

공간 SQL을 이용한 공간자료분석 기초실습

2017.12.08



장병진 (jangbi882@gmail.com)



교재 및 실습자료 다운로드

- 교재 다운로드

- <https://gaia3d.github.io/workshop>
- 서울연구원 공간 SQL을 이용한 공간자료분석 기초실습

- 설치파일

- PostgreSQL 9.5
 - <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads#windows>
- PostGIS 2.3
 - Stack builder 통해 설치 권장
 - 혹은 <https://winnie.postgis.net/download/windows/pg95/buildbot/>
- QGIS
 - <http://qgis.org/downloads/QGIS-OSGeo4W-2.18.14-1-Setup-x86.exe>

- 실습자료 다운로드

- https://github.com/Gaia3D/workshop/raw/master/20171208_%EC%84%9C%E C%9C%A8%EC%97%B0_%EA%B3%B5%EA%B0%84SQL/data.zip

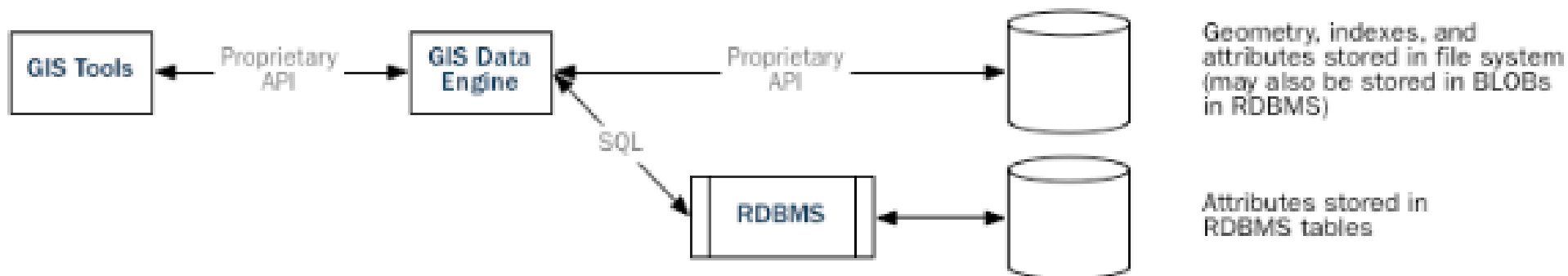
공간 SQL의 효능

DBMS를 이용한 GIS의 진화

First-Generation GIS:



Second-Generation GIS:



Third-Generation GIS:



간단한 GIS 분석 사례

• 서울시내 8차선 이상 도로에서 500미터 이내의 지하철역 찾기

1) 도로 레이어 불러서



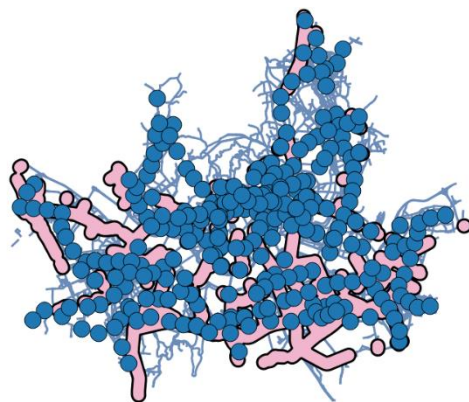
2) 8차선 이상 필터링



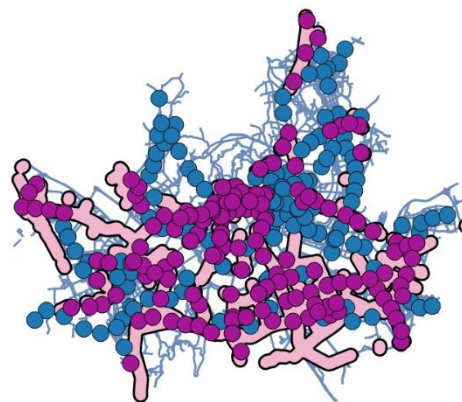
3) 선택도로 500미터 버퍼링



4) 지하철역 불러서



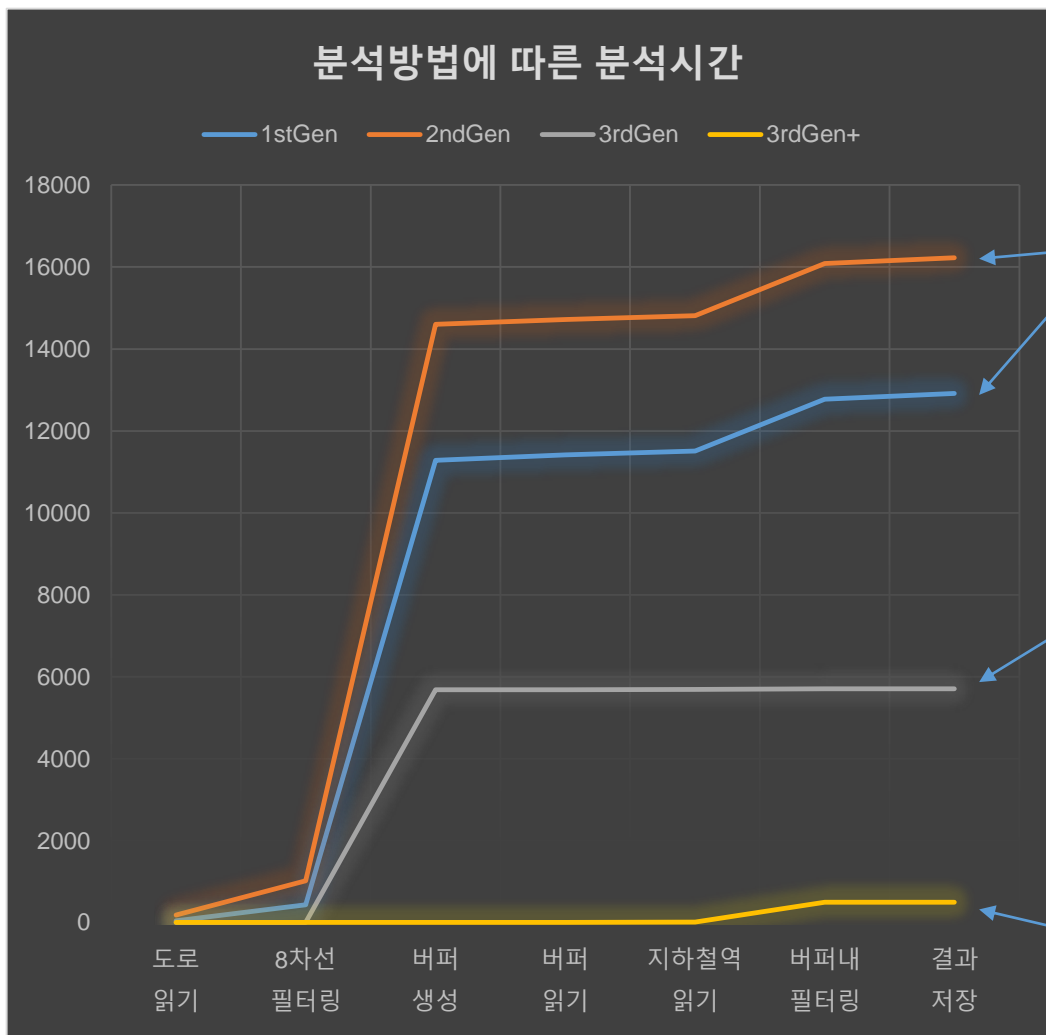
5) 버퍼도로 내의 지하철역 필터링



4가지 유형으로 구현

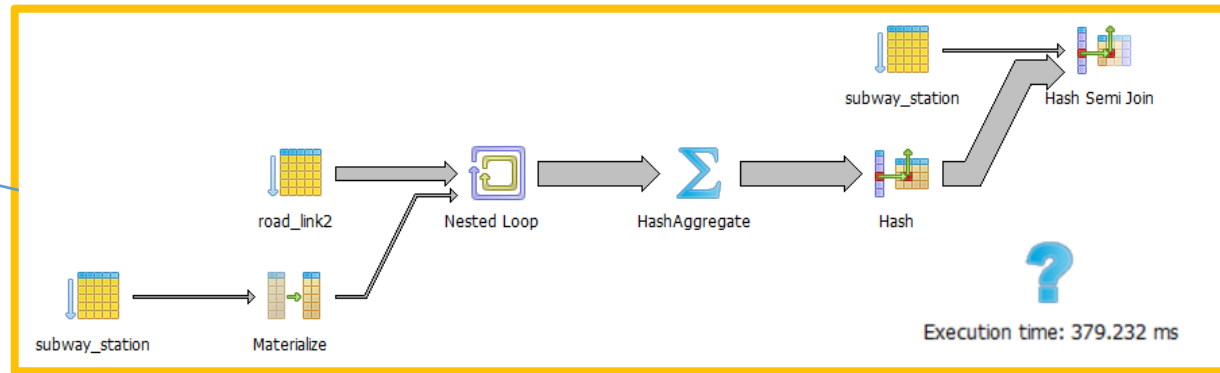
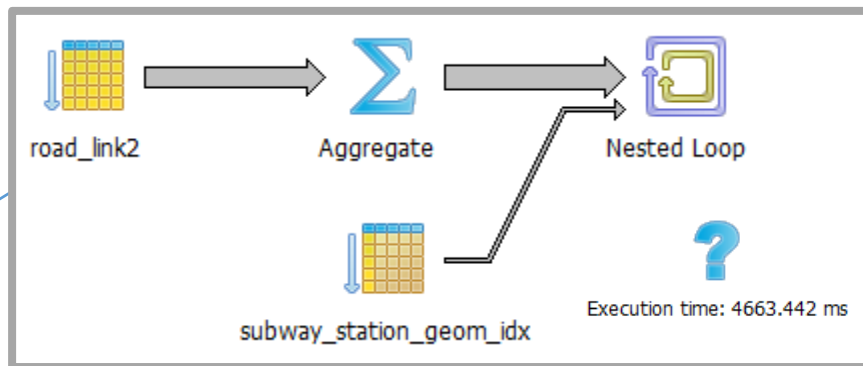
- 1세대 방식
 - 파일기반으로 GIS 툴을 이용해 분석
 - https://github.com/Gaia3D/workshop/blob/master/20171208_%EC%84%9C%EC%9C%A8%EC%97%B0_%EA%B3%B5%EA%B0%84SQL/sql/1stGen_FileBasedGis.py
- 2세대 방식
 - DB에서 데이터 불러와 GIS 툴을 이용해 분석
 - https://github.com/Gaia3D/workshop/blob/master/20171208_%EC%84%9C%EC%9C%A8%EC%97%B0_%EA%B3%B5%EA%B0%84SQL/sql/2ndGen_DbBasedGis.py
- 3세대 방식
 - 기존 분석과정 그대로 SQL로 분석
 - https://github.com/Gaia3D/workshop/blob/master/20171208_%EC%84%9C%EC%9C%A8%EC%97%B0_%EA%B3%B5%EA%B0%84SQL/sql/3rdGen_BasicSql.sql
- 3세대 방식 개선
 - DB에서 효율적으로 동작 가능한 SQL로 분석
 - https://github.com/Gaia3D/workshop/blob/master/20171208_%EC%84%9C%EC%9C%A8%EC%97%B0_%EA%B3%B5%EA%B0%84SQL/sql/3rdGen_AdvSql.sql
- 3세대 방식 더 개선
 - 더 효율적인 함수로 변경
 - https://github.com/Gaia3D/workshop/blob/master/20171208_%EC%84%9C%EC%9C%A8%EC%97%B0_%EA%B3%B5%EA%B0%84SQL/sql/3rdGen_DwithinSql.sql

수행결과 비교



도로 읽기 : 46ms
 8차선 이상 필터링 : 383ms
 500 미터 버퍼 생성: 10857ms
 버퍼 파일 읽기 : 131ms
 지하철역 읽기 : 90ms
 버퍼 안의 지하철역 필터링 : 1266ms
 결과 파일 저장 : 138ms
 =====
 전체 수행시간 : 12914ms

도로 읽기 : 180ms
 8차선 이상 필터링 : 841ms
 500 미터 버퍼 생성: 13584ms
 버퍼 파일 읽기 : 116ms
 지하철역 읽기 : 88ms
 버퍼 안의 지하철역 필터링 : 1275ms
 결과 파일 저장 : 144ms
 =====
 전체 수행시간 : 16231ms



전통적 로직을 그대로 구현한 SQL

```
select st.*  
from subway_station as st,  
(  
    select st_buffer(st_union(geom), 500) as geom  
    from road_link2 where lanes >= 8  
) as buf  
where st_within(st.geom, buf.geom)
```

하나로 합쳐서 500미터 버퍼 생성

8차선 이상 필터링

버퍼된 도로 위 지하철역 필터링

DB에서 효율적 동작하게 개선된 SQL

```
select * from subway_station  
where gid in
```

(
추출된 ID에 해당하는 지하철역만 조회

```
select distinct st.gid
```

```
from subway_station as st, road_link2 as road
```

```
where road.lanes >= 8
```

```
and ST_Distance(st.geom, road.geom) <= 500
```

```
)
```

도로에 가까운 지하철역이 여러 개 있으므로
중복 없애고 ID만 추출

8차선 이상 필터링

거리를 다 계산후
500미터 이내면 필터링

DB에서 더욱더 효율적 동작하게 개선된 SQL

```
select * from subway_station
where gid in
(
  select distinct st.gid
  from subway_station as st, road_link2 as road
  where road.lanes >= 8
    and ST_DWithin(st.geom, road.geom, 500)
)
```

500미터 이내 거리인
행만 필터링
(가능성 없는 것은
거리 계산도 않음)

Spatial SQL why, how?

