ... ] }
[, ... ]
])
```

(더 많은 옵션이 존재하나, 자주 사용되는 부분만 기재)

관계형 데이터베이스의 테이블은 종이에 작성된 테이블과 매우 유사합니다. 테이블은 행과 열로 구성됩니다. 열의 수와 순서는 고정되어 있으며 각 열의 이름이 지정됩니다. 행의 수는 가변적입니다. 즉, 행 수는 그 시점에서 얼마나 많은 데이터가 저장되는지를 나타냅니다. SQL은 테이블의 행 순서를 보장하지 않습니다. 테이블을 읽으면 명시적으로 정렬이 요구되지 않는 한

행은 불특정 순서로 반환됩니다. 또한 SQL은 행에 고유한 식별자를 할당하지 않으므로 테이블에 정확히 동일한 행이 여러 개 있을 수 있습니다.

CREATE TABLE은 현재 데이터베이스에 데이터가 비어 있는 새 테이블을 만듭니다. 테이블은 명령을 실행하는 사용자가 소유합니다. Schema 이름이 지정되면(예: CREATE TABLE myschema.mytable ...) 지정된 Schema에 테이블이 생성되고 그렇지 않으면 현재 Schema에 생성됩니다. 임시 테이블은 특수한 Schema에 존재하므로 임시 테이블을 생성할 때 Schema 이름을 지정할 수 없습니다. 테이블의 이름은 동일한 Schema에 있는 다른 테이블, 시퀀스, 인덱스, 뷰 또는 외부 테이블의 이름과 구별되어야 합니다. 위의 varchar(80)은 최대 80자의 문자열을 저장할 수 있는 데이터 형식을 지정합니다. int는 일반적인 정수용의 형태이고 real은 단정밀도 부동 소수점 숫자를 저장하는 형식입니다. date는 이름에서 알 수 있듯이 날짜입니다.

AgensSQL은 표준 SQL의 데이터형, int, smallint, real, double precision, char(N), varchar(N), date, time, timestamp나 interval을 지원합니다. 또한 일반적인 유ти리티를 위한 유형과 고급 기하 데이터 유형도 지원합니다.

12.1.2 데이터 타입

각 열에는 데이터 형식이 있습니다. 데이터 유형은 열에 할당되는 값을 제한합니다. 또한 열에 저장된 데이터에 의미가 할당되어 데이터를 계산에 사용할 수 있습니다. 예를 들어, 수치형과 선언된 열은 임의의 텍스트 캐릭터 라인은 받아들이지 않습니다. 그런 다음 숫자 형식의 열에 저장된 데이터를 산술 계산에 사용할 수 있습니다. 대조적으로 문자열 유형으로 선언된 열은 거의 모든 종류의 데이터를 허용합니다. 그러나 문자열 조인과 같은 연산에는 사용할 수 있지만 산술 계산에는 사용할 수 없습니다. 자주 사용되는 데이터형으로는, 정수를 나타내는 integer, 소수도 나타낼 수가 있는 numeric, 캐릭터 라인을 나타내는 text, 일자를 나타내는 date, 시각을 나타내는 time, 그리고 일자와 시각의 양쪽 모두를 포함한 timestamp가 있습니다.

아래의 표에 PostgreSQL에 내장된 범용 데이터 유형을 모두 보여줍니다. "별칭" 열에 나열된 이름은 PostgreSQL에서 내부적으로 사용된 이름입니다. 또한 일부 내부적으로 사용되거나 더 이상 사용되지 않는 유형을 사용할 수 있지만 여기에 나열되지 않았습니다.

이름	별칭	설명
bigint	int8	부호 있는 8바이트 정수
bigserial	serial8	자동 증가 8바이트 정수
bit [(n)]		고정 길이 비트 문자열
bit varying [(n)]	varbit [(n)]	가변 길이 비트 문자열
boolean	bool	논리 부울(참/거짓)
box		평면 위의 직사각형 상자
bytea		이진 데이터("바이트 배열")

character [(n)]	char [(n)]	고정 길이 문자열
character varying [(n)]	varchar [(n)]	가변 길이 문자열
cidr		IPv4 또는 IPv6 네트워크 주소
circle		비행기의 원
date		달력 날짜(년, 월, 일)
double precision	float8	배정밀도 부동 소수점 숫자(8바이트)
inet		IPv4 또는 IPv6 호스트 주소
integer	int,int4	부호 있는 4바이트 정수
interval [fields] [(p)]		시간 범위
json		텍스트 JSON 데이터
jsonb		이진 JSON 데이터, 분해
line		비행기 위의 무한선
lseg		평면의 선분
macaddr		MAC(미디어 액세스 제어) 주소
macaddr8		MAC(Media Access Control) 주소(EUI-64 형식)
money		통화 금액
numeric [(p, s)]	decimal [(p, s)]	선택 가능한 정밀도의 정확한 숫자
path		평면의 기하학적 경로
pg_lsn		PostgreSQL 로그 시퀀스 번호
pg_snapshot		사용자 수준 트랜잭션 ID 스냅샷
point		평면 위의 기하학적 점
polygon		평면에서 닫힌 기하학적 경로
real	float4	단정밀도 부동 소수점 숫자(4바이트)
smallint	int2	부호 있는 2바이트 정수
smallserial	serial2	자동 증가 2바이트 정수
serial	serial4	자동 증가 4바이트 정수
text		가변 길이 문자열
time [(p)] [without time zone]		시간(시간대 없음)

time [(p)] with time zone	timetz	시간대를 포함한 하루 중 시간
timestamp [(p)] [without time zone]		날짜 및 시간(시간대 없음)
timestamp [(p)] with time zone	timestamptz	시간대를 포함한 날짜 및 시간
tsquery		텍스트 검색 쿼리
tsvector		텍스트 검색 문서
txid_snapshot		사용자 수준 트랜잭션 ID 스냅샷(더 이상 사용되지 않음, 참조 pg_snapshot)
uuid		보편적으로 고유한 식별자
xml		XML 데이터

12.1.3 Default

열에 기본값을 지정할 수 있습니다. 새로 작성된 행의 일부 열에 값이 지정되어 있지 않은 경우, 이러한 공란에 각 열의 기본값으로 채워집니다. 데이터 조작 명령을 사용하여 열을 기본값으로 설정하도록 명시적으로 요청할 수도 있습니다. 명시적으로 선언된 기본값이 없으면 기본값은 NULL 값입니다. NULL값은 알 수 없는 데이터를 나타내는 것이며 일반적으로 이 방법으로 문제가 되지 않습니다. 테이블 정의에서 기본값은 열 데이터 형식 뒤에 열거됩니다.

```
CREATE TABLE products
(
    product_no integer,
    name text,
    price numeric DEFAULT 9.99
);

Column | Type | Collation | Nullable | Default
-----+-----+-----+-----+
product_no | integer |      |      |
name | text |      |      |
price | numeric |      |      | 9.99
```

기본값을 표현식으로 사용할 수 있으며, 기본값이 삽입될 때마다 적용됩니다. 흔한 예로, timestamp 열이 삽입 시간으로 설정되도록 열은 기본 CURRENT_TIMESTAMP를 가질 수 있습니다. 또 다른 예는 각 행에 "번호"를 할당하는 경우입니다. PostgreSQL에서는, 일반적으로 이하와 같이 기술하는 것으로 생성됩니다.

```
CREATE TABLE products
( product_no SERIAL,
... );
```

12.1.4 Constraint

Constraint은 가장 범용적인 제약 유형입니다. 이를 사용하여 특정 컬럼의 값이 논리 값의 표현식을 충족하도록(실제 값) 지정 할 수 있습니다. 예를 들어, 제품 가격을 반드시 양수로 만들려면 다음을 수행하십시오.

```
CREATE TABLE products
( product_no integer,
  name text,
  price numeric CHECK (price > 0)
);
```

따라서 제약 조건 정의는 기본값 정의와 마찬가지로 데이터 유형 뒤에 옵니다. 기본값과 제약 조건은 임의의 순서로 열거할 수 있습니다. 검사 제약은 CHECK 키워드 뒤에 오는 괄호로 묶인 표현식으로 구성됩니다. 검사 제약 조건 표현식에는 제한된 열이 포함되어야 합니다.

제약 조건에 개별적으로 이름을 지정할 수도 있습니다. 이름을 지정하면 오류 메시지를 쉽게 이해할 수 있으며 변경하려는 제약 조건을 볼 수 있습니다. 구문은 다음과 같습니다.

```
CREATE TABLE products
( product_no integer,
  name text,
  price numeric CONSTRAINT positive_price CHECK (price > 0)
);
```

위에서 언급했듯이 명명된 제약 조건은 CONSTRAINT 키워드로 시작하여 식별자, 제약 조건 정의를 따릅니다. (이 방법으로 제약 이름을 지정하지 않으면 시스템에서 이름을 지정합니다.)

검사 제약 조건에서 여러 열을 참조할 수도 있습니다. 예를 들어 일반 가격과 할인 가격을 저장할 때 할인 가격이 일반 가격보다 낮아야 합니다.

```
CREATE TABLE products
( product_no integer,
```

```
name text,  
price numeric CHECK (price > 0),  
discounted_price numeric CHECK (discounted_price > 0),  
CHECK (price > discounted_price)  
);
```

세 번째 제약은 새로운 구문을 사용합니다. 이것은 특정 열에 추가되지 않고 쉼 표로 구분된 열 목록의 별도 항목으로 나타납니다. 열 정의 및 이러한 제약 정의는 임의의 순서로 열거할 수 있습니다.

처음 두 개의 제약을 열 제약이라고 합니다. 반대로 세 번째 제약 조건은 열 정의와 별도로 작성되므로 테이블 제약 조건이라고 합니다. 열 제약 조건을 테이블 제약 조건으로 작성할 수는 있지만 그 반대는 가능하거나 불가능할 수 있습니다. 왜냐하면 열 제약 조건은 제약 조건과 연관된 열만 참조하기 때문입니다.

위의 예는 다음과 같이 쓸 수도 있습니다.

```
CREATE TABLEproducts  
( product_no integer,  
  name text,  
  price numeric,  
  CHECK (price > 0),  
  discounted_price numeric,  
  CHECK (discounted_price > 0),  
  CHECK (price > discounted_price)  
);
```

또는 다음과 같이 사용할 수 있습니다.

```
CREATE TABLEproducts  
( product_no integer,  
  name text,  
  price numeric CHECK (price > 0),  
  discounted_price numeric,  
  CHECK (discounted_price > 0 AND price > discounted_price)  
);
```

어떤 방법을 사용해도 무관합니다.

열 제약 조건과 마찬가지로 테이블 제약 조건에 이름을 지정할 수 있습니다.

```
CREATE TABLE products
( product_no integer,
  name text,
  price numeric,
  CHECK (price > 0),
  discounted_price numeric,
  CHECK (discounted_price > 0),
  CONSTRAINT valid_discount CHECK (price > discounted_price) );
```

검사 제약 조건에서 검사 표현식이 참 또는 널값으로 평가되면 조건이 총족된다는 점에 유의하십시오. 대부분의 표현식은 하나의 연산항목에 대해 널(NULL)이 있으면 널(NULL)로 평가됩니다. 제약 조건은 제약 대상 열에 널(NULL) 값이 포함되는 것을 방지하지 않습니다. 컬럼이 널(NULL) 값을 포함하지 않도록 하기 위해 다음절에서 설명하는 널(null) 제약조건을 사용할 수 있습니다.

12.1.5 NOT NULL Constraint

NOT NULL 제약 조건은 열이 NULL 값을 가질 수 없도록 지정합니다. 구문의 예는 다음과 같습니다.

```
CREATE TABLE products
( product_no integer NOT NULL,
  name text NOT NULL,
  price numeric );
```

NOT NULL 제약 조건은 항상 열 제약 조건으로 설명됩니다. NOT NULL 제약은 `CHECK (column_name IS NOT NULL)`라는 검사 제약과 기능적으로는 동등하지만, PostgreSQL에서는 명시적으로 NOT NULL 제약을 작성하는 것이 보다 효과적입니다. 이와 같이 작성된 NOT NULL 제약에 명시적인 이름을 붙일 수 없는 것이 단점입니다.

하나의 열에 여러 제약 조건을 적용할 수도 있습니다. 그렇게 하려면, 차례차례로 제약을 기입합니다.

```
CREATE TABLE products
( product_no integer NOT NULL,
  name text NOT NULL,
  price numeric NOT NULL CHECK (price > 0) );
```

12.1.6 UNIQUE Constraint

고유성 제약 조건은 컬럼 또는 컬럼 그룹에 포함된 데이터가 테이블의 모든 행에서 고유한지 확인합니다. 열 제약의 경우 구문은 다음과 같습니다.

```
CREATE TABLE products
( product_no integer UNIQUE,
  name text,
  price numeric );
```

또한 테이블 제약조건의 구문은 다음과 같습니다.

```
CREATE TABLE products
( product_no integer,
  name text,
  price numeric,
  UNIQUE(product_no) );
```

컬럼 세트에 대해 고유성 제약 조건을 정의하려면 컬럼 이름을 쉼표로 구분하여 테이블 제약 조건으로 작성하십시오.

```
CREATE TABLE example
( a integer,
  b integer,
  c integer,
  UNIQUE(a, c));
```

지정된 열의 값 조합이 전체 테이블에서 고유하다는 것을 지정합니다. 반드시 열 중 하나가 고유 할 필요는 없습니다 (일반적으로 고유하지 않음).

고유성 제약에는, 통상의 방법으로 이름을 할당할 수도 있습니다.

```
CREATE TABLE products
( product_no integer CONSTRAINT must_be_different UNIQUE,
  name text,
  price numeric );
```

고유성 제약 조건을 추가하면 제한조건에 지정된 열 또는 열 그룹에 대해 고유한 B-트리 색인이 자동으로 만들어집니다. 일부 행에만 적용되는 고유성 제한을 고유성 제약조건으로 작성할 수는 없지만, 이러한 제한을 고유한 부분 색인을 작성하여 실현할 수 있습니다.

일반적으로 제약 조건이 적용되는 모든 열에 대해 동일한 값을 갖는 행이 테이블에 두 개 이상의 행을 갖는 경우 고유한 제약 조건 위반이 발생합니다. 그러나 이 비교에서 두 개의 NULL값은 결코 같은 값으로 간주되지 않습니다. 즉, 고유한 제약 조건이 있더라도 제약 조건이 있는 열 중 적어도 하나에 NULL값이 있는 여러 행을 저장할 수 있습니다. 이 동작은 표준 SQL을 준수하지만 이 규칙을 따르지 않는 SQL 데이터베이스가 있으므로 응용프로그램을 개발할 때는 주의해야 합니다.

12.1.7 PRIMARY KEY

기본 키 제약 조건은 컬럼 또는 컬럼 그룹이 테이블의 행을 고유하게 식별하는 것으로 사용할 수 있음을 의미합니다. 이렇게 하려면 값이 고유하고 NULL값이 아니어야 합니다. 즉, 다음 두 테이블 정의는 동일한 데이터를 허용합니다.

```
CREATE TABLE products
( product_no integer UNIQUE NOT NULL,
  name text,
  price numeric);
```

```
CREATE TABLE products
( product_no integer PRIMARY KEY,
  name text,
  price numeric );
```

기본 키도 여러 열을 설정할 수 있으며 구문은 고유성 제약 조건과 유사합니다.

```
CREATE TABLE example
( a integer,
  b integer,
  c integer,
  PRIMARY KEY(a, c) );
```

기본 키를 추가하면 기본 키에 지정된 열 또는 열 그룹에 대해 고유한 B-트리 색인이 자동으로 만들어집니다. 또한 열에 NOT NULL 표시가 적용됩니다.

하나의 테이블은 최대 하나의 기본 키를 가질 수 있습니다. 관계형 데이터베이스 이론에서는, 모든 테이블에 기본 키가 하나 필요합니다. 이 규칙은 PostgreSQL에서는 강제되지 않지만 대부분의 경우에는 이것을 따르는 것이 좋습니다.

기본 키는 문서화 및 클라이언트 응용 프로그램 모두에서 유용합니다. 예를 들어, 행 값을 변경할 수 있는 GUI 응용프로그램이 행을 고유하게 식별하려면 아마 테이블의 기본 키를 알아야 합니다. 다른 기본 키가 선언될 때 데이터베이스 시스템이 이를 사용하는 몇 가지 장면이 있습니다. 예를 들어, 외래 키가 테이블을 참조하면 기본 키가 기본 대상 열입니다.

12.1.8 FOREIGN KEY

외래 키 제약 조건은 열 (또는 열 그룹)의 값이 다른 테이블의 행 값과 일치해야 함을 지정합니다. 이렇게 하면 연관된 두 테이블의 참조 무결성이 유지됩니다.

앞에서 예로 사용했던 `products` 테이블을 생각해 보겠습니다.

```
CREATE TABLE products
( product_no integer PRIMARY KEY,
  name text,
  price numeric );
```

또한 이러한 제품에 대한 주문을 저장하는 테이블을 작성했다고 가정합니다. 이 주문의 주문 테이블에는 실제로 존재하는 제품의 주문만 저장하고 싶습니다. 따라서 `products` 테이블을 참조하는 `orders` 테이블에 외래키 제약 조건을 정의합니다.

```
CREATE TABLE orders
( order_id integer PRIMARY KEY,
  product_no integer REFERENCES products (product_no),
  quantity integer );
```

이제 `products` 테이블에 존재하지 않는 NULL이 아닌 `product_no` 항목을 사용하여 주문을 작성할 수 없습니다.

이 경우 `orders` 테이블을 참조 테이블, `product` 테이블을 기준 테이블이라고 합니다. 마찬가지로 참조 열과 기준 열도 있습니다.

위의 명령은 다음과 같이 줄일 수 있습니다.

```
CREATE TABLE orders
( order_id integer PRIMARY KEY,
```

```
product_no integer REFERENCES products,  
quantity integer );
```

열 목록이 없기 때문에 참조 테이블의 기본 키가 참조 열로 사용됩니다.

외래 키에서도 열 그룹을 제한하거나 참조할 수 있습니다. 이것도 테이블 제약의 형태로 기술할 필요가 있습니다.

```
CREATE TABLE t1  
( a integer PRIMARY KEY,  
  b integer,  
  c integer,  
  FOREIGN KEY(b, c) REFERENCES other_table (c1, c2) );
```

물론 제한된 열 수와 유형은 참조된 열의 수와 유형이 일치해야 합니다.

외래 키 제약에는, 통상의 방법으로 이름을 할당할 수도 있습니다. 테이블은 여러 개의 외래 키 제약 조건을 가질 수 있습니다. 이것은 테이블 간의 다 대 다 관계를 구현하는 데 사용됩니다.

예를 들어, 제품 및 주문에 대한 각 테이블이 있는 경우 여러 제품에 걸쳐있는 주문을 가능하게 하려는 경우(위의 예의 구조에서는 불가능함) 다음 테이블 구조를 사용할 수 있습니다.

```
CREATE TABLE products
(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

CREATE TABLE orders
(
    order_id integer PRIMARY KEY,
    shipping_address text,
    ...
);

CREATE TABLE order_items
(
    product_no integer REFERENCES products,
    order_id integer REFERENCES orders,
    quantity integer,
    PRIMARY KEY(product_no, order_id)
);
```

마지막 테이블에서 기본 키와 외래 키가 겹치는 것에 주목하십시오.

외래 키가 제품과 연관되지 않은 주문을 작성할 수 없는 것은 이미 설명한 것입니다. 그러나 한 주문에서 참조한 제품이 주문 후 삭제되면 어떻게 될 것입니다. SQL에서는 이러한 경우도 취급할 수가 있습니다. 직관적으로 몇 가지 옵션이 있습니다.

- 참조된 항목을 삭제할 수 없음
- 기본 키도 함께 삭제

구체적인 예로서, 위의 예제의 다 대다 관계에 다음 정책을 구현해 봅시다. (order_items를 통해) 주문에서 참조된 상태로 제품을 삭제하려고 하면 이 작업을 수행할 수 없습니다. 주문이 삭제되면 주문 항목도 삭제됩니다.

```
CREATE TABLE products
(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

CREATE TABLE orders
(
    order_id integer PRIMARY KEY,
    shipping_address text,
    ...
);

CREATE TABLE order_items
(
    product_no integer REFERENCES products ON DELETE RESTRICT,
    order_id integer REFERENCES orders ON DELETE CASCADE,
    quantity integer,
    PRIMARY KEY(product_no, order_id)
);
```

DELETE RESTRICT와 DELETE CASCADE라는 두 가지는 가장 일반적인 옵션입니다. RESTRICT는 참조된 행이 삭제되는 것을 방지합니다. NO ACTION은 제약 조건이 검사될 때 참조행이 여전히 존재하는 경우 오류가 있음을 의미합니다. 이것은 아무것도 지정하지 않은 경우의 기본 동작입니다. CASCADE는 참조된 행이 삭제될 때 참조할 행도 삭제되도록 지정합니다. 다른 두 가지 옵션, SET NULL 및 SET DEFAULT가 있습니다. 이들은 참조 행이 삭제될 때 참조행의 참조 열이 각각 NULL 또는 각 열의 기본값으로 설정됩니다. 이것들은 제약을 지키는 것을 면제하지 않는다는 점에

유의하십시오. 예를 들어, 조작에 SET DEFAULT를 지정해도 기본값이 외래키 제약 조건을 충족시키지 않으면 조작이 실패합니다.

ON DELETE와 비슷하여 피 참조열이 변경(업데이트)되었을 때 호출되는 ON UPDATE도 있습니다. 이것들이 할 수 있는 액션은 같습니다. 이 경우 CASCADE는 참조된 열의 업데이트된 값이 참조 행에 복사됩니다.

통상적으로 참조 행은 그 참조열의 어느쪽이든 NULL의 경우는 외래키 제약을 채울 필요가 없습니다. MATCH FULL이 외래키 선언에 추가되면 참조열이 모두 null인 경우에만 참조행이 제약 조건을 충족하는 것을 피할 수 있습니다 (즉, NULL과 NULL이 아닌 조합은 MATCH FULL 제약 조건에 위반이 보장됩니다). 참조 행이 외래키 제약 조건을 충족시키지 못할 가능성을 제거하려면 참조열을 NOT NULL로 선언하십시오.

외래 키는 기본 키이거나 고유성 제약 조건을 구성하는 열을 참조해야 합니다. 이는 참조가 항상(기본 키 또는 고유 제약 조건의 기초가 됨) 인덱스를 가짐을 의미합니다. 따라서 참조 행과 일치하는 행이 있는지 확인하는 것이 효율적입니다. 참조된 테이블의 행 DELETE 또는 참조된 행의 UPDATE는 이전 값과 일치하는 행에 대한 참조 테이블을 스캔해야 하므로 참조 행에 인덱스를 추가하는 것이 일반적으로 좋은 생각입니다. 이것은 항상 필요한 것은 아니며, 인덱스의 방법에는 많은 선택사항이 있으므로, 외래 키 제약의 선언에서는 참조 열의 인덱스가 자동적으로 만들어지는 것은 아닙니다.

12.1.9 시스템 열

모든 테이블에는 시스템에 의해 정의된 여러 시스템 열이 있습니다. 따라서 시스템 열의 이름을 사용자 정의 열의 이름으로 사용할 수 없습니다.

tableoid

이 행을 포함하는 테이블의 OID입니다. 이 열은 특히 상속 계층 구조에서 선택 쿼리에 유용합니다 (5.10 참조). 이 열이 없으면 어떤 테이블에서 그 행이 왔는지 알기 어렵기 때문입니다. tableoid를 pg_class의 oid 열에 조인하여 테이블 이름을 얻을 수 있습니다.

xmin

이 행 버전의 삽입 트랜잭션의 식별 정보(트랜잭션 ID)입니다. (행 버전은 행의 개별 상태입니다. 행이 업데이트 될 때마다 동일한 논리 행에 대한 새 행 버전이 만들어집니다.)

cmin

삽입 트랜잭션(0부터 시작)의 명령 식별자입니다.

xmax

삭제 트랜잭션의 식별 정보(트랜잭션 ID)입니다. 삭제되지 않은 행 버전에서는 0입니다. 보이는 행 버전에서 이 열이 0이 아닌 경우가 있습니다. 이것은 일반적으로 삭제 트랜잭션이 아직 커밋되지 않았거나 삭제 시도가 룰백되었음을 의미합니다.

cmax

삭제 트랜잭션 내의 명령 식별자 또는 0입니다.

ctid

테이블 내의 행 버전의 물리적 위치를 나타냅니다. ctid는 행 버전을 신속하게 찾을 수 있지만 행 ctid는 VACUUM FULL로 업데이트하거나 이동할 때 변경됩니다. 따라서 ctid는 장기 행 식별자로 사용할 수 없습니다. 논리 행을 식별하려면 기본 키를 사용해야 합니다.

12.1.10 Inherit

PostgreSQL은 데이터베이스 설계자에게 편리한 테이블 상속을 구현합니다.

도시(cities)의 데이터 모델을 만들려고 한다고 가정합니다. 각 주에는 많은 도시가 있지만, 수도(capitals)는 1개입니다. 어느 주에 대해서도 수도를 신속하게 검색하고 싶습니다. 이것은 두 개의 테이블을 작성하여 얻을 수 있습니다. 하나는 수도의 테이블이고 다른 하나는 수도가 아닌 도시의 테이블입니다. 그러나 수도인지 여부에 관계없이 도시에 대한 데이터를 질의하고 싶을 때 상속은 이 문제를 해결할 수 있습니다. cities에서 상속되는 capitals 테이블을 정의합니다.

```
CREATE TABLE cities (
    name    text,
    population float,
    elevation int
```

```
);  
  
CREATE TABLE capitals (  
    state  char(2)  
) INHERITS (cities);
```

이 경우, `capitals` 테이블은 그 부모 테이블인 `cities` 테이블의 열을 모두 상속합니다. 수도는 하나의 추가 열 상태를 가지며 주를 표현합니다.

PostgreSQL에서 1개의 테이블은 0 이상의 테이블로부터 상속하는 것이 가능합니다. 또한 쿼리는 테이블의 모든 행 또는 테이블의 모든 행과 상속된 테이블의 모든 행을 참조할 수 있습니다. 후자가 기본 동작입니다. 예를 들어, 다음 문의는 500 피트보다 높은 고도에 위치한 모든 도시의 이름을 수도를 포함하여 검색합니다.

```
SELECT name, elevation  
FROM cities  
WHERE elevation > 500;
```

이 쿼리는 다음 결과를 출력합니다.

name		elevation
Las Vegas		2174
Mariposa		1953
Madison		845

다음 쿼리는 500 피트보다 높은 고도에 위치한 모든 도시를 검색합니다.

```
SELECT name, elevation  
FROM ONLY cities  
WHERE elevation > 500;
```

name		elevation
Las Vegas		2174
Mariposa		1953

여기서 **ONLY** 키워드는 쿼리가 `cities` 테이블만을 대상으로 하고 `cities` 이하 상속 계층에 있는 테이블은 대상으로 하지 않음을 의미합니다. 지금까지 논의한 많은 명령(`SELECT`, `UPDATE` 및 `DELETE`)이 **ONLY** 키워드를 지원합니다.

또한 명시적으로 자손 테이블이 포함되어 있음을 나타내기 위해 테이블 이름 뒤에 `*`을 쓸 수도 있습니다.

```
SELECT name, elevation  
  FROM cities*  
 WHERE elevation > 500;
```

*의 기능은 항상 디폴트이기 때문에 기입하지 않아도 됩니다. 그러나 이 구문은 기본값을 변경할 수 있었던 이전 릴리스와의 호환성을 위해 여전히 지원하고 있습니다.

특정 행이 어떤 테이블에서 온 것인지 알고 싶을 수도 있습니다. 각 테이블에는 `tableoid`라는 원본 테이블을 나타내는 시스템 열이 있습니다.

```
SELECT c.tableoid, c.name, c.elevation  
  FROM cities  
 WHERE c.elevation > 500;
```

출력은 다음과 같습니다.

tableoid		name		elevation
139793		Las Vegas		2174
139793		Mariposa		1953
139798		Madison		845

`pg_class`과 결합하면 테이블의 실제 이름을 알 수 있습니다.

```
SELECT p.relname, c.name, c.elevation  
  FROM cities c, pg_class p  
 WHERE c.elevation > 500 AND c.tableoid=p.oid;
```

출력은 다음과 같습니다.

relname		name		elevation
cities		Las Vegas		2174

```
cities | Mariposa | 1953
```

```
capitals | Madison | 845
```

동일한 효과를 얻는 또 다른 방법은 별칭 `regclass`를 사용하여 테이블의 OID를 기호적으로 표시하는 것입니다.

```
SELECT c.tableoid::regclass, c.name, c.elevation  
FROM cities c  
WHERE c.elevation > 500;
```

상위 테이블의 검사 제약 조건과 비 널 제약 조건은 `NO INHERIT` 절에 명시적으로 지정되지 않는 한 자식 테이블에 자동으로 상속됩니다. 다른 유형의 제약(유일성 제약, 기본 키, 외래 키 제약)은 상속되지 않습니다.

테이블은 하나 이상의 상위 테이블에서 상속 가능합니다. 이 경우 테이블은 상위 테이블에 정의된 열의 합이 됩니다. 자식 테이블에서 선언된 열은 이러한 열에 추가됩니다. 부모 테이블에 동일한 이름의 열이 있거나 부모 테이블과 자식 테이블에 같은 이름의 열이 있으면 열이 "통합"되어 하위 테이블에서 단 하나의 열이됩니다. 통합하려면 열은 동일한 데이터 형식을 가져야 합니다. 다른 데이터 유형의 경우 오류가 발생합니다. 상속 가능한 검사 제약과 비 `NONE` 제약은 유사한 방식으로 통합됩니다. 검사 제약 조건은 이름이 같은 경우 통합되며 조건이 다른 경우 통합에 실패합니다.

테이블 상속은 일반적으로 `CREATE TABLE` 문의 `INHERITS` 절을 사용하여 하위 테이블을 만들 때 설정됩니다. 그 밖에도, 호환성을 가지는 방법으로 정의 끝난 테이블에 새롭게 부모-자식 관계를 붙일 수도 있습니다. 이것은 `ALTER TABLE`의 `INHERIT` 형식을 사용합니다. 이를 위해 새 하위 테이블에는 상위 테이블과 이름이 같은 열이 있어야 하며 해당 열의 형식은 동일한 데이터 형식이어야 합니다. 또한 상위 테이블과 동일한 이름과 동일한 표현식에 대한 검사 제한 조건이 있어야 합니다. `ALTER TABLE`의 `NO INHERIT` 형식을 사용하여 마찬가지로 상속 관계를 하위 테이블에서 제거 할 수 있습니다. 이러한 상속 관계의 동적 추가, 동적 삭제는 상속 관계를 테이블 분할에 사용하는 경우에 유용합니다.

나중에 자식 테이블로 만들 예정인 호환성을 가진 테이블을 쉽게 만드는 방법 중 하나는 `CREATE TABLE`에서 `LIKE` 절을 사용하는 것입니다. 이렇게하면 원래 테이블과 동일한 열을 가진 새 테이블이 만들어집니다. 새 하위 테이블이 항상 상위 테이블과 일치하는 제약 조건을 가지며 호환 가능한 것으로 간주되도록 원래 테이블에 `CHECK` 제약 조건이 있으면 `LIKE`에 `INCLUDING CONSTRAINTS` 옵션을 지정해야합니다.

하위 테이블이 있으면 부모 테이블을 삭제할 수 없습니다. 또한 하위 테이블에서 상위 테이블에서 상속된 열이나 검사 제약 조건을 삭제하거나 변경할 수 없습니다. 테이블과 모든

하위 테이블을 삭제하려면 CASCADE 옵션을 사용하여 상위 테이블을 삭제하는 것이 쉬운 방법입니다.

ALTER TABLE은 컬럼 데이터 정의 및 점검 제한조건의 변경을 상속 계층 구조의 테이블에 알립니다. 다시, 다른 테이블에 종속된 컬럼의 삭제는 CASCADE 옵션을 사용할 때만 가능합니다. ALTER TABLE은 중복 열의 통합 및 거부에 대해 CREATE TABLE 중에 적용되는 규칙을 따릅니다.

12.2 ALTER TABLE

ALTER TABLE 구문은 테이블을 수정할 수 있는 명령문입니다. 테이블을 만든 후에 실수를 발견하거나 응용 프로그램 요구 사항이 변경된 경우 테이블을 삭제하고 다시 만들 수 있습니다. 그러나 테이블에 데이터가 이미 입력되어 있거나 테이블이 다른 데이터베이스 오브젝트 (예 : 외래 키 제약 조건)에서 참조되는 경우 이는 좋은 방법이 아닙니다. 따라서 기존 테이블을 변경하기 위한 일련의 명령을 제공합니다. 테이블의 데이터를 변경하는 개념이 아닌 테이블의 정의와 구조를 변경하는 데 중점을 둡니다.

12.2.1 Syntax

ALTER TABLE 테이블이름

```
[ADD 속성 이름 데이터타입] -- 컬럼 추가  
[DROP COLUMN 속성 이름] -- 컬럼 삭제  
[ALTER COLUMN 속성 이름 데이터타입] -- 컬럼 수정  
[ALTER COLUMN 속성 이름 데이터타입 [NULL | NOT NULL]] -- 컬럼 수정 (널 허용/비허용)  
[ADD PRIMARY KEY(속성 이름)] -- 기본키 추가  
[[ADD | DROP] 제약이름] -- 제약조건 추가 또는 삭제
```

12.2.2 ALTER TABLE ADD

테이블에 컬럼을 추가하는 명령문입니다. 한번의 ADD 명령으로 여러 개의 컬럼 추가 가능하고, 하나의 컬럼만 추가하는 경우에는 괄호를 사용하면 됩니다. 추가된 컬럼은 테이블의 마지막 부분에 생성되며 사용자가 컬럼의 위치를 지정할 수 없습니다. 그러나 추가된 컬럼에 기본 값은 지정할 수 있습니다. 기존 데이터가 존재하면 추가된 컬럼 값은 NULL로 입력되고, 새로 입력되는 데이터에 대해서만 기본 값이 적용됩니다.

```
ALTER TABLE '테이블 명' ADD COLUMN '추가하려는 컬럼 명' '컬럼 데이터 타입';
ALTER TABLE products ADD COLUMN description text;
```

다음 구문을 사용하면 열 제한 조건도 동시에 정의할 수 있습니다.

```
ALTER TABLE products ADD COLUMN description text CHECK(description<>"");
ALTER TABLE products ADD CHECK (name <> "");
ALTER TABLE products ADD CONSTRAINT some_name UNIQUE (product_no);
ALTER TABLE products ADD FOREIGN KEY(product_group_id) REFERENCES product_groups;
```

실제로는 CREATE TABLE내의 열의 기술에 사용되고 있는 모든 옵션을, 여기에서 사용할 수 있습니다. 그러나 기본값은 주어진 제약 조건을 만족해야 합니다. 이를 충족하지 않으면 ADD가 실패합니다. 새 열에 올바르게 값을 넣은 후에 제약 조건을 추가 할 수 있습니다.

NOT NULL 제약 조건은 테이블 제약 조건으로 작성할 수 없으므로 다음 구문을 사용하여 추가합니다.

```
ALTER TABLE products ALTER COLUMN product_no SET NOT NULL;
```

제약 조건이 즉시 검사되므로 제약 조건을 추가하기 전에 테이블의 데이터가 제약 조건에 충족되어야 합니다.

12.2.3 ALTER TABLE ALTER COLUMN

ALTER COLUMN 명령문으로 새로운 기본값을 지정하거나, 해당 컬럼의 데이터 타입 변경 등을 할 수 있습니다. 변경 대상 컬럼에 데이터가 없거나 NULL값만 존재할 경우에 SIZE를 줄일 수 있습니다.

```
ALTER TABLE '테이블 명' ALTER COLUMN '해당 컬럼명' '새로운 데이터 타입';
```

열에 새 기본값을 설정하는 구문은 다음과 같습니다.

```
ALTER TABLE products ALTER COLUMN price SET DEFAULT 7.77
```

이는 테이블의 기존 행에 아무런 영향을 주지 않습니다. 향후 INSERT 명령을 위해 단순히 기본값을 변경하는 것입니다.

기본값을 삭제하려면 다음을 수행합니다.

```
ALTER TABLE products ALTER COLUMN price DROP DEFAULT;
```

이것은 기본값을 NULL로 설정하는 것과 같습니다. 따라서 정의되지 않은 기본값을 삭제해도 오류는 발생하지 않습니다. 왜냐하면 NULL 내재적 기본값이기 때문입니다.

열을 다른 데이터 유형으로 변환하는 구문은 다음과 같습니다.

```
ALTER TABLE products ALTER COLUMN price TYPE numeric(10,2);
```

열의 기본값을 새 형식으로 변환하고 동시에 열과 관련된 모든 제약 조건을 새 형식으로 변환하려고 시도합니다. 그러나 이러한 변환은 실패할 수 있으며 예상을 초과하는 결과를 초래할 수 있습니다. 유형을 변경하기 전에 해당 열에 대한 모든 제약 조건을 제거한 다음 나중에 적절하게 변경된 제약 조건을 재 정의하는 것이 가장 좋습니다.

NOT NULL 제약 조건을 삭제할 경우에는 다음을 수행하십시오.

```
ALTER TABLE products ALTER COLUMN product_no DROP NOT NULL
```

12.2.4 ALTER TABLE RENAME

테이블명을 변경할 수 있습니다.

```
ALTER TABLE '테이블 명' RENAME TO '변경하려는 테이블 명';
```

```
ALTER TABLE products RENAME TO items;
```

RENAME 명령문으로 특정 컬럼의 이름을 변경할 수 있습니다.

```
ALTER TABLE products RENAME COLUMN product_no TO product_number;
```

12.2.5 ALTER TABLE DROP

열에 있는 모든 데이터를 지웁니다. 또한 해당 열과 관련된 테이블 제한조건도 지워집니다. 그러나 열이 다른 테이블의 외래 키제약 조건으로 참조되는 경우 PostgreSQL은 암시적으로 제약 조건을 지우지 않습니다. CASCADE를 추가하면 열에 의존하는 모든 것을 지울 수 있습니다.

```
ALTER TABLE '테이블 명' DROP COLUMN '삭제할 컬럼명';
```

```
ALTER TABLE products DROP COLUMN description CASCADE;
```

제약 조건을 삭제하려면 제약 조건 이름을 알아야 합니다. 스스로 이름을 지정하면 간단합니다. 그러나 자체적으로 이름을 지정하지 않으면 시스템 생성의 이름이 할당되므로 이를 확인해야 합니다. 이를 위해 `psql`의 `\d tablename` 명령을 사용하면 편리합니다.

다른 인터페이스도 테이블의 세부사항을 조사하는 방법을 가질 수 있습니다. 제약 이름을 알면 다음 명령으로 제약을 삭제할 수 있습니다.

```
ALTER TABLE products DROP CONSTRAINT some_name;
```

열을 삭제하는 것과 마찬가지로 다른 것이 의존하는 제약 조건을 삭제하려면 `CASCADE`를 지정해야 합니다. 예를 들어, 외래 키 제약 조건은 참조되는 열의 고유 또는 기본 키 제약 조건에 따라 달립니다.

12.3 DROP TABLE

`DROP TABLE`은 데이터베이스에서 테이블을 제거합니다. 테이블 소유자, Schema 소유자 및 수퍼유저만 테이블을 삭제할 수 있습니다. 테이블을 삭제하지 않고 행 테이블을 비우려면 `DELETE` 또는 `TRUNCATE`를 사용하십시오.

`DROP TABLE` 대상 테이블에 대해 존재하는 모든 인덱스, 규칙, 트리거 및 제약 조건을 항상 제거합니다. 단, 뷰에서 참조하는 테이블이나 다른 테이블의 외래 키 제약 조건을 삭제하려면 `CASCADE`를 지정해야 합니다. (`CASCADE` 종속 뷰를 완전히 제거하지만 외래 키의 경우 외래 키 제약 조건만 제거하고 다른 테이블은 완전히 제거하지 않습니다.)

12.3.1 Syntax

```
DROP TABLE [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

12.3.2 Example

```
DROP TABLE weather ;
```

12.4 TRUNCATE TABLE

TRUNCATE TABLE은 테이블을 초기화합니다. 용량이 줄어들고 인덱스 등도 모두 삭제됩니다. DELETE보다 수행 속도가 훨씬 빠르나, ROLLBACK을 사용하여 데이터를 복구할 수 없습니다.

12.4.1 Syntax

```
TRUNCATE [ TABLE ] [ ONLY ] name [ * ] [, ... ]
[ RESTART IDENTITY | CONTINUE IDENTITY ] [ CASCADE | RESTRICT ]
```

12.4.2 Example

```
TRUNCATE TABLE weather ;
```

12.5 주요 오브젝트 DDL

12.5.1 VIEW

View는 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 이름을 가지는 가상 테이블입니다. 저장장치 내에 물리적으로 존재하지 않지만 사용자에게 실재하는 것처럼 간주됩니다. 주로 데이터 보정작업, 처리과정 시험 등 임시적인 작업을 위한 용도로 활용됩니다.

View는 Join문의 사용 최소화로 사용상의 편의성을 최대화합니다. 필요한 데이터만 뷰로 정의해서 처리할 수 있기 때문에 관리가 용이하고 명령문이 간단해진다는 장점이 있습니다. 논리적 데이터 독립성을 제공하며, 접근 제어를 통한 자동 보안이 제공됩니다.

View를 생성하는 구문은 다음과 같습니다.

```
CREATE VIEW comedy AS SELECT *
FROM film
WHERE kind='comedy';
```

film 테이블에 있던 열을 포함하는 View가 생성됩니다.

WITH [CASCADED|LOCAL] CHECK OPTION 옵션은 VIEW의 자동 업데이트를 제어합니다. 이 옵션을 지정하면 새 행이 VIEW 정의 조건을 충족하는지 확인하기 위해 VIEW 명령을 검사합니다.

- LOCAL

새 행은 VIEW에 직접 정의된 조건에 대해서만 확인됩니다. Base VIEW에 정의된 모든 조건은 확인되지 않습니다.

- CASCADED

VIEW 및 모든 Base VIEW 조건에 대해 새 행을 확인합니다. 만약 LOCAL과 CASCADED 옵션

중

선택되지 않으면 기본으로 CASCADED가 적용됩니다.

View를 삭제하는 구문은 다음과 같습니다.

```
DROP VIEW comedy;
```

CASCADE 파라미터를 이용하면 View에 의존하는 객체와 해당 객체의 의존하는 모든 객체를 자동으로 삭제합니다.

```
DROP VIEW comedy CASCADE;
```

12.5.2 INDEX

인덱스는 주로 데이터베이스의 성능을 향상시키는데 사용됩니다. CREATE INDEX는 지정된 relation의 지정된 열에 대한 인덱스를 구성합니다. 인덱스의 키 필드는 열 이름 또는 괄호 안에 작성된 표현식으로 지정됩니다. 인덱스 방법이 다중 열 인덱스를 지원하는 경우 여러 필드를 지정할 수 있습니다. PostgreSQL은 인덱스 메서드 B-tree, GiST, SP-GiST, GIN 및 BRIN을 제공합니다.

WHERE 절이 있으면 부분 인덱스가 생성됩니다. 부분 인덱스는 테이블의 일부, 일반적으로 인덱싱에 더 유용한 부분에 대한 항목만 포함하는 인덱스입니다.

film 테이블의 title 열에 고유한 B-tree 인덱스를 생성하는 예는 다음과 같습니다.

```
CREATE UNIQUE INDEX title_idx ON films (title);
```

film 테이블의 title과 director와 rating 열이 포함된 고유한 B-tree 인덱스를 생성하는 예는 다음과 같습니다.

```
CREATE UNIQUE INDEX title_idx ON films (title) INCLUDE (director, rating);
```

중복 제거가 비활성화된 B-tree 인덱스를 생성하는 예는 다음과 같습니다.

```
CREATE INDEX title_idx ON films (title) WITH (deduplicate_items = off);
```

대소문자를 구분하지 않는 INDEX 검색을 허용하는 표현식에 인덱스를 생성하는 예는 다음과 같습니다.

```
CREATE INDEX ON films ((lower(title)));
```

13장. 데이터 제어어(**DCL**)

DCL은 Data Control Language의 약자로 데이터 제어어를 말합니다. 데이터 제어어는 데이터베이스에 접근하고 객체들을 사용하도록 권한을 주고 회수하는 등의 역할을 맡고 있습니다. DCL에는 COMMIT, ROLLBACK, GRANT, REVOKE 명령어가 있습니다.

13.1 COMMIT

COMMIT이란 모든 작업을 정상적으로 처리하겠다고 확정하는 명령어로 트랜잭션 과정을 종료하는 것입니다. 트랜잭션에 의해 만들어진 모든 변경 사항은 다른 사람에게 표시되며 충돌이 발생하더라도 내구성이 보장됩니다. 또한 처리 과정을 DB에 영구 저장하며 이전 데이터는 완전히 UPDATE됩니다.

```
COMMIT [ WORK | TRANSACTION ] [ AND [ NO ] CHAIN ]
```

13.2 ROLLBACK

ROLLBACK은 작업에서 문제가 발생 했을때 트랜잭션의 처리 과정에서 발생한 변경 사항을 취소하고, 트랜잭션 과정을 종료 시킵니다. 트랜잭션을 ROLLBACK하고 트랜잭션에서 발생한 모든 업데이트를 버립니다. 변경 사항을 취소 할때 마지막 commit을 완료한 시점으로 돌아갑니다. 즉, commit하여 저장한것만 복구합니다.

```
ROLLBACK [ WORK | TRANSACTION ] [ AND [ NO ] CHAIN ]
```

13.3 GRANT

오브젝트가 만들어지면 소유자가 할당됩니다. 일반적으로 소유자는 작성할 명령문을 실행하는 역할입니다. 대부분의 유형의 오브젝트의 경우, 초기 상태에서 소유자(또는 슈퍼유저)만이 해당 오브젝트를 사용할 수 있습니다. 다른 사용자가 이 오브젝트를 사용하려면 권한이 부여되어야 합니다.

다음과 같이 오브젝트에 대한 적절한 유형의 ALTER 명령을 사용하여 오브젝트에 새 소유자를 지정할 수 있습니다.

```
ALTER TABLE table_name OWNER TO new_owner;
```

슈퍼 유저는 언제든지 소유자를 변경할 수 있습니다.

일반 역할은 대상 오브젝트의 현재 소유자(또는 소유자 역할의 구성원)이고 새 소유자 역할의 구성원인 경우에만 소유자를 변경할 수 있습니다.

권한을 할당하려면 GRANT 명령을 사용하십시오.

예를 들어, joe라는 기존 역할과 accounts라는 기존 테이블이 있는 경우, 이 테이블을 갱신할 수 있는 권한을 부여하는 구문은 다음과 같습니다.

```
GRANT UPDATE ON accounts TO joe;
```

시스템의 모든 역할에 권한을 부여하려면 특수한 역할 이름인 PUBLIC을 사용할 수 있습니다. 또한 그룹 역할을 사용하면 데이터베이스에 많은 사용자가 있는 경우 권한을 관리 할 수 있습니다.

가능한 권한은 다음과 같습니다.

SELECT

테이블, 뷰, 구체화된 뷰 또는 다른 테이블처럼 보이는 객체에 대해 SELECT가 있는 열 또는 지정된 열(여러 개)에 허용합니다.

또한 COPY의 사용을 허용합니다. 이 권한은 UPDATE 또는 DELETE에서 기존 열을 참조하는 경우에도 필요합니다. 시퀀스에서 이 권한은 currval 함수를 사용할 수 있습니다. 대형 오브젝트에서 이 권한은 오브젝트를 조회할 수 있습니다.

INSERT

테이블, 뷰 등에 새 행을 INSERT할 수 있습니다. 특정 열만 INSERT 명령으로 지정하려는 경우 해당 열에 허용 할 수 있습니다. (따라서 다른 열에는 기본값이 설정됩니다). COPY FROM을 이용할 수도 있습니다.

UPDATE

테이블, 뷰 등의 열을 UPDATE할 수 있습니다. (실용적으로 쉽지 않은 UPDATE 명령에는 SELECT 권한이 필요합니다. 어떤 행을 업데이트할지 결정하거나 열에 대한 새 값을 계산하기 위해 테이블 열을 참조해야 합니다.)

대형 오브젝트에서 이 권한은 오브젝트에 쓰거나 자르는 것을 허용합니다.

DELETE

테이블, 뷰등의 열을 DELETE할 수 있습니다. (DELETE 명령에는 SELECT 권한도 필요합니다.
삭제할 행을 결정하기 위해 테이블 열을 참조해야 하기 때문입니다.)

TRUNCATE

테이블 또는 뷰의 TRUNCATE를 허용합니다.

REFERENCES

테이블 또는 테이블의 특정 열을 참조하는 외래 키 제약 조건을 만들 수 있습니다.

TRIGGER

테이블 또는 뷰에 트리거를 만들 수 있습니다.

CREATE

데이터베이스에 대해 데이터베이스에 새 Schema와 게시를 만들고 데이터베이스에 신뢰할 수 있는 확장을 만들 수 있습니다.

Schema의 경우 Schema에 새 오브젝트를 만들 수 있습니다. 기존 오브젝트의 이름을 바꾸려면 오브젝트를 소유하고 오브젝트를 포함하는 Schema에 대해 이 권한을 가져야 합니다.

테이블 스페이스에 대해서는, 그 테이블스페이스 내에 테이블, 인덱스, 임시파일을 작성하는 것을 허용해, 그 테이블 스페이스를 디폴트의 테이블 스페이스로서 가지는 데이터베이스를 작성하는 것을 허가합니다.

이 권한을 박탈해도 기존의 객체의 존재, 또는 그 배치를 변경하지 않습니다.

CONNECT

권한이 부여된 사람이 데이터베이스에 연결할 수 있습니다. (`pg_hba.conf`가 부과하는 제한 점검외에도)이 권한은 연결이 시작될 때 점검됩니다.

TEMPORARY

데이터베이스 사용 중 임시 테이블을 작성할 수 있습니다.

형식과 도메인의 경우 테이블, 함수 및 기타 Schema 오브젝트를 생성할 때 형식과 도메인을 사용할 수 있습니다. (이 권한의 주요 목적은 어떤 사용자가 특정 유형에 대한 종속성을 만들 수 있는지 제어하고 나중에 소유자가 이 유형을 변경하지 못하게 하는 것입니다.)

외부 데이터 래퍼의 경우 외부 데이터 래퍼를 사용하여 새 서버를 만들 수 있습니다.

외부 서버의 경우 해당 서버를 사용하여 외부 테이블을 만들 수 있습니다. 권한을 부여받은 사람은 서버에 연결된 사용자 매핑을 생성, 수정 및 삭제할 수 있습니다.

13.4 REVOKE

이전에 주어진 권한을 취소하려면 해당 이름의 REVOKE 명령을 사용하십시오.

```
REVOKE ALL ON accounts FROM PUBLIC;
```

일반적으로 오브젝트의 소유자(또는 수퍼유저)만 오브젝트에 대한 권한을 부여하거나 박탈할 수 있습니다. 그러나 "with grant option"을 붙이면 권한이 부여 된 사용자가 소유자와 마찬가지로 다른 사용자에게 권한을 부여 할 수 있습니다. 만약 나중에 grant 옵션이 박탈되면, 박탈된 유저로부터 (직접 혹은 권한 부여의 체인에 의해) 권한을 부여받은 유저는 모두 그 권한이 박탈됩니다.

오브젝트의 소유자는 소유한 일반 권한을 삭제하도록 선택할 수 있습니다. 예를 들어, 테이블을 읽기 전용으로 만들 수 있습니다. 그러나 소유자는 항상 모든 부여 옵션을 가진 것으로 취급됩니다. 따라서 언제든지 자신의 권한을 다시 부여할 수 있습니다.

14장. 데이터 조작어(DML)

DML은 Data Manipulation Language의 약자로 데이터 조작어를 뜻합니다. 정의된 데이터베이스에 입력된 레코드를 조회, 수정, 삭제하는 등의 역할을 합니다. 즉, 테이블에 있는 행과 열을 조작하는 언어입니다. 데이터베이스 사용자가 질의어를 통하여 저장된 데이터를 실질적으로 처리하는데 사용합니다. 데이터 조작어에는 SELECT, INSERT, UPDATE, DELETE가 있습니다.

14.1 INSERT

INSERT문을 사용하여 테이블에 행을 삽입합니다.

또한 일반적으로, 단순한 수치 이외의 정수는 예시와 같이, 작은따옴표(')로 묶어줘야 합니다.

```
INSERT INTO weather VALUES ('San Francisco', 46, 50, 0.25, '1994-11-27');
```

지금까지의 구문에서는 열의 순서를 기억해 두어야 했습니다. 다음과 같은 다른 구문을 사용하면 열목록을 명시적으로 제공할 수 있습니다.

```
INSERT INTO weather (city, temp_lo, temp_hi, prcp, date)
VALUES ('San Francisco', 43, 57, 0.0, '1994-11-29');
```

목록의 열은 원하는 순서대로 지정 할 수 있습니다. 또한 일부 열을 생략 할 수 있습니다. 예를 들어, 강수량을 모르는 경우 다음과 같이 할 수 있습니다.

```
INSERT INTO weather (date, city, temp_hi, temp_lo)
VALUES ('1994-11-29', 'Hayward', 54, 37);
```

또한 COPY를 사용하여 대량의 데이터를 일반 텍스트 파일에서 로드할 수 있습니다. COPY명령은 INSERT보다 유연하지는 않지만 일반적으로 목적에 특화되어 있기 때문에 일반적으로 더 빠릅니다.

예를 들면 다음과 같습니다.

```
COPY weather FROM '/home/user/weather.txt';
```

14.2 UPDATE

UPDATE는 조건을 만족하는 모든 행에서 지정된 열의 값을 변경합니다. 수정할 열만 SET절에서 언급하면 됩니다. 명시적으로 수정되지 않은 열은 이전 값을 유지합니다.

예시로 다음 명령을 사용하여 데이터를 수정 할 수 있습니다.

```
UPDATE weather  
SET temp_hi = temp_hi -2, temp_lo = temp_lo -2  
WHERE date > '1994-11-28';
```

14.3 DELETE

DELETE 명령을 사용하여 테이블에서 행을 삭제할 수 있습니다. **DELETE WHERE**에 지정된 테이블에서 절을 만족하는 행을 삭제합니다. 절이 WHERE 없으면 결과는 테이블의 모든 행을 삭제합니다.

예시로 다음 명령을 사용하여 테이블에서 행을 삭제할 수 있습니다.

```
DELETE FROM weather WHERE city = 'Hayward';
```

Hayward에 대한 기상데이터는 모두 삭제 되었습니다.

```
DELETEFROM tablename;
```

조건이 없으면 DELETE는 지정된 테이블의 모든 행을 삭제하고 테이블을 비웁니다. 시스템은 삭제 전엔 확인을 요구하는 일은 하지 않습니다.

14.4 SELECT

SELECT는 데이터 조회문입니다. 선택 목록과 테이블 목록 및 선택적 조건으로 나눌 수 있습니다. 예를 들어, 날씨의 모든 행을 검색하려면 아래 조회문을 보십시오.

```
SELECT * FROM weather;
```

다음 조회문도 같은 결과가 나옵니다.

```
SELECT city, temp_lo, temp_hi, prcp, date FROM weather;
```

선택 목록에는 단순한 열 참조 뿐만 아니라, 식을 지정 할 수도 있습니다.

예를 들어, 다음 조회문도 수행할 수 있습니다.

```
SELECT city, (temp_hi+temp_lo)/2 AS temp_avg, date FROM weather
```

AS 절을 사용하여 출력 열의 레이블 지정 부분에 유의하십시오.(AS 절은 생략할 수 있음).

필요한 행이 무엇인지 지정 하는 WHERE절을 추가하여 쿼리에 "조건화" 할 수 있습니다.

WHERE절은 논리(참값) 표현식을 가지며 이 논리 표현식이 참인 행만 리턴합니다. 자주 사용되는 논리 연산자(AND, OR, NOT)를 조건부로 사용할 수 있습니다.

예를 들어, 다음은 샌프란시스코에서 비오는 날씨 기상 데이터를 검색합니다.

```
SELECT* FROM weather  
WHERE city = 'San Francisco' AND prcp > 0.0;
```

쿼리 결과를 정렬하고 반환하도록 지정 할 수 있습니다.

```
SELECT * FROM weather  
ORDERBY city;
```

이 조회문에서는 정렬 순서가 충분히 지정 되지 않았습니다. 따라서 San Francisco의 행은 순서가 다를 수 있습니다.

그러나 다음과 같이 하면 항상 위의 결과가 됩니다.

```
SELECT* FROM weather  
ORDERBY city, temp_lo;
```

쿼리 결과에서 중복 행을 제외하도록 지정 할 수 있습니다.

```
SELECT DISTINCT city FROM weather;
```

14.4.1 JOIN

지금까지의 쿼리는 한번에 하나의 테이블에만 액세스하는 것이었습니다. 쿼리는 한 번에 여러 테이블에 액세스하거나 테이블의 여러 행을 동시에 처리하는 방식으로 단일 테이블에 액세스할

수 있습니다. 한 번에 동일한 테이블이나 다른 테이블의 여러 행에 액세스하는 쿼리를 조인 쿼리라고 합니다.

예를 들어, 모든 기상 데이터를 관련 도시의 위치정보와 함께 표시하려는 경우를 들 수 있습니다. 그렇게 하려면 `weather` 테이블의 각 행의 `city` 열을 `cities` 테이블의 모든 행의 `name` 열과 비교하고 두 값이 일치하는 조합을 선택해야 합니다.

이 작업은 다음 쿼리를 통해 수행할 수 있습니다.

```
SELECT*FROM weather, cities  
WHERE city=name;
```

city	temp_lo	temp_hi	prcp	date	name	location
San Francisco	46	50	0.25	1994-11-27	San Francisco	(-194,53)
San Francisco	42				San Francisco	(-194,53)

- `Hayward` 시에 대한 결과행이 없습니다. 이것은 `cities` 테이블에는 `Hayward`와 일치하는 항목이 없기 때문에 조인 시 `weather` 테이블의 일치하지 않는 행은 무시됩니다. 이것을 어떻게 해결할 수 있는지 잠시 후에 설명합니다.
- 도시 이름을 가진 두 개의 열이 있습니다. `weather` 테이블과 `cities` 테이블의 목록이 연결되었습니다.

또한 *를 사용하지 않고 명시적으로 출력 열 목록을 지정하겠습니다.

```
SELECT city, temp_lo, temp_hi, prcp, date, location  
FROM weather, cities  
WHERE city=name;
```

열이 모두 다른 이름이었기 때문에, 파서는 자동적으로 어느 테이블의 열인가를 찾을 수 있었습니다. 두 테이블에서 컬럼 이름이 중복되는 경우, 다음과 같이 표시할 컬럼을 표시하기 위해 컬럼 이름을 규정해야 합니다.

```
SELECT weather.city, weather.temp_lo, weather.temp_hi,  
       weather.prcp, weather.date, cities.location  
  FROM weather, cities  
 WHERE cities.name = weather.city;
```

조인 쿼리는 일반적으로 모든 열 이름을 한정하는 것이 좋습니다. 그러면 나중에 테이블 중 하나에 이름이 중복된 열이 추가되어도 쿼리가 실패하지 않습니다.

여기까지 나타낸 것과 같은 조인 쿼리는 다음과 같이 내부조인 형태로 나타낼 수 있습니다.

```
SELECT *  
  FROM weather INNER JOIN cities ON (weather.city = cities.name);
```

수행하려는 쿼리는 날씨를 스캔하고 각 행에 대해 cities 행과 일치하는 행을 찾습니다. 일치하는 행이 없으면 cities 테이블의 열 부분을 일부 "빈 값"으로 바꾸고, 이 유형의 쿼리를 외부 조인이라고 합니다

```
SELECT * FROM weather LEFT OUTER JOIN cities ON (weather.city = cities.name);
```

city | temp_lo | temp_hi | prcp | date | name | location

city	temp_lo	temp_hi	prcp	date	name	location
Hayward	36	54	1994-11-29	50	0.25	
San Francisco	46	1994-11-27	San Francisco	(-194,53)	57	0
San Francisco	43	San Francisco	(-194,53)			

이 쿼리를 LEFT OUTER JOIN이라고 합니다. 조인 연산자의 왼쪽에 지정한 테이블의 각 행이 최저라도 한 번 출력되고, 우측의 테이블에서는 왼쪽의 테이블의 행과 일치하는 것만이 출력됩니다. 오른쪽 테이블과 일치하지 않는 왼쪽 테이블의 행을 출력할 때 오른쪽 테이블의 열은 빈 값 (NULL)으로 바뀝니다.

