



# PostGIS 시작하기

Getting Started with PostGIS (Korean Version)

김우미, 유병혁 | 2018. 06. 09.



사진 출처: 내셔널지오그래픽(National Geographic) 공식 홈페이지(<https://www.nationalgeographic.com/latest-stories/>)

## PostGIS 시작하기 소개

이 문서는 콜로라도대학교 덴버 캠퍼스 지리학부 FOSS4G Lab 의 리카르도 올리베이라(Ricardo Oliveira) 님이 작성하신 'Getting Started with PostGIS' 온라인 튜토리얼 시리즈를 한국어 사용자에게 맞게 수정한 것입니다. 공식 홈페이지: <https://clas.ucdenver.edu/foss4g/training/online-tutorials>

이 글은 아래 온라인 문서로도 이용하실 수 있습니다:

- [1] 윈도우에서 PostgreSQL과 PostGIS 설치하기 | <http://blog.daum.net/geoscience/1237>
- [2] PostgreSQL/PostGIS에서 DB 생성하고 공간데이터 추가하기 | <http://blog.daum.net/geoscience/1249>
- [3] QGIS에서 PostGIS 레이어 추가하기 | <http://blog.daum.net/geoscience/1250>
- [4] PostgreSQL/PostGIS에서 SQL 언어 학습하기 | <http://blog.daum.net/geoscience/1253>
- [5] PostgreSQL/PostGIS에서 투영법 변환하기 | <http://blog.daum.net/geoscience/1254>
- [6] PostGIS에서 공간쿼리(Spatial Query) 사용하기 | <http://blog.daum.net/geoscience/1255>
- [7] PostGIS에서 공간관계(Spatial Relationship) 분석하기 | <http://blog.daum.net/geoscience/1256>

## 문서 작성자(authors)



김우미 | GIS 분석 전문가 그룹인 (주)GIS United에서 GIS분석을 맡고 있다.  
더 나은 세상을 위한 GIS분석을 추구하며 사회, 자연, 및 교육에 관심을 두고 있다.  
[umikim@gisutd.com](mailto:umikim@gisutd.com)



유병혁 | 국립공원관리공단에서 파크레인저 겸 공간 분석가로 활동 중이다.  
과학적 보호지역 관리를 위한 각종 공간정보의 분석, 활용에 관한 연구를 지향한다.  
[bhyu@knps.or.kr](mailto:bhyu@knps.or.kr)

## [1] 윈도우에서 PostgreSQL 과 PostGIS 설치하기

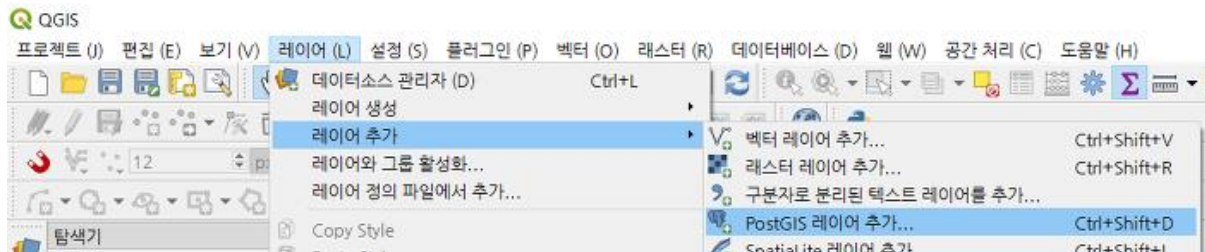
안녕하세요? 이번 글에서는 PostGIS 설치 방법을 학습해보도록 하겠습니다.

PostGIS 공식 홈페이지: <http://postgis.net/>

PostGIS 는 PostgreSQL 의 공간 데이터베이스 익스텐더(spatial database extender)입니다.

QGIS 는 PostGIS 레이어를 아래와 같이 추가할 수 있습니다. 먼저 PostgreSQL 을 설치하겠습니다.

- PostgreSQL: 오픈소스 기반의 DBMS
- PostGIS: 공간 DB 에 특화된 PostgreSQL 용 플러그인
- QGIS: PostGIS 레이어를 추가하여 공간데이터 가시화 지원



PostgreSQL(포스트그레스큐엘)은 오픈소스 객체관계형 데이터베이스 시스템(open source object-relational database system)입니다.

PostgreSQL 은 1986 년에 캘리포니아 대학교 버클리(University of California Berkley)의 'POSTGRES' 프로젝트 일부로 시작되었습니다.

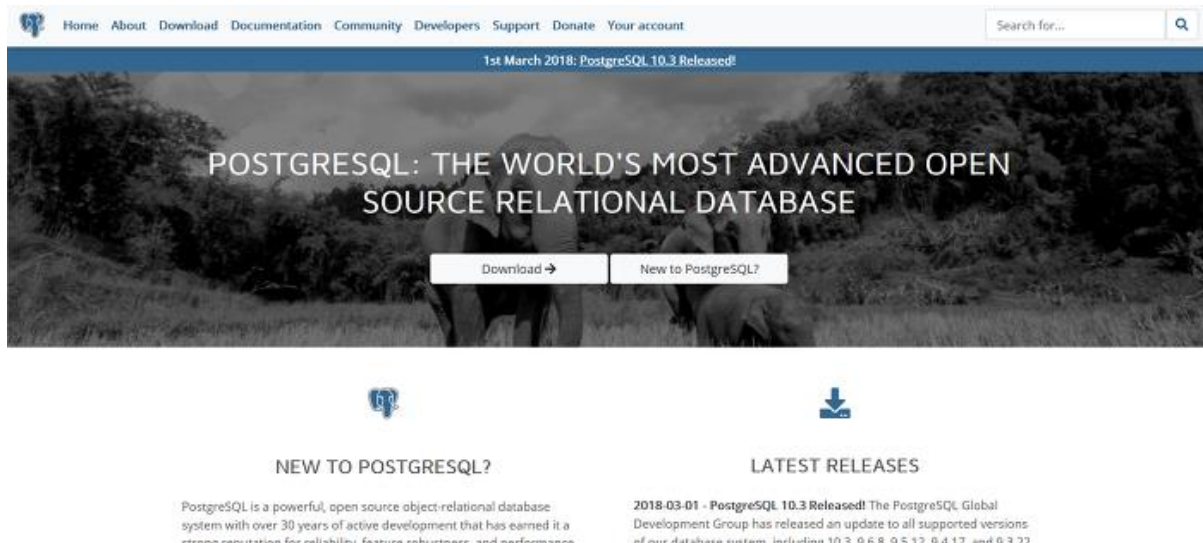
1994 년 POSTGRES 에 SQL 언어 인터프리터(interpreter)가 추가되면서 'Postgres95'라는 이름으로 웹에 공개되었습니다.

이후 Postgres95 라는 이름이 특정 년도에 한정된다는 것이 인정되면서, 1996 년부터 현재의 이름인 PostgreSQL 을 사용하게 되었습니다.

PostgreSQL 은 Postgres 프로젝트와 SQL 기능이 있는 최신 버전의 관계를 반영한 이름으로 많은 사람들은 여전히 PostgreSQL 을 Postgres(포스트그레스)라고 계속 인용하고 있습니다.

그럼, PostgreSQL 을 설치해볼까요?!

PostgreSQL 공식 홈페이지 | <https://www.postgresql.org/>



PostgreSQL 공식 홈페이지에서 Download 를 클릭하고 Windows 를 선택합니다.

## DOWNLOADS

### POSTGRESQL CORE DISTRIBUTION

The core of the PostgreSQL object-relational database management system is available in several source and binary formats.

#### BINARY PACKAGES

Pre-built binary packages are available for a number of different operating systems:

- BSD
  - [FreeBSD](#)
  - [OpenBSD](#)
- Linux
  - [Red Hat family Linux](#) (including [CentOS/Fedora/Scientific/Oracle](#) variants)
  - [Debian GNU/Linux](#) and derivatives
  - [Ubuntu Linux](#) and derivatives
  - [SuSE and OpenSuSE](#)
  - [Other Linux](#)
- [macOS](#)
- [Solaris](#)
- [Windows](#)

PostgreSQL 은 EnterpriseDB 에서 제공하는 인터랙티브 인스톨러와 BigSQL 에서 제공하는 그래픽얼 인스톨러가 있습니다.

EnterpriseDB 버전은 Visual C++ 컴파일러를, BigSQL 는 GCC(GNU Compiler Collection, 그누 컴파일러 모음)에 기반합니다.

PostGIS 인스톨러는 EnterpriseDB PostgreSQL 인스톨러와 함께 동작하도록 설계되었습니다. BigSQL 도 실험 세션의 바이너리를 이용할 수 있으나 추가항목 설치, 경로 등 차이가 있습니다.

여기서는 EnterpriseDB 인스톨러 중 PostgreSQL 10, 64 비트 윈도우 플랫폼을 선택하겠습니다. 본 글을 작성한 시점에서 PostgreSQL 의 최신 버전은 2018 년 3 월 1 일 배포된 PostgreSQL 10.3 입니다.

#### PLATFORM SUPPORT

The installers are tested by EnterpriseDB on the following platforms. They can generally be expected to run on other comparable versions:

PostgreSQL Version	64 Bit Windows Platforms	32 Bit Windows Platforms
10	2016, 2012 R2 & R1, 2008 R2, 7, 8, 10	2008 R1, 7, 8, 10
9.6	2012 R2 & R1, 2008 R2, 7, 8, 10	2008 R1, 7, 8, 10
9.5	2012 R2 & R1, 2008 R2	2008 R1
9.4	2012 R2, 2008 R2	2008 R1
9.3	2012, 2008 R2	2008 R1
9.2	2008 R2 & R1	2008 R1

EDB  
POSTGRES

DOWNLOADS BLOG ENGLISH

PRODUCTS CLOUD CUSTOMERS SERVICES AND SUPPORT TRAINING RESOURCES

SEARCH LOGIN MY ACCOUNT CONTACT

## PostgreSQL Download

Select your version

Select your operating system

DOWNLOAD NOW

Please note: Cookies should be enabled for the download process to function properly

Supported Platforms

아래와 같이 PostgreSQL 10.3 버전을 선택하였습니다. 이제 운영체제도 선택하고 파일을 다운로드 받습니다.

Select your version

PostgreSQL 10.3

PostgreSQL 9.6.8

PostgreSQL 9.5.12

PostgreSQL 9.4.17

PostgreSQL 9.3.22

PostgreSQL 9.2.24 (Not Supported)

PostgreSQL 9.1.24 (Not supported)



Select your operating system

Linux x86-32

Linux x86-64

Windows x86-32

Windows x86-64

Mac OS X

PostgreSQL 10.3

▼

Windows x86-64

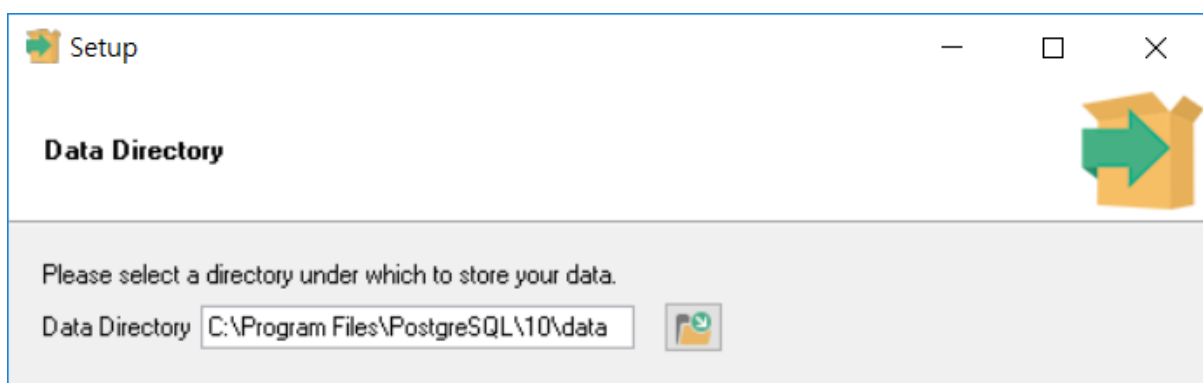
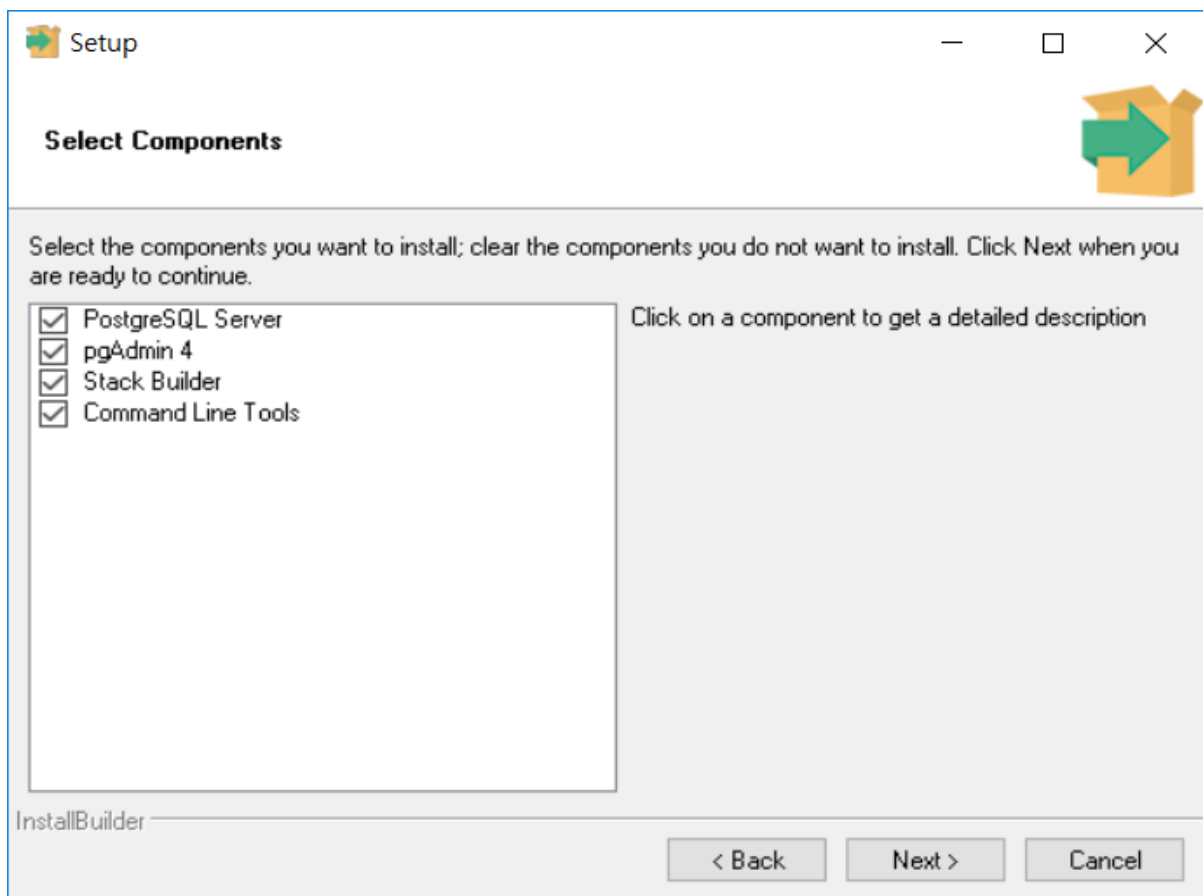
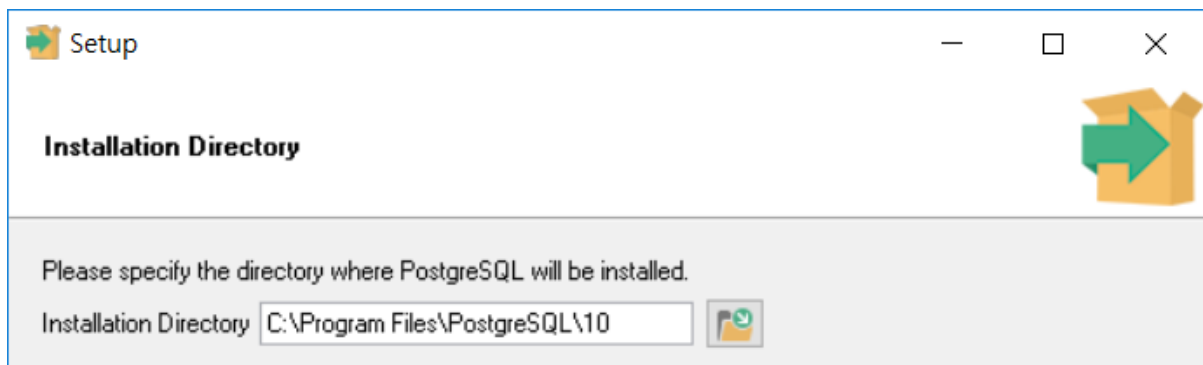
▼

DOWNLOAD NOW

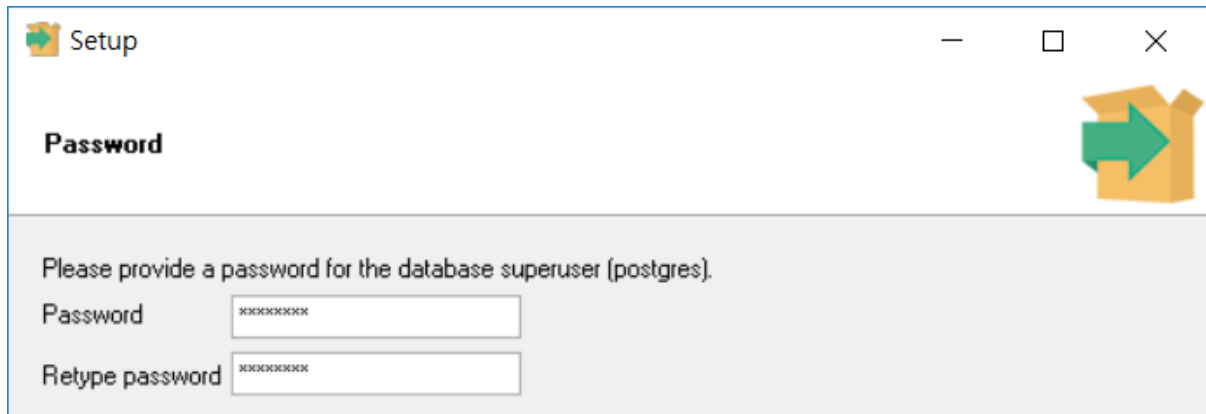
자, 이제 PostgreSQL 을 설치해볼까요?!



PostgreSQL 과 데이터가 설치되는 기본 경로는 아래와 같습니다.

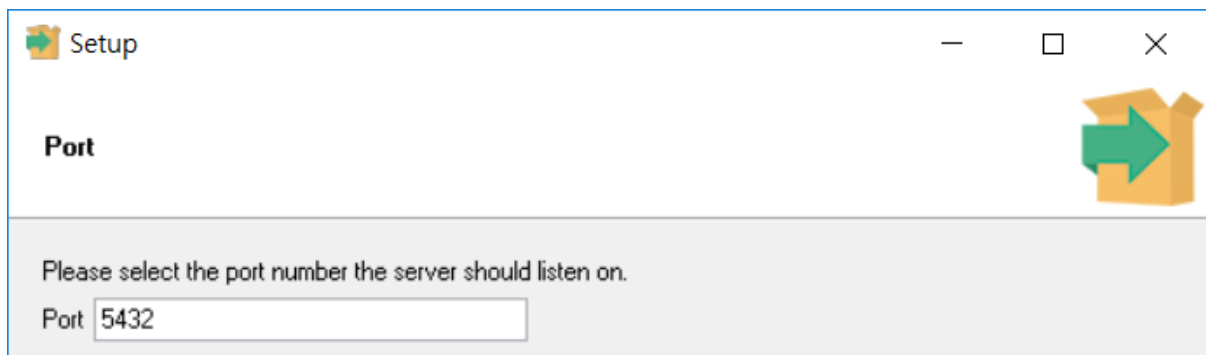


데이터베이스 운용 관리자(superuser)를 위한 패스워드를 설정합니다.



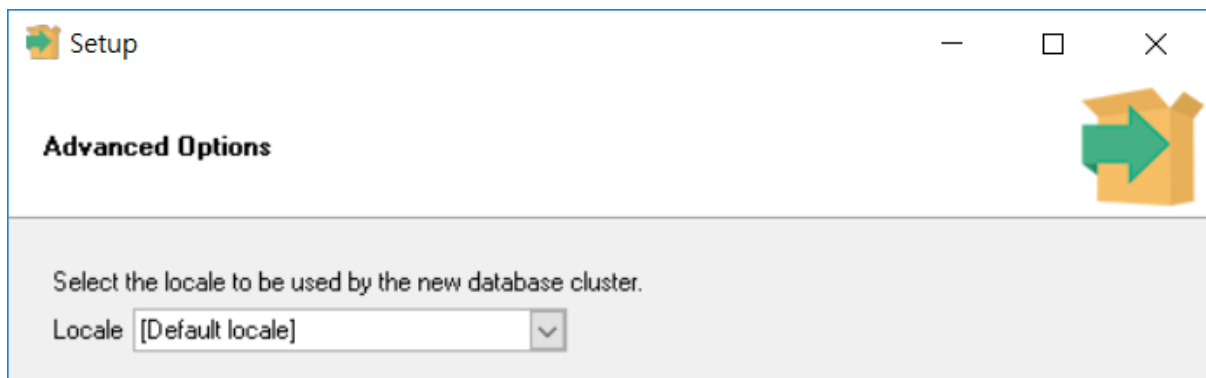
The image shows a window titled "Setup" with a close button and a maximize button. The window has a title bar with a green arrow icon. The main content area is titled "Password" and contains the text "Please provide a password for the database superuser (postgres)." Below this text are two input fields: "Password" and "Retype password". Both fields contain masked text represented by "X" characters. A green arrow icon is visible in the top right corner of the window.