가장 대표적인 Spatial SQL

```
SELECT superhero.name  
FROM city, superhero  
WHERE ST_Contains(city.geom, superhero.geom)  
AND city.name = 'Gotham';
```

- city와 superhero 테이블 사이에는 JOIN 가능한 Field가 없다.
- 하지만, '공간' 을 통해 연결성을 가질 수 있다!
- 때문에 공간적인 연결을 이용해 JOIN을 할 수 있다.

실용적인 분석의 예

- 1) 읍면동별로 최근린 3차의료기관을 산출하고
- 2) 각 읍면동별로 3차의료기관까지 도달가능한 예상시간을 구하여
- 3) 각 읍면동별 예상도달시간의 각 읍면동별 인구수를 곱한 값을
- 4) 각 시군구별로 합산한 값을 구하여
- 5) 전국의 시군구별 3차의료기관 분포가 얼마나 합리적인지 분석하라

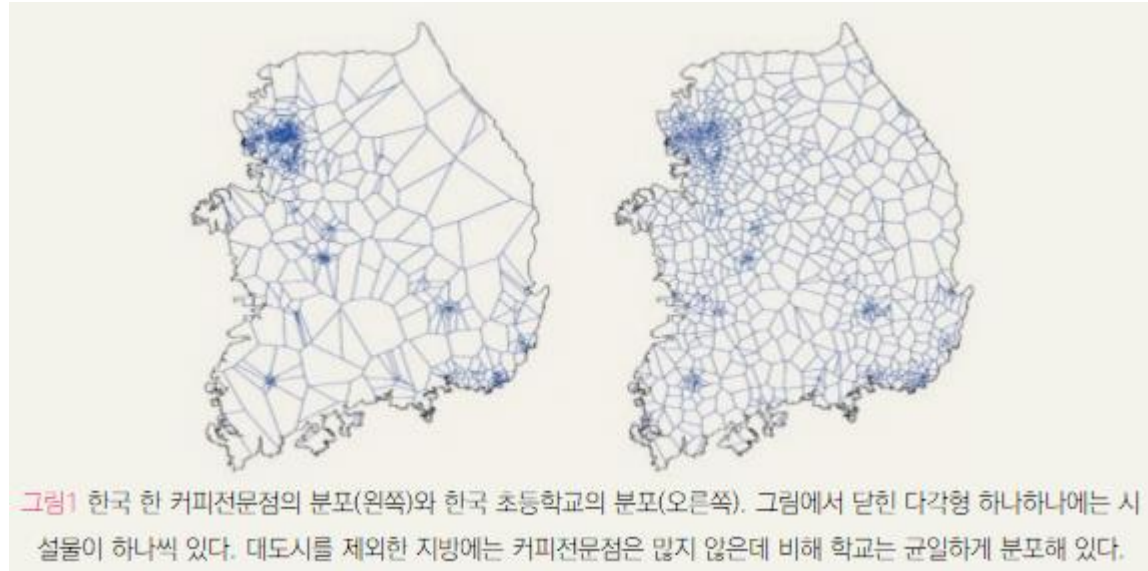
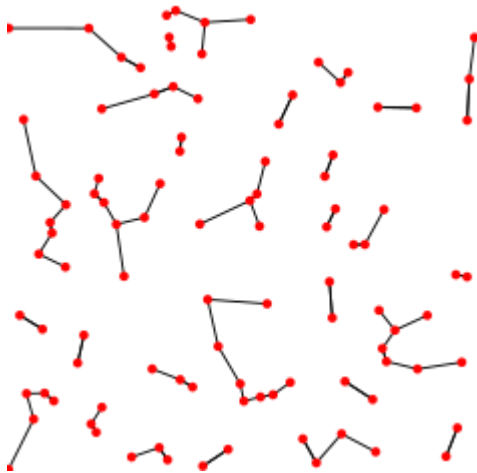


그림1 한국 한 커피전문점의 분포(왼쪽)와 한국 초등학교의 분포(오른쪽). 그림에서 달린 다각형 하나하나에는 시설물이 하나씩 있다. 대도시를 제외한 지방에는 커피전문점은 많지 않는데 비해 학교는 균일하게 분포해 있다.

출처: 세상물정의 물리학, 학교와 병원과 커피숍의 사정

Spatial SQL 이란?

- 공간정보 처리 함수를 이용할 수 있는 보통 SQL이다.
- 공간객체(Geometry/Geography/Raster)가 BLOB으로 저장된다.
- 공간 데이터는 database의 추가적인 컬럼(타입)일 뿐이다.
- 속성들은 database내의 일반 속성(컬럼)들과 동일하게 처리된다.

출처: PostGIS for Managers, Paul Ramsey

<http://s3.cleverelephant.ca/2014-postgis-for-managers.pdf>

Spatial SQL의 장점

- **“GIS in SQL”**

- database 내에서 쿼리를 이용해 데이터를 조작하고 조회

- **Shared Editing**

- 여러 사람이 동시에 편집해도 정합성 보장

- **Performance and Scale**

- 대량의 데이터와 많은 작업량을 처리

- 전통적인 GIS에서 단계별 데이터 변환/저장 시간을 줄여줌

Spatial Database 구성요소

- **Spatial data types**

- Geometry(point, line, polygon 등)

