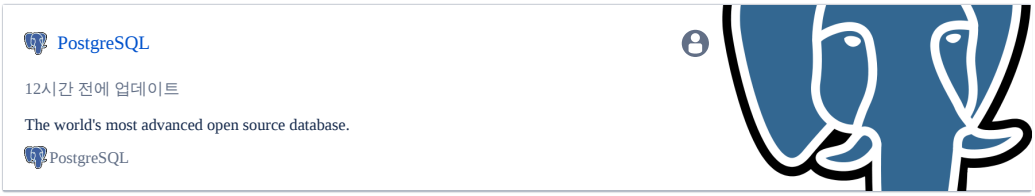
ILE3-032 PostgreSQL

OpenSource DBMS인 PostgreSQL에 대하여 학습

PostgreSQL



1996년 OpenSource로 출시된 DBMS

초기에는 BSD License이었으나, MIT License와 유사한 독립 라이센스를 사용 중

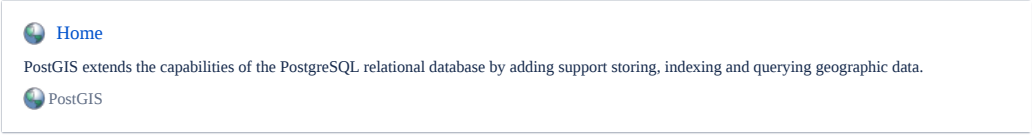
버클리 대학교에서 시작된 Ingres Project로 시작되었으며, Ingres의 문제들을 해결하기 위한 Post-Ingres Project를 시작하면서 1996년부터 SQL을 사용하기 시작하여 PostgreSQL이라는 이름으로 출시

2023년 현재 4위의 점유율을 가지고 있음(1위 Oracle, 2위 MySQL, 3위 MSSQL, 4위 PostgreSQL)

특징

- 객체형 DBMS의 특성
  연산자, 자료형, 함수, 자료형 변환자, 확장 기능 등의 데이터 베이스의 객체를 사용자가 임의로 만들 수 있는 기능을 지원
- 상속
  객체지향 언어에서 Class와 같이 테이블을 상속하여 하위 테이블을 마늘 수 있음
- 함수
  SQL 구문으로 만든 이외에도 C++. JAVA, Python 등에서 만든 함수의 호출도 지원

PostGIS



PostgreSQL의 특성을 이용하여 GIS에 대한 기능을 추가한 Spatial 확장 기능
OGC(Open Geospatial Consortium)의 표준 규격을 따

다른 DBMS(Oracle, MySQL)도 Spatial 확장 기능을 지원하나, PostGIS에 대비 지원이 부족
(지원하는 Spatial Function 개수의 경우 MySQL이 약 150종, PostGIS는 약 600종)

설치

Linux (ubuntu)

PostgreSQL Repository 설치 후 사용 중인 PostgreSQL에 맞는 PostGIS 버전 설치

Windows

EDB에서 제공하는 PostgreSQL Instance 설치 시 StackBuilder를 이용하여 설치

설치 이 후

사용할 Database에서 Extension의 추가 필요

```
CREATE EXTENSION postgis;
```

Geometry Column

PostgreSQL에서 GIS 데이터를 다루기 위한 Column을 지원

테이블 생성 시 아래와 같이 `geometry` Type을 이용하여 생성이 가능

```
CREATE TABLE roads (
  road_id SERIAL PRIMARY KEY,
  road_name VARCHAR(64),
  roads_geom geometry(LINESTRING, 3005)
);
```

Spatial Index

geometry Type에 대하여 R-Tree 기반의 Spatial Index를 지원하여 지리 데이터에 대한 빠른 검색을 지원

```
CREATE INDEX roads_gix ON roads USING GIST (roads_geom);
```

Geometry INSERT

Geometry 데이터의 저장을 위하여 PostGIS는 OGC 표준을 일부 수정한 EWKB 형태로 Geometry 데이터를 저장

> OGC 표준 WKB의 경우 해당 좌표에 대한 좌표계 정보를 저장하지 못하여 PostGIS에서는 좌표계 정보를 저장하기 위하여 WKB를 확장하여 EWKB를 사용

Table의 Geometry Type에 EPSG Code를 설정한 경우에는 INSERT 구문에서는 WKT 형태의 Geometry 정보를 사용 가능

```
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (1,'LINESTRING(191232 243118,191108 243242)','Jeff Rd');
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (2,'LINESTRING(189141 244158,189265 244817)','Geordie Rd');
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (3,'LINESTRING(192783 228138,192612 229814)','Paul St');
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (4,'LINESTRING(189412 252431,189631 259122)','Graeme Ave');
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (5,'LINESTRING(190131 224148,190871 228134)','Phil Tce');
INSERT INTO roads (road_id, roads_geom, road_name)
  VALUES (6,'LINESTRING(198231 263418,198213 268322)','Dave Cres');
```

Geometry Search

특정 범위에 있는 Geometry를 검색하기 위해서 해당 기능을 위한 함수 및 연산자를 제공

`&&` 연산자

지정된 Geometry와 Geometry의 MBR을 비교하여 해당 영역이 중첩되는 경우 TRUE를 반환`

```
SELECT ST_AsText(roads_geom) AS geom
FROM roads
```

```
WHERE
  roads_geom && ST_MakeEnvelope(191232, 243117,191232, 243119,312);
```

`ST_INTERSECT` 함수

지정된 Geometry와 Geometry를 비교하여 해당 영역이 중첩되는 경우 TRUE를 반환

```
SELECT road_id, road_name
FROM roads
WHERE ST_Intersects(roads_geom, ST_MakeEnvelope(191232, 243117,191232, 243119,312));
```

과제

1. PC에 Windows version의 PostgreSQL + PostGIS를 설치
2. 이전 과제인 POI Table CREATE 구문을 Geometry Type으로 변환
3. POI Table에 대한 Spatial Index Create 구문을 작성
4. 2,3에서 만들어진 Query를 이용하여 PostgreSQL DBMS에 Table 생성