서버가 수신할 포트 번호를 설정합니다. PostgreSQL 은 기본 포트로 5432 를 씁니다.



The image shows a window titled "Setup" with a close button and a maximize button. The window has a title bar with a green arrow icon. The main content area is titled "Port" and contains the text "Please select the port number the server should listen on." Below this text is a single input field labeled "Port" with the value "5432" entered. A green arrow icon is visible in the top right corner of the window.

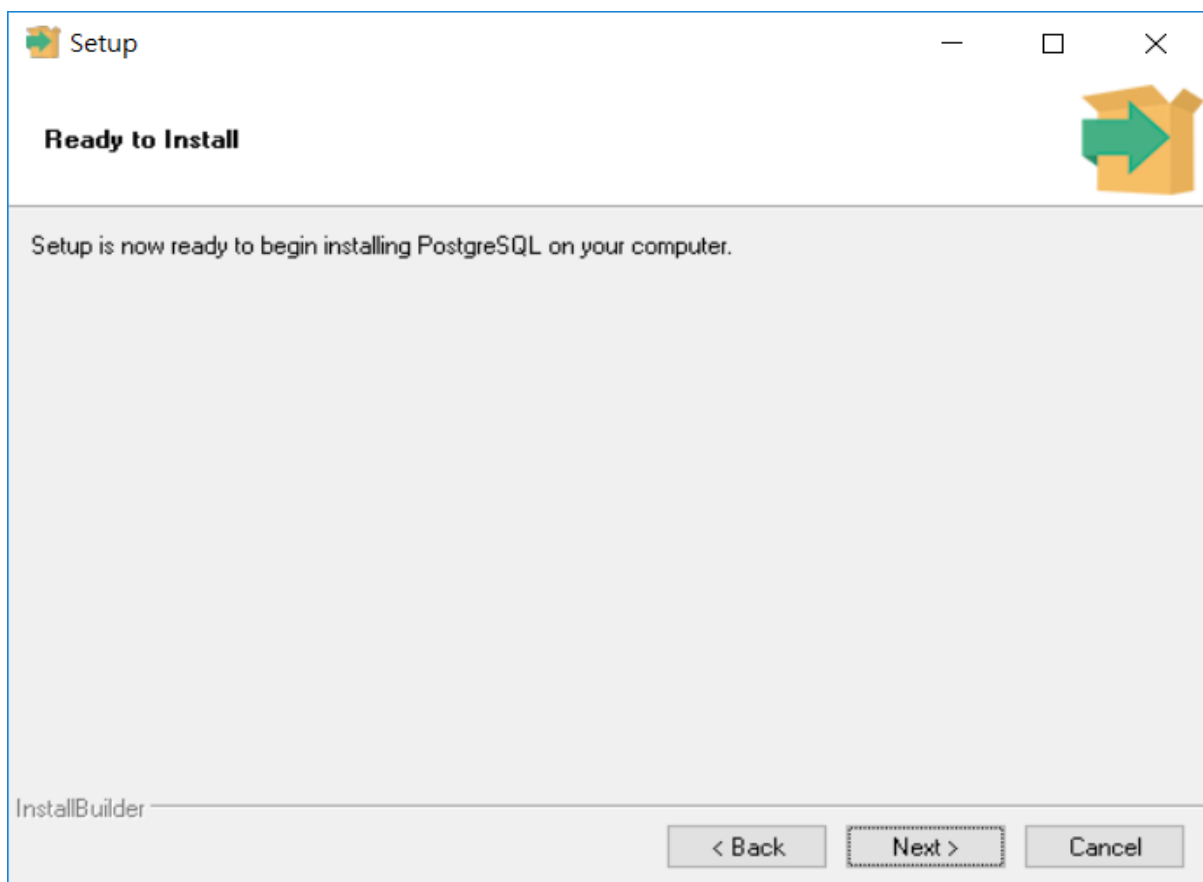
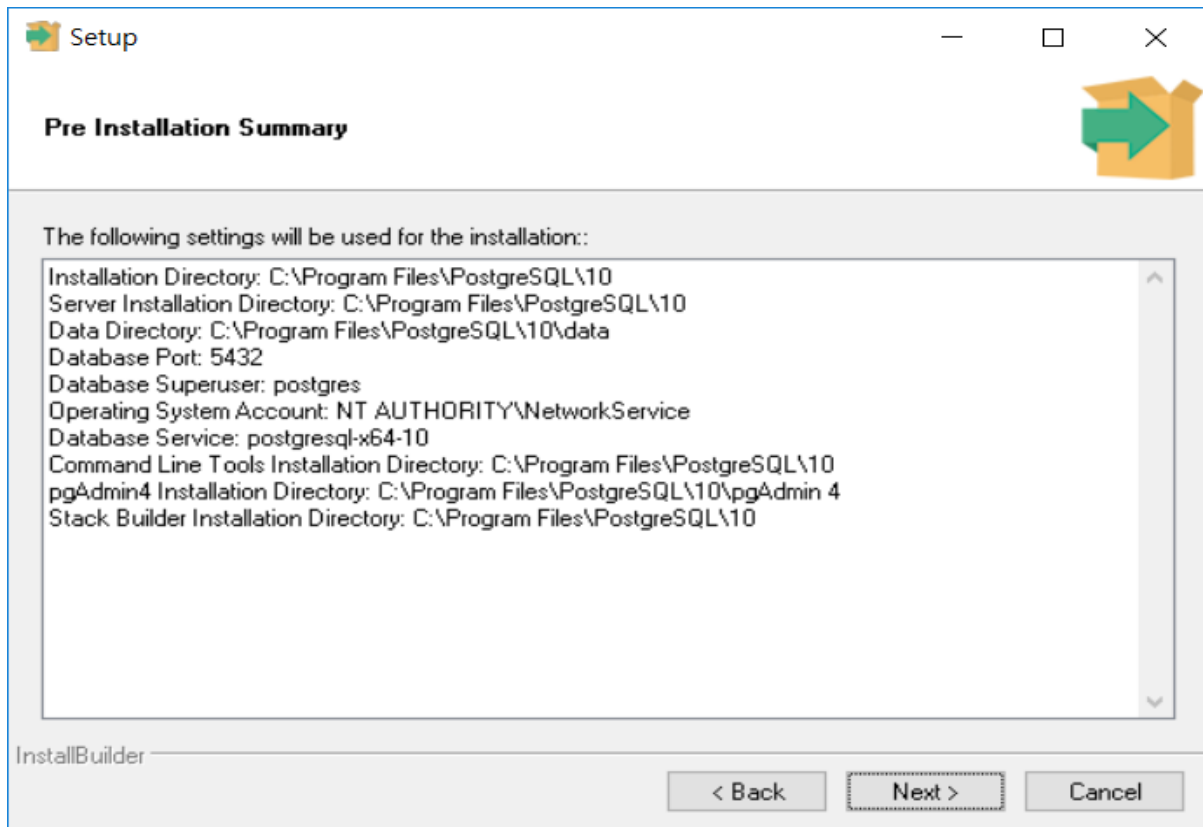
언어 설정인데요, [Default locale]로 그대로 두시면 운영체제의 언어를 자동 선택합니다.



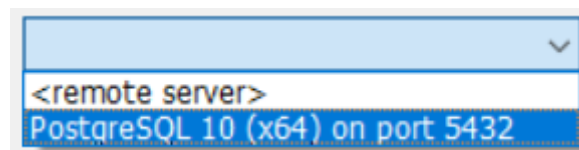
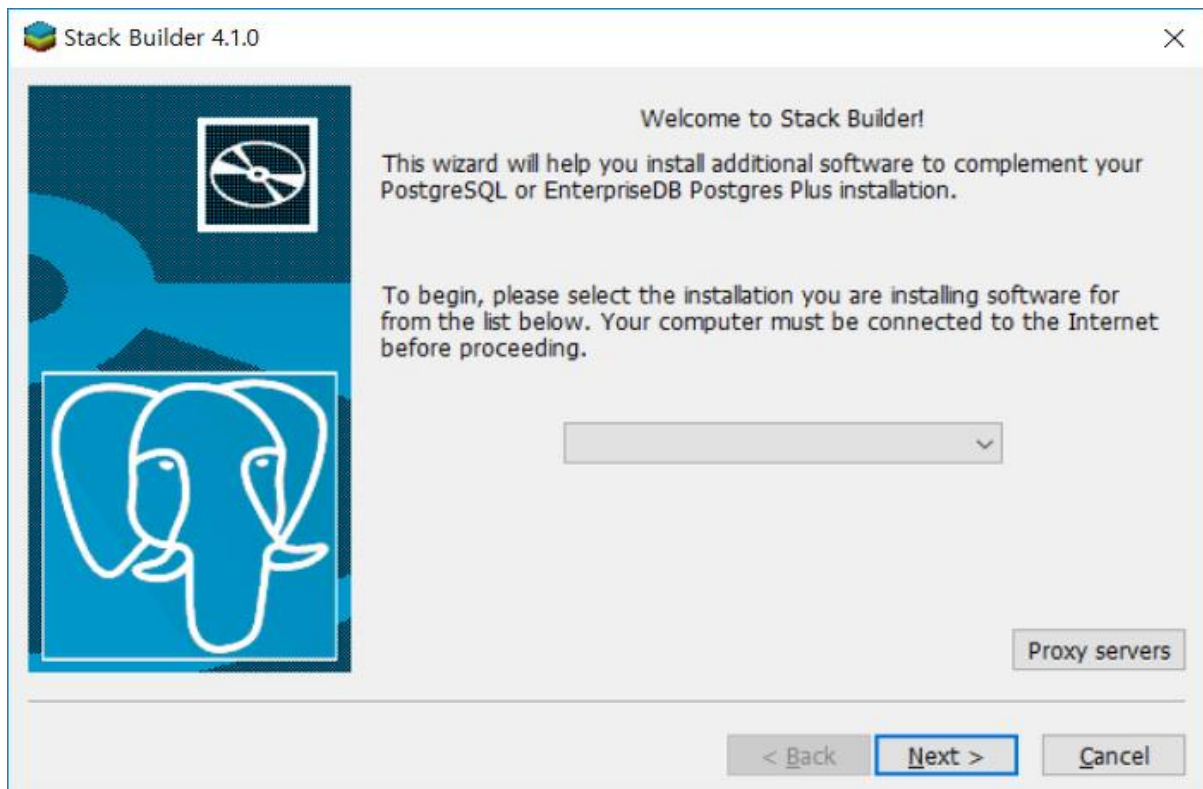
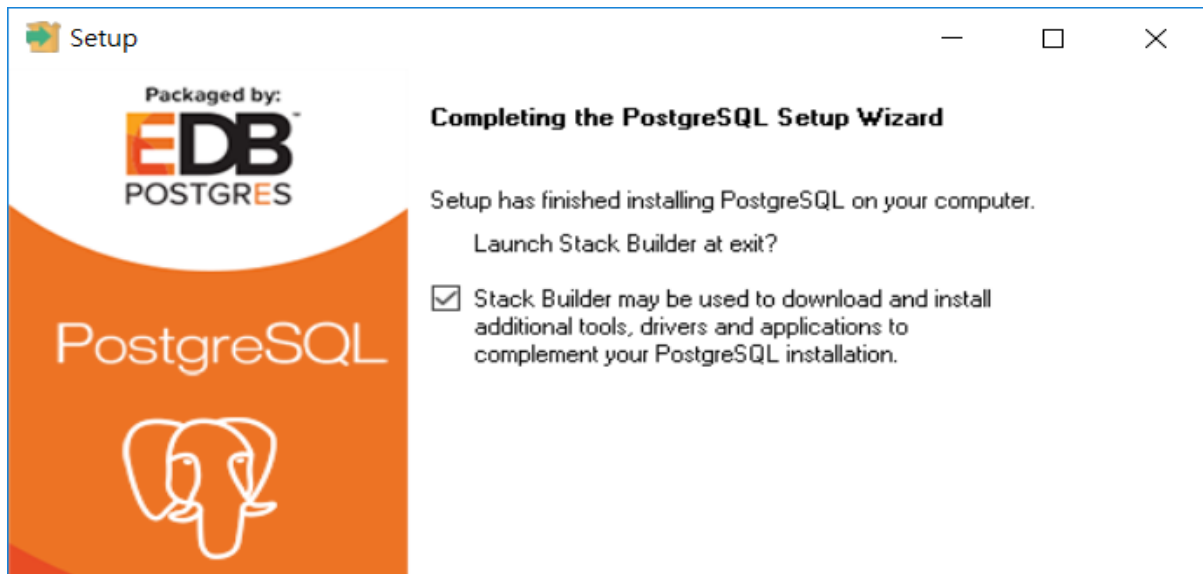
The image shows a window titled "Setup" with a close button and a maximize button. The window has a title bar with a green arrow icon. The main content area is titled "Advanced Options" and contains the text "Select the locale to be used by the new database cluster." Below this text is a dropdown menu labeled "Locale" with the value "[Default locale]" selected. A green arrow icon is visible in the top right corner of the window.

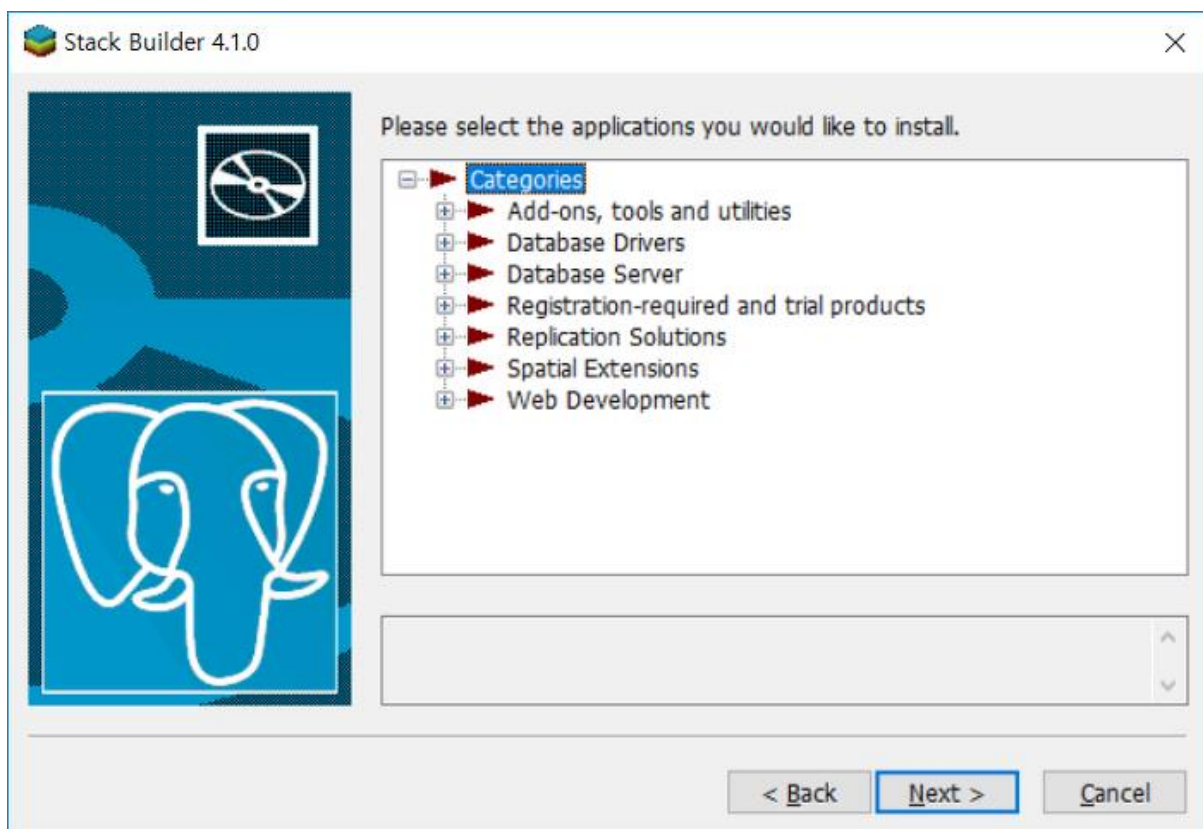
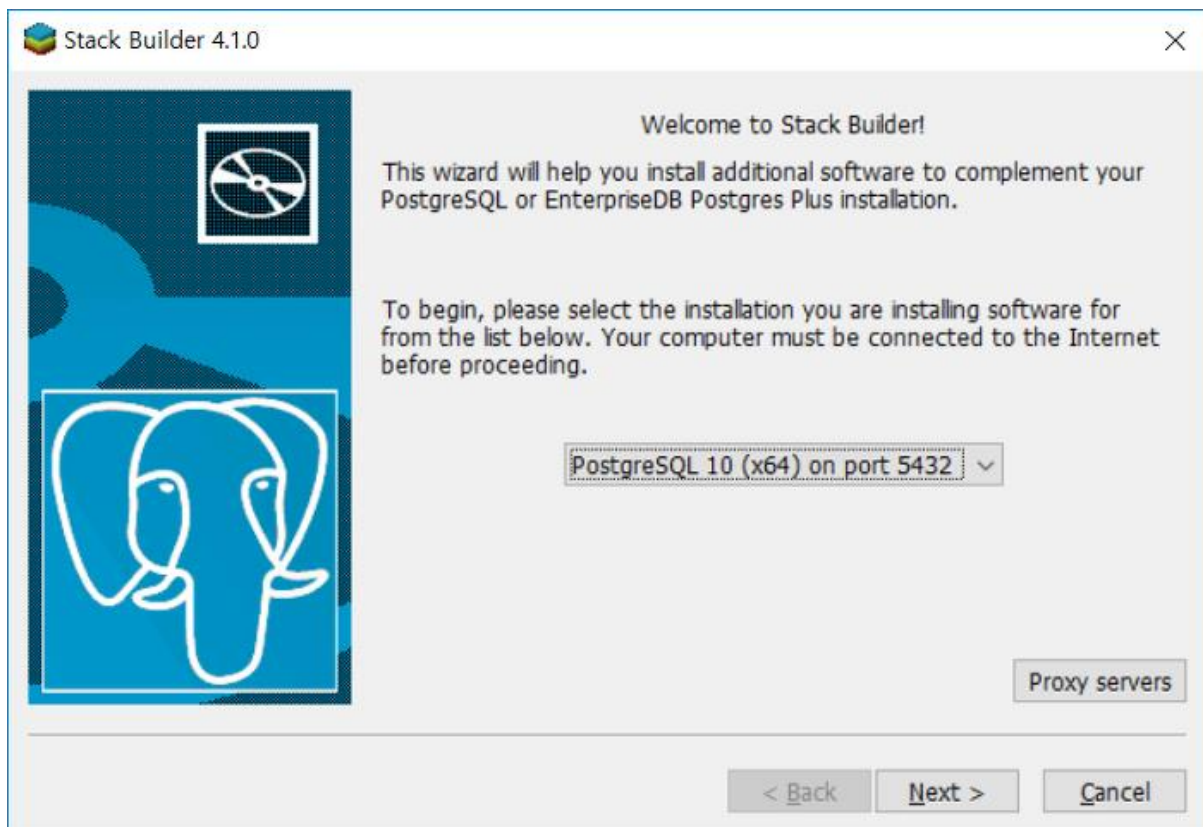
설치 설정을 확인하고, 이제 PostgreSQL 을 설치합니다.