- **Spatial indexing**

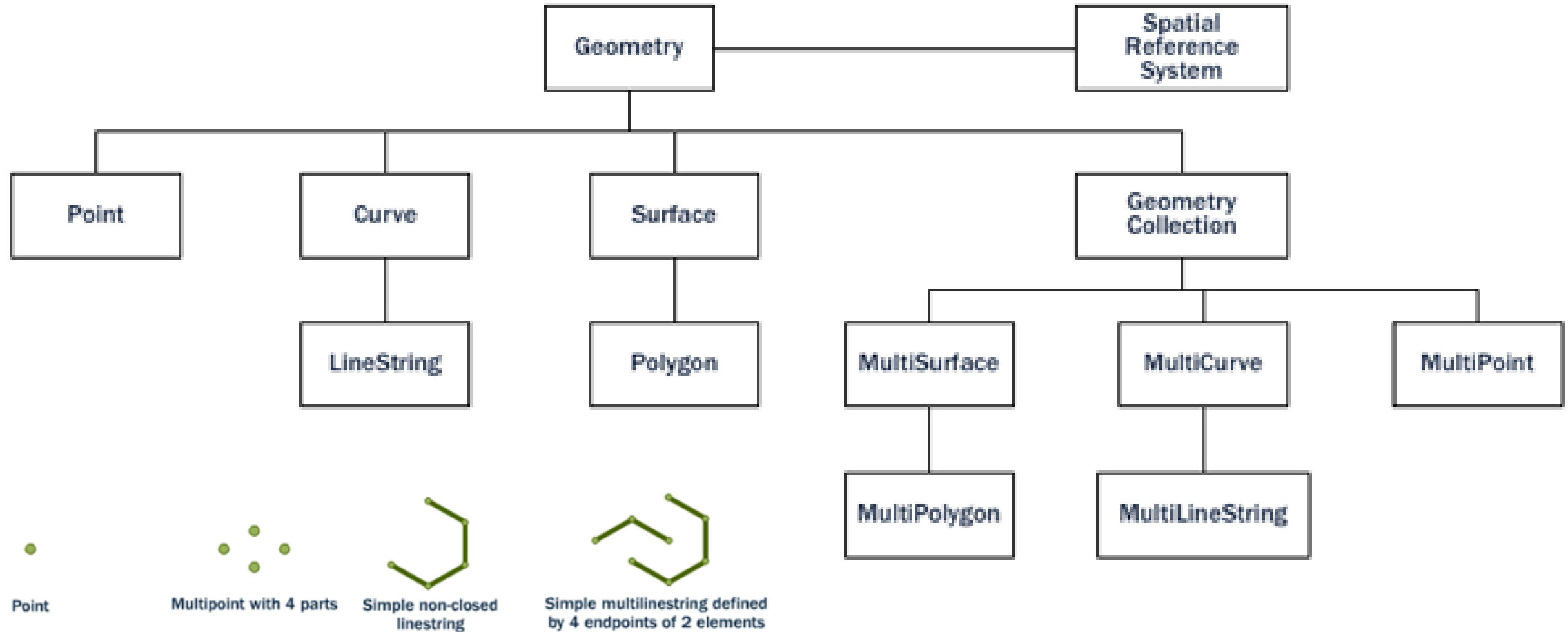
- 공간연산의 효율적 처리 지원

- **Spatial functions,**

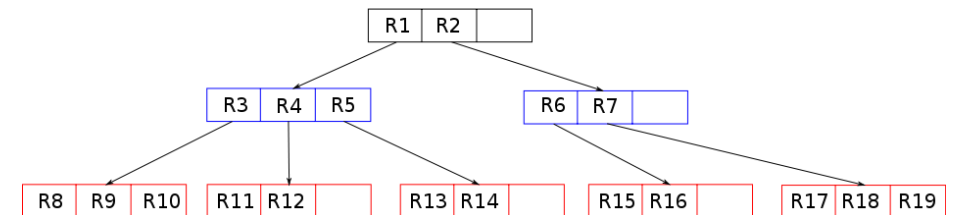
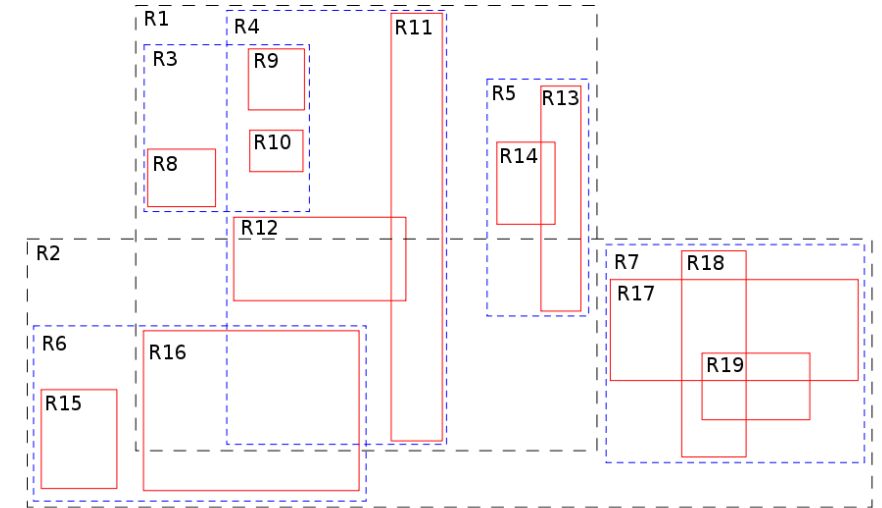
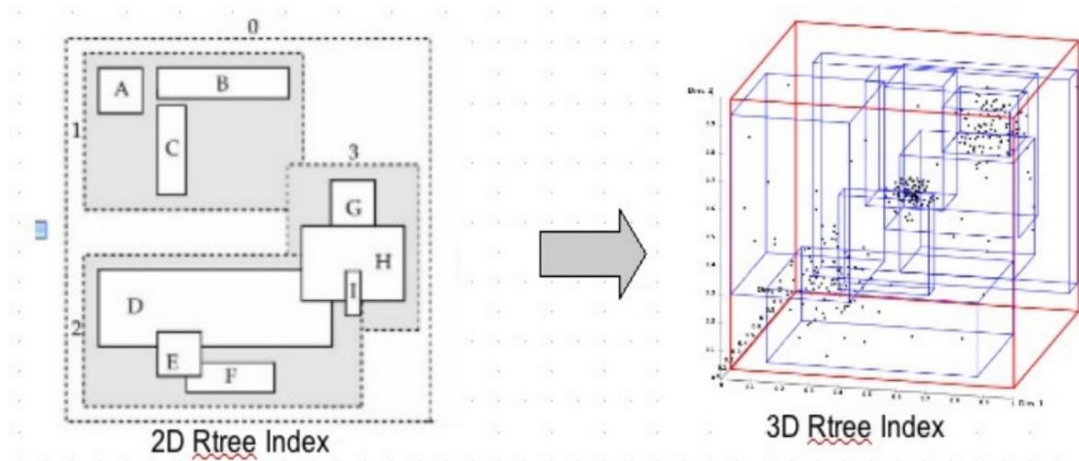
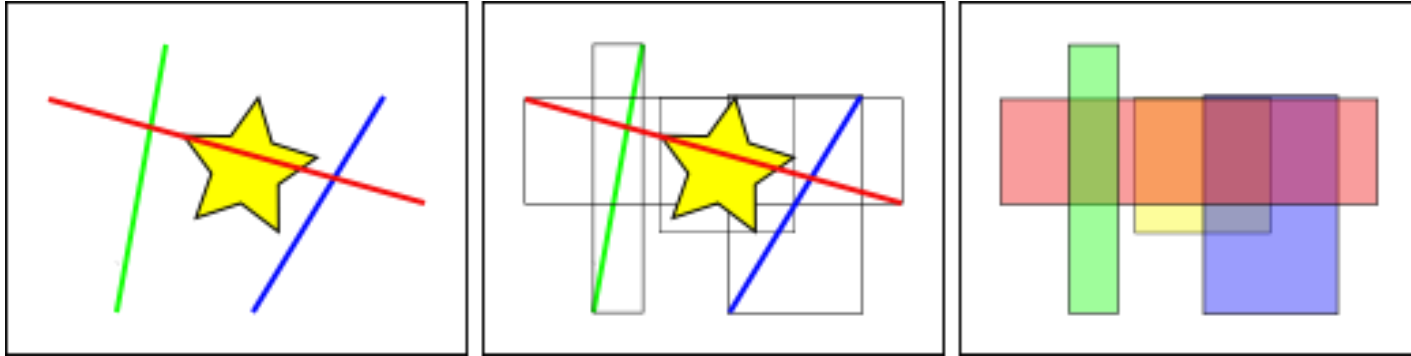
- 공간적 속성과 관계 질의

- 공간분석 단위로직 제공

Spatial Data Type

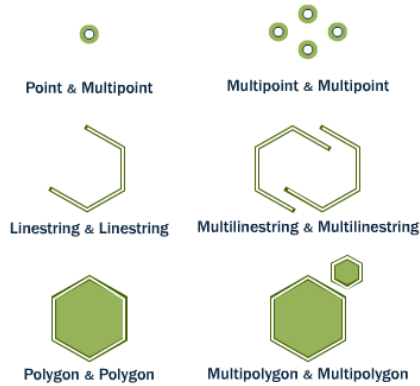


Spatial Index

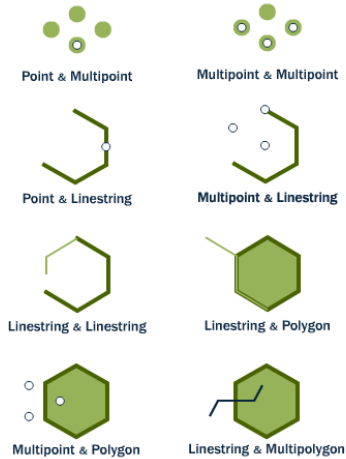


Spatial Function

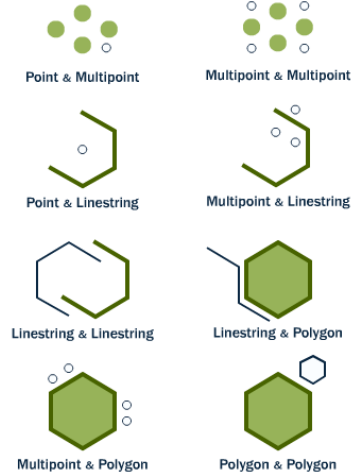
Equals



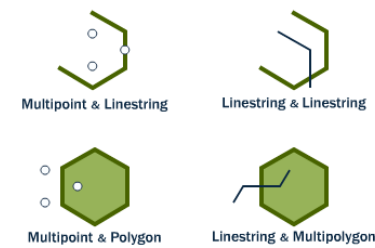
Intersects



Disjoint



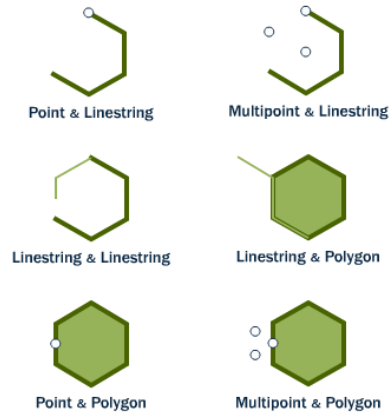
Cross



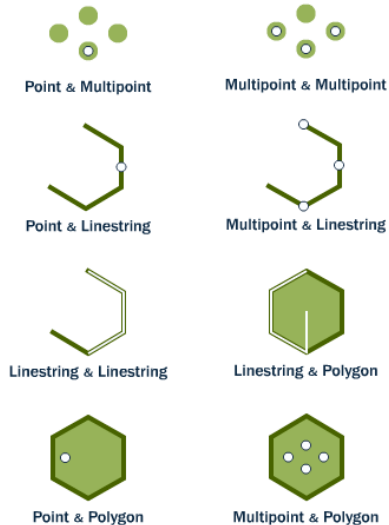
Overlap



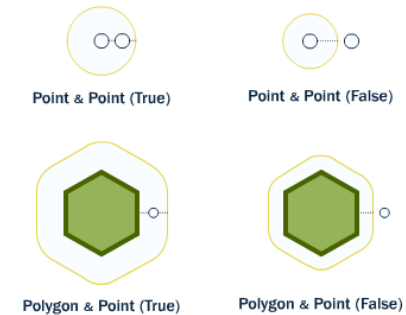
Touch



Within/Contains



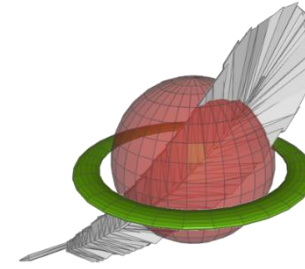
ST_Dwithin



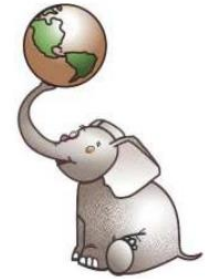
- ST_Length
- ST_Area
- ST_Distance
- ST_Buffer
- ST_Scale
- ST_Rotate
- ST_Transform
- ST_Intersection
- ST_Union
- ST_Simplify
- ST_Convexhull
- ST_AsText
- ST_AsBinary
- ST_FromText
- A && B
- A @ B
- A ~ B
- A <-> B
- A <#> B
- A || B
- A <=> B