또한 테이블은 스스로 결합할 수 있습니다. 이것을 Self join이라고 합니다. 예를 들어, 다른 기상 데이터의 기온 범위 내에 있는 기상 데이터를 모두 검색하는 것을 생각해 봅시다. weather 각

행의 temp_lo와 temp_hi를 다른 weather 행의 temp_lo와 temp_hi 열을 비교해야 합니다. 다음 쿼리를 사용하여 수행할 수 있습니다

```
SELECT W1.city, W1.temp_lo ASlow, W1.temp_hi AShigh, W2.city,  
      W2.temp_lo ASlow, W2.temp_hi AShigh FROM weather W1,  
      weather W2 WHERE  
  
W1.temp_lo <  
      W2.temp_lo AND W1.temp_lo  
>  
      W2.temp_hi;
```

city	low high	city	low high
-----+-----+	-----+-----+	-----+-----+	-----+-----+

```
San Francisco| 43 |57 | San Francisco | 46 |50  
Hayward | 37 | (2 rows)54 | San Francisco | 46 |50
```

이제 조인의 왼쪽과 오른쪽을 구분할수 있도록 weather 테이블에 W1 과 W2라는 레이블이 붙어있습니다. 또한 입력량을 줄이기 위해 다른 쿼리에서도 이런 종류의 별칭을 사용할 수 있습니다.

예를 들면 다음과 같습니다

```
SELECT* FROM weather w, cities c  
WHERE w.city=c.name;
```

14.4.2 집계 함수

대부분의 다른 관계형 데이터베이스 제품과 마찬가지로 AgensSQL은 집계 함수를 지원합니다. 집계 함수는 여러 행으로부터 하나의 결과를 계산합니다. SELECT 구문에서만 사용되며 주로 count(총수), max(최대), min(최소), sum(총합), avg(평균)과 같은 연산을 수행하는데 사용됩니다.

COUNT

특정 열의 행 개수를 세는 함수입니다. COUNT(*)로 작성하면 테이블에 존재하는 모든 행의 개수가 반환되고, 특정 열에 대해서 COUNT를 수행하면 해당 열이 NULL이 아닌 행의 개수를 반환합니다.

```
SELECT COUNT(*) FROM tablename;  
SELECT COUNT(Name) FROM tablename;
```

MIN/MAX

MIN과 MAX는 최솟값과 최댓값을 구하는 함수입니다. 사용법은 위의 COUNT와 동일합니다.

```
SELECT MAX(Age) FROM tablename;
```

GROUP BY 절과 결합된 집계도 매우 유용합니다. 예를 들어, 다음 명령을 사용하여 도시별로 최저 온도의 최대 값을 찾을 수 있습니다.

```
SELECT city, max(temp_lo)  
      FROM weather  
     GROUP BY city  
    City      |max  
-----+-----  
Hayward   |37  
San Francisco | 46  
(2 rows)
```

각 집계 결과는 도시와 일치하는 전체 테이블 행에 대한 연산 결과입니다. 다음과 같이 HAVING을 사용하여 그룹화된 행을 필터링 할 수 있습니다.

```
SELECT city, max(temp_lo)  
      FROM weather  
     GROUP BY city  
HAVING max(temp_lo) < 40;  
  
city | max  
-----+-----  
Hayward | 37  
(1 rows)
```

또한 WHERE 절도 같이 사용할 수 있습니다. 모든 temp_lo의 값이 40 미만의 도시만을 출력하고 "S"로 시작하는 이름의 도시만을 대상으로 하려면 다음을 보십시오.

```
SELECT city, max(temp_lo)  
      FROM weather  
     WHERE city LIKE 'S%' 1  
     GROUP BY city  
HAVING max(temp_lo) < 40
```

WHERE와 **HAVING**의 기본적인 차이점은 **WHERE**은 그룹과 집계함수를 계산하기 전에 입력 행을 먼저 선택합니다. **HAVING**은 그룹과 집계를 계산한 후 그룹화 된 행을 선택한다는 것입니다. 따라서 **WHERE** 절은 집계 함수를 가질 수 없습니다. 반면에 **HAVING** 절에는 항상 집계 함수를 사용합니다.

앞의 예에서는 **WHERE** 내에 도시 이름 제한하여 적용할 수 있습니다. 왜냐하면 집계함수를 사용하지 않기 때문입니다. 이것은 **HAVING**에 제한을 추가하는 것보다 효율적입니다.

V. 성능 관리

15장. 대용량 데이터 처리 및 관리

15.1 Table Partitioning

Partitioning은 논리적으로 하나의 큰 테이블을 작은 물리적 조각으로 분할하는 것을 말합니다. 파티셔닝은 다음과 같은 여러 가지 이점을 제공할 수 있습니다.

- 적재된 데이터가 대용량이라도 데이터 접근 시 파티션 단위로 액세스 범위를 줄여 쿼리 성능이 향상됩니다.
- 파티션 별로 데이터가 분산 저장되므로 디스크 성능이 향상됩니다.
- 파티셔닝으로 인한 쿼리 수정이 없어 조회 테이블 관리를 위한 응용프로그램 개발이 필요 없습니다.
- 파티션 단위로 데이터를 옮기거나 인덱스를 재생성 할 수 있습니다.
- 대량 데이터 적재 및 삭제 작업시 개별 파티션 단위의 추가, 삭제, 분리 등의 DDL문을 통해 손쉽고 빠르게 수행할 수 있습니다.

15.1.1 지원하는 Partitioning 기법

- Range Partition

테이블은 키 열 또는 열 집합에 의해 정의된 “RANGE(범위)”로 분할되며 데이터는 서로 다른 파티션에 할당된 값 범위 간에 겹치지 않습니다.

예를 들어, date 값을 기준으로 매달 1일에서 말일까지 범위로 월별로 Partition을 나누는 경우입니다.

- List Partition

테이블은 각 분할 영역에 나타나는 키 값을 명시적으로 나열하여 분할됩니다.

예를 들어, location 값을 기준으로 서울, 부산, 인천 등 Data 값들의 List를 기준으로 Partition을 나누는 경우입니다.

- Hash Partition

테이블은 각 파티션에 대한 Hash 계수와 나머지를 지정하여 분할됩니다. 각 파티션에는 Hash 함수를 적용한 나머지 값을 활용하여 분할 저장 됩니다.
Partition 을 나눌 명확한 방법이 없을 경우 유용합니다. 특히 향후 거대해질 것으로 예상되는데 초기단계에서 Partition 을 설정하기 어려울 때 더욱 유용합니다.
예를 들어, 자동으로 증가하는 id 값을 기준으로 Partition 을 정의할 수 있습니다.

15.1.2 Syntax

- PARENT PARTITION TABLE 생성

```
CREATE TABLE table_name (
    column_name data_type
    [, ...]
) PARTITION BY { RANGE | LIST | HASH } ( { column_name | ( expression ) } [, ...] )
```

- CHILD PARTITION TABLE 생성

```
CREATE TABLE table_name PARTITION OF parent_table
{ FOR VALUES partition_bound_spec | DEFAULT }

partition_bound_spec is:

# RANGE PARTITION
FROM ( { partition_bound_expr | MINVALUE | MAXVALUE } [, ...] )
TO ( { partition_bound_expr | MINVALUE | MAXVALUE } [, ...] )

# LIST PARTITION
IN ( partition_bound_expr [, ...] )

# HASH PARTITION
WITH ( MODULUS numeric_literal, REMAINDER numeric_literal )
```

- PARENT/CHILD PARTITION TABLE 삭제

```
DROP TABLE table_name
```

- CHILD PARTITION TABLE 분리

```
ALTER TABLE name
DETACH PARTITION partition_name
```

- CHILD PARTITION TABLE 부착

```
ALTER TABLE name
ATTACH PARTITION partition_name
{ FOR VALUES partition_bound_spec | DEFAULT }
```

15.1.3 Example

1) RANGE PARTITION

RANGE PARTITION TABLE 생성 (PARENT PARTITION)

```
CREATE TABLE measurement (
    id      serial,
    city_id  int not null,
    logdate   date not null,
    peaktemp   int,
    unitsales  int
) PARTITION BY RANGE (logdate); -- 파티션 키를 logdate로 지정
```

SUB PARTITION 생성

```
CREATE TABLE measurement_y2020 PARTITION OF measurement FOR VALUES FROM
('2020-01-01') TO ('2021-01-01');
```

```
CREATE TABLE measurement_y2021 PARTITION OF measurement FOR VALUES FROM
('2021-01-01') TO ('2022-01-01');
```

```
CREATE TABLE measurement_y2022 PARTITION OF measurement FOR VALUES FROM
('2022-01-01') TO ('2023-01-01');
```

PARENT PARTITION에 DATA INSERT

- 부모 파티션에 데이터를 insert하면 SUB파티션 생성시 정의된 범위에 따라 실제로 SUB파티션에 데이터가 저장됩니다.

```
INSERT INTO measurement (id, city_id, logdate, peaktemp, unitsales) values
(1111, 1, '2020-01-05', 39, 50),
(1112, 2, '2021-01-05', 45, 50),
(1113, 3, '2022-01-05', 41, 50);
```

PARTITION TABLE별 데이터 확인

```

SELECT * FROM measurement;
+-----+-----+-----+
| id   | city_id | logdate | peaktemp | unitsales |
+-----+-----+-----+
| 1111 |     1 | 2020-01-05 |      39 |      50 |
| 1112 |     2 | 2021-01-05 |      45 |      50 |
| 1113 |     3 | 2022-01-05 |      41 |      50 |
+-----+
(3 rows)

SELECT * FROM measurement_y2020;
+-----+-----+-----+
| id   | city_id | logdate | peaktemp | unitsales |
+-----+-----+-----+
| 1111 |     1 | 2020-01-05 |      39 |      50 |
+-----+
(1 row)

SELECT * FROM measurement_y2021;
+-----+-----+-----+
| id   | city_id | logdate | peaktemp | unitsales |
+-----+-----+-----+
| 1112 |     2 | 2021-01-05 |      45 |      50 |
+-----+
(1 row)

SELECT * FROM measurement_y2022;
+-----+-----+-----+
| id   | city_id | logdate | peaktemp | unitsales |
+-----+-----+-----+
| 1113 |     3 | 2022-01-05 |      41 |      50 |
+-----+
(1 row)

```

2) LIST PARTITION

```

# 기존 존재하는 TABLE 제거
DROP TABLE measurement;

# LIST PARTITION TABLE 생성
CREATE TABLE measurement (
    id      serial,
    city_id int not null,
    logdate date not null,
    peaktemp int,
    unitsales int
) PARTITION BY LIST (city_id); -- 파티션 키를 city_id로 지정

# SUB PARTITION 생성
CREATE TABLE measurement_c1 PARTITION OF measurement FOR VALUES in(1);

CREATE TABLE measurement_c2 PARTITION OF measurement FOR VALUES in(2);

CREATE TABLE measurement_c3 PARTITION OF measurement FOR VALUES in(3);

# PARENT PARTITION || DATA INSERT
INSERT INTO measurement (id, city_id, logdate, peaktemp, unitsales) values
(1111, 1,'2020-01-05', 39, 50),

```

```
(1112, 2, '2021-01-05', 45, 50),  
(1113, 3, '2022-01-05', 41, 50);
```

PARTITION TABLE 별 데이터 확인

```
SELECT * FROM measurement;  
id | city_id | logdate | peaktemp | unitsales  
----+-----+-----+-----+  
1111 | 1 | 2020-01-05 | 39 | 50  
1112 | 2 | 2021-01-05 | 45 | 50  
1113 | 3 | 2022-01-05 | 41 | 50  
(3 rows)
```

```
SELECT * FROM measurement_c1;  
id | city_id | logdate | peaktemp | unitsales  
----+-----+-----+-----+  
1111 | 1 | 2020-01-05 | 39 | 50  
(1 row)
```

```
SELECT * FROM measurement_c2;  
id | city_id | logdate | peaktemp | unitsales  
----+-----+-----+-----+  
1112 | 2 | 2021-01-05 | 45 | 50  
(1 row)
```

```
SELECT * FROM measurement_c3;  
id | city_id | logdate | peaktemp | unitsales  
----+-----+-----+-----+  
1113 | 3 | 2022-01-05 | 41 | 50  
(1 row)
```

3) HASH PARTITION

```
# 기존 존재하는 TABLE 제거  
DROP TABLE measurement;
```

```
# HASH PARTITION TABLE 생성  
CREATE TABLE measurement (  
    id      serial,  
    city_id int not null,  
    logdate date not null,  
    peaktemp int,  
    unitsales int  
) PARTITION BY HASH (id); -- 파티션 키를 id로 지정
```

```
# SUB PARTITION 생성  
CREATE TABLE measurement_id1 PARTITION OF measurement FOR VALUES WITH  
(modulus 3, remainder 0);
```

```
CREATE TABLE measurement_id2 PARTITION OF measurement FOR VALUES WITH  
(modulus 3, remainder 1);
```

```
CREATE TABLE measurement_id3 PARTITION OF measurement FOR VALUES WITH  
(modulus 3, remainder 2);
```

PARENT PARTITION에|| DATA INSERT

```
INSERT INTO measurement (id, city_id, logdate, peaktemp, unitsales) values  
(1111, 1,'2020-01-05', 39, 50),  
(1112, 2, '2021-01-05', 45, 50),  
(1113, 3, '2022-01-05', 41, 50);
```

PARTITION TABLE별 데이터 확인

```
SELECT * FROM measurement;  
id | city_id | logdate | peaktemp | unitsales  
----+-----+-----+-----+  
1111 | 1 | 2020-01-05 | 39 | 50  
1112 | 2 | 2021-01-05 | 45 | 50  
1113 | 3 | 2022-01-05 | 41 | 50  
(3 rows)
```

```
SELECT * FROM measurement_id1;  
id | city_id | logdate | peaktemp | unitsales  
----+-----+-----+-----+  
(0 rows)
```

```
SELECT * FROM measurement_id2;  
id | city_id | logdate | peaktemp | unitsales  
----+-----+-----+-----+  
1111 | 1 | 2020-01-05 | 39 | 50  
(1 row)
```

```
SELECT * FROM measurement_id3;  
id | city_id | logdate | peaktemp | unitsales  
----+-----+-----+-----+  
1112 | 2 | 2021-01-05 | 45 | 50  
1113 | 3 | 2022-01-05 | 41 | 50  
(2 rows)
```

4) Partition 제거, 분리, 부착

PARTITION TABLE 제거

```
DROP TABLE measurement_id3;
```

PARTITION TABLE 제거 확인

```
\d+ measurement;  
postgres=# \d+ measurement;
```

```
Partitioned table "public.measurement"
```

Column	Type	Collation	Nullable	Default	Storage	Stats
target						

```

id      | integer |      | not null | nextval('measurement_id_seq'::regclass) | plain  |
|
city_id | integer |      | not null |                                         | plain   |
logdate | date   |      | not null |                                         | plain   |
peaktemp | integer |     |      |                                         | plain   |
unitsales | integer |    |      |                                         | plain   |
Partition key: HASH (id)
Partitions: measurement_id1 FOR VALUES WITH (modulus 3, remainder 0),
            measurement_id2 FOR VALUES WITH (modulus 3, remainder 1)

```

PARTITION TABLE 분리

```
ALTER TABLE measurement DETACH PARTITION measurement_id1;
```

PARTITION TABLE 분리 확인

```
\d+ measurement;
```

```
Partitioned table "public.measurement"
```

Column	Type	Collation	Nullable	Default	Storage	Stats
target	Description					

```

id      | integer |      | not null | nextval('measurement_id_seq'::regclass) | plain  |
|
city_id | integer |      | not null |                                         | plain   |
logdate | date   |      | not null |                                         | plain   |
peaktemp | integer |     |      |                                         | plain   |
unitsales | integer |    |      |                                         | plain   |
Partition key: HASH (id)
Partitions: measurement_id2 FOR VALUES WITH (modulus 3, remainder 1)

```

PARTITION TABLE 부착

```
ALTER TABLE measurement ATTACH PARTITION measurement_id1 FOR VALUES WITH
(modulus 3, remainder 0);
```

PARTITION TABLE 부착 확인

```
\d+ measurement;
```

```
Partitioned table "public.measurement"
```

Column	Type	Collation	Nullable	Default	Storage	Stats
target	Description					

```

id      | integer |      | not null | nextval('measurement_id_seq'::regclass) | plain  |
|
city_id | integer |      | not null |                                         | plain   |
logdate | date   |      | not null |                                         | plain   |
peaktemp | integer |     |      |                                         | plain   |
unitsales | integer |    |      |                                         | plain   |
Partition key: HASH (id)
Partitions: measurement_id1 FOR VALUES WITH (modulus 3, remainder 0),
            measurement_id2 FOR VALUES WITH (modulus 3, remainder 1)

```

15.2 병렬 처리

AgensSQL은 Query 응답속도를 더 빠르게 하기 위해 옵티マイ저에서 병렬쿼리가 특정 쿼리에 대한 가장 빠른 실행계획이라 판단되면 여러개의 CPU를 활용할 수 있는 Parallel Query 기능이 있습니다. 많은 양의 데이터에서 사용자에게 몇 개의 행만 반환하는 쿼리에서 일반적으로 가장 유용하며 빠르게 실행될 수 있습니다.

15.2.1 병렬 처리 조건

- 1) `max_parallel_workers_per_gather`의 값을 0보다 큰 값으로 설정해야 합니다. 해당 파라미터의 기본 값이 2이므로 기본 설정인 경우 문제는 없습니다.
- 2) 시스템이 `single user mode`에서 실행되지 않아야 합니다. 해당 모드에서는 전체 데이터베이스 시스템이 단일 프로세스에서 실행되므로 `Background worker`를 사용할 수 없어서 `parallel` 동작을 할 수 없습니다.

15.2.2 examples

```
# Test Table 생성
create table people(
    id      integer not null
, born    date not null
, phone   text null
, dept    text null
, create_dt date not null
);

# Test Data Insert
insert into people
select series as id,
(timestamp '1970-01-01' + random() * (timestamp '2002-12-31' - timestamp '1970-01-01'))::date
as born,
'010-' || LPAD(trunc(random()*9999)::text,4,'0') || '-' || LPAD(trunc(random()*9999)::text,4, '0') as
phone,
substr('서울대전대구부산', trunc(random() * 4)::integer*2 + 1, 2) as dept,
now() + series * INTERVAL '1 day' as create_dt
from generate_series(1, 500000) series;

# Parallel Query
explain select * from people where id=12345;

-----  
QUERY PLAN
-----
Gather (cost=1000.00..7022.34 rows=1625 width=76)
  Workers Planned: 2
```

```
-> Parallel Seq Scan on people (cost=0.00..5859.84 rows=677 width=76)
   Filter: (id = 12345)
(4 rows)
```

15.3 SQL Hint 기능 (pg_hint_plan)

AgensSQL의 옵티マイ저는 각 쿼리에 대하여 실행 비용을 산정하여 최적의 실행계획을 선택하여 쿼리를 실행합니다. 그러나 옵티マイ저의 실행계획의 경우에 따라서는 완벽하지 않습니다. 따라서 pg_hint_plan을 이용하여 특정 쿼리의 성능 향상을 위하여 인덱스를 지정하거나, 조인 순서 및 방법을 선택하여 지정할 수 있습니다. 그러나 잘못된 Hint는 비효율이 발생할 수 있습니다.

15.3.1 Hint Plan 유형

1) SCAN

Hint	설명
SeqScan	Table Full Scan 방식을 유도
IndexScan	Index Scan 방식을 유도
IndexOnlyScan	Index Only Scan 방식을 유도한다 Index Only Scan 불가시 Index Scan으로 동작
BitmapScan	Bitmap Scan 방식을 유도

2) JOIN

Hint	설명
NestLoop	NL Join으로 유도
HashJoin	Hash Join으로 유도
MergeJoin	Merge Join으로 유도

15.3.2 Syntax

```
pg_hint_plan의 경우 쿼리의 앞에 /*+ HINT */ 의 형식으로 사용합니다.
```

```

# SCAN
/*+
{ SeqScan | IndexScan | IndexOnlyScan | BitmapScan }(table_name [ Index_name ] )
*/
# JOIN
/*+
{ NestLoop | HashJoin | MergeJoin }(table_name | ( expression ) )
*/

```

15.3.3 Example

```

# Database 접속
$ asql -U agens -d postgres

# pg_hint_plan 생성
create extension pg_hint_plan;

# Table 생성
create table tab1(
    id      integer not null
    , born   date not null
    , phone  text null
    , dept   text null
    , create_dt date not null
);

create table tab2(
    id      integer not null
    , born   date not null
    , phone  text null
    , dept   text null
    , create_dt date not null
);

# Test Data Insert
insert into tab1
select series as id,
(timestamp '1970-01-01' + random() * (timestamp '2002-12-31' - timestamp
'1970-01-01'))::date as born,
'010-'||LPAD(trunc(random()*9999)::text,4,'0')||'-'||LPAD(trunc(random()*9999)::text,4, '0')
as phone,
substr('서울대전대구부산', trunc(random() * 4)::integer*2 + 1, 2) as dept,
now() + series * INTERVAL '1 day' as create_dt
from generate_series(1, 500000) series;

```

```

insert into tab2
select series as id,
(timestamp '1990-01-01' + random() * (timestamp '2002-12-31' - timestamp
'1970-01-01'))::date as born,
'010-'||LPAD(trunc(random()*9999)::text,4,'0')||'-'||LPAD(trunc(random()*9999)::text,4, '0')
as phone,
substr('서울대전대구부산', trunc(random() * 4)::integer*2 + 1, 2) as dept,
now() + series * INTERVAL '1 day' as create_dt
from generate_series(1, 500000) series;

# Index 생성
create index idx_tab1 on tab1(phone);

# SCAN 활용
/*+
SeqScan (tab1)
*/
explain select * from tab1 where phone='010-7872-1770';

```

QUERY PLAN

```

Gather (cost=1000.00..8021.17 rows=2500 width=76)
Workers Planned: 2
-> Parallel Seq Scan on tab1 (cost=0.00..6771.17 rows=1042 width=76)
  Filter: (phone = '010-7872-1770'::text)

```

```

/*+
IndexScan (tab1 idx_tab1)
*/
explain select * from tab1 where phone='010-7872-1770';

```

QUERY PLAN

```

Index Scan using idx_tab1 on tab1 (cost=0.42..7780.17 rows=2500 width=76)
Index Cond: (phone = '010-7872-1770'::text)

```

```

/*+
IndexOnlyScan (tab1 idx_tab1)
*/
explain select phone from tab1 where phone='010-7872-1770';

```

QUERY PLAN

```

Index Only Scan using idx_tab1 on tab1 (cost=0.42..7780.17 rows=2500 width=32)
Index Cond: (phone = '010-7872-1770'::text)

```

```

# JOIN 활용
/*+

```

```

NestLoop(a b)
*/
explain select * from tab1 a, tab2 b where a.id=b.id;

        QUERY PLAN
-----
Gather (cost=1000.00..2169156497.28 rows=812565000 width=152)
  Workers Planned: 2
    -> Nested Loop (cost=0.00..2087898997.28 rows=338568750 width=152)
      Join Filter: (a.id = b.id)
      -> Parallel Seq Scan on tab2 b (cost=0.00..5521.27 rows=135428 width=76)
      -> Seq Scan on tab1 a (cost=0.00..9167.00 rows=500000 width=76)

/*+
HashJoin(a b)
*/
explain select * from tab1 a, tab2 b where a.id=b.id;

        QUERY PLAN
-----
Hash Join (cost=11480.09..28461922.09 rows=812565000 width=152)
  Hash Cond: (a.id = b.id)
    -> Seq Scan on tab1 a (cost=0.00..9167.00 rows=500000 width=76)
    -> Hash (cost=7417.26..7417.26 rows=325026 width=76)
      -> Seq Scan on tab2 b (cost=0.00..7417.26 rows=325026 width=76)

/*+
MergeJoin(a b)
*/
explain select * from tab1 a, tab2 b where a.id=b.id;

        QUERY PLAN
-----
Merge Join (cost=130332.13..12321682.26 rows=812565000 width=152)
  Merge Cond: (b.id = a.id)
    -> Sort (cost=51618.21..52430.77 rows=325026 width=76)
      Sort Key: b.id
      -> Seq Scan on tab2 b (cost=0.00..7417.26 rows=325026 width=76)
    -> Materialize (cost=78713.92..81213.92 rows=500000 width=76)
      -> Sort (cost=78713.92..79963.92 rows=500000 width=76)
        Sort Key: a.id
      -> Seq Scan on tab1 a (cost=0.00..9167.00 rows=500000 width=76)

```

15.4 Partition Pruning 기능

Partition Pruning은 Partitioning Table에서 쿼리의 성능을 향상 시키기 위한 기능입니다. Partitioning Table을 조회하는 쿼리의 경우 파티션 조건에 충족하는 행이 있는 Partition Table만을 대상으로 쿼리 하도록 합니다.

15.4.1 기능 활성화 조건

```
postgresql.conf 설정
```

```
enable_partition_pruning = on
```

또는 ALTER SYSTEM 명령을 통한 동적 반영

```
alter system set enable_partition_pruning = on ;
```

```
select pg_reload_conf() ;
```

15.4.2 Example

Range Partition Table

```
explain select * from measurement where logdate = '2022-06-01';
          QUERY PLAN
```

```
-----  
Seq Scan on measurement_y2022 measurement (cost=0.00..31.25 rows=8 width=20)  
  Filter: (logdate = '2022-06-01'::date)
```

List Partition Table

```
explain select * from measurement where city_id=3;
          QUERY PLAN
```

```
-----  
Seq Scan on measurement_c3 measurement (cost=0.00..31.25 rows=8 width=20)  
  Filter: (city_id = 3)
```

Hash Partition Table

```
explain select * from measurement where id=3;
          QUERY PLAN
```

```
-----  
Seq Scan on measurement_id2 measurement (cost=0.00..31.25 rows=8 width=20)  
  Filter: (id = 3)
```

15.5 다양한 Index 활용

데이터베이스는 조건에 맞는 데이터를 검색하기 위하여 테이블의 모든 데이터를 검색 해야 합니다. 이를 효율적으로 검색하는 방법은 조건에 필요한 데이터 인덱스를 생성하면 조건에 맞추어 필요한 데이터를 더 효율적으로 검색할 수 있습니다.

15.5.1 Index 유형

AgensSQL은 B-tree, Hash, GIN, BRIN 등 여러 인덱스 유형을 제공합니다. 기본적으로 CREATE INDEX 명령으로 생성하는 경우 B-tree 인덱스로 생성 됩니다.

B-tree	순서로 데이터를 정렬하여 범위 조건 쿼리를 사용하는 순차적인 일반 데이터에 사용을 고려합니다.
Hash	Hash 값을 이용하여 정렬하여 불특정 단순 동등성 조건 쿼리를 사용하는 데이터에 사용을 고려 합니다.
GIN	GIN(Generalized Inverted Index)연산자를 사용하여 인덱스를 생성합니다. 특정 조건(ex. LIKE) 쿼리를 이용 데이터에 사용을 고려 합니다.
BRIN	데이터 블록 범위 인덱스로 데이터를 페이지 단위로 인덱스 하는 방법입니다. 대량의 데이터에서 특정 컬럼 값이 저장 순서에 따라 범위 정렬되어 있는 경우 효율적 입니다.

15.5.2 Examples

```
# Test Table 생성
create table people(
    id      integer not null
  , born    date not null
  , phone   text null
  , dept    text null
  , create_dt date not null
);

# Test Data Insert
insert into people
select series as id,
(timestamp '1970-01-01' + random() * (timestamp '2002-12-31' - timestamp '1970-01-01'))::date as
born,
'010-' || LPAD(trunc(random()*9999)::text,4,'0') || '-' || LPAD(trunc(random()*9999)::text,4, '0') as
phone,
substr('서울대전대구부산', trunc(random() * 4)::integer*2 + 1, 2) as dept,
now() + series * INTERVAL '1 day' as create_dt
```

```
from generate_series(1, 500000) series;

# B-tree 인덱스 생성
CREATE INDEX people_id_btree ON people (id);

# B-tree 인덱스 활용
EXPLAIN SELECT * FROM people where id='12345';
    QUERY PLAN
-----
Index Scan using people_id_btree on people (cost=0.42..8.44 rows=1 width=33)
  Index Cond: (id = 12345)

# Hash 인덱스 생성
CREATE INDEX people_born_hash ON people USING HASH (born);

# Hash 인덱스 활용
EXPLAIN SELECT * FROM people where born='1987-01-01';
    QUERY PLAN
-----
Bitmap Heap Scan on people (cost=4.33..160.20 rows=42 width=33)
  Recheck Cond: (born = '1987-01-01'::date)
    -> Bitmap Index Scan on people_born_hash (cost=0.00..4.32 rows=42 width=0)
      Index Cond: (born = '1987-01-01'::date)

# GIN 인덱스 생성 (pg_trgm 설치 필요)
CREATE EXTENSION pg_trgm;
CREATE INDEX people_phone_gin ON people USING GIN(phone gin_trgm_ops);

# GIN 인덱스 활용
EXPLAIN SELECT * FROM people where phone LIKE '%1987%';
    QUERY PLAN
-----
Bitmap Heap Scan on people (cost=55.00..4386.02 rows=4000 width=76)
  Recheck Cond: (phone ~~ '%1987%'::text)
    -> Bitmap Index Scan on people_phone_gin (cost=0.00..54.00 rows=4000 width=0)
      Index Cond: (phone ~~ '%1987%'::text)
(4 rows)

# BRIN 인덱스 생성
CREATE INDEX people_create_brin ON people USING BRIN(create_dt);

# BRIN 인덱스 활용
EXPLAIN SELECT * FROM people where create_dt < '2023-12-07';
    QUERY PLAN
-----
Bitmap Heap Scan on people (cost=13.27..4369.67 rows=4950 width=33)
  Recheck Cond: (create_dt < '2023-12-07'::date)
    -> Bitmap Index Scan on people_create_brin (cost=0.00..12.03 rows=15152 width=0)
      Index Cond: (create_dt < '2023-12-07'::date)
```

VI. 보안

16장. Password Profile

AgensSQL을 사용하면 데이터베이스 슈퍼유저가 명명된 프로파일을 만들 수 있습니다. 각 프로필은 암호 및 md5 인증을 강화하는 암호 관리 규칙을 정의합니다. 프로필의 규칙은 다음과 같습니다.

- 실패한 로그인 시도 횟수
- 과도한 로그인 시도 실패로 인해 계정 잠금
- 비밀번호를 만료 표시
- 암호 만료 후 유예 기간 정의
- 암호 복잡성에 대한 규칙 정의
- 암호 재사용을 제한하는 규칙 정의

Password Profile은 유사한 인증 요구 사항을 공유하는 역할 그룹을 쉽게 관리하는 데 사용됩니다. 암호 요구 사항이 변경되면 해당 프로필과 연결된 각 사용자에게 새 요구 사항이 적용되도록 프로필을 수정할 수 있습니다.

프로필은 한 명 이상의 사용자와 연결할 수 있습니다. 사용자가 서버에 연결하면 서버는 자신의 로그인 역할과 연결된 프로필을 적용합니다. 프로파일은 클러스터 내의 모든 데이터베이스에서 공유되지만, 각 클러스터는 여러 개의 프로파일을 가질 수 있습니다. 여러 데이터베이스에 대한 액세스 권한을 가진 단일 사용자는 클러스터 내의 각 데이터베이스에 연결할 때 동일한 프로파일을 사용합니다.

AgensSQL은 다음 이름의 프로파일을 만듭니다. DEFAULT 대체 프로필을 지정하지 않으면 역할이 생성될 때 새 역할과 연결되어 DEFAULT 프로필을 삭제할 수 없습니다.

DEFAULT 프로필에는 다음 특성이 포함되어 있습니다.

FAILED_LOGIN_ATTEMPTS	UNLIMITED
PASSWORD_LOCK_TIME	UNLIMITED
PASSWORD_LIFE_TIME	UNLIMITED
PASSWORD_GRACE_TIME	UNLIMITED
PASSWORD_REUSE_TIME	UNLIMITED
PASSWORD_REUSE_MAX	UNLIMITED
PASSWORD_VERIFY_FUNCTION	NULL

데이터베이스 superuser만 ALTER PROFILE 명령을 사용하여 DEFAULT 프로필의 매개 변수 값을 수정할 수 있습니다.

16.1 Password Profile

다음 생성할 카탈로그 뷰에는 데이터베이스 오브젝트의 정의가 포함되어 있습니다.

16.1.1 ag_profile_view

각 Profile의 설정 정보를 확인 합니다.

- Example

```
# ag_profile_view 생성
CREATE VIEW ag_profile_view as SELECT
prfname as Profile,
case when prffailedloginattempts = -2 then 'Unlimited'
      when prffailedloginattempts = -1 then 'Default'
      else prffailedloginattempts::text
end as Attempts,
case when prfpasswordlocktime = -2 then 'Unlimited'
      when prfpasswordlocktime = -1 then 'Default'
      else (prfpasswordlocktime/60/60/24)::text
end as LockTime,
case when prfpasswordlifetime = -2 then 'Unlimited'
      when prfpasswordlifetime = -1 then 'Default'
      else (prfpasswordlifetime/60/60/24)::text
end as LifeTime,
case when prfpasswordgracetime = -2 then 'Unlimited'
      when prfpasswordgracetime = -1 then 'Default'
      else (prfpasswordgracetime/60/60/24)::text
end as GraceTime,
case when prfpasswordreusetime = -2 then 'Unlimited'
      when prfpasswordreusetime = -1 then 'Default'
      else (prfpasswordreusetime/60/60/24)::text
end as ReuseTime,
case when prfpasswordreusemax = -2 then 'Unlimited'
      when prfpasswordreusemax = -1 then 'Default'
      else prfpasswordreusemax::text
end as ReuseMax,
(select pg_proc.proname from pg_proc
where pg_proc.oid=ag_profile.prfpasswordverifyfunc) as VerifyFunc
FROM ag_profile;

# ag_profile_view 조회
select * from ag_profile_view ;
```

```
profile | attempts | locktime | lifetime | gracetime | reusetime | reusemax | verifyfunc
-----+-----+-----+-----+-----+-----+-----+
default | Unlimited |
(1 row)
```

16.1.2 ag_password_view

각 Role / User의 Profile 설정, Account 상태, Password 변경 일자를 확인 합니다.

- Example

```
# ag_password_view 생성
CREATE VIEW ag_password_view as SELECT
PG_ROLES.ROLNAME as RoleName,
PG_ROLES.ROLPROFILE as PROFILE,
AG_GET_ROLE_STATUS(PG_ROLES.OID) as Account,
to_char(pg_authid.rolpasswordsetat,'YYYY-MM-DD HH24:MI:SS') as PasswordSetDate
FROM PG_ROLES, pg_authid
WHERE PG_ROLES.rolname=pg_authid.rolname;

# ag_password_view 조회
select * from ag_password_view;
  rolename    | profile | account | passwordsetdate
-----+-----+-----+
  pg_monitor    | default | OPEN   |
  pg_read_all_settings  | default | OPEN   |
  pg_read_all_stats    | default | OPEN   |
  pg_stat_scan_tables  | default | OPEN   |
  pg_read_server_files  | default | OPEN   |
  pg_write_server_files | default | OPEN   |
  pg_execute_server_program | default | OPEN   |
  pg_signal_backend    | default | OPEN   |
  agens            | default | OPEN   | 2023-03-13 11:08:56
(9 rows)
```

16.2 Create Password Profile

16.2.1 Syntax

새로운 password profile을 만들려면 다음 구문을 사용하십시오.

```
CREATE PROFILE <profile_name> [LIMIT {<parameter> <value>} ...];
```

LIMIT 절과 하나 이상의 공백으로 구분된 매개 parameter/value 쌍 뒤에 AgensSQL에 의해 시행하는 규칙을 지정합니다.

16.2.2 Parameters

- profile_name : Profile 이름을 지정합니다.
- parameter : Profile에 의해 제한되는 특성을 지정합니다.
- value : Profile의 제한 값 설정
- 각 Parameters의 지원 값

FAILED_LOGIN_ATTEMPTS

사용자가 시도할 수 있는 실패한 로그인 시도 횟수를 지정합니다. 지원되는 값은 다음과 같습니다.

- 0 보다 큰 INTEGER (단위 횟수)
- DEFAULT : DEFAULT 프로필에 지정된 FAILED_LOGIN_ATTEMPTS 값입니다.
- UNLIMITED : 연결하는 사용자는 무제한 로그인 시도를 시도할 수 있습니다.

PASSWORD_LOCK_TIME

서버가 FAILED_LOGIN_ATTEMPTS로 인해 잠긴 계정의 잠금을 해제되기 전에 경과해야 하는 시간을 지정합니다. 지원되는 값은 다음과 같습니다.

- 0 보다 크거나 같은 NUMERIC 값 (단위 : 일)
- 하루의 소수 부분을 지정하려면 10진수 값을 지정하십시오. 예를 들어 값 4.5를 사용하여 4 일 12 시간을 지정합니다.
- DEFAULT : DEFAULT 프로필에 지정된 PASSWORD_LOCK_TIME의 값입니다.
- UNLIMITED : 데이터베이스 수퍼유저가 수동으로 잠금을 해제할 때까지 계정이 잠깁니다.

PASSWORD_LIFE_TIME

사용자에게 새 암호를 입력하라는 메시지가 표시되기 전에 현재 암호를 사용할 수 있는 일 수를 지정합니다. PASSWORD_LIFE_TIME 절을 사용하여 암호가 만료된 후 연결이 거부되기 전에 경과할 일 수를 지정하는 경우 PASSWORD_GRACE_TIME 절을 포함합니다. PASSWORD_GRACE_TIME이 지정되지 않은 경우 암호는 기본값인 PASSWORD_GRACE_TIME

으로 지정된 날짜에 만료되며 사용자는 새 암호가 제공될 때까지 명령을 실행할 수 없습니다. 지원되는 값은 다음과 같습니다.

- 0 보다 크거나 같은 NUMERIC 값. (단위 : 일)
- 하루의 소수 부분을 지정하려면 10진수 값을 지정하십시오. 예를 들어 값 4.5를 사용하여 4 일 12 시간을 지정합니다.
- DEFAULT : DEFAULT 프로필에 지정된 PASSWORD_LIFE_TIME 값입니다.
- UNLIMITED : 암호에 만료 날짜가 없습니다.

PASSWORD_GRACE_TIME

암호가 만료된 후 사용자가 암호를 변경해야 할 때까지의 유예 기간을 지정합니다. 유예 기간이 만료되면 사용자는 연결할 수 있지만 만료된 암호를 업데이트할 때까지 명령을 실행할 수 없습니다. 지원되는 값은 다음과 같습니다.

- 0 보다 크거나 같은 NUMERIC 값. (단위: 일)
- 하루의 소수 부분을 지정하려면 10진수 값을 지정하십시오. 예를 들어 값 4.5를 사용하여 4 일 12 시간을 지정합니다.
- DEFAULT : DEFAULT 프로필에 지정된 PASSWORD_GRACE_TIME 값입니다.
- UNLIMITED : 유예 기간은 무한합니다.

PASSWORD_REUSE_TIME

사용자가 암호를 다시 사용하기 전에 기다려야 하는 일 수를 지정합니다.

PASSWORD_REUSE_TIME 및 PASSWORD_REUSE_MAX 매개변수는 함께 사용하기 위한 것입니다. 이러한 매개변수 중 하나에 대해 유한한 값을 지정하고 다른 하나는 UNLIMITED로 지정하면 이전 암호를 다시 사용할 수 없습니다. 두 매개변수가 모두 UNLIMITED로 설정되면 암호 재사용에 대한 제한이 없습니다. 지원되는 값은 다음과 같습니다.

- 0 보다 크거나 같은 NUMERIC 값. (단위 : 일)
- 하루의 소수 부분을 지정하려면 10진수 값을 지정하십시오. 예를 들어 값 4.5를 사용하여 4 일 12 시간을 지정합니다.
- DEFAULT : DEFAULT 프로필에 지정된 PASSWORD_REUSE_TIME의 값입니다.
- UNLIMITED : 암호는 제한 없이 재사용할 수 있습니다.

PASSWORD_REUSE_MAX

비밀번호를 재사용하기 전에 발생해야 하는 비밀번호 변경 횟수를 지정합니다.

PASSWORD_REUSE_TIME 및 PASSWORD_REUSE_MAX 매개변수는 함께 사용하기 위한 것입니다. 이러한 매개변수 중 하나에 대해 유한한 값을 지정하고 다른 하나는 UNLIMITED

로 지정하면 이전 암호를 다시 사용할 수 없습니다. 두 매개변수가 모두 UNLIMITED로 설정되면 암호 재사용에 대한 제한이 없습니다. 지원되는 값은 다음과 같습니다.

- 0 보다 크거나 같은 INTEGER 값 (단위 : 횟수)
- DEFAULT : DEFAULT 프로파일에 지정된 PASSWORD_REUSE_MAX 값입니다.
- UNLIMITED : 제한 없이 암호를 다시 사용할 수 있습니다.

PASSWORD_VERIFY_FUNCTION

암호 복잡성을 지정합니다. 지원되는 값은 다음과 같습니다:

- PL/pgSQL function 이름
- DEFAULT : DEFAULT 프로필에 지정된 PASSWORD_VERIFY_FUNCTION 값입니다.
- NULL

16.2.3 Examples

(기본 구성으로 \$PGDATA/pg_hba.conf 설정에서 Local 인증 방식이 설정되어 있어 Password 테스트 접속시 -h옵션을 이용한 host를 통한 접속이 필요합니다.)

- 1) 다음 명령은 acctgrp라는 프로파일을 생성합니다. 프로필은 사용자가 5번의 시도에서 올바른 암호로 인증되지 않은 경우 계정이 하루 동안 잠기도록 지정합니다.

```
CREATE PROFILE acctgrp LIMIT  
FAILED_LOGIN_ATTEMPTS 5  
PASSWORD_LOCK_TIME 1;
```

- 2) 다음 명령은 sales라는 프로필을 생성합니다. 프로필은 사용자가 7일마다 암호를 변경해야 함을 지정합니다.

```
CREATE PROFILE sales LIMIT  
PASSWORD_LIFE_TIME 7  
PASSWORD_GRACE_TIME 1;
```

프로필에 지정된 7일이 경과하기 전에 암호를 변경하지 않은 경우 로그인 시 경고가 표시됩니다. 1일의 유예 기간이 지나면 암호를 변경할 때까지 계정에서 명령을 호출할 수 없습니다.

- 3) 다음 명령은 accts라는 프로파일을 만듭니다. 프로필은 사용자가 암호를 마지막으로 사용한 후 1일 이내에 암호를 재사용할 수 없으며 암호를 다시 사용하기 전에 최소 5번 이상 암호를 변경해야 함을 지정합니다.

```
CREATE PROFILE accts LIMIT  
PASSWORD_REUSE_TIME 1  
PASSWORD_REUSE_MAX 5;
```

- 4) Profile 생성 정보를 확인 합니다.

```
SELECT * FROM ag_profile_view;  
profile | attempts | locktime | lifetime | gracetime | reusetime | reusemax | verifyfunc  
-----+-----+-----+-----+-----+-----+-----+  
default | Unlimited | Unlimited | Unlimited | Unlimited | Unlimited | Unlimited |  
acctgrp | 5 | 1 | Default | Default | Default | Default |  
sales | Default | Default | 7 | 1 | Default | Default |  
accts | Default | Default | Default | Default | 1 | 5 |  
(4 rows)
```

16.3 Create Password Profile Function

PASSWORD_VERIFY_FUNCTION을 지정할 때 사용자가 암호를 변경할 때 적용할 보안 규칙을 지정하는 사용자 지정 기능을 제공할 수 있습니다. 예를 들어, 새 암호는 n자 이상이어야 하며 특정 값을 포함할 수 없도록 규정하는 규칙을 지정할 수 있습니다.

16.3.1 Syntax

새로운 Password Profile Function을 만들려면 다음 구문을 사용하십시오.

```
CREATE OR REPLACE FUNCTION <function_name> (<user_name> VARCHAR,  
                                              <new_password> VARCHAR,  
                                              <old_password> VARCHAR) RETURN boolean
```

16.3.2 Parameters

- user_name : 유저네임
- new_password : 새 패스워드
- old_password : 이전 패스워드

데이터베이스 Superuser가 비밀번호를 변경하면 세 번째 매개변수는 항상 NULL입니다. 이 함수는 부울 값을 반환합니다. 함수가 true를 반환하고 예외를 발생시키지 않으면 암호가 허용됩니다. 함수가 false를 반환하거나 예외가 발생하면 암호가 거부됩니다. 함수에서 예외가 발생하면 지정된 오류 메시지가 사용자에게 표시됩니다. 함수가 예외를 발생시키지 않지만 false를 반환하면 다음 오류 메시지가 표시됩니다.

```
ERROR: password verification for the specified password failed / 지정한 암호에 대한 암호 확인 실패
```

함수는 데이터베이스 Superuser로 생성해야 합니다.

16.3.3 Examples

예제에서는 프로필 및 사용자 정의 함수를 생성한 다음 해당 함수를 프로파일과 연결합니다.

- 1) CREATE PROFILE 명령은 acctgrp_profile이라는 프로필을 생성합니다.

```
CREATE PROFILE acctgrp_profile;
```

- 2) verify_password라는 (Schema 정규화된) 함수를 생성합니다.

```
CREATE OR REPLACE FUNCTION verify_password(user_name varchar,
new_password varchar, old_password varchar)
RETURNS boolean
LANGUAGE plpgsql
AS
$$
BEGIN
IF (length(new_password) < 5)
THEN
RAISE EXCEPTION 'too short';
RETURN false;
END IF;

IF substring(new_password FROM old_password) IS NOT NULL
THEN
RAISE EXCEPTION 'includes old password';
RETURN false;
END IF;
RETURN true;
END;
$$;
```

이 기능은 먼저 암호가 최소 5자 이상인지 확인한 다음 새 암호를 이전 암호와 비교합니다. 새 암호가 5자 미만이거나 이전 암호를 포함하는 경우 이 함수는 오류를 발생시킵니다.

- 3) verify_password 함수의 소유권을 postgres 데이터베이스 수퍼유저로 설정합니다.

```
ALTER FUNCTION verify_password(varchar, varchar, varchar) OWNER TO agens;
```

- 4) verify_password 함수를 프로파일에 설정합니다.

```
ALTER PROFILE acctgrp_profile LIMIT PASSWORD_VERIFY_FUNCTION verify_password;
```

- 5) 프로파일을 적용한 사용자(alice)를 만든 다음 잘못된 암호와 유효한 암호를 설정하여 기능이 작동하는지 확인합니다.

```
CREATE ROLE alice WITH LOGIN PASSWORD 'hello' PROFILE acctgrp_profile;
```

- 6) Profile 설정 정보를 확인 합니다.

```
SELECT * FROM ag_profile_view WHERE profile = 'acctgrp_profile';
   profile  | attempts | locktime | lifetime | gracetime | reusetime | reusemax | verifyfunc
-----+-----+-----+-----+-----+-----+-----+
acctgrp_profile | Default | Default | Default | Default | Default | Default | verify_password
(1 row)
```

- 7) 앤리스가 데이터베이스에 연결하여 암호를 변경하려고 할 때, 그녀는 프로파일 기능에 의해 설정된 규칙을 준수해야 합니다. SuperUser가 아닌경우는 암호를 변경할 때 REPLACE 절을 포함해야 합니다.

```
asql -U alice -d postgres
ALTER ROLE alice PASSWORD 'hey';
ERROR: missing REPLACE clause
```

- 8) 앤리스가 데이터베이스에 연결하여 암호를 변경하려고 할 때, 새 암호는 5자 이상이어야 합니다.

```
ALTER USER alice PASSWORD 'hey' REPLACE 'hello';
ERROR: too short
```

```
CONTEXT: PL/pgSQL function verify_password(character varying,character varying,character varying) line 5 at RAISE
```

- 9) 만일 alice 암호를 변경할 경우 새 암호에 이전 암호가 포함되지 않아야 합니다.

```
ALTER USER alice PASSWORD 'helloworld' REPLACE 'hello';
ERROR: includes old password
CONTEXT: PL/pgSQL function verify_password(character varying,character varying,character varying) line 10 at RAISE
```

- 10) 새 암호를 사용할 수 있으면 명령이 오류 없이 완료됩니다.

```
ALTER USER alice PASSWORD 'temp_password' REPLACE 'hello';
```

- 11) 확인 기능을 제거하려면 SuperUser로 접속하여 password_verify_function을 NULL로 설정합니다.

```
ALTER PROFILE acctgrp_profile LIMIT password_verify_function NULL;
그러면 모든 암호 제약이 해제됩니다:
asql -U alice -d postgres
ALTER USER alice PASSWORD 'password' REPLACE 'temp_password';
```

- 12) Profile 및 Password 정보를 확인 합니다.

```
SELECT * FROM ag_profile_view WHERE profile = 'acctgrp_profile';
   profile | attempts | locktime | lifetime | gracetime | reusetime | reusemax | verifyfunc
-----+-----+-----+-----+-----+-----+-----+
acctgrp_profile | Default |
(1 row)

SELECT * FROM ag_password_view WHERE rolename='alice';
   rolename |   profile   | account | passwordsetdate
-----+-----+-----+
alice  | acctgrp_profile | OPEN  | 2023-03-13 12:35:13
(1 row)
```

16.4 How to Alter a Password Profile

ALTER PROFILE 명령을 사용하여 사용자 정의 프로파일을 수정합니다.

SQL은 두 가지 형식의 명령을 지원합니다.

16.4.1 Syntax

- 1) Rename Profile

```
ALTER PROFILE <profile_name> RENAME TO <new_name>;
```

RENAME TO를 사용하여 프로파일 이름을 변경하려면 이름을 변경합니다.

- 2) profile parameters value 변경

```
ALTER PROFILE <profile_name> LIMIT {<parameter value>}[...];
```

LIMIT 절과 하나 이상의 공백으로 구분된 매개 변수/값 쌍을 포함하여 시행하는 규칙을 지정합니다.

16.4.2 Parameters

- profile_name : 프로필 이름을 지정합니다.
- new_name : 새로운 프로필 이름을 지정합니다.
- parameter : 프로파일에 의해 제한되는 특성을 지정합니다.
- value : 매개 변수 제한을 지정합니다.

16.4.3 Examples

- 1) acctgrp_profile이라는 이름의 프로파일을 수정합니다.

```
ALTER PROFILE acctgrp_profile  
LIMIT FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME 1;
```

acctgrp_profile은 로그인 역할이 서버에 연결을 시도할 때 실패한 연결 시도 횟수를 카운트합니다. 프로필은 사용자가 세 번의 시도에서 올바른 암호로 인증되지 않은 경우 계정이 하루 동안 잠기도록 지정합니다.

- 2) acctgrp_profile의 이름을 payables_profile로 변경합니다.

```
ALTER PROFILE acctgrp_profile RENAME TO payables_profile;
```

- 3) profile을 확인 합니다.

```
SELECT * FROM ag_profile_view WHERE profile = 'acctgrp_profile';
profile | attempts | locktime | lifetime | gracetime | reusetime | reusemax | verifyfunc
-----+-----+-----+-----+-----+-----+
(0 rows)

SELECT * FROM ag_profile_view WHERE profile = 'payables_profile';
profile | attempts | locktime | lifetime | gracetime | reusetime | reusemax | verifyfunc
-----+-----+-----+-----+-----+-----+
payables_profile | 3 | 1 | Default | Default | Default | Default |
(1 row)
```

16.5 How to Delete a Password Profile

프로파일을 삭제하려면 DROP PROFILE 명령을 사용합니다.

16.5.1 Syntax

```
DROP PROFILE <profile_name> [CASCADE|RESTRICT];
```

CASCADE 절을 포함하여 현재 프로파일과 연결된 모든 사용자를 DEFAULT 프로파일에 재할당한 다음 프로파일을 삭제합니다. 기본동작은 역할과 관련된 프로파일을 삭제하지 않도록 서버에 지시하는 RESTRICT 입니다.

16.5.2 Parameters

profile_name : 삭제할 프로필의 이름

16.5.3 Examples

- 1) payables_profile이라는 이름의 프로파일을 삭제합니다.

```
DROP PROFILE payables_profile RESTRICT;
ERROR: cannot drop profile payables_profile, 1 user[s] associated with profile.
```

명령의 RESTRICT 절은 프로파일과 연결된 역할이 있는 경우 payables_profile을 삭제하지 않도록 서버에 지시합니다.

- 2) payables_profile이라는 이름의 프로파일을 삭제합니다.

```
DROP PROFILE payables_profile CASCADE;
```

이 명령은 먼저 payables_profile 프로파일과 연결된 모든 역할을 DEFAULT 프로파일과 다시 연결한 다음 payables_profile 프로파일을 삭제합니다.

16.6 How to associating a Profile with an Existing Role/USER

프로필을 생성한 후에는 Alter User... Profile 또는 Alter Role...을 사용할 수 있습니다. 프로파일을 역할과 연결하는 PROFILE 명령입니다.

16.6.1 Syntax

데이터베이스 수퍼유저만 프로파일 관리를 시행하는 ALTER USER|ROLE 절을 사용할 수 있습니다.

```
ALTER USER|ROLE <name> [[WITH] option[...]]
```

여기서 option은 다음과 같은 호환 가능한 절이 될 수 있습니다:

```
PROFILE <profile_name>
| ACCOUNT {LOCK|UNLOCK}
```

- PROFILE 절과 profile_name을 포함하여 미리 정의된 프로파일을 역할과 연결하거나 사용자와 연결된 미리 정의된 프로파일을 변경합니다.
- ACCOUNT 절과 LOCK 또는 UNLOCK 키워드를 포함하여 사용자 계정을 잠금 또는 잠금 해제 상태로 지정합니다.
- 각 로그인 역할에는 프로필이 하나만 있을 수 있습니다. 현재 로그인 역할과 연결된 프로파일을 검색하려면 pg_roles View를 확인 합니다.

16.6.2 Parameters

- name : 프로필의 이름.
- profile_name : User 및 Role과 연결될 프로필의 이름.

16.6.3 Examples

- 1) 사용자와 프로파일 생성

```
CREATE USER joe with password 'password';
CREATE PROFILE acctgrp_profile;
```

- 2) ALTER USER/ROLE... PROFILE 명령을 사용하여 acctgrp라는 프로파일과 joe라는 사용자를 연결합니다.

```
ALTER USER joe PROFILE acctgrp_profile;
or
ALTER ROLE joe PROFILE acctgrp_profile;
```

- 3) ag_password_view 를 이용하여 joe인 사용자의 프로파일을 확인 합니다.

```
SELECT * FROM ag_password_view WHERE rolename='joe';
rolename | profile | account | passwordsetdate
-----+-----+-----+
joe   | acctgrp_profile | OPEN | 2023-03-13 12:42:48
(1 row)
```

16.7 How to Unlock a Locked Account

Super User로 ALTER USER|ROLE... 명령을 사용하여 Profile을 명시적으로 잠그거나 잠금 해제할 수 있습니다.

16.7.1 Syntax

```
ALTER USER|ROLE <name> ACCOUNT {LOCK|UNLOCK}
```

ACCOUNT LOCK 절을 포함하여 역할을 즉시 잠금니다. 잠기면 역할의 LOGIN 기능이 비활성화됩니다.

ACCOUNT UNLOCK 절을 사용하여 역할의 잠금을 해제합니다

16.7.2 Parameters

- **name** : 잠기거나 잠금 해제되는 USER | ROLE 이름입니다.

16.7.3 Examples

- 1) ACCOUNT LOCK 절을 사용하여 joe라는 역할을 잠금니다 . 계정은 ACCOUNT UNLOCK 절로 잠금 해제될 때까지 잠긴 상태로 유지됩니다.

```
ALTER ROLE joe ACCOUNT LOCK;
```

- 2) ag_password_view 를 이용하여 joe인 사용자의 상태를 확인 합니다.

```
SELECT * FROM ag_password_view WHERE rolename='joe';
rolename | profile | account | passwordsetdate
-----+-----+-----+
joe   | acctgrp_profile | LOCKED | 2023-03-13 12:42:48
(1 row)
```

- 3) ACCOUNT UNLOCK 절을 사용하여 joe 라는 역할의 잠금을 해제합니다 .

```
ALTER USER joe ACCOUNT UNLOCK;
```

역할은 데이터베이스 SuperUser가 ACCOUNT UNLOCK 명령을 사용하여 잠금을 해제할 때까지 잠긴 상태로 유지됩니다.

16.8 How to Create a New Role Associated with a Profile

데이터베이스 SuperUser는 CREATE USER | ROLE 명령의 절을 사용하여 역할을 생성할 때 지정된 프로파일을 역할에 할당하거나 역할에 대한 프로파일 관리 세부 정보를 지정할 수 있습니다.

16.8.1 Syntax

데이터베이스 SuperUser만이 프로필 관리를 적용하는 CREATE USER | ROLE 절을 사용할 수 있습니다.

```
CREATE USER|ROLE <name> [[WITH] <option> [...]]
```

여기서 option은 다음과 같은 호환 가능한 절이 될 수 있습니다:
PROFILE <profile_name>
ACCOUNT {LOCK | UNLOCK}

- CREATE ROLE | USER... PROFILE은 연결된 프로필과 함께 새로운 역할을 AgensSQL 데이터베이스 클러스터에 추가합니다.
- PROFILE 절과 profile_name을 포함하여 미리 정의된 프로파일을 역할과 연결하거나 사용자와 연결된 미리 정의된 프로파일을 변경합니다.
- ACCOUNT 절과 LOCK 또는 UNLOCK 키워드를 포함하여 사용자 계정을 잠금 또는 잠금 해제 상태로 지정합니다.
- 각 로그인 역할에는 프로필이 하나만 있을 수 있습니다. 현재 로그인 역할과 연결된 프로파일을 검색하려면 pg_roles View를 확인 합니다.

16.8.2 Parameters

- name : 역할/사용자의 이름입니다.
- profile_name : 역할과 연결된 프로필의 이름입니다.

16.8.3 Examples

- 1) CREATE USER / ROLE를 사용하여 acctgrp_profile 프로필 과 연결된 smith라는 로그인 역할을 만듭니다.

```
CREATE USER smith PROFILE acctgrp_profile PASSWORD 'secretpwd';
or
CREATE ROLE smith PROFILE acctgrp_profile LOGIN PASSWORD 'secretpwd';
```

smith는 암호 secretpwd를 사용하여 서버에 로그인할 수 있습니다.

- 2) ag_password_view 를 이용하여 smith인 사용자의 프로파일을 확인 합니다.

```
SELECT * FROM ag_password_view WHERE rolename='smith';
rolename | profile | account | passwordsetdate
-----+-----+-----+
smith  | acctgrp_profile | OPEN | 2023-03-13 12:46:48
(1 row)
```

17장 . Data Redaction

Data Redaction은 특정 사용자에게 표시되는 데이터를 동적으로 변경하여 민감한 데이터 노출을 제한합니다. CREATE REDACTION POLICY 명령을 사용하여 Data Redaction 정책을 등록 합니다. 이 명령은 정책이 적용되는 테이블, 열, 사용자를 결정 및 기타 옵션을 지정합니다. SuperUser와 테이블 소유자는 데이터 편집을 건너뛰고 원본 데이터를 볼수 있습니다. 다른 모든 사용자는 수정 정책이 적용되어 제한된 데이터를 볼 수 있습니다.

17.1 CREATE REDACTION POLICY

CREATE REDACTION POLICY는 테이블에 대한 새로운 데이터 수정 정책을 정의합니다.

17.1.1 Syntax

```
CREATE REDACTION POLICY <name> ON <table_name>
[ FOR ( <expression> ) ]
[ ADD [ COLUMN ] <column_name> USING <funcname_clause> ]
```

CREATE REDACTION POLICY 명령은 열 데이터를 교정하여 테이블에 대한 Redaction 보안 정책을 정의합니다. 새로 생성된 Data Redaction 정책은 기본적으로 활성화됩니다.