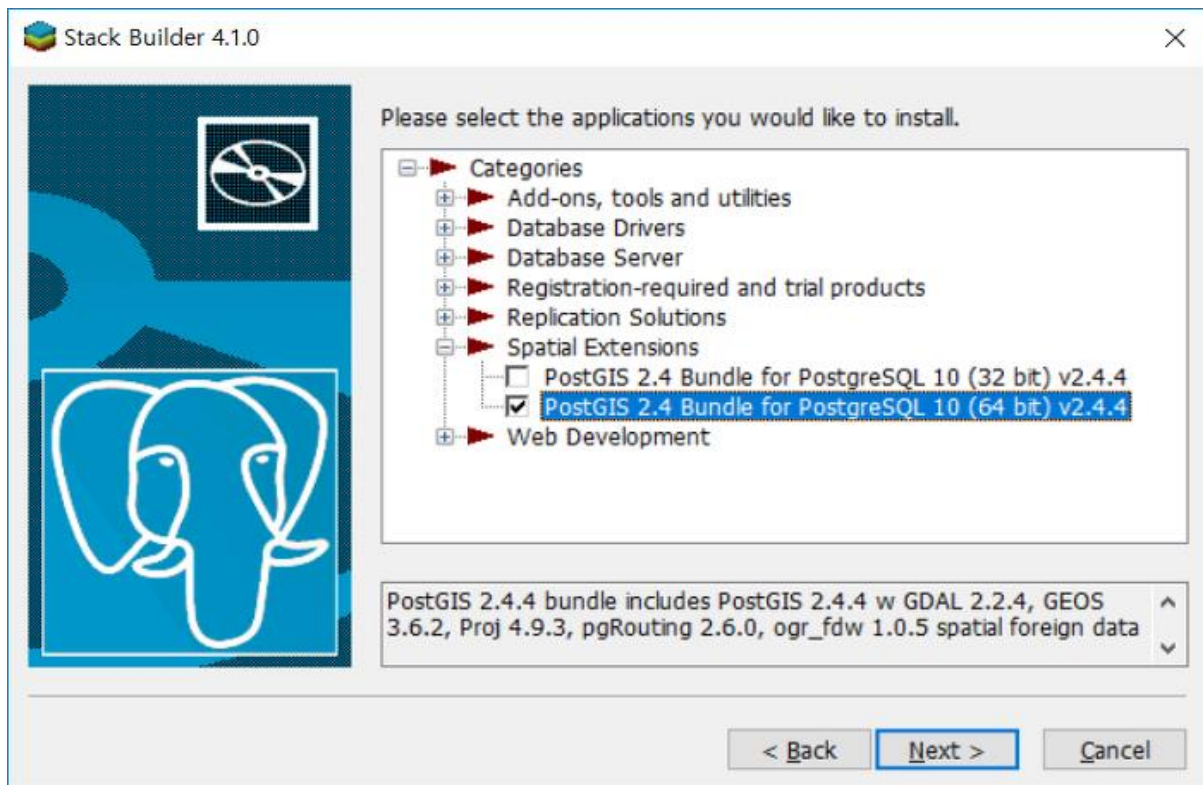


PostgreSQL 설치가 완료되면 Stack Builder 실행을 묻습니다. Stack Builder 는 PostgreSQL 을 보완하는 추가적인 도구, 드라이버, 애플리케이션을 다운로드 받고 설치하는데 이용됩니다. 여기서는 PostGIS 를 설치해야 하므로 Stack Builder 를 이어서 실행합니다.

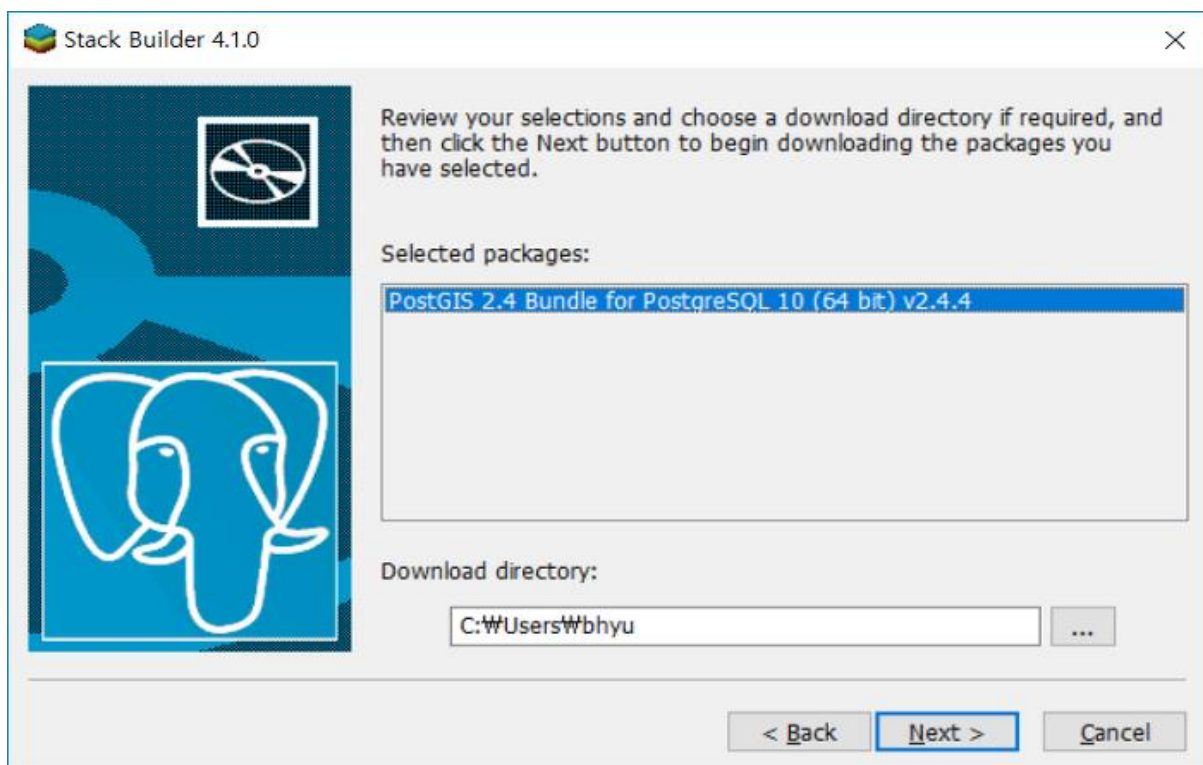




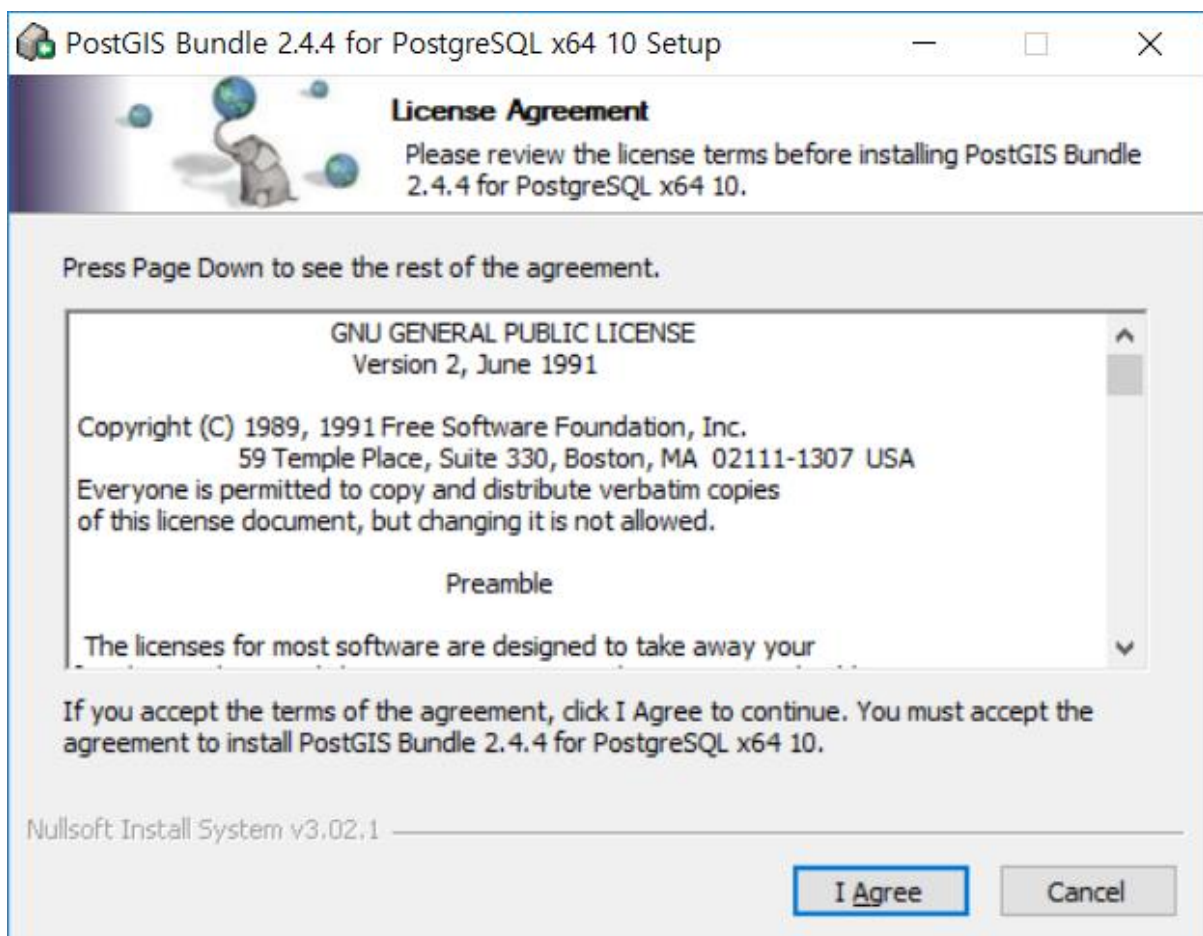
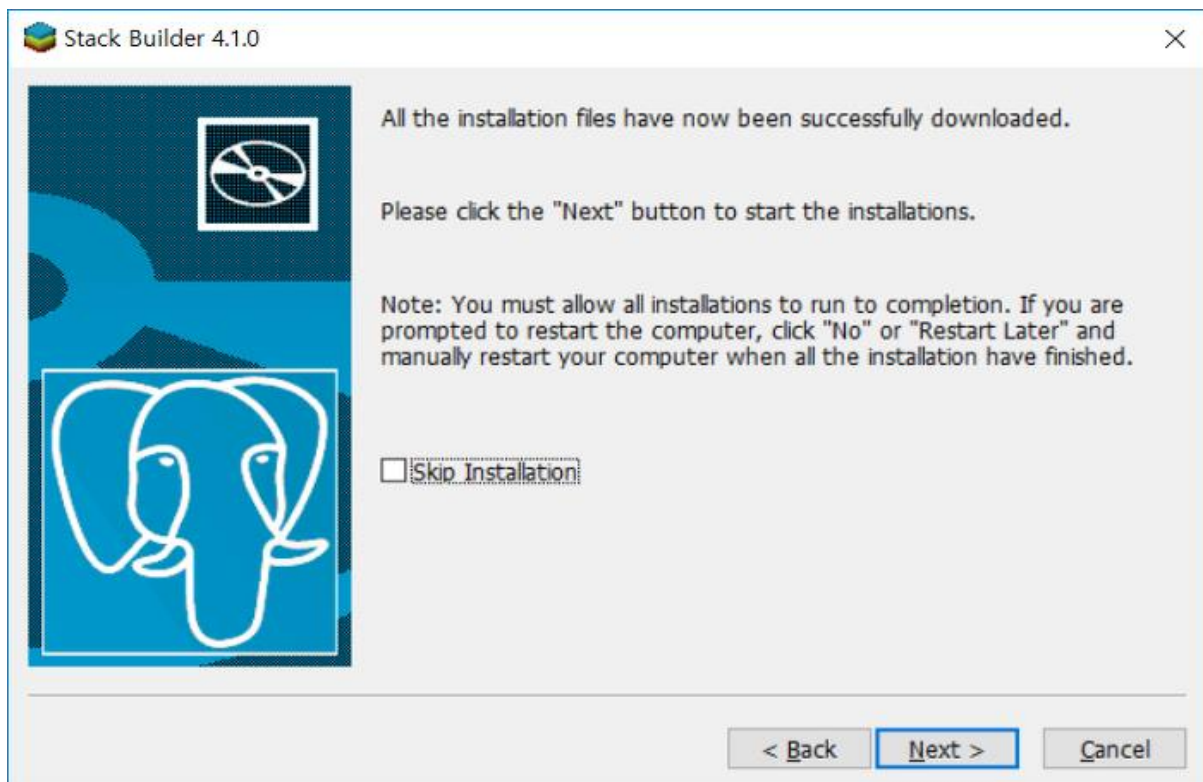
Stack Builder 에서 Spatial Extensions 하부의 PostGIS 를 선택합니다. 설치 버전은 PostGIS 개발팀이 2018 년 4 월 6 일 배포한 버그 수정 패치(<http://postgis.net/2018/04/06/postgis-patches/>)입니다.

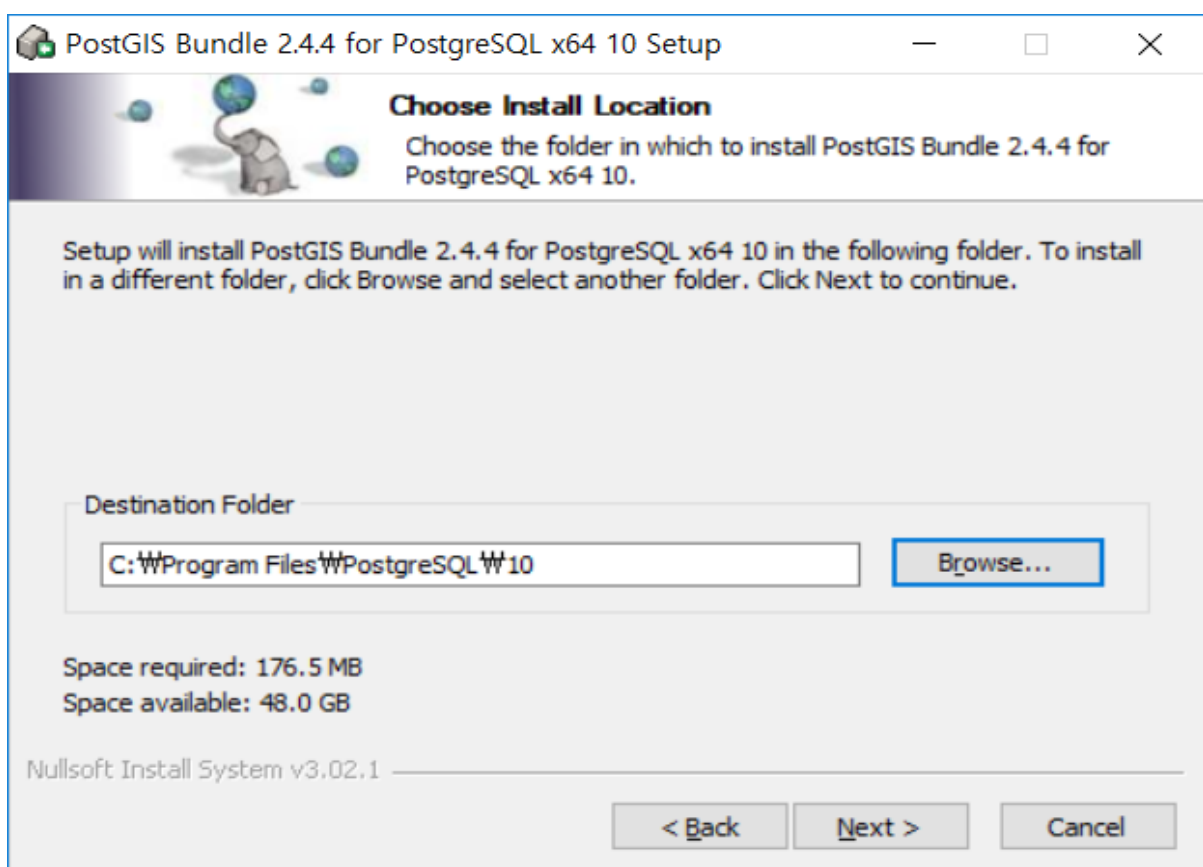
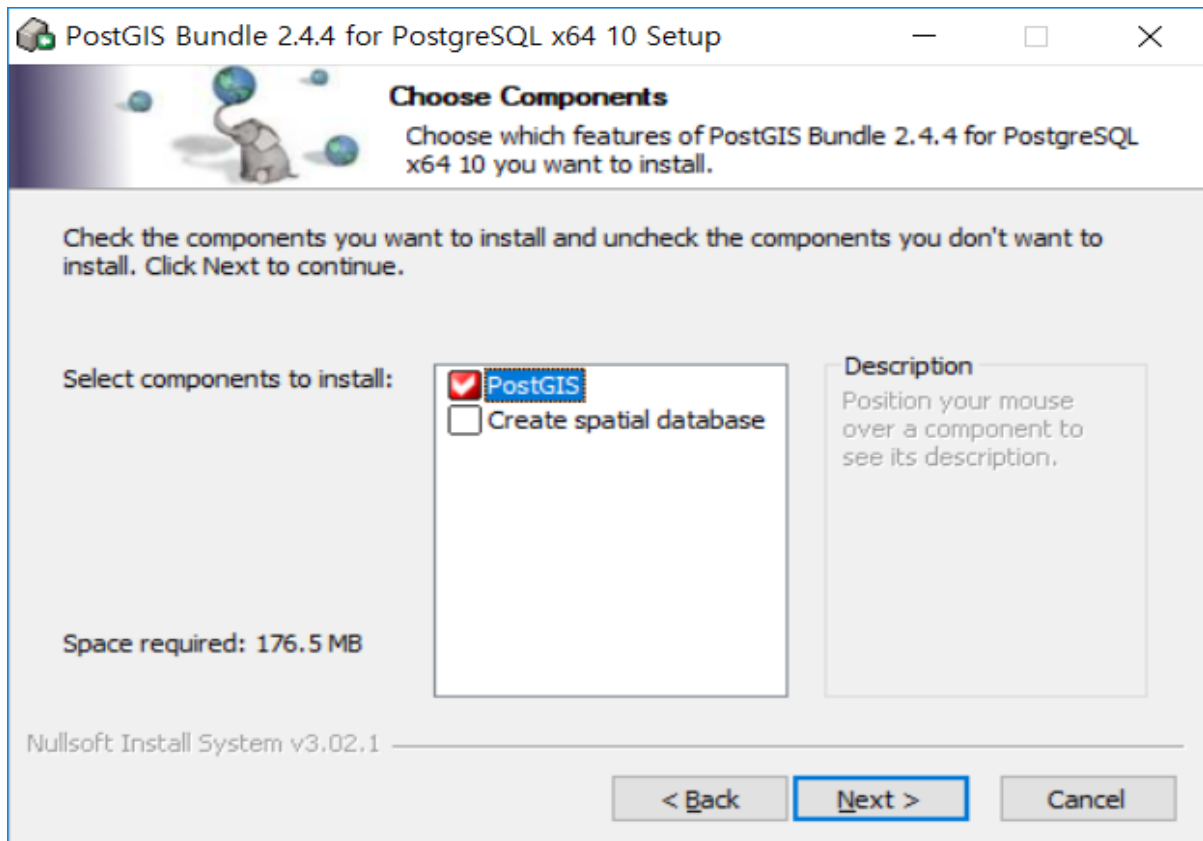


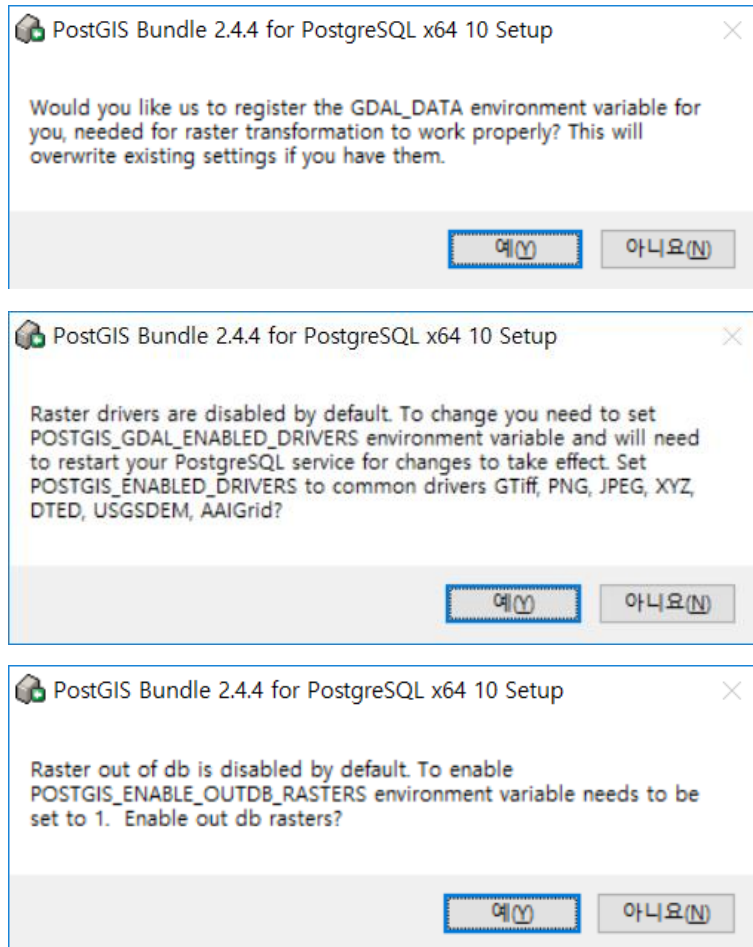
이후 설치 화면은 아래와 같습니다.