Spatial DBMS 비교

비용절감



SpatialLite

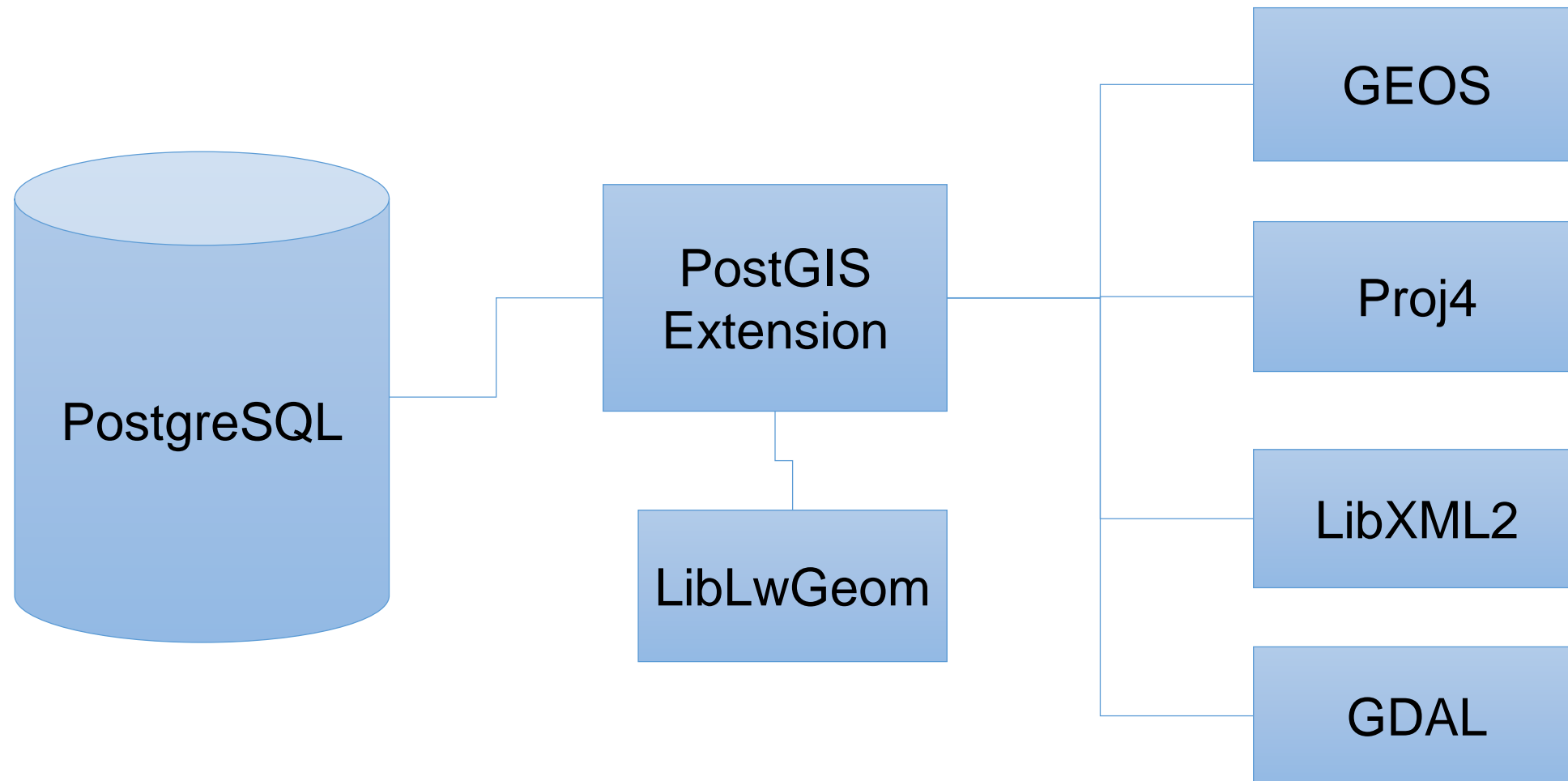


PostGIS

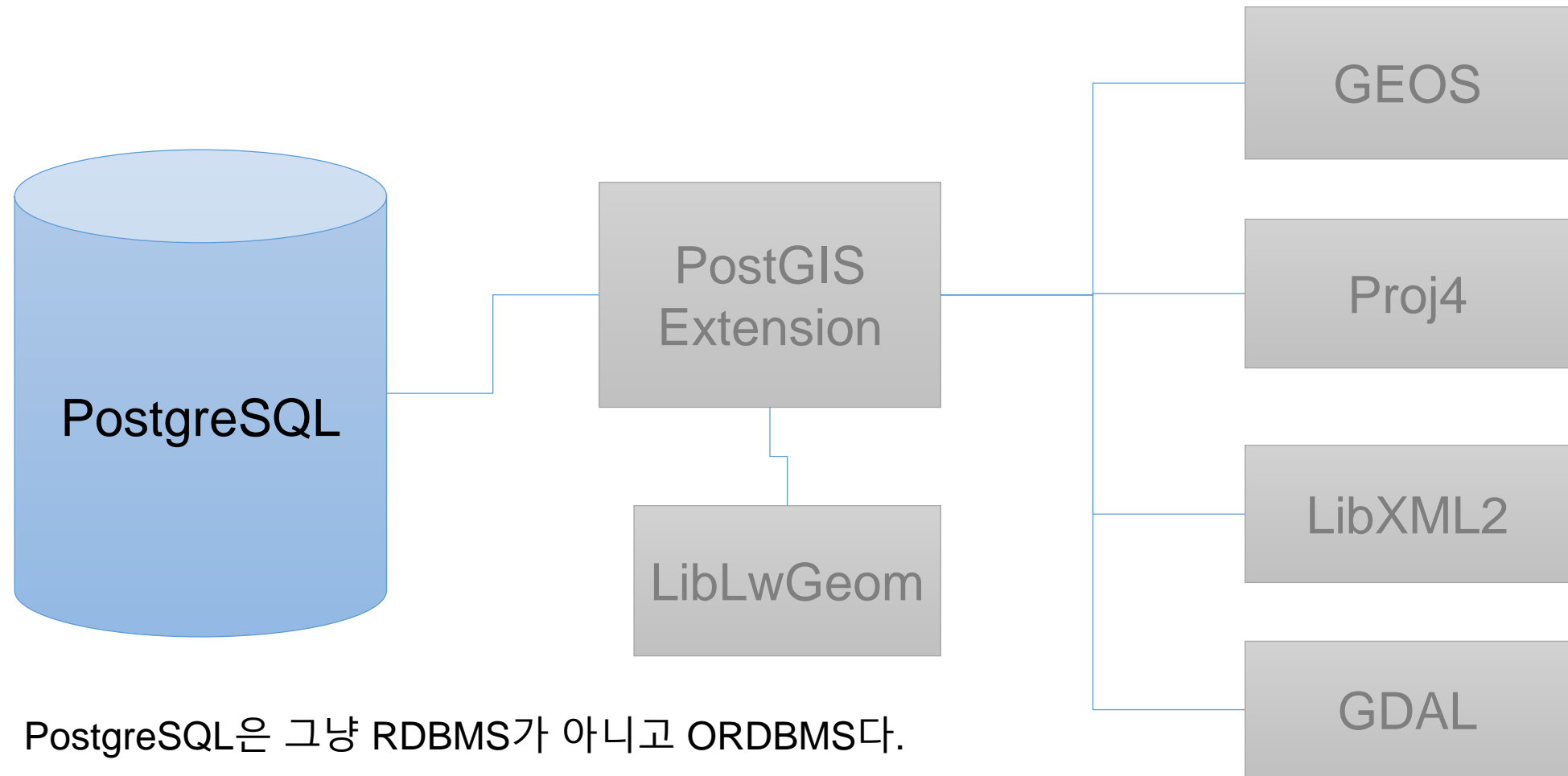


ORACLE®
S P A T I A L

PostGIS Architecture

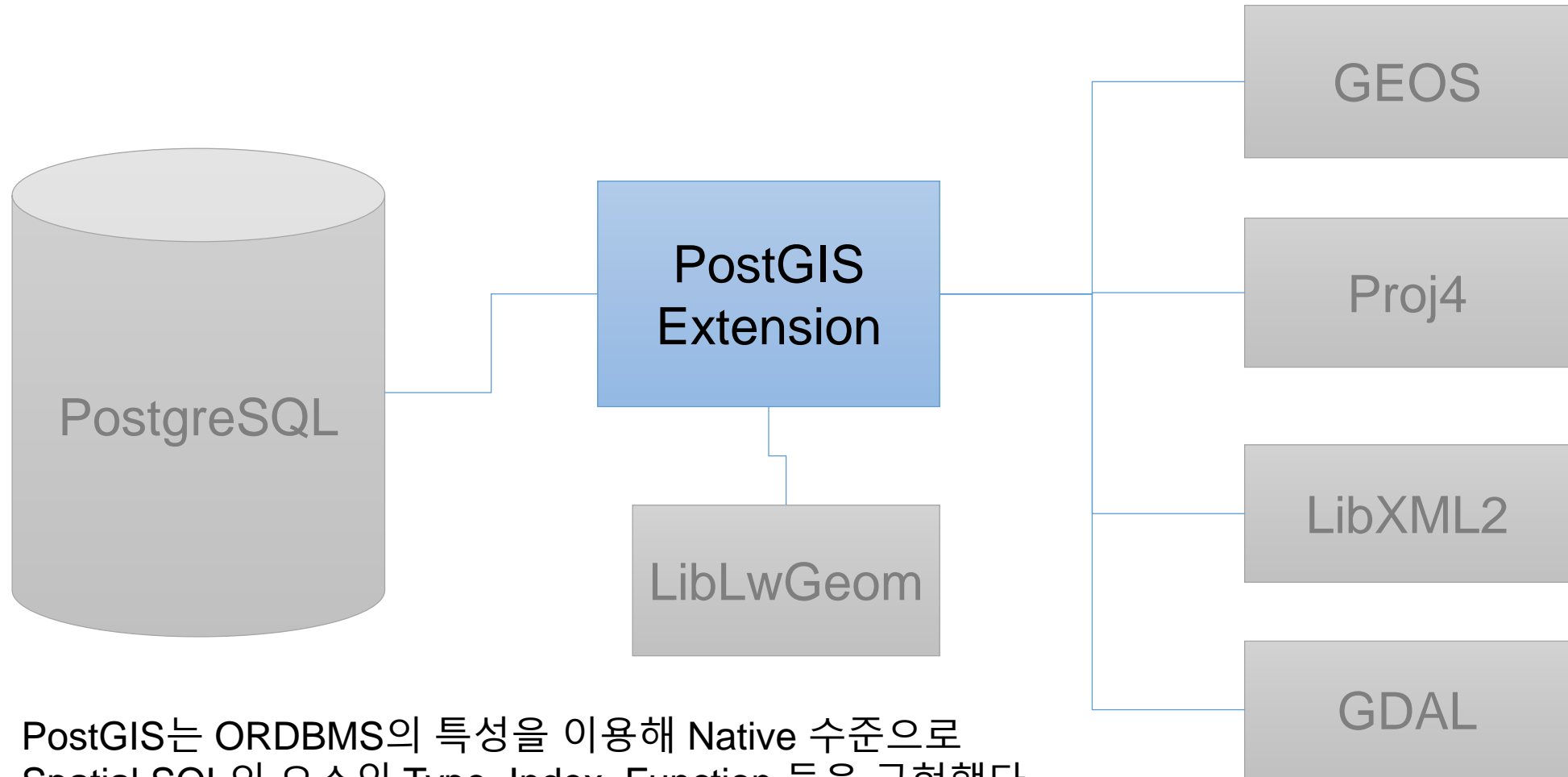


PostGIS Architecture



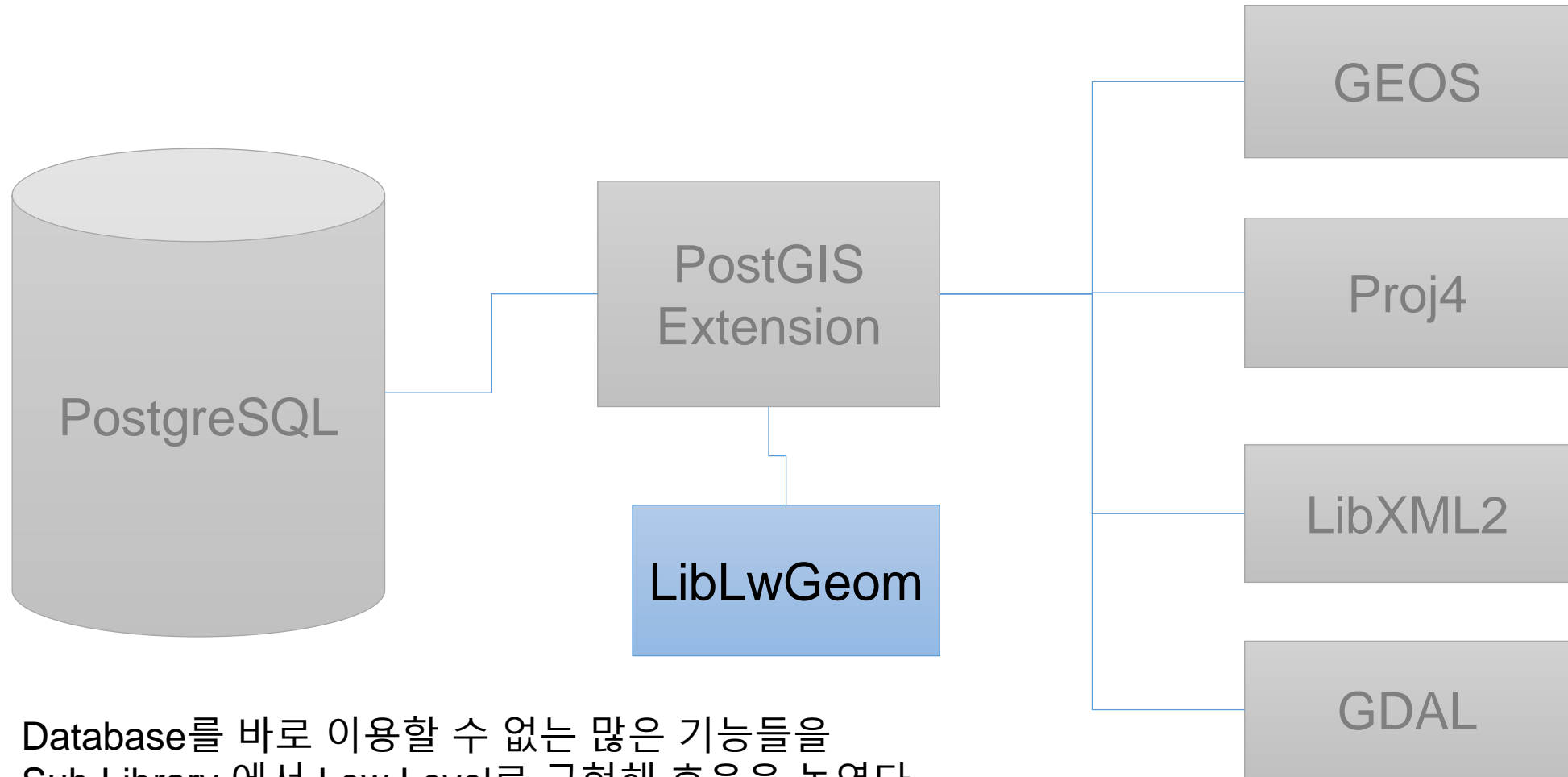
PostgreSQL은 그냥 RDBMS가 아니고 ORDBMS다.