FOR : 대상 User | Role 조건식.

ADD COLUMN : 대상 COLUMN

USING : Redaction 함수

17.1.2 Parameters

- name : Data Redaction Policy명
- table_name : Data Redaction Table명
- expression: 제한 User 정책 조건
- column_name : Data Redaction Column명
- funcname_clause : Data Redaction Function명

17.1.3 Examples

1) Data Redaction 사전 준비 (Table, Data, User, Function)

```
# employees Table 생성
CREATE TABLE employees (
    id      integer GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    name    varchar(40) NOT NULL,
    ssn     varchar(11) NOT NULL,
    phone   varchar(10),
    birthday date,
    salary   money,
    email    varchar(100)
);

# Data Insert
INSERT INTO employees (name, ssn, phone, birthday, salary, email)
VALUES
    ('Sally Sample', '020-78-9345', '5081234567', '1961-02-02', 51234.34,
    'sally.sample@enterprisag.com'),
    ('Jane Doe', '123-33-9345', '6171234567', '1963-02-14', 62500.00,
    'jane.doe@gmail.com'),
    ('Bill Foo', '123-89-9345', '9781234567', '1963-02-14', 45350,
    'william.foe@hotmail.com');

# User 생성 및 권한 부여 (hr, scott)
CREATE USER hr with password '1234';
CREATE USER scott with password '1234';
GRANT ALL ON employees TO hr, scott;

# SSN Column Redaction 함수 생성
CREATE OR REPLACE FUNCTION redact_ssn (ssn varchar(11))
RETURNS varchar(11) AS $$$
BEGIN
    /* replaces 020-12-9876 with xxx-xx-9876 */
    return overlay (ssn placing 'xxx-xx' from 1);
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION redact_ssn_new (ssn varchar(11))
RETURNS varchar(11) AS $$$
BEGIN
    /* replaces 020-12-9876 with ***-**-9876 */
    return overlay (ssn placing '***-**' from 1);
END;
$$ LANGUAGE plpgsql;

# Salary Column Redaction 함수 생성
CREATE OR REPLACE FUNCTION redact_salary ()
RETURNS money AS $$
```

```
BEGIN  
return 0::money;  
END;  
$$ LANGUAGE plpgsql;
```

2) Data Redaction POLICY 생성

```
CREATE REDACTION POLICY redact_policy_personal_info ON employees FOR (session_user != 'hr')  
ADD COLUMN ssn USING redact_ssn(ssn),  
ADD COLUMN salary USING redact_salary();
```

- Table : employees
- 제한 조건 : hr User 제외
- Redaction Column : SSN, SALARY

3) hr User를 이용하여 employees Table 조회

```
# hr user 접속  
\c postgres hr  
  
# employees Table 조회  
SELECT * FROM EMPLOYEES;  
id | name | ssn | phone | birthday | salary | email  
---+-----+-----+-----+-----+-----+  
1 | Sally Sample | 020-78-9345 | 5081234567 | 1961-02-02 | ₩51,234 |  
sally.sample@enterprisag.com  
2 | Jane Doe | 123-33-9345 | 6171234567 | 1963-02-14 | ₩62,500 | jane.doe@gmail.com  
3 | Bill Foo | 123-89-9345 | 9781234567 | 1963-02-14 | ₩45,350 |  
william.foe@hotmail.com  
(3 rows)
```

4) scott User를 이용하여 employees Table 조회

```
# scott user 접속  
\c postgres scott  
  
# employees Table 조회  
SELECT * FROM EMPLOYEES;  
id | name | ssn | phone | birthday | salary | email  
---+-----+-----+-----+-----+-----+  
1 | Sally Sample | xxx-xx-9345 | 5081234567 | 1961-02-02 | ₩0 | sally.sample@enterprisag.com
```

```
2 | Jane Doe  | xxx-xx-9345 | 6171234567 | 1963-02-14 | 0 | jane.doe@gmail.com
3 | Bill Foo  | xxx-xx-9345 | 9781234567 | 1963-02-14 | 0 | william.foe@hotmail.com
(3 rows)
```

17.2 ALTER REDACTION POLICY

ALTER REDACTION POLICY는 테이블에 대한 Data Redaction Policy를 변경합니다.

17.2.1 Syntax

```
ALTER REDACTION POLICY <name> ON <table_name> RENAME TO <new_name>

ALTER REDACTION POLICY <name> ON <table_name> FOR ( <expression> )

ALTER REDACTION POLICY <name> ON <table_name> { ENABLE | DISABLE }

ALTER REDACTION POLICY <name> ON <table_name>
ADD [ COLUMN ] <column_name> USING <funcname_clause>

ALTER REDACTION POLICY <name> ON <table_name>
MODIFY [ COLUMN ] <column_name> USING <funcname_clause>

ALTER REDACTION POLICY <name> ON <table_name>
DROP [ COLUMN ] <column_name>
```

FOR : 대상 User | Role 조건식을 변경합니다.

ENABLE : Redaction 정책을 활성화 합니다.

DISABLE : Redaction 정책을 비활성화합니다.

ADD COLUMN : COLUMN에 Redaction 기능을 추가 합니다.

MODIFY COLUMN : COLUMN의 Redaction 기능을 변경 합니다.

DROP COLUMN : COLUMN을 Redaction기능을 제외 합니다.

17.2.2 Parameters

- name : Data Redaction Policy명
- table_name : Data Redaction Table명
- new name : 변경할 새로운 Data Redaction Policy명
- expression: 제한 User 정책 조건
- column_name : Data Redaction Column명
- funcname_clause : Data Redaction Function명

17.2.3 Examples

- 1) 제한 User 조건을 수정 합니다. (hr, manager 정책 해제)

```
ALTER REDACTION POLICY redact_policy_personal_info ON employees  
FOR (session_user != 'hr' AND session_user != 'manager');
```

- 2) ssn에 대한 Redaction Function을 redact_ssn_new로 변경합니다.

```
ALTER REDACTION POLICY redact_policy_personal_info ON employees  
MODIFY COLUMN ssn USING redact_ssn_new(ssn);
```

17.3 DROP REDACTION POLICY

DROP REDACTION POLICY는 테이블에서 Redaction 정책을 제거합니다.

17.3.1 Syntax

```
DROP REDACTION POLICY <name> ON <table_name>
```

17.3.2 Parameters

- name : Data Redaction Policy 명
- table_name : Data Redaction Table 명

17.3.3 Examples

- 1) employees 테이블에서 redact_policy_personal_info 라는 데이터 수정 정책을 삭제 합니다.

```
DROP REDACTION POLICY redact_policy_personal_info ON employees;
```

18장. 감사기능 (Audit)

18.1 개요

AgensSQL은 데이터베이스 보안 관리자 및 운영자가 감사 로깅 기능을 사용하여 데이터베이스 활동을 추적하고 분석할 수 있도록 합니다. AgensSQL 감사 로깅은 지정된 데이터베이스 이용 정보를 포함한 감사 로그 파일을 생성합니다.

postgresql.conf 또는 postgresql.auto.conf에 지정된 매개변수로 Audit 기능을 제어 할 수 있습니다.

18.1.1 특징

- 1) Audit 범위
 - ag_audit_statement
 - ag_audit_connect
 - ag_audit_disconnect
- 2) Audit Log 및 Directory
 - ag_audit_directory
 - ag_audit_filename
- 3) Audit Log File 관리
 - ag_audit_rotation_size
 - ag_audit_rotation_age
- 4) Audit Tag
 - ag_audit_tag

18.2 Parameters

postgresql.conf의 매개변수를 사용하여 데이터베이스 Audit를 구성할 수 있습니다. Audit 기능을 사용하기 위해선 기본적으로 logging_collector 설정이 on으로 설정 되어 있어야 합니다. Audit 기능의 경우 서비스 부하 및 Log Disk Full 등의 이슈가 있으므로 서비스에 모니터링이 가능한 경우에만 적용을 권고 합니다.

- **ag_audit**

데이터베이스 감사 기능을 활성화하거나 비활성화합니다. csv 값은 데이터베이스 감사 기능을 활성화합니다. 이러한 값은 감사 정보가 저장되는 파일 형식을 나타냅니다. 기본값은 **none**으로 데이터베이스 감사 기능을 비활성화합니다.

- **ag_audit_directory**

Audit 로그 파일을 저장 디렉터리를 지정합니다. 디렉터리의 경로는 데이터 폴더에 상대적이거나 절대적일 수 있습니다. 기본값은 ag_audit으로 \$PGDATA/ag_audit 디렉토리입니다.

- **ag_audit_filename**

감사 정보가 저장될 감사 파일의 파일 이름을 지정합니다. 기본 파일 이름은 ag-audit-%Y%m%d_%H%M%S입니다.

- **ag_audit_rotation_size**

파일 크기에 따라 로그파일의 교체가 발생합니다. 주석 처리하거나 0으로 설정하면 파일 크기에 따른 파일 교체가 발생하지 않습니다. 단위는 Kb이며 기본값은 0입니다.

- **ag_audit_rotation_age**

로그파일 교체가 발생하는 시간(분)을 지정합니다. 이 기능을 사용하지 않으려면 이 매개 변수를 0으로 설정하면 됩니다. 기본값은 0입니다.

- **ag_audit_connect**

데이터베이스 연결 시도에 대한 감사 범위를 설정합니다. 연결 시도에 대한 감사를 사용하지 않으려면 none으로 설정합니다. 실패한 연결 시도를 감사하려면 값을 failed 값으로 설정합니다. 모든 연결 시도를 감사하려면 값을 all 값으로 설정합니다. 기본값은 failed입니다.

- **ag_audit_disconnect**

연결된 사용자가 데이터베이스 연결 끊김을 감사할 수 있습니다. 연결 끊김 감사를 사용하려면 값을 all 값으로 설정합니다. 사용하지 않으려면 값을 none로 설정합니다. 기본값은 none입니다.

- **ag_audit_statement**

세션 감사를 기록할 명령문 클래스를 지정합니다. 가능한 값은 다음과 같습니다

- READ: SELECT, COPY
- WRITE: INSERT, UPDATE, DELETE, TRUNCATE, COPY
- FUNCTION : 함수 호출 및 DO 블록
- ROLE : 역할 및 권한과 관련된 명령문 (예: GRANT, REVOKE, CREATE/ALTER/Drop ROLE)

- DDL : ROLE 클래스에 포함되지 않은 모든 DDL
 - MISC : 기타 명령(예: DISCOVER, FETCH, CHECKPOINT, VACUUM, SET).
 - MISC_SET : 기타 SET 명령(예: SET ROLE)
 - ERROR: 모든 데이터베이스 오류
 - ALL: 위의 내용을 모두 포함합니다
- 쉼표로 구분된 목록을 사용하여 여러 클래스를 제공할 수 있습니다. 기본값은 DDL,ERROR입니다.

- **ag_audit_tag**

이 구성 매개 변수를 사용하여 각 항목에 대한 감사 로그 파일에 포함될 문자열 값을 추적 태그로 지정합니다.

18.3 Audit Log File

감사 로그 파일은 ag_audit 구성 매개 변수의 설정에 따라 CSV 형식으로 생성됩니다. 다음 표에는 Audit 로그의 csv Logfile에 나타나는 순서대로 필드가 나열되어 있습니다.

(예시 Log

2023-03-02 16:17:31.859

KST,"agens","postgres",9819,"210.104.181.77:35837",64004d8b.265b,3,"SELECT",2023-03-02 16:17:31
KST,3/4,0,AUDIT,00000,"select * from test;","","SESSION",,,,,,,,"asql!","READ","AgensAudit","statement")

Field	Description
log_time	시간 (ex. 2023-03-02 16:17:31.859 KST)
User	접속 User (ex. agens)
Database	접속 Database (ex. postgres)
process_id	OS Process_id (ex. 9819)
Host	접속 Host 정보 (ex. "210.104.181.77:35837")
session_id	접속 Session Id (ex. 64004d8b.265b)
session_line_num	세션내의 작업 순서 (ex. 3)
process_status	처리 상태 (ex. SELECT)
session_start_time	세션이 시작된 시간 (2023-03-02 16:17:31 KST)

virtual_transaction_id	가상의 트랜잭션 ID (ex. 3/4)
transaction_id	트랜잭션 ID (ex. 0)
error_severity	오류여부, Error의 경우 Error 아닐경우 AUDIT (ex. AUDIT)
sql_state_code	SQL 반환 코드 (ex. 00000)
message	감지한 SQL (ex. select * from test;)
audit_type	감사 유형 (ex. SESSION)
query	Error의 경우 Error SQL
query_pos	Error의 경우 Error 위치
application_name	접속 AP명 (ex. asql)
command_tag	작업 형태 (ex. READ)
audit_tag	설정된 Audit 태그 (ex. AgensAudit)
type	감지된 Audit 타입 (ex. statement)

18.3.1 Examples

postgresql.conf 파일에 다음과 같은 구성이 있는 경우

```
logging_collector = on
ag_audit = 'csv'
ag_audit_connect = 'all'
ag_audit_disconnect = 'all'
ag_audit_statement = 'all'
ag_audit_tag = 'AgensAudit'
```

asql을 이용하여 Audit을 확인 합니다. (Audit 경로 \$PGDATA/ag_audit/)

- Query


```
# asql -U agens -d postgres -h 210.104.181.77 -p 5432
asql (13.7)
Type "help" for help.
```
- Audit Log


```
2023-03-02 17:03:46.625
KST,"agens","postgres",10268,"210.104.181.77:35841",64005862.281c,1,"authentication",2023-03-02
17:03:46 KST,3/34,0,AUDIT,00000,"connection authorized: user=agens database=postgres
```

```
application_name=asql" ,,,,"AgensAudit","connect"

    • Query
postgres=# SHOW ag_audit_connect;
ag_audit_connect
-----
all
(1 row)

    • Audit Log
2023-03-02 17:03:46.625
KST,"agens","postgres",10268,"210.104.181.77:35841",64005862.281c,2,"SHOW",2023-03-02
17:03:46 KST,3/35,0,AUDIT,00000,"SHOW
ag_audit_connect;","","SESSION",,,,"asql","MISC","AgensAudit","statement"

    • Query
postgres=# SHOW ag_audit_statement;
ag_audit_statement
-----
all
(1 row)

    • Audit Log
2023-03-02 17:03:46.625
KST,"agens","postgres",10268,"210.104.181.77:35841",64005862.281c,3,"SHOW",2023-03-02
17:03:46 KST,3/36,0,AUDIT,00000,"SHOW
ag_audit_statement;","","SESSION",,,,"asql","MISC","AgensAudit","statement"

    • Query
postgres=# CREATE ROLE adminuser;
CREATE ROLE

    • Audit Log
2023-03-02 17:03:46.625
KST,"agens","postgres",10268,"210.104.181.77:35841",64005862.281c,4,"CREATE ROLE",2023-03-02
17:03:46 KST,3/38,541,AUDIT,00000,"CREATE ROLE
adminuser;","","SESSION",,,,"asql","ROLE","AgensAudit","statement"

    • Query
postgres=# ALTER ROLE adminuser WITH LOGIN, SUPERUSER, PASSWORD 'password';
ERROR: syntax error at or near ","
LINE 1: ALTER ROLE adminuser WITH LOGIN, SUPERUSER, PASSWORD 'passwo...
                                     ^
    • Audit Log
2023-03-02 17:03:46.625
KST,"agens","postgres",10268,"210.104.181.77:35841",64005862.281c,5,"idle",2023-03-02 17:03:46
KST,3/39,0,ERROR,42601,"syntax error at or near "",","","ALTER ROLE adminuser WITH LOGIN,
SUPERUSER, PASSWORD 'password';",32,,,"asql","","AgensAudit","error"

    • Query
postgres=# ALTER ROLE adminuser WITH LOGIN SUPERUSER PASSWORD 'password';
```

ALTER ROLE

- Audit Log

2023-03-02 17:03:46.625

KST,"agens","postgres",10268,"210.104.181.77:35841",64005862.281c,6,"ALTER ROLE",2023-03-02 17:03:46 KST,3/40,542,AUDIT,00000,"ALTER ROLE adminuser WITH LOGIN SUPERUSER PASSWORD <REDACTED>","","SESSION",,,,,,,,"asql","","ROLE","AgensAudit","statement"

- Query

```
postgres=# CREATE DATABASE auditdb;
CREATE DATABASE
```

- Audit Log

2023-03-02 17:03:46.625

KST,"agens","postgres",10268,"210.104.181.77:35841",64005862.281c,7,"CREATE DATABASE",2023-03-02 17:03:46 KST,3/41,543,AUDIT,00000,"CREATE DATABASE auditdb;","","SESSION",,,,,,,,"asql","","DDL","AgensAudit","statement"

- Query

```
postgres=# ALTER DATABASE auditdb OWNER TO adminuser;
ALTER DATABASE
```

- Audit Log

2023-03-02 17:03:46.625

KST,"agens","postgres",10268,"210.104.181.77:35841",64005862.281c,8,"ALTER DATABASE",2023-03-02 17:03:46 KST,3/42,544,AUDIT,00000,"ALTER DATABASE auditdb OWNER TO adminuser;","","SESSION",,,,,,,,"asql","","DDL","AgensAudit","statement"

- Query

```
postgres=# \c auditdb adminuser
```

Password for user adminuser:

You are now connected to database "auditdb" as user "adminuser".

- Audit Log

2023-03-02 17:17:18.093

KST,"adminuser","auditdb",10403,"210.104.181.77:35843",64005b8e.28a3,1,"authentication",2023-03-02 17:17:18 KST,4/56,0,AUDIT,00000,"connection authorized: user=adminuser database=auditdb application_name=asql",,,,,,,,"AgensAudit","connect"

2023-03-02 17:03:46.625

KST,"agens","postgres",10268,"210.104.181.77:35841",64005862.281c,9,"idle",2023-03-02 17:03:46 KST,,0,AUDIT,00000,"disconnection: session time: 0:13:31.472 user=agens database=postgres host=210.104.181.77 port=35841",,,,,,,,"asql","","AgensAudit","disconnect"

- Query

```
auditdb=# CREATE SCHEMA agdb;
CREATE SCHEMA
```

- Audit Log

2023-03-02 17:17:18.093

KST,"adminuser","auditdb",10403,"210.104.181.77:35843",64005b8e.28a3,2,"CREATE SCHEMA",2023-03-02 17:17:18 KST,4/57,545,AUDIT,00000,"CREATE SCHEMA

```
agdb;,,, "SESSION",,,,,, "asql","","DDL","AgensAudit","statement"
```

- Query

```
auditdb=# SET search_path TO agdb;  
SET
```

- Audit Log

```
2023-03-02 17:17:18.093  
KST,"adminuser","auditdb",10403,"210.104.181.77:35843",64005b8e.28a3,3,"SET",2023-03-02  
17:17:18 KST,4/58,0,AUDIT,00000,"SET search_path TO  
agdb;,,, "SESSION",,,,,, "asql","","MISC","AgensAudit","statement"
```

- Query

```
auditdb=# CREATE TABLE department (  
deptno      INTEGER NOT NULL CONSTRAINT dept_pk PRIMARY KEY,  
dname       VARCHAR(14) CONSTRAINT dept_dname_uq UNIQUE,  
loc         VARCHAR(13)  
);  
CREATE TABLE
```

- Audit Log

```
2023-03-02 17:17:18.093  
KST,"adminuser","auditdb",10403,"210.104.181.77:35843",64005b8e.28a3,4,"CREATE  
TABLE",2023-03-02 17:17:18 KST,4/59,546,AUDIT,00000,"""CREATE TABLE department (  
deptno      INTEGER NOT NULL CONSTRAINT dept_pk PRIMARY KEY,  
dname       VARCHAR(14) CONSTRAINT dept_dname_uq UNIQUE,  
loc         VARCHAR(13)  
);""",,"SESSION",,,,,, "asql","","DDL","AgensAudit","statement"
```

- Query

```
auditdb=# \q
```

- Audit Log

```
2023-03-02 17:17:18.093  
KST,"adminuser","auditdb",10403,"210.104.181.77:35843",64005b8e.28a3,5,"idle",2023-03-02  
17:17:18 KST,,0,AUDIT,00000,"disconnection: session time: 0:04:33.816 user=adminuser  
database=auditdb host=210.104.181.77 port=35843",,,,,,,,,,, "asql","","AgensAudit","disconnect"
```

VII. 추가기능

19장. AHM

19.1. 제품 소개

AHM은 PostgreSQL, AgensSQL과 AgensGraph 데이터베이스 서버에서 발생하는 SPOF (Single point of failure)를 해결하기 위해 Bitnine에서 개발한 고가용성 컴포넌트입니다. AHM은 분산 메커니즘을 사용하여 Primary 또는 Standby 데이터베이스의 장애를 감지하고 서비스 연속성을 보장하기 위해 자동으로 적절한 조치를 취합니다.

AHM은 클라이언트 응용 프로그램에 대한 데이터베이스 서버의 가용성을 보장하는 프로그램으로, 각 Primary 및 Standby 데이터베이스와 함께 설치됩니다.

Primary 데이터베이스 및 시스템에 장애가 발생할 경우 AHM은 서로 조정하여 사용 가능한 Standby 데이터베이스 서버 중 하나를 새 Primary 데이터베이스로 지정 한 뒤 이전 Primary 서버에서 VIP(가상 IP)를 기동하여 빠르게 서비스를 정상화 합니다.

19.1.1 Features

- Failover 관리
- 클라이언트 응용 프로그램에 단일 접속 제공(가상 IP 사용)
- 각 Failover 관련 결정에 대해 합의를 구축
- Database and AHM Health check
- 클러스터 상태 관리
- User가 제공한 Pre 및 Post 스크립트 실행

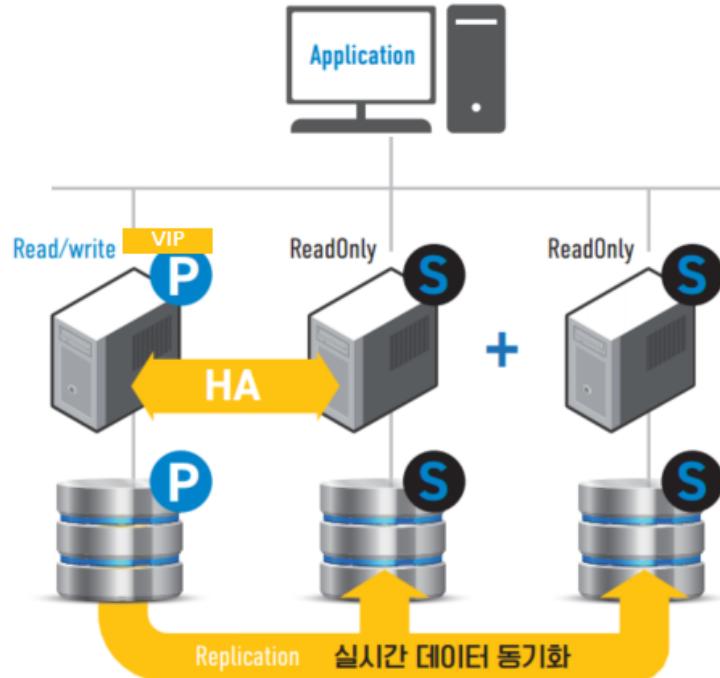
19.1.2 AHM Cluster 구성 요소

- Primary AgensSQL 서버
- 1 또는 그 이상의 스트리밍 복제된 Standby AgensSQL 서버
- 각 Primary 및 Standby 서버에 AHM
- PRIMARY IP, STANDBY IP, VIP 3개의 IP 필요
- 0 또는 그 이상의 AHM witness 노드

AgensSQL Standby 서버 수는 가용성 및 재해 복구 요구 사항에 따라 달라집니다. AHM은 쿼럼의 다수결 메커니즘을 사용하여 리더를 선택하고 Failover를 수행하기 때문에 가용성 요구 사항 및 필수 데이터 보존 정책에 따라 허수의 AHM 노드를 사용하는 것이 좋습니다.

-일반적인 AHM 클러스터

일반적인 AHM 클러스터는 AgensSQL Primary 와 Standby 서버로 구성되어 있습니다.



19.2 AHM Architecture

AHM 노드가 기동 되면 로컬 데이터베이스에 연결하여 데이터베이스 상태를 확인합니다.

- AHM이 데이터베이스에 연결할 수 없는 경우, 리더가 선정될 수 없습니다.
- 데이터베이스가 복구모드로 확인되면 AHM은 Standby 역할을 수행합니다.
- 데이터베이스가 복구모드가 아닌 경우 AHM은 Primary역할을 수행합니다.
- Primary 데이터베이스가 있는 AHM 노드는 AHM 클러스터의 과반수가 동의하면 VIP를 획득합니다.
- Primary AHM은 Standby 노드의 우선 순위에 따라 Successor 노드를 선택합니다.
- 리더에 장애가 발생 할 경우 Successor 노드는 새로운 리더가 됩니다.
- Failover가 발생할 경우 새로 선택된 AHM 노드는 로컬 데이터베이스 서버에서 노드 Promotion을 수행합니다.

** Promotion : Standby로 대기상태에 있는 Database서버가 Primary 로 활성화 되는 과정

19.3 AHM 설치

AHM은 Linux에서 실행됩니다. AHM을 설치 파일은 Bitnine 홈페이지(www.bitnine.net)에 접속하여 설치 파일을 다운로드하여 진행 할 수 있습니다.

19.3.1 AHM 설치 최소 사양

CPU	1 GHz 이상 프로세서
RAM	2GB 이상
DISK	1GB 이상 여유 공간 (AgensSQL 설치 파일을 제외한 DATA 공간이 필요합니다)
OS	CentOS 7 이상

19.3.2 사전 빌드된 바이너리 파일을 사용하여 설치

1. AHM 설치 User 생성하기
`# useradd agens`
2. user 암호 설정
`# passwd agens`
3. 생성된 agens 계정으로 Binary 압축 해제
`$ tar -xvf AHM_1.0.tar.gz`
4. 환경 파일 설정
`export PATH=/home/agens/AHM_1.0/bin:$PATH`
5. 환경 파일 적용
`$ source ~/.bash_profile`

19.4 AHM 구성

AHM는 yaml 형식의 AHM 구성 파일이 필요합니다. 기본적으로 /usr/local/etc/ directory에서 ahm_config.yml 파일을 불러옵니다. 또는 ahm 바이너리에 -f 명령어를 사용하여 구성 파일 경로를 제공할 수 있습니다.

19.4.1 ahm_config 파라미터 정보

파라미터 명	설명
conn_string	로컬 PG 서버에 대한 연결정보
node_name	AHM 노드 이름, postgresql.conf conninfo의 application_name과 동일하게 설정
nodes_filename	AHM 노드 목록을 파일(AHM_HOME/etc/nodes.conf) 경로
is_witness	Witness 노드여부
priority	Fail Over시 사용되는 노드 우선 순위 (낮을수록 우선)
ahm_port	AHM 노드에 대한 TCP/IP 수신 포트 (기본 포트 : 9500)
ipc_port	IPC에 대한 수신 포트 (기본 포트 : 9595)
data_dir	AHM 데이터 디렉토리 (/tmp)
socket_dir	AHM 소켓파일이 생성될 경로 (/tmp)
log_filename	로그 파일 이름
log_prefix	로그 표기 방식
log_verbose	상세 로그 출력을 활성화합니다
health_check_period	Health check 주기 (단위 : 초)
max_consec_failures	최대 연속 health check 실패 대기 횟수(이후 FailOver 진행)
db_user	데이터베이스 접속 사용자명
db_database	데이터베이스명
db_port	AgensSQL 포트

db_bin_dir	ag_ctl이 포함된 디렉토리 (\$AGHOME/bin)
db_data_dir	데이터 디렉토리 (\$PGDATA)
db_config_dir	데이터베이스 configuration 디렉터리
pre_Promotion_command	FailOver 전에 실행할 스크립트
Promotion_command	FailOver 사용자 정의 명령어
post_Promotion_command	FailOver 후 실행할 스크립트
virtual_IP	VIP 주소
if_cmd_path	If 파일 경로 (/bin)
if_up_cmd	VIP 기동 명령어
if_down_cmd	VIP 정지 명령어
arping_cmd	arping 명령어
arping_path	arping 파일 경로 (/usr/sbin)
ping_path	ping 파일 경로
follow_command	Primary 변경 시 Follow Primary 명령어
service_restart_command	ahm 재시작 명령어
restore_command	아카이브 복구 명령어
archive_cleanup_command	재시작 지점 후 정리 명령어
use_recovery_min_apply_delay	복구 구성에서 recovery_min_apply_delay를 활성화/비활성화
recovery_min_apply_delay	복구 변경 사항을 적용하는 최소 지연 시간
db_connect_timeout	데이터베이스 서버에 대한 연결 시간 초과
primary_conninfo_options	primary_conninfo 복구에 대한 추가 옵션
standby_startup_delay	Standby 기동 대기 시간

use_pg_rewind	pg_rewind을 이용한 복구 설정
beacon_message_interval_seconds	노드 상태 확인 주기 (단위 : 초), AHM 노드 장애를 감지하기 위한 내부 상태 점검 메커니즘으로도 사용됩니다.
beacon_message_retry_count	장애 노드로 변경하기 위한 노드 상태 체크 실패 횟수
promote_leader_timeout_seconds	Standby에서 Primary로 promote시키는 명령에 대한 타임아웃
wait_for_primary_timeout_seconds	시작 시 클러스터가 Primary 노드를 기다리는 대기 시간
pid_file_name	AHM PID 파일명(ahm.pid)
failover_consensus_with_half_votes	2노드 만으로 FailOver 활성화

19.4.2 Ahm Failover Interval

- AHM 장애시

항목	파라미터	설정 값 (디폴트)
AHM 상태 체크	beacon_message_interval_seconds	8초 주기
AHM 상태 체크 재시도	beacon_message_retry_count	2회 반복
Primary 기동 대기	wait_for_primary_timeout_seconds	20초
Failover Interval		약 36초 ~ 44초

- Database 장애시

항목	파라미터	설정 값 (디폴트)
Database 상태 체크	health_check_period	30초 주기
Database 응답대기	db_connect_timeout	5초
Database 상태 체크 재시도	max_consec_failures	5회
Primary 기동 대기	wait_for_primary_timeout_seconds	20초

Failover Interval	195초 ~ 230초
-------------------	-------------

- **Example:**

노드 별로 각각 진행

-ahm_config.yml 구성

```
> cat ahm_config.yml
is_witness: false
priority : 1
data_dir: '/tmp/' #directory to hold temporary ahm data and pid file
#log_severity: DEBUG

ahm_port: 9500
ipc_port: 9595
socket_dir: /tmp
#nodes_filename: "#default CWD/nodes.conf"
node_name: AHM-NODE-1
conn_string: host=localhost port=5432 dbname=postgres user=agens

beacon_message_interval_seconds: 8
beacon_message_retry_count: 2
promote_leader_timeout_seconds: 20
wait_for_primary_timeout_seconds: 20

#
# PG Database Health Check Configurations
#
db_connect_timeout: 5
health_check_period: 30
max_consec_failures: 5

#
# Virtual IP configurations
#
virtual_IP: '210.104.181.42'
if_cmd_path: '/bin'
if_up_cmd: '/usr/bin/sudo /sbin/ip addr add ${IP}/24 dev eno1 label eno1:0'
if_down_cmd: '/usr/bin/sudo /sbin/ip addr del ${IP}/24 dev eno1'
arping_path: '/usr/sbin'
arping_cmd: '/usr/bin/sudo /usr/sbin/arping -U ${IP} -w 1 -l eno1'
ping_path : '/bin'
```

```
#  
# Logging configurations  
#  
log_filename: ""  
log_prefix: '%l PID:%p %t'  
log_verbose: false #true value  
  
#  
# Database cluster configuration  
#  
db_user: 'agens'  
db_database: 'postgres'  
db_port: 5432  
db_bin_dir: '/home/agens/AgensSQL-2.13.7.0/bin'  
db_data_dir: '/home/agens/AgensSQL-2.13.7.0/db_cluster'  
db_config_dir: '/home/agens/AgensSQL-2.13.7.0/db_cluster'  
  
#  
# Promotion configurations  
#  
pre_promotion_command: '/home/agens/AHM/etc/failover.sh'  
promotion_command: ""  
post_promotion_command: ""  
  
#  
# Failover configurations  
#  
#follow_command:""  
standby_startup_delay: 3  
use_pg_rewind: false  
failover_consensus_with_half_votes: true  
  
#  
# Replication configurations  
#  
restore_command: ""  
archive_cleanup_command: ""  
use_recovery_min_apply_delay: false  
recovery_min_apply_delay: 0  
primary_conninfo_options: ""  
  
pid_file_name : 'ahm.pid'
```

-Failover Script 구성

```
> cat ahm_config.yml
is_witness: false
priority : 1
data_dir: '/tmp/' #directory to hold temporary ahm data and pid file
#log_severity: DEBUG

ahm_port: 9500
ipc_port: 9595
socket_dir: /tmp
#nodes_filename: "" #default CWD/nodes.conf
node_name: AHM-NODE-1
conn_string: host=localhost port=5432 dbname=postgres user=agens

beacon_message_interval_seconds: 8
beacon_message_retry_count: 2
promote_leader_timeout_seconds:      20
wait_for_primary_timeout_seconds:   20

#
#    PG Database Health Check Configurations
#
db_connect_timeout: 5
health_check_period: 30
max_consec_failures: 5

#
#    Virtual IP configurations
#
virtual_IP: '210.104.181.42'
if_cmd_path: '/bin'
if_up_cmd: '/usr/bin/sudo /sbin/ip addr add ${IP}/24 dev eno1 label eno1:0'
if_down_cmd: '/usr/bin/sudo /sbin/ip addr del ${IP}/24 dev eno1'
arping_path: '/usr/sbin'
arping_cmd: '/usr/bin/sudo /usr/sbin/arping -U ${IP} -w 1 -l eno1'
ping_path : '/bin'

#
#    Logging configurations
```

```
#  
log_filename: ""  
log_prefix: '%l PID:%p %t'  
log_verbose: false #true value  
  
#  
# Database cluster configuration  
#  
db_user: 'agens'  
db_database: 'postgres'  
db_port: 5432  
db_bin_dir: '/home/agens/AgensSQL-2.13.7.0/bin'  
db_data_dir: '/home/agens/AgensSQL-2.13.7.0/db_cluster'  
db_config_dir: '/home/agens/AgensSQL-2.13.7.0/db_cluster'  
  
#  
# Promotion configurations  
#  
pre_promotion_command: '/home/agens/AHM/etc/failover.sh'  
promotion_command: ""  
post_promotion_command: ""  
  
#  
# Failover configurations  
#  
#follow_command:""  
standby_startup_delay: 3  
use_pg_rewind: false  
failover_consensus_with_half_votes: true  
  
#  
# Replication configurations  
#  
restore_command: ""  
archive_cleanup_command: ""  
use_recovery_min_apply_delay: false  
recovery_min_apply_delay: 0  
primary_conninfo_options: ""  
  
pid_file_name : 'ahm.pid'
```

- **Ahm** 기동

```
> ahm -f /home/agens/etc/ahm_config.yml
```

19.4.3 노드 구성

노드 구성 파일에는 클러스터에 있는 모든 노드의 주소가 포함되어 있습니다.

파일은 IP:PORT 형식을 사용하여 AHM 노드 주소를 나타내고 각 라인에 정확히 하나의 IP:PORT 항목이 들어갑니다.

예를 들어, 2개의 노드 AHM 클러스터의 nodes.conf 파일은 다음과 같습니다.

```
> cat nodes.conf
# IP:port list of AHM nodes
# one ip:port pair per line
# empty and commented lines will be ignored

210.104.181.77:9500
210.104.181.78:9500
```

기본적으로 ahm은 현재 작업 디렉토리에서 nodes.conf 파일을 찾고, nodes.conf의 경로는 **nodes_filename** 구성 매개변수를 사용하여 ahm 구성 파일에서 구성할 수도 있습니다.

Note

모든 AHM 노드에 대해 동일한 nodes.conf 파일을 사용해도 됩니다. AHM은 로컬 노드에 대한 항목을 자동으로 감지하고 원격 노드 주소만 사용하여 클러스터를 생성합니다.

19.5 AHM 실행

AHM을 실행하려면 ahm binary를 실행하세요.

```
ahm [ -f CONFIG_FILE ] [ -d ] [ -n ] [stop|start]
```

Example:

```
> ahm -f /home/agens/etc/ahm_config.yml
```

AHM은 다양한 명령줄 인수를 허용합니다. 허용되는 인수 목록을 ahm -help을 사용하여 확인 할 수 있습니다.

```
> ahm -- help
ahm version 1.0,
HA manager for Agens[SQL|Graph] server

Usage:
ahm [ -f CONFIG_FILE ] [ -d ] [stop|start]
-f, --config-file=CONFIG_FILE
      Set the path to the ahm_config.yml configuration file
      (default: /usr/local/etc/ahm_config.yml)
-h, --help      Print this help
-d, --debug     Debug mode
-n --no-detach Do not detach terminal
```

Note

모든 AHM 노드에 대해 동일한 nodes.conf 파일을 사용해도 됩니다. AHM은 로컬 노드에 대한 항목을 자동으로 감지하고 원격 노드 주소만 사용하여 클러스터를 생성합니다. 11

19.6 AHM 유 틸리티 (Client Interface)

AHM은 DB 관리자를 위한 클러스터 관리 및 모니터링 인터페이스를 제공합니다. 이를 통해 클러스터의 상태 확인, 클러스터 종료, Standby 노드를 attach 또는 promote와 같은 작업을 수행할 수 있습니다.

19.6.1 ahm_cluster_status

본 유 틸리티는 현재 클러스터 상태를 제공합니다. 해당 유 틸리티를 다음 옵션과 함께 사용할 수 있습니다.

```
ahm_cluster_status [option ...]  
-h, --host= 대상 서버 IP 또는 VIP  
-p, --port= PORT
```

클러스터가 Primary 및 Standby 2개의 노드로 구성되어 있는 경우 다음과 같은 결과를 확인할 수 있습니다.

```
> ahm_cluster_status -p 9595 -h 210.104.181.42

Cluster Status
Leader Node : AHM-NODE-1
Leader Host : 210.104.181.77
Successor Node : AHM-NODE-2
Successor Host : 210.104.181.78
VIP : 210.104.181.42
Leader-hold-VIP : True
Remote Nodes : 1

Node Database Address Port Priority Witness State
AHM-NODE-1 PRIMARY 210.104.181.77 9500 1 False LEADER
AHM-NODE-2 STANDBY 210.104.181.78 9500 2 False STANDBY
```

--

다음은 AHM 노드의 State-name 입니다.

PG State Number	STATE-NAME	DEFINITION
0	DEAD	Node is not initialized
1	LOADING	Loading configuration files
2	INITIALIZING	
3	PARTICIPATE_IN_ELECTION	
4	STAND_FOR_LEADER	Node contested as Leader in election
5	LEADER	Node is cluster leader
6	STANDBY	Node is a standby
7	LOST	Node is Lost and not reachable
8	WAITING_FOR_LEADER_FORMATATION	Node is waiting for Leader formation
9	TAKING_OVER	Node announcing a leader contestant
10	IN_RESOURCE_LIMITATION	Node does not have valid Standby to be a leader
11	SHUTDOWN	Node is shut downed

다음은 AHM 노드로 구성된 데이터베이스의 state-name 입니다.

PG State Number	STATE-NAME	DEFINITION
0	UNKNOWN	Unknown state
1	PRIMARY	Primary database
2	STANDBY	Standby database
3	FAILED	Failed database
4	QUARANTINED	Quarantined
5	NOT_CONFIGURED	Database not configured

19.6.2 ahm_attach_node

본 유ти리티는 이전에 연결이 끊긴 ahm 노드를 클러스터에 연결해줍니다.

ahm_attach_node [Option...]

-h, --host= 대상서버 IP

-p, --port= PORT

```
> ahm_attach_node -h 210.104.181.78 -p 9595
```

Command Success

Info: attach database executed successfully

19.6.3 ahm_shutdown_cluster

ahm_shutdown_cluster 유ти리티는 AgensSQL 데이터베이스를 중단하지 않고 모든 AHM 노드를 종료합니다.

ahm_shutdown_cluster

```
> ahm_shutdown_cluster
```

Command Success

Info: Shutdown request sent to remote nodes

19.6.4 ahm_promote_node

본 유ти리티는 AgensSQL 서버를 새로운 Primary AgensSQL 서버로 promote 시킵니다.

ahm_promote_node [Option...]

-h, --host= 대상서버 IP

-p, --port= PORT

```
> ahm_promote_node -h 210.104.181.77 -p 9595
```

Command Success

Info: Promote node task successfully scheduled

19.7 지원하는 Failover 시나리오

AHM은 클러스터로 구성된 노드의 장애를 모니터링하여 failover의 발생을 탐지합니다.

AHM의 failover 시나리오 세트는 매우 구체적이고 제한적입니다.

* **Failover** : Primary-Standby로 구성된 HA구성에서 Primary 노드가 장애가 발생하여 서비스를 못할 경우 HA솔루션에서 감지하여 Standby 서버가 서비스 가능한 Primary로 전환 되는 과정

- Failover 발생 할 수 있는 상황
 - Primary database가 충돌하거나 종료된 경우
 - Primary database를 hosting하는 노드가 충돌하거나 연결할 수 없는 경우

AHM은 이러한 조건의 정확성을 확인하기 위해 모든 시도를 합니다. AHM이 기본 데이터베이스 또는 노드에 장애가 발생했음을 확인할 수 없는 경우 AHM은 클러스터에서 Failover 작업을 수행하지 않습니다.

AHM은 다음과 같은 Failover 시나리오를 지원합니다.

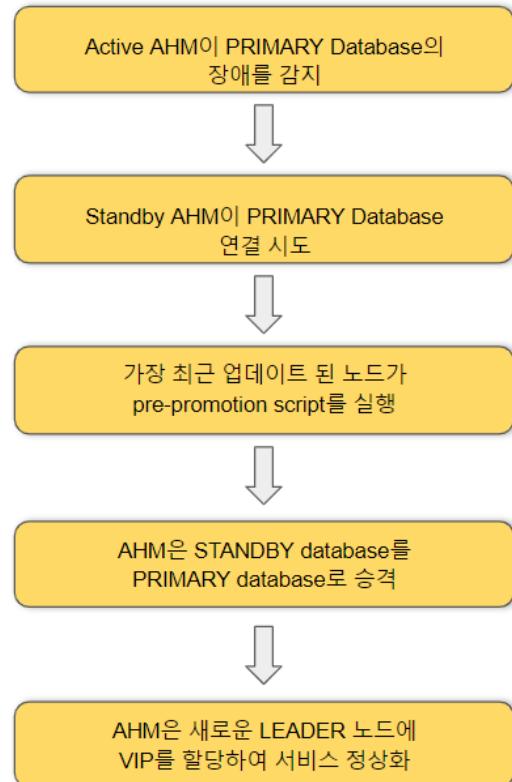
19.7.1 Primary database is down

Primary database 노드에서 실행 중인 AHM이 Primary database에서 발생한 장애를 감지하면 AHM은 장애 확인 프로세스를 시작합니다.

Primary 노드의 AHM이 Primary 데이터베이스에서 실패를 감지하면 모든 AHM이 Primary 데이터베이스에 직접 연결을 시도합니다.

Primary 데이터베이스 서버에 연결할 수 있는 AHM이 없는 경우 AHM은 failover 프로세스를 시작합니다.

최신 노드의 Standby AHM은 pre-promotion 스크립트(해당하는 경우)를 실행하고 Standby 데이터베이스를 Primary 데이터베이스로 promote하고 virtual IP 주소를 할당합니다.



- 장애 노드 복구 방법

- 장애 발생한 노드에서 데이터베이스를 다시 복구 해주세요.
- 장애를 복구한 노드에서 `ahm_attach_node` 명령어를 호출합니다.

- 복구 된 Standby Node를 Primary 로 재전환

장애가 발생하였던 노드를 Standby로 복구 후 Primary로 다시 변환 할 수 있습니다.

복구 된 Node에서 `ahm_promote_node` 명령어를 호출하여 노드를 Primary 역할로 다시 승격합니다.

Note

AHM은 장애가 발생한 Primary 데이터베이스를 재구축하여 Standby 상태로 만들지

않습니다. 재구성하기 전에 Primary 가 실패한 이유를 확인하고 새 Primary 에서 모든 데이터를 사용할 수 있는지 확인하는 것이 중요합니다. 서버를 Standby 상태로 복원할 준비가 되면 이전 데이터 디렉토리를 제거하고 서버를 복원할 수 있습니다. 자세한 내용은 Standby 서버 설정에 대한 AgensSQL 설명서를 참조하십시오.

19.7.2 Standby database is down

Standby AHM이 데이터베이스 장애를 감지하면 다른 AHM에 알리고 노드를 LOST 상태로 표시합니다.

Standby 데이터베이스를 정상 상태로 되돌린 후 `ahm_attach_node` 명령을 호출하여 standby 데이터베이스를 클러스터로 되돌립니다.

19.7.3 Primary AHM exits or node fails

리더 노드가 충돌하거나 노드에 장애가 발생하면 Standby 노드는 장애를 감지하고 적절한 경우 장애 조치를 시작합니다.

Primary AHM이 클러스터에서 종료될 경우, 모든 AHM은 Primary 데이터베이스에 직접 연결하도록 합니다. 데이터베이스에 연결할 수 있으면, AHM은 Primary AHM의 장애를 통지합니다.

최신 노드의 Standby AHM은 fencing 스크립트(해당되는 경우)를 실행하고 Standby 데이터베이스를 Primary 데이터베이스로 `promote`하고 virtual IP 주소를 standby 노드에 할당합니다. 해당되는 경우 AHM은 post-Promotion 스크립트를 실행합니다.



19.7.4 Standby AHM exits or node fails

클러스터에서 Standby AHM이 종료하거나 실패 오류가 발생하면, 다른 AHM은 해당 Standby AHM이 클러스터에 연결되지 않음을 감지합니다.

장애가 감지되면 이외의 AHM은 해당 STANDBY 데이터베이스에 연결을 시도합니다.

AHM이 문제가 있음을 확인하면 AHM은 해당 노드를 손실된 것으로 표시하고 LOST 상태로 변경합니다.



19.8 시나리오 Example

- AHM 구성

구성 정보

Physical	Primary	Standby
IP	210.104.181.77	210.104.181.78
VIP	210.104.181.42	
Port	5432	
AGHOME	/home/agens/AgensSQL-2.13.7.0/	
PGDATA	/home/agens/AgensSQL-2.13.7.0/db_cluster	
Replication User	repluser	

- 1) AHM 설치 진행 (Primary-Standby 각 노드에 모두 구성)

AHM 설치 파일 압축해제

```
$ tar -zxvf AHM_v1.0.tar.gz
```

- 2) 환경 설정 (Primary-Standby 각 노드에 모두 구성)

```
$ vi ~/.bash_profile  
export PATH=$AGHOME/bin:/home/agens/AHM/bin:$PATH  
source ~/.bash_profile
```

agens User sudo 권한 추가 (root User로 진행)

```
$ vi /etc/sudoers  
## Allows people in group wheel to run all commands  
%wheel  ALL=(ALL)      ALL  
agens   ALL=(ALL)      ALL  
  
## Same thing without a password  
agens       ALL=(ALL)      NOPASSWD: ALL각 노드 간 SSH 설정 진행
```

ssh 설정

(root User 진행)
ssh-keygen -t rsa
ssh-copy-id root@210.104.181.77
ssh-copy-id root@210.104.181.78
ssh-copy-id agens@210.104.181.77
ssh-copy-id agens@210.104.181.78

(agens User 진행)
ssh-keygen -t rsa
ssh-copy-id root@210.104.181.77
ssh-copy-id root@210.104.181.78
ssh-copy-id agens@210.104.181.77
ssh-copy-id agens@210.104.181.78

3) AHM Config 설정 (Primary-Standby 각 노드에 모두 설정)

AHM Config

```
$ cp /home/agens/AHM/etc/ahm_config.yml.sample /home/agens/AHM/etc/ahm_config.yml
```

node_name 부분 이외 모두 동일하게 구성

```

$ vi /home/agens/AHM/etc/ahm_config.yml
is_witness: false
priority : 1
data_dir: '/tmp/' #directory to hold temporary ahm data and pid file
#log_severity: DEBUG

ahm_port: 9500
ipc_port: 9595
socket_dir: /tmp
#nodes_filename: " #default CWD/nodes.conf
node_name: AHM-NODE-1
conn_string: host=localhost port=5432 dbname=postgres user=agens

beacon_message_interval_seconds: 8
beacon_message_retry_count: 2
promote_leader_timeout_seconds: 20
wait_for_primary_timeout_seconds: 20

#
# PG Database Health Check Configurations
#
db_connect_timeout: 5
health_check_period: 30
max_consec_failures: 5

#
# Virtual IP configurations
#
virtual_IP: '210.104.181.42'
if_cmd_path: '/bin'
if_up_cmd: '/usr/bin/sudo /sbin/ip addr add ${IP}/24 dev eno1 label eno1:0'
if_down_cmd: '/usr/bin/sudo /sbin/ip addr del ${IP}/24 dev eno1'
arping_path: '/usr/sbin'
arping_cmd: '/usr/bin/sudo /usr/sbin/arping -U ${IP} -w 1 -l eno1'
ping_path : '/bin'

#
# Logging configurations
#
log_filename: ""
log_prefix: '%l PID:%p %t '
log_verbose: false #true value

#
# Database cluster configuration

```

```

#
db_user: 'agens'
db_database: 'postgres'
db_port: 5432
db_bin_dir: '/home/agens/AgensSQL-2.13.7.0/bin'
db_data_dir: '/home/agens/AgensSQL-2.13.7.0/db_cluster'
db_config_dir: '/home/agens/AgensSQL-2.13.7.0/db_cluster'

#
#  Promotion configurations
#
pre_promotion_command: '/home/agens/AHM/etc/failover.sh'
promotion_command: ""
post_promotion_command: ""

#
#  Failover configurations
#
#follow_command:""
standby_startup_delay: 3
use_pg_rewind: false
failover_consensus_with_half_votes: true

#
#  Replication configurations
#
restore_command: ""
archive_cleanup_command: ""
use_recovery_min_apply_delay: false
recovery_min_apply_delay: 0
primary_conninfo_options: ""

pid_file_name : 'ahm.pid'

```

Nodes Config

```
$ cp nodes.conf.sample nodes.conf
```

```
$ vi nodes.conf
210.104.181.77:9500 #Node1 IP:AHM Port
210.104.181.78:9500 #Node2 IP:AHM Port
```

Fail-Over Script

```
$ cp failover.sh.sample failover.sh
```

```
$ vi failover.sh
HOST_IP=`ifconfig eno1 | grep 'inet ' | awk '{ print $2}'`
VIP=210.104.181.42

if [ "$HOST_IP" = 210.104.181.77 ]
then
    ssh -T root@210.104.181.78 '/sbin/ip addr del 210.104.181.42/24 dev eno1'
    ssh -T agens@210.104.181.78 '/home/agens/AgensSQL-2.13.7.0/bin/ag_ctl stop -D
/home/agens/AgensSQL-2.13.7.0/db_cluster'
elif [ "$HOST_IP" = 210.104.181.78 ]
then
    ssh -T root@210.104.181.77 '/sbin/ip addr del 210.104.181.42/24 dev eno1'
    ssh -T agens@210.104.181.77 '/home/agens/AgensSQL-2.13.7.0/bin/ag_ctl stop -D
/home/agens/AgensSQL-2.13.7.0/db_cluster'
else
    echo $HOST_IP
    echo "Not Nodes"
fi
exit 0
```

4) AHM 기동 (각 노드에서 실행)

```
$ nohup ahm -f /home/agens/AHM/etc/ahm_config.yml &
```

5) 상태 확인

\$ Ahm cluster status -h 210.104.181.42

정상적인 Primary-Standby 상태임을 확인 할 수 있습니다.

Cluster Status
Leader Node: AHM-NODE-1
Leader Host: 210.104.181.77
Successor Node: AHM-NODE-2
Successor Host: 210.104.181.78
Virtual IP : 210.104.181.42

Leader-Hold-VIP: True

Remote Nodes: 1

Node	Database	Address	Port	Priority	Witness	State
AHM-NODE-1	PRIMARY	210.104.181.77	9500	1	False	LEADER
AHM-NODE-2	STANDBY	210.104.181.78	9500	2	False	STANDBY

- Primary Database 장애 시 Standby /Failover 가능

- 1) Primary Node DB Down

```
$ ag_ctl stop
```

- 2) FailOver 확인

```
$ ahm_cluster_status -h 210.104.181.42
```

1번 노드의 장애로 "FAILED" 상태로 변경 되며 2번 노드로 Failover 완료 후 "PRIMARY" 상태로 보여짐을 확인 할 수 있습니다.

Cluster Status

Leader Node: AHM-NODE-2

Leader Host: 210.104.181.78

Successor Node: AHM-NODE-2

Successor Host: 210.104.181.78

Virtual IP : 210.104.181.42

Leader-Hold-VIP: False

Remote Nodes: 1

Node Database Address Port Priority Witness State

AHM-NODE-1	FAILED	210.104.181.77	9500	1	False	STANDBY
AHM-NODE-2	PRIMARY	210.104.181.78	9500	2	False	LEADER

- Failover 노드 정상화

- 1) FAILED Node 정상화 (FAILED Node에서 진행)

```
$ ag_ctl start
```

(DB 손상 시 pg_basebackup를 이용하여 DB 복구 진행 필요)

2) AHM Attach 진행

```
$ ahm_attach_node
```

```
Command Success  
Info: attach database executed successfully
```

3) 상태 확인

```
$ ahm_cluster_status -h 210.104.181.42
```

1번 노드의 장애로 “FAILED” 상태에서 정상화되면서 “STANDBY” 상태로 보여짐을 확인 할 수 있습니다.

```
Command Success  
Info: attach database executed successfully
```

```
[agens:/home/agens]#ahm_cluster_status
```

```
Cluster Status
```

```
Leader Node: AHM-NODE-2
```

```
Leader Host: 210.104.181.78
```

```
Successor Node: AHM-NODE-2
```

```
Successor Host: 210.104.181.78
```

```
Virtual IP : 210.104.181.42
```

```
Leader-Hold-VIP: False
```

```
Remote Nodes: 1
```

Node	Database	Address	Port	Priority	Witness	State
AHM-NODE-1	STANDBY	210.104.181.77	9500	1	False	STANDBY
AHM-NODE-2	PRIMARY	210.104.181.78	9500	2	False	LEADER

- Primary AHM 장애 시 Standby /Failover 가능
- Primary Node AHM Down

```
$ ahm stop
```

- FailOver 확인

```
$ ahm_cluster_status -h 210.104.181.42
```

2번 노드의 장애로 State “LOST” 상태로 변경 되며 2번 노드로 Failover 완료 후 State “LEADER” 상태로 보여짐을 확인 할 수 있습니다. (AHM0이 LOST 상태에서는 Database State는 의미 없습니다.

```
Cluster Status
Leader Node: AHM-NODE-1
Leader Host: 210.104.181.77
Successor Node: Not Set
Successor Host: Not Set
Virtual IP : 210.104.181.42
Leader-Hold-VIP: False
Remote Nodes: 1
```

Node	Database	Address	Port	Priority	Witness	State
AHM-NODE-1	PRIMARY	210.104.181.77	9500	1	False	LEADER
AHM-NODE-2	PRIMARY	210.104.181.78	9500	1	False	LOST

- Failover 노드 정상화

4) LOST Node DB 정상화 (LOST Node에서 진행)

pg_basebackup를 이용하여 DB 복구 진행

```
$ rm -rf $PGDATA
$ pg_basebackup -h 210.104.181.77 -D $AGDATA -U repluser -p 5432 -v -P -R
--wal-method=stream
$ ag_ctl start
```

5) AHM 기동 진행 (LOST Node에서 진행)

```
$ nohup ahm -f /home/agens/AHM/etc/ahm_config.yml &
```

6) 상태 확인

```
$ ahm_cluster_status -h 210.104.181.42
```

2번 노드의 장애로 “LOST” 상태에서 정상화되면서 “STANDBY” 상태로 보여짐을 확인 할 수 있습니다.

Cluster Status

Leader Node: AHM-NODE-1

Leader Host: 210.104.181.77

Successor Node: AHM-NODE-2

Successor Host: 210.104.181.78

Virtual IP : 210.104.181.42

Leader-Hold-VIP: False

Remote Nodes: 1

Node	Database	Address	Port	Priority	Witness	State
------	----------	---------	------	----------	---------	-------

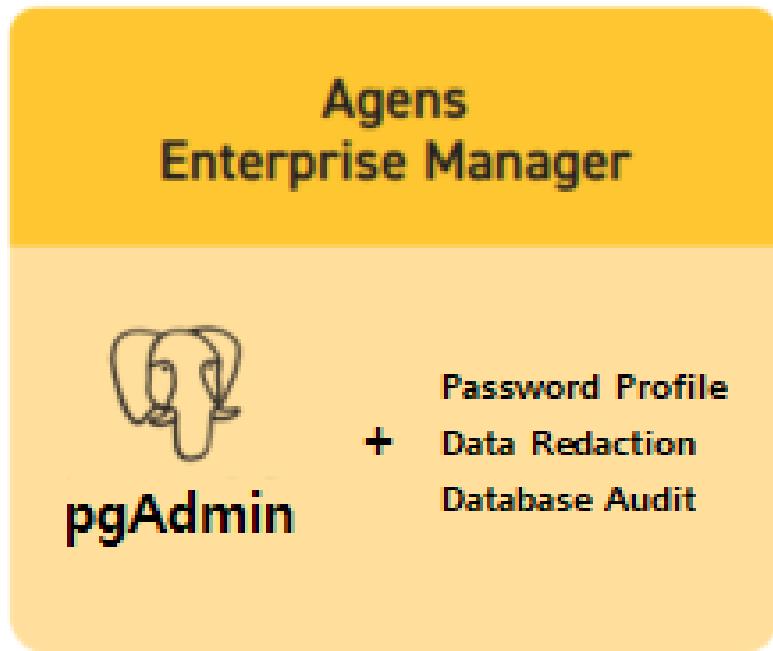
AHM-NODE-1	PRIMARY	210.104.181.77	9500	1	False	LEADER
------------	---------	----------------	------	---	-------	--------

AHM-NODE-2	STANDBY	210.104.181.78	9500	1	False	STANDBY
------------	---------	----------------	------	---	-------	---------

20장. AEM

20.1 Agens Enterprise Manager이란?

Agens Enterprise Manager란 오픈소스 매니지먼트 툴인 PgAdmin에 AgensSQL 확장 모듈을 결합한 매니지먼트 툴입니다. AgensSQL의 운영 및 관리의 학습 곡선을 줄여주는 사용자 친화적인 모니터링 대시보드입니다.



