PostGIS 2.0

한국어 사용자 설명서

지은이 PostGIS 프로젝트 운영위원회

펴낸이 한국오픈소스GIS포럼

PostGIS 2.0 한국어 사용자 설명서

이 책은 한국오픈소스 GIS 포럼의 지원을 받아 제작되었음을 알립니다.

초판 1 쇄 인쇄: 2013 년 11 월 8 일
초판 1 쇄 발행: 2013 년 11 월 13 일

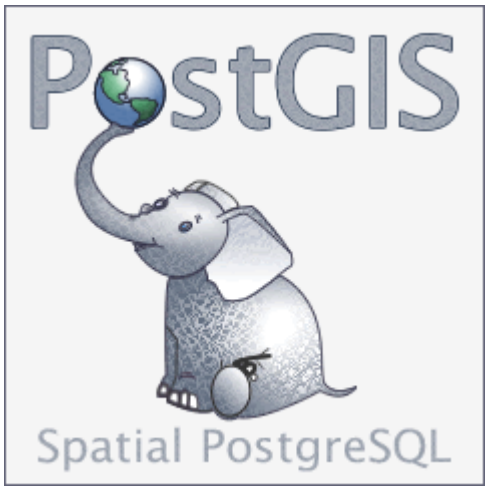
지은이: PostGIS 프로젝트 운영위원회
옮긴이: 조성룡
감수: 김경룡, 김기웅, 김지윤, 문수현, 박주용, 박희구, 신현범, 윤정환, 이동훈, 이민파, 이한진, 장병진, 장성희, 차갑상
표지 디자인: 신명순
편집: 김서인

펴낸이: 한국오픈소스 GIS 포럼
펴낸곳: 가이아쓰리디㈜
주소: 대전 유성구 관평동 1359 한신에스메카 230 호
전화: 042-330-0400
팩스: 042-330-0410
출판등록: 제 2012-000016 호
ISBN: 978-89-969532-1-0 (93000)

이 도서의 국립중앙도서관 출판시도서목록(CIP)은 서지정보유통지원시스템 홈페이지(http://seoji.nl.go.kr)와 국가자료공동목록시스템(http://www.nl.go.kr/kolisnet)에서 이용하실 수 있습니다.(CIP 제어번호: CIP2013022931)

Abstract

PostGIS is an extension to the PostgreSQL object-relational database system which allows GIS (Geographic Information Systems) objects to be stored in the database. PostGIS includes support for GiST-based R-Tree spatial indexes, and functions for analysis and processing of GIS objects.



This is the manual for version 2.0.5SVN

목 차

감사의 글
“PostGIS 2.0 Manual”의 한글화 작업에 참여하신 분들

김경룡, 김기웅, 김지윤, 문수현, 박주용, 박희구, 신현범, 윤정환, 이동훈, 이민파, 이한진, 장병진, 장성희, 조성룡, 차갑상

“PostGIS 2.0 Manual”을 “PostGIS 2.0 한국어 사용자 설명서”로 거듭나도록 기꺼이 참여해 주셔서 고맙습니다.

또한 “PostGIS 2.0 한국어 사용자 설명서”가 나올 수 있도록 아낌없는 지원을 해주신 한국오픈소스 GIS 포럼에도 감사의 인사를 남깁니다.

Chapter 1.소개

PostGIS 는 Refractions Research Inc. 에 의해 개발된 것으로 공간데이터베이스 기술 리서치 프로젝트입니다. Refractions 은 캐나다 브리티쉬 컬럼비아주 빅토리아시에 위치한 GIS 및 데이터베이스 컨설팅 회사로서 데이터 통합 및 고객 지향 소프트웨어 개발 전문 기업입니다. 우리는 전체 오픈 GIS 지원, 고급 위상기하적 구조 (범위, 표면, 네트워크), GIS 데이터를 편집하고 볼 수 있는 데스크 탑 사용자 인터페이스 도구들과 웹 기반 접근 도구들을 포함하는 다양한 중요 GIS 기능들을 지원하기 위한 PostGIS 를 지원, 개발할 예정입니다.

PostGIS 는 OSGeo 재단의 육성 프로젝트 입니다. PostGIS 는 지속적으로 성장하고 있으며 많은 FOSS4G 개발자들로부터 자금을 조달 받고 있습니다. 더하여, PostGIS 는 다양한 기능을 바탕으로 세계 도처에 있는 여러 기업들을 대상으로 이익을 창출하고 있습니다.

1.1. 프로젝트 운영 위원회

PostGIS 프로젝트 운영 위원회(PSC)는 총괄적 운영, 출시 주기, 문서화 그리고 PostGIS 프로젝트를 향상 시키려는 노력을 조정하는 역할을 합니다. 더하여, PSC 는 일반 사용자 지원을 제공하며 PostGIS 커뮤니티로부터의 패치들을 승인하고 수락하는 역할을 합니다. PSC 는 또한 개발자 커밋 엑세스 (developer commit access), PSC 신규 회원 또는 주요 API 변화에 관련된 다양한 이슈들을 해결하는 역할을 합니다.

Mark Cave-Ayland 버그 수정, 유지보수, PostGIS 와 PostgreSQL 출시와 PostGIS 간의 조정, 공간 지수 선별 및 결합, 로더/덤퍼(loader/dumper), Shape file GUI Loader, 새로운 기능 개선 제품들의 통합 담당.

Regina Obe 문서화, PostGIS 뉴스그룹의 일반 사용자 지원, 윈도우즈 생산 및 실험적 설계, X3D 지원, Tiger Geocoder 지원, 관리 기능, 그리고 새로운 기능 또는 주요 코드 변화에 관한 스모크 테스팅(Smoke testing)

Bborie Park 래스터 개발, GDAL 과의 통합, 래스터 로더(Raster Loader), 사용자 지원, 일반 버그 수정, 다양한 OS 환경 테스트(Slackware, Mac, Windows, 기타 등)

Paul Ramsey (Chair) PostGIS 프로젝트 공동 설립자. 일반 버그 수정, 지형 서포트, 지형 및 가하 인덱스 지원(2 차원, 3 차원, n 차원 인덱스 그리고 공간 인덱스 모두), 근본적인 기하 내부 구조, GEOS 기능, GEOS 릴리즈, 로더/덤퍼(Loader/Dumper)와 Shape 파일 GUI 로더 간의 정렬.

Sandro Santilli 버그 수정, 유지보수, 새로운 GEOS 기능의 통합 및 GEOS 출시 버전 과의 정렬, 위상 기하 지원, 래스터 프레임워크 및 저레벨 API 등

1.2. 과거와 현재 공헌자

Chris Hodgson 일반 개발, 사이트 그리고 빌드봇(buildbot) 유지보수, OSGeo 육성 프로젝트 관리

Kevin Neufeld 이전 PSC 회원. 문서화 작업과 문서화 지원 도구 개발, PostGIS 뉴스그룹의 고급 유저 지원, PostGIS 유지보수 기능 강화

Dave Blasby PostGIS 의 원 개발자이자 공동 창업자. 데이브는 서버측 객체, 인덱스 결합, 그리고 많은 서버 측의 분석적 기능들을 작업하였음.