PostGIS Architecture



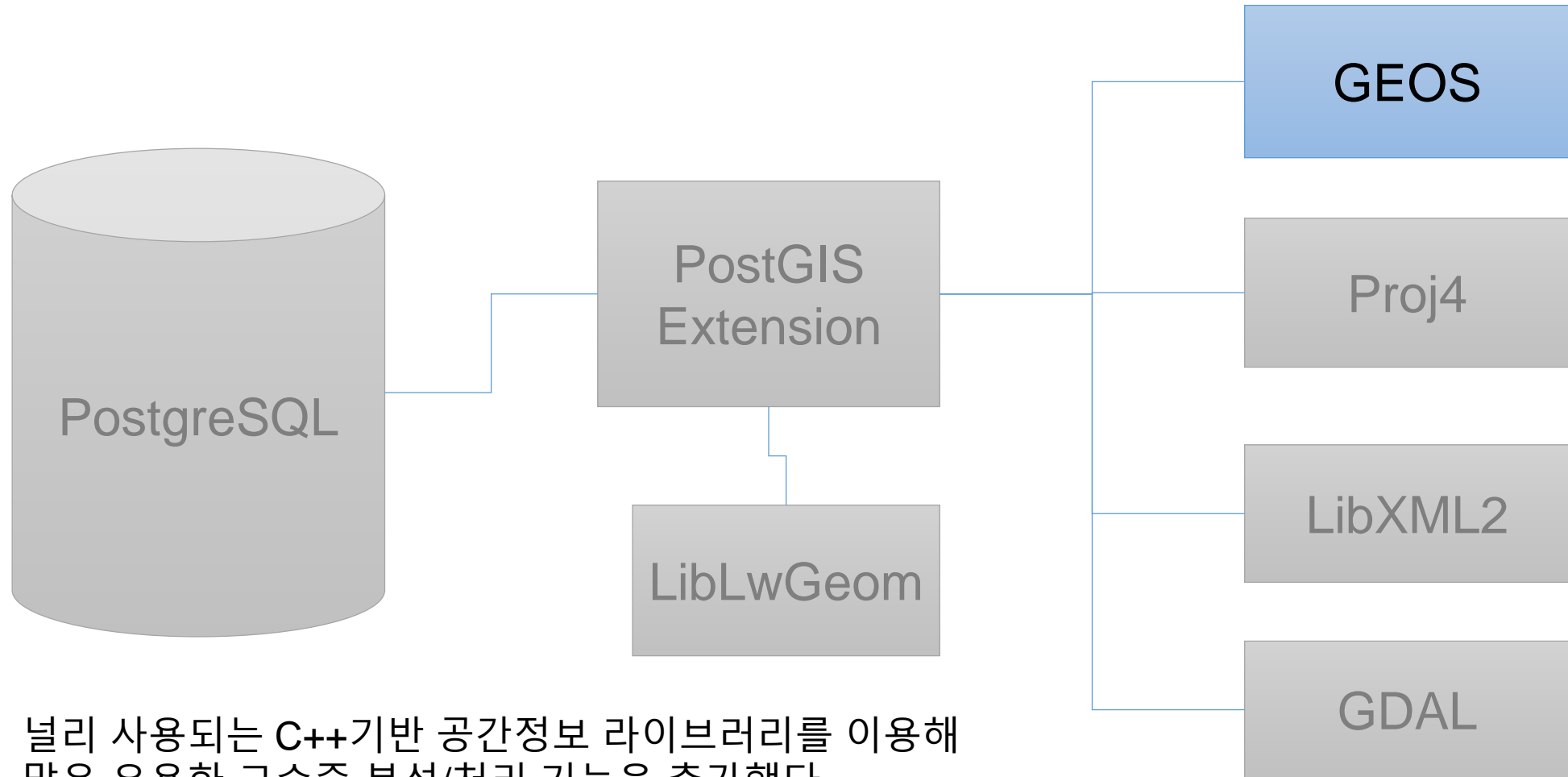
PostGIS는 ORDBMS의 특성을 이용해 Native 수준으로 Spatial SQL의 요소인 Type, Index, Function 들을 구현했다.

PostGIS Architecture



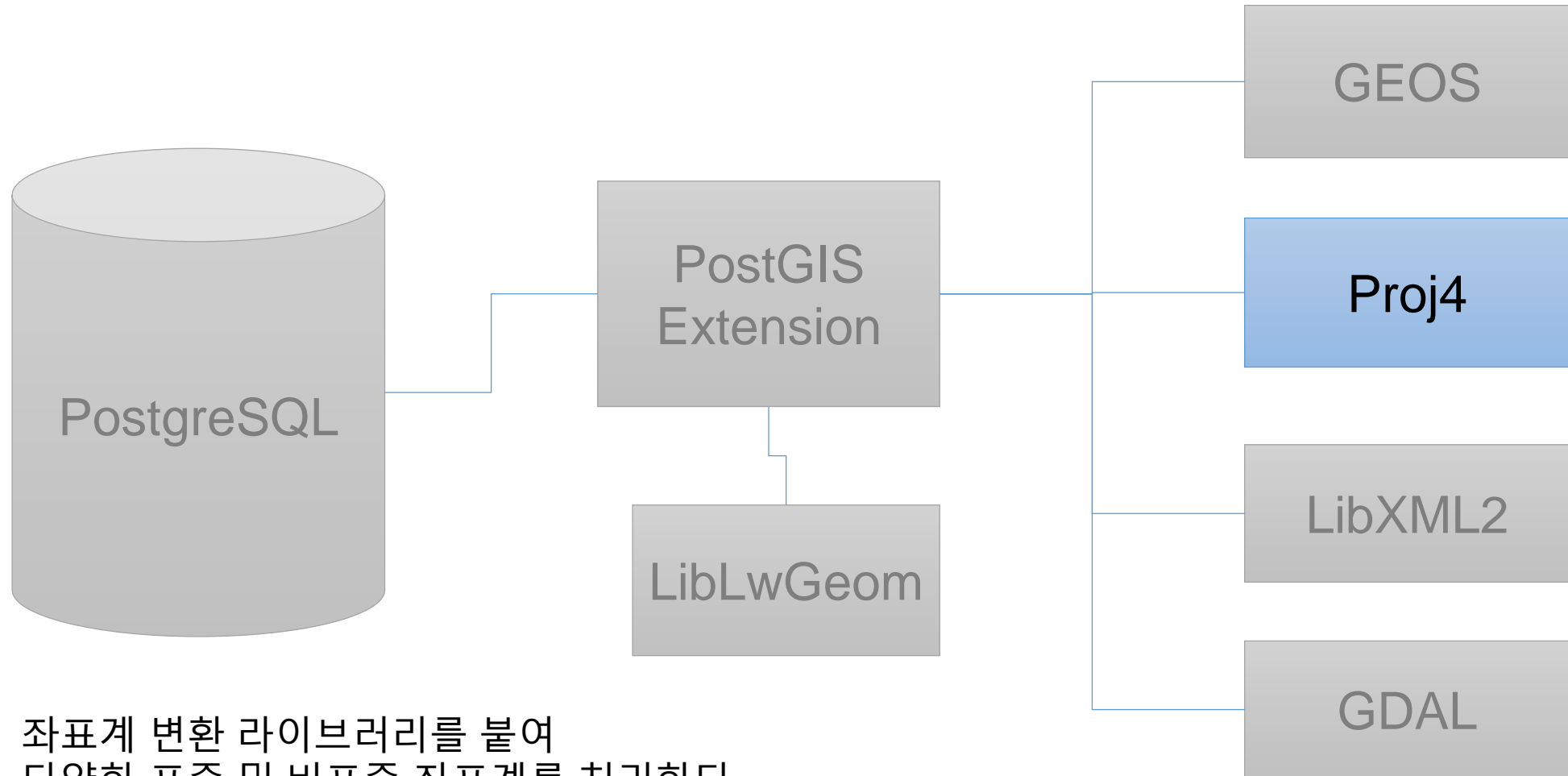
Database를 바로 이용할 수 없는 많은 기능들을
Sub Library 에서 Low Level로 구현해 효율을 높였다.

PostGIS Architecture



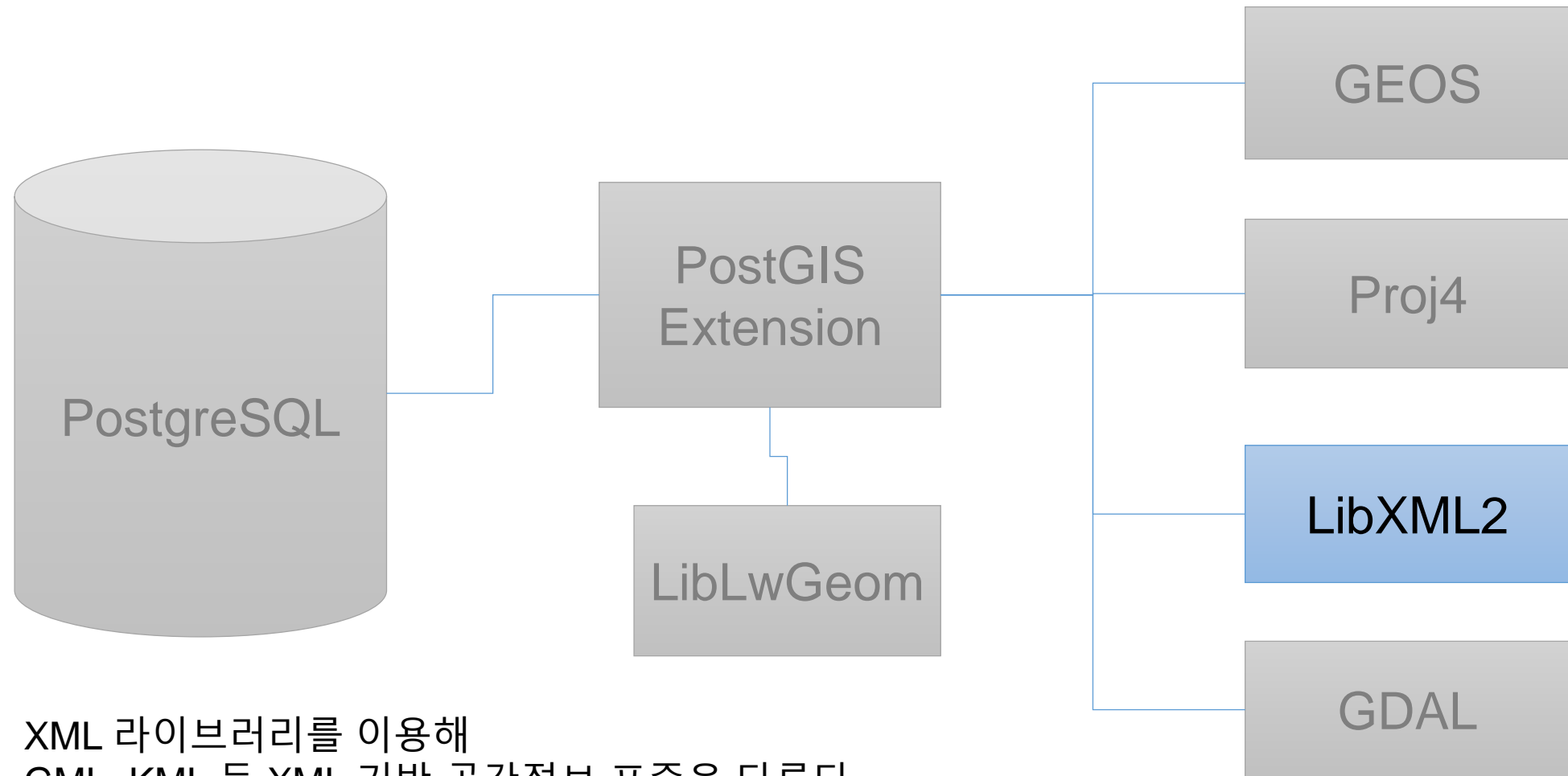
널리 사용되는 C++기반 공간정보 라이브러리를 이용해 많은 유용한 고수준 분석/처리 기능을 추가했다.