20.1.1 AEM 기능 개요

- 편리한 인터페이스로 AgensSQL 서버 관리
- 서버 CPU 및 메모리 사용량 상태 표시
- 실시간 세션 및 트랜잭션 잠금 모니터링
- 운영 환경 상태 모니터링 및 통계 데이터 수집
- Script, SQL Job, Scheduling 관리 모듈 제공
- 편리한 인터페이스로 Password Profile 구성 가능
- 편리한 인터페이스로 감사 대상 지정 가능
- 편리한 인터페이스로 HA(AHM) 상태 모니터링 가능

20.2 Agens Enterprise Manager 설치

20.2.1 Agens Enterprise Manager 최소 사양

CPU	1 GHz 이상 프로세서
RAM	2GB 이상
DISK	1GB 이상 여유 공간 (AgensEM 설치 파일을 제외한 추가 공간이 필요할 수 있음)

OS	Windows 7 이상
----	--------------

20.2.2 설치 파일 다운로드

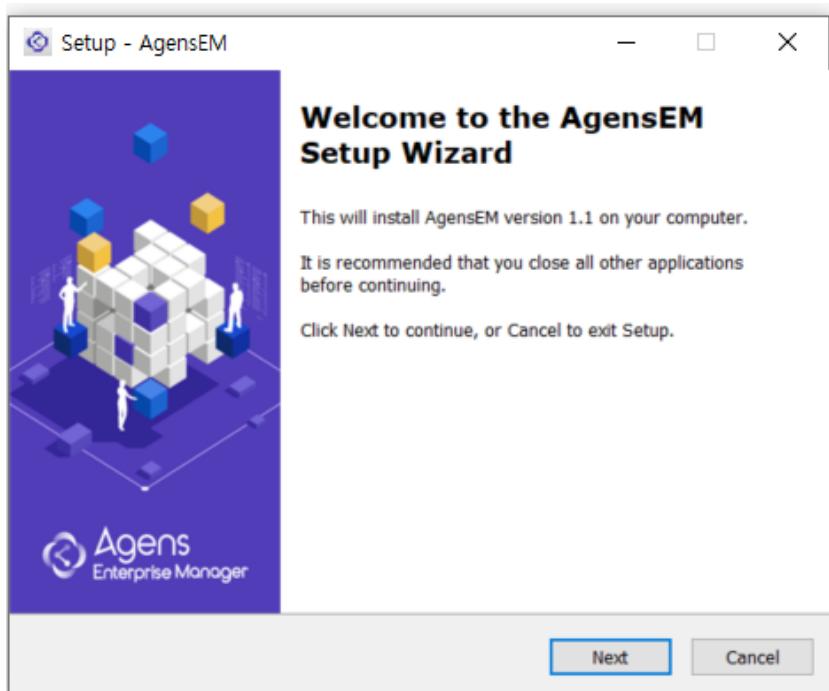
<https://bitnine.net/> 웹사이트 접속 하여 Products 메뉴에서 Agens EnterPrise Manager 다운로드 진행
(추후 지원 예정)

20.2.3 Installer를 이용하여 설치 진행

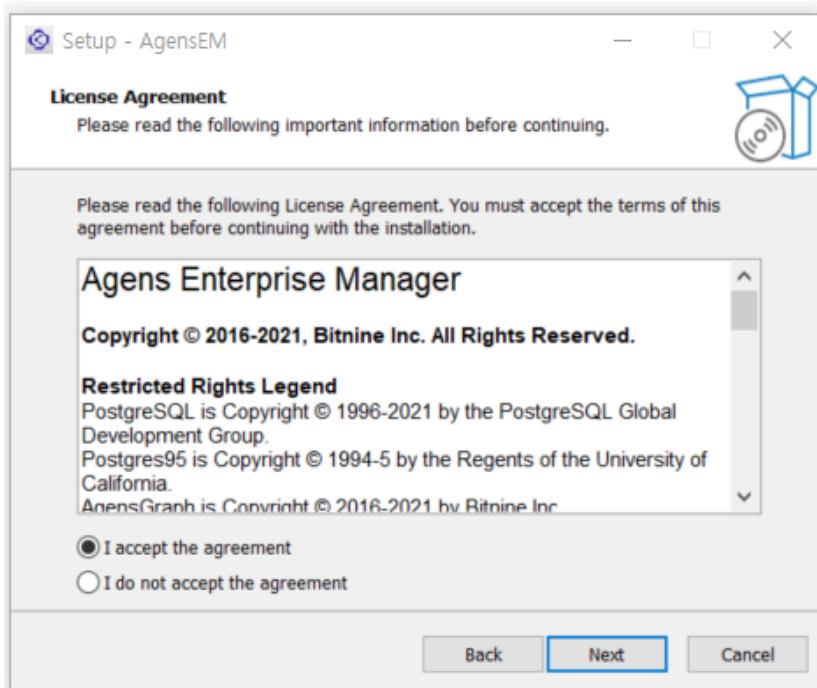
- 1) 인스톨러 실행



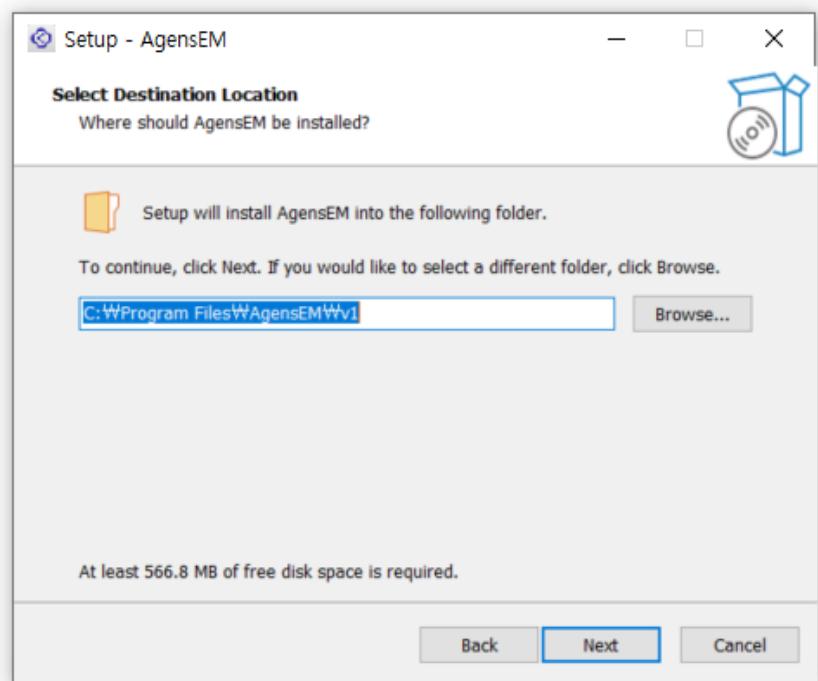
- 2) Next 선택 (Setup Wizard)



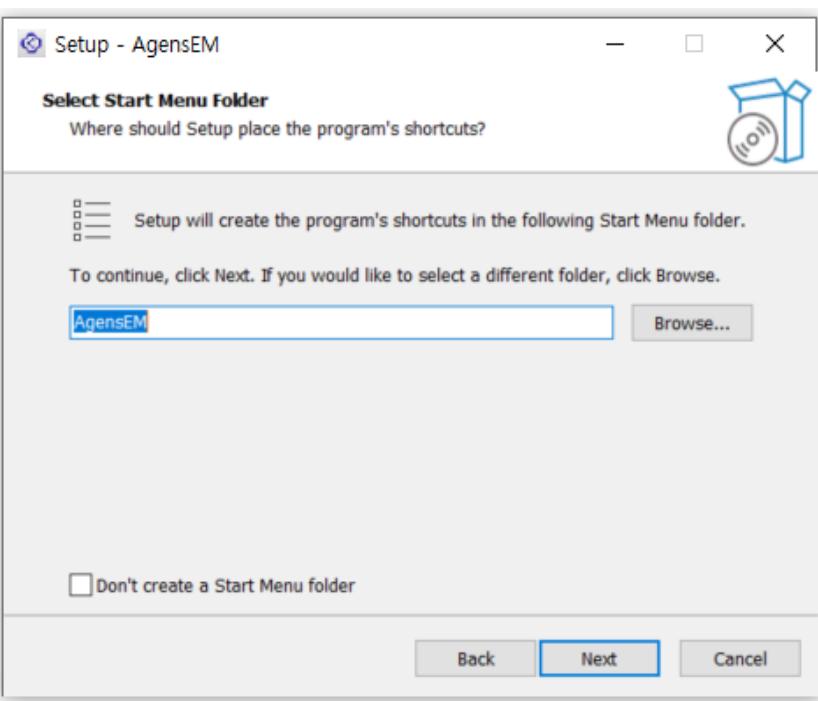
3) Accept 선택후 Next 진행 (License Agreement)



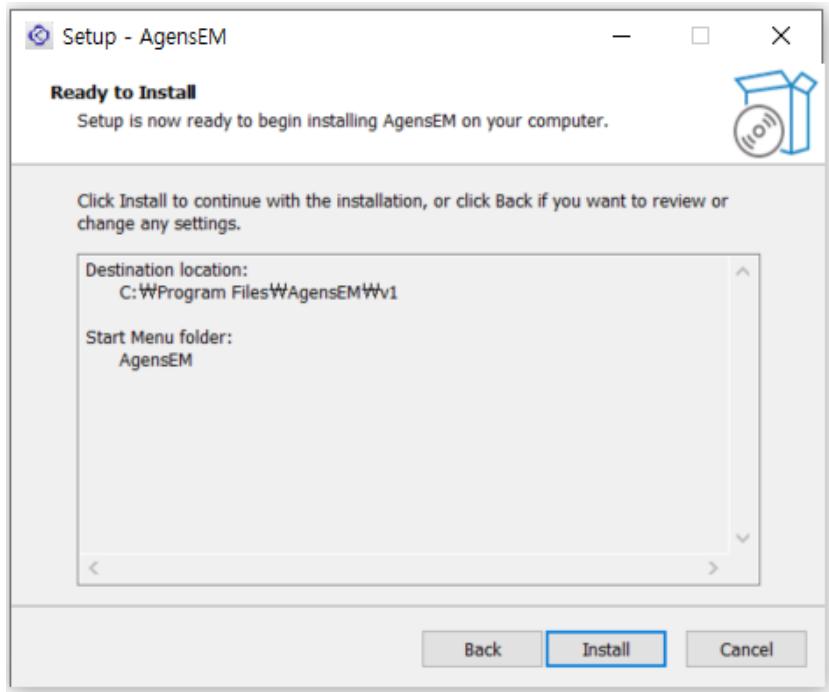
4) 설치 위치 설정 후 Next 진행



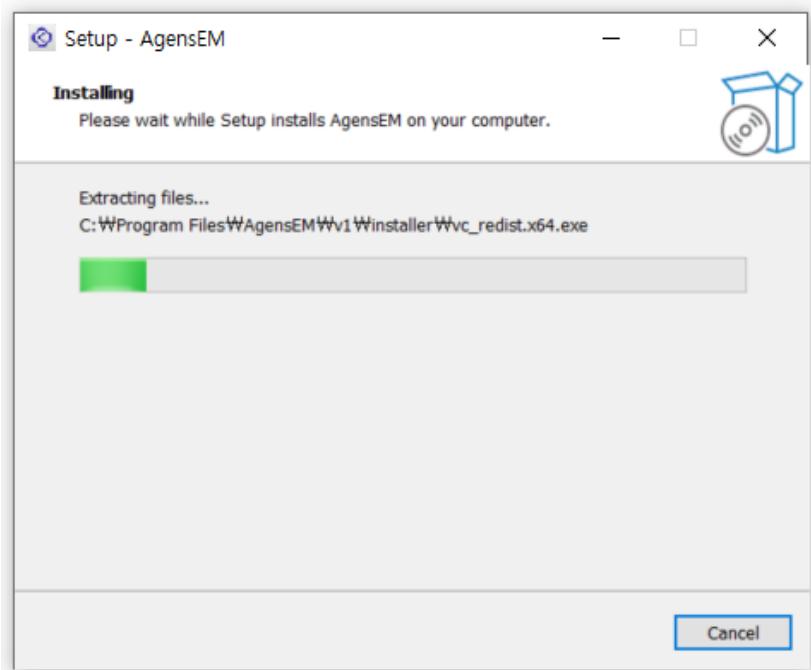
5) 시작메뉴 폴더 선택 후 Next 진행



6) 설치 정보 확인 후 Install 선택하여 설치 진행



7) 설치 중



- 8) Finish 선택하여 설치 완료



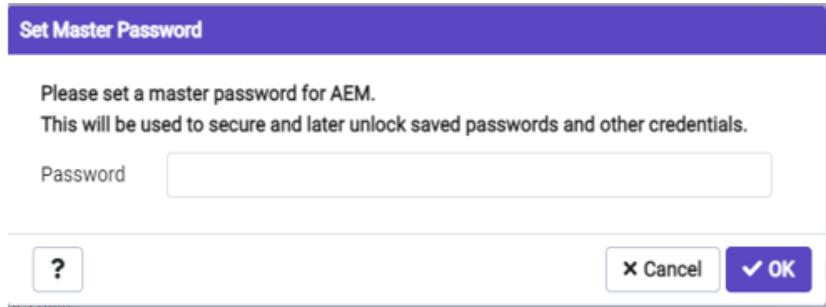
20.3 환경 설정 및 Database 추가

20.3.1 AEM 실행

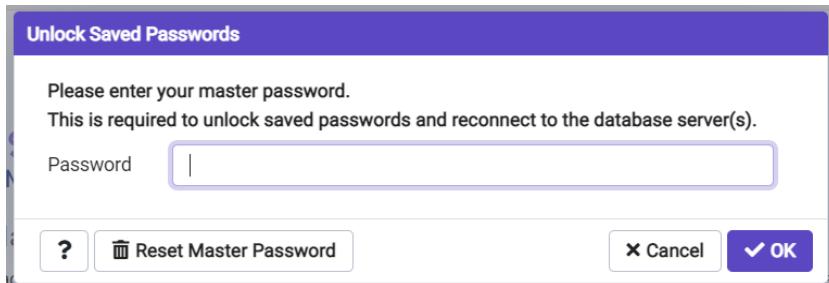
- 1) 시작 메뉴에서 Agens EM v1 실행



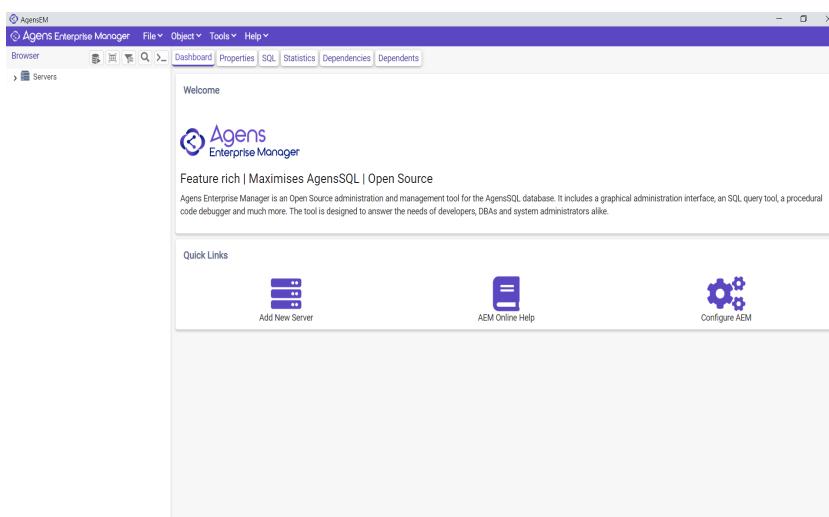
- 2) 최초 설치 후 실행시 비밀번호 설정을 진행 합니다. (AgensEM 마스터 Password 를 설정합니다.)



3) 이후 프로그램 실행 시 최초에 설정한 비밀번호 입력을 요구 합니다.



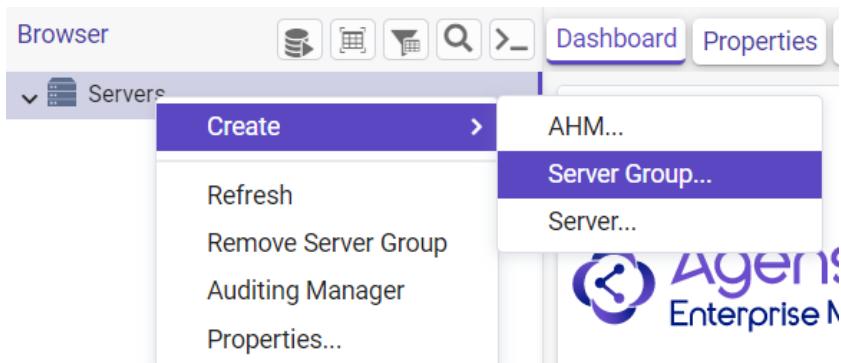
4) AEM 실행 초기 화면



20.3.2 서버 그룹 추가

1) 서버 그룹을 생성하여 Database Server를 그룹별로 관리 가능 합니다.

서버 그룹을 우클릭하여 Server Group을 추가 합니다.

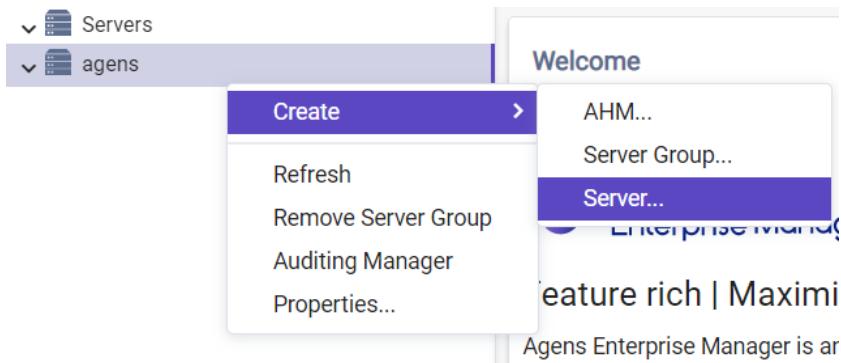


- 2) 서버 그룹 이름을 255자 이내로 지정 합니다.

A screenshot of a dialog box titled 'Create - Server Group'. The dialog has a 'General' tab selected. Inside the tab, there's a 'Name' input field containing the value 'agens'. At the bottom of the dialog, there are three buttons: 'Close' (with a red X icon), 'Reset' (with a circular arrow icon), and 'Save' (with a disk icon).

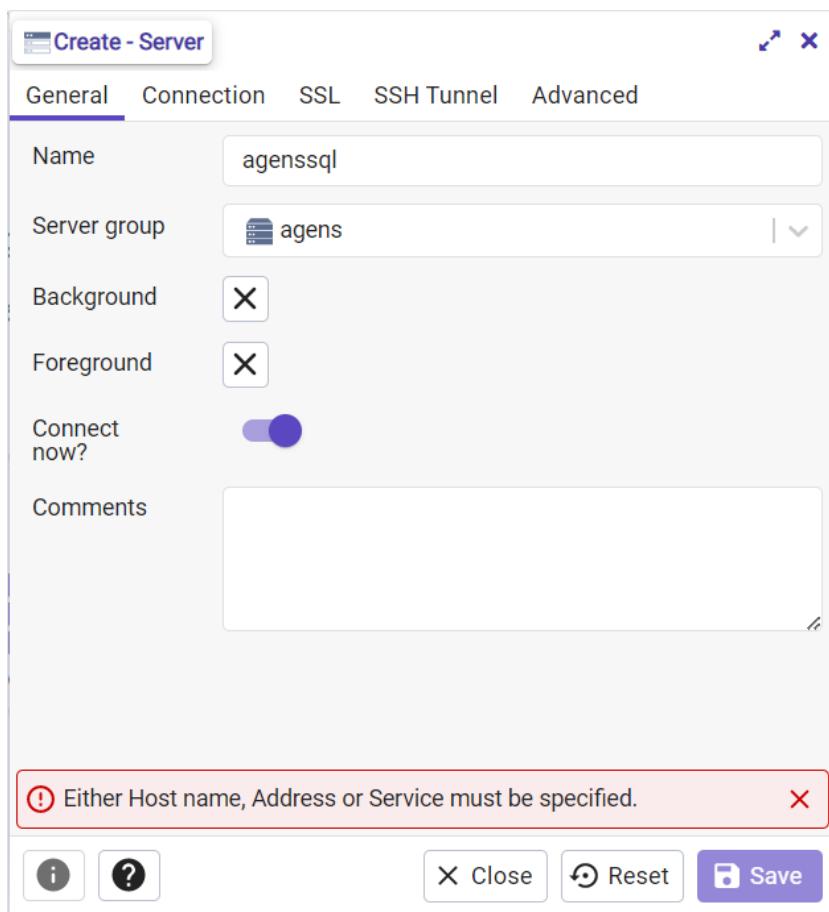
20.3.3 서버 추가

- 1) 서버 그룹을 우클릭하여 Database 서버를 추가 합니다.



- 2) Database 명칭과 상위 그룹을 선택 합니다.

Name - Database 이름을 255자 이내로 지정 합니다.
Server Group - 드롭다운 목록 상자에서 Server Group을 지정 합니다.
Background - 표시 글자 배경 색상을 선택 합니다.
Foreground - 표시 글자 색상을 선택 합니다.
Connect now? - 바로 연결 여부를 선택 할 수 있습니다.
Comment - Database 메모를 저장하며, 최대 1000자 이내로 지정할 수 있습니다.



3) 접속 정보를 입력 후 Save를 선택하여 서버를 추가합니다.

Host name / address - DB 서버의 IP 주소를 입력 합니다. 255자 이내로 입력 가능합니다.
Port - DB 접속 포트를 입력 합니다.

Maintenance Database - 접속 DB 이름을 입력 합니다. 255자 이내로 입력 가능합니다.

Username - 접속 유저명을 입력 합니다.

Kerberos authentication? - Kerberos 인증 사용 여부를 확인합니다.

Password - 접속 유저의 패스워드를 입력합니다.

Save password? - 패스워드를 저장 여부를 확인 합니다.

Role - 서버 인증 후 클라이언트에 전달될 권한이 있는 Role의 이름을 지정합니다.

Service - 서비스 이름을 지정합니다.

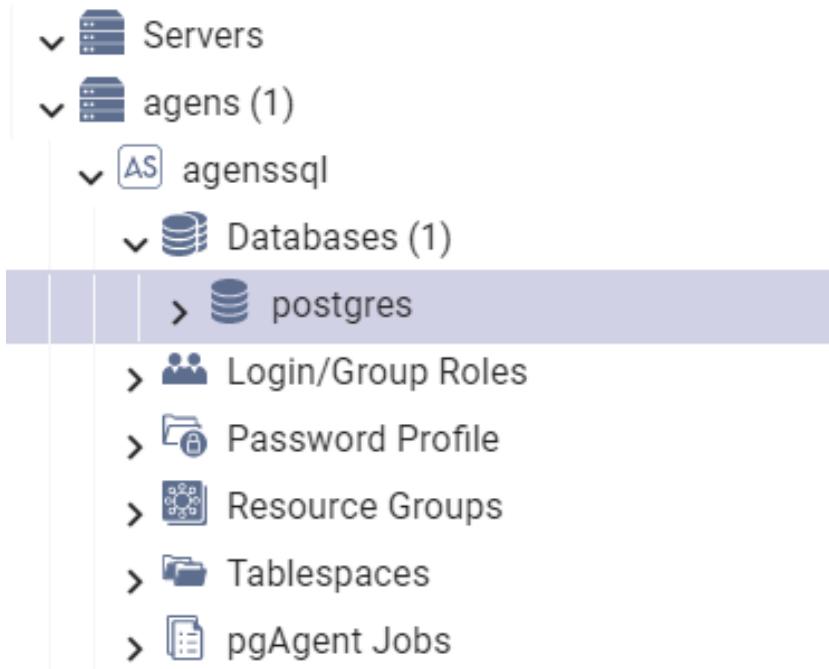
Create - Server

General Connection SSL SSH Tunnel Advanced

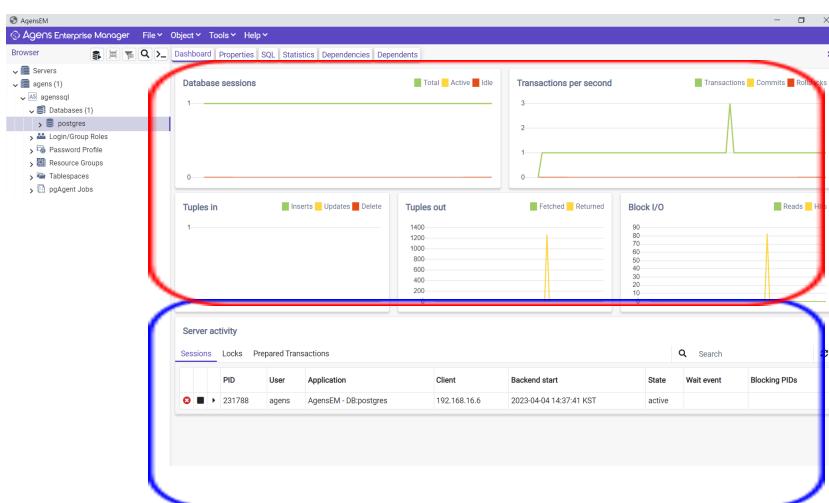
Host name/address	192.168.0.69
Port	5432
Maintenance database	postgres
Username	agens
Kerberos authentication?	<input checked="" type="checkbox"/>
Password
Save password?	<input checked="" type="checkbox"/>
Role	
Service	

Buttons: Close Reset Save

- 4) Browser 트리 메뉴에 database 서버가 추가됨을 확인 할 수 있습니다.



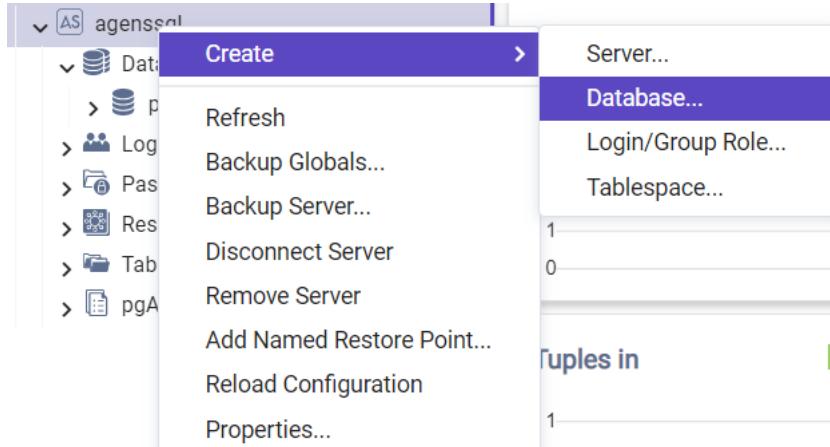
- 5) Database 선택 후 Dashboard 패널 탭에 진입하면, 빨간 네모칸의 대시보드 화면에서 해당 Database의 상태 정보를 한눈에 쉽게 확인 할 수 있습니다.
- 세션개수, 트랜잭션 상황, 데이터 변경 상태, Block I/O
파란 네모칸에서는 현재 접속된 세션정보와 탭을 클릭하여 Lock정보도 확인 할 수 있습니다.



20.4 CREATE

20.4.1 Database 생성

- 1) Server에서 우클릭하여 Create > Database를 선택하여 Database 생성합니다.



- 2) General

Database - Database 이름을 255자 이내로 지정합니다.

Owner - 드롭다운 목록 상자에서 Database 소유자를 선택합니다.

Comment - Database의 메모를 저장하며, 최대 1000자 이내로 지정할 수 있습니다.

A screenshot of the 'Create - Database' dialog box. The 'General' tab is selected. It has five tabs at the top: General, Definition, Security, Parameters, and Advanced. The 'General' tab shows three fields: 'Database' with the value 'agens', 'Owner' with a dropdown menu showing 'agens', and 'Comment' with an empty text area. At the bottom of the dialog are four buttons: 'Close', 'Reset', 'Save' (which is highlighted in blue), and a question mark icon.

3) Definition

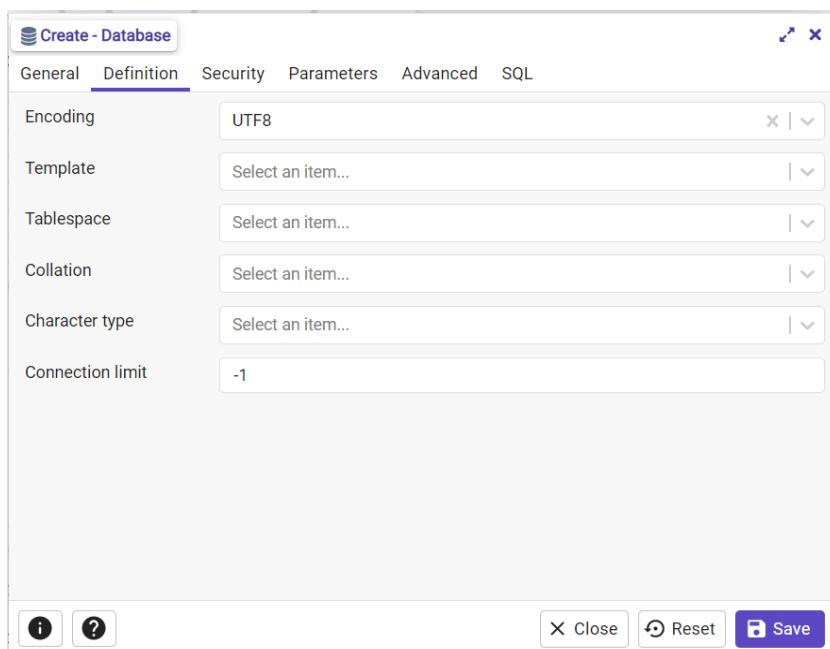
Encoding - 드롭다운 목록 상자에서 데이터베이스 문자 집합을 설정합니다. 기본값은 UTF8입니다.

Template - 드롭다운 목록 상자에서 template Database를 설정합니다. 기본값은 template1입니다.

Tablespace - 드롭다운 목록 상자에서 Database가 저장 될 Tablespace를 설정합니다.

Collation / Character type - 드롭다운 목록 상자에서 데이터베이스 문자 분류를 선택합니다. 이것은 소문자, 대문자 및 숫자와 같은 문자의 분류에 영향을 미칩니다. 설정하지 않을 경우 템플릿 데이터베이스와 동일하게 설정됩니다.

Connection limit - 동시 연결 수 제한을 지정합니다. 기본값(-1)은 무제한 연결을 허용합니다. (Super User는 제한하지 않습니다.)



4) Security

Privileges 추가 아이콘(+)을 클릭하여 해당 Database의 권한을 부여 합니다.

Grantee - 권한을 받을 대상을 지정 합니다.

Privileges - 부여할 권한을 선택 합니다.

Grantor - 해당 권한의 소유자 표시 합니다.

Security label을 사용하는경우 추가 아이콘(+)을 클릭하여 해당 Security labels를 생성합니다.

Provider - Security label provider가 구성된 경우 Provider 이름을 255자 이내로 입력합니다.

Security labels - Security label 이름을 255자 이내로 입력합니다.

Create - Database

General Definition Security Parameters Advanced SQL

Privileges		
Grantee	Privileges	Grantor
<input type="button" value=""/> test	CTc	agens

Security labels	
Provider	Security label

5) Parameters

데이터베이스 사용할 특정 파라미터를 설정 합니다.

Create - Database

General Definition Security Parameters Advanced SQL

Parameters		
Name	Value	Role

6) Advanced

특정 Schema에 대하여 제한 할 수 있습니다.

Create - Database

General Definition Security Parameters Advanced SQL

Schema restriction Select an item... | ▾

Note: Changes to the schema restriction will require the Schemas node in the browser to be refreshed before they will be shown.

Close Reset Save

7) SQL

생성 SQL 문을 출력합니다.

Create - Database

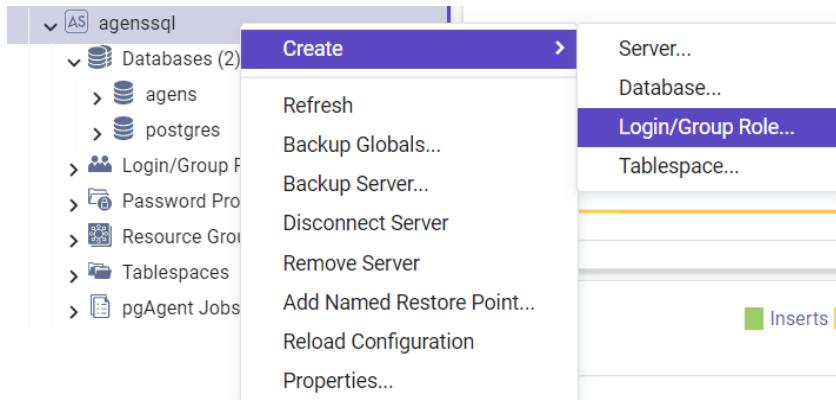
General Definition Security Parameters Advanced SQL

```
1 CREATE DATABASE agens
2   WITH
3     OWNER = agens
4     ENCODING = 'UTF8'
5     CONNECTION LIMIT = -1;
6
7 GRANT ALL ON DATABASE agens TO test;
```

Close Reset Save

20.4.2 Login/Group Role 생성

- 1) 서버에서 우클릭하여 Create > Login/Group Role 를 선택하여 접속 계정 및 Role을 생성합니다.



- 2) General

Name - User / Role 이름을 설정합니다. 255자 이내로 지정 할 수 있습니다.

Comment - User / Role 의 메모를 저장하며, 최대 1000자 이내로 지정할 수 있습니다.

A screenshot of the 'Create - Login/Group Role' dialog box. At the top, it says 'Create - Login/Group Role' with a close button. Below that is a tab bar with 'General' selected, followed by 'Definition', 'Privileges', 'Membership', 'Password Profile', 'Parameters', 'Security', and 'SQL'. The 'General' tab shows a 'Name' field containing 'test' and a 'Comments' field which is empty. At the bottom of the dialog are three buttons: a blue info icon, a blue question mark icon, a 'Close' button with a red 'X', a 'Reset' button with a circular arrow, and a 'Save' button with a disk icon.

3) Definition

Password - User / Role의 비밀번호를 설정합니다.

Account expires - 암호의 만료일자를 설정합니다. 미지정 시 무제한으로 설정됩니다.

Connection limit - 동시 연결 수 제한을 지정합니다. 기본값(-1)은 무제한 연결을 허용합니다. (Super User는 제한하지 않습니다.)

General Definition Privileges Membership Password Profile Parameters Security SQL

Password:

Account expires: No Expiry

Please note that if you leave this field blank, then password will never expire.

Connection limit: -1

3) Privileges

Can login? - Login 권한을 부여합니다. (ON으로 설정하는 경우 User로 생성, OFF로 설정하는 경우 Role로 생성)

Superuser? - Super User 권한을 부여합니다.

Create roles? - Role 생성 권한을 부여합니다.

Create database? - Database 생성 권한을 부여합니다.

Inherit rights from the parent roles? - 하위 상속 권한을 부여합니다.

Can initiate streaming replication and backups? - Replication 및 Backup 권한을 부여합니다.

General Definition Privileges Membership Password Profile Parameters Security SQL

Can login?

Superuser?

Create roles?

Create databases?

Inherit rights from the parent roles?

Can initiate streaming replication and backups?

4) Membership

권한을 공유 합니다.

Member of - 추가아이콘 (+) 을 이용하여 권한을 공유 받을 User / Role을 선택합니다.

Members - 추가아이콘 (+) 을 이용하여 권한을 공유할 User / Role을 선택합니다.

The screenshot shows the 'Membership' tab of the 'Login Role - test' configuration. It displays two sections: 'Member of' and 'Members'. In the 'Member of' section, there is a table with one row for 'pg_monitor' with the 'WITH ADMIN' privilege. In the 'Members' section, there is a table with one row for 'agens' with the 'WITH ADMIN' privilege. At the bottom, there are 'Save' and 'Cancel' buttons.

5) Password Profile

Associations Password Profile - 유저에 적용된 Password Profile을 선택합니다.

The screenshot shows the 'Password Profile' tab of the 'Create - Login/Group Role' configuration. It displays a single row for the 'default' password profile under the 'Associations Password Profile' section. At the bottom, there are 'Save' and 'Cancel' buttons.

6) Parameters

추가아이콘 (+)을 이용하여 User /Role에 적용될 특정 파라미터를 설정 할 수 있습니다.

The screenshot shows the 'Parameters' tab of a configuration interface for a 'Login Role - test'. The tab bar includes General, Definition, Privileges, Membership, Password Profile, Parameters (which is selected), Security, and SQL. Below the tab bar is a table with three columns: Name, Value, and Database. A '+' button is located at the top right of the table area. At the bottom of the interface are several buttons: Close, Reset, and Save.

7) Security

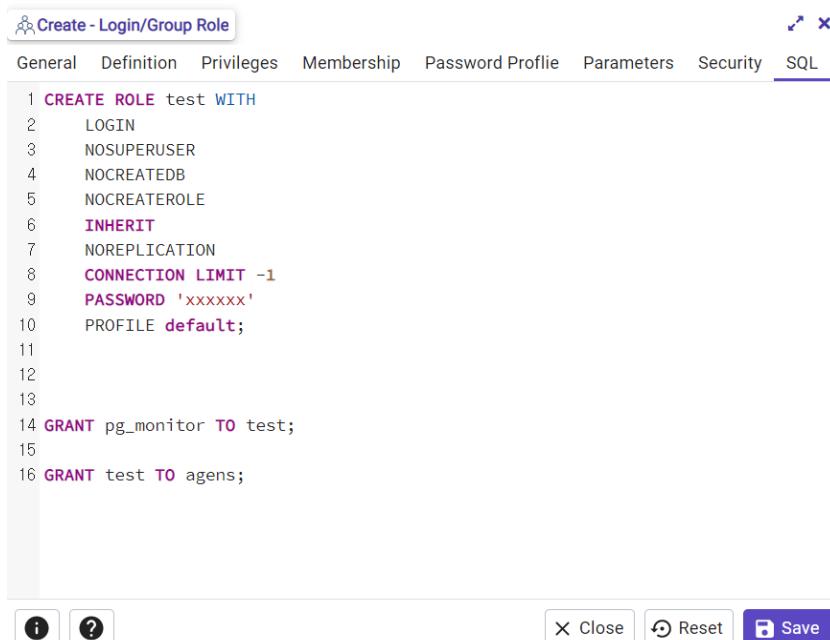
Security label을 사용하는경우 추가 아이콘(+)을 클릭하여 해당 Security labels를 생성합니다.

Provider - Security label provider 구성된 경우 Provider 이름을 255자 이내로 입력합니다.
Security labels - Security label 이름을 255자 이내로 입력합니다.

The screenshot shows the 'Security' tab of the same configuration interface. The tab bar includes General, Definition, Privileges, Membership, Password Profile, Parameters, Security (which is selected), and SQL. Below the tab bar is a table with two columns: Provider and Security label. A '+' button is located at the top right of the table area. At the bottom of the interface are several buttons: Close, Reset, and Save.

8) SQL

SQL 실행문을 생성합니다.



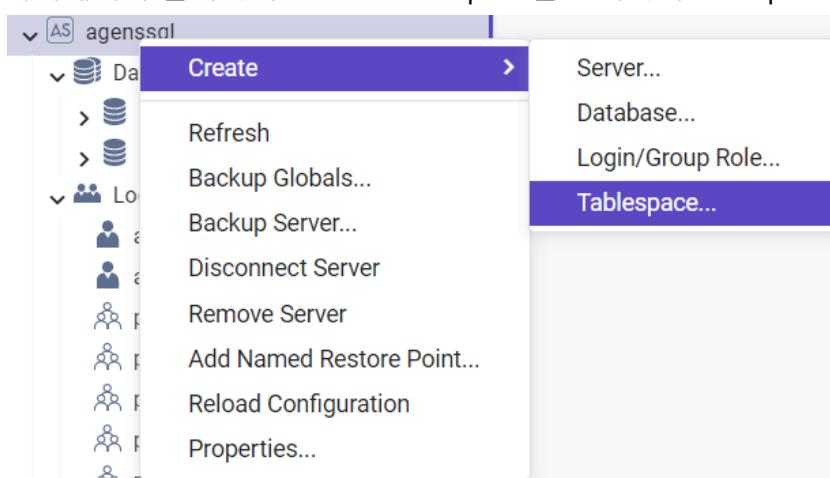
The screenshot shows the 'Create - Login/Group Role' dialog in pgAdmin. The 'SQL' tab is selected. The code area contains the following SQL script:

```
1 CREATE ROLE test WITH
2   LOGIN
3   NOSUPERUSER
4   NOCREATEDB
5   NOCREATEROLE
6   INHERIT
7   NOREPLICATION
8   CONNECTION LIMIT -1
9   PASSWORD 'xxxxxx'
10 PROFILE default;
11
12
13
14 GRANT pg_monitor TO test;
15
16 GRANT test TO agens;
```

At the bottom right of the dialog are 'Close', 'Reset', and 'Save' buttons.

20.4.3 Tablespace 생성

1) 서버에서 우클릭하여 Create > Tablespace 를 선택하여 Tablespace 생성 합니다.



2) General

Name - Tablespace 이름을 255자 이내로 지정합니다.

Owner - 드롭다운 목록 상자에서 Tablespace 소유자를 선택합니다.

Comment - Tablespace 메모를 저장하며, 최대 1000자 이내로 지정할 수 있습니다.

Create - Tablespace

General Definition Parameters Security SQL

Name	test_data
Owner	agens
Comment	

Buttons: Close, Reset, Save

3) Definition

Location - Tablespace가 생성될 경로를 지정합니다.

Create - Tablespace

General **Definition** Parameters Security SQL

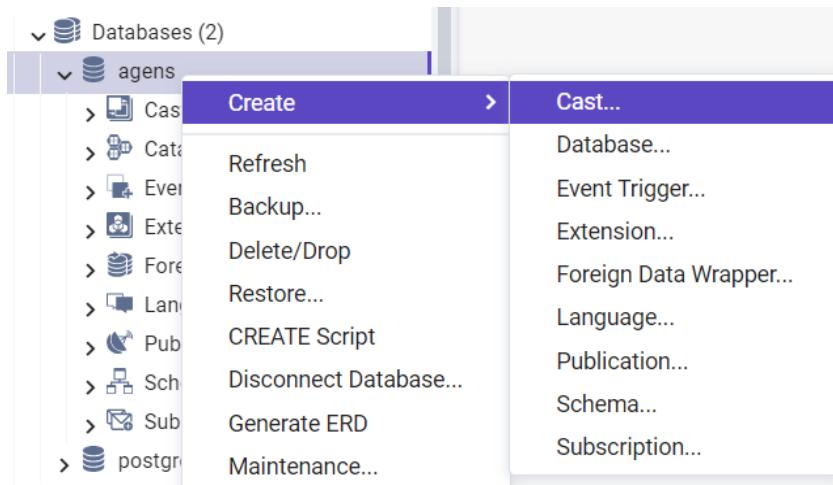
Location	/home/agens/data
----------	------------------

Buttons: Close, Reset, Save

20.4.4 Cast 생성

새로운 데이터 타입의 변환 함수를 정의하는데 사용합니다.

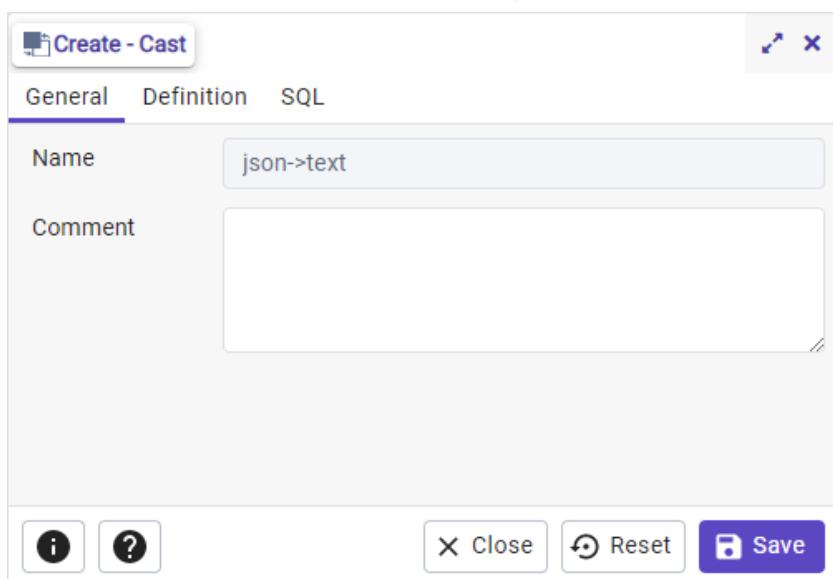
- 1) Database에서 우클릭하여 Create > Cast를 이용하여 Cast를 생성합니다.



- 2) General

Name - Definition 설정 시 자동으로 생성 됩니다.

Comment - 해당 Cast의 메모를 저장하며, 최대 1000자 이내로 지정할 수 있습니다.



3) Definition

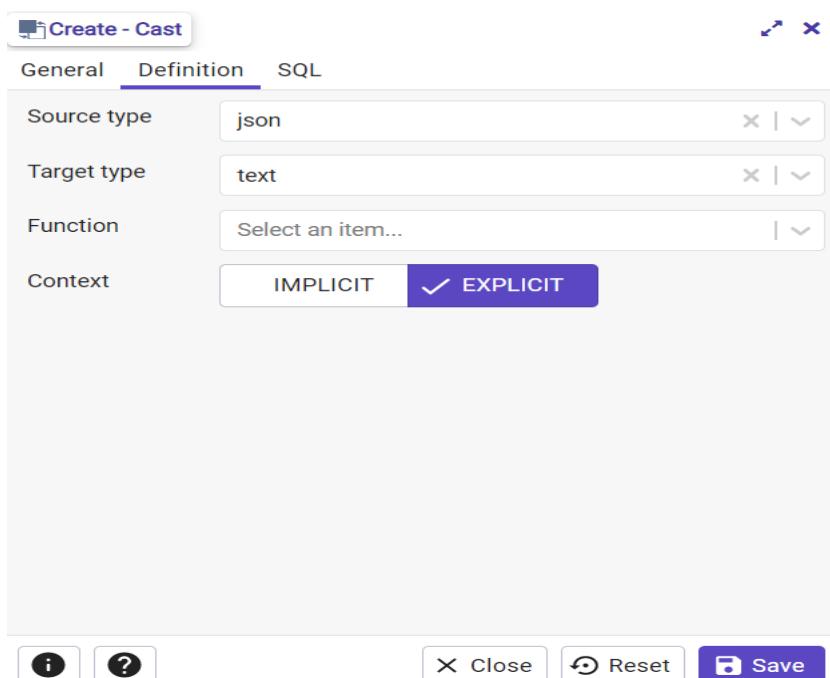
Source type - 드롭다운 목록 상자에서 Source Data Type을 지정합니다.

Target type - 드롭다운 목록 상자에서 Target Data Type을 지정합니다. (변환가능 타입이어야 합니다)

Function - 해당 변환 함수가 존재하는 경우 선택합니다.

Context - 해당 타입 적용 방법을 설정합니다.

암시적 사용(IMP�LICIT), 명시적 사용 (EXPLICIT)



4) SQL

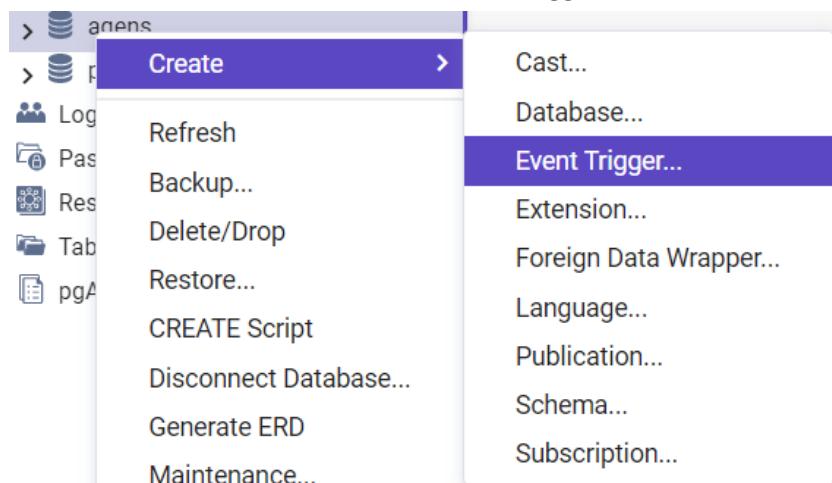
SQL 명령문을 생성합니다.



20.4.5 EventTrigger 생성

이벤트 트리거를 생성합니다. 단일 테이블에 연결되고 DML 이벤트만 캡처하는 일반 트리거와 달리 이벤트 트리거는 특정 데이터베이스에 대해 전역적이며 DDL 이벤트를 캡처할 수 있습니다.

- 1) Database에서 우클릭하여 Create > Event Trigger를 이용하여 Event Trigger를 생성합니다



- 2) General

Name - Event Trigger 이름을 255자 이내로 지정합니다.

Owner - 드롭다운 목록 상자에서 Event Trigger의 소유자를 선택합니다.

Comment - 해당 Event Trigger 메모를 저장하며, 최대 1000자 이내로 지정할 수 있습니다.

A screenshot of the 'Create - Event Trigger' dialog box. At the top, it says 'Create - Event Trigger'. Below that, there are tabs for 'General', 'Definition', 'Security', and 'SQL', with 'General' being the active tab. In the 'General' tab, there are three fields: 'Name' with the value 'test_trigger', 'Owner' with the value 'agens', and 'Comment' which is empty. At the bottom of the dialog box, there are buttons for 'Close', 'Reset', and 'Save', with 'Save' being highlighted in purple.

3) Definition

Trigger enabled? - Trigger 상태를 선택합니다. (Enable, Disable, Replica, Always)

Trigger function - Trigger 동작 Function명을 지정합니다.(사전에 생성 필요)

Event - Event Trigger 실행 시기를 지정합니다. (DDL COMMAND START, DDL COMMAND END, SQL DROP)

When TAG in - 트리거가 실행될 특정 SQL 지정합니다. (작은 따옴표(')과 쉼표(,)로 구분하여 작성합니다.)

Create - Event Trigger

General **Definition** Security SQL

Trigger enabled?	Enable
Trigger function	public.my_event_trigger_function
Event	DDL COMMAND START
When TAG in	1

Buttons: Close Reset Save

4) Security

Security label을 사용하는 경우 추가 아이콘(+)을 클릭하여 해당 Security labels를 생성합니다.

Provider - Security label provider가 구성된 경우 Provider 이름을 255자 이내로 입력합니다.

Security label - Security label 이름을 255자 이내로 입력합니다.

Create - Event Trigger

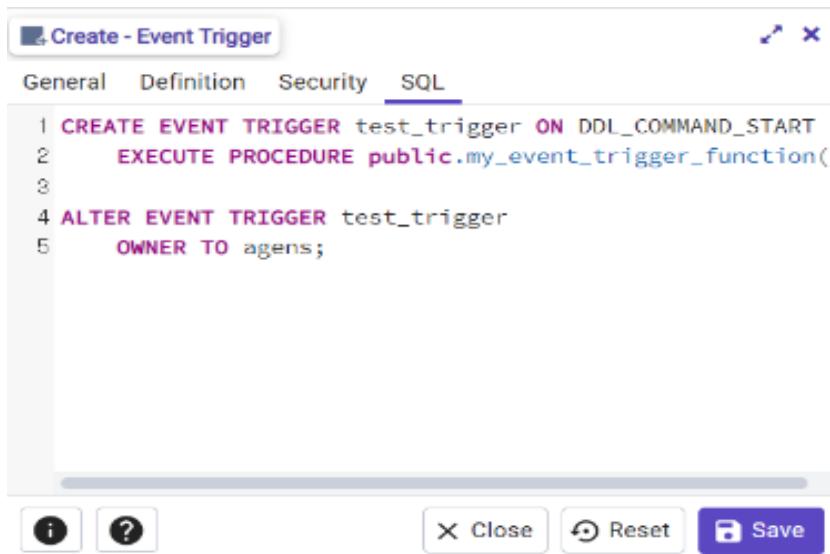
General Definition **Security** SQL

Security labels	
Provider	Security label
+ (Add New Row)	

Buttons: Close Reset Save

5) SQL

SQL 명령문을 생성합니다.



The screenshot shows a software interface titled 'Create - Event Trigger'. At the top, there are tabs for General, Definition, Security, and SQL. The SQL tab is selected, displaying the following SQL code:

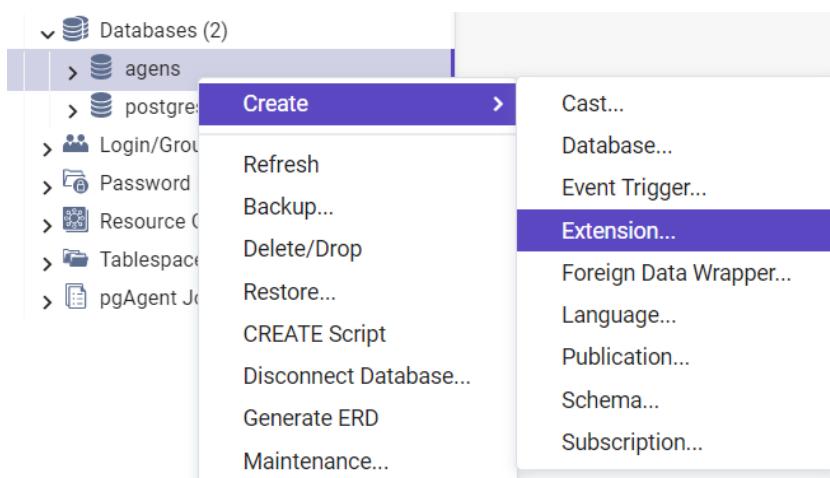
```
1 CREATE EVENT TRIGGER test_trigger ON DDL_COMMAND_START
2     EXECUTE PROCEDURE public.my_event_trigger_function()
3
4 ALTER EVENT TRIGGER test_trigger
5     OWNER TO agens;
```

At the bottom of the dialog, there are buttons for Close, Reset, and Save.

20.4.6 Extension 생성

Extension은 기능을 추가하는 SQL 오브젝트 모음입니다. Extension을 사용하는 각 데이터베이스에 Extension을 설치 해야 합니다. Extension을 데이터베이스에 로드하기 전에 필수 파일이 설치되어 있어야 합니다.

1) Database에서 우클릭하여 Create > Extension를 이용하여 Extension을 생성합니다



2) General

Name - 드롭다운 목록 상자에서 생성할 Extension을 지정합니다.

Comment - Extension 생성 시 해당 Extension의 설명이 자동으로 입력됩니다.

Create - Extension

General Definition SQL

Name: postgres_fdw

Comment:

Buttons: Close, Reset, Save

3) Definition

Schema - Extension의 소유자를 지정 합니다.

Version - 설치 할 Extension의 버전을 지정 합니다.

Create - Extension

General Definition SQL

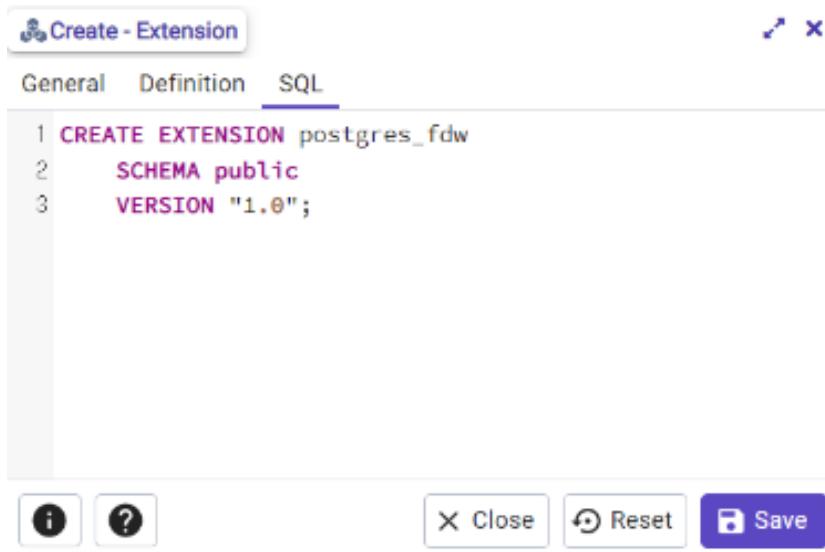
Schema: public

Version: 1.0

Buttons: Close, Reset, Save

4) SQL

SQL 명령문을 생성합니다.



```
CREATE EXTENSION postgres_fdw
SCHEMA public
VERSION "1.0";
```

The screenshot shows the 'Create - Extension' dialog in pgAdmin. The 'SQL' tab is selected, displaying the following SQL code:

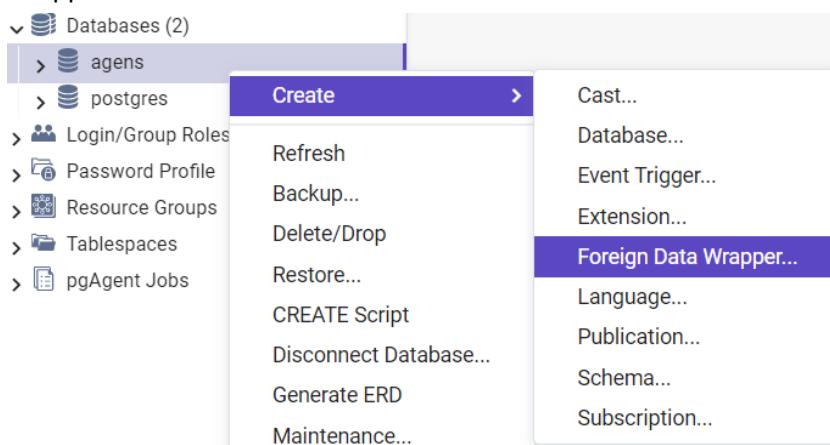
```
1 CREATE EXTENSION postgres_fdw
2     SCHEMA public
3     VERSION "1.0";
```

Below the code, there are buttons for 'Close', 'Reset', and 'Save'.

20.4.7 Foreign Data Wrapper 생성

Foreign Data Wrapper는 AgensSQL과 다른 데이터 소스에 저장된 데이터 간의 연결을 지원합니다. 생성하기 위해선 FDW 함수가 존재해야 합니다. 슈퍼유저를 이용하여 생성이 가능합니다.

- 1) Database에서 우클릭하여 Create > Foreign Data Wrapper를 이용하여 Foreign Data Wrapper를 생성합니다.



2) General

Name - FDW 이름을 255자 이내로 지정합니다.

Owner - 드롭다운 목록 상자에서 FDW 소유자를 지정합니다.

Comment - 해당 FDW 메모를 저장하며, 최대 1000자 이내로 지정할 수 있습니다.

Create - Foreign Data Wrapper

General Definition Options Security SQL

Name	pg_fdw
Owner	agens
Comment	(Empty)

i ? Close Reset Save

3) Definition

Handler - Handler 함수를 지정합니다. 이것은 외부 테이블에 대한 실행 함수를 검색하기 위해 호출될 기준 함수의 이름입니다.

Validator - Validator 함수를 지정합니다. 외부 서버, 사용자 매핑 및 외부 테이블에 대한 옵션을 확인하기 위해 호출되는 함수의 이름입니다.

Create - Foreign Data Wra

General Definition Options Security SQL

Handler	public.postgres_fdw_handler
Validator	public.postgres_fdw_validator

i ? Close Reset Save

4) Options

해당 FDW에서 사용할 Option과 Value를 추가합니다.

The screenshot shows the 'Create - Foreign Data Wrapper' dialog box with the 'Options' tab selected. At the top, there are tabs for General, Definition, Options (which is underlined), Security, and SQL. Below the tabs is a table titled 'Options' with two columns: 'Option' and 'Value'. A large empty area follows, and at the bottom are buttons for Close, Reset, and Save.

5) Security

Privileges 추가아이콘 (+)을 클릭하여 해당 FDW의 권한을 부여합니다.

Grantee - 권한을 받을 대상을 지정합니다.

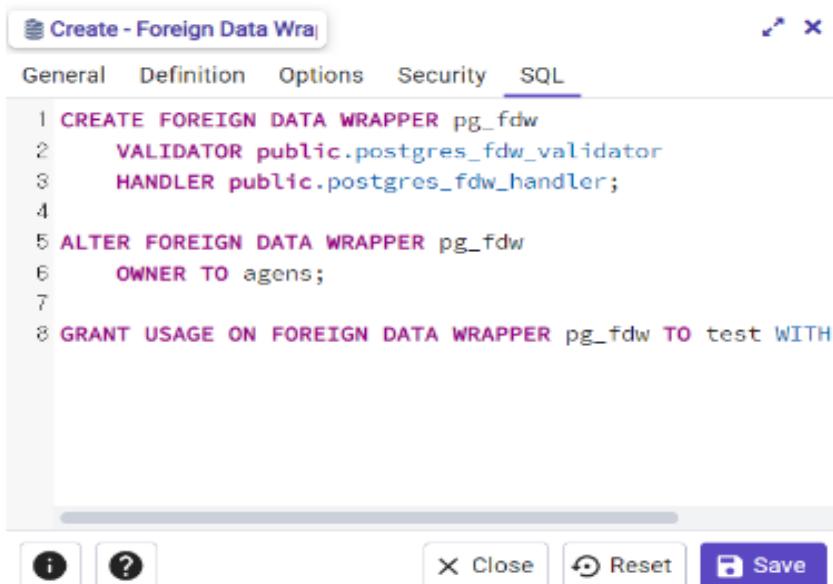
Privileges - 부여할 권한을 선택합니다.