Jeff Lounsbury Shape 파일 로더/덤퍼(loader/dumper)의 본래 개발자. 현재 PostGIS 프로젝트 소유자 대표

Olivier Courti XML(KML, GML)/GeoJSON 입출력 기능들, 3D 지원 및 버그 수정 (Input output XML (KML,GML)/GeoJSON functions, 3D support and bug fixes).

Mark Leslie 진행중인 유지보수 및 핵심 기능 개발. 강화된 곡선 지원. Shape 파일 GUI 로더.

Pierre Racine 전반적인 래스터 아키텍처, 프로토타이핑, 프로그래밍 지원

Nicklas Avén 거리 함수 향상(3D 거리 및 관계 함수 포함), 윈도우즈 테스트 및 일반 유저 지원

Jorge Arévalo 래스터 개발, GDAL 드라이버 지원, 로더

Mateusz Loskot 래스터 로더, 저레벨 래스터 API 함수

David Zwarg 래스터 개발

기타 공헌자: 개인 알파벳 순: Alex Bodnaru, Alex Mayrhofer, Andrea Peri, Andreas Forø Tollefsen, Andreas Neumann, Anne Ghisla, Barbara Phillipot, Ben Jubb, Bernhard Reiter, Brian Hamlin, Bruce Rindahl, Bruno Wolff III, Bryce L. Nordgren, Carl Anderson, Charlie Savage, Dane Springmeyer, David Skea, David Techer, Eduin Carrillo, Even Rouault, Frank Warmerdam, George Silva, Gerald Fenoy, Gino Lucrezi, Guillaume Lelarge, IIDA Tetsushi, Ingvild Nystuen, Jeff Adams, Jose Carlos Martinez Llari, Kashif Rasul, Klaus Foerster, Kris Jurka, Leo Hsu, Loic Dachary, Luca S. Percich, Maria Arias de Reyna, Mark Sondheim, Markus Schaber, Maxime Guillaud, Maxime van Noppen, Michael Fuhr, Nikita Shulga, Norman Vine, Rafal Magda, Ralph Mason, Richard Greenwood, Silvio Grosso, Steffen Macke, Stephen Frost, Tom van Tilburg, Vincent Picavet

기타 공헌자: 기업 스폰서 PostGIS 프로젝트에 직접적인 금전을 지원하거나, 개발 시간 및 호스팅에 기여를 한 기업들. 알파벳 순: Arrival 3D, Associazione Italiana per l'Informazione Geografica Libera (GFOSS.it), AusVet, Avencia, Azavea, Cadcorp, CampToCamp, City of Boston (DND), Clever Elephant

Solutions, Cooperativa Alveo, Deimos Space, Faunalia, Geographic Data BC, Hunter Systems Group, Lidwala Consulting Engineers, LisaSoft, Logical Tracking & Tracing International AG, Michigan Tech Research Institute, Norwegian Forest and Landscape Institute, OpenGeo, OSGeo, Oslandia, Paragon Corporation, R3 GIS,, Refractions Research, Regione Toscana-SIGTA, Safe Software, Sirius Corporation plc, Stadt Uster, UC Davis Center for Vectorborne Diseases, University of Laval, U.S Department of State (HIU), Vizzuality, Zonar Systems

크라우드 펀딩 캠페인 크라우드 펀딩 캠페인은 많은 수의 사람들을 지원할 수 있는 기능 개발에 필요한 자금을 조달하기 위한 것입니다. 각 자금은 특정 기능 혹은 기능들의 세트에 특정적으로 집중되어 있습니다. 충분한 지원자들과 지원기관들의 기여와 함께 필요한 자금의 작은 부분에 속하는 각 스폰서 칩들(sponsor chips)로 많은 부분에 도움을 줄 업무에 필요한 자금을 조달할 수 있습니다. 많은 사람들이 공동출자할 의향을 파악할 수 있는 좋은 아이디어가 있다면 PostGIS 뉴스그룹에 의견을 올려 주세요. 이 아이디어는 현실화될 수 있습니다.

PostGIS 2.0.0 은 이러한 전략 아래 탄생된 첫 번째 출시작입니다. 저희는 PledgBank 를 이용했으며 이를 통해 두 번의 성공적인 캠페인을 진행.

Postgistropology – 10 명 이상의 스폰서들. toTopGeometry 기능 구축 및 2.0.0 버전에서 위상 기하 기능 지원 강화를 위해 개개인이 250 달러씩 지원함.

Postgis6windows – 20 명 이상의 스폰서들. PostGIS 64 비트 윈도우 이슈 관련 해결을 위한 업무에 필요한 금액 지원을 위해 각 100 달러씩 지원. 현재 PostGIS 2.0.0 를 위한 64 비트 베타 버전이 출시되어 있으며, 향후 최종 출시 버전이 PostgreSQL stack builder 에서 이용 가능함.

라이브러리 중요한 지원 GEOS(지오메트리 운영 라이브러리)와 모든 것을 작동하도록 만든 Martin Davis 의 알고리즘 작업, Mateouz Losk, Sandro Santilli(strk), Paul Ramsey 외 다른 이들의 진행중인 유지보수 및 지원 작업.

GDAL(공간지리 데이터 추출 라이브러리) – Fracnk Warmerda 와 다른 이들에 의하여 PostGIS 2.0.0 에 소개된 래스터 기능을 작동시키기 위해 사용되었음. 같은 방식으로 PostGIS 를 지원하기 위해 필요한 GDAL 의 개선부분들은 GDAL 프로젝트에 재 기여됨.

Proj4(좌표변환 지도제작 영상 라이브러리)와 이를 창조하고 유지하기 위한 Gerald Evenden 와 Frank Warmerdam 의 작업.

마지막으로 그러나 역시 중요한, PostGIS 가 기초하는 PostgreSQL DBMS. PostGIS 의 대단한 유동성과 속도는 확장성, 강력한 쿼리 플래너, GIST 인덱스, 그리고 PostgreSQL 에서 제공되는 다양한 SQL 기능들이 없이는 불가능 했음.

1.3. 추가 정보

- 최신 소프트웨어, 문서 그리고 새로운 정보들은 http://postgis.net 에서 확인 가능
- GEOS 기하 운영 라이브러리에 대한 상세 정보는 http://trac.osgeo.org/geos/ 에서 확인 가능
- Proj4 재투영 라이브러리에 대한 상세 정보는 http://trac.osgeo.org/proj/ 에서 확인 가능

- PostgreSQL 데이터베이스 서버에 대한 상세 정보는 PostgreSQL 메인 사이트 http://www.postgresql.org 에서 확인 가능
- MapServer 인터넷 지도 서버에 관한 추가 정보는 http://mapserver.org 에서 확인 가능
- "Simple Features for Specification for SQL"은 OpenGIS Consortium 웹 사이트: http://www.opengeospatial.org/ 에서 확인 가능

Chapter 2.설치

이 장에서는 PostGIS 를 설치하기 위해 요구되는 모든 과정들을 설명합니다.