PostGIS Architecture



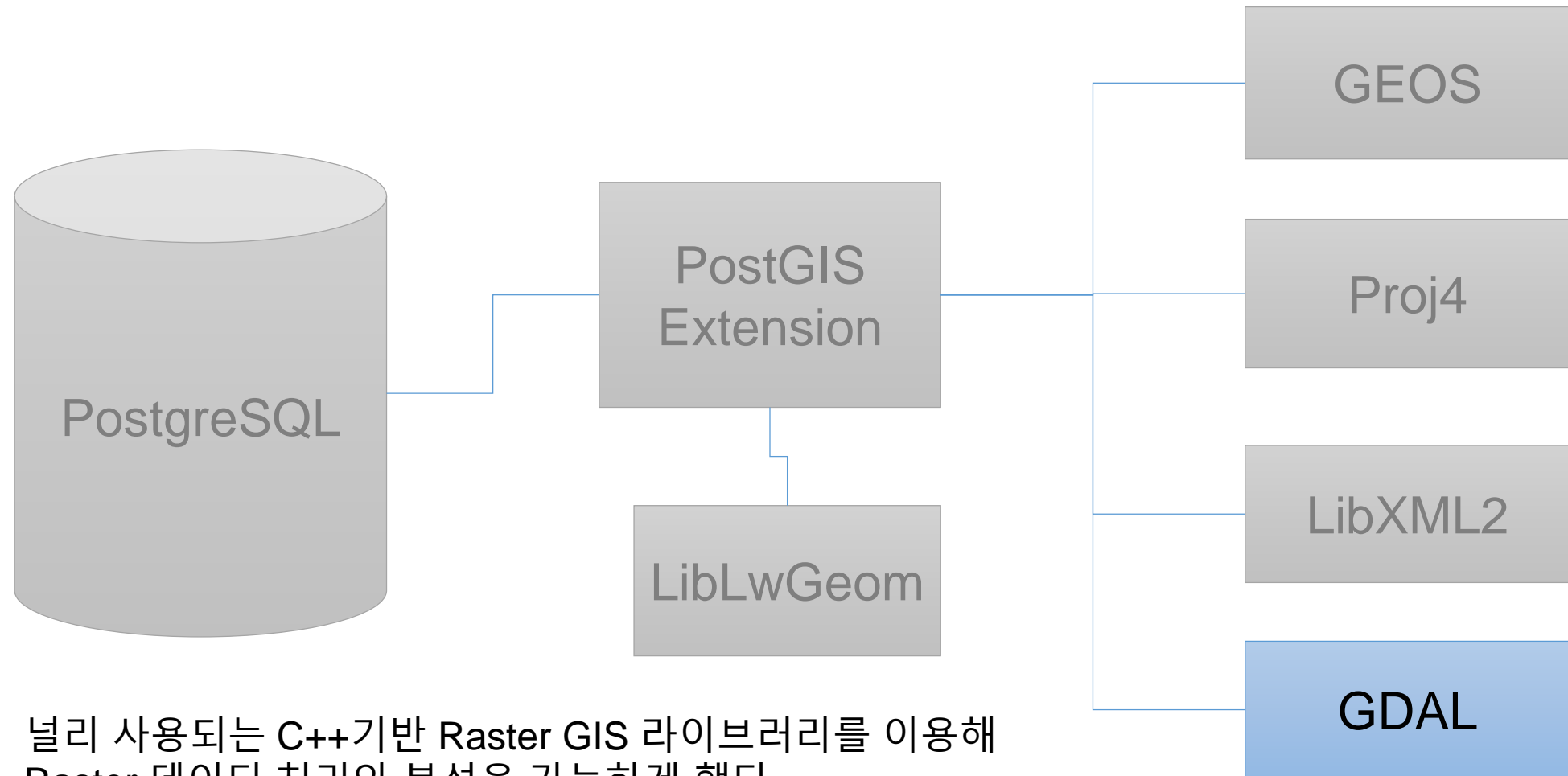
좌표계 변환 라이브러리를 붙여
다양한 표준 및 비표준 좌표계를 처리한다.

PostGIS Architecture



XML 라이브러리를 이용해
GML, KML 등 XML 기반 공간정보 표준을 다룬다.

PostGIS Architecture

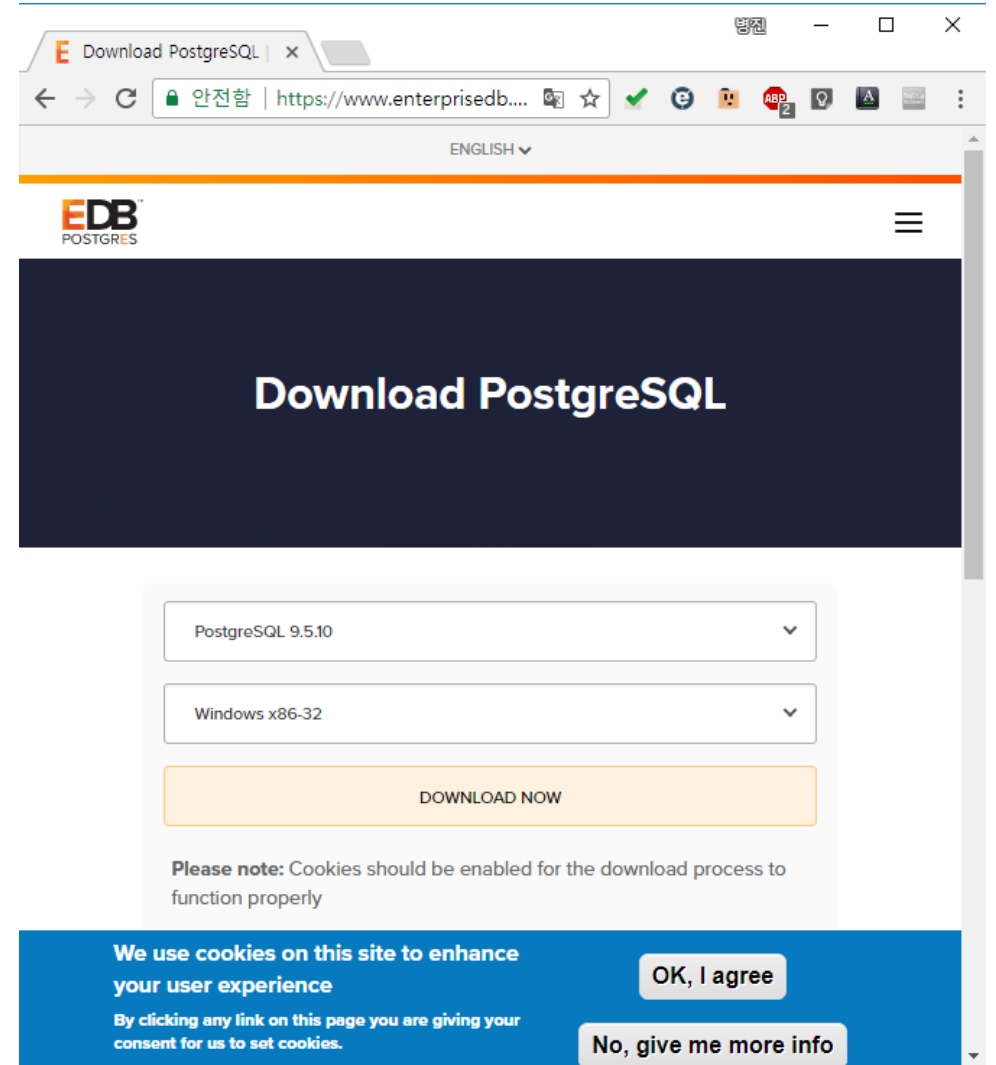


널리 사용되는 C++기반 Raster GIS 라이브러리를 이용해 Raster 데이터 처리와 분석을 가능하게 했다.

PostGIS 설치

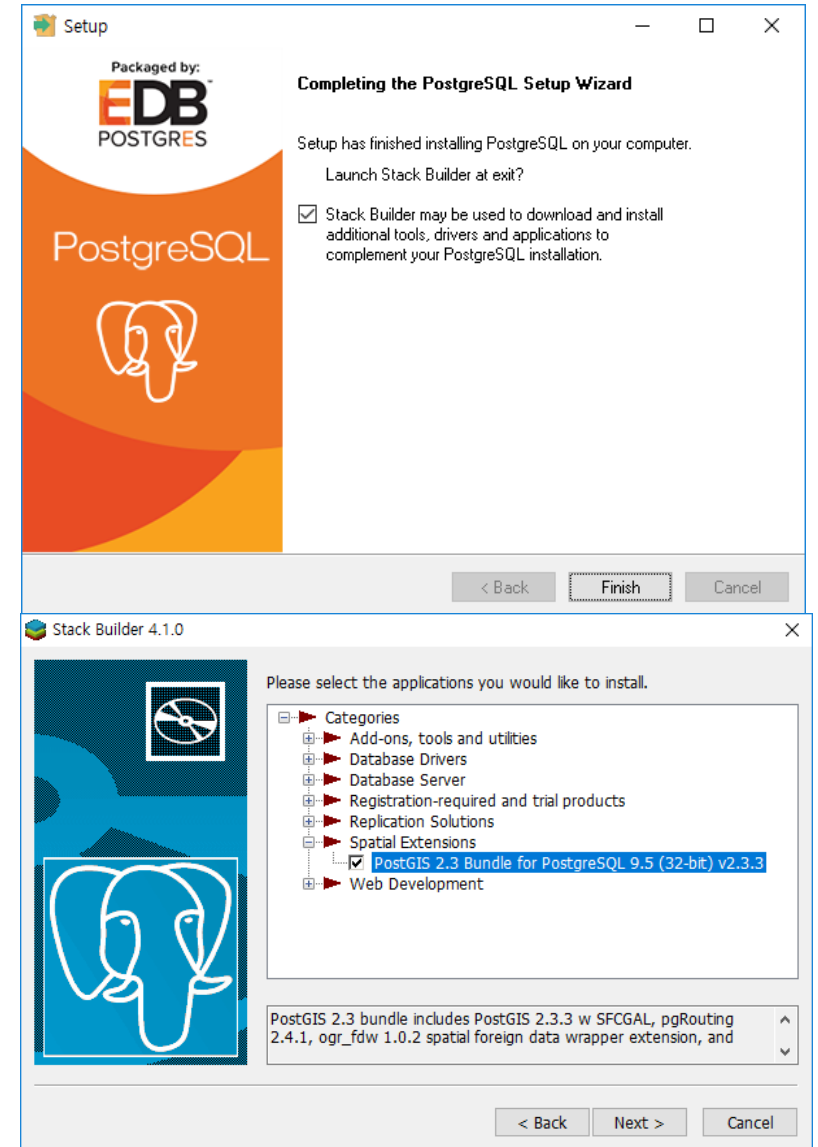
설치파일 다운로드

- PostgreSQL 9.5
 - <https://www.postgresql.org/download/windows/>
 - Download the installer (EDB 배포본)
 - BigSQL 배포본은 PostGIS 설치 불가
- PostGIS 2.3
 - PostgreSQL 버전에 맞는 버전 설치 필수
 - Stack Builder 로 설치
- QGIS 2.18
 - 공간정보를 편하게 다룰 수 있는 툴
 - <http://qgis.org/ko/site/forusers/download.html>




설치

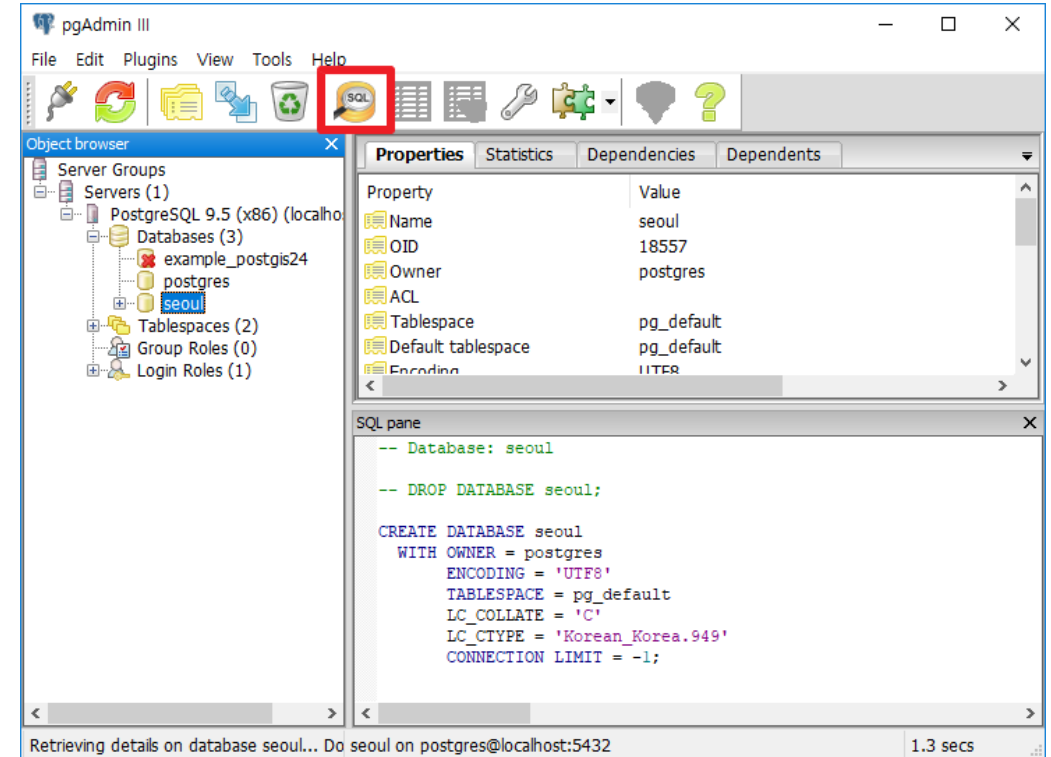
- PostgreSQL 설치
 - 다음, 다음으로 거의 설치 가능
 - Superuser(postgres) 암호로 보통 postgres
- PostGIS
 - PostgreSQL 다 설치 후 Stack Builder 에서 설치
 - Spatial Extension 밑의 PostGIS 2.3 선택
- QGIS
 - 설치 파일 실행으로 쉽게 설치가능
 - 샘플 데이터 세트는 공식 매뉴얼을 따라할 사람만 필요 (안 설치 해도 됨)
 - 오늘 실습에도 QGIS 필요