Grantor - 해당 권한의 소유자 표시합니다.

The screenshot shows the 'Create - Foreign Data Wrapper' dialog box with the 'Security' tab selected. At the top, there are tabs for General, Definition, Options, Security (which is underlined), and SQL. Below the tabs is a table titled 'Privileges' with three columns: 'Grantee', 'Privileges', and 'Grantor'. The 'Grantee' column contains a user named 'test'. The 'Privileges' column contains 'U*'. The 'Grantor' column contains a user named 'agens'. A large empty area follows, and at the bottom are buttons for Close, Reset, and Save.

6) SQL

SQL 명령문을 생성합니다.



The screenshot shows a software interface for creating a Foreign Data Wrapper. The title bar says 'Create - Foreign Data Wrapper'. Below it is a tab bar with 'General', 'Definition', 'Options', 'Security', and 'SQL', where 'SQL' is underlined. The main area contains the following SQL code:

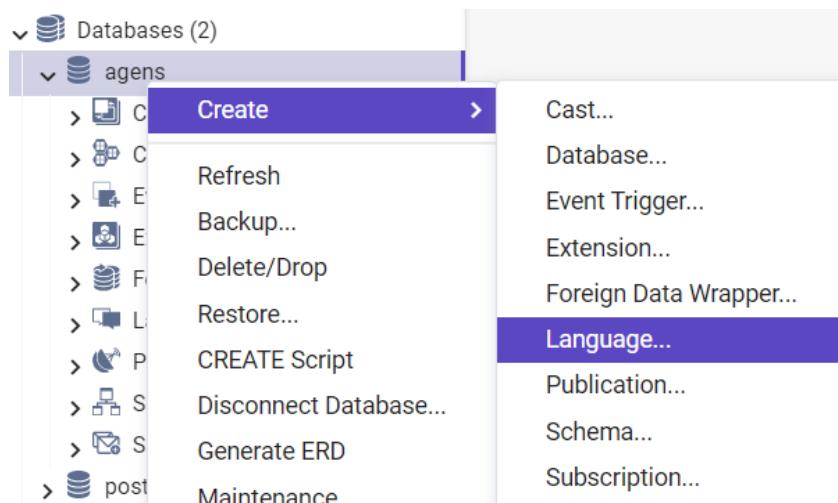
```
1 CREATE FOREIGN DATA WRAPPER pg_fdw
2   VALIDATOR public.postgres_fdw_validator;
3   HANDLER public.postgres_fdw_handler;
4
5 ALTER FOREIGN DATA WRAPPER pg_fdw
6   OWNER TO agens;
7
8 GRANT USAGE ON FOREIGN DATA WRAPPER pg_fdw TO test WITH
```

At the bottom are buttons for 'Close', 'Reset', and 'Save'.

20.4.8 Language 생성

절차적 언어를 구성합니다. 구성하기 위하여 해당 언어를 지원하는 함수와 실행함수 유효성 검사 함수가 사전에 구성되어야 합니다.

- 1) Database에서 우클릭하여 Create > Language를 이용하여 Language를 생성합니다.



2) General

Name - Language의 이름을 255자 이내로 지정 합니다.

Owner - 드롭다운 목록 상자에서 Language의 소유자를 지정 합니다.

Comment - Language의 메모를 저장하며, 최대 1000자 이내로 지정할 수 있습니다.

Create - Language

General Definition Security SQL

Name	test_lang
Owner	agens
Comment	

Buttons: Close Reset Save

3) Definition

Trusted? - 신뢰도를 설정 합니다. (On : 모두 사용가능 OFF : 슈퍼유저만 사용가능)

Handler function - 해당언어를 지원하는 함수를 지정합니다.

Inline function - 코드 블록을 실행할 함수를 지정합니다.

Validator function - 유효성 검사를 담당할 함수를 지정합니다.

Create - Language

General **Definition** Security SQL

Trusted?	<input checked="" type="checkbox"/>
Handler function	plpgsql_call_handler
Inline function	plpgsql_inline_handler
Validator function	plpgsql_validator

Buttons: Close Reset Save

4) Security

Privileges 추가아이콘 (+) 을 클릭하여 해당 Language의 권한을 부여합니다.

Grantee - 권한을 받을 대상을 지정합니다.

Privileges - 부여할 권한을 선택합니다.

Grantor - 해당 권한의 소유자를 표시합니다.

Security label을 사용하는경우 추가 아이콘(+) 을 클릭하여 해당 Security labels를 생성합니다.

Provider - Security label provider 구성된 경우 Provider 이름을 255자 이내로 입력합니다.

Security labels - Security Label 이름을 255자 이내로 입력합니다.

Privileges		
Grantee	Privileges	Grantor
test	U*	agens

Security labels	
Provider	Security label
agens	

Close Reset Save

4) SQL

SQL 명령문을 생성 합니다.

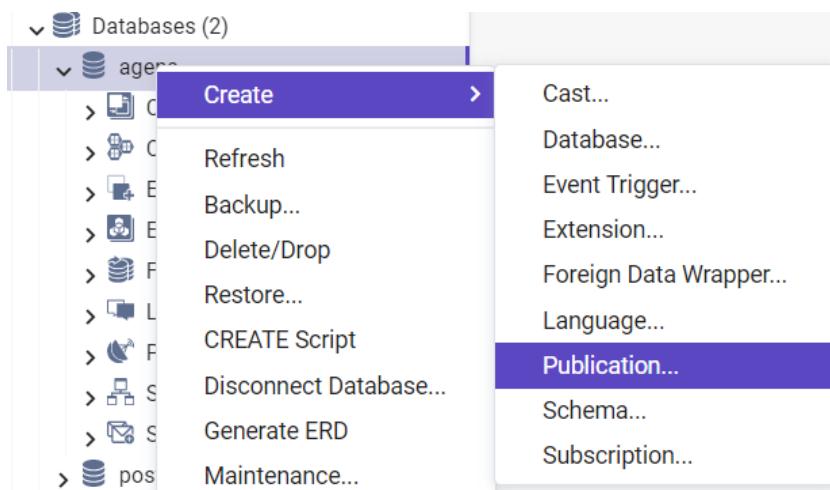
```
1 CREATE TRUSTED PROCEDURAL LANGUAGE test_lang
2   HANDLER plpgsql_call_handler
3   INLINE plpgsql_inline_handler
4   VALIDATOR plpgsql_validator
5 ;
6 ALTER LANGUAGE test_lang
7   OWNER TO agens;
8 GRANT USAGE ON LANGUAGE test_lang TO test WITH GRANT OPT
```

Close Reset Save

20.4.9 Publication 생성

논리적 복제에 사용할 대상 Publication을 생성 합니다.

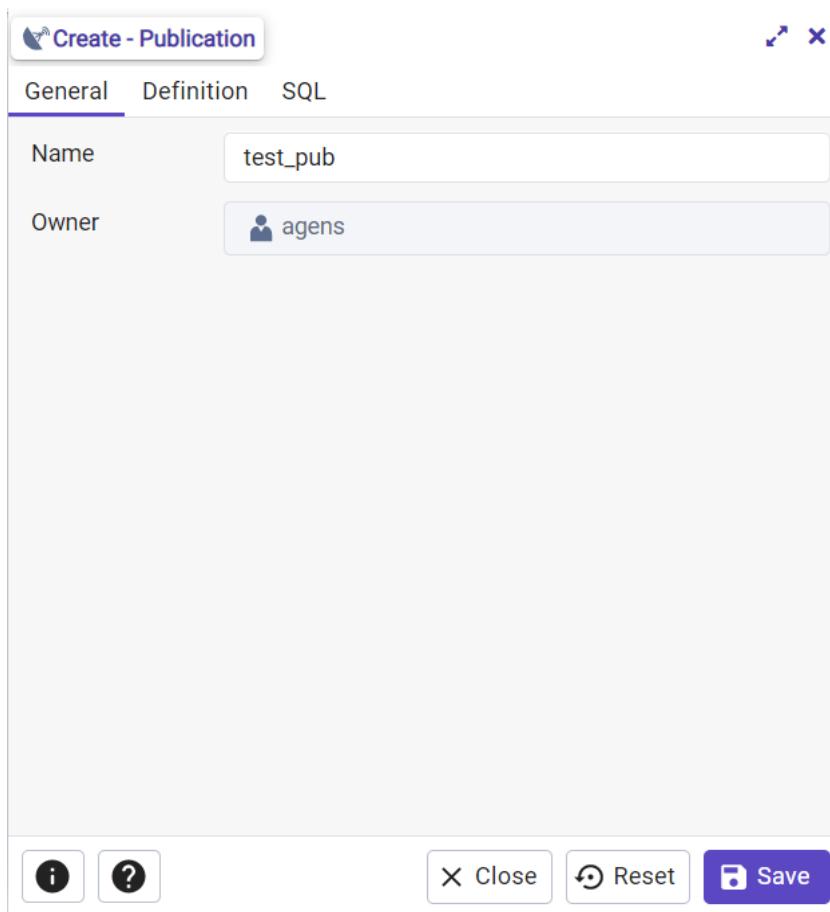
- 1) Database에서 우클릭하여 Create > Publication을 이용하여 Publication을 생성합니다



- 2) General

Name - Publication 이름 255자 이내로 지정 합니다.

Owner - Publication 소유자를 표시합니다. (접속된 사용자로 자동 지정)



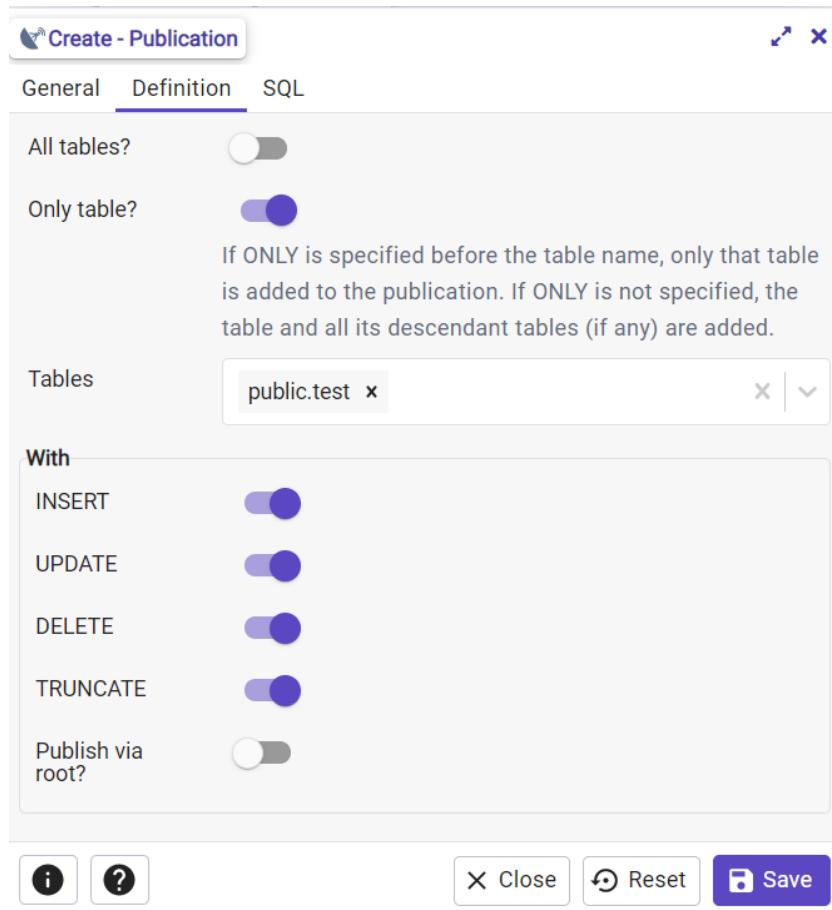
3) Definition

All tables? - 이후에 생성되는 모든 테이블의 복제를 허용합니다.

Only table? - 지정된 테이블만 복제를 허용합니다.(Tables 지정 후 ON 설정)

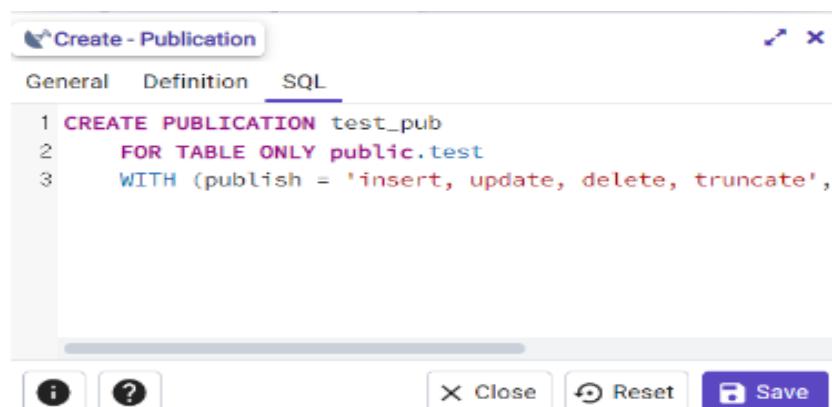
Tables - 테이블 지정 복제 방식, 테이블을 지정합니다.

With - 복제할 작업에 대하여 설정합니다.



4) SQL

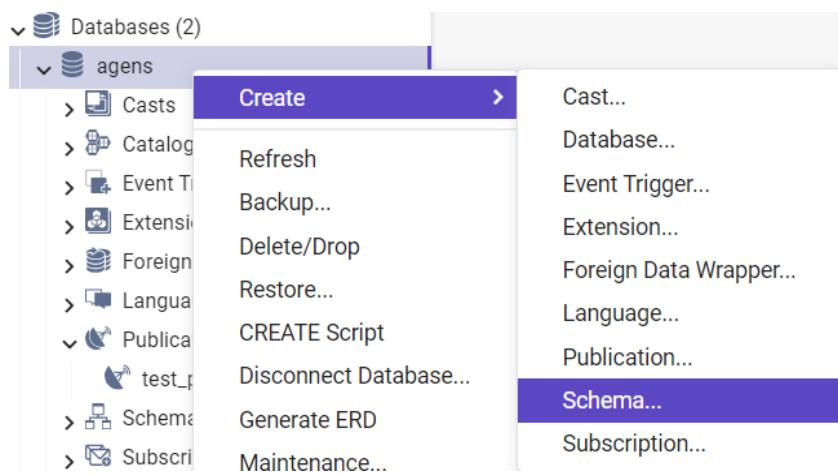
SQL 명령문을 생성합니다.



20.4.10 Schema 생성

데이터베이스 객체를 그룹화 할 수 있는 Schema를 추가 생성합니다.

- 1) Database에서 우클릭하여 CREATE > Schema 를 이용하여 Schema를 생성합니다



- 2) General

Name - Schema 이름을 255자 이내로 지정합니다.

Owner - 드롭다운 목록 상자에서 Schema 소유자를 지정합니다.

Comment - 해당 Schema의 메모를 저장하며, 최대 1000자 이내로 지정할 수 있습니다.

The screenshot shows the 'Create - Schema' dialog box. The 'General' tab is selected. The 'Name' field is set to 'test_schema'. The 'Owner' dropdown is set to 'agens'. The 'Comment' field is empty. At the bottom, there are buttons for 'Close', 'Reset', and 'Save'.

4) Security

Privileges 추가아이콘 (+) 을 클릭하여 해당 Schema의 권한을 부여합니다.

Grantee - 권한을 받을 대상을 지정합니다.

Privileges - 부여할 권한을 선택합니다.

Grantor - 해당 권한의 소유자를 표시합니다.

Security label을 사용하는경우 추가 아이콘(+) 을 클릭하여 해당 Security labels를 생성합니다.

Provider - Security label provider가 구성된 경우 Provider 이름을 255자 이내로 입력합니다.

Security labels - Security label 이름을 255자 이내로 입력합니다.

Create - Schema

General Security **Default privileges** SQL

Privileges

Grantee	Privileges	Grantor
<input type="button" value="trash"/> test	C*U*	agens

Security labels

Provider	Security label

3) Default privileges

해당 오브젝트(Tables, Sequences, Functions, Types)의 추가아이콘 (+) 을 이용하여 Schema 내부 객체들의 권한을 부여합니다.

Create - Schema

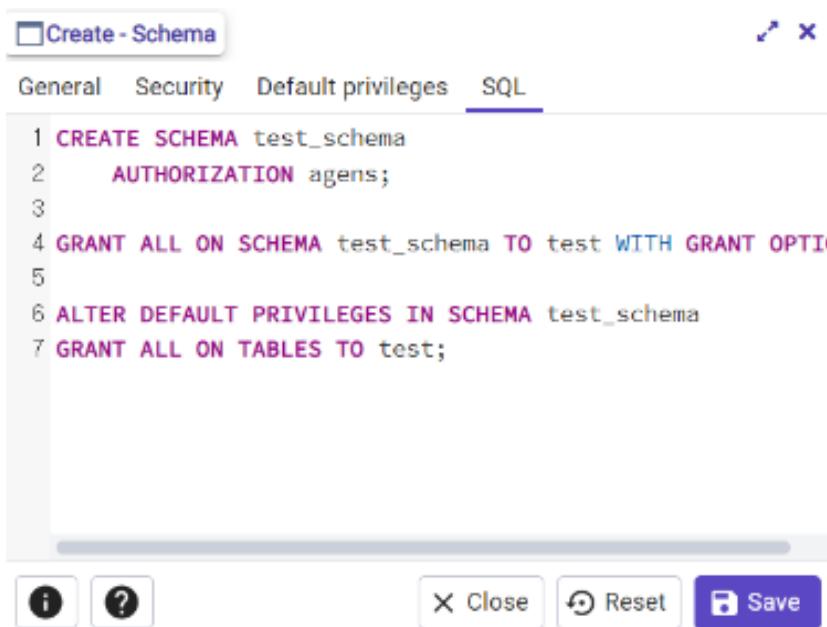
General Security **Default privileges** SQL

Tables

Grantee	Privileges	Grantor
<input type="button" value="trash"/> test	arwdDxt	agens

4) SQL

SQL 명령령문을 생성합니다.



The screenshot shows a 'Create - Schema' dialog box. At the top, there are tabs for General, Security, Default privileges, and SQL, with 'SQL' being the active tab. Below the tabs is a code editor containing the following SQL script:

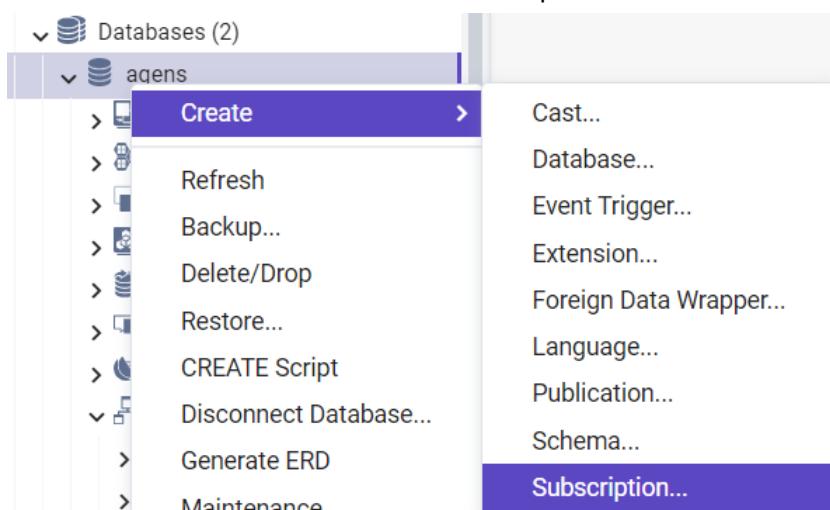
```
1 CREATE SCHEMA test_schema
2     AUTHORIZATION agens;
3
4 GRANT ALL ON SCHEMA test_schema TO test WITH GRANT OPTION
5
6 ALTER DEFAULT PRIVILEGES IN SCHEMA test_schema
7 GRANT ALL ON TABLES TO test;
```

At the bottom of the dialog box are several buttons: Close, Reset, and Save (highlighted in blue).

20.4.11 Subscription 생성

논리적 복제에 사용할 대상의 Subscription을 생성합니다. (Publication과 다른 Data Cluster에서 설정해야 합니다.)

1) Database에서 우클릭하여 CREATE > Subscription 를 이용하여 Subscription을 생성합니다.



2) General

Name - Subscription 이름을 255자 이내로 지정합니다.

Owner - Subscription의 Owner가 표시됩니다. (접속된 사용자로 자동지정)

The screenshot shows the 'Create - Subscription' dialog box with the 'General' tab selected. It contains two input fields: 'Name' with the value 'test_sub' and 'Owner' with the value 'agens'. At the bottom are buttons for 'Close', 'Reset', and 'Save'.

3) Connection

Host name / address - Publication이 설정된 Server의 IP 주소를 입력합니다.

Port - Publication DB의 접속 Port를 입력합니다.

Database - Publication DB의 이름을 입력합니다.

Username - Publication DB의 접속 user를 입력합니다.

Password - Publication DB User의 password를 입력합니다.

Connection timeout - Publication DB 접속에 Connection Timeout을 설정합니다.

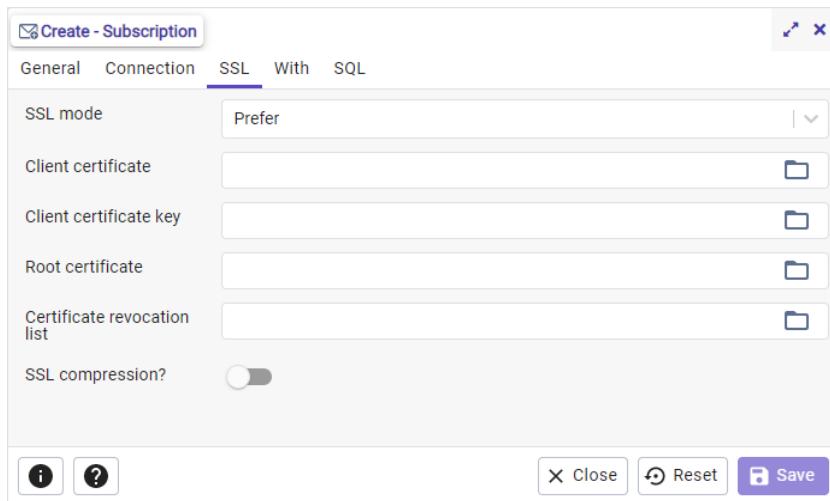
Passfile - passfile을 이용하여 접속할 경우 passfile의 경로를 지정합니다.

Publication - Publication을 지정합니다. 접속정보 입력 후 새로고침하여 드롭다운 탭에서 Publication을 선택합니다.

The screenshot shows the 'Create - Subscription' dialog box with the 'Connection' tab selected. It contains several input fields: 'Host name/address' (192.168.0.73), 'Port' (5431), 'Database' (agens), 'Username' (agens), 'Password' (....), 'Connection timeout' (10), and a 'Passfile' field which is empty. The 'Publication' field contains 'test_pub' with a delete button. Below the publication field is a note: 'Click the refresh button to get the publications'. At the bottom are buttons for 'Close', 'Reset', and 'Save'.

4) SSL

SSL 보안 접속을 설정합니다. SSL 연결을 설정하지 않은 경우 별도 설정이 필요 없습니다.



5) With

Copy data? - 복제 시작 전 기존 데이터 복제 여부를 설정합니다.

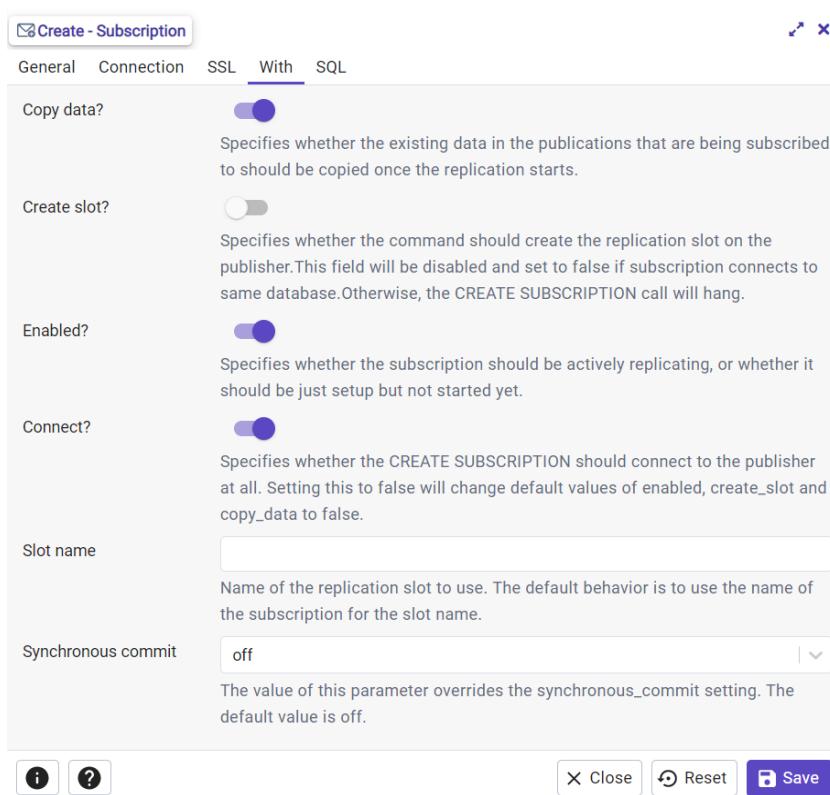
Create slot? - 데이터 복제에 사용할 복제 슬롯 생성 여부, 동일 서버에 있는 경우 기본적으로 No로 설정됩니다.

Enabled? - 복제 시작 여부를 설정합니다.

Connect? - 복제 연결 여부는 기본적으로 ON으로 설정됩니다.

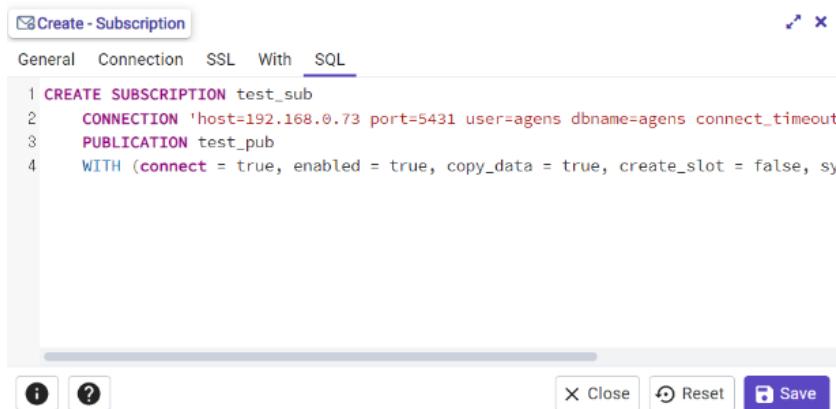
Slot name - 복제 슬롯을 사용할 경우 복제 슬롯 이름을 지정합니다.

Synchronous commit - 복제 대상 DB와의 동기화 설정 Publication의 안전을 위하여 off로 설정하는 것이 안전합니다.



6) SQL

SQL문을 생성합니다.

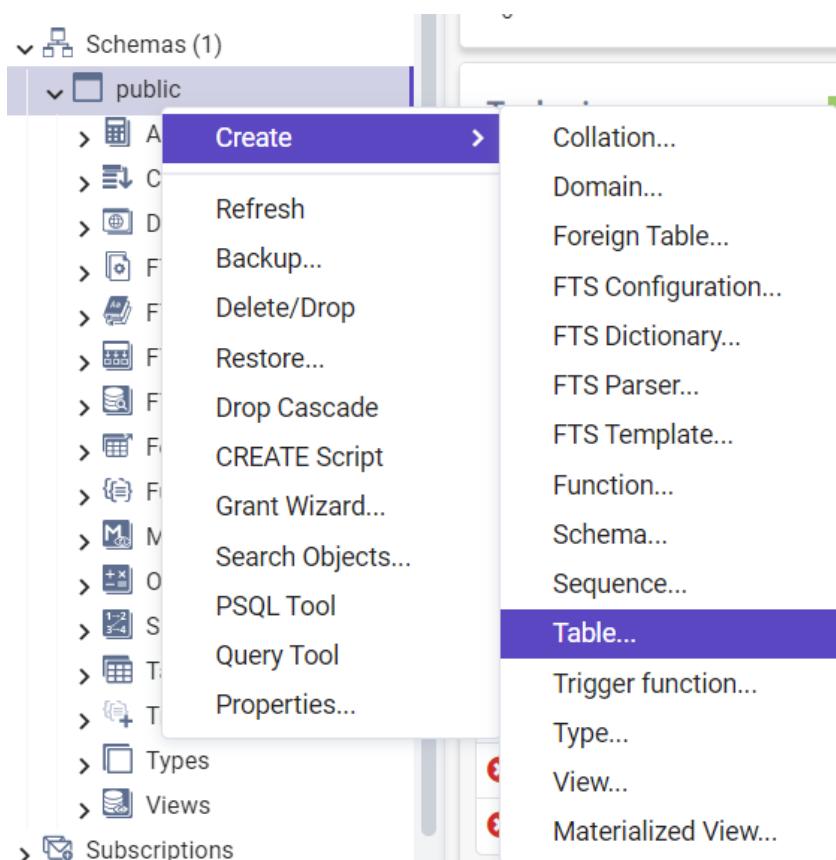


```
CREATE SUBSCRIPTION test_sub
    CONNECTION 'host=192.168.0.73 port=5431 user=agens dbname=agens connect_timeout=10'
    PUBLICATION test_pub
    WITH (connect = true, enabled = true, copy_data = true, create_slot = false, sync_commit = true)
```

20.4.12 Table 생성

Table 을 생성합니다.

- 1) Schema의 하위 Schema에서 우클릭하여 CREATE > Table 을 이용하여 Table 을 생성합니다.



2) General

Name - 테이블을 설명하는 이름을 추가합니다. 테이블은 동일한 Schema에 있는 기존 테이블, 시퀀스, 인덱스, 뷰, 외부 테이블 또는 데이터 유형과 동일한 이름을 가질 수 없습니다. 이 필드는 필수이며, 225자 이내로 테이블 이름을 지정할 수 있습니다.

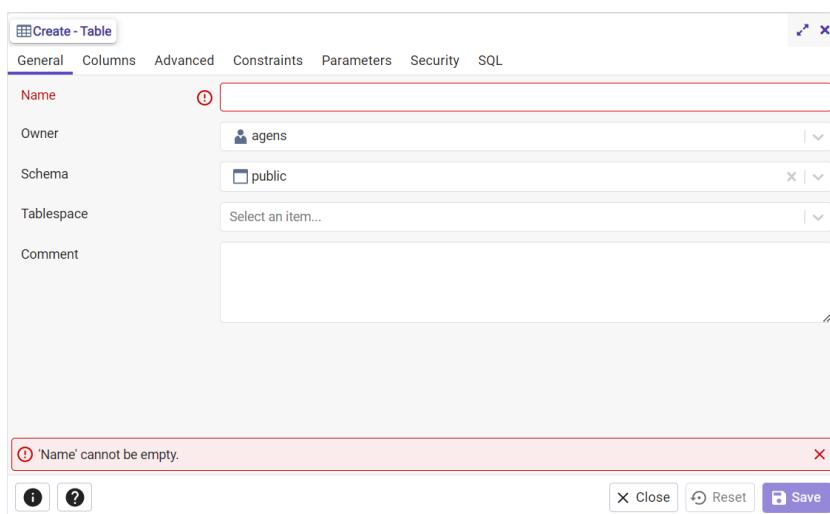
Owner - 드롭다운 목록 상자에서 테이블 소유자를 선택합니다. 기본적으로 테이블 소유자는 테이블을 생성하는 User입니다.

Schema - 드롭다운 목록 상자에서 테이블이 위치할 schema의 이름을 선택합니다.

Tablespace - 드롭다운 목록 상자에서 테이블이 저장될 tablespace를 선택합니다.

Partitioned table? - Partition 테이블을 생성하려는 경우 활성화합니다.

Comment - 테이블에 대한 메모를 저장하며, 최대 1000자 이내로 지정할 수 있습니다.



3) Columns

Inherit from table(s) - 드롭다운 목록 상자를 사용하여 상위 테이블을 지정합니다.

테이블은 선택한 상위 테이블에서 열을 상속합니다. Inherit from table(s) 필드 내부를 클릭하여 드롭다운 목록에서 테이블 이름을 선택합니다. 다른 상위 테이블을 추가하려면 반복합니다. 상위 이름 왼쪽에 있는 x를 클릭하여 선택한 테이블을 삭제합니다. 상속된 열 이름과 데이터 유형은 현재 대화 상자에서 편집할 수 없습니다. 부모 수준에서 수정해야 합니다.

추가 아이콘(+)을 클릭하여 열 테이블에서 열 이름과 해당 데이터 유형을 지정합니다.

Name - 열을 설명하는 이름을 추가합니다.

Data type - 드롭다운 목록 상자를 사용하여 열의 데이터 유형을 선택합니다. 여기에는 배열 지정자가 포함될 수 있습니다.

Length/Precision - 활성화된 경우 길이/정밀도 및 배율 필드를 사용하여 숫자 값의 최대 유효 자릿수 또는 텍스트 값의 최대 문자 수를 지정합니다.

Scale - 삭제 필요

Not NULL? - 열에 NOT NULL 조건을 지정하려면 True로 전환하십시오.

Primary key? - 열이 기본 키 제약 조건임을 지정하려면 True로 전환하십시오.

Default - 열에서 사용할 default 값을 설정합니다.

열을 추가하려면 추가 아이콘(+)을 선택합니다. 열을 삭제하려면 행 왼쪽에 있는 휴지통 아이콘을 클릭하고 행 삭제 팝업에서 삭제를 확인합니다.

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
d	bigint			<input type="checkbox"/>	<input checked="" type="checkbox"/>	3
dg	serial			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
dgf	xml			<input type="checkbox"/>	<input type="checkbox"/>	

4) Advanced

Table의 고급 기능을 설정 합니다.

RLS Policy? - ON 설정 시 행단위 보안 기능인 Row Level Security를 사용합니다.

Force RLS Policy? - ON 설정 시 강제 RLS 정책을 사용합니다.

Of type - 사용자 정의 복합 Type을 가져옵니다.

Fill factor - Table의 저장시 블록의 채우기 비율을 지정합니다. 기본값은 100입니다.

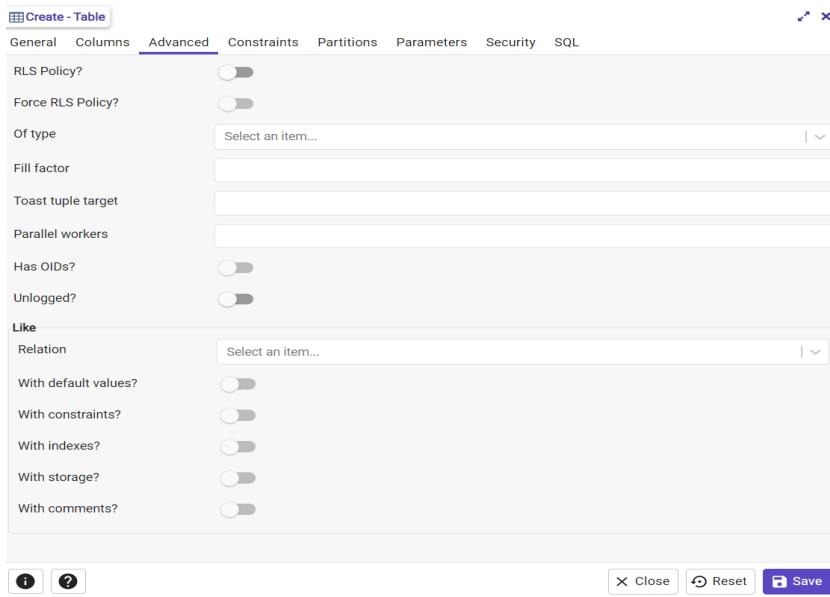
Toast tuple target - 대량 데이터 사용 시 Toast tuple target 값을 지정합니다.

Parallel workers - 테이블의 병렬처리시 병렬처리 수를 설정합니다.

Has OIDs? - 각 행에 시스템 할당 오브젝트 식별자를 지정합니다.

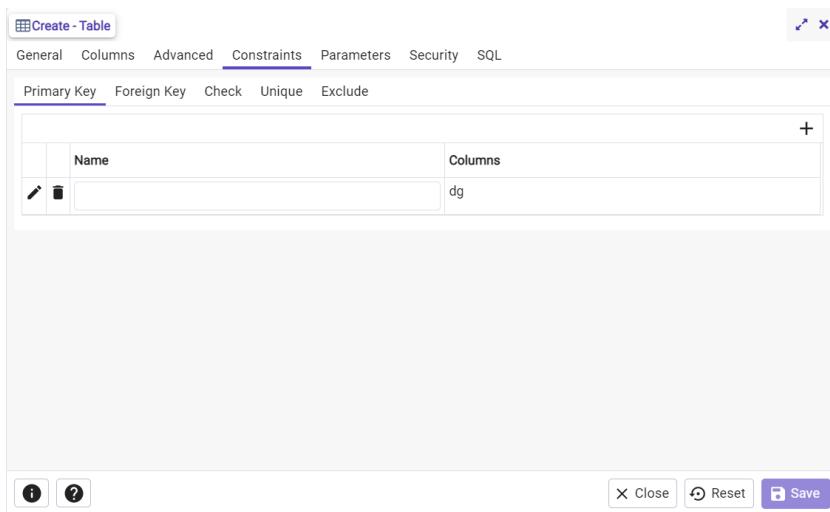
Unlogged? - Table의 Logging을 비활성화 합니다. 속도에 장점이 있지만 장애시 데이터 손실이 발생합니다.

Like - 테이블을 선택하여 해당 테이블의 정의를 사용하여 테이블을 생성합니다. **with** 옵션을 사용하여 Default values, constraint, index, storage, comments 선택하여 생성 할 수 있습니다.

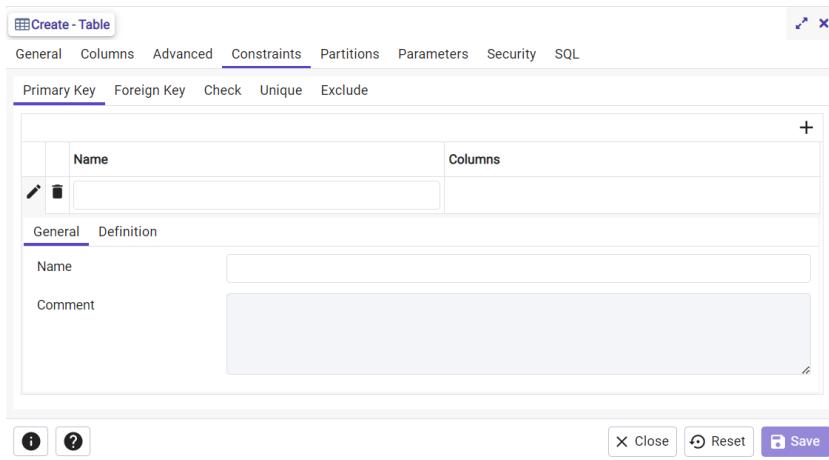


5) Constraints

제약 조건 탭의 필드를 사용하여 테이블 또는 열 제약 조건을 제공합니다. 선택적 제약 조건 절은 새 행 또는 업데이트된 행이 INSERT 또는 UPDATE 작업이 성공 하기 위해 충족해야 하는 제약 조건(테스트)을 지정합니다. 제약 조건 패널에서 다음 탭 중 하나를 선택하여 적절한 제약 조건 유형을 선택합니다.



Primary Key - 테이블의 각 행에 대한 고유 식별자를 제공합니다. 추가 아이콘(+)을 눌러서 Primary Key 를 추가할 수 있습니다.



- General

Name - Primary key 제약 조건에 대한 이름을 설정합니다.

Comment - Primary key에 대한 메모를 제공하십시오.

- Definition

Columns - 드롭다운 목록 상자에서 하나 이상의 열 이름을 선택합니다. 제약 조건에 대해 선택한 열은 고유해야 합니다.

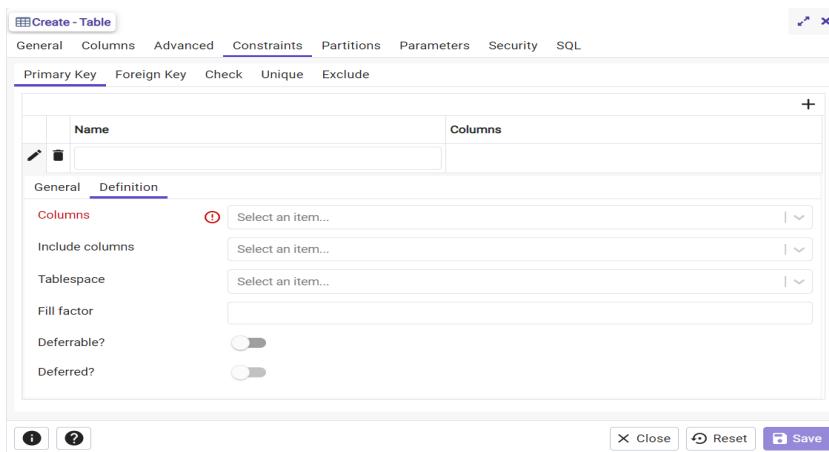
Include columns - 드롭다운 목록 상자에서 하나 이상의 열 이름을 선택합니다. primary key의 열을 하나 이상의 열의 조합으로 만들려고 할 때 추가합니다.

Tablespace - 드롭다운 목록 상자에서 primary key 제약 조건이 저장될 테이블스페이스를 선택합니다.

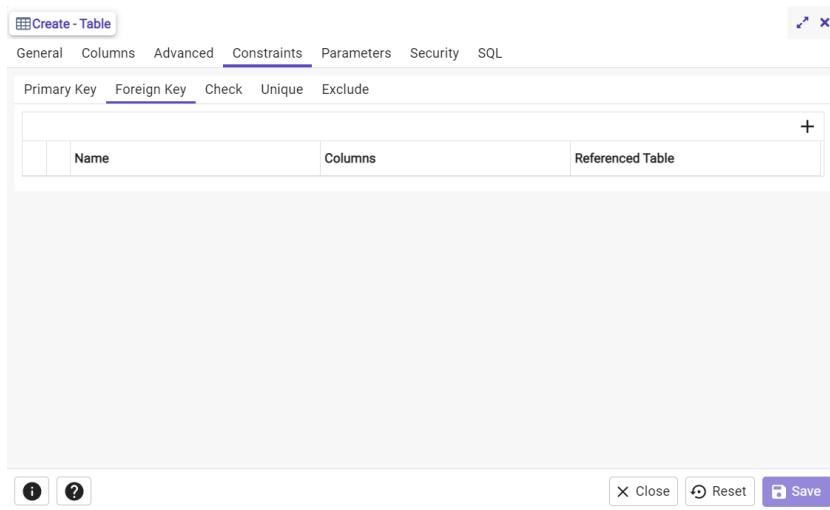
Fill factor - 테이블 및 인덱스의 채우기 비율을 지정합니다. 테이블의 채우기 비율은 10과 100 사이의 백분율입니다.

Deferrable? - 제약 조건의 타이밍을 연기할 수 있고 명령문이 끝날 때까지 연기할 수 있음을 지정합니다. 기본값은 '아니요'입니다.

Deferred? - 제약 조건의 타이밍이 명령문의 끝으로 연기되도록 지정합니다. 기본값은 '아니요'입니다.



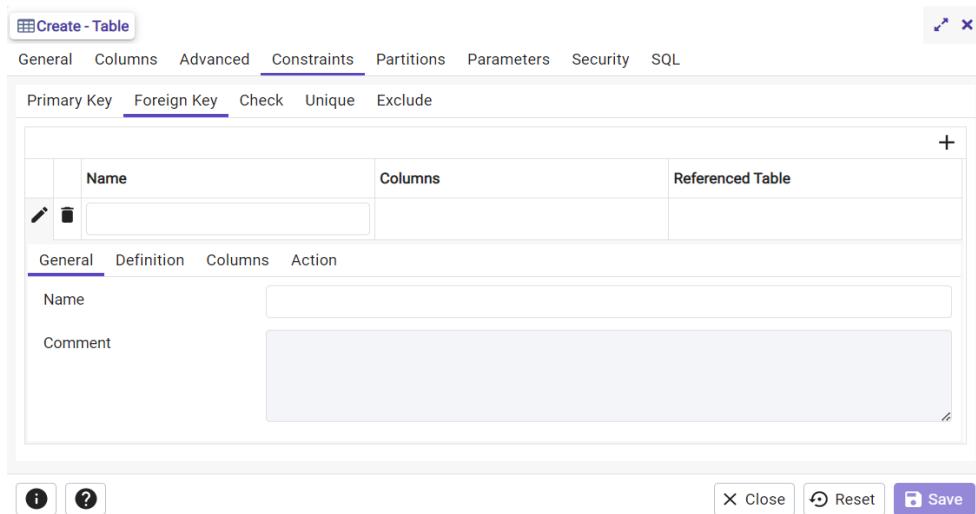
Foreign Key - 두 테이블 간의 참조 무결성을 유지합니다. 추가 아이콘(+)을 눌러서 Foreign Key 를 추가할 수 있습니다.



- General

Name - Foreign key 제약 조건에 대한 이름을 설정합니다.

Comment - Foreign key 에 대한 메모를 제공하십시오.



- Definition

Deferrable? - 제약 조건의 타이밍을 연기할 수 있고 명령문이 끝날 때까지 연기할 수 있음을 지정합니다. 기본값은 '아니요'입니다.

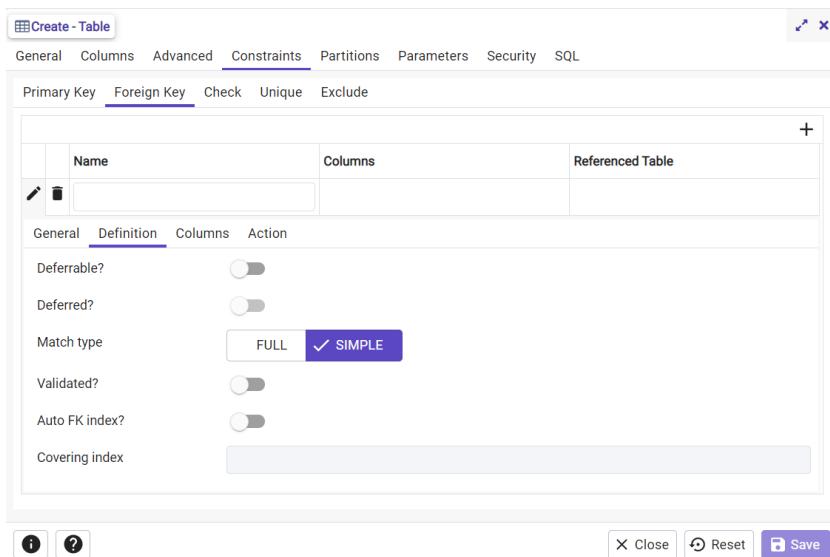
Deferred? - 제약 조건의 타이밍이 명령문의 끝으로 연기되도록 지정합니다. 기본값은 '아니요'입니다.

Match type - 제약 조건에 의해 적용되는 일치 유형을 지정합니다. 열이 null인 경우 다중 열 외래 키의 모든 열이 null이어야 함을 나타내려면 FULL을 선택하고, 단일 외래 키 열이 null일 수 있도록 지정하려면 SIMPLE을 선택합니다.

Validated? - 수정 사항을 저장할 때 유효성을 검사하도록 지시합니다.

Auto FK index? - 자동 인덱스 기능 활성화 / 비활성화를 선택합니다.

Covering index - Auto FK Index 가 활성화 되어 있는 경우에만 활성화 되며 인덱스 이름을 생성합니다.



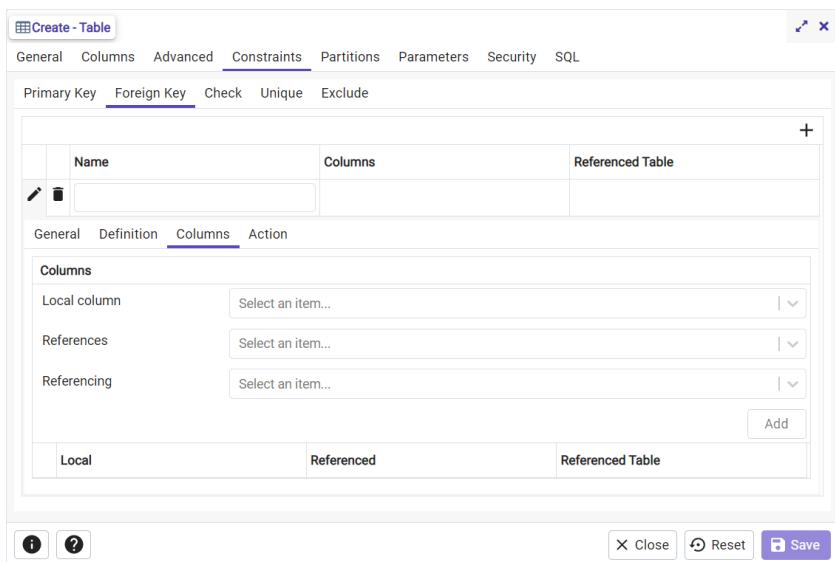
- Columns

외래 키 제약 조건을 사용하려면 테이블의 하나 이상의 열이 참조 테이블 행의 참조 열에 있는 값과 일치하는 값만 포함해야 합니다.

Local column - 드롭다운 목록 상자를 사용하여 외부 테이블과 비교할 현재 테이블의 열을 지정합니다.

References - 드롭다운 목록 상자를 사용하여 비교 열이 있는 테이블의 이름을 지정합니다.

Referencing - 드롭다운 목록 상자를 사용하여 외래 테이블의 열을 지정합니다.



- Action

드롭다운 목록 상자를 사용하여 테이블 내의 데이터가 업데이트되거나 삭제될 때 수행될 외래 키 제약 조건과 관련된 동작을 지정합니다.

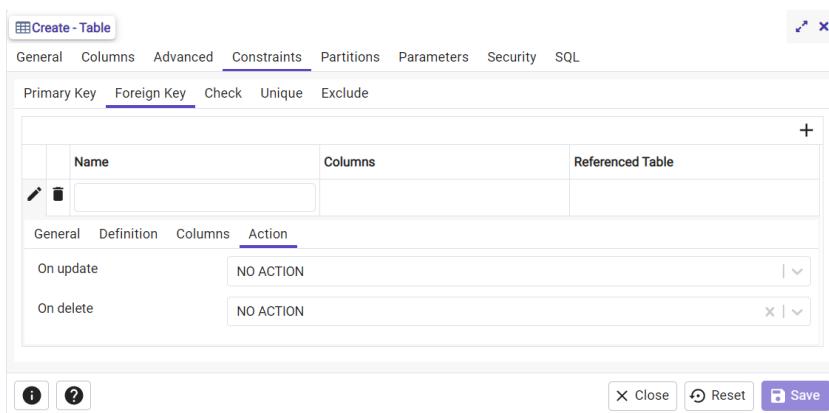
On update - 드롭다운 목록 상자를 사용하여 테이블의 데이터가 업데이트될 때 수행할 작업을 선택합니다.

On delete - 드롭다운 목록 상자를 사용하여 테이블의 데이터가 삭제될 때 수행할 작업을 선택합니다.

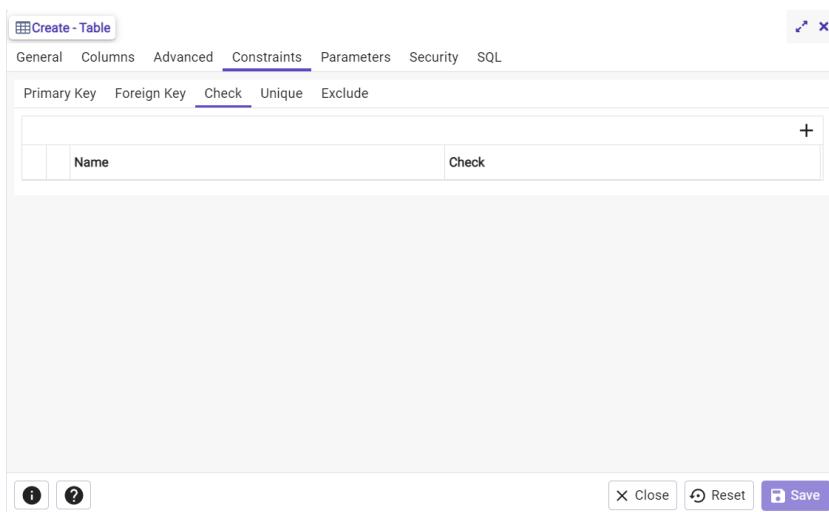
지원되는 작업은 다음과 같습니다.

작업	설명
NO ACTION	삭제 또는 업데이트가 외래 키 제약 조건 위반을 생성함을 나타내는 오류를 생성합니다. 제약 조건이 지연된 경우 참조 행이 여전히 존재하면 제약 조건 검사 시 이 오류가 생성됩니다. 이것이 기본값입니다.

RESTRICT	삭제 또는 업데이트로 인해 외래 키 제약 조건 위반이 발생함을 나타내는 오류가 발생합니다. 확인을 연기할 수 없다는 점을 제외하면 NO ACTION과 동일합니다.
CASCADE	삭제된 행을 참조하는 모든 행을 삭제하거나 참조하는 열의 값을 각각 참조된 열의 새 값으로 업데이트합니다.
SET NULL	참조 열을 null로 설정합니다.
SET DEFAULT	참조 열을 기본값으로 설정합니다. 참조된 테이블에는 기본값과 일치하는 행이 있어야 합니다(null이 아닌 경우). 그렇지 않으면 작업이 실패합니다.



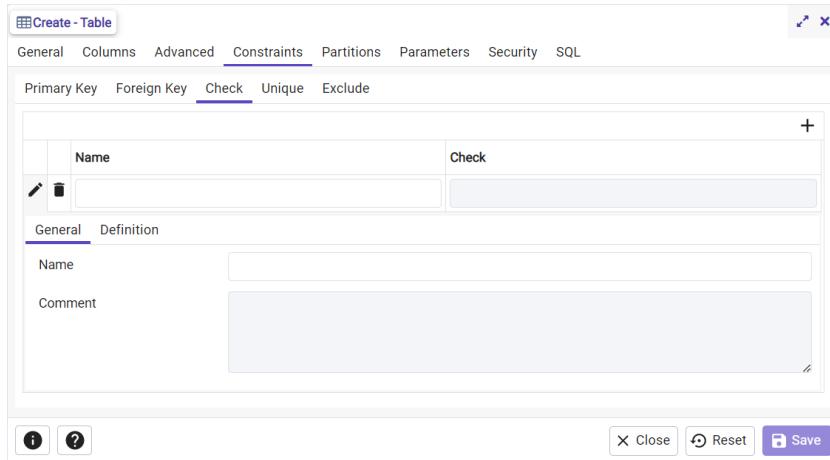
Check - 삽입 또는 수정하기 전에 데이터가 유효한지 또는 조건을 충족해야 합니다. 추가 아이콘(+)을 눌러서 Check 를 추가할 수 있습니다.



- General

Name - Check 제약 조건에 대한 이름을 설정합니다.

Comment - Check에 대한 메모를 제공하십시오.



- Definition

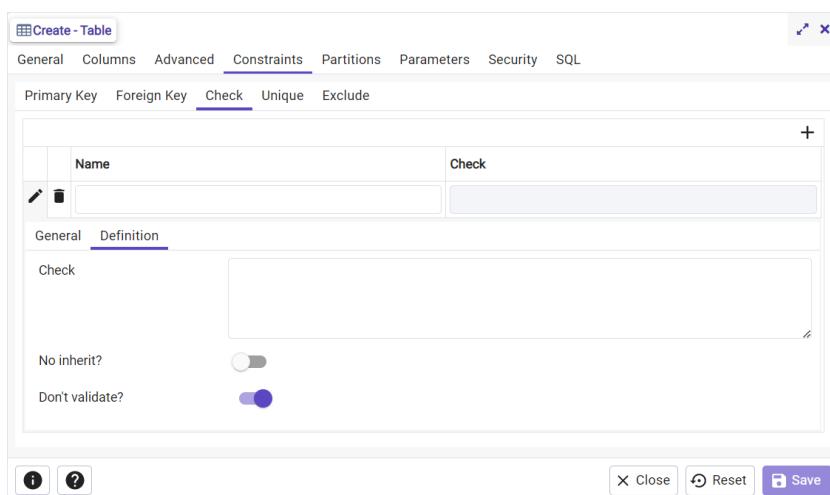
Check - 행이 충족해야 하는 표현식을 제공하십시오. 이 필드는 필수입니다.

No inherit? - 제약 조건이 테이블의 자식에 의해 자동으로 상속되지 않도록 지정합니다.

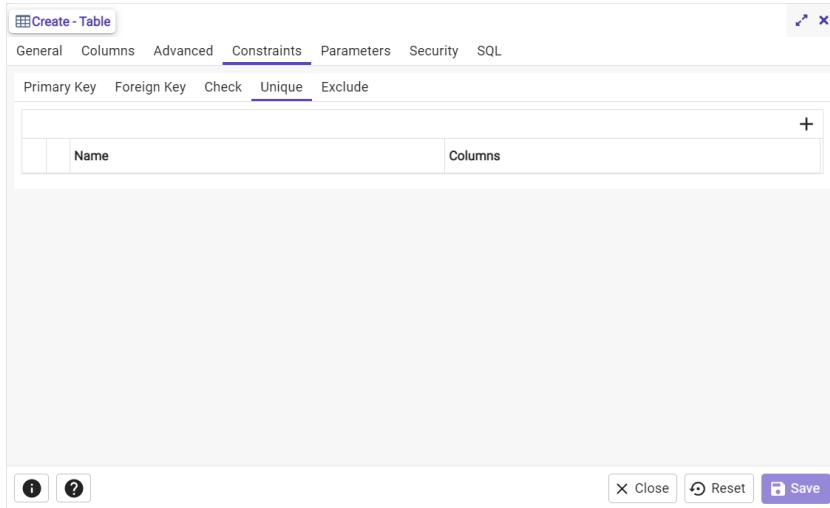
기본값은 No이며 제약 조건이 모든 자식에 의해 상속됨을 의미합니다.

Don't validate? - 기존 데이터의 유효성 검사를 건너뛰려면 '아니요' 위치로 전환합니다.

제약 조건은 테이블의 모든 행에 적용되지 않을 수 있습니다. 기본값은 '예'입니다.



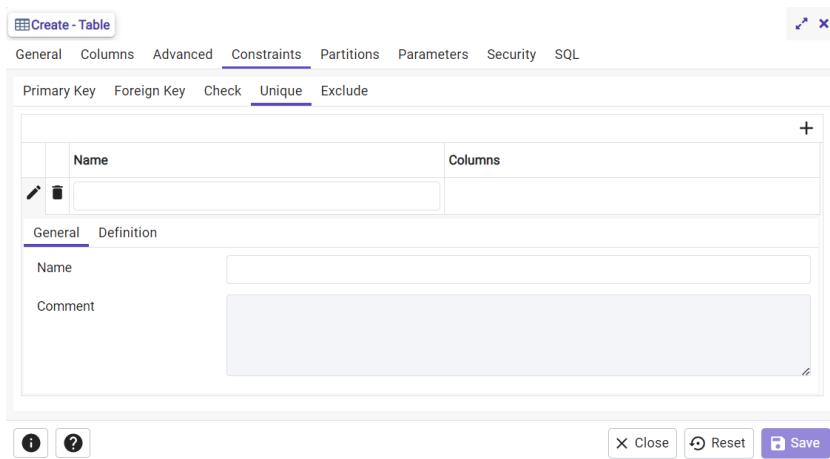
Unique - 열 또는 열 그룹에 포함된 데이터가 테이블의 모든 행에서 고유한지 확인합니다.



- General

Name - Unique 제약 조건에 대한 이름을 설정합니다.

Comment - Unique 에 대한 메모를 제공하십시오.