2.1. 짧은 설명

래스터 지원은 현재 선택 사항이지만, 기본으로 설치됩니다. 설치를 위해서는 PostgreSQL 9.1 이상 버전 및 extension 이 필요합니다. PostgreSQL 9.1 이상 버전 사용자는 2.4.3 'PostGIS extensions 생성 및 배치'를 참조해주십시오.

모든 .sql 파일들은 `share/contrib/postgis-2.0` 폴더에 설치됩니다.

`postgis_comments.sql`, `raster_comments.sql`, `topology_comments.sql` 은 pgAdmin III 또는 psql 을 통해 접근할 수 있는 각 기능에 대한 빠른 도움말을 생성합니다. psql 에서 command 에 `\dd ST_SetPoint` 와 같은 식으로 이용할 수 있습니다.

```
tar xvfz postgis-2.0.5SVN.tar.gz
cd postgis-2.0.5SVN
./configure --with-raster --with-topology --with-gui
make
make install
createdb yourdatabase
createlang plpgsql yourdatabase
psql -d yourdatabase -f postgis.sql
psql -d yourdatabase -f postgis_comments.sql
psql -d yourdatabase -f spatial_ref_sys.sql
psql -d yourdatabase -f rtpostgis.sql
psql -d yourdatabase -f raster_comments.sql
psql -d yourdatabase -f topology/topology.sql
psql -d yourdatabase -f doc/topology_comments.sql
```

topology_comments.sql 은 선택 요소이므로 기본 설치되지 않습니다. make comments 또는 make topology_comments.sql 을 실행하실 경우 topology_comments.sql 가 docs 폴더에 생성될 것입니다.

이 장 나머지 부분에서는 위에서 설명된 각 설치 순서에 관한 자세한 정보를 다룹니다.

2.2. 요구 사양

PostGIS 생성과 사용을 위해서는 다음의 사양들이 요구됩니다.

요구사항

- PostgreSQL 8.4 - 9.2. Minor 버전. PostgreSQL 의 완전 설치 (서버 헤더 포함)가 요구됨. PostgreSQL 은 http://www.postgresql.org 에서 이용 가능
  전체 PostgreSQL/PostGIS 기능 지원표 및 PostGIS/GEOS 기능 지원표는 http://trac.osgeo.org/postgis/wiki/UsersWikiPostgreSQLPostGIS 를 참조하십시오.

- GNU C compiler (`gcc`). 그 밖에 다른 ANSI C complier 들이 PostGIS 를 컴파일하기 위해 사용될 수 있으나 `gcc` 로 컴파일 했을 경우 훨씬 적은 오류가 발생됩니다.

- GNU Make(`gmake` 또는 `make`). 많은 시스템들에서 GNU `make` 는 `make` 의 기본 버전입니다. `make -v` 를 통해 버전을 확인하십시오. 다른 `make` 버전들의 경우 PostGIS `Makefile` 을 제대로 처리하지 못할 수 도 있습니다.

- Proj4 투영 및 좌표변환 라이브러리 4.6.0 또는 그 이상 버전.
  Proj4 라이브러리는 PostGIS 내에서 투영 및 좌표변환을 제공하기 위해서 사용됩니다. proj4 는 http://trac.osgeo.org/proj/에서 다운로드 가능합니다.

- GEOS 기하학 라이브러리(geometry library)는 3.2.2. 또는 그 이상 버전이 가능하고, GEOS 3.3.2+ 이상 버전을 권장합니다. GEOS 3.3 이 없으면 위상 예외 처리, ST_ValidDetail 및 ST_MakeValid 와 같은 기하 검보정 향상 등과 같은 주요한 기능이 누락될 수 있습니다. GEOS 3.3.2 는 토폴로지 지원(Topology support)을 위해서도 필요합니다. GEOS 는 http://trac.osgeo.org/geos/로부터 다운로드 가능하며 3.3 또는 그 이상 버전은 구 버전들과 호환이 됩니다.

- LibXML2, version 2.5.x 또는 그 이상 버전. LibXML 은 몇 가지 imports 기능 (ST_GeomFromGML 및 ST_GeomFromKML)에 사용되고 있습니다. LibXML2 는 http://xmlsoft.org/downloads.html 에서 다운로드 가능합니다.

- JSON-C, version 0.9 버전 또는 그 이상 버전. JSON-C 는 ST_GeomFromGeoJson 을 통해 GeoJSON 을 불러오기 위해 사용되고 있습니다. JSON-C 는 http://oss.metaparadigm.com/json-c/에서 다운로드 가능합니다.

- GDAL, version 1.6 또는 그 이상 버전(1.9 또는 그 이상 버전이 선호됩니다. 그 보다 낮은 버전에서는 작동되지 않는 것들이 존재합니다.) 이것은 래스터 지원은 물론 `CREATE EXTENSION postgis` 설치를 가능하게 하기 때문에 래스터가 필요하지 않더라도 설치를 추천합니다. `http://trac.osgeo.org/gdal/wiki/DownloadSource`

선택사항

- GDAL 은 래스터나 `CREATE EXTENSION postgis` 가 필요 없을 경우에는 선택하실 필요가 없습니다. 다만, 다른 확장(extension) 기능에서 postgis extension 을 필요로 할 수 있으므로 GDAL 설치를 매우 권장합니다.
- GTK(GTK+2.0 (2.8 이상)버전을 요구합니다.) hp2pgsql-gui shape file 로더를 컴파일 하기 위함. http://www.gtk.org
- CUnit(`CUnit`). 이는 회귀검증(regression testing)을 위해서 필요합니다. http://cunit.sourceforge.net/
- Apache Ant(`ant`)는 java 디렉토리 내의 드라이버를 활용하기 위해 필요합니다. Ant 는 http://ant.apache.org 에서 이용 가능합니다.
- DocBook(`xsltproc`)은 문서를 발행할 때 필요합니다. DocBook 은 http://www.docbook.org/에서 이용 가능합니다.
- DBLatex(`dblatex`)는 PDF 형식의 문서를 만들 때 필요합니다. DBLatex 는 http://dblatex.sourceforge.net/에서 이용 가능합니다.
- ImageMagick(`convert`)는 설명서에 사용된 이미지를 생성하는 데 필요합니다. ImageMagick 는 http://www.imagemagick.org/에서 이용 가능합니다.