공간정보 들어갈 수 있는 Database 만들기

- pgAdmin 3 실행
 - PG를 쉽게 다룰 수 있게 하는 기본 툴
 - postgres 유저의 암호를 알아야 사용 가능
- 새 Database 만들기
 - seoul 이라는 이름으로 만들기
 - 트리의 Databases 항목에서 오른쪽 마우스
 - New Database...
 - Name: seoul
 - [OK]
- seoul database를 공간정보 처리 가능하게 만들기
 - Databases 하위 seoul 항목 선택하고
 - SQL 창 띄워서

```
create extension postgis;
```
 - [F5] 키나 실행()버튼 눌러 실행




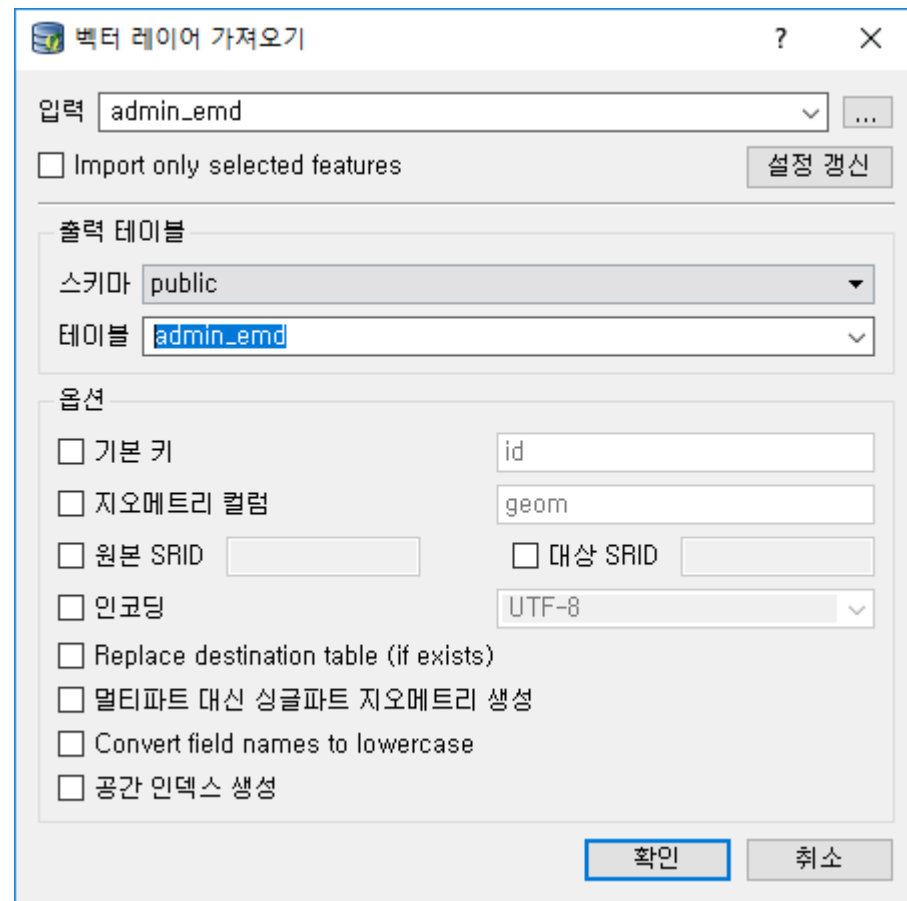
공간자료를 PostGIS에 올리기

QGIS 이용 (1/2)

- admin_sgg 파일 읽기
 - QGIS에서 읽어
 - **한글 잘 나오는지 확인**
 - 레이어 속성에서
데이터 소스 코드화: system
- PostGIS 연결 추가
 - Browser Panel에서
 - PostGIS 오른쪽 클릭
 - 새 연결...
 - 정보입력
 - 이름: pg95
 - 호스트: localhost
 - 데이터베이스: seoul
 - 사용자이름: postgres, 저장 체크
 - 비밀번호: postgres, 저장 체크

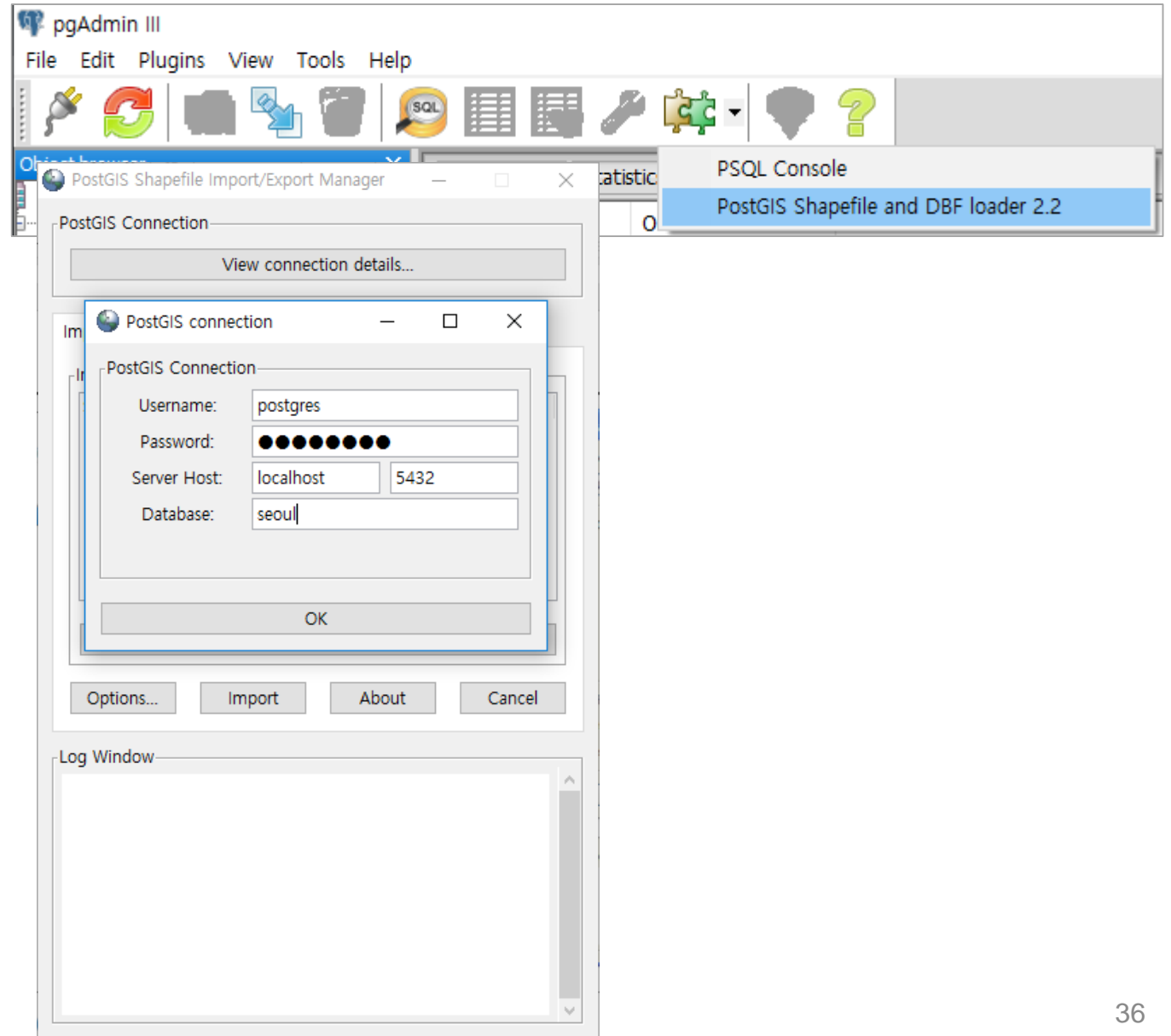
QGIS 이용 (2/2)

- DB 관리자 실행
 - QGIS 메뉴에서
 - 플러그인-플러그인 관리 및 설치...
 - 설치됨 탭에서 DB Manager 활성화 확인
 - 데이터베이스-DB관리자-DB관리자
- 데이터 올리기
 - DB 관리자 Tree에서
 - PostGIS - pg5 - public 선택 (연결확인)
 - 벡터 레이어 가져오기 버튼 
 - 입력: admin_sgg
 - 테이블: admin_sgg
- 데이터 확인
 - public 확장
 - admin_sgg 더블클릭
 - QGIS에서 확인




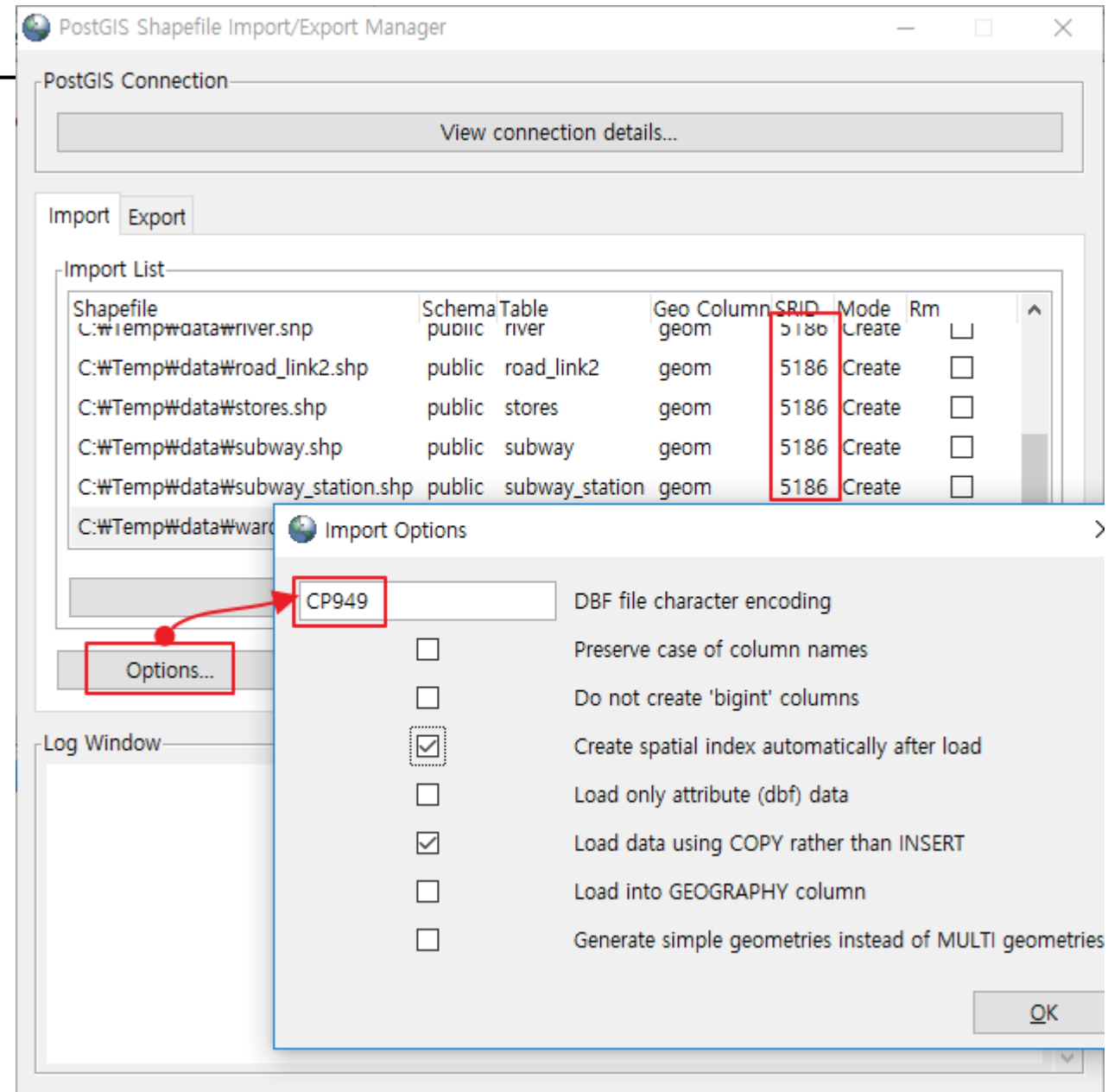
Shape Loader 이용 (1/2)

- Shape Loader 실행
 - PostgreSQL과 함께 설치된 pgAdmin III 실행
 - Servers에서 PostgreSQL 9.5 선택하고 더블클릭
 - 암호(postgres) 입력해 연결
 - Databases에서 seoul 선택
 - 툴 중에서 PostGIS Shapefile and DBF loader 2.2 실행
- 연결 설정
 - [View connection detail...] 버튼
 - Username: postgres
 - Password: postgres
 - Server Host: localhost 5432
 - Database: seoul



Shape Loader 이용 (2/2)

- Import할 파일 선택
 - [Add File] 선택
 - 데이터가 있는 폴더(한글X)로 이동해 import 할 파일 모두 선택
 - [Open]
- 좌표계 지정
 - 파일 리스트에서 SRID 값을 모두 5186으로 변경
 - 단, road_link_geographic 은 4326으로 변경
- 한글 코드페이지 지정
 - [Options...] 선택
 - DBF file character encoding에 CP949 입력
 - [OK]
- [Import]
- pgAdmin III에서 확인
 - Seoul – Schemas – public - Tables
 - 원하는 테이블 선택하고
 - [View data] 버튼 



Spatial SQL 실습

```
SELECT
*
FROM
stores;
```



39

조건절을 추가 했고요.