- Definition

Columns - 드롭다운 목록 상자에서 하나 이상의 열 이름을 선택합니다. 제약 조건에 대해 선택한 열은 고유해야 합니다. Unique 제약 조건은 동일한 테이블에 대해 정의된 Primary key 제약 조건과 달라야 합니다.

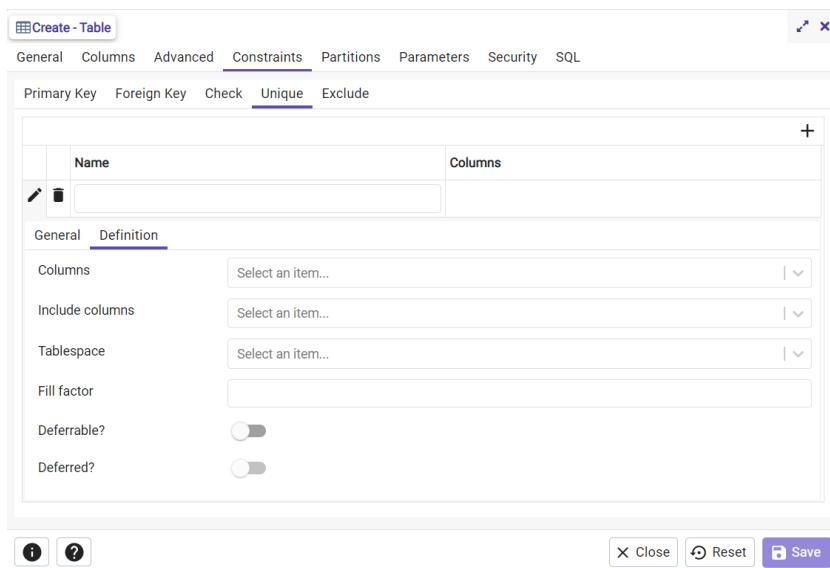
Include columns - 드롭다운 목록 상자에서 하나 이상의 열 이름을 선택합니다. primary key 의 열을 하나 이상의 열의 조합으로 만들려고 할 때 추가합니다.

Tablespace - 드롭다운 목록 상자에서 Unique 제약조건건이 저장될 테이블스페이스를 선택합니다.

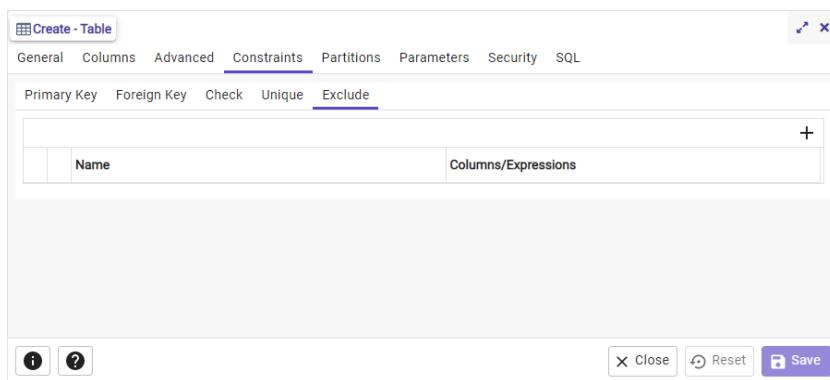
Fill factor - 테이블 및 인덱스의 채우기 비율을 지정합니다. 테이블의 채우기 비율은 10과 100 사이의 백분율이며, 기본값은 100입니다.

Deferrable? - 제약 조건의 타이밍을 연기할 수 있고 명령문이 끝날 때까지 연기할 수 있음을 지정합니다. 기본값은 '아니요'입니다.

Deferred? - 제약 조건의 타이밍이 명령문의 끝으로 연기되도록 지정합니다. 기본값은 '아니요'입니다.



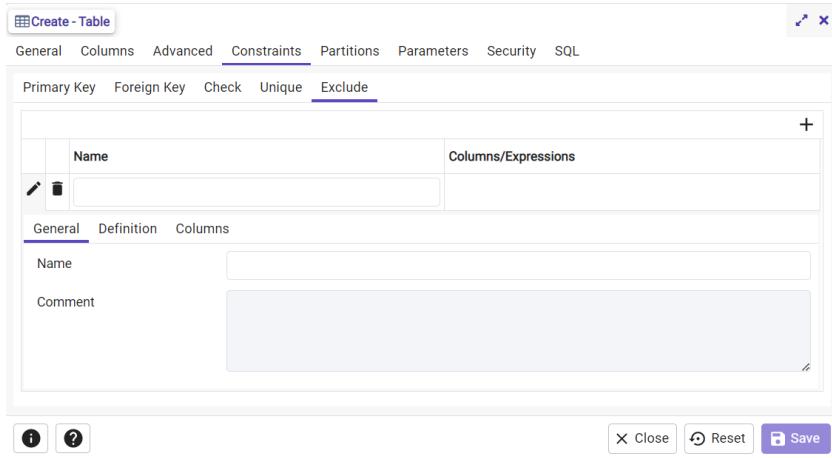
Exclude - 추가 아이콘(+)을 눌러서 Exclude 를 추가할 수 있습니다.



- General

Name - Exclude 제약 조건에 대한 이름을 설정합니다.

Comment - Exclude에 대한 메모를 제공하십시오.



- Definition

Tablespace - 드롭다운 목록 상자를 사용하여 Exclude 제약조건과 연관된 인덱스가 상주할 Tablespace를 선택합니다.

Access method - 드롭다운 목록 상자를 사용하여 Exclude 제약조건을 구현할 때 사용할 인덱스 유형을 지정합니다.

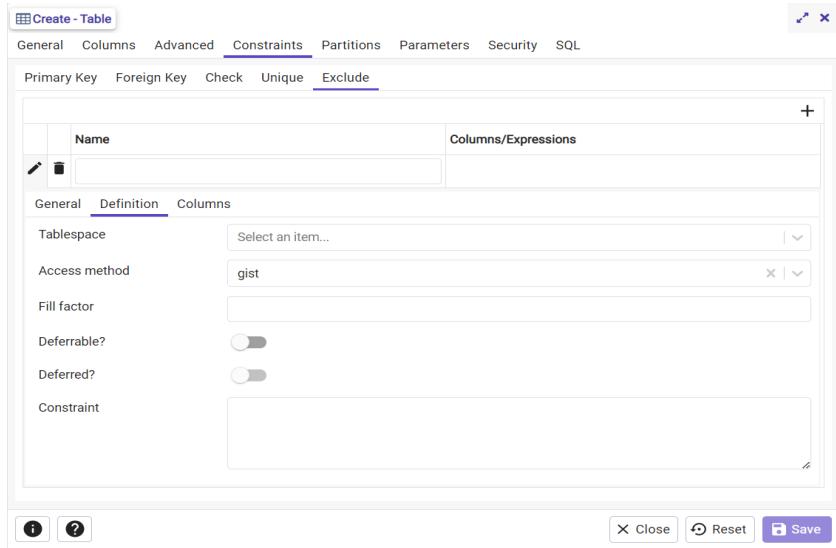
Index 유형	설명
gist	Gist 인덱스를 지정합니다.(기본값)
spgist	공간 분할된 Gist 인덱스를 지정합니다.
btree	B-tree 인덱스를 지정합니다.
hash	Hash 인덱스를 지정합니다.

Fill factor - 테이블 및 인덱스의 채우기 비율을 지정합니다. 테이블의 채우기 비율은 10과 100 사이의 백분율이며, 기본값은 100입니다.

Deferrable? - 제약 조건의 타이밍을 연기할 수 있고 명령문이 끝날 때까지 연기할 수 있음을 지정합니다. 기본값은 '아니요'입니다.

Deferred? - 제약 조건의 타이밍이 명령문의 끝으로 연기되도록 지정합니다. 기본값은 '아니요'입니다.

Constraint - 행이 테이블에 포함되기 위해 충족해야 하는 조건을 제공합니다.



- **Columns**

Is expression - column / expression 선택합니다.

Column - 드롭다운 목록 상자에서 열을 선택합니다.

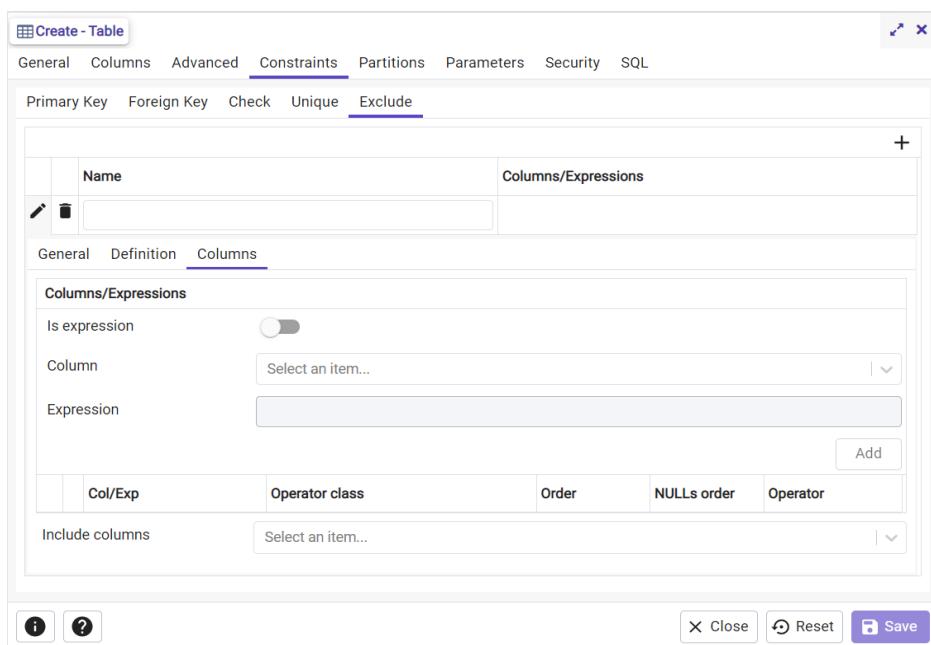
Expression - 표현식을 지정합니다.

Operator class - 드롭다운 목록 상자를 사용하여 열의 인덱스에서 사용할 연산자 클래스를 지정합니다.

Order - 내림차순 정렬 순서를 지정하려면 DESC로 설정하고, 기본값은 오름차순 정렬 순서를 지정하는 ASC입니다.

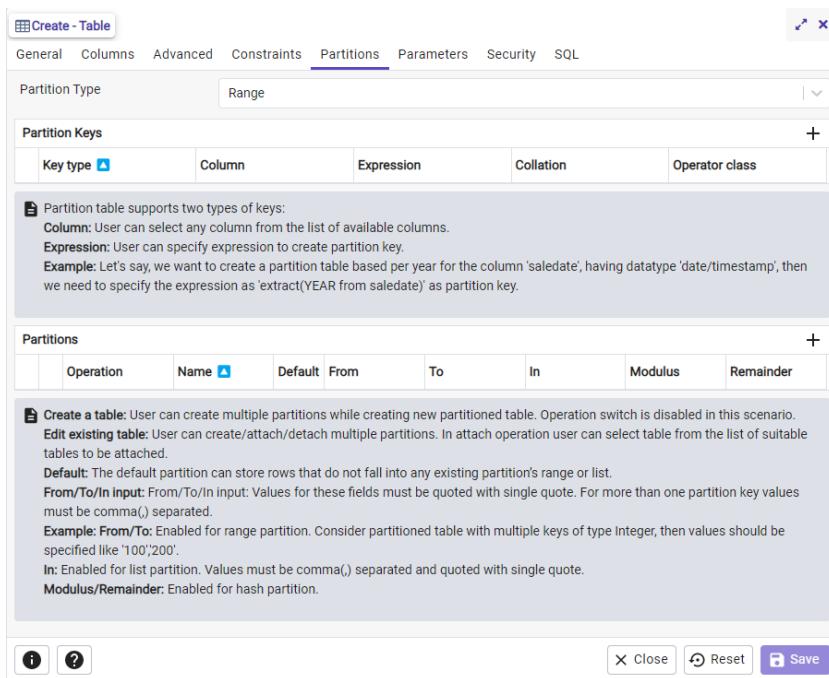
NULLs order - LAST는 NULL에 대한 오름차순 정렬 순서를 정의합니다. 기본값은 내림차순 정렬 순서를 정의하는 FIRST입니다.

Operator - 드롭다운 목록 상자를 사용하여 비교 또는 조건부 연산자를 지정합니다.



6) Partitions

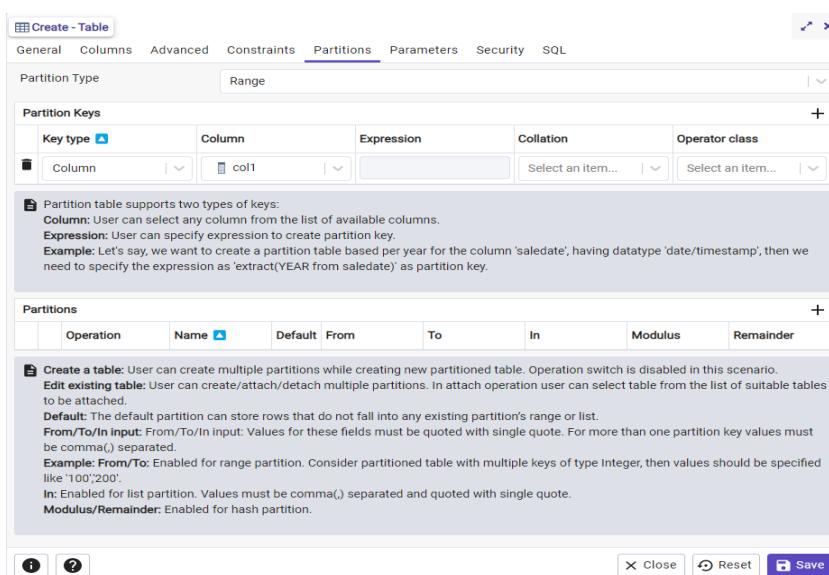
테이블의 파티션을 생성합니다. General 탭에서 Partitioned table? 항목을 활성화 하면 Partitions 탭에서 Partition 생성을 할 수 있습니다.



Partition Type - RANGE, LIST, HASH 파티션 유형을 선택할 수 있습니다.

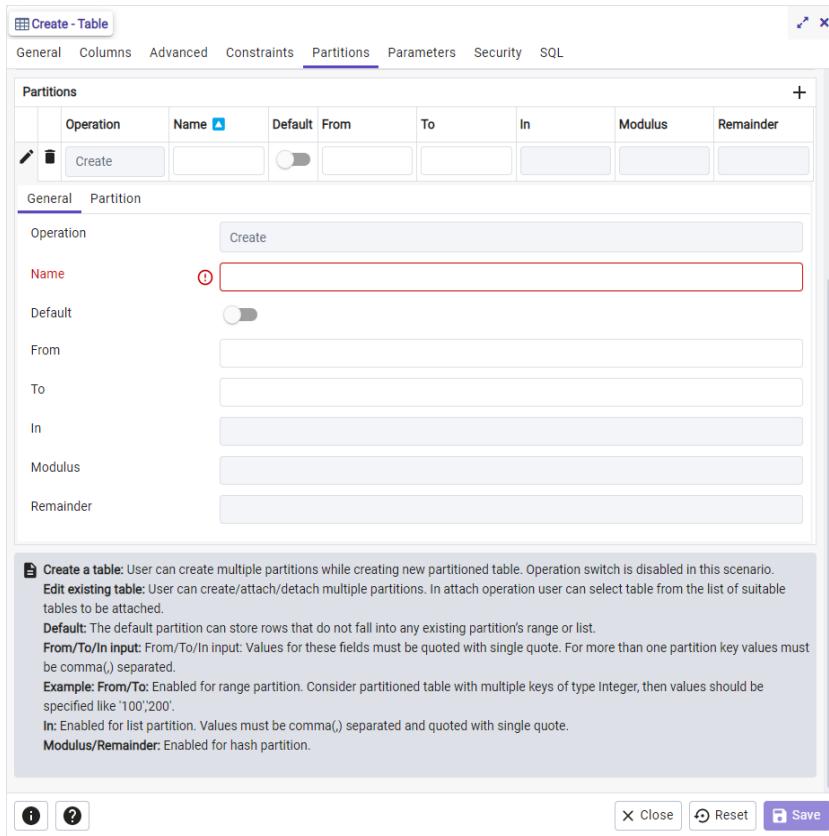
Partition Keys - 파티션 생성을 위한 Partition key를 설정할 수 있으며, 추가 아이콘(+)을 눌러서 추가할 수 있습니다.

- Key type - 파티션 키 유형을 선택합니다.
- Column - Key type에서 column을 선택한 경우 Column 필드에서 파티션 열을 선택합니다.
- Expression - Key type에서 Expression을 선택한 경우 표현식을 지정합니다.



Partition Panel - 테이블의 파티션을 정의하며 추가 아이콘(+)을 눌러서 각 파티션을 추가합니다.

- Operation - 파티션을 연결하며, 기본값은 'create' 입니다.
- Name - 파티션 이름을 설정합니다.
- Default - 파티션 유형이 Range 또는 List 이면 Default 필드가 활성화됩니다.
- From / To - 파티션 유형이 Range 이면 From 및 To 필드가 활성화됩니다.
(작은따옴표를 이용하여 데이터 포맷에 맞게 작성하여야 합니다 ex.'2020-01-01')
- In - 파티션 유형이 List 이면 In 필드가 활성화됩니다.
- Modulus / Remainder - 파티션 유형이 Hash 이면 Modulus 및 Remainder 필드가 활성화됩니다.



7) Parameters

Table, TOAST Table에 대하여 Vacuum 및 Analyze 임계값을 설정합니다.

Create - Table

General Columns Advanced Constraints Partitions **Parameters** Security SQL

Table TOAST table

Custom auto-vacuum?

Autovacuum Enabled? NOT SET YES NO

Label	Value	Default
ANALYZE scale factor		0.1
ANALYZE base threshold		50
FREEZE maximum age		200000000
VACUUM cost delay		2
VACUUM cost limit		-1
VACUUM scale factor		0.2
VACUUM base threshold		50
FREEZE minimum age		50000000
FREEZE table age		150000000

i **?**

Create - Table

General Columns Advanced Constraints Partitions **Parameters** Security SQL

Table TOAST table

Custom auto-vacuum?

Autovacuum Enabled? NOT SET YES NO

Label	Value	Default
FREEZE maximum age		200000000
VACUUM cost delay		2
VACUUM cost limit		-1
VACUUM scale factor		0.2
VACUUM base threshold		50
FREEZE minimum age		50000000
FREEZE table age		150000000

i **?**

8) Security

Privileges 추가아이콘 (+) 을 클릭하여 해당 Table의 권한을 부여합니다.

Grantee - 권한을 받을 대상을 지정 합니다.

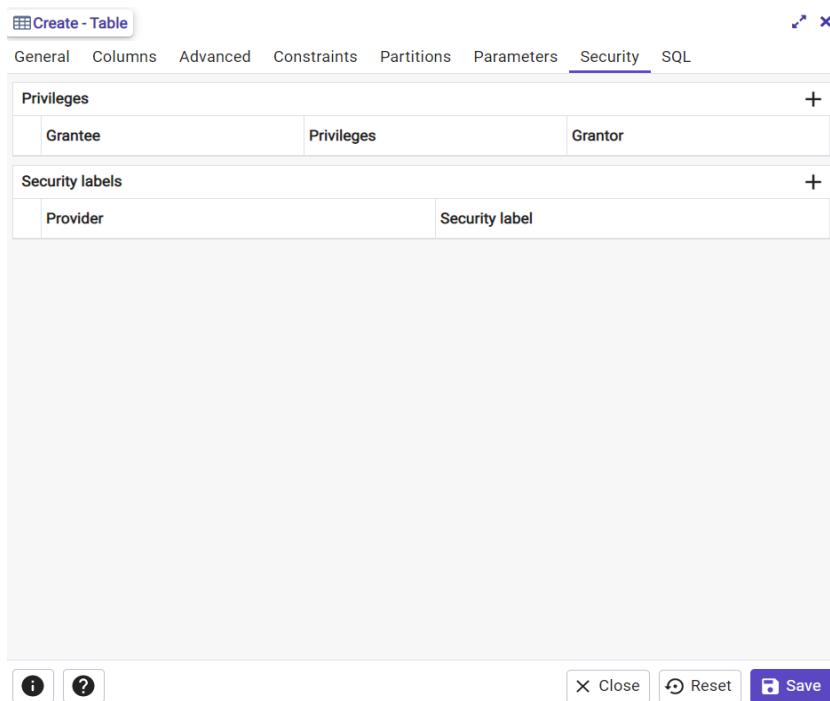
Privileges - 부여할 권한을 선택 합니다.

Grantor - 해당 권한의 소유자 표시 합니다.

Security label을 사용하는경우 추가 아이콘(+) 을 클릭하여 해당 Security labels를 생성합니다.

Provider - Security Provider 구성된 경우 Provider이름을 255자 이내로 입력합니다.

Security label - Security Label 이름을 255자 이내로 입력합니다.



9) SQL

SQL 명령문을 생성합니다.

20.5 Tools

20.5.1 ERD Tool

ERD(Entity-Relationship Diagram) 도구는 데이터베이스 테이블, 열 및 상호 관계를 그래픽으로 표현하는 데이터베이스 디자인 도구입니다. ERD는 데이터베이스 관리자가 데이터베이스를 개발하고 운영할 때 따라야 할 충분한 정보를 제공할 수 있습니다. ERD 도구를 사용하면 다음을 수행할 수 있습니다.

- 데이터베이스 테이블과 해당 관계를 디자인하고 시각화합니다.
- 다이어그램에 메모를 추가합니다.
- 깔끔한 시각화를 위해 테이블과 링크를 자동 정렬합니다.
- 다이어그램을 저장하고 나중에 열어 작업을 계속하십시오.
- 데이터베이스 설계에서 실행할 준비가 된 SQL을 생성합니다.
- 기존 데이터베이스에 대한 데이터베이스 다이어그램을 생성합니다.
- 브라우저 트리에서 다이어그램으로 테이블을 끌어다 놓습니다.

1) Tool bar

ERD 도구 모음은 자주 수행하는 작업에 대한 바로 가기를 상황에 맞는 아이콘을 사용하여 제공합니다. 이 옵션은 강조 표시된 아이콘에 대해 활성화되고 회색으로 표시된 아이콘에 대해 비활성화 됩니다.

Tool bar의 아이콘 위로 마우스를 가져가면 아이콘의 기능을 설명하는 룰팁이 표시됩니다.

File 옵션

아이콘	동작	단축키
Load from file	이전에 저장한 다이어그램을 로드합니다.	Ctrl + O
Save	이전에 저장한 다이어그램을 빠르게 저장하거나 다이어그램을 파일로 저장합니다.	Ctrl + S
Save as	새 브라우저 대화 상자를 열고 다이어그램을 저장할 새 위치를 지정합니다.	Ctrl + Shift + S

Export 옵션

아이콘	동작	단축키
Generate SQL	다이어그램에 대한 DDL SQL을 생성하고 생성된 SQL 실행 준비가 된 Query Tool을 엽니다. 각	Option/Alt + Ctrl + S

	CREATE Table DDL 앞에 DROP Table DDL 문을 두려면 With DROP Table 옵션을 선택할 수 있습니다.	
Download image	ERD 다이어그램을 이미지 형식으로 저장합니다.	Option/Alt + Ctrl + I

Editing 옵션

아이콘	동작	단축키
Add table	다이어그램에 새로운 테이블을 추가할 때 클릭하면 테이블 세부 정보를 입력할 수 있는 테이블 대화 상자가 열립니다.	Option/Alt + Ctrl + A
Edit table	다이어그램에서 테이블을 편집할 때 클릭하면 테이블 세부정보를 변경할 수 있는 테이블 대화 상자가 열립니다. 테이블을 선택하면 활성화 됩니다.	Option/Alt + Ctrl + E
Clone table	전체 테이블 구조를 복제하고 자동 생성된 이름으로 이름을 지정하고 다이어그램에 넣습니다.	Option/Alt + Ctrl + C
Drop table/link	테이블이나 링크를 삭제할 때 사용합니다. 테이블이나 링크를 선택하고 이 버튼을 클릭하여 드롭해야 합니다.	Option/Alt + Ctrl + D

테이블 관계 옵션

아이콘	동작	단축키
1M (One-to-Many link)	일대다 관계 대화 상자가 열리고 두 테이블 간의 관계가 추가됩니다. 선택한 테이블은 참조 테이블이 되며 링크의 많은 끝점을 갖게됩니다.	Option/Alt + Ctrl + O
MM (Many-to-Many link)	다대다 관계 대화 상자가 열리고 두 테이블 간의 관계가 추가됩니다. 이 옵션은 두 개의 관련 테이블에 대해 선택한 열을 기반으로 새 테이블을 만들고 연결합니다.	Option/Alt + Ctrl + M

Node 색 옵션

아이콘	동작
Fill Color	채우기 색상을 사용하여 테이블 노드의 배경색을 변경합니다. 이는 그룹 테이블을 식별하려는 경우에 유용합니다. 일단 설정되면 새로 추가된 모든 테이블은 동일한 색상을 사용합니다.
Text Color	채우기 색상을 기반으로 테이블 노드의 텍스트 색상을 변경하여 텍스트를 쉽게 읽을 수 있도록 하려면 텍스트 색상을 사용합니다.

Utility 옵션

아이콘	동작	단축키
Add/Edit note	데이터베이스를 설계하는 동안 테이블 노드에 메모를 작성할 때 사용합니다.	Option/Alt + Ctrl + N
Auto align	모든 테이블과 링크를 자동으로 정렬합니다.	Option/Alt + Ctrl + L
Show fewer details / Show more details	열 세부정보 표시를 전환할 때 사용하며, 열 세부정보를 더 적게 또는 더 많이 표시할 수 있습니다.	Option/Alt + Shift + D

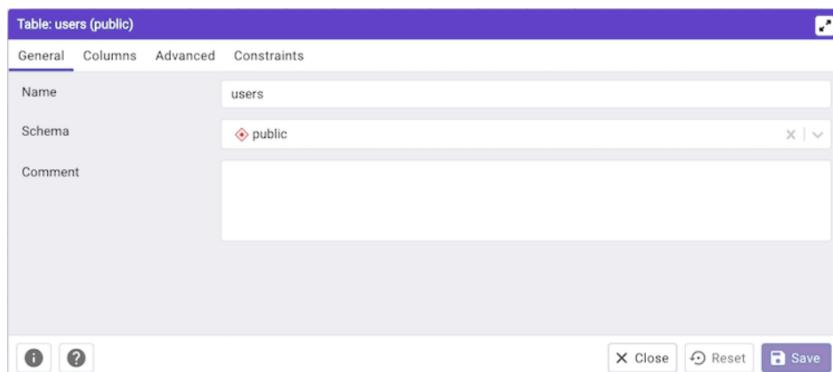
Zoom 옵션

아이콘	동작	단축키
Zoom to fit	자동으로 확대/축소하고 모든 테이블을 화면에 맞춥니다.	Option/Alt + Shift + F
Zoom in	다이어그램을 확대할 때 사용합니다.	Option/Alt + Shift + “+”
Zoom out	다이어그램을 축소할 때 사용합니다.	Option/Alt + Shift + “-”

Add table

Add table에서는 다음을 수행할 수 있습니다.

- 테이블 구조 세부 정보를 변경합니다.
- 기존 테이블을 편집하거나 새 테이블을 추가하는 데 사용할 수 있습니다.
- 다른 필드에 대한 정보는 [Table 생성](#)을 참조하십시오.



테이블 노드

테이블 노드는 테이블 세부사항을 그래픽 표현으로 표시합니다.

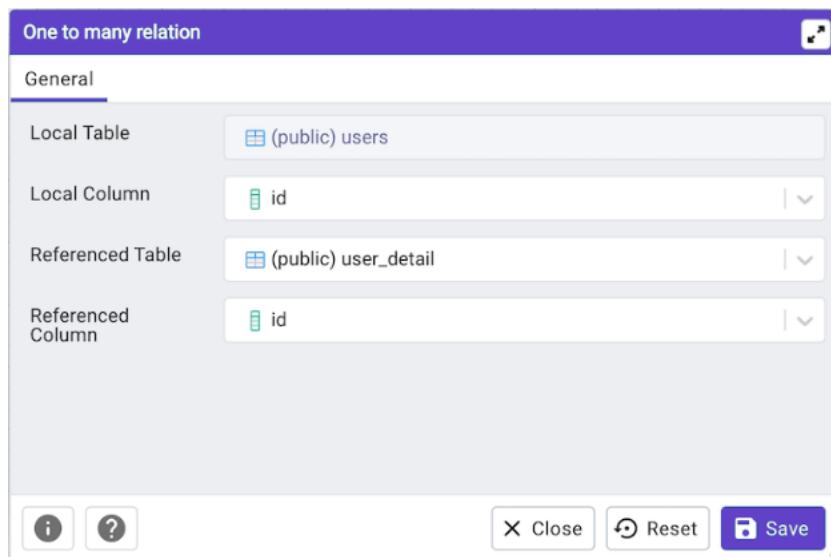
- 위쪽 표시줄에는 열 세부 정보 표시 유형을 전환하는 데 사용되는 세부 정보 툴 단추가 있습니다. 메모가 추가 된 경우에만 표시되는 메모 버튼도 있습니다. 이 버튼을 클릭하면 메모를 빠르게 변경할 수 있습니다.
- 첫 번째 행에는 테이블의 Schema 이름이 표시됩니다. (e.g. public)
- 두 번째 행에는 테이블 이름이 표시됩니다. (e.g. tb_person)
- 노드를 클릭하고 드래그하여 캔버스에서 이동할 수 있습니다.
- 테이블 노드를 두 번 클릭하거나 도구 모음에서 편집 단추를 클릭하면 테이블 세부 정보를 변경할 수 있는 테이블 대화 상자가 열립니다. 다른 필드에 대한 정보는 [Table 생성](#)을 참조하십시오.

public
tb_person
person_id integer
auth_date timestamp without time zone
auth_key character varying(255)
company character varying(255)
create_date timestamp without time zone
email character varying(255)
is_enable integer
name character varying(255)
password character varying(255)

1M(One-to-Many link)

일대다 링크 대화 상자를 사용하여 다음을 수행할 수 있습니다.

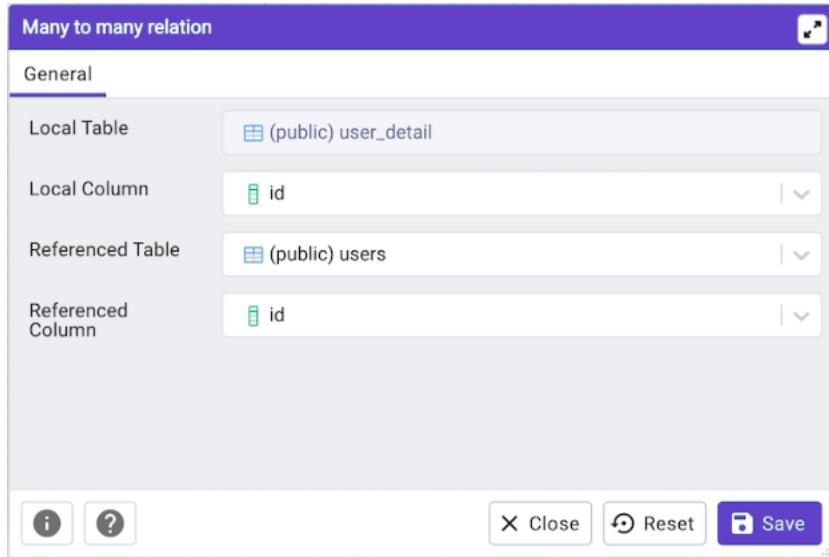
- 두 테이블 간에 외래 키 관계를 추가합니다.
- Local Table은 테이블을 참조하고 많은 끝점이 있는 테이블입니다.
- Local Column은 참조하는 열입니다.
- Referenced Table은 참조되는 테이블이며 하나의 끝점이 있습니다.
- Referenced Column은 참조되는 열입니다.



MM(Many-to-Many link)

다대다 링크 대화 상자를 사용하여 다음을 수행할 수 있습니다.

- 두 테이블 간에 다대다 관계를 추가합니다.
- 두 테이블에서 파생된 열이 있는 관계 테이블을 만들고 테이블에 연결합니다.
- 왼쪽 테이블은 연결할 첫 번째 테이블입니다. 새 관계 테이블이 있는 링크의 한 끝점을 받게 됩니다.
- 왼쪽 열 첫 번째 테이블의 열로, 항상 기본 키가 됩니다.
- 오른쪽 테이블은 연결할 두 번째 테이블입니다. 새 관계 테이블이 있는 링크의 한 끝점을 받게 됩니다.
- 오른쪽 열 항상 기본 키가 되는 두 번째 테이블의 열입니다.



테이블 링크

테이블 링크는 테이블 간의 관계를 보여줍니다.

- 링크의 한 줄 끝점에는 참조되는 열이 표시됩니다.
- 링크의 세 줄 끝점에는 참조하는 열이 표시됩니다.
- 참조되거나 참조되는 열 중 하나가 테이블에서 제거되면 링크가 삭제됩니다.
- 링크를 클릭하고 드래그하여 캔버스에서 이동할 수 있습니다.

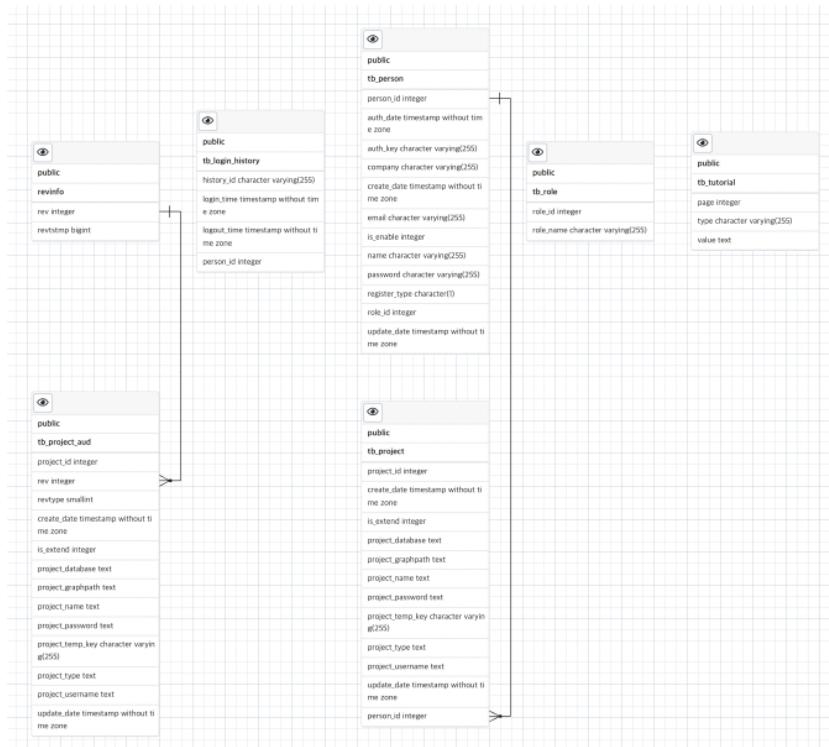
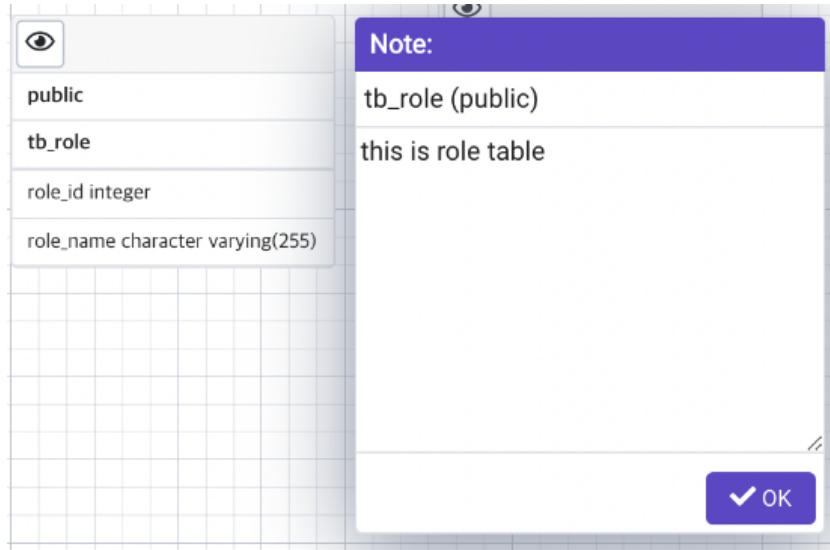


Table Note

- 메모 팝업을 사용하여 데이터베이스를 디자인하는 동안 일부 메모를 표시할 수 있습니다.
- 도구 모음 노트 단추를 사용하여 팝업을 엽니다.
- 일부 메모가 테이블에 추가되면 테이블 노드에 메모 단추가 있습니다. 버튼을 클릭하여 메모를 확인/업데이트할 수 있습니다.



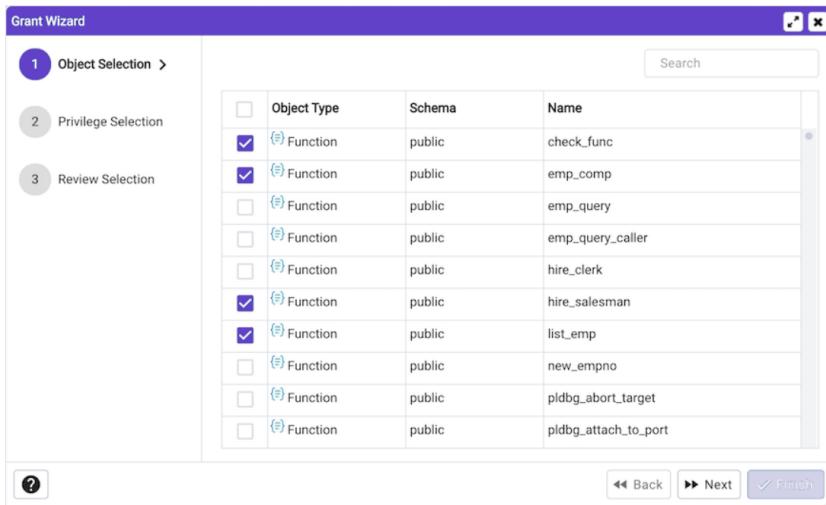
20.5.2 Grant Wizard

하나 이상의 데이터베이스 오브젝트의 권한을 관리할 수 있는 그래픽 인터페이스입니다. Search를 이용하여 데이터베이스 오브젝트, 역할 및 권한을 빠르게 선택할 수 있습니다. Object Selection, Privilege Selection 및 Review 같은 일련의 창을 통해 Grant Wizard를 구성합니다.

Grant Wizard Tool를 시작하려면 AgensEM에서 데이터베이스 오브젝트를 선택한 다음 메뉴 모음에서 Tools를 탐색하여 Grant Wizard 옵션을 클릭합니다.

Object Selection 창의 필드를 사용하여 권한을 수정하고 있는 오브젝트를 선택합니다. Object Type 또는 이름으로 Search 필드를 사용하여 데이터베이스 오브젝트를 찾거나 스크롤 막대를 사용하여 액세스 가능한 모든 오브젝트 목록을 스크롤합니다.

- 테이블의 각 행에는 오브젝트 식별자가 나열됩니다. Grant Wizard의 대상으로 오브젝트를 포함하려면 왼쪽 열의 확인란을 선택합니다. 표에는 다음이 표시됩니다.
 - Object Type 필드
 - Schema 필드
 - Name 필드 (Object Name)

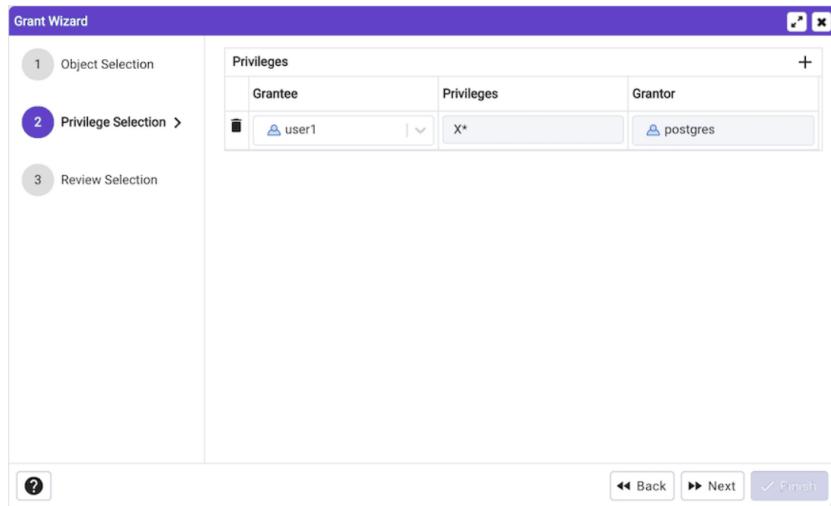


계속하려면 다음 버튼을 클릭하고 권한을 수정하지 않고 닫으려면 X 버튼을 클릭합니다.

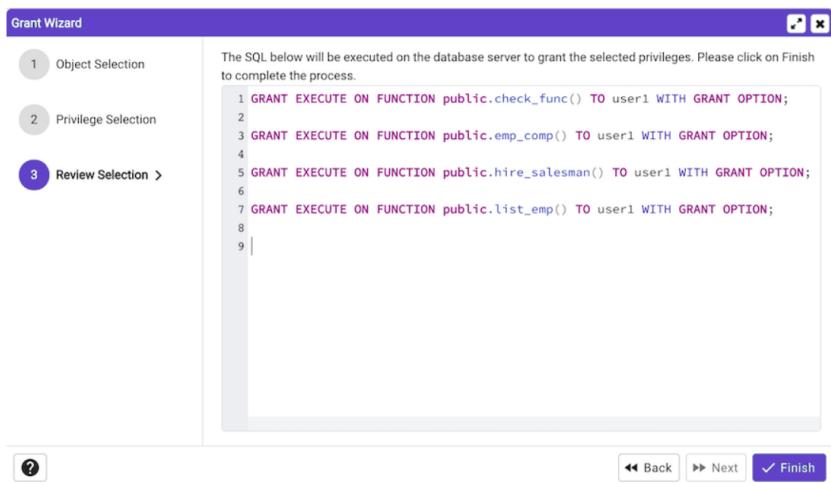
Privilege Selection 창의 필드를 사용하여 권한을 부여합니다. GRANT OPTION으로 권한을 부여하면 Grantee는 오브젝트에 대한 권한을 다른 사람에게 부여할 수 있는 권한을 갖게 됩니다. WITH GRANT OPTION이 이후에 취소되면 해당 Grantee로부터 해당 오브젝트에 대한 액세스 권한을 받은(직접 또는 일련의 부여를 통해) 모든 역할은 오브젝트에 대한 권한을 잃게 됩니다.

- 추가 아이콘(+)을 클릭하여 권한을 할당합니다.
- Grantee 필드의 드롭다운 목록 상자에서 Role/User 이름을 선택합니다.
- Privileges 필드 내부를 클릭합니다. 지정된 사용자에게 선택한 권한을 부여하려면 하나 이상의 권한 왼쪽에 있는 확인란을 선택합니다. 데이터베이스 오브젝트에 대한 권한이 이전에 부여된 경우 그룹 또는 사용자에 대한 권한을 선택 취소하면 해당 권한이 취소됩니다.
- 권한을 부여하기 위한 현재 사용자가 권한 Grantor 필드에 표시됩니다.
- 추가 아이콘(+)을 클릭하여 다른 Role/User에 권한을 할당합니다. 권한을 취소하려면 행 왼쪽에 있는 휴지통 아이콘을 클릭하고 삭제를 확인하십시오.

계속하려면 Next 버튼을 클릭하고, 추가 데이터베이스 오브젝트를 선택하거나 선택 취소하려면 Back 버튼을 클릭하고, 권한을 수정하지 않고 닫으려면 x 버튼을 클릭합니다.



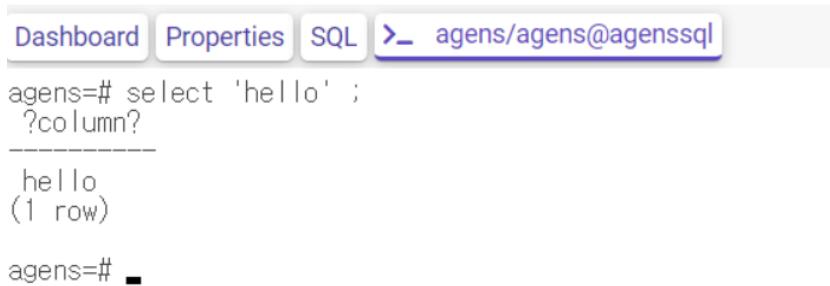
Grant Wizard 는 SQL 명령을 생성합니다.



20.5.3 PSQL Tool

PSQL Tool을 사용하면 브라우저를 통해 psql 명령줄 인터페이스를 사용하여 AgensSQL Advanced 서버에 연결할 수 있습니다.

- Tool 또는 브라우저 트리 컨텍스트 메뉴에서 PSQL Tool을 열거나 브라우저 트리 맨 위에 있는 PSQL Tool 버튼을 사용합니다.
- PSQL은 브라우저 트리에서 현재 연결된 데이터베이스에 연결합니다.



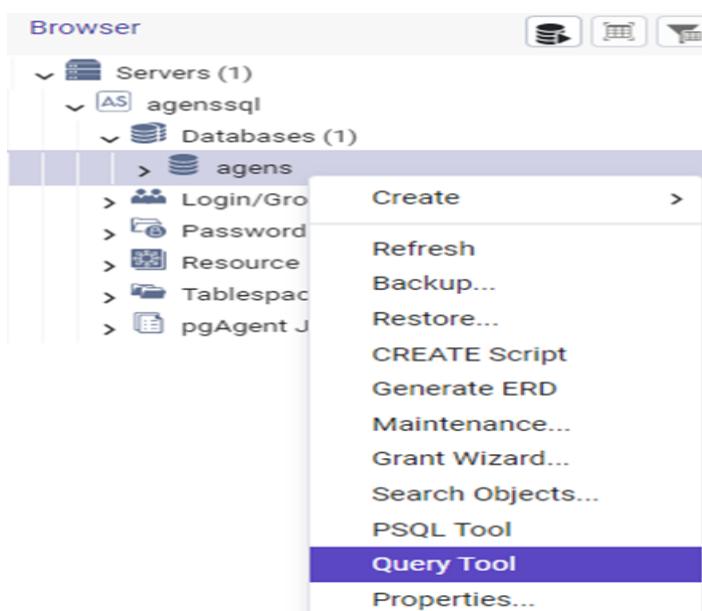
The screenshot shows the PSQL Tool interface. At the top, there are tabs: Dashboard, Properties, SQL, and a selected tab labeled > agens/agens@agenssql. Below the tabs, a command-line interface window displays the following output:

```
agens=# select 'hello' ;
?column?
-----
hello
(1 row)

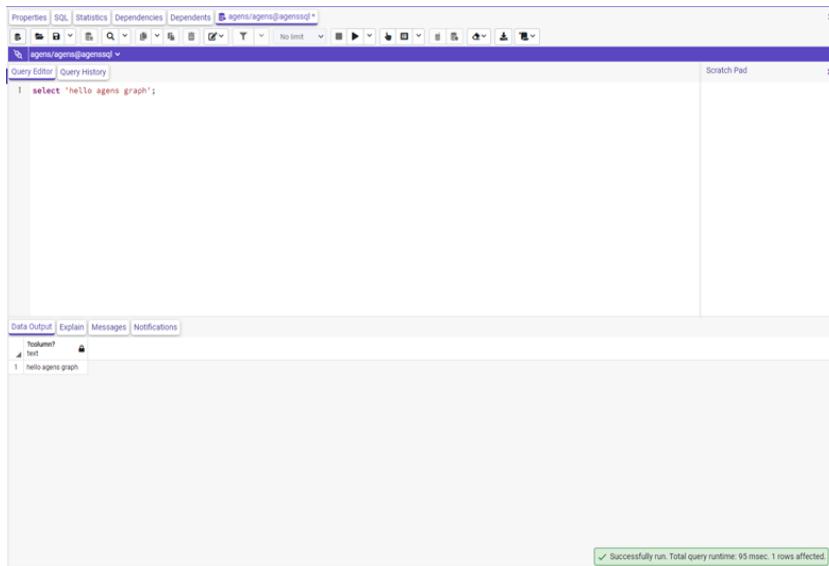
agens=# -
```

20.5.4 Query Tool

- 1) Database 서버 하위의 Database를 선택하여 우클릭 후 Query Tool을 클릭하여 SQL 편집기로 진입합니다.



- 2) SQL 편집기에서 Select 'hello AgensSQL' 이라고 입력 후 F5 key를 눌러 쿼리를 실행 합니다.
 하단 Data Output 탭에서 쿼리 결과를 확인 할 수 있습니다.
 SQL 편집기를 이용하여 Query를 편리하게 사용할 수 있습니다.



The screenshot shows the AgensGraph Management Studio interface. At the top, there's a toolbar with various icons. Below it is a navigation bar with tabs: Properties, SQL, Statistics, Dependencies, Dependents, and a connection dropdown set to 'agens/agens@agensql'. The main area has two panes: 'Query Editor' on the left containing the query 'select 'hello agens graph'' and 'Scratch Pad' on the right. Below these is a 'Data Output' tab which displays the results of the query:

Result	hello agens graph
1	hello agens graph

At the bottom right of the interface, there's a message: 'Successfully run. Total query runtime: 95 msec. 1 rows affected.'

20.5.5 Schema Diff

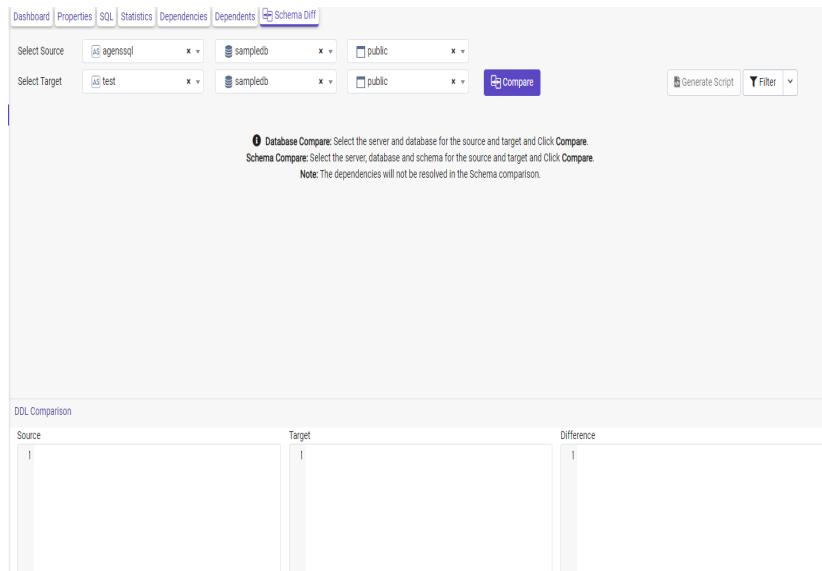
Schema Diff 는 두 데이터베이스 또는 두 Schema 간에 오브젝트를 비교할 수 있는 기능입니다.
 Tools 메뉴를 사용하여 Schema Diff에 액세스합니다.

Schema Diff 기능을 사용하면 다음을 수행할 수 있습니다.

- 데이터베이스 오브젝트를 비교하고 동기화합니다(소스에서 대상으로).
- 데이터베이스 오브젝트 간의 차이점을 시각화합니다.
- 대상 데이터베이스 오브젝트에 대한 SQL 문의 차이점을 나열합니다.
- 동기화 스크립트를 생성합니다.
- 원본 및 대상 데이터베이스 서버는 동일한 버전이어야 합니다.

Tools 메뉴에서 Schema Diff를 클릭하여 선택 패널을 엽니다. 데이터베이스를 비교하려면 소스 및 대상 서버와 데이터베이스를 선택하십시오. Schema를 비교하려면 원본 및 대상 서버, 데이터베이스 및 Schema를 선택합니다. 오브젝트를 선택한 후 비교 버튼을 클릭합니다.

각각의 탭에서 Schema Diff 의 여러 사본을 동시에 열 수 있습니다 . Schema Diff 사본을 닫으려면 탭 표시줄의 오른쪽 상단 모서리에 있는 X를 클릭하십시오. 마우스 오른쪽 버튼을 클릭하고 "Rename Panel" 옵션을 선택하여 패널의 이름을 바꿀 수 있습니다.



Preferences 대화 상자를 사용하여 다음을 지정합니다.

- Schema Diff가 새 브라우저 탭에서 열립니다. 새 브라우저 탭에서 열기 옵션을 true로 설정합니다 .
- Schema Diff는 문자열 오브젝트를 비교하는 동안 공백을 무시해야 합니다. 공백 무시 옵션을 true로 설정합니다 .
- Schema Diff는 오브젝트를 비교하는 동안 소유자를 무시해야 합니다. 소유자 무시 옵션을 true로 설정합니다 .

Schema Diff 패널은 두 개의 패널로 나뉩니다. (Objects 비교 패널 및 DDL 비교 패널)

- Schema Diff Objects 비교 패널

Objects 비교 패널에서 동일한 주 버전의 소스 및 대상 서버와 비교할 데이터베이스를 선택할 수 있습니다. 연결 여부와 관계없이 브라우저 트리 아래에 나열된 모든 서버를 선택할 수 있습니다. 연결되지 않은 서버를 선택하면 서버를 사용하기 전에 암호를 묻는 메시지가 표시됩니다.

다음으로 비교할 데이터베이스를 선택합니다. 데이터베이스는 같거나 다를 수 있습니다. (동일한 서버 내에서 또는 다른 서버에서)



서버 및 데이터베이스를 선택한 후 비교 버튼을 클릭하여 비교 결과를 얻습니다.



Database Objects의 드롭다운 목록을 사용하여 DDL 문을 봅니다.

Objects 비교 패널의 오른쪽 상단 모서리에는 다음 비교 기준에 따라 Database Objects 를 필터링하는 데 사용할 수 있는 필터 옵션이 있습니다.

- **Identical** – 동일한 SQL 문이 있는 두 데이터베이스에서 발견되면 비교 결과가 동일합니다.
- **Different** – 두 데이터베이스 모두에서 발견되지만 SQL 문이 다른 경우 비교 결과가 다릅니다.
- **Source Only** – 소스 데이터베이스에서만 발견되고 대상 데이터베이스에서는 발견되지 않는 경우 비교 결과는 소스만입니다.
- **Target Only** – 대상 데이터베이스에서만 발견되고 소스 데이터베이스에서는 발견되지 않는 경우 비교 결과는 대상만입니다.

Objects 비교 패널에서 Database Objects를 클릭하면 DDL 비교 패널에 해당 DDL 문이 표시됩니다.

- Schema Diff DDL 비교 패널

DDL 비교 패널에는 세 개의 열이 표시됩니다.

- 첫 번째 열은 원본 데이터베이스에서의 DDL 문을 표시합니다.
- 두 번째 열은 대상 데이터베이스에서의 DDL 문이 표시됩니다.
- 세 번째 열은 SQL 문의 차이점을 표시합니다.

```
Source
1 CREATE TABLE IF NOT EXISTS public.mock_data
2 (
3     id integer,
4     first_name text COLLATE pg_catalog."default",
5     last_name text COLLATE pg_catalog."default",
6     email text COLLATE pg_catalog."default",
7     gender text COLLATE pg_catalog."default",
8     ip_address text COLLATE pg_catalog."default"
9 )
10
11 TABLESPACE pg_default;
12
13 ALTER TABLE IF EXISTS public.mock_data
14     OWNER to bitrue;

Target
1 DROP TABLE IF EXISTS public.mock_data CASCADE;
```

모든 데이터베이스의 DDL 문을 검토하여 SQL 문의 차이점을 확인할 수 있습니다.

또한 원본 데이터베이스의 SQL 문을 기반으로 대상 데이터베이스에서 발견된 차이의 SQL 스크립트를 생성할 수 있습니다. 스크립트를 생성하려면 Objects 비교 패널에서 Database Objects의 확인란을 선택한 다음 Objects 비교 패널의 오른쪽 상단 모서리에 있는 스크립트 생성 버튼을 클릭합니다.

Database Objects 를 선택하고 Generate Script 을 클릭하여 새 탭에서 Query Tool을 열면 Query Editor 에 표시되는 SQL 문이 달라집니다 .

DDL Comparison Panel에서 생성된 차이를 확인하기 위해 Database Objects를 클릭하고 해당 Database Objects 의 체크박스를 선택하지 않은 경우 AgensEM은 새 탭에서 Query Tool을 열어 SQL문의 차이를 보여줍니다. Query Editor에서 Copy 버튼을 사용하여 DDL 비교 패널에서 생성된 차이를 복사 할 수도 있습니다. 대상 데이터베이스에 SQL 문을 적용하여 데이터베이스를 동기화합니다.

Source	Target	Difference
	<pre> 1 CREATE TABLE IF NOT EXISTS public.mock_data 2 (3 id integer, 4 first_name text COLLATE pg_catalog."default", 5 last_name text COLLATE pg_catalog."default", 6 email text COLLATE pg_catalog."default", 7 gender text COLLATE pg_catalog."default", 8 ip_address text COLLATE pg_catalog."default" 9) 10 11 TABLESPACE pg_default; 12 13 ALTER TABLE IF EXISTS public.mock_data 14 OWNER to bitnine; </pre>	<pre> 1 DROP TABLE IF EXISTS public.mock_data CASCADE; </pre>

20.5.6 Backup Globals

Backup Globals 대화 상자를 사용하여 클러스터 내의 모든 데이터베이스 Object와 해당 데이터베이스에서 공유하는 Objects를 다시 만드는 일반 텍스트 스크립트를 만듭니다. Objects에는 Tablespace, roles, object properties가 포함됩니다. AgensEM Query Tool을 사용하여 일반 텍스트 스크립트를 재생하고 백업에서 오브젝트를 다시 생성할 수 있습니다.

General 탭의 필드를 사용하여 다음을 지정합니다.

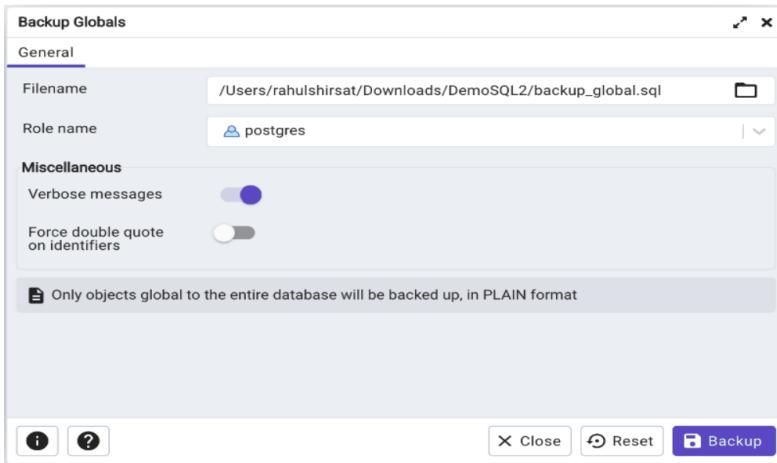
- **Filename** 필드에 백업 파일의 이름을 입력합니다. 선택적으로 오른쪽에 있는 브라우저 아이콘(줄임표)을 선택하여 디렉토리를 탐색하고 아카이브를 포함할 파일을 선택합니다.
- **Role name** 옆에 있는 드롭다운 목록 상자를 사용하여 다음을 사용하여 선택한 서버에 대한 연결 권한이 있는 User를 지정합니다. 이 User를 백업 중 인증에 사용됩니다.

Miscellaneous 필드 상자에서 스위치를 이동하여 백업에 포함되어야 하는 명령문 유형을 지정합니다.

- 백업에서 상태 메시지를 제외하려면 **Verbose messages** 스위치를 **False**로 이동합니다. 기본값은 ‘True’입니다.