2.3. 소스 획득

PostGIS 소스를 http://postgis.net/stuf/postgis-2.0.4SVN.tar.gz 에서 다운로드 하십시오.

```
wget http://postgis.net/stuff/postgis-2.0.4SVN.tar.gz
tar -xvzf postgis-2.0.4SVN.tar.gz
```

현재 작업 디렉토리에 `postgis-2.0.5SVN` 라는 디렉토리를 생성할 것입니다.

다른 방법으로, svn 저장소(repository)인 http://svn.osgeo.org/postgis/branches/2.0 에서 소스를 받을 수 있습니다.

```
svn checkout http://svn.osgeo.org/postgis/branches/2.0/ postgis-2.0.5SVN
```

설치를 계속하려면 새로 만든 `postgis-2.0.5SVN` 디렉토리로 변경합니다.

2.4. 설치

많은 OS 시스템들은 현재 PostgreSQL/PostGIS 를 위한 사전 설치된 패키지들을 포함하고 있습니다. 많은 경우에, 최첨단 버전이 필요하시거나 꼭 패키지를 사용해야 할 상황이 아니라면 패키지가 필요하지 않습니다. 이 단원에서는 일반적 설명을 하고 있으므로, 더 상세한 설명은 원한다면 PostGIS User contributed compile/install guides 와 PostGIS Dev Wiki 를 참조하십시오. 다양한 OS 를 지원하는 사전 패키지들은 PostGIS Pre-built Packages 에 열거되어 있습니다. 만약 윈도우 사용자라면 Stackbuilder 또는 PostGIS Windows download site 을 통해 안정적인 버전을 받을 수 있습니다. 또한 1~2 주에 한번씩은 최신 버전을 제공하고 있습니다.

PostGIS 모듈은 PostgreSQL 의 확장 기능입니다. PostGIS 2.0.5SVN 은 컴파일 하기 위해 full PostgreSQL 서버 헤더(server headers) 접근을 필요로 합니다. 이는 PostgreSQL 8.4 버전 또는 이상 버전에서 설치됩니다. 그 이전 버전의 PostgreSQL 에서는 지원되지 않습니다.

아직 PostgreSQL 을 설치하지 않으셨다면 PostgreSQL 설치 가이드를 참조하십시오. http://www.postgresql.org

GEOS 기능을 위해서는 PostgreSQL 설치 시 표준 c++ 라이브러리에 대한 확실한 연결이 필요할 수도 있습니다.

```
LDFLAGS=-lstdc++ ./configure [YOUR OPTIONS HERE].5SVN
```

기존 개발 도구에서의 예외적인 상호작용을 해결하는 방법으로, 연결이 안 되는 등의 문제가 생겼을 때 써볼 수 있는 방법입니다. 이래도 안되면 PostgreSQL 을 다시 컴파일 해야 할 수도 있습니다.

다음의 단계들은 PostGIS 소스에 대한 설정과 편집에 대한 과정을 설명합니다. 이는 리눅스 사용자들을 위한 것이므로 윈도우 또는 맥은 해당되지 않습니다.

2.4.1. 리눅스 설치를 위한 설정(configuration)

리눅스 설치시 첫번째 단계는 소스 코드를 이용하여 Makefile (생성파일)을 생성하는 것입니다. Makefile 은 쉘 스크립트 실행을 통해 만들어집니다.

./configure

이 명령은 다른 추가 매개 변수(Parameter)없이 자동적으로 PostGIS 소스 코드를 시스템에 설치하기 위해 필요한 필수 구성 요소들과 라이브러리를 설치하기 위한 시도를 합니다. **./configure** 명령만으로 필요한 라이브러리와 비 표준 경로에 설치된 프로그램을 찾아냅니다.

아래 리스트는 가장 일반적으로 사용되는 매개 변수(parameters)들만을 나열한 것입니다. 전체 목록을 원하신다면 **--help** 또는 **--help=short** 를 사용하십시오.

--prefix=PREFIX 이는 PostGIS 라이브러리와 SQL 스크립트들이 설치될 경로입니다. 기본적으로 PostgreSQL 가 설치된 경로와 같은 곳에 설치됩니다.

이 파라미터는 현재 작동되지 않습니다. 따라서, 현재는 PostgreSQL 설치 경로에만 설치될 것입니다. 해당 버그 추적을 위해서는 http://trac.osgeo.org/postgis/ticket/635 를 참조하십시오.

--with-pgconfig=FILE PostgreSQL 은 pg_config 라 불리는 유틸리티를 제공합니다. pg_config 는 PostGIS 와 같은 extension 들이 PostgreSQL 설치 디렉토리의 정확한 위치를 찾는 것을 가능케 합니다. PostGIS 를 기반으로 설치될 특정 PostgreSQL 설치를 수동으로 명시하기 위해서는 매개변수 (**--with-pgconfig=/path/to/pg_config**)를 사용하십시오.

--with-gdalconfig=FILE 권장 라이브러리인 GDAL 설치 경로를 찾습니다. GDAL 은 래스터 지원에 필요한 기능을 제공합니다. PostGIS 를 기반으로 설치될 특정 GDAL 설치를 수동으로 명시하기 위해서는 매개변수**(--with-gdalconfig=/path/to/gdal-config)**를 사용하십시오.

--with-geosconfig=FILE 권장 라이브러리인 GEOS 는 geos-config 라 불리는 유틸리티를 제공합니다. geos-config 는 GEOS 설치 경로를 찾습니다. PostGIS 를 기반으로 GEOS 설치를 수동으로 명시하기 위해서는 매개변수(**--with-geosconfig=/path/to/geos-config**)를 사용하십시오.

--with-xml2config=FILE LibXML 은 GeomFromKML/GML 프로세스를 진행하기 위해 필요한 라이브러리입니다. 이것은 Libxml 을 설치하면 찾을 수 있습니다. Libxml 을 설치하지 않았거나 특정 버전을 원하신다면 PostGIS 를 xml2-config 라고 하는 설정 파일에 명시해야 합니다. xml2-config 설정 파일은 소프트웨어가 LibXML 설치 경로에 위치할 수 있도록 합니다. PostGIS 에 사용할 특정 LibXML 설치를 수동으로 명시하기 위해서는 매개변수(**>--with-xml2config=/path/to/xml2-config**)를 사용하십시오.

--with-projdir=DIR Proj4 는 투영 및 좌표변환 라이브러리입니다. PostGIS 를 기반으로 설치될 특정 Proj4 설치를 수동으로 명시하기 위해서는 매개변수 (**--with-projdir=/path/to/projdir**)를 사용하십시오.

--with-libiconv=DIR Iconv 가 설치되는 경로

--with-jsondir=DIR JSON-C 는 MIT-라이센스 JSON 라이브러리로 PostGIS ST_GeomFromJSON 지원에 필요합니다. PostGIS 를 기반으로 설치될 특정 JSON-C 설치를 수동으로 명시하기 위해서는 매개변수 **(--with-jsondir=/path/to/jsondir)**를 사용하십시오.