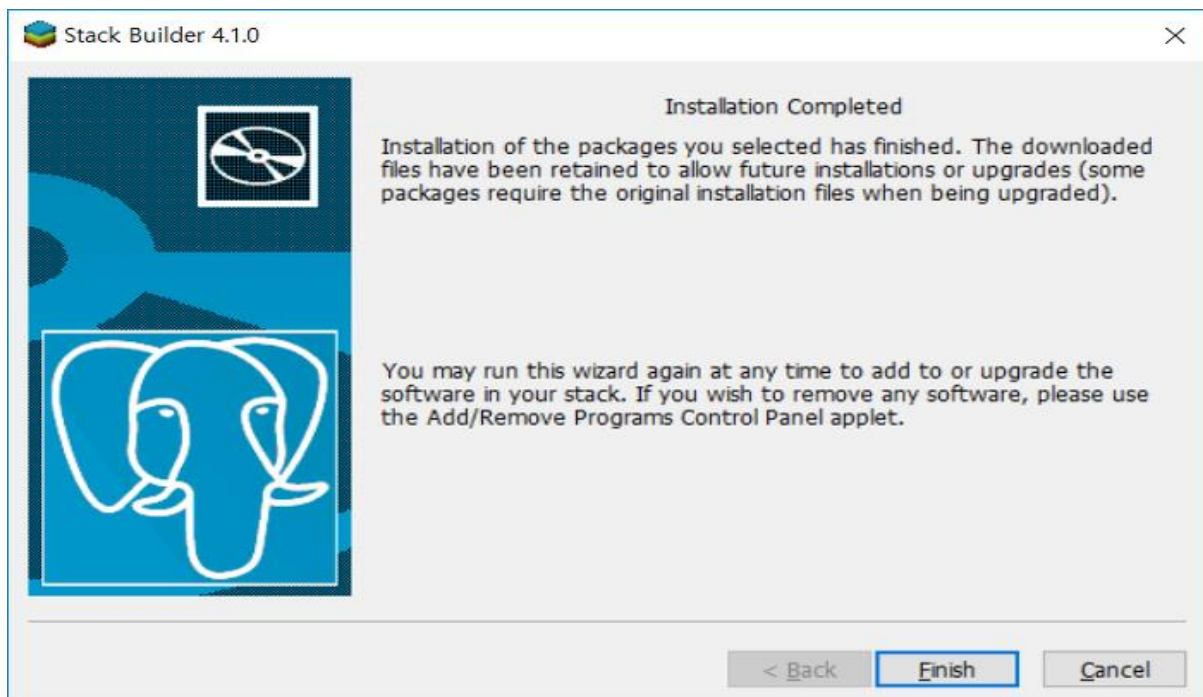








이제 PostGIS 까지 설치를 마쳤습니다. PostgreSQL 을 한 번 실행해볼까요?!

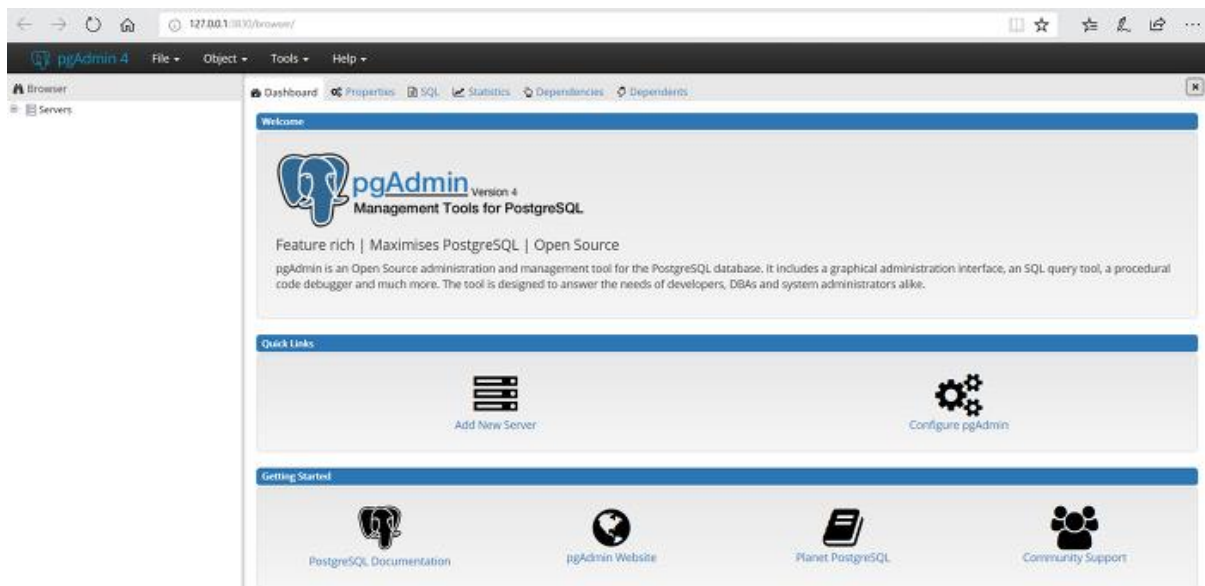
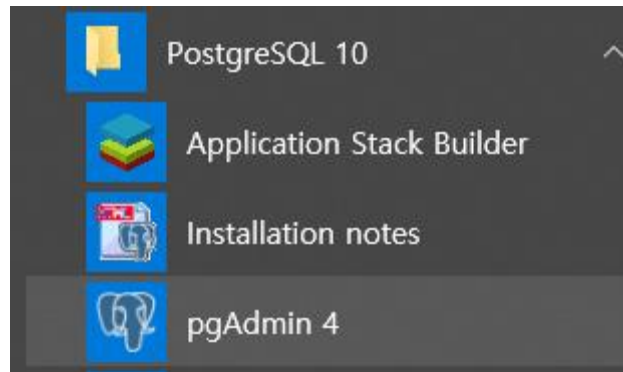




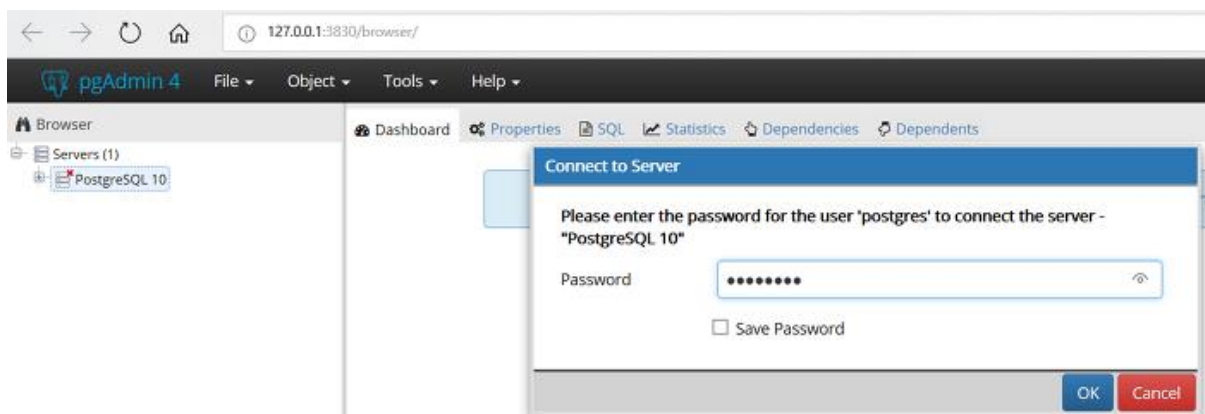
PostgreSQL 10 하부에 위치한 pgAdmin 4 를 실행합니다. pgAdmin 은 PostgreSQL 을 위한 오픈소스 관리개발 플랫폼(Open Source administration and development platform for PostgreSQL)입니다.

공식 홈페이지: <https://www.pgadmin.org/>

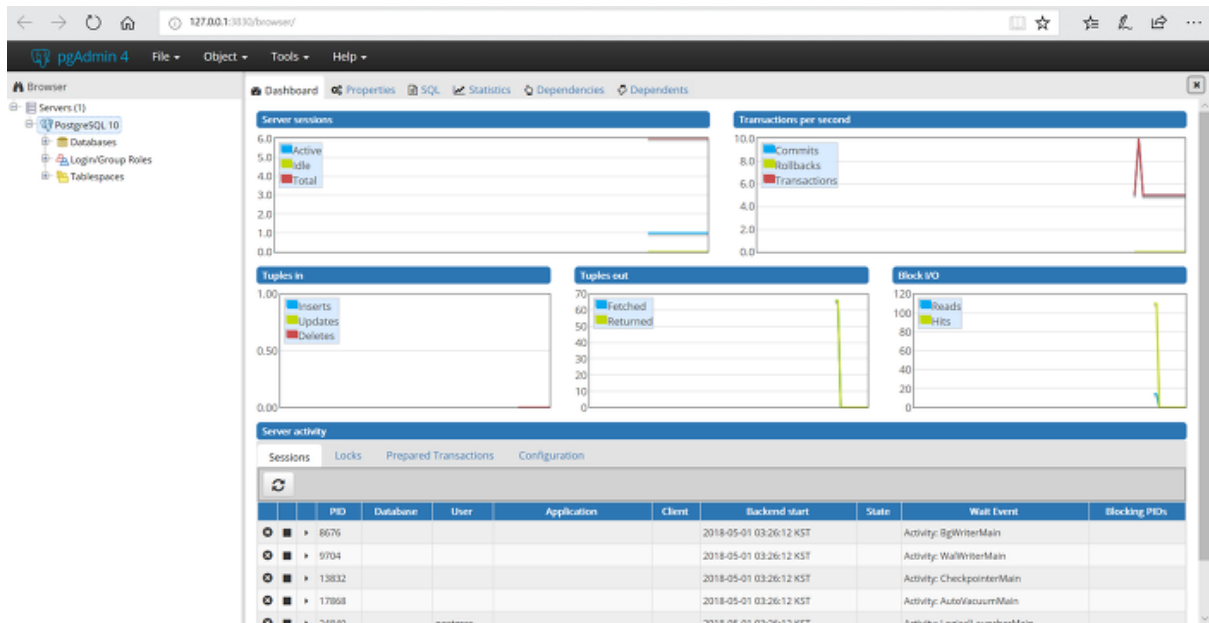
PostgreSQL 9.6 버전 이후부터는 pgAdmin 4, 9.5 버전까지는 pgAdmin 3 가 적용됩니다.



좌측 Brower > Servers (1) > PostgreSQL 10 을 더블클릭하고 설치할 때 입력했던 패스워드를 입력합니다.



아래와 같이 PostgreSQL 10 이 활성화되었습니다. 이제 PostGIS 를 학습해 볼까요?!



## [2] PostgreSQL/PostGIS 에서 DB 생성하고 공간데이터 추가하기

안녕하세요? 이전 글에서는 PostgreSQL/PostGIS 를 설치해 봤습니다. 이번 글에는 PostgreSQL/PostGIS 에서 DB 를 생성하고 공간데이터를 추가하는 방법을 학습해 보겠습니다.

일단 공간데이터를 획득하기 위해 서울시에서 제공하는 오픈데이터 포털에 접속해 보겠습니다.

서울열린데이터광장 | <http://data.seoul.go.kr/>

'서울시 법정구역 읍면동 공간정보'를 검색해 보겠습니다.



아래 '서울시 법정구역 읍면동 공간정보 (좌표계: WGS1984)' 데이터셋에서,

데이터셋 (2건)

결과 더보기 +

<a href="#">원천</a> <a href="#">SHEET</a> <a href="#">MAP</a> <a href="#">FILE</a> <a href="#">OPEN API</a> 서울시 법정구역 읍면동 공간정보 (좌표계: ITRF2000) <일반행정> 행정 제공기관 : 서울특별시 수정일 : 2016-04-12	<a href="#">원천</a> <a href="#">SHEET</a> <a href="#">MAP</a> <a href="#">FILE</a> <a href="#">OPEN API</a> 서울시 법정구역 읍면동 공간정보 (좌표계: WGS1984) <일반행정> 행정 제공기관 : 서울특별시 수정일 : 2016-04-12
---	--

아래 'MAP' 버튼을 클릭합니다.

MAP

SHP 파일목록의 'TC\_SPBE17\_2015\_W\_SHP.zip'을 다운로드하고 압축을 해제합니다.

Sheet

Open Api

Map

File

SHP 파일목록

NO	파일명	파일크기(KB)	마지막수정일	최초공개일	다운로드
1	TC_SPBE17_2015_W_SHP.zip	1830833	2016.04.14	2016.04.14	DOWN

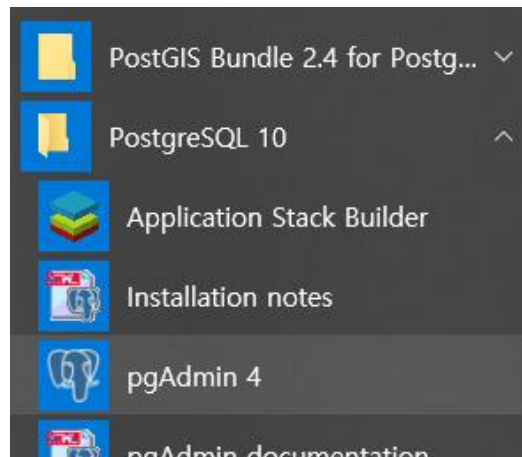
DXF 파일목록

NO	파일명	파일크기(KB)	마지막수정일	최초공개일	다운로드
1	TC_SPBE17_2015_W_DXF.zip	1771771	2016.04.14	2016.04.14	DOWN

GML 파일목록

NO	파일명	파일크기(KB)	마지막수정일	최초공개일	다운로드
1	TC_SPBE17_2015_W_GML.zip	1513367	2016.04.14	2016.04.14	DOWN

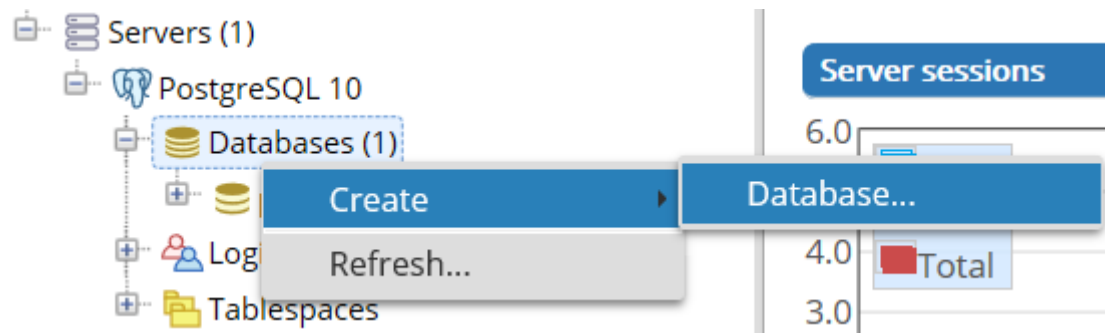
이제 공간데이터가 있으니 시작해볼까요?! pgAdmin 4 프로그램을 실행합니다.



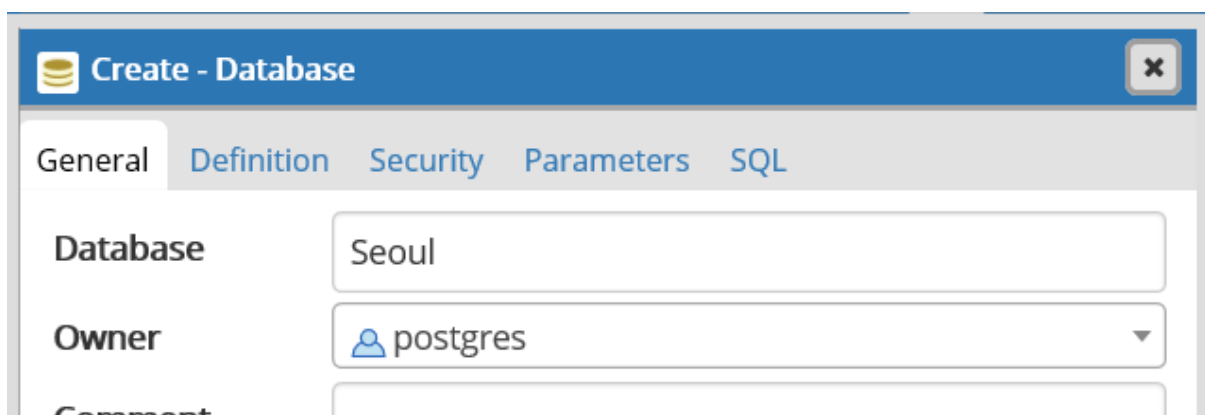
pgAdmin 은 PostgreSQL 을 조작하도록 안내하는 그래픽 유저 인터페이스(Graphic User Interface, GUI) 입니다.



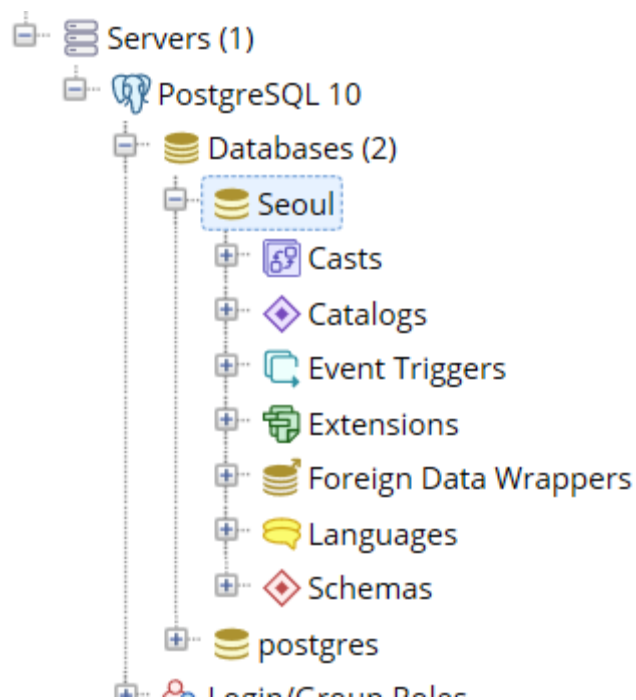
아래 그림처럼 Databases 카테고리를 우 클릭하고 'Create > Database'를 선택합니다.



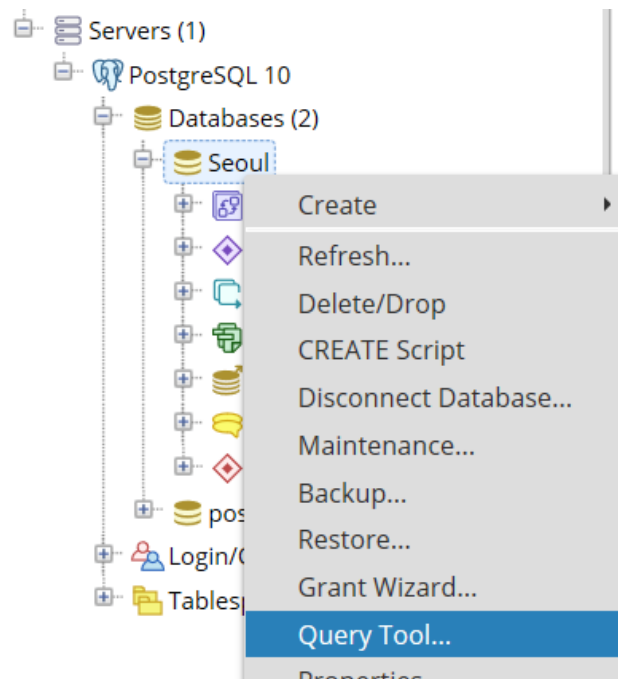
데이터베이스명을 입력합니다. 여기서는 Seoul 이라고 지정했습니다.



Seoul DB 가 생성되었습니다. 이제 PostGIS 확장 모듈을 데이터베이스 안에 로드해야 할 차례입니다.

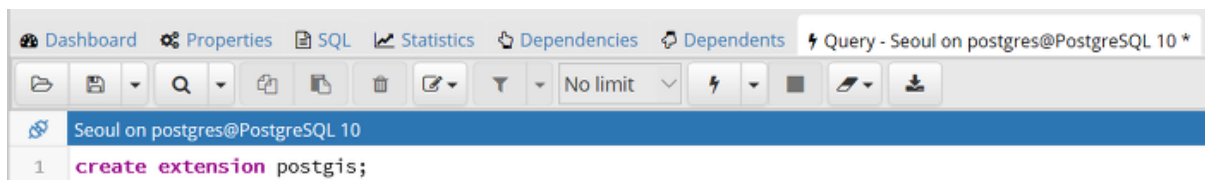


SQL 쿼리 빌더를 열어서 명령문을 적어보겠습니다. DB 명을 우클릭 후 Query Tool 을 선택합니다.