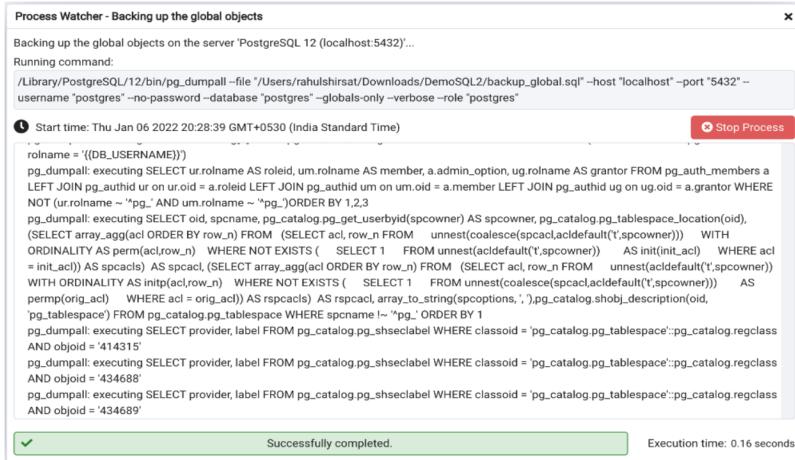
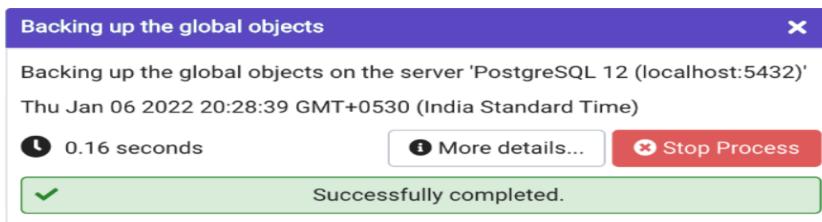
- 대소문자를 변경하지 않고 식별자에 이름을 지정하려면 Force double quote on identifiers 스위치를 True로 이동합니다. 기본값은 ‘False’입니다.

선택한 항목에 따라 명령을 작성하고 실행하려면 Backup 버튼을 클릭하십시오. 작업을 저장하지 않고 종료하려면 Close 버튼을 클릭하십시오.



Stop Process 단추를 사용하여 백업 프로세스를 중지합니다.

백업이 성공하면 성공을 확인하는 팝업 창이 나타납니다. Process Watcher를 시작하기 위한 팝업 창에서 자세한 내용을 보려면 여기를 클릭하십시오를 클릭합니다. Process Watcher는 백업과 관련된 모든 활동을 기록하고 문제 해결을 위한 추가 정보를 제공합니다.



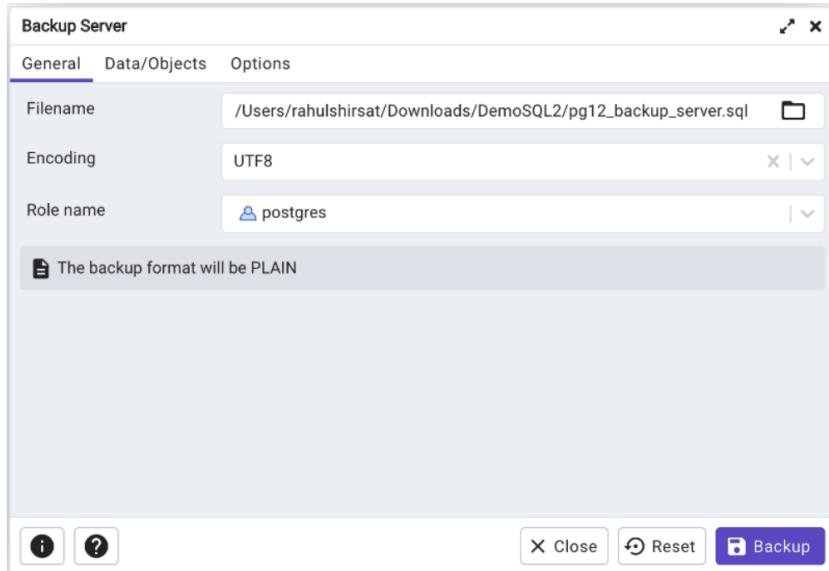
백업에 실패한 경우 Process Watcher에서 반환된 오류 메시지를 검토하여 문제를 해결하십시오.

20.5.7 Backup Server

Backup Server 대화 상자를 사용하여 선택한 서버를 다시 생성할 일반 텍스트 스크립트를 생성합니다. AgensEM Query Tool을 사용하여 일반 텍스트 스크립트를 재생하고 서버를 다시 생성할 수 있습니다.

General 탭의 필드를 사용하여 다음을 지정합니다.

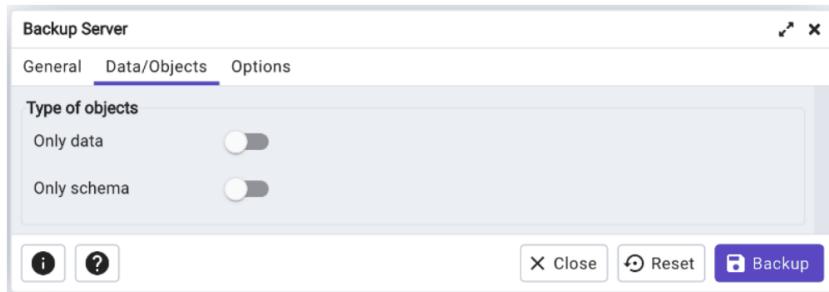
- **Filename** 필드에 백업 파일의 이름을 입력합니다. 선택적으로 오른쪽에 있는 브라우저 아이콘을 선택하여 디렉토리를 탐색하고 아카이브를 포함할 파일을 선택합니다.
- **Encoding** 드롭다운 목록 상자를 사용하여 아카이브에 사용해야 하는 문자 인코딩 방법을 선택합니다.
- **Role name** 옆의 드롭다운 목록 상자를 사용하여 선택한 서버에 대한 연결 권한이 있는 User를 지정합니다. 이 User는 백업 중 인증에 사용됩니다.



계속하려면 Data/Objects 탭을 클릭합니다. Data/Objects 탭의 필드를 사용하여 pg_dump에 해당하는 데이터 또는 AgensEM 객체와 관련된 옵션을 제공합니다.

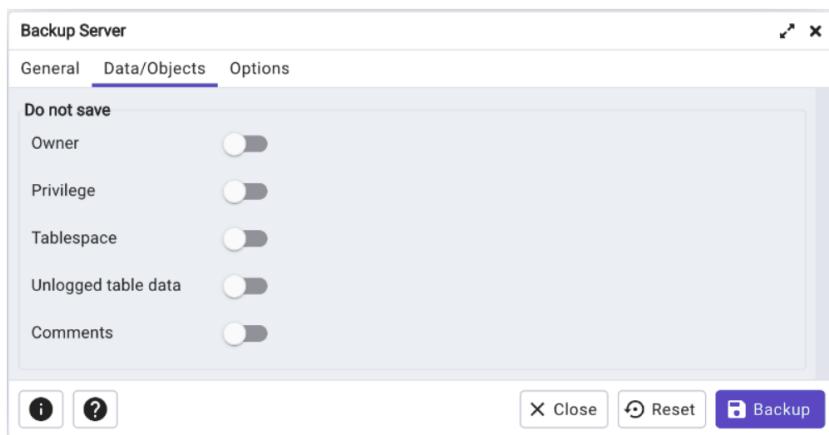
Data/Objects 필드 상자에서 스위치를 이동하여 백업할 객체 유형에 대한 세부 정보를 지정합니다.

- 백업을 데이터로 제한하려면 Only data 옆에 있는 스위치를 True로 이동합니다 .
- 백업을 Schema 수준 데이터베이스 오브젝트로 제한하려면 Only Schema 옆의 스위치를 True로 이동합니다 .



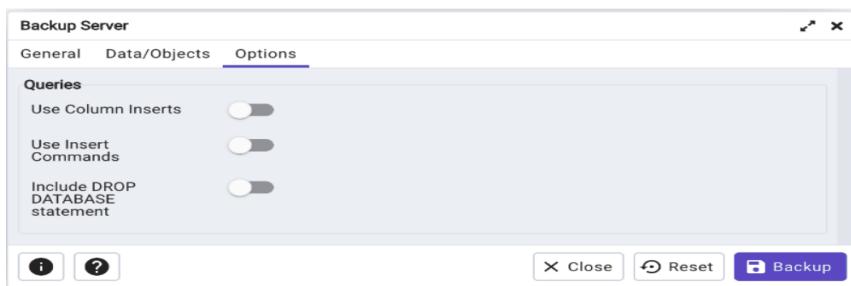
Do not save 필드 상자에서 스위치를 이동하여 백업에 포함되지 않을 오브젝트를 선택합니다.

- Owner 옆의 스위치를 True로 이동하여 오브젝트 소유권을 설정하는 명령을 제외합니다.
- 액세스 권한을 생성하는 명령을 제외하려면 Privilege 옆의 스위치를 True로 이동합니다.
- 테이블스페이스를 제외하려면 Tablespace 옆의 스위치를 True로 이동합니다 .
- Unlogged table data 옆의 스위치를 True로 이동하여 unlogged 테이블의 내용을 제외합니다.
- 설명을 설정하는 명령을 제외하려면 Comments 옆에 있는 스위치를 True로 이동합니다.



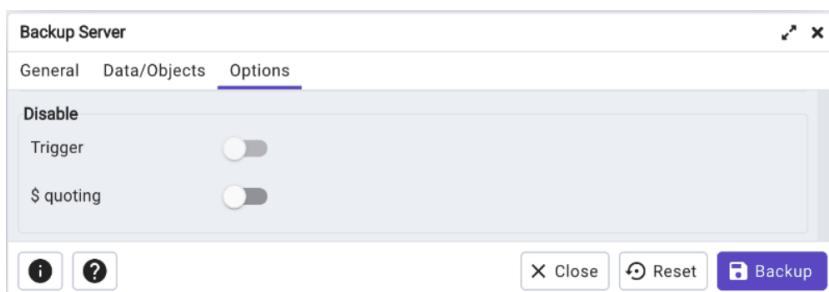
Queries 필드 상자에서 스위치를 이동하여 백업에 포함되어야 하는 문 유형을 지정합니다.

- Use Column Inserts 옆의 스위치를 True로 이동하여 데이터를 INSERT 문 형식으로 덤프하고 명시적인 열 이름을 포함합니다. 참고: 이로 인해 백업에서 복원이 느려질 수 있습니다.
- COPY 명령을 사용하는 대신 INSERT 문 형식으로 데이터를 덤프하려면 Use Insert commands 옆의 스위치를 True로 이동합니다. 참고: 이로 인해 백업에서 복원이 느려질 수 있습니다.
- Include DROP DATABASE statement 옆의 스위치를 True로 이동하여 백업 중에 오브젝트를 다시 만들기 전에 같은 이름을 가진 기존 데이터베이스 오브젝트를 삭제하는 명령을 백업에 포함합니다.



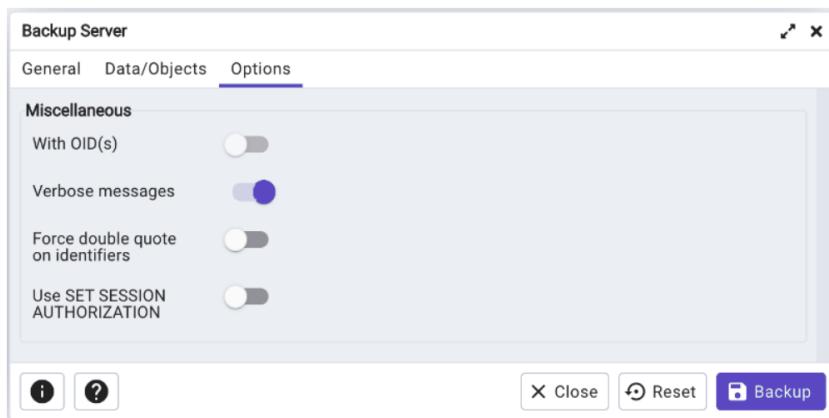
비활성화 필드 상자에서 스위치를 이동하여 백업에서 제외해야 하는 문 유형을 지정합니다.

- 데이터가 로드되는 동안 대상 테이블에서 트리거를 비활성화하는 명령을 포함하려면 Trigger (데이터 전용 백업을 생성할 때 활성화됨) 옆의 스위치를 True로 이동합니다.
- 함수 본문 내에서 달러 인용을 활성화하려면 \$ Quoting 옆의 스위치를 True로 이동하십시오. 비활성화된 경우 함수 본문은 SQL 표준 문자열 구문을 사용하여 인용됩니다.



기타 백업 옵션을 지정하려면 **Miscellaneous** 필드 상자에서 스위치를 이동하십시오.

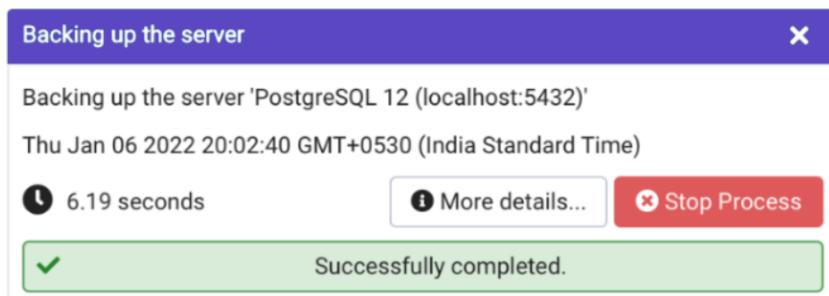
- 각 테이블에 대한 테이블 데이터의 일부로 오브젝트 식별자를 포함하려면 **With OID(s)** 옆의 스위치를 **True**로 이동합니다 .
- **Verbose messages** 옆의 스위치를 **False**로 이동하여 pg_dump가 자세한 메시지를 제외하도록 지시합니다 .
- **Force double quote on identifiers** 옆의 스위치를 **True**로 이동하여 모든 식별자를 강제로 인용합니다.
- **Use SET SESSION AUTHORIZATION** 옆의 스위치를 **True**로 이동하여 ALTER OWNER 명령 대신 SET SESSION AUTHORIZATION 명령을 사용하여 오브젝트 소유권을 결정하는 명령문을 포함합니다.

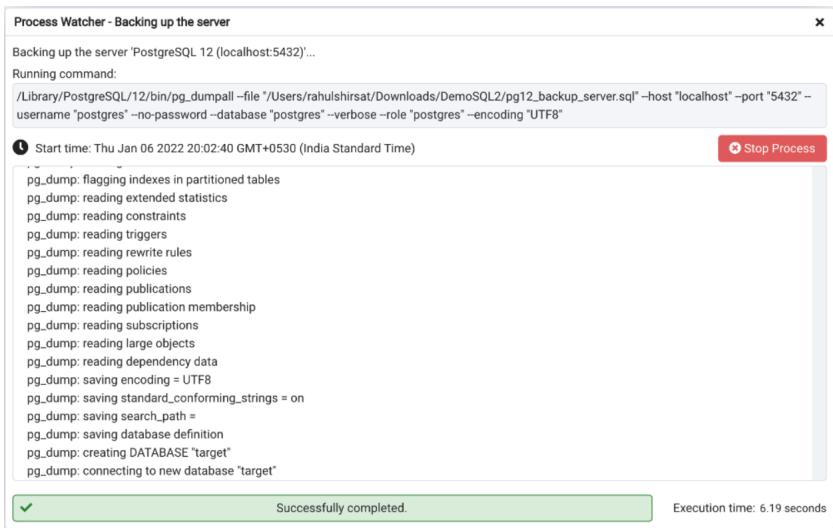


선택한 항목에 따라 명령을 작성하고 실행하려면 **Backup** 버튼을 클릭하십시오 . 작업을 저장하지 않고 종료하려면 **Close** 버튼을 클릭하십시오 .

Stop Process 버튼을 사용하여 백업 프로세스를 중지합니다.

백업이 성공하면 성공을 확인하는 팝업 창이 나타납니다. **Process Watcher**를 시작하기 위한 팝업 창에서 자세한 내용을 보려면 여기를 클릭합니다 . **Process Watcher**는 백업과 관련된 모든 활동을 기록하고 문제 해결을 위한 추가 정보를 제공합니다.





백업에 실패한 경우 Process Watcher에서 반환된 오류 메시지를 검토하여 문제를 해결하십시오.

20.5.8 Backup

AgensEM은 pg_dump 유ти리티를 사용하여 일반 텍스트 또는 아카이브 형식으로 백업을 쉽게 생성할 수 있는 방법을 제공합니다. 그런 다음 클라이언트 애플리케이션(psql 또는 Query Tool 등)을 사용하여 일반 텍스트 백업 파일을 복원하거나 AgensSQL pg_restore 유ти리티를 사용하여 아카이브된 백업을 복원할 수 있습니다. pg_dump 유ти리티에는 백업하려는 모든 데이터베이스 오브젝트에 대한 읽기 액세스 권한이 있어야 합니다.

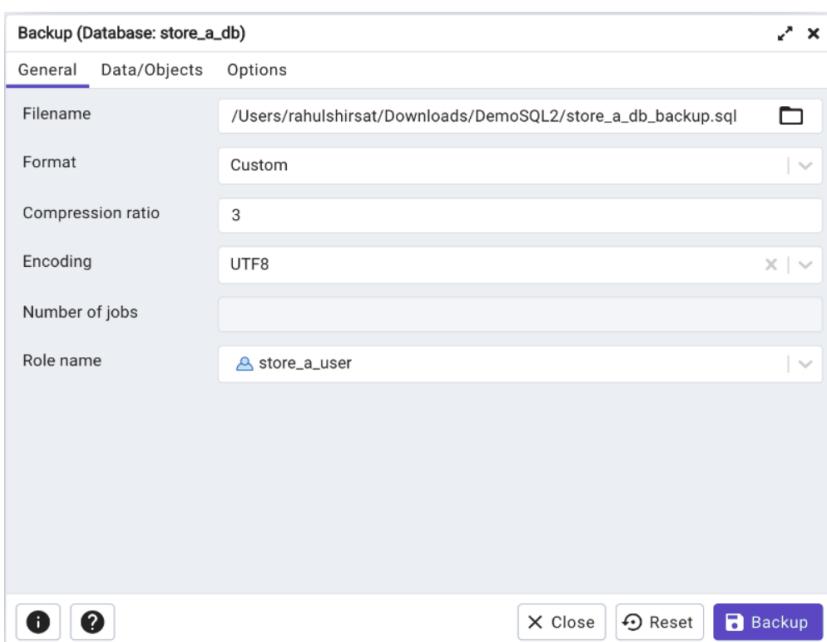
단일 테이블, Schema 또는 전체 데이터베이스를 백업할 수 있습니다. AgensEM 트리 컨트롤에서 백업 소스의 이름을 선택하고 마우스 오른쪽 버튼을 클릭하여 컨텍스트 메뉴를 열고 **Backup**을 선택하여 **Backup** 대화 상자를 엽니다. 선택한 오브젝트의 이름이 대화상자 제목 표시줄에 나타납니다.

General 탭의 필드를 사용하여 백업 매개변수를 지정합니다.

- **Filename** 필드에 백업 파일의 이름을 입력합니다. 선택적으로 오른쪽에 있는 브라우저 아이콘(…을 선택하여 디렉토리로 이동하고 아카이브를 포함할 파일을 선택합니다.
 - **Format** 필드의 드롭다운 목록 상자를 사용하여 응용 프로그램에 가장 적합한 형식을 선택하십시오. 각 형식에는 장점과 단점이 있습니다.
 - **Custom**을 선택하여 `pg_restore`와 함께 사용하여 데이터베이스 복사본을 만들 수 있는 사용자 지정 아카이브 파일을 만듭니다. 사용자 지정 아카이브 파일 형식은 `pg_restore`로 복원해야 합니다. 이 형식은 백업 파일에서 복원할 데이터베이스 오브젝트를 선택할 수 있는 기회를

제공합니다. 사용자 정의 아카이브 형식은 기본적으로 압축되므로 중대형 데이터베이스에 권장됩니다.

- Tar를 선택하여 pg_restore로 복원할 수 있는 tar 아카이브 파일을 생성합니다. .tar 형식은 압축을 지원하지 않습니다.
- 일반 텍스트 스크립트 파일을 만들려면 Plain을 선택합니다. 일반 텍스트 스크립트 파일에는 데이터베이스 오브젝트를 다시 만들고 테이블 데이터를 로드하기 위해 psql 명령줄에서 실행할 수 있는 SQL 문과 명령이 포함되어 있습니다. 일반 텍스트 백업 파일은 원하는 경우 psql 프로그램을 사용하여 데이터베이스 오브젝트를 복원하기 전에 텍스트 편집기에서 편집할 수 있습니다. 일반 형식은 일반적으로 소규모 데이터베이스에 권장됩니다. Blob에는 스크립트 덤프가 권장되지 않습니다. 스크립트 내의 SQL 명령은 데이터베이스를 마지막으로 저장된 데이터베이스 상태로 재구성합니다. 일반 텍스트 스크립트를 사용하여 다른 시스템에서 또는 다른 아키텍처에서 (수정과 함께) 데이터베이스를 재구성할 수 있습니다.
- pg_restore와 함께 사용하기에 적합한 디렉토리 형식 아카이브를 생성하려면 Directory를 선택하십시오. 이 파일 형식은 덤프되는 각 테이블 및 BLOB에 대해 하나의 파일이 있는 디렉토리와 덤프된 오브젝트를 pg_restore가 읽을 수 있는 기계 판독 가능 형식으로 설명하는 목차 파일을 만듭니다. 이 형식은 기본적으로 압축되어 있습니다.
-
- Compression ratio 필드를 사용하여 백업에 대한 압축 수준을 선택합니다. 압축을 사용하지 않으려면 0 값을 지정하십시오. 최대 압축 값을 9로 지정하십시오. tar 아카이브는 압축을 지원하지 않는다는 점에 유의하십시오.
- Encoding 드롭다운 목록 상자를 사용하여 아카이브에 사용해야 하는 문자 인코딩 방법을 선택합니다.
- Number of jobs(해당되는 경우)를 사용하여 병렬 백업에서 동시에 덤프될 테이블 수를 지정합니다.
- Role name 옆의 드롭다운 목록 상자를 사용하여 백업을 소유하는 User를 지정합니다.

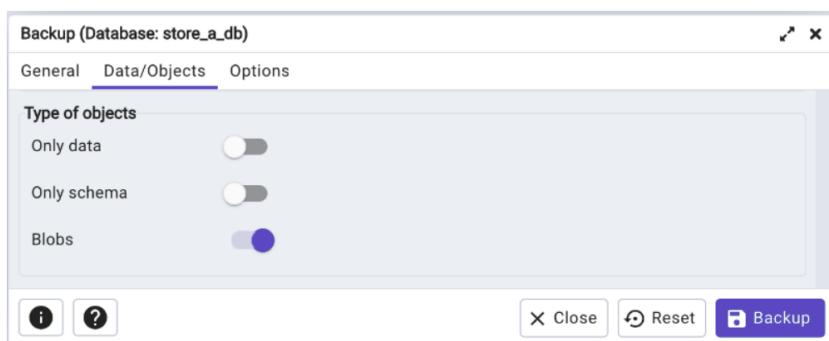


계속 하려면 Data/Objects 탭을 클릭합니다. Data/Objects 탭의 필드를 사용하여 pg_dump에 해당하는 데이터 또는 AgensEM 객체와 관련된 옵션을 제공합니다.

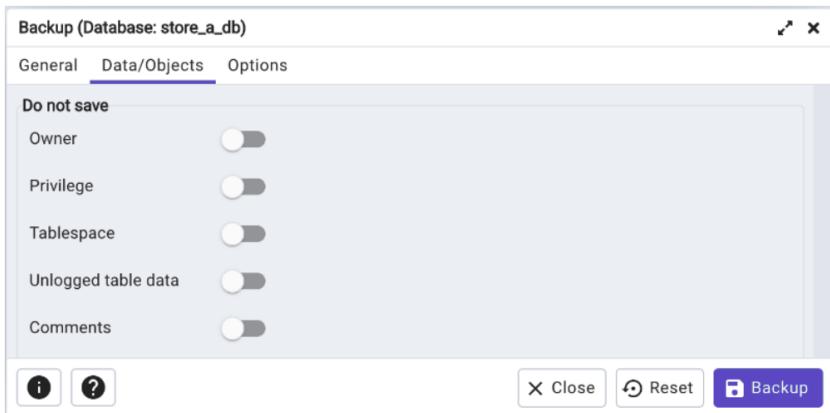
- Sections 필드 상자에서 스위치를 이동하여 백업할 오브젝트 부분을 선택합니다.
 - 데이터 또는 사후 데이터 항목 목록에 포함되지 않은 모든 데이터 정의 항목을 포함하려면 Pre-data 옆의 스위치를 True로 이동합니다.
 - Data 옆의 스위치를 True로 이동하여 실제 테이블 데이터, 큰 오브젝트 내용 및 시퀀스 값을 백업합니다.
 - 유효성 검사 제약 조건 이외의 인덱스, 트리거, 규칙 및 제약 조건 정의를 포함하려면 Post-data 옆의 스위치를 True로 이동합니다.



- Type of Objects 필드 상자에서 스위치를 이동하여 백업할 Object Type에 대한 세부 정보를 지정합니다.
 - 백업을 데이터로 제한하려면 Only data 옆에 있는 스위치를 True로 이동합니다.
 - 백업을 Schema 수준 데이터베이스 오브젝트로 제한하려면 Only schema 옆의 스위치를 이동합니다.
 - 백업에서 큰 오브젝트를 제외하려면 Blobs 옆의 스위치를 False로 이동합니다



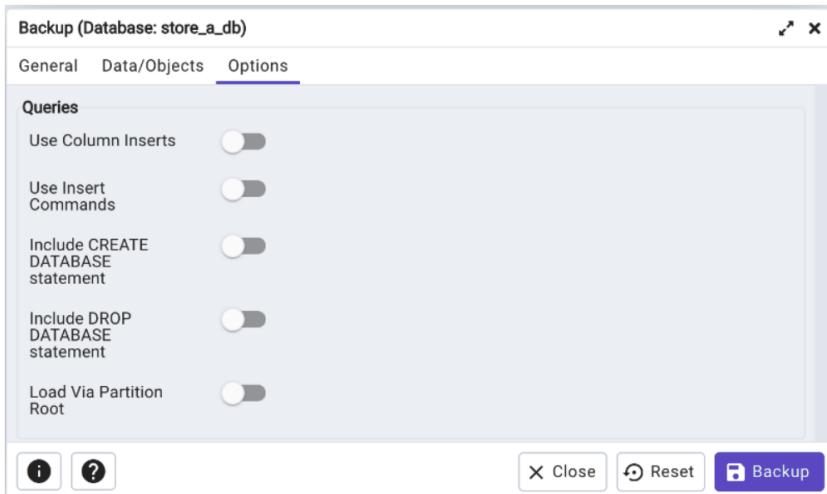
- Do not save field 상자에서 스위치를 이동하여 백업에 포함되지 않을 오브젝트를 선택합니다.
 - Owner 옆의 스위치를 True로 이동하여 오브젝트 소유권을 설정하는 명령을 제외합니다.
 - 액세스 권한을 생성하는 명령을 제외하려면 Privilege 옆의 스위치를 True로 이동합니다.
 - 테이블스페이스를 제외하려면 Tablespace 옆의 스위치를 True로 이동합니다
 - Unlogged table data 옆의 스위치를 True로 이동하여 unlogged 테이블의 내용을 제외합니다.
 - 설명을 설정하는 명령을 제외하려면 Comment 옆에 있는 스위치를 True로 이동합니다. 참고: 이 옵션은 11보다 크거나 같은 데이터베이스 서버에서만 볼 수 있습니다.



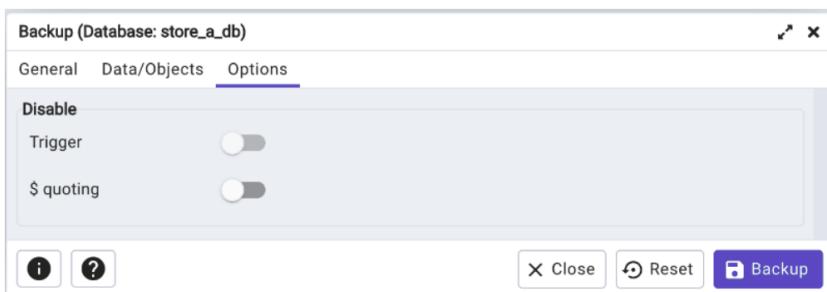
계속하려면 Options 탭을 클릭하십시오. 이러한 추가 필드를 사용하여 DDL 문, 상세 메시지 포함 또는 pg_dump 옵션에 해당하는 세션 권한 설정 사용과 같은 옵션을 지정합니다.

- Queries 필드 상자에서 스위치를 이동하여 백업에 포함되어야 하는 문 유형을 지정합니다.
 - Use Column Inserts 옆의 스위치를 True로 이동하여 데이터를 INSERT 문 형식으로 덤프하고 명시적인 열 이름을 포함합니다. 참고: 이로 인해 백업에서 복원이 느려질 수 있습니다.
 - COPY 명령을 사용하는 대신 INSERT 문 형식으로 데이터를 덤프하려면 Use Insert commands 옆의 스위치를 True로 이동합니다. 참고: 이로 인해 백업에서 복원이 느려질 수 있습니다.
 - Include CREATE DATABASE statement 옆의 스위치를 True로 이동하여 백업을 복원할 때 새 데이터베이스를 생성하는 명령을 백업에 포함합니다.

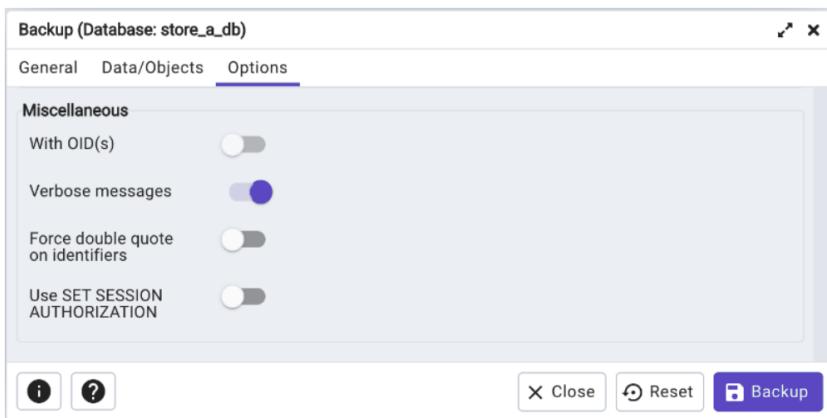
- Include DROP DATABASE statement 옆의 스위치를 True로 이동하여 백업 중에 오브젝트를 다시 만들기 전에 같은 이름을 가진 기존 데이터베이스 오브젝트를 삭제하는 명령을 백업에 포함합니다.
- Load Via Partition Root 옆의 스위치를 True로 이동하여 파티션된 테이블에 대한 COPY 또는 INSERT 문을 덤프할 때 파티션 자체가 아닌 이를 포함하는 파티션 계층 구조의 루트를 대상으로 지정하십시오.



- Disable 필드 상자에서 스위치를 이동하여 백업에서 제외해야 하는 문 유형을 지정합니다.
 - 데이터가 로드되는 동안 대상 테이블에서 트리거를 비활성화하는 명령을 포함하려면 Trigger (데이터 전용 백업을 생성할 때 활성화됨) 옆의 스위치를 True로 이동합니다.
 - 함수 본문 내에서 달러 인용을 활성화하려면 \$ quoting 옆의 스위치를 True로 이동하십시오. 비활성화된 경우 함수 본문은 SQL 표준 문자열 구문을 인용합니다.



- 기타 백업 옵션을 지정하려면 miscellaneous 필드 상자에서 스위치를 이동하십시오.
 - 각 테이블에 대한 테이블 데이터의 일부로 오브젝트 식별자를 포함하려면 With OID(s) 옆의 스위치를 True로 이동합니다.
 - Verbose messages 옆의 스위치를 왼쪽 위치로 이동하여 pg_dump가 자세한 메시지를 제외하도록 지시합니다.
 - Force double quote on identifiers 옆의 스위치를 True로 이동하여 모든 식별자를 강제로 인용합니다.
 - Use SET SESSION AUTHORIZATION 옆의 스위치를 True로 이동하여 ALTER OWNER 명령 대신 SET SESSION AUTHORIZATION 명령을 사용하여 오브젝트 소유권을 결정하는 명령문을 포함합니다.

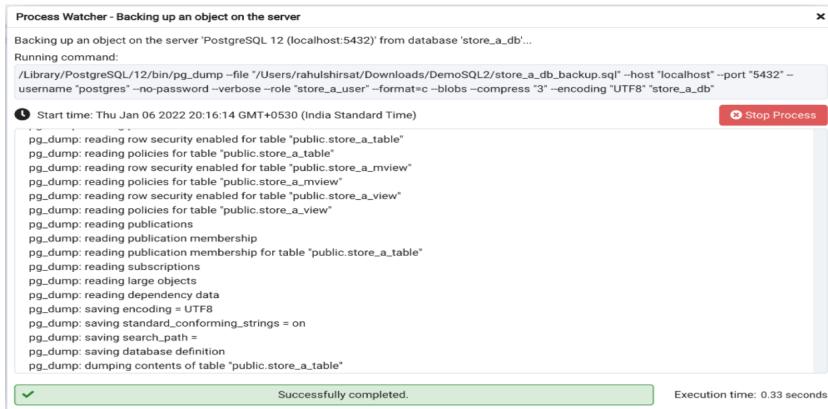
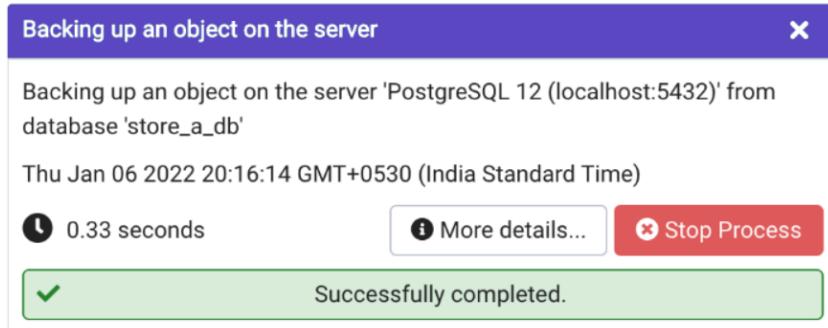


pg_dump 명령에 통합될 세부 정보를 지정한 경우:

- Backup 버튼을 클릭하여 Backup 대화상자에서 선택한 항목을 기반으로 백업을 작성하는 명령을 작성하고 실행합니다.
- 작업을 저장하지 않고 종료하려면 Close 버튼을 클릭하십시오 .

Stop Process 버튼을 사용하여 백업 프로세스를 중지합니다.

백업이 성공하면 성공을 확인하는 팝업 창이 나타납니다. 팝업 창에서 자세한 정보를 클릭하여 Process Watcher를 실행하십시오 . Process Watcher는 백업과 관련된 모든 활동을 기록하고 문제 해결을 위한 추가 정보를 제공합니다.



백업에 실패한 경우 Process Watcher에서 백업 명령이 반환한 오류 메시지를 검토할 수 있습니다

20.5.9 Restore

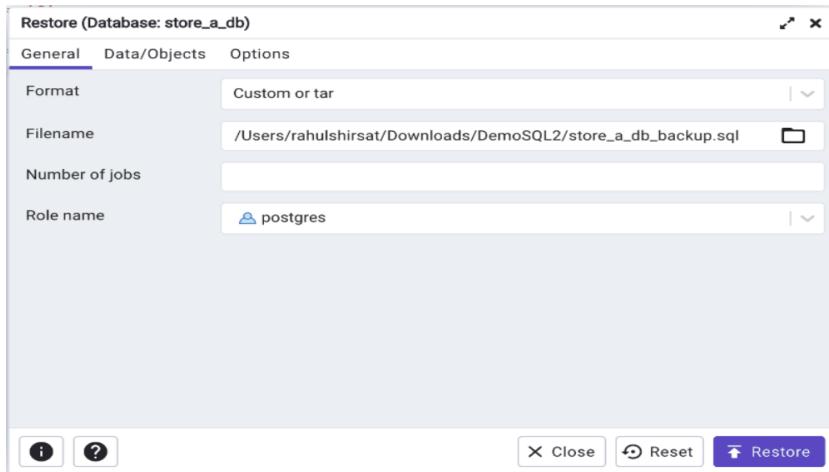
Restore는 AgensEM Backup 대화상자에서 가져온 사용자 지정, tar 또는 디렉터리 형식 백업을 사용하여 데이터베이스 또는 데이터베이스 오브젝트를 재생성하는 기능을 제공 합니다. Backup 대화상자는 pg_dump 클라이언트 유ти리티의 옵션을 호출합니다. Restore 대화 상자는 pg_restore 클라이언트 유ти리티의 옵션을 호출합니다.

Query Tool를 사용하여 Backup 대화상자에서 만든 일반 텍스트 백업 중에 생성된 스크립트를 재생할 수 있습니다.

General 탭의 필드를 사용하여 복원 프로세스에 대한 일반 정보를 지정합니다.

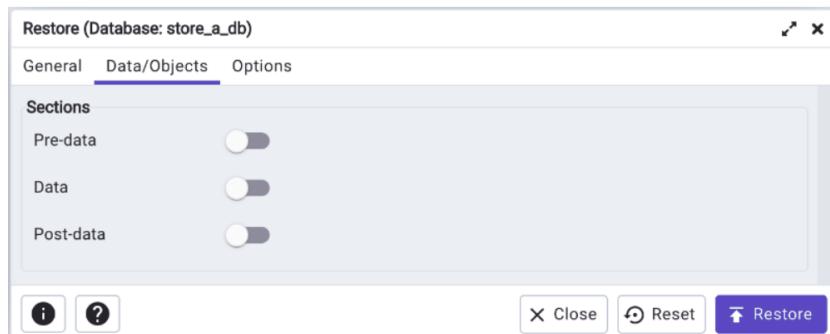
- Format 필드의 드롭다운 목록 상자를 사용하여 백업 파일의 형식을 선택합니다.
 - Custom 또는 tar를 선택하여 사용자 지정 아카이브 파일에서 복원하여 백업된 오브젝트의 복사본을 만들습니다.

- 압축된 디렉터리 형식 아카이브에서 복원할 Directory 를 선택합니다 .
- **Filename** 필드에 백업 파일의 전체 경로를 입력합니다. 선택적으로 오른쪽에 있는 브라우저 아이콘(줄임표)을 선택하여 디렉토리로 이동하고 아카이브가 포함된 파일을 선택합니다.
- **Number of jobs** 필드를 사용하여 pg_restore 가 복원을 처리하기 위해 여러(동시) 작업을 사용해야 하는지 여부를 지정합니다. 각 작업은 서버에 대한 별도의 연결을 사용합니다.
- **Role name** 옆에 있는 드롭다운 목록 상자를 사용하여 복원 프로세스 중에 서버 인증에 사용할 User를 지정합니다.



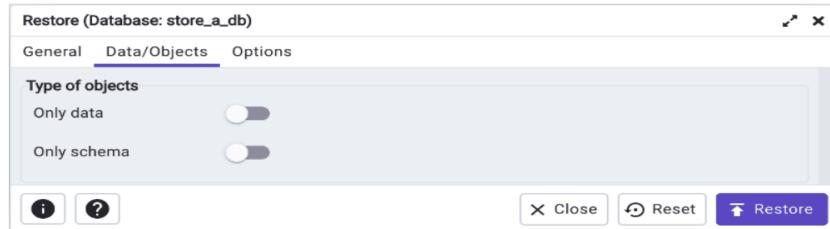
계속하려면 Data/Objects 탭을 클릭합니다 . Data/Objects 탭의 필드를 사용하여 데이터와 관련된 옵션 또는 pg_restore 옵션에 해당하는 AgensEM 오브젝트를 지정합니다.

- **Sections** 상자 의 스위치를 사용하여 복원할 콘텐츠를 지정합니다.
 - 데이터 또는 사후 데이터 항목 목록에 포함되지 않은 모든 데이터 정의 항목을 복원하려면 Pre-data 옆의 스위치를 True로 이동합니다.
 - Data 옆의 스위치를 True로 이동하여 실제 테이블 데이터, 큰 오브젝트 내용 및 시퀀스 값을 복원합니다.
 - 인덱스, 트리거, 규칙 및 제약 조건(검증된 확인 제약 조건 제외)의 정의를 복원하려면 Post-data 옆의 스위치를 True로 이동합니다.

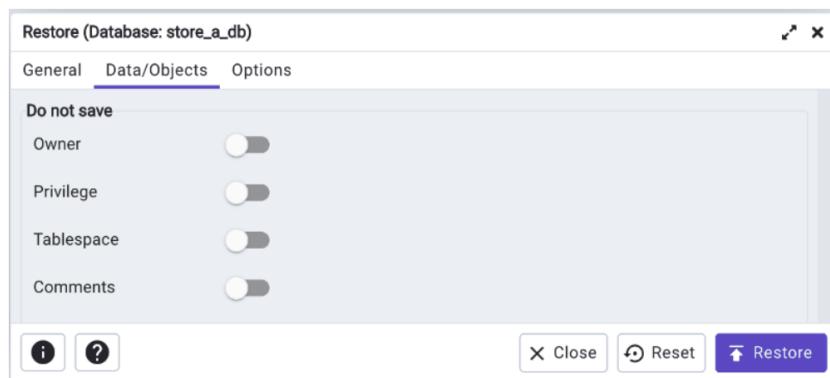


- **Type of Objects** 상자 의 스위치를 사용하여 복원할 오브젝트를 지정합니다.
 - Only data 복원 옆에 있는 스위치를 True로 이동하여 데이터 복원을 제한합니다.

- 복원을 Schema 수준 데이터베이스 오브젝트로 제한하려면 Only schema 옆의 스위치를 이동합니다.

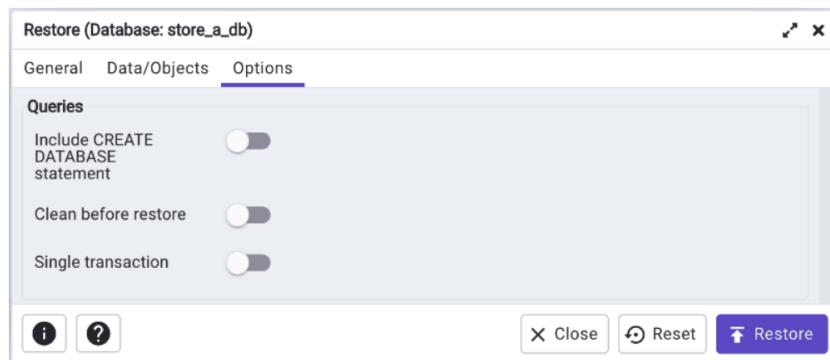


- Do not save 상자 의 스위치를 사용하여 복원하지 않을 오브젝트를 지정합니다.
 - Owner 옆의 스위치를 True로 이동하여 오브젝트 소유권을 설정하는 명령을 제외합니다.
 - 액세스 권한을 생성하는 명령을 제외하려면 Privilege 옆의 스위치를 True로 이동합니다.
 - 테이블스페이스를 제외하려면 Tablespace 옆의 스위치를 True로 이동합니다
 - 설명을 설정하는 명령을 제외하려면 Comments 옆에 있는 스위치를 True로 이동합니다.

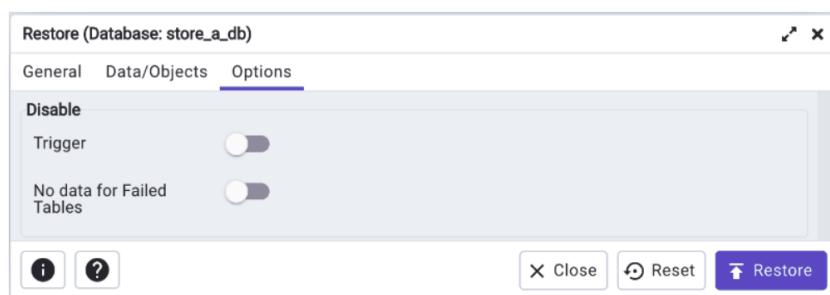


계속하려면 Options 탭을 클릭하십시오 . 이러한 추가 필드를 사용하여 복원 전 정리, 상세 메시지 또는 pg_restore 옵션에 해당하는 세션 권한 설정 사용과 같은 옵션을 지정합니다.

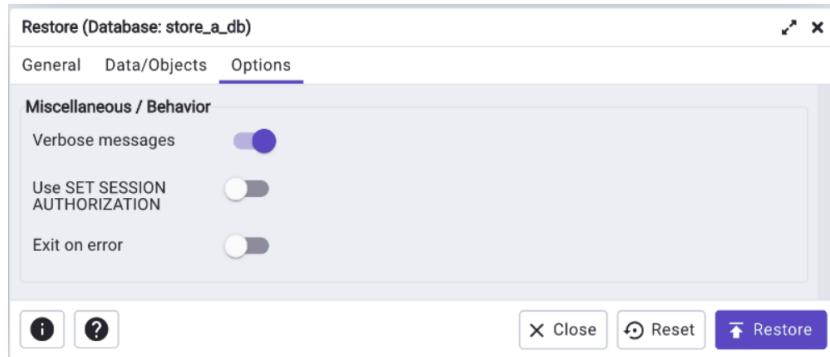
- Queries 상자 의 스위치를 사용하여 복원에 포함되어야 하는 문의 유형을 지정합니다.
 - 복원을 수행하기 전에 새 데이터베이스를 생성하는 명령을 포함하려면 Include CREATE DATABASE statement 옆의 스위치를 True로 이동합니다 .
 - 복원하기 전에 각 기존 데이터베이스 오브젝트(및 데이터)를 삭제하려면 Clean before restore 옆의 스위치를 True로 이동합니다 .
 - Single transaction 옆의 스위치를 True로 이동하여 복원을 단일 트랜잭션으로 실행합니다(즉, 내보낸 명령을 BEGIN/COMMIT에 래핑). 이렇게 하면 모든 명령이 성공적으로 완료되거나 변경 사항이 적용되지 않습니다. 이 옵션은 –exit-on-error 를 의미합니다 .



- Disable 상자 의 스위치를 사용하여 복원에서 제외해야 하는 문의 유형을 지정합니다.
 - 데이터가 로드되는 동안 대상 테이블에서 트리거를 비활성화하는 명령을 포함하려면 Trigger (데이터 전용 복원을 생성할 때 활성화됨) 옆의 스위치를 True로 이동합니다.
 - No data for Failed Tables 옆의 스위치를 True로 이동하여 트리거에 실패한 데이터를 무시합니다.



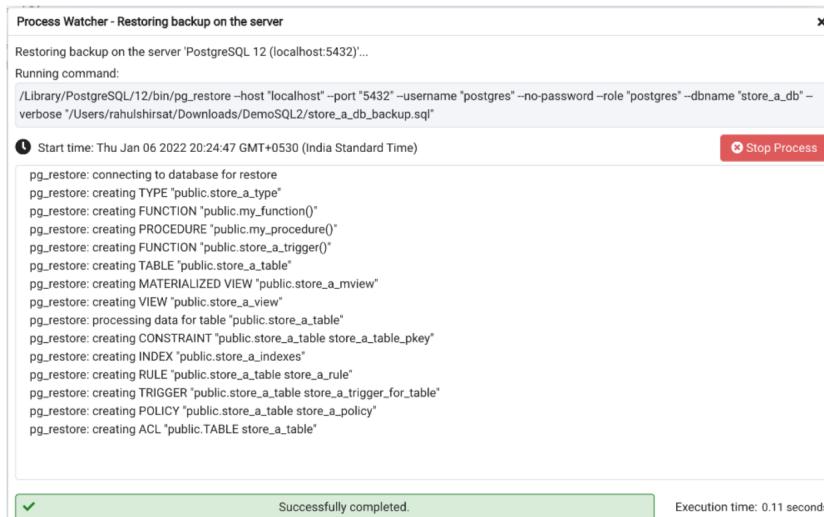
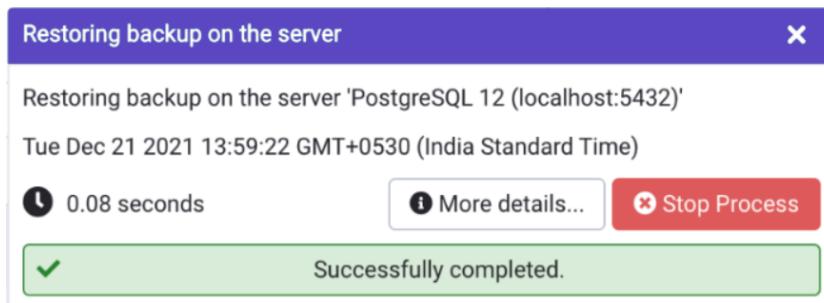
- Miscellaneous/Behavior 상자의 스위치를 사용하여 기타 복원 옵션을 지정합니다.
 - Verbose messages 옆에 있는 스위치를 False로 이동하여 pg_restore 에 자세한 메시지를 제외하도록 지시합니다.
 - Use SET SESSION AUTHORIZATION 옆의 스위치를 True로 이동하여 ALTER OWNER 명령 대신 SET SESSION AUTHORIZATION 명령을 사용하여 오브젝트 소유권을 결정하는 명령문을 포함합니다.
 - Exit on error 옆의 스위치를 True로 이동하여 pg_restore 에 SQL 명령 전송에 오류가 있는 경우 복구를 종료하도록 지시합니다. 기본값은 복원을 계속하고 복원이 끝날 때 오류 수를 표시하는 것입니다.



`pg_restore` 명령에 통합될 세부 정보를 지정한 경우 복원 버튼을 클릭하여 프로세스를 시작하거나 취소 버튼을 클릭하여 작업을 저장하지 않고 종료합니다. 복원이 성공했는지 확인하는 팝업이 나타납니다.

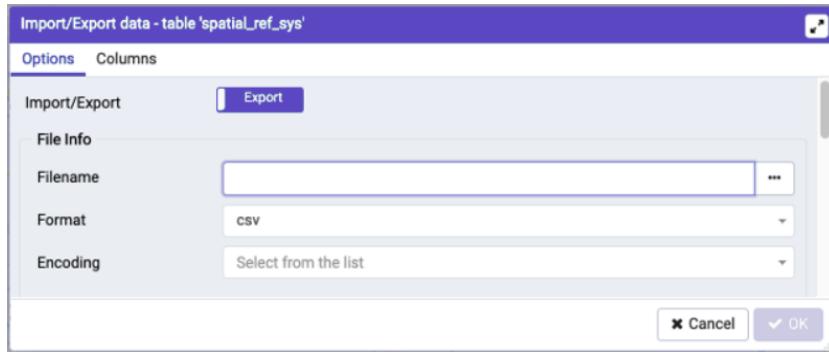
프로세스 중지 버튼을 사용하여 복원 프로세스를 중지합니다.

Process Watcher 실행 팝업에 대한 자세한 내용을 보려면 여기를 클릭 하십시오 . Process Watcher는 복원과 관련된 모든 활동을 기록하고 복원 명령에 문제가 발생할 경우 문제 해결을 위한 추가 정보를 제공합니다.



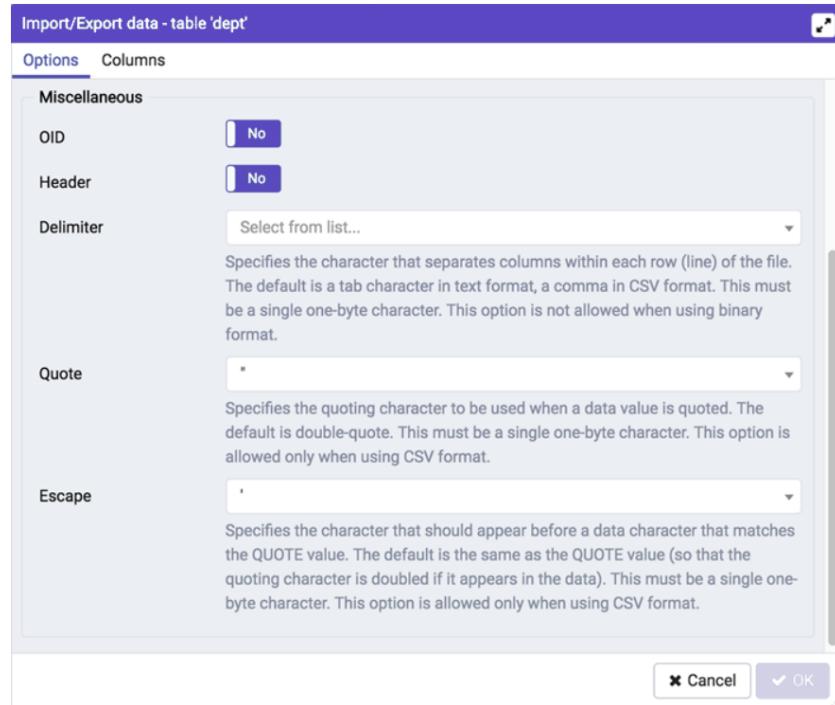
20.5.10 Import/Export Data

Import/Export 대화 상자를 사용하여 테이블에서 파일로 데이터를 복사하거나 파일에서 테이블로 데이터를 복사합니다. Import/Export 대화 상자는 옵션 및 Columns을 통해 Import/Export를 구성합니다.

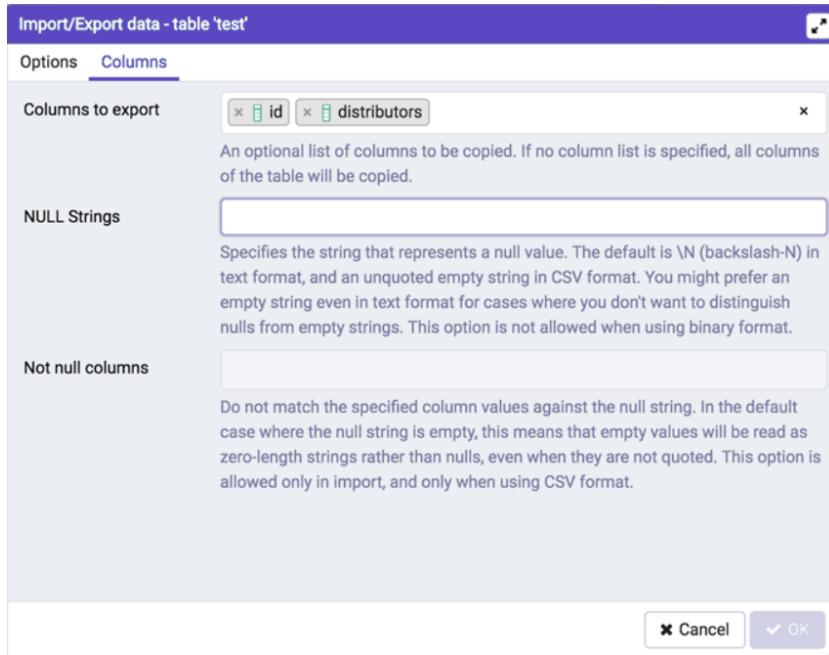


옵션 탭의 필드를 사용하여 가져오기 및 내보내기 기본 설정을 지정합니다.

- Import/Export 스위치를 Export로 이동하여 서버가 파일에서 테이블로 데이터를 가져오도록 지정합니다. 기본값은 ‘내보내기’입니다 .
- File Info 필드 상자의 필드를 사용하여 원본 또는 대상 파일에 대한 정보를 지정합니다.
 - Filename 필드에 소스 또는 대상 파일의 이름을 입력합니다 . 선택적으로 오른쪽에 있는 브라우저 아이콘(줄임표)을 선택하여 디렉토리를 탐색하고 파일을 선택합니다.
 - Format 필드의 드롭다운 목록 상자를 사용하여 파일 형식을 지정합니다.
 - .bin 파일용 바이너리
 - .csv 파일의 경우 csv
 - .txt 파일의 텍스트
 - Encoding 필드의 드롭다운 목록 상자를 사용하여 문자 인코딩 유형을 지정합니다.



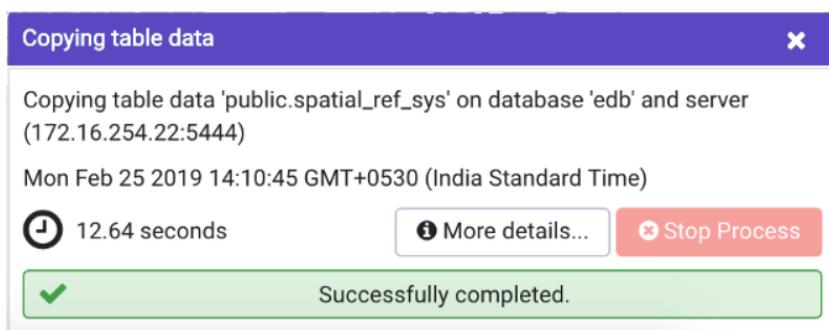
- Miscellaneous 필드 상자의 필드를 사용하여 추가 정보를 지정합니다.
 - OID 열을 포함하려면 OID 스위치를 True로 이동합니다. OID는 수정할 수 없는 시스템 할당 값입니다. 기본값은 '아니요'입니다.
 - 데이터 행과 함께 표 머리글을 포함하려면 Header 스위치를 True로 이동합니다. 테이블 머리글을 포함하면 파일의 첫 번째 행에 열 이름이 포함됩니다.
 - 데이터를 내보내는 경우 Delimiter 필드에서 대상 파일 내의 열을 구분할 구분 기호를 지정합니다. 구분 문자는 콜론, 세미콜론, 세로 막대 또는 탭일 수 있습니다.
 - Quote 필드에 사용되는 인용 문자를 지정하십시오. 인용은 문자열 열에만 적용할 수 있으며(예: 숫자 열은 인용되지 않음) 데이터 유형에 관계없이 모든 열에 적용할 수 있습니다. 인용에 사용되는 문자는 작은따옴표 또는 큰따옴표일 수 있습니다.
 - Escape 필드의 QUOTE 값과 일치하는 데이터 문자 앞에 표시되어야 하는 문자를 지정하십시오.



Columns의 필드를 사용하여 Import/Export 열을 선택합니다.

- Columns to export 필드 내부를 클릭하여 드롭다운 목록 상자에서 하나 이상의 열을 선택 취소합니다. 선택 항목을 삭제하려면 열 이름 왼쪽에 있는 X를 클릭합니다. 드롭다운 목록에 액세스하려면 필드 내부의 빈 공간을 클릭합니다.
- NULL Strings 필드를 사용하여 원본 또는 대상 파일 내에서 null 값을 나타내는 문자열을 지정합니다.
- 활성화된 경우 Not null columns 필드 내부를 클릭하여 NULL 값을 확인하지 않을 열을 하나 이상 선택합니다. 열을 삭제하려면 열 이름 왼쪽에 있는 X를 클릭합니다.

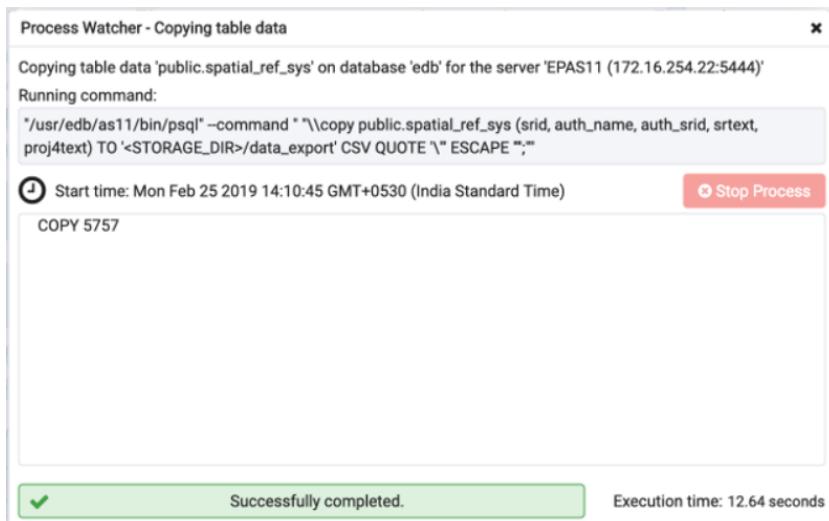
Import/Export 대화 상자를 완료한 후 확인 버튼을 클릭하여 가져오기 또는 내보내기를 수행합니다. AgensEM은 백그라운드 프로세스가 완료되면 알려줍니다.