--with-gui 데이터 불러오기 GUI (GTK 2.0 이상 버전 필요)를 이용할 때 적용합니다. shp2pgsql 을 이용할 때 shp2pgsql-gui 화면 인터페이스를 생성합니다.

--with-raster 래스터 지원을 필요로 할 때 이용합니다. 이것은 rtpostgis-2.0.5SVN 라이브러리와 rtpostgis.sql 파일을 실행할 것입니다. 최종 출시 제품에서는 래스터 지원을 기본으로 지원할 것이기 때문에 필요로 하지 않을 것입니다.

--with-topology 토폴로지 지원을 필요로 할 때 이용하며, topology.sql 파일을 실행할 것입니다. 이것에 상응하는 다른 라이브러리는 postgis-2.0.5SVN 라이브러리 내에 존재하지 않습니다.

--with-gettext=no 기본적으로 PostGIS 는 gettext(다국어 처리 오픈소스)를 지원하고 이와 호환되지만, 로더 손상으로 일부 호환 불가능 문제가 있을 수 있습니다. 이런 경우 이 명령어로 gettext 를 미사용으로 설정할 수 있습니다. 자세한 내용은 http://trac.osgeo.org/postgis/ticket/748 을 참조하십시오. 이것은 GUI 를 불러올 때 다국어 처리를 위한 것으로 미사용으로 설정한다고 해서 크게 놓치는 것은 없습니다.

PostGIS 를 SVN repository 에서 획득하셨다면, 먼저 아래 스크립트를 실행하십시오.
./autogen.sh
이 스크립트는 **configure** 스크립트를 생성해줍니다.
만약 tarball(유닉스 tar 유틸리티로 만든 압축파일) 형태로 PostGIS 를 획득하셨다면 이미 **configure** 가 생성되었기 때문에 **./autogen.sh** 실행은 필요하지 않습니다.

2.4.2. 생성(Building)

한번 Makefile 이 생성되면 PostGIS 생성은 매우 쉽습니다.

make

이 때 결과 표시 마지막 줄에 "`PostGIS was built successfully. Ready to install.`"이 나타나야 합니다.

PostGIS v1.4.0 을 기준으로 모든 기능들은 문서화를 통해 생성된 주석(comments)을 가지고 있습니다. 공간 데이터베이스에 주석을 넣고 싶으시다면 docbook 을 필요로 하는 명령어를 실행시키십시오. postgis_comments.sql 과 다른 패키지 주석 파일들인 raster_comments.sql, topology_comments.sql 들은 doc 폴더 내 tar.gz 배포 파일에 패키징 되어 있으므로, tar ball 로부터 설치를 한 경우에는 따로 주석을 만들 필요가 없습니다.

make comments

이것은 빠른 참조(quick reference) 또는 학습용 유인물에 적합한 참조 자료 html 을 생성합니다. 생성을 위해서는 xsltproc 가 요구되며 doc 폴더 내에 `topology_cheatsheet.html`, `tiger_geocoder_cheatsheet.html`, `raster_cheatsheet.html`, `postgis_cheatsheet.html` 4 개의 파일을 생성합니다.

html 과 pdf 형식으로 미리 만들어진 파일들을 PostGIS / PostgreSQL Study Guides 에서 다운로드 받으실 수 있습니다.

make cheatsheets

2.4.3. PostGIS extensions 생성 및 배치

PostgreSQL 9.1 이상을 사용 중이라면 PostGIS extentions 은 자동적으로 생성 및 설치됩니다.

소스 저장소(source repository)로부터 생성할 경우에는, 먼저 기능 설명(function descriptions) 부터 생성해야 합니다. docbook 을 설치하셨다면 기능 설명(function descriptions)은 생성이 되고, 다음 명령을 통해 수동으로 생성할 수도 있습니다.

```
make comments
```

이미 tar ball 과 함께 미리 패키지화된 것들이므로 tar ball 로부터 설치 한다면 따로 주석(comments)을 생성할 필요가 없습니다.

만약 PostgreSQL 9.1 을 바탕으로 생성 중이라면 extensions 은 설치 과정의 일환으로 자동 생성될 것입니다. extensions 폴더로부터의 생성 또는 파일 복사를 통해 다른 서버에 적용할 수 있습니다.

```
cd extensions
cd postgis
make clean
make
make install
cd ..
cd postgis_topology
make clean
make
make install
```

확장(extension) 파일은 OS 에 상관없이 PostGIS 버전만 같으면 적용에 문제가 없습니다. 그러므로 PostGIS binaries 가 설치된 서버에 확장 파일만 복사해도 문제가 없습니다.

만약 extension 을 수동으로 또는 다른 서버에 설치하고 싶으면 확장 파일들을 PostgreSQL / share / extension 폴더에 복사하십시오.

- 여기에는 확장 파일의 버전 등에 대한 정보를 나타내는 컨트롤 파일들이 존재합니다. postgis.control, postgis_topology.control.
- 각 extension 의 /sql 폴더 내에 모든 파일들이 있습니다. 이 파일들은 postgreSQL 의 share/extension 폴더 extensions/postgis/sql/*.sql, extensions/postgis_topology/sql/*.sql 에 복사되어야 합니다.

다음 PgAdmin -> extensions 에서 postgis, postgis_topology 가 보여져야 합니다.

만약 psql 을 이용 중이라면 다음의 쿼리(query)를 실행함으로써 확인할 수 있습니다.

```
SELECT name, default_version,installed_version
FROM pg_available_extensions WHERE name LIKE 'postgis%' ;
     name          | default_version | installed_version
-----------------+-----------------+-------------------
postgis            | 2.0.5SVN        | 2.0.5SVN
postgis_topology  | 2.0.5SVN        |
```

쿼리로 데이터베이스 내 extension 을 확인할 때, installed_version 항목(column)을 언급할 수 있습니다. 만약 아무 레코드가 없다면 서버에 postgis extension 이 전혀 설치되어 있지 않음을 뜻합니다. PgAdmin III 1.14 이상 버전에서는 데이터베이스 브라우저 트리의 extension 에서 마우스 오른쪽 버튼 클릭을 통해 업그레이드 또는 삭제를 허용합니다.

extension 이 이용 가능한 상태라면 pgAdmin extension 인터페이스 또는 다음의 sql 명령을 실행함으로써 선택한 데이터베이스 안에 postgis extension 을 설치할 수 있습니다.

```
CREATE EXTENSION postgis;
CREATE EXTENSION postgis_topology;
```

tables spatial_ref_sys, layer, topology 확장 테이블은 백업되지 않습니다. 이것들은 postgis 또는 postgis_topolgy extension 이 백업이 될 경우에만 백업이 가능합니다. PosgGIS 2.0.1 에서는 데이터베이스 백업시 srid 레코드만이 백업됩니다. 이와 관한 문제를 발견하면 trac 티켓을 발행해주십시오. extension 테이블의 구조들은 CREATE EXTENSION 과 함께 생성되기 때문에 백업되지 않습니다. 이러한 방식은 PostgreSQL extension 모델에 적용되기 때문에 조치를 취할 수 있는 방법이 없습니다.