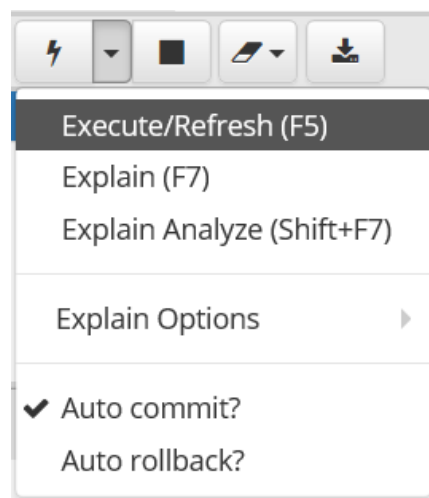


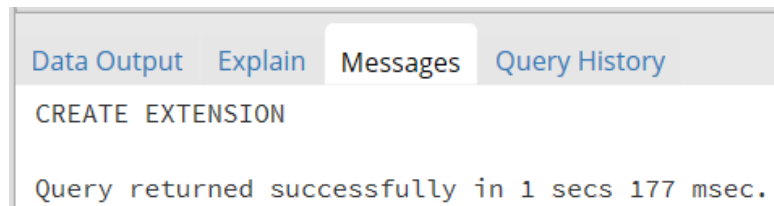
쿼리 빌더가 나타나면 PostGIS 확장 모듈을 로드하기 위한 명령어를 다음과 같이 입력합니다.

```
create extension postgis;
```

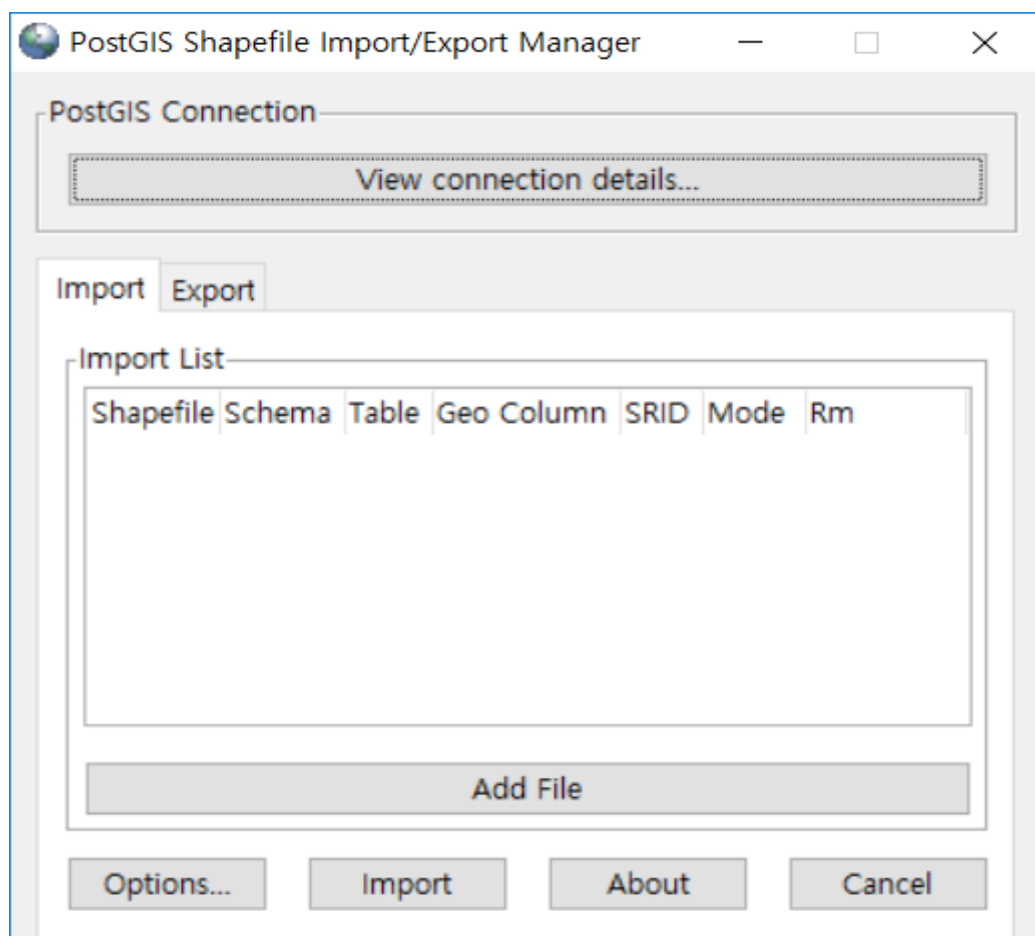
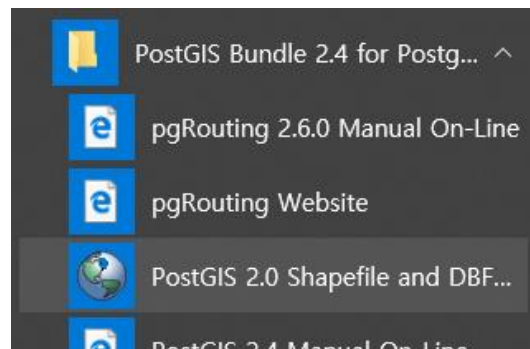


그리고 쿼리 실행 단추를 누르거나 F5 를 누르면 실행이 됩니다.



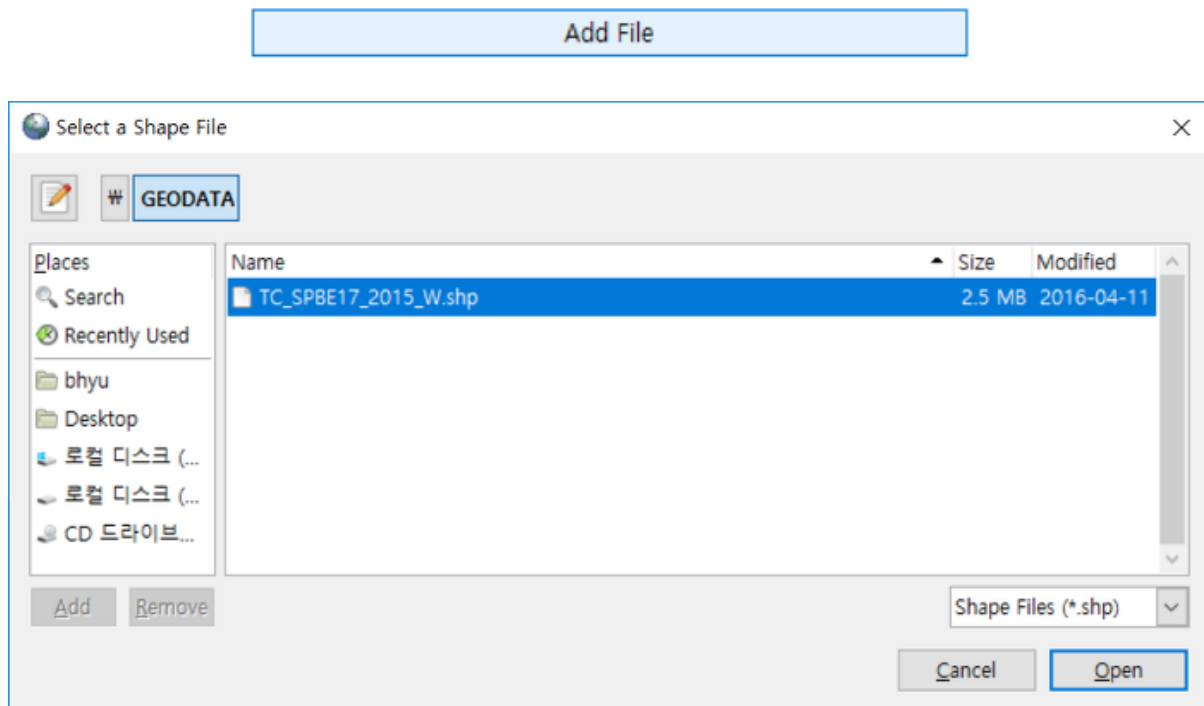


이제 PostGIS Shapefile Importer/Export Manager 를 사용해서 SHP 파일을 데이터베이스안에 올려보겠습니다.

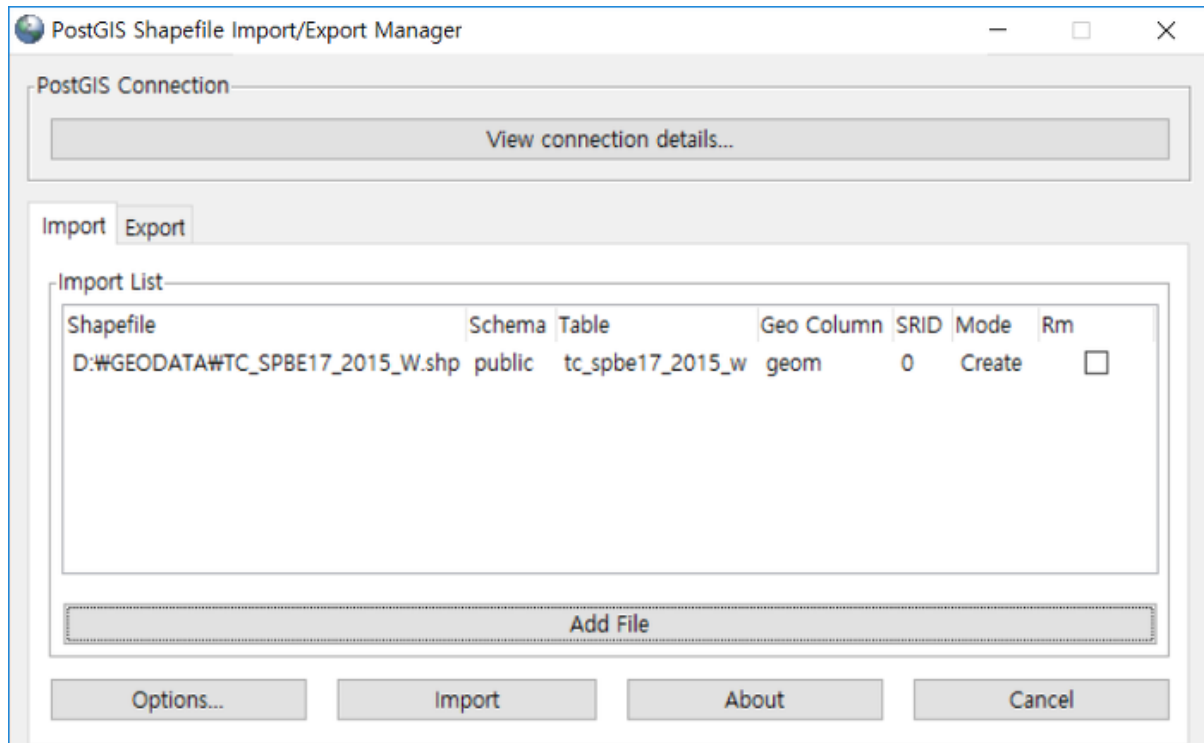




파일 추가(Add File) 버튼을 눌러서 SHP 파일을 지정합니다.



Import List 에 현재 올라갈 SHP 파일의 특성을 나타내는 정보들이 보입니다.



여기서 SRID 란 공간 참조 식별자(Spatial Reference Identifier)의 줄임 말입니다. 풀어서 이야기하면, 데이터의 지리적 좌표계와 투영법에 대한 정보를 구분해 주는 유일한 식별 코드라고 할 수 있습니다.

SRID  
0

로드하고자 하는 데이터의 SRID 정보를 알기 위해 .prj 로 끝나는 파일을 메모장에서 열어보겠습니다.

TC\_SPBE17\_2015\_W.dbf  
TC\_SPBE17\_2015\_W.prj  
TC\_SPBE17\_2015\_W.shp  
TC\_SPBE17\_2015\_W.shx

TC\_SPBE17\_2015\_W.prj - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID  
["WGS_1984",6378137.0,298.257223563]],PRIMEM  
["Greenwich",0.0],UNIT["Degree",0.0174532925199433]]
```

이 좌표계의 SRID 코드를 알기 위해서는 아래 웹사이트에 정보를 복사해서 붙여 넣거나 prj 파일을 업로드한 후 Convert 버튼을 눌러서 결과를 확인합니다. SRID 코드는 4326 이라는 것을 알 수 있습니다.

Prj2EPSG | <http://prj2epsg.org/search>

# Prj2EPSG

Boundless

Prj2EPSG is a simple service for converting **well-known text** projection information from **.prj files** into standard **EPSG codes**.

Paste a WKT definition or type keywords below:

```
GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID  
["WGS_1984",6378137.0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT  
["Degree",0.0174532925199433]]
```

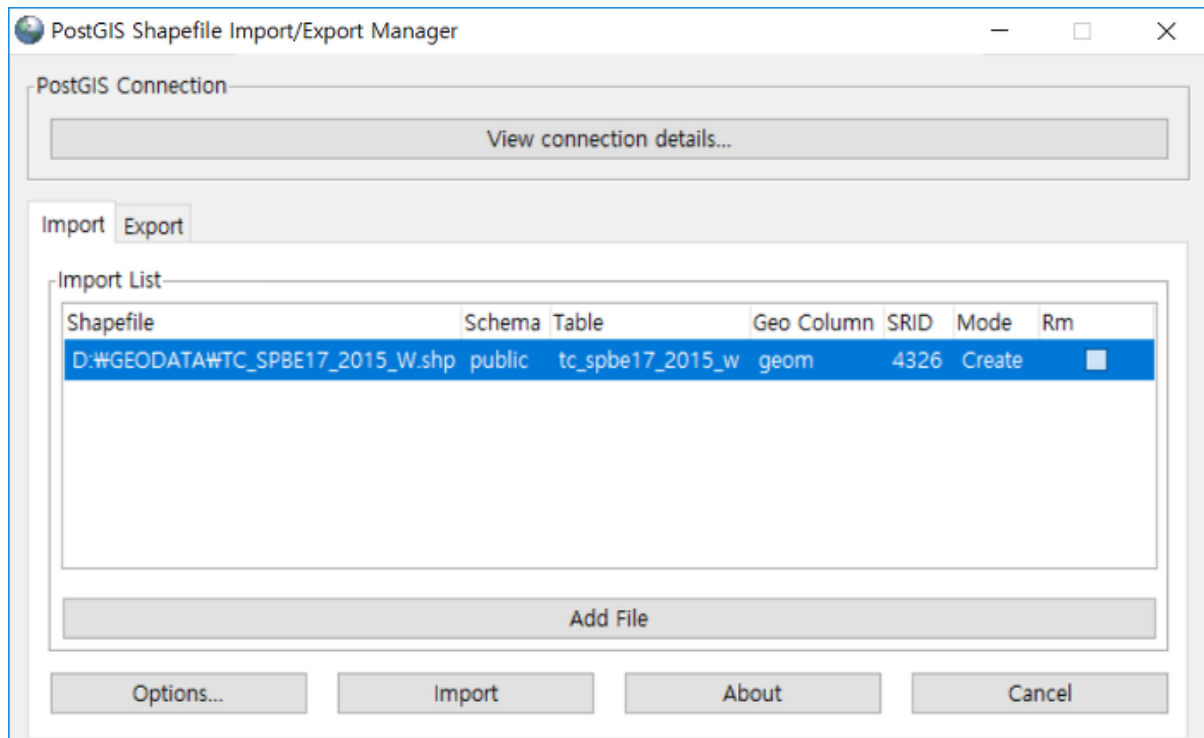
or upload a .prj file:

## Results

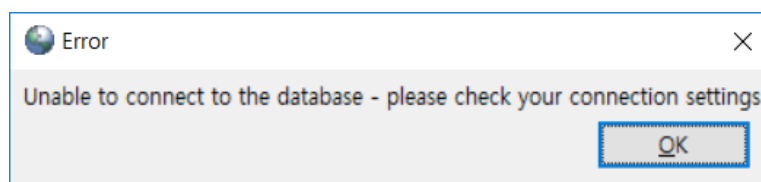
Found a single exact match for the specified search terms:

- [4326](#) - GCS\_WGS\_1984

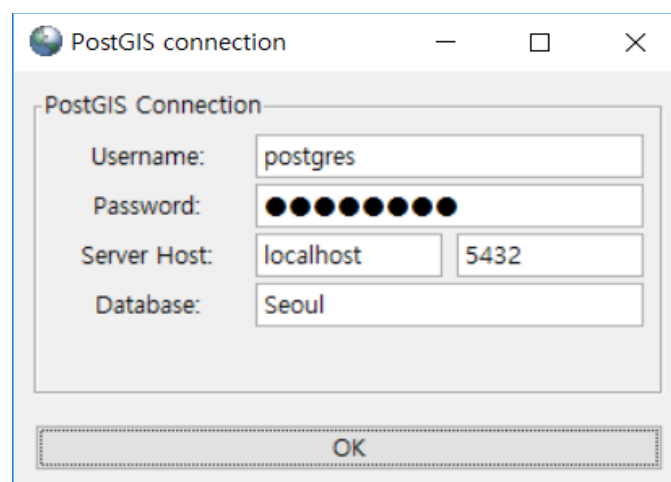
PostGIS 에서 SRID 부분을 클릭해서 해당 코드를 입력하겠습니다.



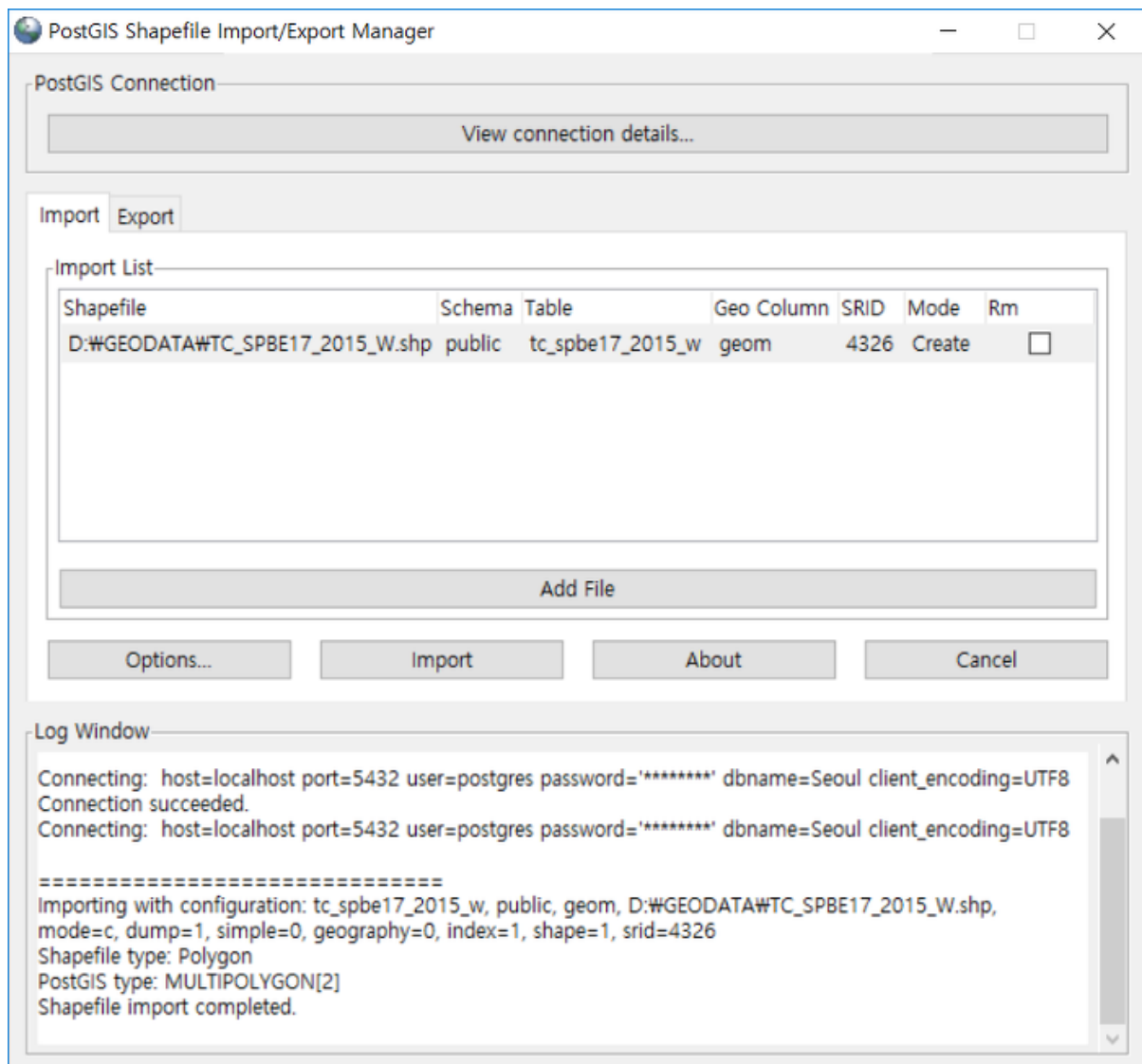
Import 버튼을 클릭했더니 아래와 같이 DB 연결 오류 창이 표시됩니다. View connection details 버튼을 클릭하고 DB 연결 설정을 확인해 보겠습니다.



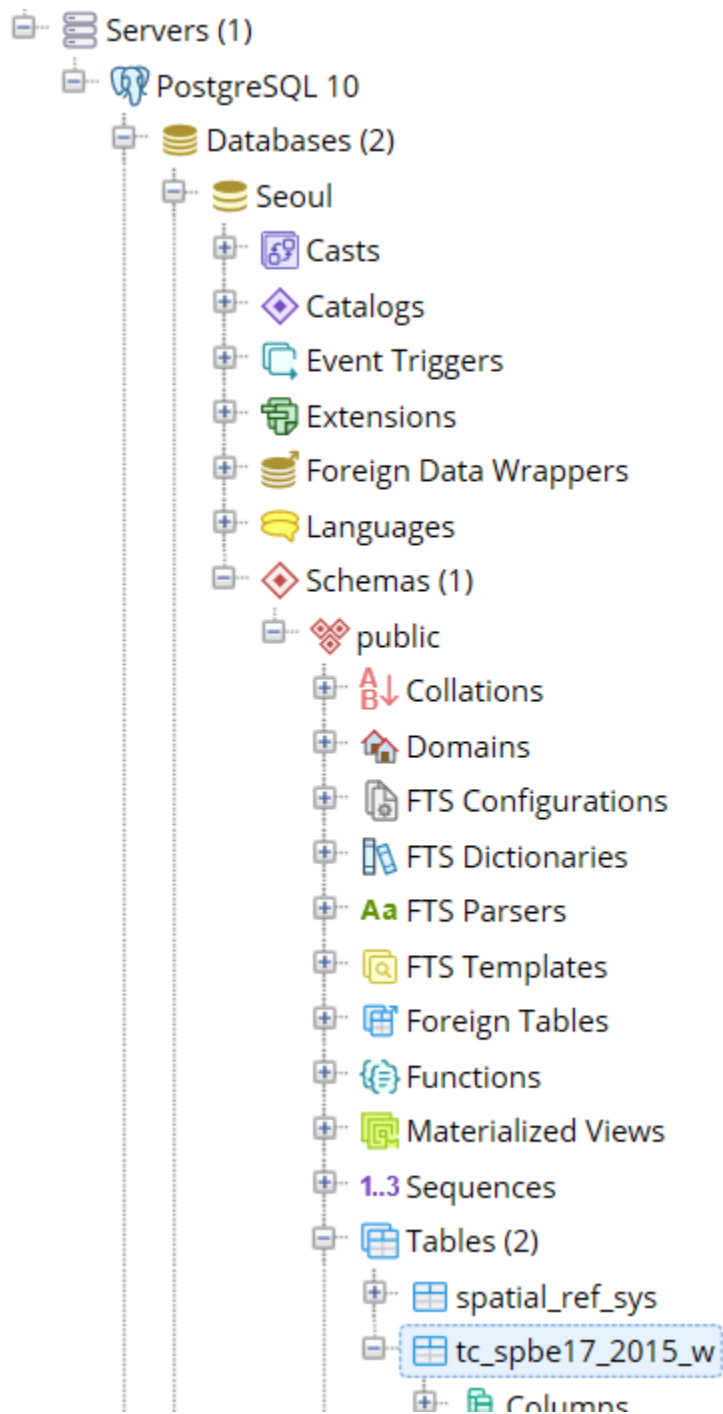
Username, Password, Server Host 를 입력하고 Database 는 Seoul 을 지정합니다.



다시 Import 버튼을 클릭하면 SHP 파일이 Seoul DB 안에 추가됩니다. 이런 방식으로 다른 파일도 SRID 코드 설정 후 Import 버튼을 클릭하시면 됩니다.



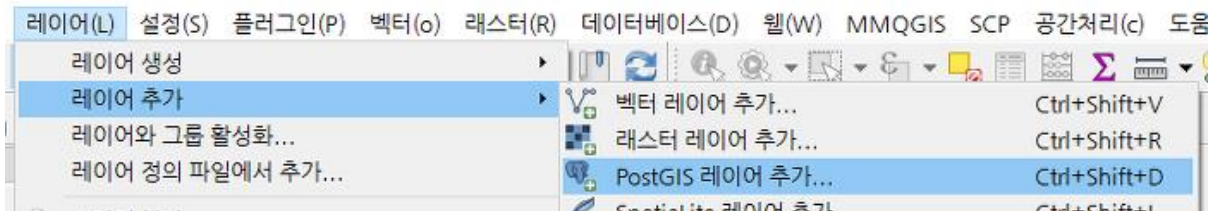
이제 결과를 확인해볼까요?! 아래와 같이 테이블 목록에 서울시 읍면동 SHP 파일이 추가되었습니다.



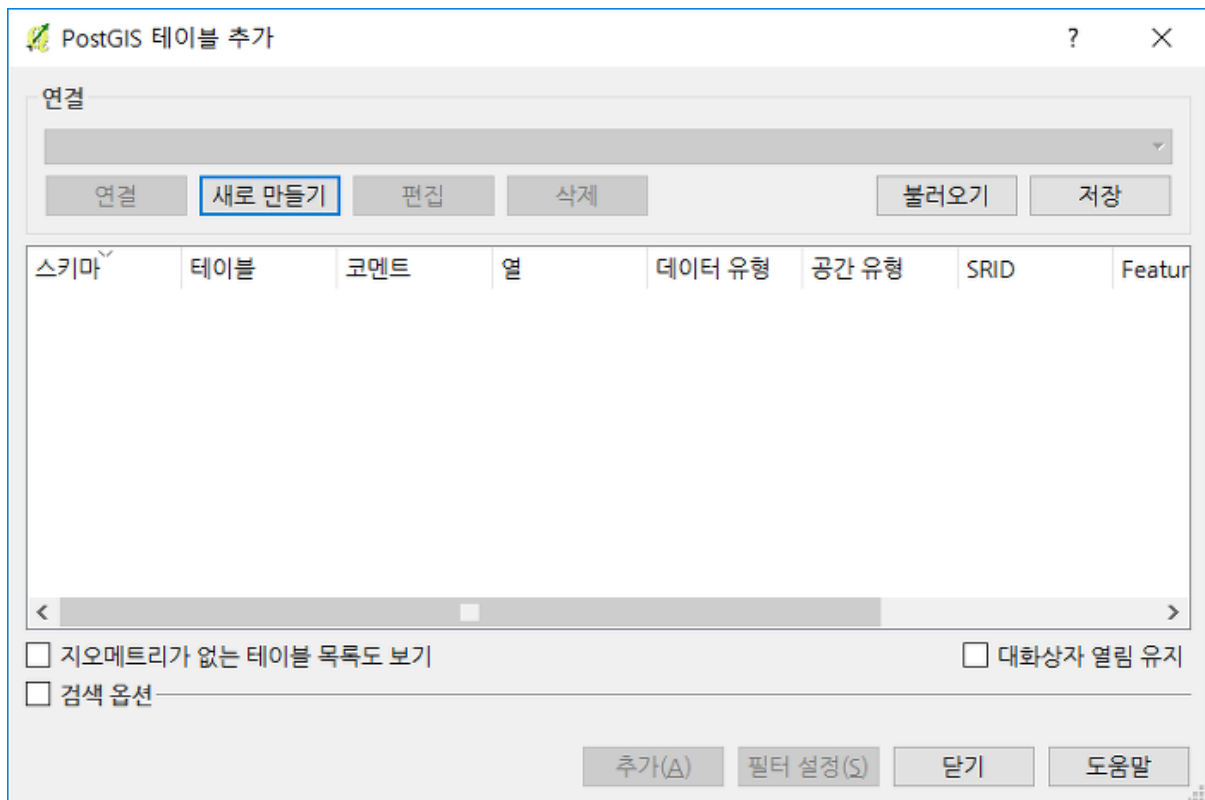
### [3] QGIS 에서 PostGIS 레이어 추가하기

안녕하세요? 앞서 PostGIS 에서 DB 를 생성하고 공간데이터를 추가해 봤는데요, 이번 글에서는 QGIS 에서 이러한 PostGIS 레이어를 추가해보도록 하겠습니다.

QGIS 를 실행하고 '레이어 > 레이어 추가 > PostGIS 레이어 추가'를 클릭합니다.



PostGIS 테이블 추가 창에서 '새로 만들기'를 클릭하고,



'새 PostGIS 연결을 만들기' 창에서 앞서 작업한 Seoul DB 정보를 입력합니다.

'레이어 레지스트리에 있는 레이어만 보이기'도 체크해 보겠습니다.

새 PostGIS 연결을 만들기

연결 정보

이름: Seoul

서비스:

호스트:

포트: 5432

데이터베이스: Seoul

SSL 모드: 사용불가

Authentication | Configurations

사용자 이름: postgres ☒ 저장

비밀번호:  ☐ 저장

Test Connection

☒ 레이어 레지스트리에 있는 레이어만 보이기

☐ 제한되지 않는 행의 유형을 결정하지 않음(GEOMETRY)

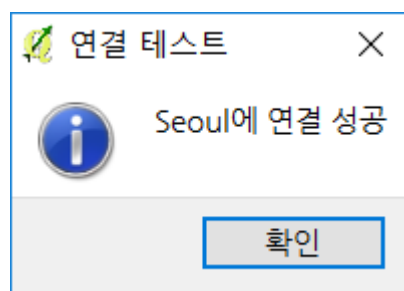
☐ 'public' 스키마에서만 찾기

☐ 지오메트리가 없는 테이블 목록도 보기

☐ 추정된 테이블 메타데이터 사용

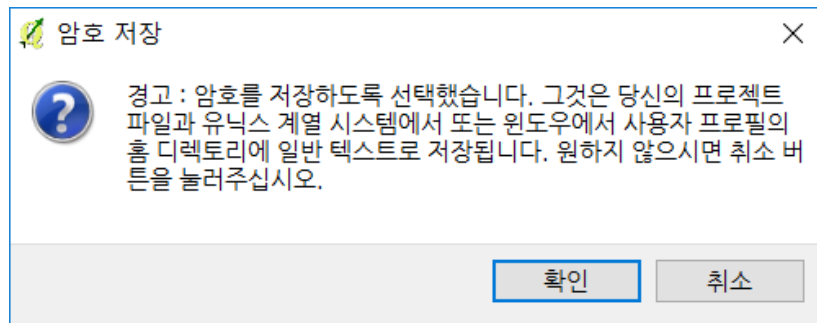
확인 취소 도움말

'Test Connection' 버튼을 클릭하여 연결 테스트를 확인합니다.

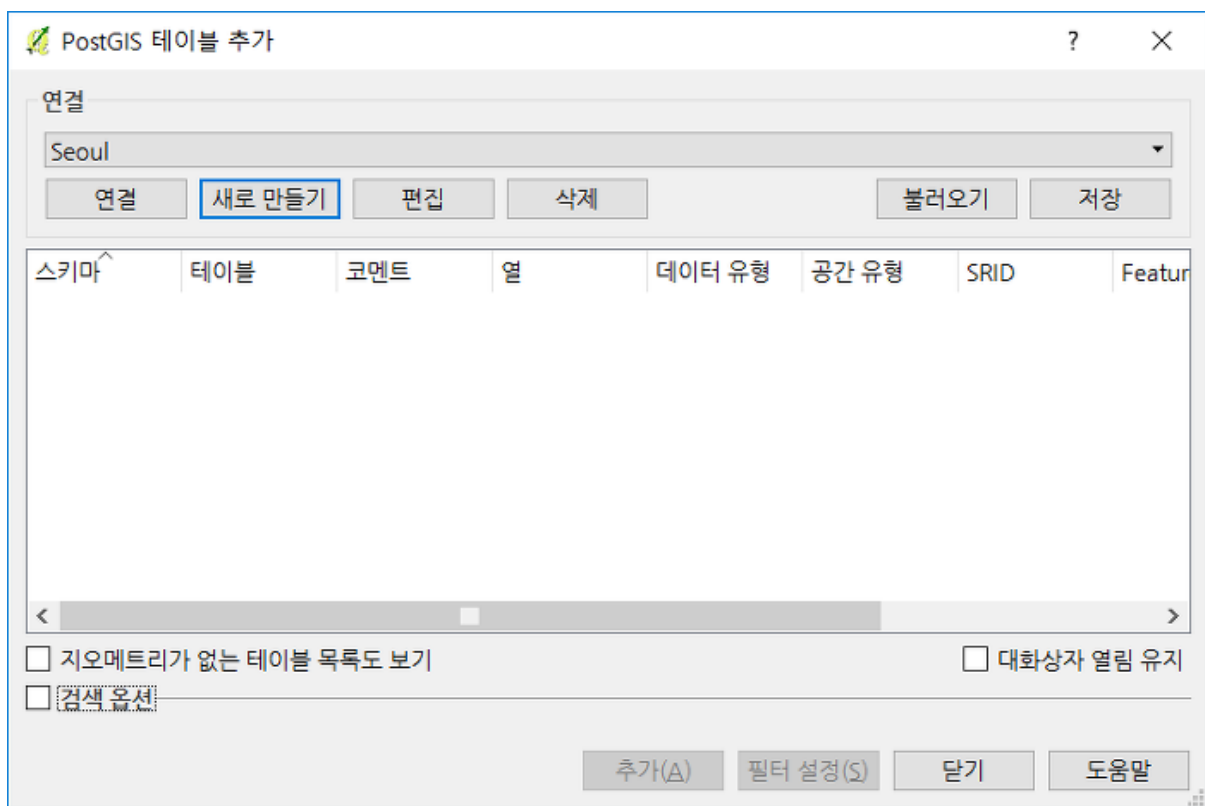




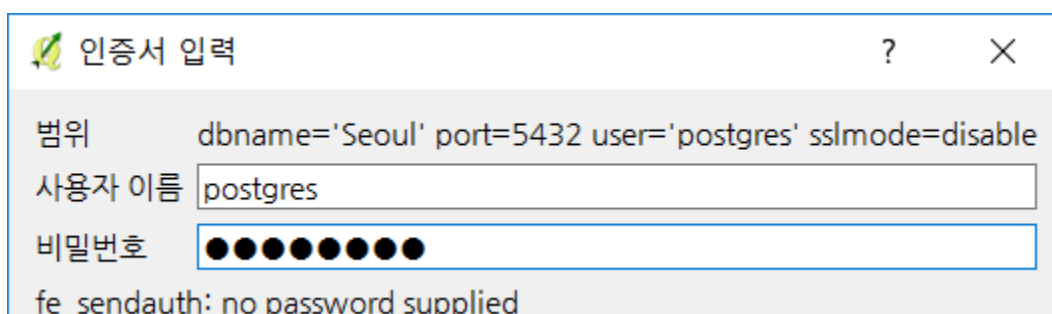
참고로 비밀번호 저장을 체크할 경우에는 아래와 같은 안내 메시지가 표시됩니다.

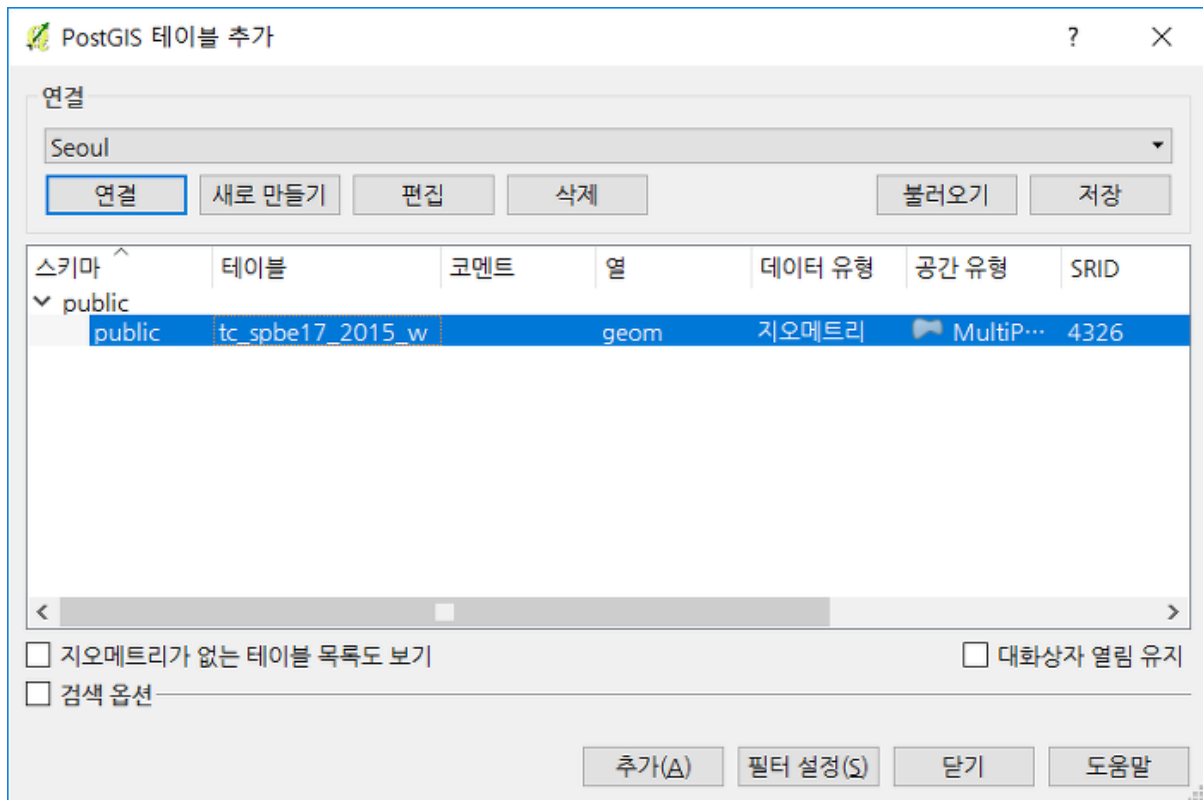


이제 아래와 같이 Seoul DB 가 목록에 추가되었는데요, '연결' 버튼을 클릭해 보겠습니다.

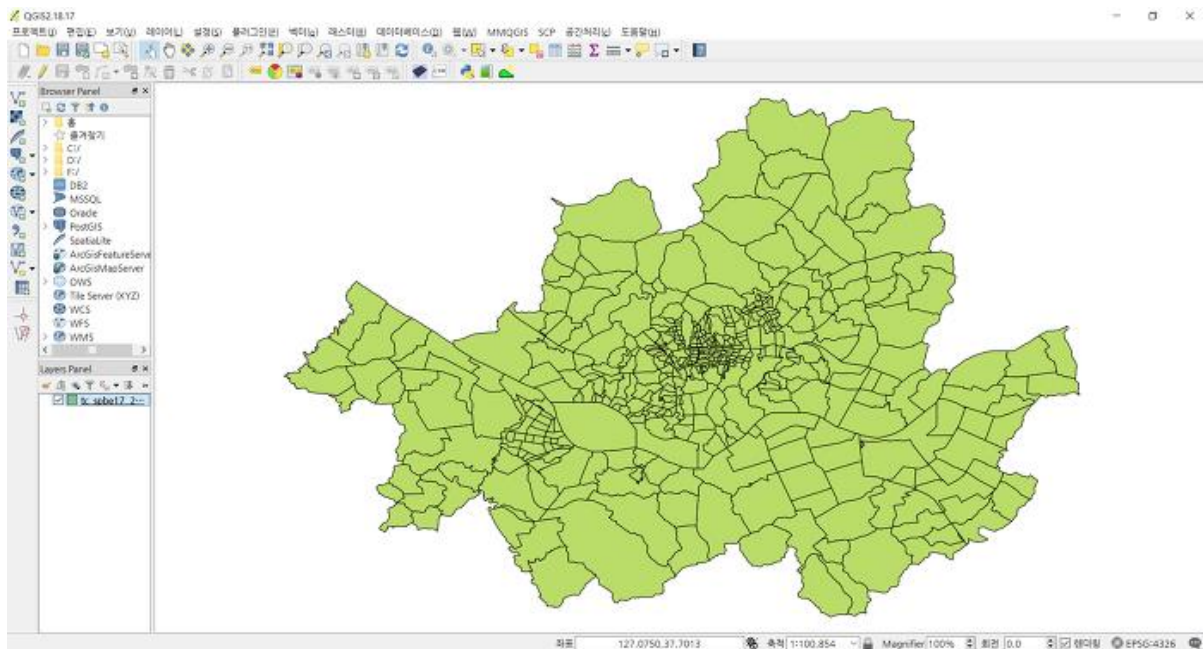


비밀번호까지 입력하고 나면 레이어 목록이 표시됩니다.





추가 버튼을 클릭하면 아래와 같이 DB에 저장되어 있는 PostGIS 레이어가 표시됩니다.



## [4] PostgreSQL/PostGIS 에서 SQL 언어 학습하기

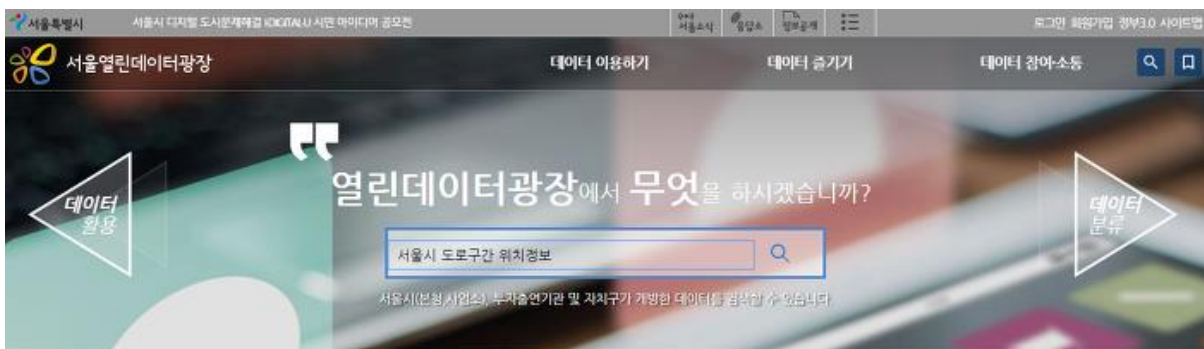
안녕하세요? 앞서 QGIS 에서 PostGIS 레이어를 추가해봤는데요, 이번 글에서는 SQL 쿼리를 통해 공간데이터를 다루어보도록 하겠습니다.

### SQL 이란?

SQL 은 관계형 데이터베이스 관리 시스템의 데이터를 관리하기 위해 설계된 프로그래밍 언어로, 구조화 쿼리 언어(Structured Query Language)라고도 합니다. SQL 을 통해서 우리는 데이터에 대해서 질문을 던져볼 수 있고, 데이터를 실제로도 다뤄볼 수 있습니다.

이번 글에서는 기본적인 명령문 수행을 위해서 SQL 을 어떻게 이용하는지 정리해보고자 합니다.

서울시 도로구간 위치정보(좌표계: WGS1984) 데이터셋으로 학습해보겠습니다. 공간데이터를 획득하기 위해 서울열린데이터광장(<http://data.seoul.go.kr/>)에 접속하고 '서울시 도로구간 위치정보'를 검색해보겠습니다.



아래 '서울시 도로구간 위치정보 (좌표계:WGS1998)' 데이터셋에서 MAP 버튼을 클릭합니다.

### 데이터셋 (2건)

결과 더보기 +

<div>원천 SHEET MAP FILE OPEN API</div> <div>서울시 도로구간 위치정보 (좌표계: WGS1984)</div> <div>&lt;일반행정 &gt; 행정</div> <div>제공기관 : 서울특별시 수정일 : 2014-10-09</div>	<div>원천 SHEET MAP FILE OPEN API</div> <div>서울시 도로구간 위치정보 (좌표계: ITRF2000)</div> <div>&lt;일반행정 &gt; 행정</div> <div>제공기관 : 서울특별시 수정일 : 2014-10-09</div>
--	---

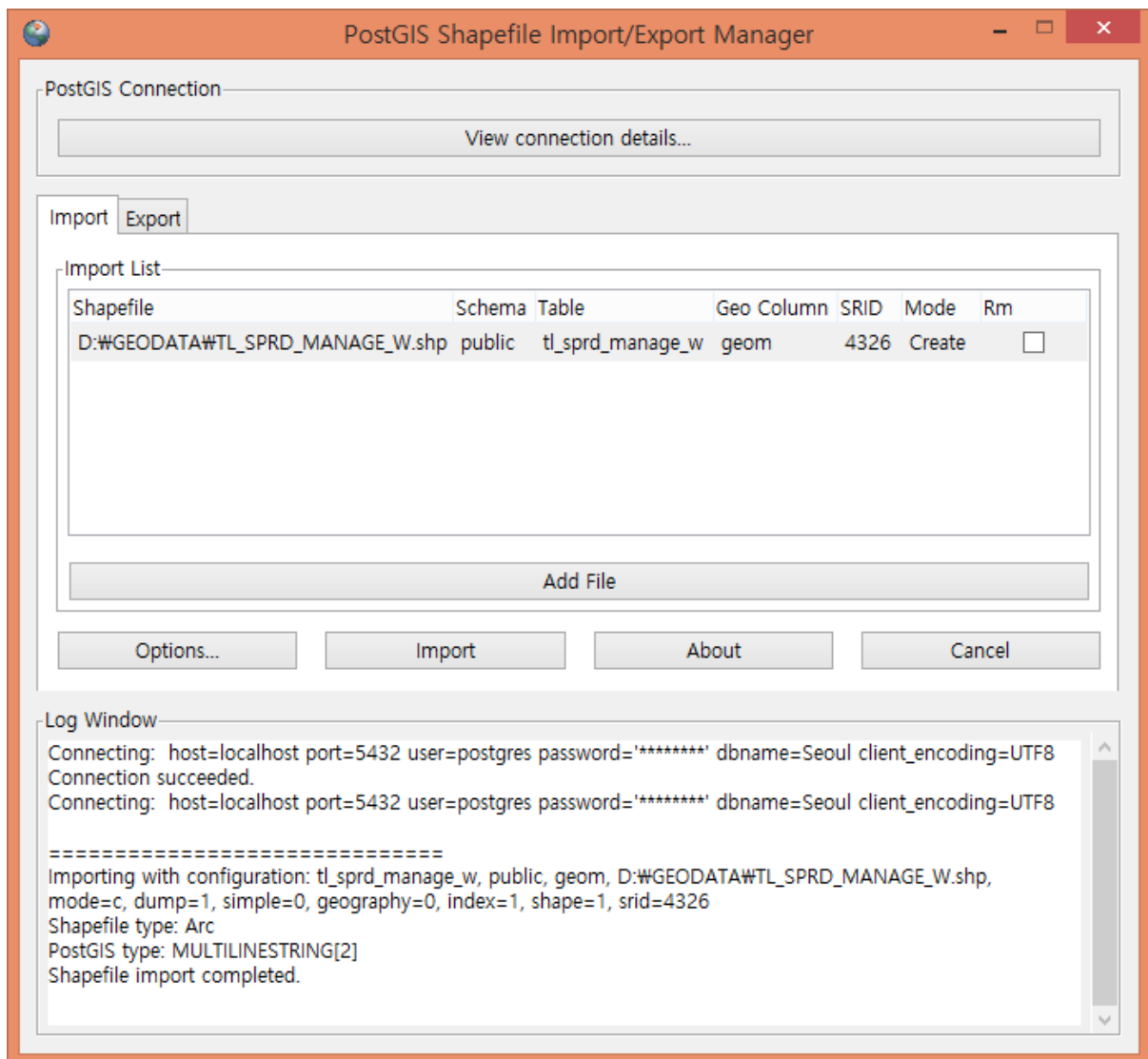
SHP 파일목록의 'TL\_SPRD\_MANAGE\_W\_SHP.zip'을 다운로드하고 압축을 해제합니다.

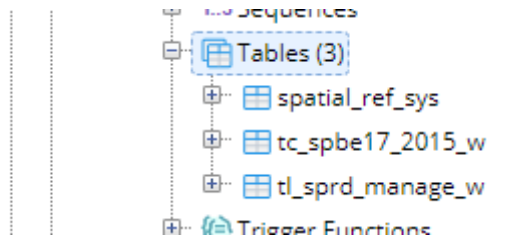
Sheet	Open Api	Map	File
-------	----------	-----	------

#### SHP 파일목록

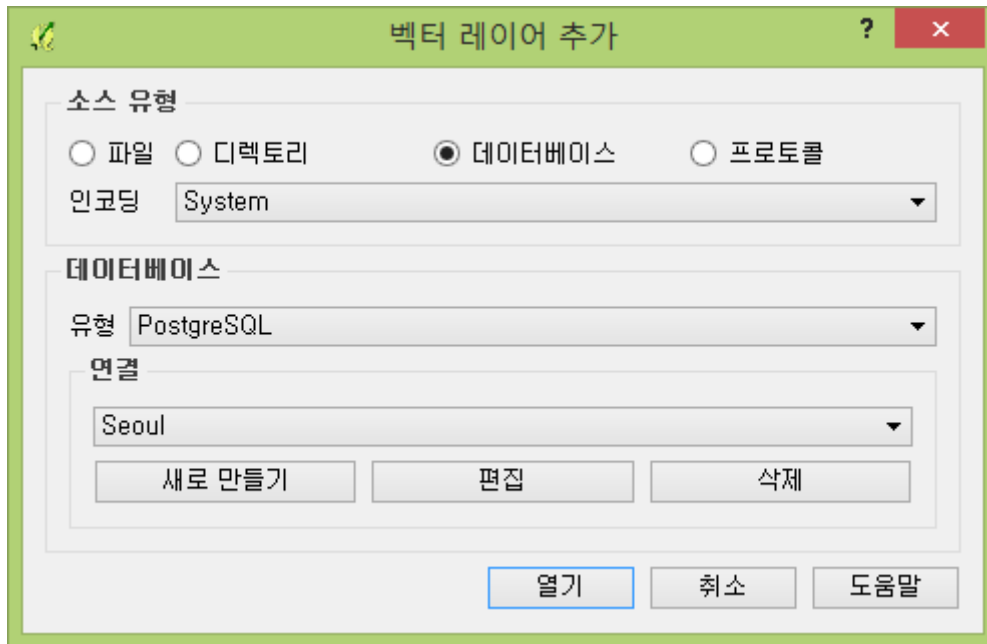
NO	파일명	파일크기(KB)	마지막수정일	최초공개일	다운로드
1	TL_SPRD_MANAGE_W_SHP.zip	15413429	2014.10.15	2014.10.15	DOWN

이전 글을 참고하여 Seoul DB 에 TL\_SPRD\_MANAGE\_W\_SHP 셰이프파일을 추가합니다.

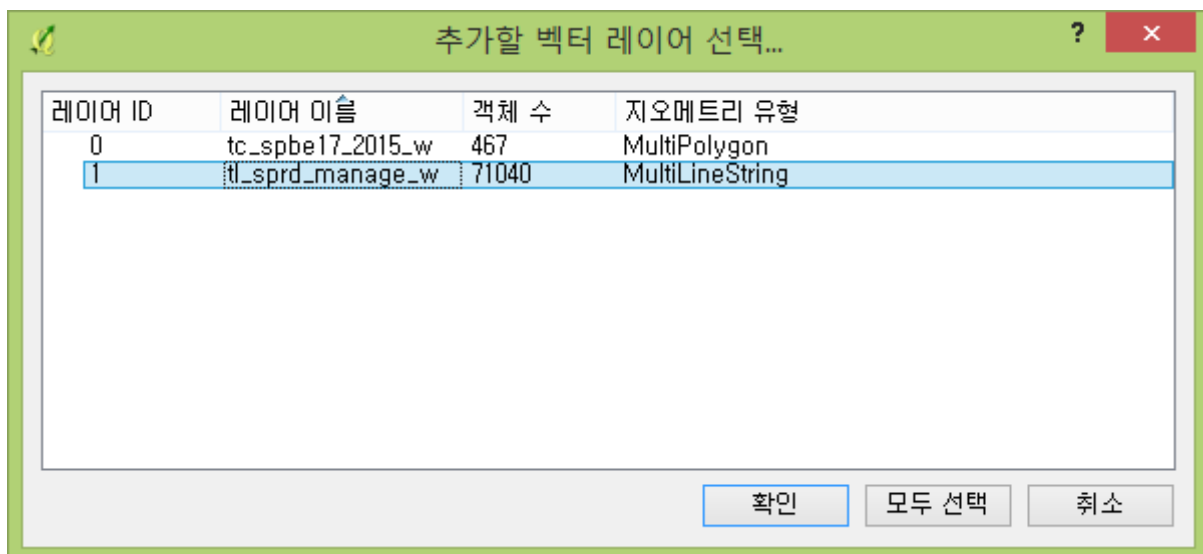




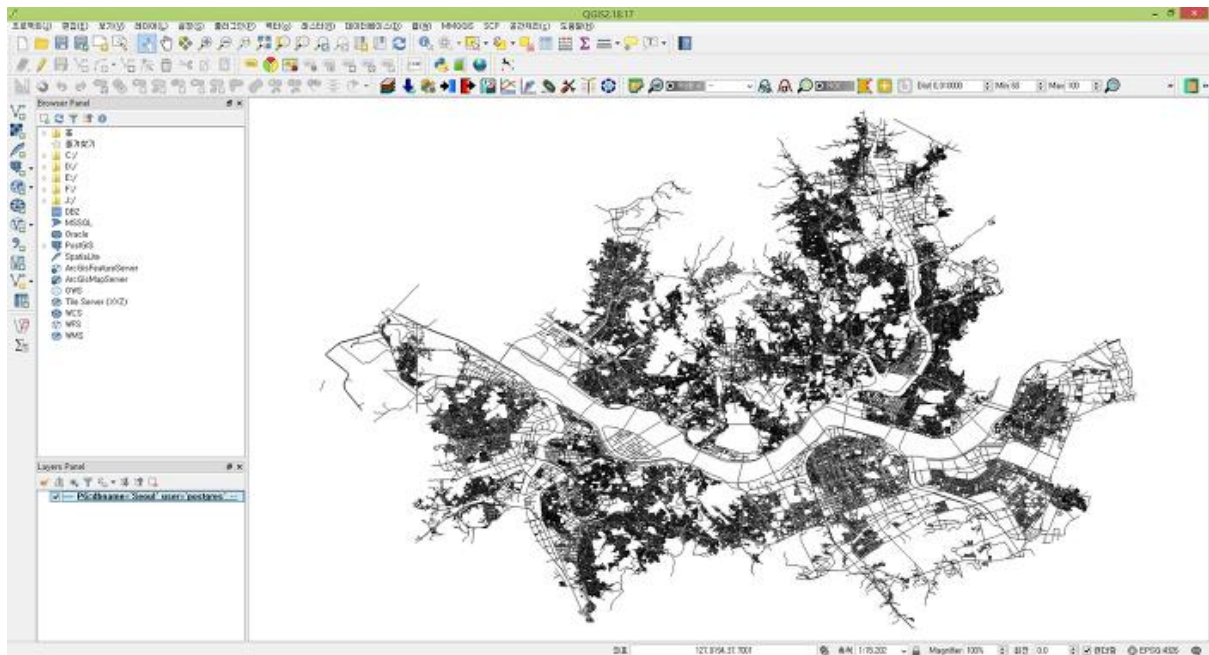
해당 파일을 QGIS 에서 레이어 추가해 보겠습니다. QGIS 2.18 에서 '벡터 레이어 추가'를 사용하실 경우엔, 소스 유형은 데이터베이스, 데이터베이스 유형은 PostgreSQL, 연결은 Seoul 로 선택해주시면 되겠습니다.



'추가할 벡터 레이어 선택' 창에서 해당 도로 레이어를 선택합니다.



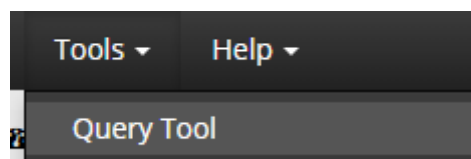
아래와 같이 서울시 도로구간 레이어가 추가되었습니다.



이제 pgAdmin 에서 SQL 구문을 시작해보겠습니다.



참고로, Query Tool 은 pgAdmin 상단 메뉴에서 'Tools > Query Tool'를 선택하시면 됩니다.



'Select'은 가장 기본적인 SQL 쿼리문입니다. 특정 테이블에서 특정 조건에 기반한 데이터를 가지고 오고 싶을 때 사용합니다

도로 데이터셋에서 모든 데이터를 불러오겠습니다. 도로 데이터셋의 테이블 이름은 'tl\_sprcd\_manage\_w'로 설정되어 있습니다.

아래와 같은 쿼리를 작성하고 실행 버튼 혹은 F5 를 누르겠습니다. \*(별표)는 모든 데이터를 불러오겠다는 뜻입니다.