SELECT

*

FROM

stores

WHERE

brand=' 이마트 ' ;

함수도 써볼까요?

```
SELECT  
  COUNT(nam)  
FROM  
stores  
WHERE  
addr like '%영등포구%';
```

브랜드별 점포수도 분석해 보지요.

```
SELECT  
brand,  
COUNT(nam) as "점포수"  
FROM  
stores  
GROUP BY  
brand;
```

- 단따옴표 vs 쌍따옴표
 - 단따옴표는 문자열을 의미
 - 쌍따옴표는 객체명을 의미
 - 객체명이 소문자일때는 그냥 적어도 됨
 - 한글이나 대문자는 꼭 쌍따옴표 필요

표준편차 같은 것도 분석해 볼 수 있지요.

SELECT

brand,

AVG(char_length(nam)),

STDDEV(char_length(nam)),

FROM

stores

GROUP BY

brand;

```
SELECT  
ST_AsText (geom)  
FROM  
firestation;
```

```
"POINT(199024.707810483 551054.345970521)"  
"POINT(187630.779273731 550959.271440678)"  
"POINT(188697.498274389 547889.595624078)"  
"POINT(188083.069315405 544286.907097312)"  
"POINT(191513.595677515 546567.657631612)"  
"POINT(192709.499498786 543934.716259131)"  
"POINT(195773.812578709 541616.937450973)"  
"POINT(199276.918538287 544361.102824507)"  
"POINT(202429.008902833 543188.485493705)"  
"POINT(212591.013743505 544507.787855193)"  
"POINT(211046.709538044 547793.045668135)"  
"POINT(194081.877897298 549999.530447381)"  
"POINT(197473.050073947 548306.166380932)"  
"POINT(207313.523766092 549495.358416943)"  
"POINT(201350.396790628 551700.90690202)"  
"POINT(194250.633663578 555746.764731673)"  
"POINT(194343.893334265 552645.674846625)"  
"POINT(198162.519362067 552609.548072939)"  
"POINT(205895.927248916 551623.426933408)"  
"POINT(202805.156974282 555885.932392841)"  
"POINT(208356.511341705 557326.3187219)"  
"POINT(206265.001878799 559863.523634983)"  
"POINT(203790.301551923 562717.534532934)"
```

길이를 구하는 거야 껌이지요~~~

```
SELECT  
link_id,  
ST_Length (geom)  
FROM  
road_link_geographic;
```

헉! 사실은 공간정보 지식이 필요해요 T.T

도로 길이 총합이 1보다 작아요~~~~



미터 단위의 좌표계로 변환해야 하네요.

```
SELECT  
link_id,  
ST_Length(ST_Transform(geom, 5179)),  
FROM  
road_link_geographic;
```

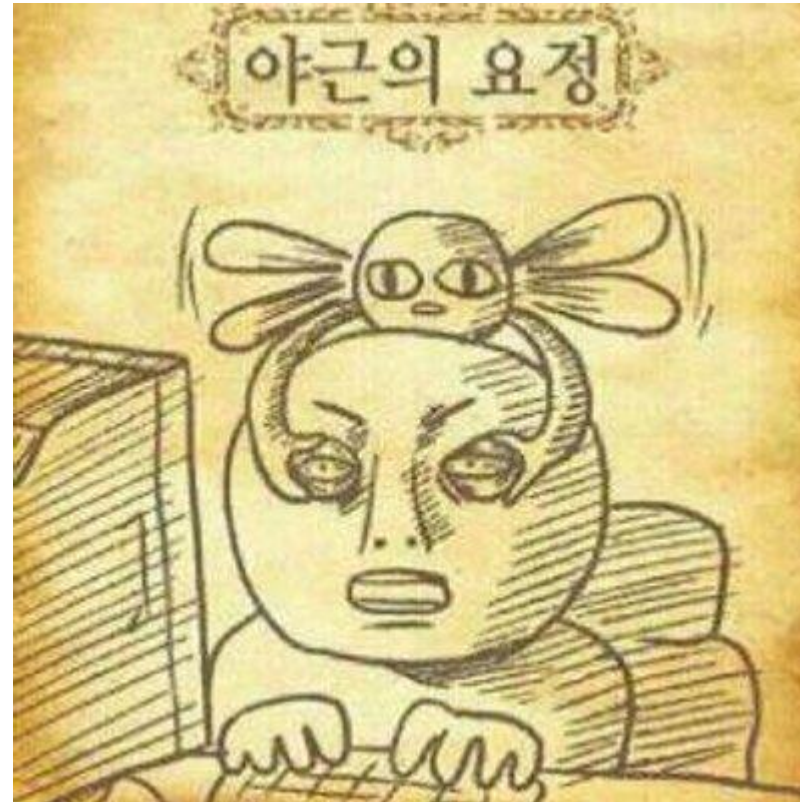
- **EPSG:5179**
 - 미터단위 TM 좌표계
 - 국토지리정보원 온맵, 네이버지도에서 사용중
 - 전국 규모 사용에 적합

이제 면적도 계산할 수 있어요.

```
SELECT  
ufid,  
ST_Area(ST_TRANSFORM(geom, 5179))  
FROM  
building  
LIMIT 100;
```


헉! 사실은 이렇게 해야해요. 불필요한 변환은 야근의 요정을 불러요.

```
SELECT  
ufid,  
ST_Area(geom)  
FROM  
building  
LIMIT 100;
```



버... 버... 버퍼링 알아요? – 공간 컬럼 추가

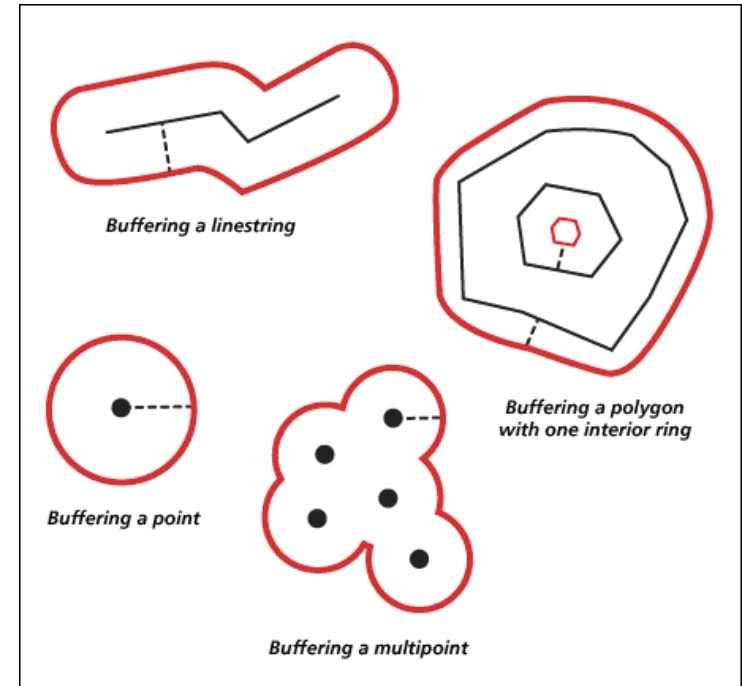
ALTER TABLE

strokes

ADD Column

buffer

geometry(Polygon, 4326)



- EPSG:4326

- 도단위 경위도 좌표계
- GPS에서 오는 좌표값과 동일
- 전세계적 자료 다룰때 가장 많이 사용

버... 버... 버퍼링 알아요? – 공간 컬럼에 넣기

UPDATE

stores

SET

buffer=

ST_Transform

(ST_Buffer(ST_Transform(geom, 5186), 30), 4326);

IT의 피인 JSON과 XML로 변환

```
SELECT ST_AsGeoJSON(ST_Transform(geom, 4326)) FROM stores;  
SELECT ST_AsGML(geom) FROM stores;
```

```
{  
  "type": "Feature",  
  "geometry": {  
    "type": "Point",  
    "coordinates": [127.09, 37.59]  
  },  
  "properties": {  
    "name": "Dinagat Islands"  
  }  
}
```

```
<Feature>  
  <name>Dinagat Islands</name>  
  <position>  
    <gml:Point srsName="EPSG:5186">  
      <gml:coordinates>  
        207965.03669,555284.45453  
      </gml:coordinates>  
    </gml:Point>  
  </position>  
</Feature>
```

선이 몇 개 점으로 그려졌는지 쫓아야~

```
SELECT  
ST_NPoints(geom) as num_point  
FROM  
road_link2  
ORDER BY num_point DESC  
limit 100;
```

공간적으로 걸치는 것을 이용해 JOIN을~

SELECT

shop.nam as "매장명",

metro.nam as "인근역"

FROM

stores as shop, subway_station as metro

WHERE

ST_Intersects(

ST_Buffer(shop.geom, 500), metro.geom);

속성조건도 함께 필터링

SELECT

shop.nam as "매장명",

metro.nam as "인근역"

FROM

stores as shop, subway_station as metro

WHERE

shop.addr LIKE '%영등포%'

AND

ST_Intersects(

ST_Buffer(shop.geom, 500), metro.geom);

공공시설 분포분석

각 동별로 가장 가까운 소방서는?

SELECT

emd.emd_cd,

fire.nam,

ST_Distance(ST_Centroid(emd.geom), fire.geom)

AS dist

FROM

admin_emd **AS** emd, firestation **AS** fire

ORDER BY emd, emd_cd, dist;

가장 가까운 소방서까지의 거리에 동별인구 곱하자

```
SELECT
    min_t.emd_cd, emd_nm, pop, fire_nm,
    dt.dist, pop*dt.dist as pop_dist
FROM
    (
        SELECT
            emd.emd_cd,
            min(ST_Distance(
                ST_Centroid(emd.geom), fire.geom)
            ) AS dist
        FROM
            admin_emd AS emd, firestation
            AS fire
        GROUP BY emd_cd
    ) AS min_t,
```

```
(
    SELECT
        emd.emd_cd,
        emd.emd_nm,
        emd.pop2008 as pop,
        fire.nam as fire_nm,
        ST_Distance(
            ST_Centroid(emd.geom), fire.geom
        ) AS dist
    FROM
        admin_emd AS emd, firestation AS fire
    ) AS dt
WHERE
    min_t.emd_cd = dt.emd_cd
    AND min_t.dist = dt.dist;
```

공공시설 접근지수가 낮은 20개 동 찾기

- 공공시설 접근지수가 낮은 20개 동 찾기

...

ORDER BY pop_dist

LIMIT 20

- 동별 소방서 접근지수의 평균과 분산은?

SELECT

AVG (pop*dt.dist) ,
STDDEV (pop*dt.dist)

...

https://github.com/Gaia3D/workshop/blob/master/20171208_%EC%84%9C%EC%9C%A8%EC%97%B0_%EA%B3%B5%EA%B0%84SQL/sql/emd_fire_pop_dist.sql

지하철역으로도 해 보자



IT'S YOUR HOMEWORK!

들어주셔서 감사합니다~