만약 extension 없이 2.0.5SVN 을 설치했다면, 업그레이드 스크립트: postgis_upgrade_20_minor.sql,raster_upgrade_20_minor.sql,topology_upgrade_20_minor.sql 실행을 통해 최신 버전으로 업그레이드할 수 있습니다.

만일 래스터 지원없이 postgis 를 설치했다면, `rtpostgis.sql` 을 사용하여 래스터 지원 설치를 할 수 있습니다.

그런 다음 extension 기능 패키지를 설치하기 위한 다음 명령을 실행할 수 있습니다.

```
CREATE EXTENSION postgis FROM unpackaged;
CREATE EXTENSION postgis_topology FROM unpackaged;
```

2.4.4. 테스트

PostGIS 생성(build)을 테스트하고 싶다면 아래 명령을 실행하십시오.

Make check

위 명령어는 실제 PostgreSQL 데이터베이스 바탕으로 생성된 라이브러리를 이용하여 다양한 확인과 회귀 테스트를 실행할 것입니다.

PostgreSQL, GEOS, 또는 Proj4 를 표준이 아닌 경로에 설치한 경우, LD_LIBRARY_PATH 환경 변수에 해당 라이브러리 경로를 설정해주어야 합니다.

현재, **make check** 검사들을 실시할 때에는 `PATH` 와 `PGPORT` 환경 변수를 따릅니다. PostgreSQL 의 설정 매개변수인 **--with-pgconfig** 에 명시되어 있어도 이것을 적용하지 않습니다. 따라서, PostgreSQL 설치시의 환경설정과 일치하도록 PATH 를 수정해주십시오.

성공하였다면 테스트의 결과는 아래와 비슷하게 나올 것입니다.

```
         CUnit - A Unit testing framework for C - Version 2.1-0
        http://cunit.sourceforge.net/

Suite: print_suite
  Test: test_lwprint_default_format ... passed
  Test: test_lwprint_format_orders ... passed
  Test: test_lwprint_optional_format ... passed
  Test: test_lwprint_oddball_formats ... passed
  Test: test_lwprint_bad_formats ... passed
Suite: Misc Suite
  Test: test_misc_force_2d ... passed
  Test: test_misc_simplify ... passed
  Test: test_misc_count_vertices ... passed
  Test: test_misc_area ... passed
  Test: test_misc_wkb ... passed
Suite: PointArray Suite
  Test: test_ptarray_append_point ... passed
  Test: test_ptarray_append_ptarray ... passed
Suite: PostGIS Computational Geometry Suite
```

```
  Test: test_lw_segment_side ... passed
  Test: test_lw_segment_intersects ... passed
  Test: test_lwline_crossing_short_lines ... passed
  Test: test_lwline_crossing_long_lines ... passed
  Test: test_lwline_crossing_bugs ... passed
  Test: test_lwpoint_set_ordinate ... passed
  Test: test_lwpoint_get_ordinate ... passed
  Test: test_point_interpolate ... passed
  Test: test_lwline_clip ... passed
  Test: test_lwline_clip_big ... passed
  Test: test_lwmline_clip ... passed
  Test: test_geohash_point ... passed
  Test: test_geohash_precision ... passed
  Test: test_geohash ... passed
  Test: test_isclosed ... passed
Suite: PostGIS Measures Suite
  Test: test_mindistance2d_tolerance ... passed
  Test: test_rect_tree_contains_point ... passed
  Test: test_rect_tree_intersects_tree ... passed
  Test: test_lwgeom_segmentize2d ... passed
Suite: WKT Out Suite
  Test: test_wkt_out_point ... passed
  Test: test_wkt_out_linestring ... passed
  Test: test_wkt_out_polygon ... passed
  Test: test_wkt_out_multipoint ... passed
  Test: test_wkt_out_multilinestring ... passed
:
:
--Run Summary: Type     Total    Ran  Passed  Failed
            suites      17     17    n/a      0
            tests      143    143    143      0
            asserts   1228   1228   1228      0

Creating spatial db postgis_reg
 Postgis 2.0.0SVN - 2011-01-11 15:33:37
   GEOS: 3.3.0-CAPI-1.7.0
   PROJ: Rel. 4.6.1, 21 August 2008

Running tests

 loader/Point.............. ok
 loader/PointM............. ok
 loader/PointZ............. ok
 loader/MultiPoint............. ok
 loader/MultiPointM............. ok
 loader/MultiPointZ............. ok
 loader/Arc.............. ok
 loader/ArcM............. ok
 loader/ArcZ.......... ok
 loader/Polygon............. ok
 loader/PolygonM............. ok
 loader/PolygonZ............. ok
 regress. ok
 regress_index. ok
 regress_index_nulls. ok
 lwgeom_regress. ok
 regress_lrs. ok
```

```
 removepoint. ok
 setpoint. ok
 simplify. ok
 snaptogrid. ok
 affine. ok
 measures. ok
 long_xact. ok
 ctors. ok
 sql-mm-serialize. ok
 sql-mm-circularstring. ok
 sql-mm-compoundcurve. ok
 sql-mm-curvepoly. ok
 sql-mm-general. ok
 sql-mm-multicurve. ok
 sql-mm-multisurface. ok
 polyhedralsurface. ok
 out_geometry. ok
 out_geography. ok
 in_gml. ok
 in_kml. ok
 iscollection. ok
 regress_ogc. ok
 regress_ogc_cover. ok
 regress_ogc_prep. ok
 regress_bdpoly. ok
 regress_proj. ok
 dump. ok
 dumppoints. ok
 wmsservers_new. ok
 tickets. ok
 remove_repeated_points. ok
 split. ok
 relatematch. ok
 regress_buffer_params. ok
 hausdorff. ok
 clean. ok
 sharedpaths. ok
 snap. ok

Run tests: 55
Failed: 0
```

2.4.5. 설치

PostGIS 설치를 위해서는 다음을 입력하십시오.

make install

이것은 **--prefix** 설정 파라미터에 정의된 하위 경로에 PostGIS 설치 파일을 복사할 것입니다.

- 로더(loader)와 덤퍼(dumper) binaries 들은 `[prefix]/bin.`에 설치됩니다.
- postgis.sql 와 같은 SQL 파일들은 `[prefix]/share/contrib` 에 설치됩니다.

- PostGIS 라이브러리들은 `[prefix]/lib` 에 설치됩니다.

만약 기존에 postgis_comments.sql, raster_comments.sql 파일을 생성하기 위해 **make comments** 명령어를 실행한 적이 있으시다면 make comments-install 실행을 통해 sql 파일을 설치하십시오.

make comments-install

Xsltproc 의 적용 이후 일반적인 설치로부터 postgis_comments.sql, raster_comments.sql, topology_comments.sql 는 분리되었습니다.