```
Seoul on postgres@PostgreSQL 10
1 select * from tl_sprd_manage_w
```

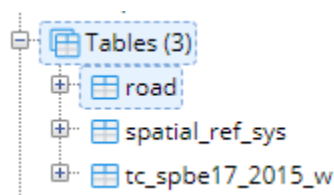
짠! 데이터의 테이블이 불러와졌습니다.

Seoul on postgres@PostgreSQL 10								
1 select * from tl_sprd_manage_w								
Data Output Explain Messages Query History								
	gid	sig_cd	rds_man_no	rn	rn_cd	eng_rn	ntfc_de	rn_dlb_de
	integer	character varying (5)	integer	character varying (80)	character varying (7)	character varying (80)	character varying (8)	character vary
1	1	11110	1188	율곡로6길	4100239	Yulgok-ro 6-gil	20100702	20100623
2	2	11110	2581	자하문로2길	4100282	Jahamun-ro 2-gil	20100702	20100623
3	3	11110	2215	경향34길	4100507	Pyeongchang 34-gil	20100702	20100623
4	4	11110	2216	평창36길	4100508	Pyeongchang 36-gil	20100702	20100623

이 테이블 상의 데이터를 다뤄 보겠습니다. 그런데 테이블 명이 너무 기니, 테이블 명을 간단하게 바꿔보겠습니다. 다음과 같은 쿼리를 입력합니다. tl...로 시작하는 긴 테이블 명을 간단히 'road'로 바꾸어주겠습니다.

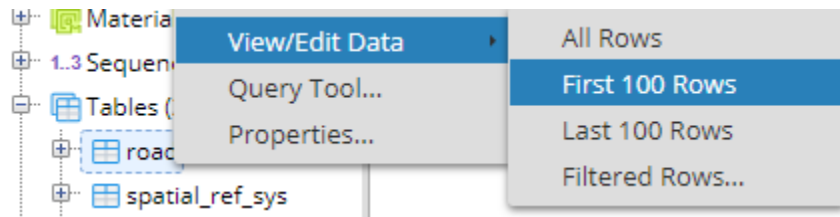
```
Seoul on postgres@PostgreSQL 10
1 alter table tl_sprd_manage_w rename to road
```

아래와 같이 테이블 명이 road 로 변경되었습니다.



해당 테이블의 데이터를 보거나 편집할 때는 테이블 명을 우클릭한 후, 'View/Edit Data' 하부 메뉴를 선택하시면 됩니다. 첫번째 100 줄을 확인해 보겠습니다.





아래와 같이 road 데이터 100 줄이 표시됩니다. 여기서 road\_lt 컬럼은 '도로 길이'를 나타냅니다.

Seoul on postgres@PostgreSQL 10

```

1 SELECT * FROM public.road
2 ORDER BY gid ASC LIMIT 100
3

```

Data Output Explain Messages Query History

p_cn	rep_cn	road_bt	road_lt	bsl_int	nir_cd_no
character varying (80)	character varying (80)	integer	integer	character varying (5)	character varying (13)
니동 14-4	운니동 84-1	2	64	10	[null]
의동 37-4	통의동 42-8	1	11	10	[null]
차동 463	평창동 447	8	305	20	[null]

avg 쿼리문을 사용하여 도로 길이의 평균값을 구해 보겠습니다. 짠! 서울시 도로 길이의 평균값을 구했습니다. 속도가 정말 빠르죠?! 같은 방식으로 합계나 특정 비율을 구할 수도 있습니다.

Seoul on postgres@PostgreSQL 10

```

1 select avg(road_lt) from road

```

Data Output Explain Messages Query History

	avg
	numeric
1	152.1337697072072072


다음 질의문을 학습해 보겠습니다. 이번에는 폭이 5m 미만인 도로의 이름과 폭을 산출해 보겠습니다. 데이터에서 rn 칼럼은 도로명, road\_bt 칼럼은 도로폭을 나타냅니다.

rn character varying (80)	road_bt integer
을곡로6길	2
자하문로2길	1
평창34길	8
평창76길	8

쿼리문은 다음과 같습니다. where 는 가장 쓰임새가 많은 SQL 쿼리 중 하나입니다.

Seoul on postgres@PostgreSQL 10	
1	<code>select rn, road_bt from road where road_bt &lt; 5</code>

자, 결과가 추출되었습니다. 폭이 5m 미만인 도로들이 선택되었습니다.

 Seoul on postgres@PostgreSQL 10

1 `select rn, road_bt from road where road_bt < 5`

Data Output Explain Messages Query History			
	rn character varying (80)	road_bt integer	
1	을곡로6길	2	
2	자하문로2길	1	
3	진흥로22길	2	
4	지호루21길	2	

이제 SQL 쿼리문으로 공간데이터를 다루실 수 있겠죠?!

참고할 만한 기본적인 쿼리문 몇 가지를 소개합니다.

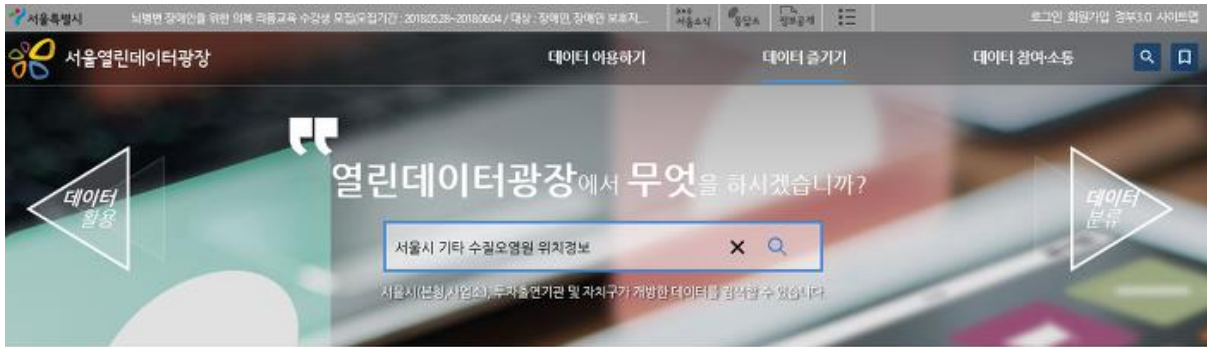
=	Qqual to
<>	Not Equal to
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
between	Range between two value
like	Match a character pattern
as	Change the name a field name

Select	Will return data based on the query
*	Return all
Alter table	Changes some aspect of the table, in our example the name
Rename to	Rename the table's name, to be used with the Alter table statement
Avg	Calculate and return the average of a given column
Group By	Group the result-set by one or more columns
Where	Compare the predicate and return only the rows that are

## [5] PostgreSQL/PostGIS 에서 투영법 변환하기

이번 글에서는 PostgreSQL/PostGIS 에서 투영법을 변환하는 방법을 학습해 보도록 하겠습니다.

일단, 서울열린데이터광장(<http://data.seoul.go.kr>)에서 '서울시 기타 수질오염원 위치정보'를 검색합니다.



아래와 같이 2 종 좌표계로 된 데이터셋이 제공되는데요, 이번에는 ITRF2000 좌표계 파일을 선택해 보겠습니다.

### 데이터셋 (2건)

결과 더보기 +

원천
SHEET
MAP
FILE
OPEN API

서울시 기타 수질오염원 위치정보 (좌표계: WGS1984)  
<환경> 수질환경  
제공기관: 서울특별시 수정일: 2015-08-10

원천
SHEET
MAP
FILE
OPEN API

서울시 기타 수질오염원 위치정보 (좌표계: ITRF2000)  
<환경> 수질환경  
제공기관: 서울특별시 수정일: 2014-11-07

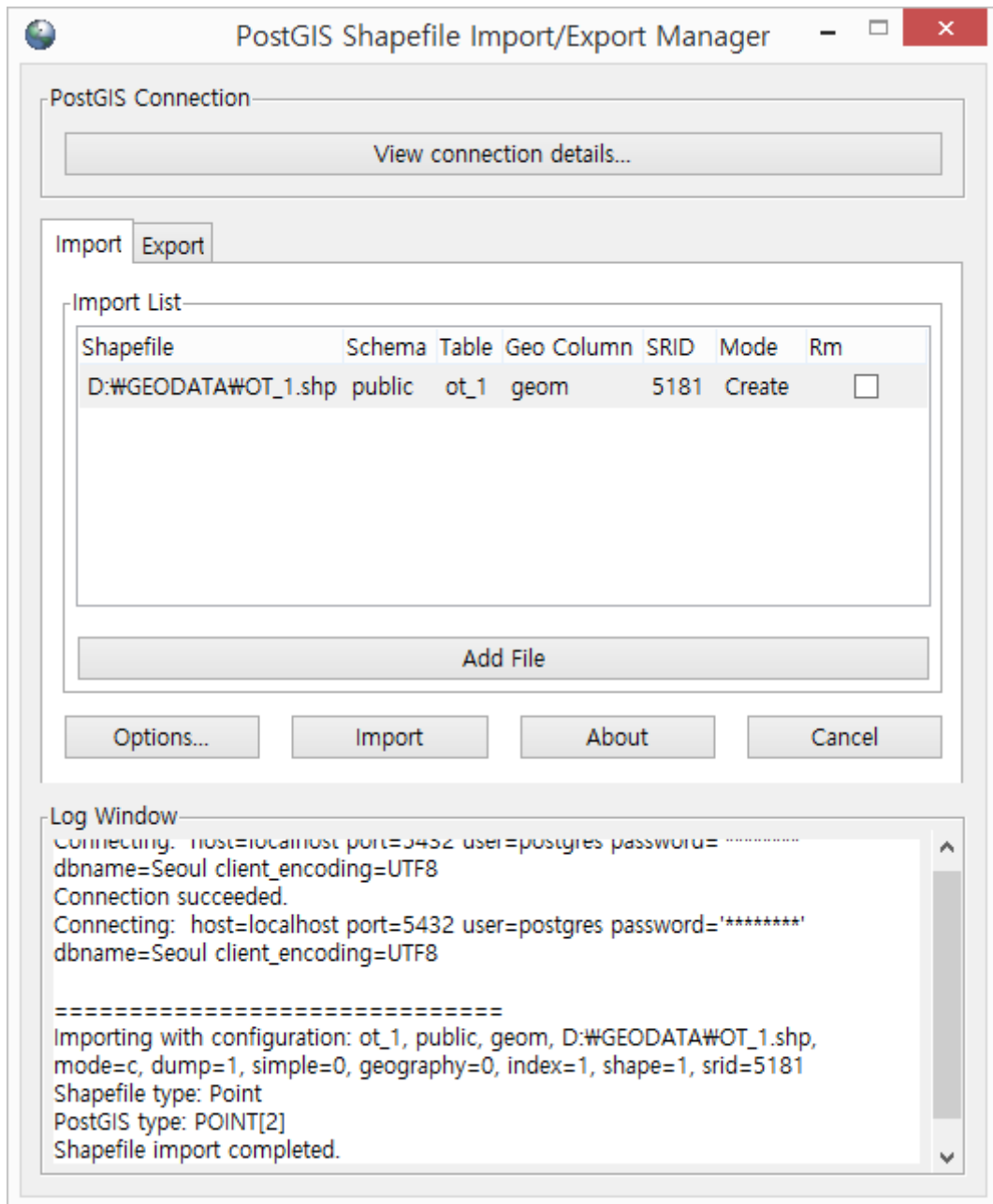
MAP 버튼을 클릭하고 OT\_1\_SHP.zip 파일을 다운로드하고 압축을 해제합니다.

Sheet	Open Api	Map	File
-------	----------	-----	------

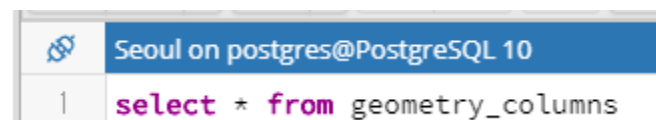
SHP 파일목록

NO	파일명	파일크기(KB)	마지막수정일	최초공개일	다운로드
1	OT_1_SHP.zip	362468	2014.11.07	2014.11.07	DOWN

PostGIS Shapefile Import/Export Manager 를 통해 해당 공간데이터를 임포트합니다. 이때 SRID 는 5181 로 설정합니다.



자, 아래 쿼리를 통해 현재 PostgreSQL 에 추가된 데이터 목록을 확인해 보겠습니다.



앞서 2 개 추가한 공간데이터(읍면동, 도로)는 SRID 가 4326, 방금 추가한 공간데이터(기타 수질오염원)의 SRID 는 5181 인 것을 확인할 수 있습니다.

공간쿼리를 적용하기 위해서는 데이터들이 서로 같은 좌표계를 가지고 있어야 합니다. 따라서 ot\_1 투영법을 4326 으로 변환해 보도록 하겠습니다.

Seoul on postgres@PostgreSQL 10