Stop Process 버튼을 사용하여 Import/Export 프로세스를 중지합니다.

알림의 자세한 내용을 보려면 여기를 클릭 하십시오 링크를 사용하여 Process Watcher를 열고 가져오기 또는 내보내기를 수행한 명령 실행에 대한 자세한 정보를 검토하십시오.

Server Mode에서 AgensEM을 실행 중인 경우 Process Watcher 창에서 아이콘을 클릭하여 Storage Manager에서 파일 위치를 열 수 있습니다. Storage Manager를 사용하여 클라이언트 시스템에 백업 파일을 다운로드 할 수 있습니다.



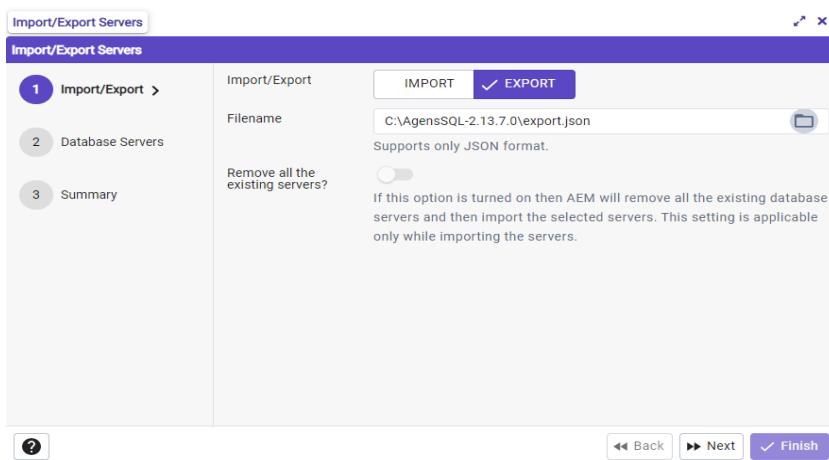
20.5.11 Import/Export Servers

서버 정의(및 해당 그룹)를 JSON 파일로 내보내고 동일하거나 다른 시스템으로 다시 가져와 AgensEM을 쉽게 사전 구성할 수 있습니다.

Tools에서 Import/Export Servers 옵션을 클릭합니다 .

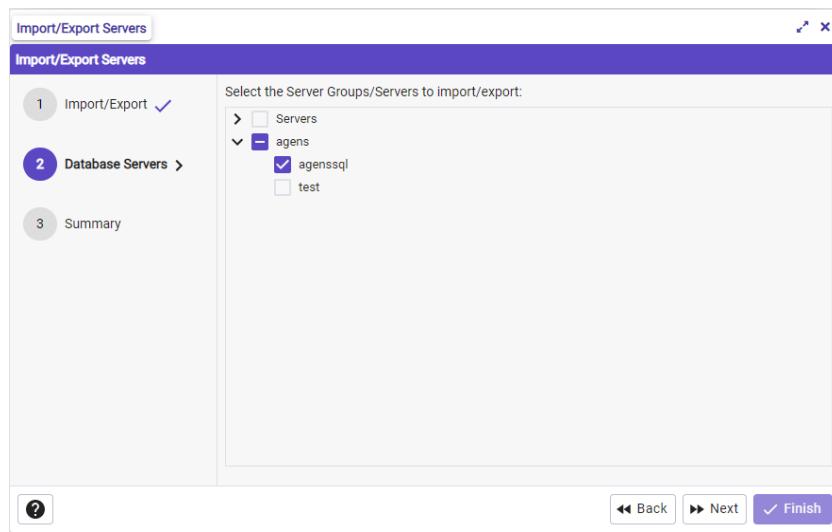
- Import/Export 필드를 사용하여 Import/Export 서버 그룹/서버를 선택합니다.
- Filename 필드를 사용하여 서버를 가져올 JSON 파일을 선택하거나 서버를 JSON 형식으로 내보낼 내보내기의 경우 새 파일을 만듭니다.
- Remove all the existing servers? 새로 선택한 서버를 가져오기 전에 모든 기존 서버를 제거할지 여부를 지정하는 필드입니다. 이 필드는 Export Server의 경우에만 적용됩니다.

계속하려면 다음 버튼을 클릭하고 마법사를 닫으려면 X 버튼을 클릭합니다.



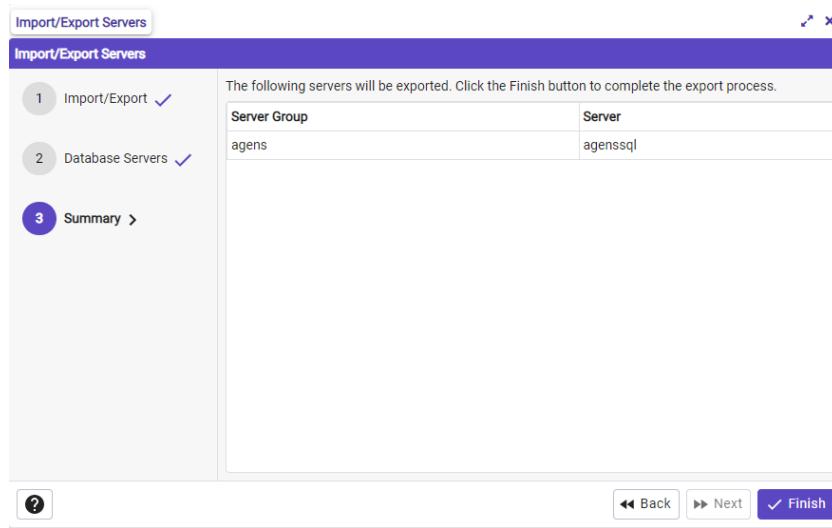
- Database Servers 페이지에서 Import/Export 작업을 실행할 서버 그룹/서버를 선택합니다.

계속하려면 다음 버튼을 클릭하고 마법사를 닫으려면 X 버튼을 클릭합니다.



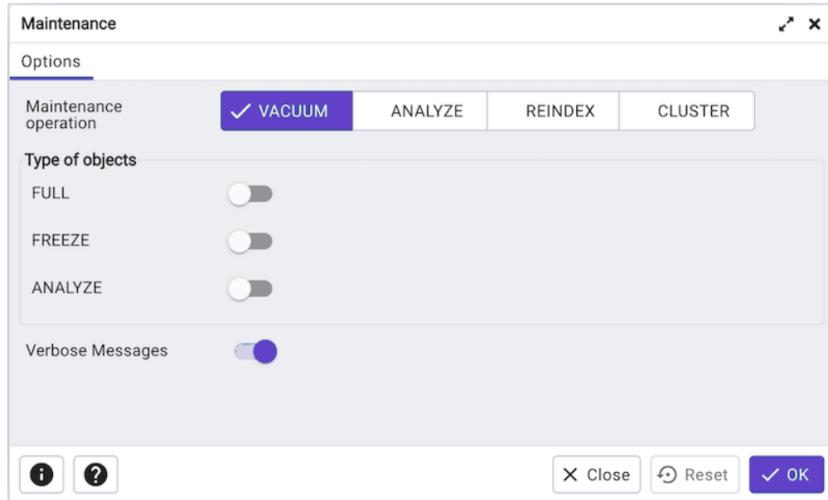
Summary 페이지에서 Import/Export할 서버의 요약을 확인하십시오.

마침 버튼을 클릭하여 마법사를 닫습니다.



20.5.12 Maintenance

Tools메뉴의 Maintenance 대화 상자를 사용하여 데이터베이스 또는 선택한 데이터베이스 객체를 VACUUM, ANALYZE, REINDEX 또는 CLUSTER하십시오.



이 유ти리티는 임시 유지 관리 목적에 유용하지만 정기적인 일정에 따라 자동 VACUUM 작업을 수행하는 것이 좋습니다.

유지 보수 작업 옆에 있는 버튼을 선택하여 유지 보수 유형을 지정합니다.

- VACUUM을 클릭하여 데드 튜플이 사용한 스토리지를 회수하기 위해 선택한 데이터베이스 또는 테이블을 스캔합니다.
 - 데드 스페이스 없이 테이블 파일의 완전히 새로운 버전을 작성하여 테이블을 압축하려면 FULL 스위치를 True로 이동하십시오. 기본값은 'False'입니다.
 - 더 이상 업데이트가 없을 때 테이블의 데이터를 고정하려면 FREEZE 스위치를 True로 이동합니다. 기본값은 'False'입니다.
 - 테이블의 내용이 충분히 변경될 때마다 ANALYZE 명령을 실행하려면 ANALYZE 스위치를 True로 이동하십시오. 기본값은 'False'입니다.
- ANALYZE를 클릭하여 쿼리 플래너에서 사용하는 저장된 통계를 업데이트합니다. 이렇게 하면 쿼리 최적화 프로그램이 최적의 성능을 위해 가장 빠른 쿼리 계획을 선택할 수 있습니다.
- 비정상적인 데이터 패턴의 삽입으로 인해 인덱스가 퇴화된 경우 인덱스를 다시 작성하려면 REINDEX를 클릭하십시오. 예를 들어 색인 값이 증가하는 행을 삽입하고 낮은 색인 값을 삭제하면 이런 일이 발생합니다.
- CLUSTER를 클릭하여 AgensSQL에 선택한 테이블을 클러스터링하도록 지시합니다.

프로세스 출력에서 상태 메시지를 제외하려면 Verbose Messages 스위치를 False로 이동합니다. 기본적으로 상태 메시지가 포함됩니다.

대화 상자를 완료하면 확인을 클릭하여 백그라운드 프로세스를 시작합니다. 유지 관리 작업을 수행하지 않고 대화 상자를 종료하려면 취소를 클릭하십시오.

AgensEM은 백그라운드 프로세스가 완료되면 알려줍니다.

20.5.13 Search Objects

이 대화 상자를 사용하면 데이터베이스에 있는 거의 모든 종류의 오브젝트를 검색할 수 있습니다. 데이터베이스 또는 하위 노드를 마우스 오른쪽 버튼으로 클릭하고 "Search Object"을 선택하여 액세스할 수 있습니다. 단축키(기본 ALT+SHIFT+S)를 눌러 액세스할 수도 있습니다.

최소 패턴 길이는 3자입니다. 수행된 검색은 대소문자를 구분하지 않으며 이름에 패턴이 포함된 모든 객체를 찾습니다. 현재 Object Name만 검색할 수 있습니다. e.g. abc, %ab%, ab%c, %%% 등

결과는 Object Name, Object Type 및 브라우저의 오브젝트 경로와 함께 표시됩니다. 결과 행을 두 번 클릭하여 브라우저에서 객체를 선택할 수 있습니다. 오브젝트가 회색으로 표시되면 기본 설정에서 해당 Object Type을 활성화하지 않았으므로 두 번 클릭할 수 없음을 의미합니다. 함수 및 프로시저 이름에 추가된 줄임표를 클릭하여 인수를 볼 수 있습니다.

Object Type 드롭다운에서 하나를 선택하여 특정 Object Type을 기준으로 필터링할 수 있습니다. 객체 유형 중 하나를 선택할 때 검색 버튼을 누르면 해당 유형만 데이터베이스에서 가져옵니다. 데이터베이스 서버가 지원하지 않거나 기본 설정에서 활성화되지 않은 경우 Object Type이 드롭다운에 표시되지 않습니다.

Search Objects - sampledb/bitnline@test		
%id		All types
Object name	Type	Browser path
id	Columns	Schemas/public/Tables/people/Columns/id
id	Columns	Schemas/public/Tables/people_range/Columns/id
id	Columns	Schemas/public/Tables/people_list/Columns/id
id	Columns	Schemas/public/Tables/people_hash/Columns/id
id	Columns	Schemas/public/Tables/people_rght/Columns/id
id	Columns	Schemas/public/Tables/mock_data/Columns/id

6 matches found.

? ✖ Close

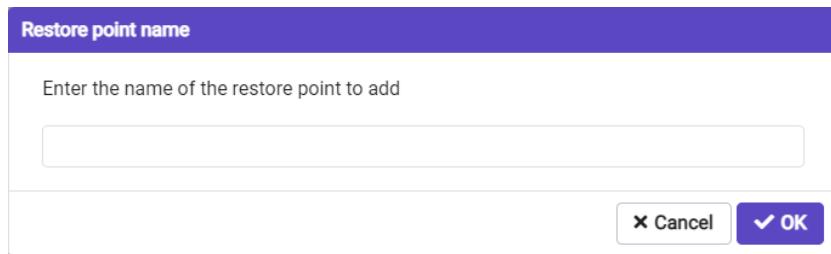
20.5.14 Add Named Restore Point

Add Named Restore Point 대화 상자를 사용하여 복구 파일에서 사용할 서버 상태의 명명된 스냅샷을 만듭니다. 명명된 복원 지점을 만들려면 서버의 `postgresql.conf` 파일에서 `replica`, `logical` 또는 `minimal` 중의 하나로 `wal_level` 값을 지정해야 합니다. 복원 지점을 만들려면 데이터베이스 수퍼유저여야 합니다.

Restore point name 대화 상자

Restore point name 대화 상자가 시작되면 `Enter the name of the restore point to add` 필드를 사용하여 복원 지점을 설명하는 이름을 제공합니다.

OK 버튼을 클릭하여 복원 지점을 저장합니다. 작업을 저장하지 않고 종료하려면 Cancel 버튼을 클릭하십시오.



20.5.15 Pause Replay of WAL

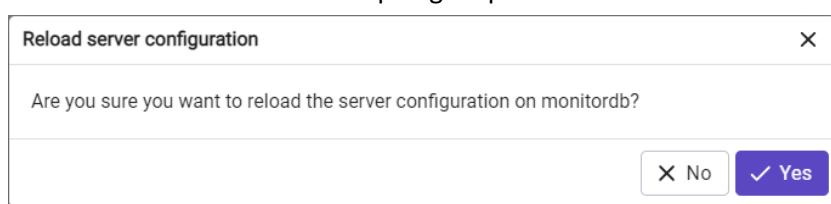
Standby DB에서 WAL log 적용을 일시 중지할 때 사용하며 내부적으로 `pg_wal_replay_pause()` 함수를 실행합니다.

20.5.16 Resume Replay of WAL

일시 중지해 놓은 WAL log 적용을 재개할 때 사용하며 내부적으로 `pg_wal_replay_resume()` 함수를 실행합니다.

20.5.17 Reload Configuration

Instance 를 재시작 하지 않고 `postgresql.conf` 파일의 변경사항을 적용할 때 사용합니다.

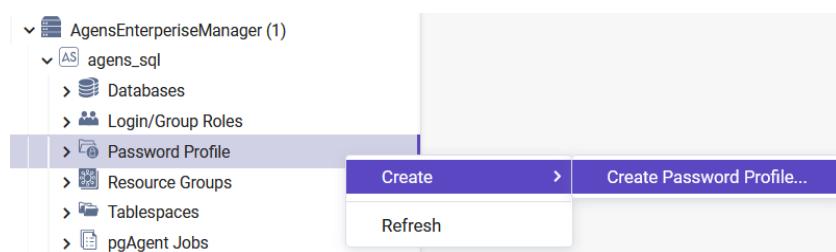


20.6 Password Profile

Password Profile는 사용자 또는 Role에 기본 설정을 할당할 수 있는 기능입니다. User/Role과 마찬가지로 클러스터 수준 object이며 공유 object입니다.

20.6.1 Password Profile 생성

Password Profile는 트리 메뉴를 통해 대화 상자 창을 시작할 수 있습니다. 서버의 하위 메뉴에서 Password Profile 폴더 모양의 아이콘을 볼 수 있습니다.



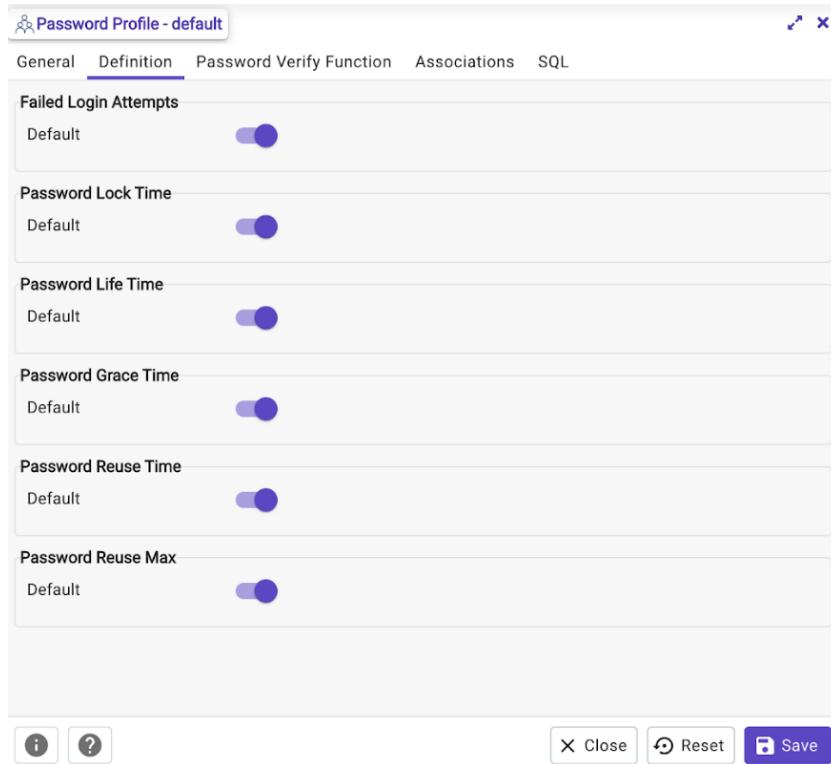
Password Profile를 생성할 때 마우스 오른쪽 버튼을 클릭하여 Create -> Create Password Profile 하위 메뉴를 통해 대화창을 실행할 수 있습니다.

20.6.2 General

가장 먼저 실행되는 메뉴는 아래의 기본 메뉴입니다. 기본적으로 제공되는 Password Profile의 이름을 255자 이내로 설정할 수 있습니다. 이름 외에 다른 설정 값이 없으면 나머진 모든 값은 기본값으로 저장됩니다.

A screenshot of the 'Create - Password Profile' dialog box. The title bar says 'Create - Password Profile'. Below it is a toolbar with icons for info and help. The main area has tabs: 'General' (selected), 'Definition', 'Password Verify Function', 'Associations', and 'SQL'. Under 'General', there is a 'Name' field containing 'test_profile'. At the bottom are buttons for 'Close', 'Reset', and 'Save'.

20.6.3 Definition



1) FAILED LOGIN ATTEMPTS

로그인 시도 실패 허용 횟수를 설정할 수 있습니다. FAILED_LOGIN_ATTEMPTS에 대해 다음과 같은 값을 사용할 수 있습니다.

- DEFAULT - DEFAULT 프로필에 지정된 FAILED_LOGIN_ATTEMPTS 값입니다.
- UNLIMITED - 허용되는 로그인 실패 횟수를 무제한으로 설정합니다.
- ATTEMPT FAILED LIMIT - 0 이상의 정수 값을 설정합니다.

2) PASSWORD LOCK TIME

서버가 FAILED LOGIN ATTEMPTS로 인해 잠긴 계정의 잠금을 해제하기 전에 경과해야 하는 시간을 지정합니다. PASSWORD LOCK TIME에 대해 다음과 같은 값을 사용할 수 있습니다.

- DEFAULT - DEFAULT 프로필에 지정된 PASSWORD LOCK TIME의 값입니다.
- UNLIMITED - 데이터베이스 수퍼유저가 수동으로 잠금을 해제할 때까지 계정이 잠겨 있습니다.
- DAY - 0 이상의 정수 값을 설정합니다.
- HOUR - 0 이상의 정수 값을 설정합니다.

- MINUTE - 0 이상의 정수 값을 설정합니다.

3) PASSWORD LIFE TIME

사용자에게 새 암호를 입력하라는 메시지가 표시되기 전에 암호를 사용할 수 있는 기간을 나타내는 값입니다. PASSWORD GRACE TIME을 지정하지 않으면 PASSWORD LIFE TIME으로 설정된 기간 이전의 데이터베이스 역할에 대한 연결이 거부됩니다. PASSWORD GRACE TIME이 지정된 경우 지정된 기간 동안 추가로 설정할 수 있으며, 새 암호가 제공될 때까지 사용자는 명령을 실행할 수 없습니다.

- DEFAULT - DEFAULT 프로필에 지정된 PASSWORD LIFE TIME의 값입니다.
- UNLIMITED - 암호 만료 날짜가 없습니다.
- DAY - 0 이상의 정수 값을 설정합니다.
- HOUR - 0 이상의 정수 값을 설정합니다.
- MINUTE - 0 이상의 정수 값을 설정합니다.

4) PASSWORD GRACE TIME

암호가 만료된 후 사용자가 암호를 변경해야 할 때까지의 유예 기간을 지정합니다. PASSWORD GRACE TIME이 지정된 경우 사용자는 연결할 수 있지만 만료된 암호를 업데이트할 때까지 명령을 실행할 수 없습니다.

- DEFAULT - DEFAULT 프로필에 지정된 PASSWORD GRACE TIME 값입니다.
- UNLIMITED - 유예 기간은 무한합니다.
- DAY - 0 이상의 정수 값을 설정합니다.
- HOUR - 0 이상의 정수 값을 설정합니다.
- MINUTE - 0 이상의 정수 값을 설정합니다.

5) PASSWORD REUSE TIME

PASSWORD REUSE MAX와 함께 사용하도록 설계된 파라미터 값입니다. 이전에 사용한 것과 동일한 비밀번호를 설정할 수 있을 때까지 남은 일수를 의미합니다.

- DEFAULT - DEFAULT 프로필에 지정된 PASSWORD REUSE TIME 값입니다.
- UNLIMITED - 제한 없이 암호를 재사용할 수 있습니다.
- DAY - 0 이상의 정수 값을 설정합니다.
- HOUR - 0 이상의 정수 값을 설정합니다.
- MINUTE - 0 이상의 정수 값을 설정합니다.

6) PASSWORD REUSE MAX

비밀번호 변경에 허용되는 최대값을 의미합니다. PASSWORD REUSE TIME 및 PASSWORD REUSE MAX 매개 변수는 함께 사용됩니다.

- DEFAULT - DEFAULT 프로필에 지정된 PASSWORD REUSE MAX 값입니다.
- UNLIMITED - 제한 없이 암호를 재사용할 수 있습니다.
- REUSE MAX - 0 이상의 정수 값을 설정합니다.

20.6.4 Password Verify Function

Create - Password Profile

General Definition Password Verify Function Associations SQL

Mode LOAD NEW EDIT

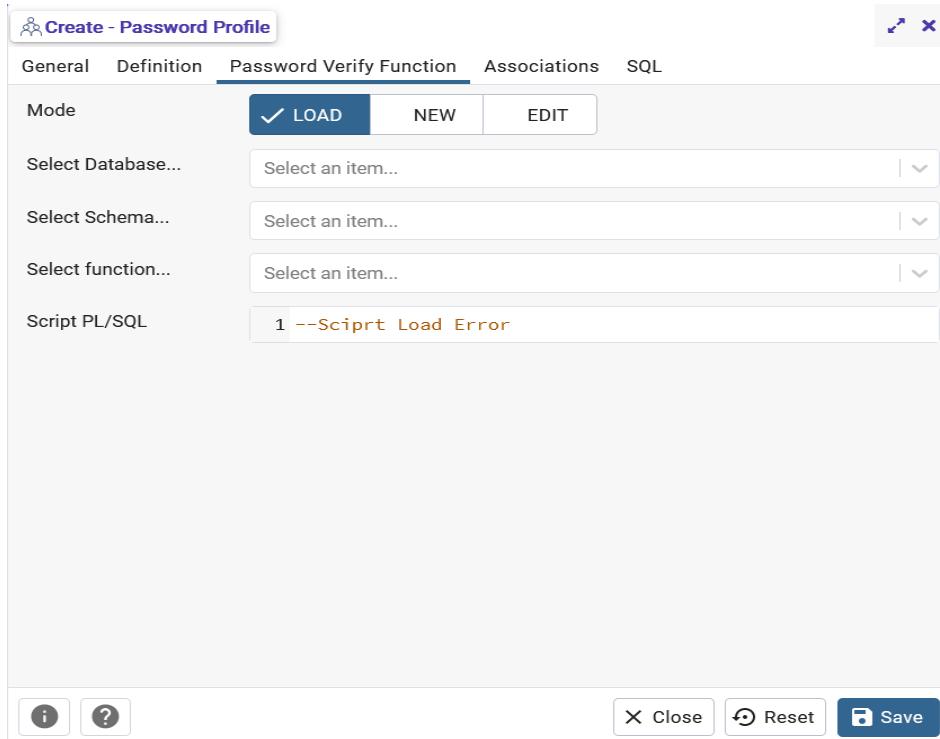
Select Database... Select an item... | ▾

Select Schema... Select an item... | ▾

Select function... Select an item... | ▾

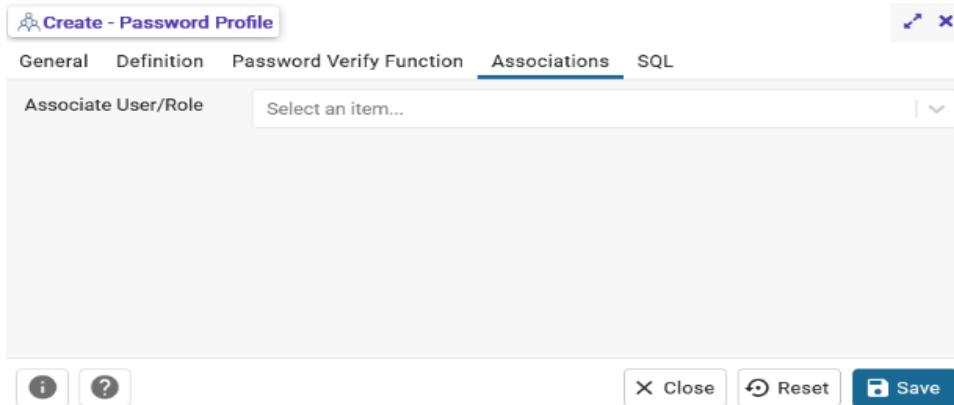
Script PL/SQL
1 --Sciprt Load Error

Close Reset Save



Password Profile의 경우 프로필에서 사용할 수 있는 암호 유효성 검사 기능을 지정할 수 있습니다. 암호 규칙은 함수를 통해 지정할 수 있으며 프로파일이 적용되는 Role 함수의 조건을 만족하는 암호를 갖도록 강제할 수 있습니다.

20.6.5 Associations



Password Profile에 적용되는 User 또는 Role을 적용할 수 있습니다. 동일한 프로필을 다양한 Role/User에게 적용할 수 있습니다.

20.6.6 SQL

사용자가 선택한 설정에 대해 실행할 SQL 계획을 볼 수 있습니다.

```
1 CREATE PROFILE
2   test2
3     LIMIT
4
5   FAILED_LOGIN_ATTEMPTS 10
6   PASSWORD_LOCK_TIME 1
7   PASSWORD_LIFE_TIME 1
8   PASSWORD_GRACE_TIME 1
9   PASSWORD_REUSE_TIME 1
10  PASSWORD_REUSE_MAX 10
11 ;
12 -- associating roles with profile.
13 ALTER role "test_role" PROFILE "test2" ;
14
15
16 -- assigning verification function to profile.
17 ALTER PROFILE test2 LIMIT PASSWORD_VERIFY_FUNCTION verify_password;
```

The screenshot shows the 'SQL' tab selected in the 'Create - Password Profile' interface. A large text area contains a sample SQL script for creating a password profile and associating it with a role. The script includes various clauses like CREATE PROFILE, ALTER role, and ALTER PROFILE. At the bottom are three buttons: 'Close', 'Reset', and a blue 'Save' button.

본 프로필은 예시로 설정된 프로필 값입니다. 사용자가 설정한 값은 실제 SQL 실행 시 적용되는 문법으로 변환되어 예상되는 SQL 구문이 텍스트 영역에 표시됩니다.

- **Save**- 프로필을 클릭하여 생성합니다.
- **Close**- 프로필 생성을 취소하고 대화 상자를 종료합니다.

- *Reset* - 대화 상자 내부에 설정된 값을 초기화합니다.

20.7 Data Redaction

Data Redaction 정책은 정보를 안전하게 관리하는 데 사용됩니다.

특정 사용자에게 표시되는 데이터를 동적으로 변경하여 민감한 데이터 노출을 제한합니다.

전화번호 등 민감한 개인정보(예: (82-10-XXX-XXX))의 경우 예시와 같이 일부 데이터만 사용자에게 표시되어 사용자에게 힌트를 제공하지만 전체 데이터는 공개하지 않습니다.

보안 정보:

- 아래의 사용자는 원본 데이터가 표시 됩니다.
- 슈퍼유저, 테이블 소유자, 테이블에 대한 전체 액세스 권한이 있는 사용자

20.7.1 Redaction Policies 확인

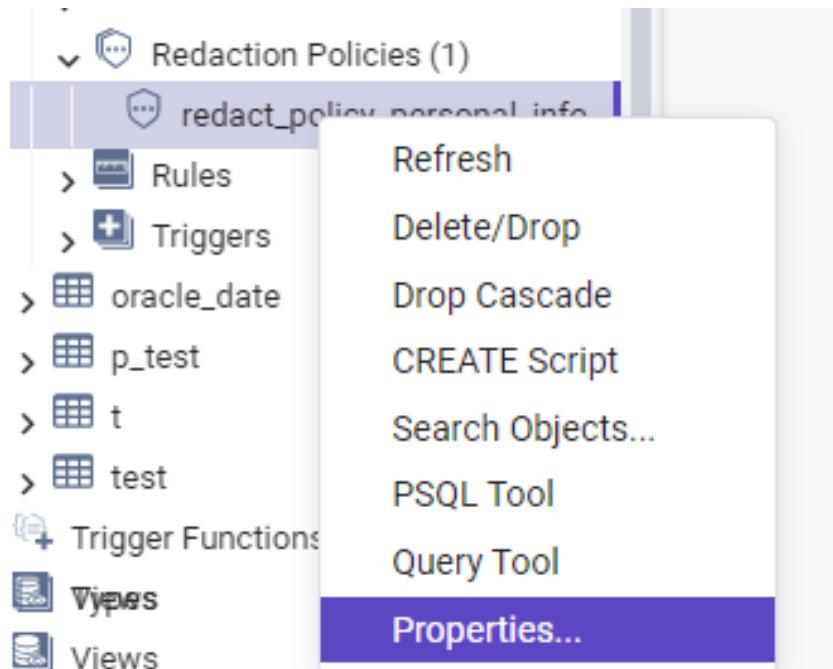
The screenshot shows a tree-based navigation interface for a database table named 'employees'. The tree structure is as follows:

- Tables (5)
 - employees
 - Columns
 - Compound Triggers
 - Constraints
 - Indexes
 - Partitions
 - RLS Policies
 - Redaction Policies (1)

A specific item, 'redact_policy_personal_info', under the 'Redaction Policies' section of the 'employees' table, is highlighted with a light blue background.

- Redaction Policy를 할당하는 경우 database -> Schemas -> Table -> Redaction Policy 트리 메뉴에서 해당 정책을 확인할 수 있습니다.

Redaction Policy 정보 조회:



- 트리 메뉴에서 Redaction Policy 오브젝트를 마우스 오른쪽 버튼으로 클릭하고 Properties를 클릭합니다.
- Redaction Policy 객체의 설정 및 상태 정보를 확인할 수 있습니다.
- ENABLE 상태

redact_policy_personal_info

General SQL

Name	redact_policy_personal_info
Redaction Enable	<input checked="" type="checkbox"/>
Redaction policy expression	{OPEXPR :opno 643 :opfuncid 659 :opresulttype 16 :o }
Redaction relation table	employees

i ? Close Reset Save

Redaction Enable

- DISABLE 상태

```
ALTER REDACTION POLICY redact_policy_personal_info ON employees DISABLE;
```

redact_policy_personal_info

General SQL

Name	redact_policy_personal_info
Redaction Enable	<input type="checkbox"/>
Redaction policy expression	{OPEXPR :opno 643 :opfuncid 659 :opresulttype 16 :o }
Redaction relation table	employees

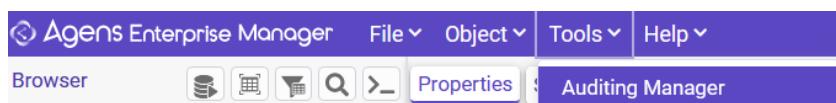
i ? Close Reset Save

20.8 Auditing Manager

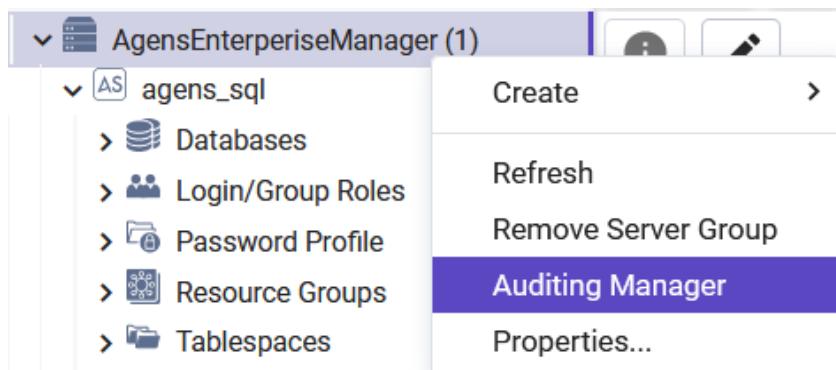
AgensSQL은 감사 로깅 기능을 사용하여 데이터베이스 활동을 추적하고 분석할 수 있는 데이터베이스, 보안 관리자 및 감사 연산자를 제공합니다. AgensSQL Audit Logging은 모든 관련 정보가 포함된 감사 로그 파일을 생성합니다. 감사 로그는 다음과 같은 정보를 기록하도록 구성할 수 있습니다. AEM을 이용한 Audit 설정은 `postgresql.auto.conf`에 설정되어 동적으로 동작하며 `postgresql.conf`의 설정보다 우선 적용됩니다.

20.8.1 Auditing Manager 실행

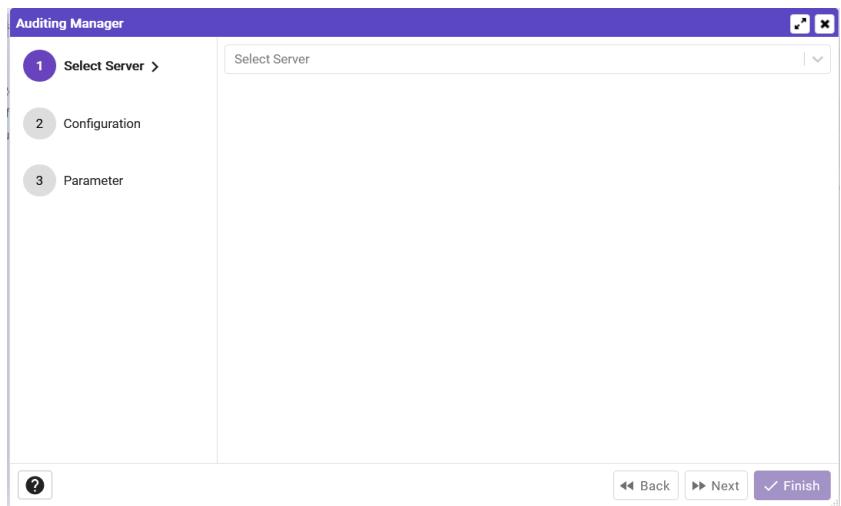
상단 Tools의 Auditing Manager 대화 상자를 열 수 있습니다.



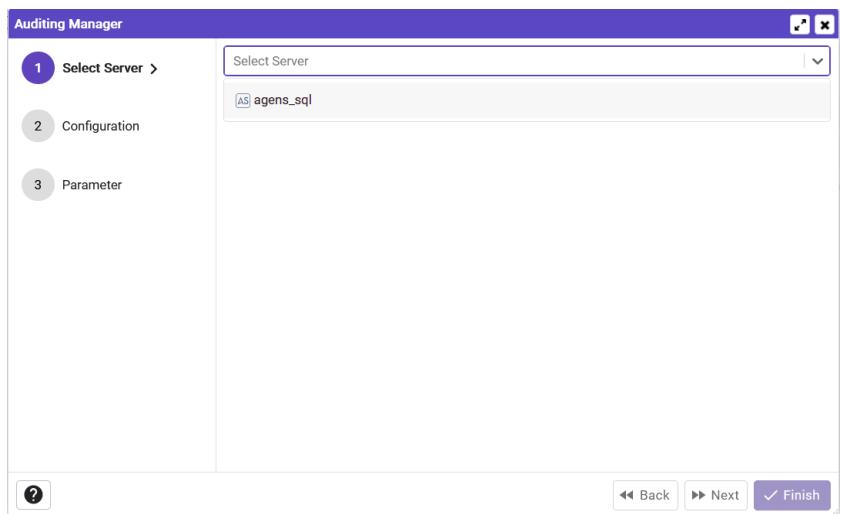
또는 트리 섹션의 상황에 맞는 메뉴에서 감사 관리자를 열 수 있습니다.



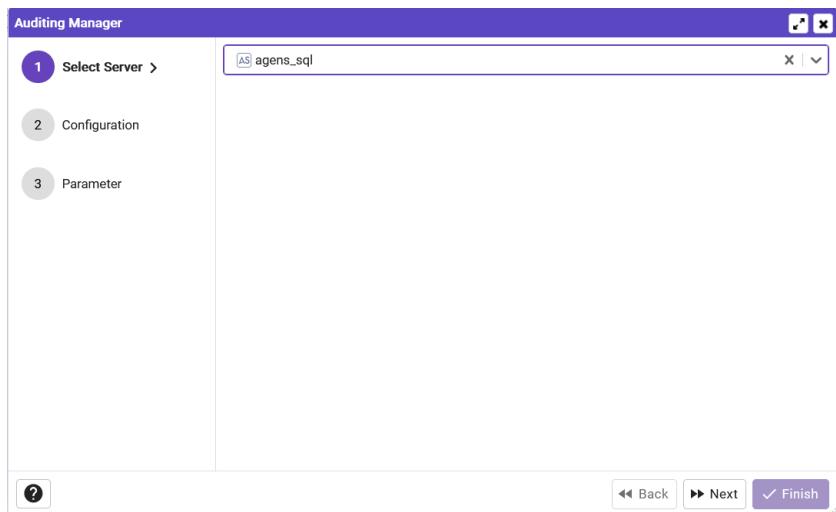
감사 관리자는 대화 상자를 통해 서버 구성을 변경하는 도구이며 보시는 것처럼 첫 번째 메뉴입니다.



감사 구성을 변경할 서버를 선택 합니다.

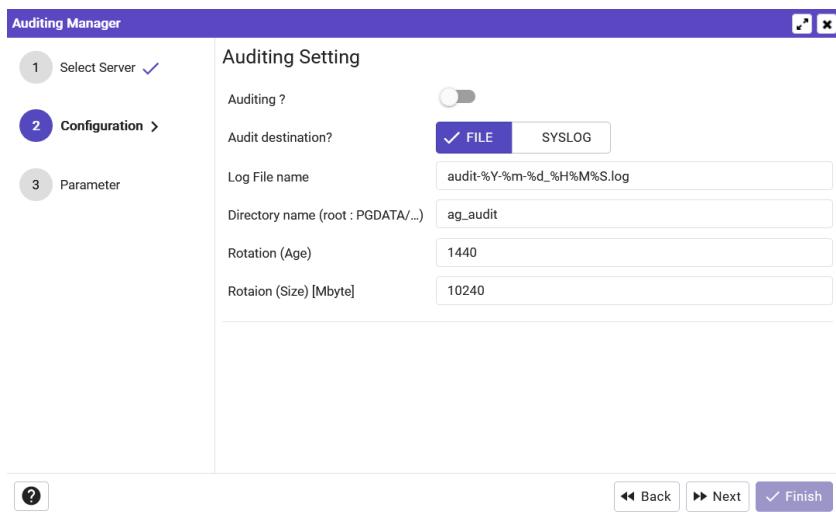


선택완료



20.8.2 Auditing 구성

다음은 구성 섹션입니다. 이제 서버별 구성 설정을 변경할 수 있습니다.



사용 가능한 옵션은 다음과 같습니다. 이 옵션은 에이전트를 참조하십시오. SQL의 감사 기능.

- **auditing?**

감사 기능 사용 여부를 선택합니다.

- **audit destination?**

감사 로그 정보를 ag_audit_directory 매개 변수에 의해 지정된 디렉터리에 기록할지 또는 syslog 프로세스에 의해 관리되는 디렉터리 및 파일에 기록할지 여부를 지정합니다. 파일로 설정하여 기본 설정인 ag_audit_directory에서 지정한 디렉토리를 사용합니다. syslog로 설정하여 /etc/syslog.conf 파일에 구성된 대로 syslog 프로세스 및 해당 위치를 사용합니다. syslog 설정이 에이전트에 유효합니다. Linux 호스트에서 실행 중이며 윈도우즈 시스템에서는 지원되지 않는 SQL Server입니다.

참고: 최근 Linux 버전에서는 syslog가 rsyslog로 대체되었으며 구성 파일은 /etc/rsyslog.conf에 있습니다.

- **Log file name**

감사 정보가 저장될 파일 이름을 지정합니다. 기본 파일 이름은 ag-audit-%Y%m%d_%H%M%S입니다. 이스케이프 시퀀스 %Y, %m 등은 시스템 날짜 및 시간에 따라 적절한 전류 값으로 대체됩니다.

- **Directory name**

로그 파일을 만들 디렉터리를 지정합니다. 디렉터리의 경로는 데이터 폴더에 대한 상대 경로 또는 절대 경로일 수 있습니다. 기본값은 PGDATA/ag_audit 디렉터리입니다.

- **Rotation(Age)**

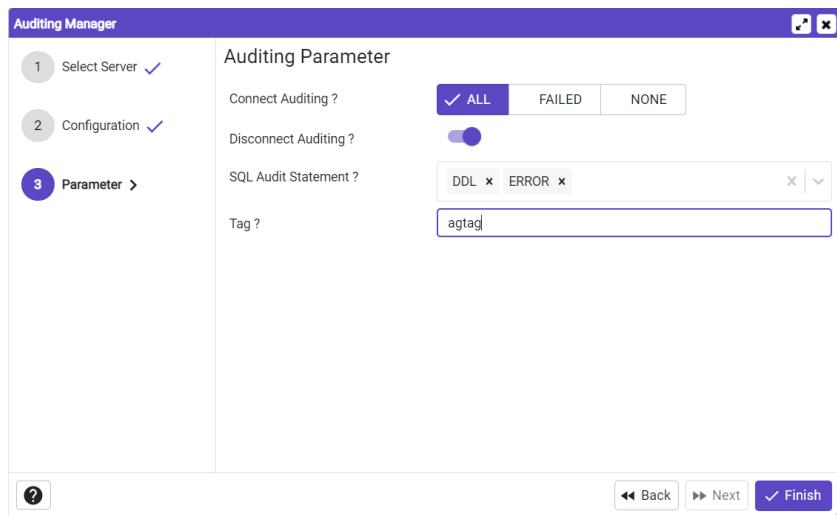
새 로그 파일을 만들어야 하는 순환 시간(분)을 지정합니다. 이 기능을 사용하지 않으려면 이 매개 변수를 기본값 0으로 설정하십시오.

- **Rotation(Size)**

로그를 순환할 파일 크기(MB)를 지정합니다. 기본값은 0MB입니다. 매개 변수가 주석 처리되거나 0으로 설정된 경우 파일 크기를 기준으로 파일을 순환하지 마십시오.

20.8.3 Auditing Parameter 설정

Parameter 섹션은 감사 파라미터를 표시합니다.



또한 일부 대상 서버의 감사 구성을 조작할 수 있습니다. 몇 가지 경우는 여기에 있습니다.

- **Connect Auditing?**

사용자의 데이터베이스 연결 시도에 대한 감사를 사용 가능으로 설정합니다. 모든 연결 시도에 대한 감사를 사용하지 않으려면 ag_audit_connect를 none으로 설정합니다. 실패한 모든 연결

시도를 감사하려면 값을 기본값인 실패로 설정합니다. 모든 연결 시도를 감사하려면 값을 '모두'로 설정하십시오.

- **Disconnect Auditing?**

연결된 사용자가 데이터베이스 연결 끊김을 감사할 수 있습니다. 연결 끊김 감사를 사용하려면 값을 ALL로 설정합니다. 사용하지 않으려면 값을 기본값인 '없음'으로 설정합니다.

- **SQL Audit Statement?**

세션 감사 로깅으로 기록할 명령문 클래스를 지정합니다. 가능한 값은 다음과 같습니다.

- READ: SELECT, COPY
- WRITE: INSERT, UPDATE, DELETE, TRUNCATE, COPY
- FUNCTION : 함수 호출 및 DO 블록
- ROLE : 역할 및 권한과 관련된 명령문 (예: GRANT, REVOKE, CREATE/ALTER/Drop ROLE)
- DDL : ROLE 클래스에 포함되지 않은 모든 DDL
- MISC : 기타 명령(예: DISCOVER, FETCH, CHECKPOINT, VACUUM, SET)
- MISC_SET : 기타 SET 명령(예: SET ROLE)
- ERROR: 모든 데이터베이스 오류
- ALL: 위의 내용을 모두 포함

쉼표로 구분된 목록을 사용하여 여러 클래스를 제공할 수 있습니다. 기본값은 DDL,ERROR입니다.

쉼표로 구분된 목록을 사용하여 여러 클래스를 제공할 수 있으며 클래스 앞에 - 기호를 붙여 클래스를 뺄 수 있습니다.(세션 감사 로깅 참조) 기본값은 DDL,ERROR입니다.

- **Tag?**

이 구성 매개 변수를 사용하여 각 항목에 대한 감사 로그 파일에 포함될 문자열 값을 추적 태그로 지정합니다.

20.9 Agens HA Manager

데이터베이스 서버는 함께 작동하여 주 서버가 실패할 경우 두 번째 서버가 빠르게 인계받도록하여 서비스 지연 시간을 최소화 할 수 있도록 합니다. AHM (Agens High Availability Manager)을 이용하여 HA 클러스터를 구성하고, AEM에서 클러스터의 상태를 모니터링하고, 클러스터 하위의 노드상태들을 관리 할 수 있습니다.

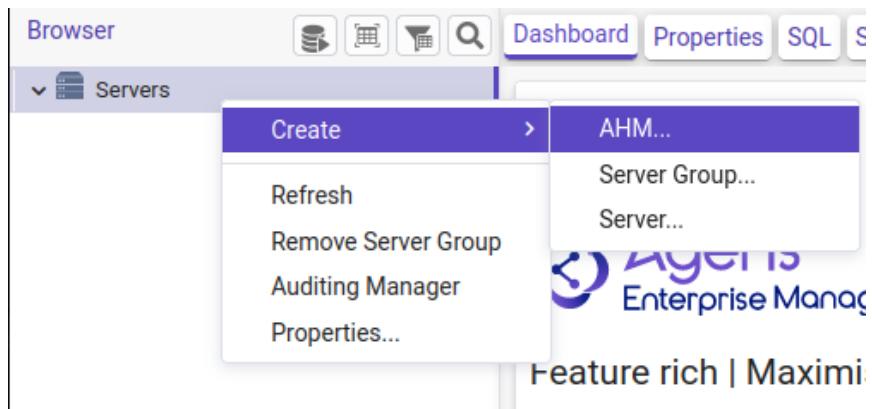
20.9.1 AHM Monitoring

1) AHM을 사용한 클러스터 구성

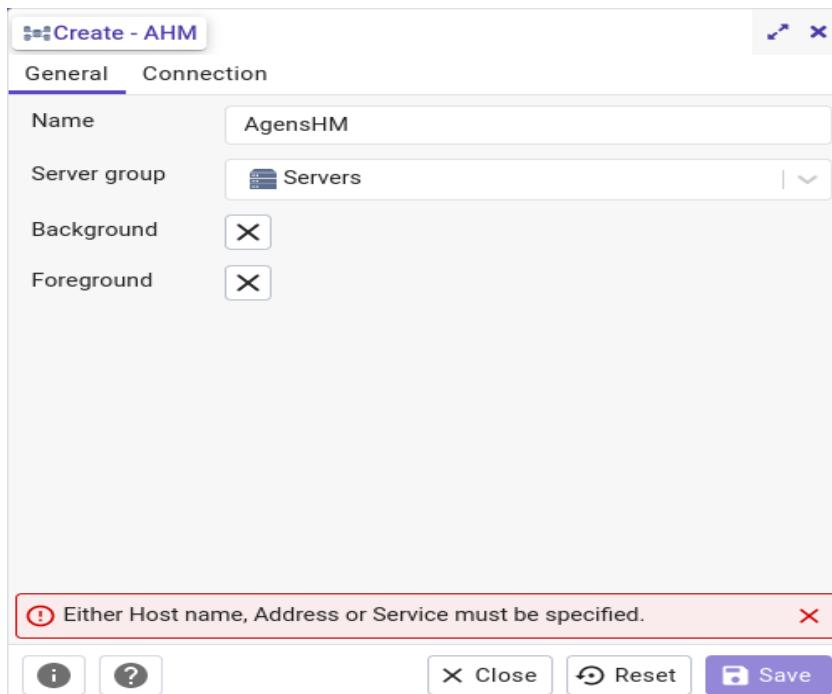
AHM Cluster 구성 가이드를 참조 하십시오.

2) AHM 클러스터 정보 등록

트리 메뉴에서 ServerGroup을 우클릭한 후 Create 메뉴에서 “AHM...” 을 선택

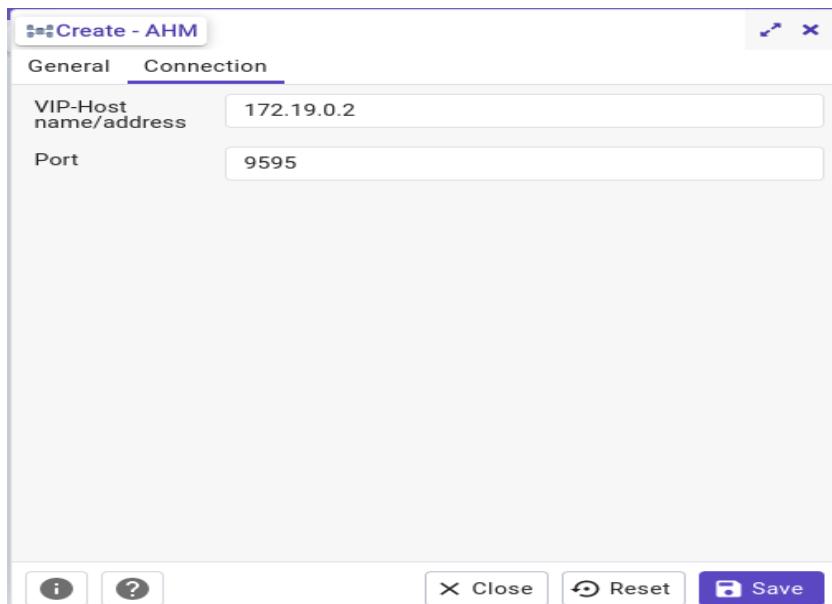


3) 등록할 클러스터의 기본 정보를 입력



- **Name** : 좌측 트리 메뉴에 추가될 이름을 255자 이내로 지정합니다.
- **Server Group** : 좌측 트리 메뉴의 상위 그룹인 서버그룹을 지정할 수 있습니다.
- **Background** - 표시 글자 배경 색상을 선택합니다.
- **Foreground** - 표시 글자 색상을 선택 합니다.

4) 등록할 클러스터의 연결 정보를 입력



- **name / address** : AHM 프로세스가 기동중인 VIP 주소입니다. (255자 이내)
- **Port** : AHM 프로세스가 기동중인 포트번호입니다.

5) DashBoard를 활용한 HA 클러스터 모니터링

등록한 클러스터로부터 스테이터스 정보를 받아 화면을 구성합니다.

The screenshot shows the 'HA cluster summary' and 'Node list of cluster' sections of the Dashboard. The 'HA cluster summary' section on the left contains four items: 'Current primary node' (AHM-NODE-1), 'Total nodes' (2), 'Remote Node Count' (1), and 'Virtual IP address' (192.168.0.10). The 'Node list of cluster' section on the right displays a table with two rows of data:

Name	Host	Port	Node State	Database State	Priority	Is witness
AHM-NODE-1	172.19.0.2	9500	LEADER	PRIMARY	1	False
AHM-NODE-2	172.19.0.3	9600	STANDBY	STANDBY	2	False

● HA Cluster Summary 메뉴

- **Reload** : Reload 버튼을 클릭할 시, Cluster로부터 정보를 다시 받아온 후 화면을 구성합니다.
- **Current Primary node** : 현재 primary 노드를 표시합니다.
- **Total nodes** : 클러스터에 연결된 전체 node 수를 표시합니다.
- **Remote Node Count** : 클러스터에 연결된 원격 node 수를 표시합니다.
- **Virtual IP address** : 현재 VIP를 표시합니다.

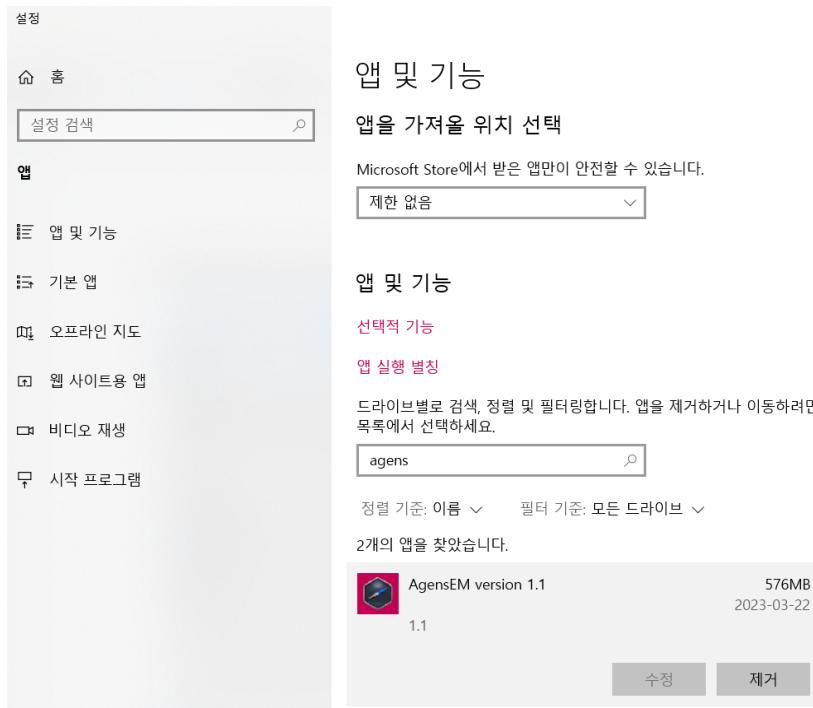
● Node list of Cluster 메뉴

- **Name** : AHM으로 기동중인 노드의 이름입니다.
- **Host** : 호스트 주소에 해당됩니다.
- **Port** : AHM이 기동중인 포트번호에 해당됩니다.
- **Node State** : 현재 노드의 상태를 표시하는 문자입니다.
- **Database State** : HA가 구성된 데이터베이스의 상태를 표시합니다.
- **Priority** : 클러스터 내부에서 Primary database로 승격되는 우선순위를 표시합니다.
- **Is witness** : 해당 노드가 witness노드로 작동중인지의 여부입니다.
 - witness 노드란 데이터베이스가 없는 상태로 HA 클러스터 구성에 포함되는 노드를 칭합니다.

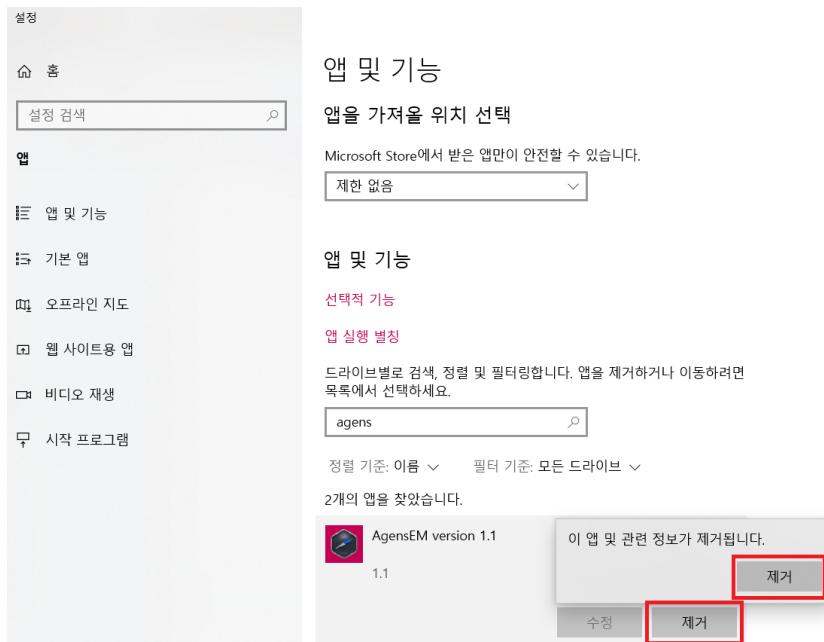
20.10 Agens Enterprise Manager 제거

20.10.1 프로그램 추가/제거를 이용한 제거

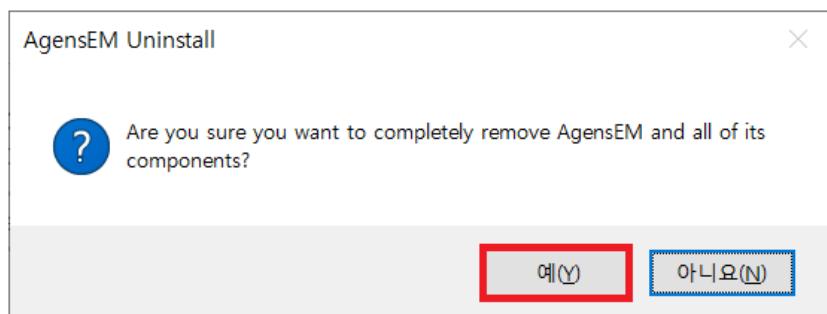
1) 프로그램 추가/제거 메뉴에서 AgensEM 찾기



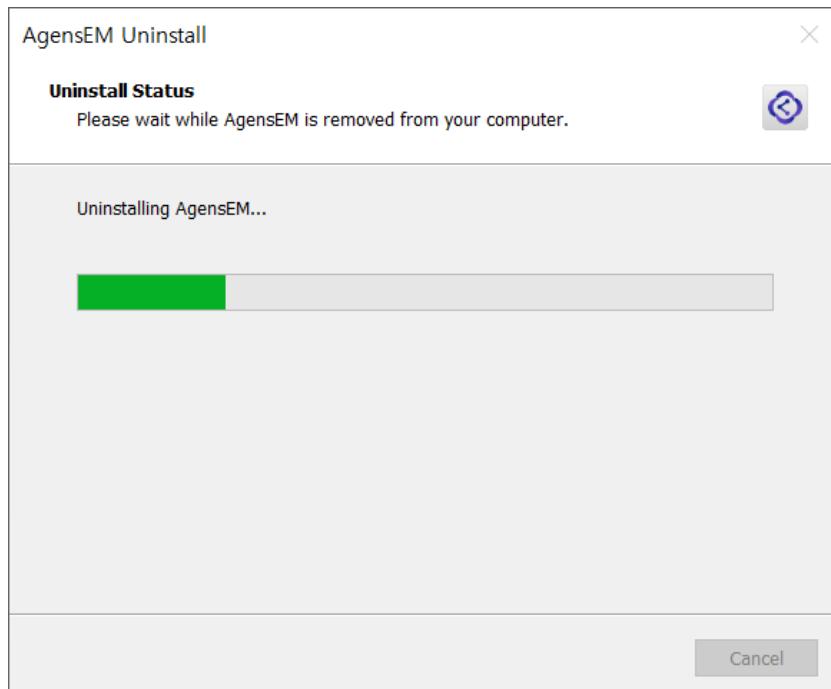
2) 제거 선택



3) Uninstall 진행



4) 제거 진행 중



5) 제거 완료