2.5. PostgreSQL 9.1 미만 버전에 공간정보가 적용 데이터베이스 생성하기

PostGIS 데이터베이스를 생성하기 위한 첫 번째 절차는 간단한 PostgreSQL 데이터베이스를 생성하는 것입니다.

createdb [yourdatabase]

많은 PostGIS 기능들은 PL/pgSQL procedural 언어로 쓰여져 있습니다. 따라서, 새로운 데이터베이스에 PL/pgSQL 을 활성화하는 것이 PostGIS 데이터베이스를 생성하기 위한 두 번째 단계입니다. 이는 아래 명령어를 통해 가능하고, PostgreSQL 8.4 이상 버전의 경우에는 일반적으로 이미 설치되어 있습니다.

createlang plpgsql [yourdatabase]

이제 postgis.sql definitions 파일을 로딩함으로써 (환경설정시 `[prefix]/share/contrib` 에 경로 설정) PostGIS 객체 및 정의 함수(function definition)를 데이터베이스에 로딩하십시오.

psql -d [yourdatabase] -f postgis.sql

EPSP 좌표계 정의 식별자의 완전한 세팅을 위해서는 `spatial_ref_sys.sql` 정의 파일을 로드할 수 있으며 `spatial_ref_sys` 테이블을 덧붙일 수 있습니다. 이것은 ST_Transform() 기능을 수행할 수 있게 해줍니다.

psql -d [yourdatabase] -f spatial_ref_sys.sql

주석(comments)을 PostGIS 기능에 추가하고 싶은 경우에는 공간 데이터베이스에 `postgis_comments.sql` 을 로딩하는 것이 첫 번째 과정입니다. 주석은 **psql** 터미널 창에서 **\dd [function_name]**을 입력하여 조회할 수 있습니다.

psql -d [yourdatabase] -f postgis_comments.sql

래스터 지원 설치

psql -d [yourdatabase] -f rtpostgis.sql

래스터 지원 주석(comments) 설치. Psql, PgAdmin 또는 다른 PostgreSQL 도구에서 각 래스터 기능을 위한 신속한 도움말을 제공합니다.

psql -d [yourdatabase] -f raster_comments.sql

토폴로지 지원 설치

psql -d [yourdatabase] -f topology/topology.sql

토폴로지 지원 주석 설치. Psql, PgAdmin 또는 다른 PostgreSQL 도구에서 토폴로지(위상관계 처리) 기능을 위한 신속한 도움말을 제공합니다.

psql -d [yourdatabase] -f topology/topology_comments.sql

만약 새로운 DB 에 기존 버전의 백업을 복원하고 싶다면 다음을 실행하십시오.

psql -d [yourdatabase] -f legacy.sql

대안으로 `legacy_minimal.sql` 이 있습니다. 이는 테이블을 복원하고 MapServer 와 Geoserver 와 같은 소프트웨어와 연동하는 등의 설치를 대신하여 실행할 수 있습니다. 거리 / 길이 등과 같은 뷰(view)들을 가지고 계신다면 완전한 legacy.sql 이 필요합니다.

복구와 청소(cleanup)를 마친 뒤 앞으로 사용하지 않을 기능들을 제거하기 위해서는 `uninstall_legacy.sql` 을 실행하십시오.

2.6. EXTENSIONS 을 활용한 공간 데이터베이스 생성

Postgre 9.1 이상 버전을 사용하고 계시고 extensions/ postgis 모듈을 컴파일하고 설치하였다면 공간 데이터베이스를 생성하실 수 있습니다.

createdb [yourdatabase]

핵심 postgis extension 은 PostGIS 도형. 지리, 래스터, spatial_ref_sys 및 모든 기능들과 주석을 간단한 명령어로 설치합니다.

```
CREATE EXTENSION postgis;
```

psql 에서의 명령입니다.

psql -d [yourdatabase] -c "CREATE EXTENSION postgis;"

토폴로지는 별도 extension 로써 패키지화 되어 있고, 아래의 명령어로 설치 가능합니다

psql -d [yourdatabase] -c "CREATE EXTENSION postgis_topology;"

만약 새로운 DB 에 이전 버전의 백업을 복원하고 싶다면

psql -d [yourdatabase] -f legacy.sql

복구와 청소(cleanup)를 마친 뒤 앞으로 사용하지 않을 기능들을 제거하기 위해서는 `uninstall_legacy.sql` 을 실행하십시오.

2.7. Tiger Geocoder 의 설치, 업그레이드 및 데이터 불러오기

Tiger geocoder(Tiger 는 미국 통계청의 통계지리데이터, geocorder 는 주소-좌표 변환 기능)는 지역용이기 때문에 핵심 PostGIS 스트립트들과 함께 설치/업그레이드 되지 않습니다. extras 폴더에 있는 것들은 모두 일반적인 PostGIS 설치/업그레이드 기본 설정으로 설치되지 않습니다. Tiger geocoder 와 같은 추가 기능들은 PostGIS 패키지로 배포되지 않지만 postgis-2.0.5SVN.tar.gz 파일로 언제든지 이용 가능합니다. `extras/tiger_geocoder/tiger_2010/README` 에서 개요를 제공합니다.

Tar 를 인스톨하지 않으셨고 윈도우를 사용하고 계시다면 PostGIS Tarball 의 압축을 풀기 위해 http://www.7-zip.org/를 이용하십시오.

2.7.1. PostGIS 데이터베이스에서 Tiger Geocoder 활성화 방법

우선 앞의 설명에 따라 PostGIS 를 설치하십시오.

extras 폴더가 없다면 http://postgis.net/stuff/postgis-2.0.5SVN.tar.gz 에서 다운로드 받으십시오.

tar xvfz postgis-2.0.5SVN.tar.gz

cd postgis-2.0.5SVN/extras/tiger_geocoder/tiger_2010

실행 가능한 서버 등의 경로로 `tiger_loader.sql` 을 편집하십시오.

만약 Tiger geocoder 를 처음으로 설치하는 경우, 윈도우는 create_geocode.bat 스크립트를, Linux/Unix/ Max OSX 는 create_geocode.sh 을 PostgreSQL 설정에 맞게 편집하십시오. 이 파일을 편집하지 않을 경우에는 공통적으로 포함된 요소의 경로 정보 정도만 포함됩니다. Loader_Generate_Script 명령을 실행한 후 생성 스크립트를 편집할 수 있습니다.

tiger 스키마가 데이터베이스 안에 있는 지와 검색 범위(search_path)에 해당되는 지를 확인하십시오. 안되어 있다면 아래 명령을 함께 추가하십시오.

```
ALTER DATABASE geocoder SET search_path=public, tiger;
```

표준화 주소 기능은 까다로운 주소를 제외하고는 동작합니다. 아래와 비슷하게 나오는지 테스트 해보십시오.

```
SELECT pprint_addy(normalize_address('202 East Fremont Street, Las Vegas,
Nevada 89101')) As pretty_address;
pretty_address
---------------------------------------
202 E Fremont St, Las Vegas, NV 89101
```

2.7.2. Tiger Geocoder 업그레이드

2.0 버전과 함께 Tiger Geocoder 가 이미 설치되어 있다면, 중간 버전의 tar ball 을 이용해서도 언제든지 기능들을 업그레이드 시킬 수 있습니다.