```
1 select * from geometry_columns
```

Data Output Explain Messages Query History

	f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
1	Seoul	public	tc_spbe17_2015...	geom	2	4326	MULTIPOLYGON
2	Seoul	public	road	geom	2	4326	MULTILINESTRING
3	Seoul	public	ot_1	geom	2	5181	POINT

좌표계를 변환하는 쿼리는 아래와 같습니다. 쿼리문이 길어질 때는 아래와 같이 각 파트를 새 줄로 작성하는 것이 좋습니다. 작성 후에는 아래 줄에서 'F5' 버튼을 실행하시면 됩니다.

Seoul on postgres@PostgreSQL 10

```
1 alter table ot_1
2 alter column geom type geometry(POINT, 4326)
3 using ST_Transform(geom, 4326)
```

alter table 은 ot\_1 데이터의 변경을 선언하며, alter column 은 변경할 컬럼을 정확히 지시합니다. geom 은 모든 공간데이터를 지시하는 컬럼이며, type 은 처리할 데이터의 지오메트리 유형을 정의해 줍니다.

using은 해당 컬럼에 적용할 명령어를 정의하며, 여기서는 정의된 SRID에 기반한 새 지오메트리를 반환합니다.

자, 다시 데이터 목록을 확인해 보면 ot\_1 의 SRID 가 4326 으로 변경된 것을 확인하실 수 있습니다.

Seoul on postgres@PostgreSQL 10

```
1 select * from geometry_columns
```

Data Output Explain Messages Query History

	f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
1	Seoul	public	tc_spbe17_2015...	geom	2	4326	MULTIPOLYGON
2	Seoul	public	road	geom	2	4326	MULTILINESTRING
3	Seoul	public	ot_1	geom	2	4326	POINT

## [6] PostGIS 에서 공간쿼리(Spatial Query) 사용하기

안녕하세요? 앞서 PostGIS 에서 투영법 변환을 학습해 보았습니다.  
이번 글에서는 PostGIS 에서 공간쿼리(Spatial Query)를 사용해보겠습니다.

이전에 GIS 관련 경험이 있으신 분은 이번 시간의 개념을 쉽게 이해할 수 있을 텐데요, 이번 시간엔 GIS 분석을 새로운 방식으로 해볼 수 있다는 점에서 의미가 있습니다. 먼저, 공간쿼리를 이용하여 지오메트리의 값을 분석해보겠습니다.

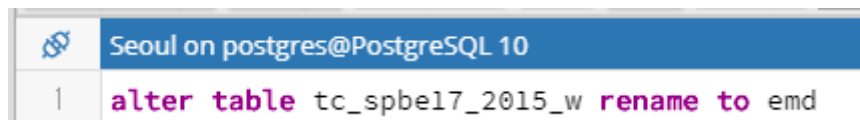
### 지오메트리(geometry)란?

모양, 크기, 위치 등과 같은 도형의 정보와 속성을 뜻합니다. GIS 에서 지오메트리는 점(point), 선(line), 면(polygon)으로 구성됩니다.

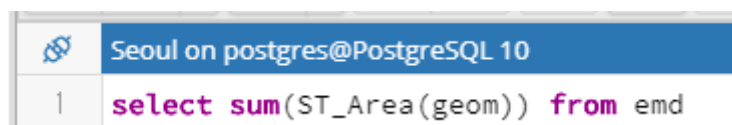
아래와 같이 앞서 실습을 통해 서울시의 점, 선, 면 데이터를 DB 에 등록한 상태입니다.

	f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
1	Seoul	public	tc_spbe17_2015...	geom	2	4326	MULTIPOLYGON
2	Seoul	public	road	geom	2	4326	MULTILINESTRING
3	Seoul	public	ot_1	geom	2	4326	POINT

일단, '서울시 법정구역 읍면동 공간정보' 데이터셋의 이름을 emd 로 변경하겠습니다.



emd 테이블을 우클릭하고 'Query Tool'을 실행합니다. 아래 쿼리는 읍면동(emd) 테이블에 있는 서울 읍면동 면적의 합을 구하는 함수입니다.



짠! 면적의 합이 계산되어 나왔습니다. 그런데 단위가 너무 작죠?! 왜 그럴까요?!

	sum double precision
1	0.0617593379523809

그 이유는 테이블의 SRID 가 위경도로 된 지리 좌표계로 정의(4326)되어 있기 때문입니다.

Seoul on postgres@PostgreSQL 10

1 select \* from geometry\_columns

Data Output Explain Messages Query History

	f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
1	Seoul	public	emd	geom	2	4326	MULTIPOLYGON
2	Seoul	public	road	geom	2	4326	MULTILINESTRING
3	Seoul	public	ot_1	geom	2	4326	POINT

PostGIS 에서는 지리 좌표계 정의 상태에서도 제곱미터 단위의 면적을 구할 수 있습니다. 여기서는 이전 실습을 복습하는 차원에서 아래와 같은 방식으로 좌표계를 5186 으로 변환하겠습니다.

Seoul on postgres@PostgreSQL 10	
1	<code>alter table emd</code>
2	<code>alter column geom type geometry(MULTIPOLYGON, 5186)</code>
3	<code>using ST_Transform(geom, 5186)</code>

자, 다시 면적을 계산해볼까요?!

'ST'는 'Spatial Temporal'의 약어인데요, 'ST'로 시작하는 쿼리는 공간적 쿼리의 기능을 합니다. 여기서 'ST\_Area'는 지오메트리의 면적을 계산하며, sum 은 면적의 합을 계산하는데 사용됩니다.

Seoul on postgres@PostgreSQL 10	
1	<code>select sum(ST_Area(geom)) from emd</code>

아래와 같이 서울시 면적이 605 제곱 킬로미터로 계산됩니다.

Data Output Explain M	
	sum double precision
1	605696949.022324



그렇다면 읍면동별 면적을 구하고 싶을 때는 어떻게 해야 할까요?! 일단 읍면동 명칭은 emd\_nm 컬럼이 가지고 있습니다.

	gid [PK] integer	emd_cd character varying (10)	emd_nm character varying (40)	emd_eng_nm character varying (40)	esri_pk bigint	shape_area numeric	shape_len numeric	geom geometry
1	1	11410103	합동	Hap-dong	5	10433157684573	100552793686	010600002...
2	2	11650103	우면동	Umyeon-dong	26	10421539261493	855573202793	010600002...
3	3	11170118	원효로4가	Wonhyoro 4(sa)-ga	35	10315954778309	299142939701	010600002...
4	4	11140130	남산동2가	Namsan-dong 2(sa)-ga	45	10329532392103	703288958392	010600002...

각 읍면동의 면적을 구하고, 읍면동의 이름을 면적에 따른 내림차순(Descending order)으로 정렬하는 공간쿼리문입니다.

```
Seoul on postgres@PostgreSQL 10
1 select emd_nm, ST_Area(geom) from emd order by shape_area DESC
```

아래와 같이 결과가 잘 나왔습니다!

서울가 읍면동 중 신림동의 면적이 가장 넓고, 상계동이 그 뒤를 따르고 있습니다.

궁금한 정보를 쿼리문으로 작성하여 추출해낼 수 있다는 것이 PostGIS 공간쿼리의 매력입니다.

Seoul on postgres@PostgreSQL 10		
1	select emd_nm, ST_Area(geom) from emd order by shape_area DESC	
Data Output Explain Messages Query History		
	emd_nm character varying (40)	st_area double precision
1	신림동	18138088.5400143
2	상계동	15844703.7758548
3	진관동	11565047.1605112
4	도곡동	9522247.20950000

면(polygon)으로 된 데이터를 다루어봤으니, 이번엔 선(line) 데이터로 공간쿼리 기능을 연습해보겠습니다. 지난 시간에 다뤘던 '서울시 도로구간 위치정보' 데이터셋을 이용하겠습니다. 마찬가지로 투영법을 변경합니다.

```
Seoul on postgres@PostgreSQL 10
1 alter table road
2 alter column geom type geometry(MULTILINESTRING, 5186)
3 using ST_Transform(geom, 5186)
```

해당 road 테이블은 'rn' 컬럼에 도로 이름을 저장하고 있습니다.

	gid [PK] Integer	sig_cd character varying (5)	rds_man_no Integer	rn character varying (80)	rn_cd character varying (7)	eng_rn character varying (80)	ntfc_de character varying (8)	rn_dlb_de character varying (8)
1	1	11110	1188	율곡로6길	4100239	Yulgok-ro 6-gil	20100702	20100623
2	2	11110	2581	자하문로2길	4100282	Jahamun-ro 2-gil	20100702	20100623
3	3	11110	2215	평창34길	4100507	Pyeongchang 34-gil	20100702	20100623

'ST\_Length' 공간쿼리 기능을 이용하면 선의 길이를 구할 수 있습니다. 입력해볼까요?!

road 테이블 안에서 ST\_Length 쿼리기능으로 도로 길이를 구하고, rn 을 길이에 따른 내림차순(order by length DESC)으로 결과를 요청했습니다.

```
Seoul on postgres@PostgreSQL 10
1 select rn, ST_Length(geom) as length from road order by length DESC
```

결과는 아래와 같습니다. 경부고속도로가 가장 길고, 그 다음은 마포나루 길이 긴 것으로 나타납니다.

	rn character varying (80)	length double precision
1	경부고속도로	11360.3920853626
2	마포나루길	9204.91652238451
3	강변북로	8872.37407656553

공간쿼리를 통해서 이런 식으로 원하는 지오메트리 값을 도출할 수 있습니다. 다만, 도로 데이터같은 경우엔 길이 여러 개 겹쳐져 있는 경우도 있기 때문에 정확한 분석을 위해서는 데이터 구조의 확인이 중요할 것으로 보입니다.

다음 시간에는 PostGIS 에서 공간관계(Spatial Relationship)를 분석해 보겠습니다.

## [7] PostGIS 에서 공간관계(Spatial Relationship) 분석하기

안녕하세요? 이전 글에서는 하나의 특정 데이터셋에서 공간 데이터를 추출하는 방법을 정리해 봤습니다. 이번 글은 두 개의 다른 데이터셋에서 공간관계(Spatial Relationship)를 추출하는 방법을 학습해 보겠습니다.

다음 3 가지 관계 유형을 배워 보겠습니다: Intersect, Contains, Within.

Intersects 함수는 데이터의 지오메트리를 체크하고 서로 중첩되는 것만을 반환합니다. 예를 들면, 읍면동과 중첩되는 도로는 아래 쿼리를 통해 확인할 수 있습니다.

일단 도로와 읍면동 테이블이 표시되어야 할 텐데요, 아래 r 과 e 는 각 테이블에 대한 별칭(Alias)입니다. 이러한 별칭은 키입력을 줄여주며 보다 정리된 쿼리를 작성할 수 있습니다.

```
Seoul on postgres@PostgreSQL 10
1 select r.rn, e.emd_nm
2 from road as r, emd as e
3 where e.emd_nm = '연남동' AND ST_Intersects(r.geom, e.geom)
```

여기서는 읍면동 명칭이 '연남동'이면서 도로와 읍면동이 중첩되는 레코드를 지시하였습니다. 따라서, 결과는 아래와 같이 연남동과 중첩되는 도로 명칭이 계산됩니다. 생각보다 간단하죠?!

Data Output			Explain	Messages	Query History
	rn	emd_nm			
	character varying (80)	character varying (40)			
1	연희로	연남동			
2	연희로	연남동			
3	연희로	연남동			
4	도곡로	연남동			

이번에는 Within 에 대해 학습해 보겠습니다. Within 은 주어진 지오메트리가 또다른 지오메트리 내부에 위치하는지를 확인합니다. 여기서는 기타 수질오염원이 어느 읍면동에 속하는지 질의해 보도록 하겠습니다.

```

Seoul on postgres@PostgreSQL 10
1 select o.name, e.emd_nm
2 from ot_1 as o, emd as e
3 where ST_Within(o.geom, e.geom)

```

쿼리문은 틀린 부분이 없는데 쿼리 결과가 표시되지 않습니다. 왜 그럴까요?!

Data Output	Explain	Messages	Query History
	<b>name</b> character varying (38)	<b>emd_nm</b> character varying (40)	

그 이유는, 아래와 같이 기타 수질오염원과 읍면동 데이터셋이 서로 다른 SRID 로 정의되어 있기 때문입니다.

Seoul on postgres@PostgreSQL 10

1 select \* from geometry\_columns

Data Output

Explain

Messages

Query History

	f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
1	Seoul	public	emd	geom	2	5186	MULTIPOLYGON
2	Seoul	public	road	geom	2	4326	MULTILINESTRING
3	Seoul	public	ot_1	geom	2	4326	POINT


기타 수질오염원 테이블을 5186 좌표계로 정의한 후, 다시 실행해 보겠습니다.

```

Seoul on postgres@PostgreSQL 10
1 alter table ot_1
2 alter column geom type geometry(POINT, 5186)
3 using ST_Transform(geom, 5186)

```

이제 아래와 같이 각 읍면동에 포함되는 기타 수질오염원 명칭을 확인할 수 있습니다.


Seoul on postgres@PostgreSQL 10

```

1  select o.name, e.emd_nm
2  from ot_1 as o, emd as e
3  where ST_Within(o.geom, e.geom)

```

Data Output			Explain	Messages	Query History
	name	emd_nm			
	character varying (38)	character varying (40)			
1	(주)우림자동차	우면동			
2	이병천스튜디오	우면동			
3	(주)현대모터스	우면동			
4	한사랑이매거진	의정부시			

위 쿼리문은 아래와 같이 읍면동 면적을 추가하고, 해당 면적을 기준으로 한 내림차순 결과를 반환할 수도 있습니다.

Seoul on postgres@PostgreSQL 10

```
1 select o.name, e.emd_nm, e.shape_area
2 from ot_1 as o, emd as e
3 where ST_Within(o.geom, e.geom)
4 order by e.shape_area DESC
```

Data Output Explain Messages Query History

	name character varying (38)	emd_nm character varying (40)	shape_area numeric
1	고성만내과의원	신림동	0.00184727440060
2	연세양지가정의원	신림동	0.00184727440060
3	WIDE PHOTO	신림동	0.00184727440060
4	제이사진과	신림동	0.00184727440060

위 shape\_area 컬럼은 지리좌표계로 계산된 결과인데요, 아래 쿼리문을 통해 해당 테이블의 정보를 갱신할 수도 있습니다.

```
Seoul on postgres@PostgreSQL 10
1 update emd set shape_area = ST_Area geom
```

결과는 아래와 같습니다.

	name character varying (38)	emd_nm character varying (40)	shape_area numeric
1	정우등물병원	신림동	18138088.5400143
2	중앙성심의원	신림동	18138088.5400143
3	명진칼라	신림동	18138088.5400143
4	사미칼라현상소	신림동	18138088.5400143

ST\_DWithin 함수는 사용자가 입력한 거리값에 따라 지오메트리 A 와 지오메트리 B 를 비교합니다. 함수의 일반적인 형식은 ST\_DWithin(geometry A, geometry B, radius)입니다. 여기서 반경은 사용하는 데이터와 측정 단위가 같아야 합니다.

모든 기타 수질오염원으로터 100m 반경 내에 있는 읍면동을 확인해 보겠습니다.

```
Seoul on postgres@PostgreSQL 10
1 select o.name as station, e.emd_nm as boundary
2 from emd as e, ot_1 as o
3 where ST_DWithin(e.geom, o.geom, 100)
4 order by station ASC
```

적용 결과는 아래와 같습니다.

	station character varying (38)	boundary character varying (40)
1	(북)에스엠씨치과의원	한남동
2	(사)국제사랑의봉사단사랑...	논현동
3	(사)국제사랑의봉사단사랑...	역삼동
4	사미칼라현상소	신림동

이번에는 기타 수질오염원 100m 반경 내 읍면동의 면적 합계를 분석해 보겠습니다. 하나의 기타 수질오염원(100m 버퍼)은 다수의 읍면동을 포함하고 있습니다. 아래 쿼리에서는 이것을 기타 수질오염원으로 그룹화하고 sum(e.shape\_area)를 통해 면적 합계를 산출하고 있습니다.

```

Seoul on postgres@PostgreSQL 10
1 select o.name as station, sum(e.shape_area) as area
2 from emd as e, ot_1 as o
3 where ST_DWithin(e.geom, o.geom, 100)
4 group by o.name
5 order by sum(e.shape_area) ASC

```

아래와 같이 수질오염원 별로 100m 버퍼 내 포함되는 읍면동 면적 합계를 산출한 결과입니다.

	station character varying (38)	area numeric
1	프리골드	97334.7732855499
2	코지	101466.2384010964
3	금토	102622.1778976061
4	여지	102622.1778976061

보다 다양한 공간관계 기능은 PostGIS 문서를 참조하시면 되겠습니다.

8.9. Spatial Relationships and Measurements:

[http://postgis.net/docs/reference.html#Spatial\\_Relationships\\_Measurements](http://postgis.net/docs/reference.html#Spatial_Relationships_Measurements)