만약 extras 폴더가 없다면, http://postgis.net/stuff/postgis-2.0.5SVN.tar.gz 에서 다운로드 받으십시오.

tar xvfz postgis-2.0.5SVN.tar.gz

cd postgis-2.0.5SVN/extras/tiger_geocoder/tiger_2010

윈도우는 upgrade_geocoder.bat 파일, Linux/Unix/Mac OSX 이라면 upgrade_geocoder.sh 를 찾으십시오. postgis 데이터베이스에 대한 자격 부여와 상응하는 스크립트를 실행시키기 위해 찾은 파일을 편집하십시오.

2.7.3. Tiger Data 불러오기

Tiger 데이터를 로딩하기 위한 보다 자세한 설명은 extras/tiger_geocoder/tiger_2010/README 에서 이용 가능합니다. 여기서는 일반적인 과정만 안내해 드립니다.

인구조사 웹사이트에서 필요한 주의 데이터를 다운로드 받습니다. 압축을 풀면 주 단위로 별개의 세트로 이뤄져 있습니다. 각 주 테이블은 tiger 스키마 정의에 따라 접속 및 쿼리 할 수 있고, Drop_State_Tables_Generate_Script 를 다시 불러오거나 삭제하는 등의 작업을 실시할 수 있습니다.

데이터를 로딩하기 위해서는 다음과 같은 도구(tool)들이 필요합니다:

- 인구조사 웹사이트에서 받은 압축 파일을 풀기 위한 툴
  Unix 계열 시스템에서는 대부분 unzip 으로 해제할 수 있습니다. 윈도우의 경우, 무료 압축/압축해제 툴인 7-zip 이 http://www.7-zip.org/에서 다운로드 가능합니다.
- PostGIS 를 기본 설치할 때의, `shp2pgsql` 명령행
- 웹 갈무리 툴인 `wget` 은 대부분의 Unix/Linux 시스템에 설치되어 있습니다.
  윈도우를 사용 중이면 http://gnuwin32.sourceforge.net/packages/wget.htm 에서 다운로드 가능합니다.

데이터를 로딩하기 위해서는 Loader_Generate_Script 를 참조하십시오. 주 단위로 로딩할 수 있고, 모든 주를 한번에 로딩할 필요가 없습니다.

원하는 데이터가 로딩이 된 후 Install_Missing_Indexes 에서 설명된 대로 다음을 실행하십시오.

```
SELECT install_missing_indexes();
```

설치가 안된 인덱스를 보여줍니다. 실행이 잘되는지 확인하려면 Geocode 를 이용하여 해당 주의 주소를 변환시켜 보십시오.

2.8. 템플릿을 이용하여 공간 데이터베이스 생성하기

PostGIS 배포 패키지(특히 PostGIS1.1.5 이상 버전의 Win32 인스톨러)는 template_postgis 라 불리는 템플릿 데이터베이스를 포함하고 있습니다. PostgreSQL 에 `template_postgis` 데이터베이스가 존재하고 있다면 사용자 또는 응용프로그램에서 공간 데이터베이스를 간단한 명령어를 통해 생성 가능합니다. 두 경우 모두 새로운 데이터베이스들을 생성할 수 있는 권한을 사용자가 가지고 있어야 한다는 점을 주의하십시오.

셸(shell)에서:

```
# createdb -T template_postgis my_spatial_db
```

SQL 에서:

```
postgres=# CREATE DATABASE my_spatial_db TEMPLATE=template_postgis
```

2.9. 업그레이딩

공간 데이터베이스 업그레이딩은 대체의 또는 새로운 PostGIS 객체 정의(object definitions)를 요구하기 때문에 까다로울 수 있습니다.

안타깝게도 실제 데이터베이스 내에서 모든 정의들이 쉽게 대체될 수 있지 않습니다. 그러므로 때론 dump/reload 방식이 최고의 선택일 수 있습니다.

PostGIS 는 사소하거나 버그 수정 버전을 위한 SOFT UPGRADE 와 주요 버전을 위한 HARD UPGRADE 를 제공합니다.

PostGIS 업그레이드에 앞서 데이터를 미리 백업해두시는 것은 언제나 중요합니다. pg_dump 를 할 때 –Fc flag 를 이용하면 HARD UPGRADE 시 dump 를 복원할 수 있습니다.

2.9.1. 소프트 업그레이드

extension 을 포함하여 데이터베이스를 설치하실 경우, extension 모델 또한 업그레이드 하여야 합니다. 이전 SQL 스크립트 방식으로 설치한 경우, 마찬가지로 SQL 스크립트 방식으로 업그레이드해야 합니다.

2.9.1.1. extension 이 없는 Pre 9.1 이상 버전의 소프트 업그레이드

이 단원은 extension 없이 PostGIS 를 설치한 경우만 해당됩니다. 만약 extension 가 있으신 상태에서 이 단원에서 설명하는대로 설치를 하시면 아래와 같은 메시지를 얻게 되실 것입니다:

```
can't drop ... because postgis extension depends on it
```

컴파일 하신 뒤 반드시 몇 몇의 `postgis_upgrade*.sql` 파일들을 찾아 이 중 PostGIS 버전에 맞는 것을 설치하십시오. 예를 들어 PostGIS1.3~1.5 의 경우 `postgis_upgrade_13_to_15.sql` 를 사용하셔야 합니다. PostGIS 버전 1 에서 2 로 업그레이드 하는 경우는 HARD UPGRADE 를 이용하십시오.

```
psql -f postgis_upgrade_20_minor.sql -d your_spatial_database
```

`rtpostgis_upgrade*.sql` 과 `topology_upgrade*.sql` 을 통해 똑같은 절차가 래스터 그리고 토폴로지 extension 에도 적용됩니다.

```
psql -f rtpostgis_upgrade_20_minor.sql -d your_spatial_database
```

```
psql -f topology_upgrade_20_minor.sql -d your_spatial_database
```

만약 버전 업그레이드를 위한 구체적인 `postgis_upgrade*.sql` 을 찾을 수 없다면 이는 해당 버전을 너무 일찍 사용하고 계신다는 것을 의미하며 HARD UPGRADE 가 필요합니다.

The PostGIS_Full_Version 을 이용하면 이러한 종류의 업그레이드가 필요할 경우 “procs need upgrade” 메시지를 통해 알려드릴 것입니다.

2.9.1.2. 소프트 업그레이드 9.1+ extension 을 사용하는 경우

extension 과 함께 PostGIS 를 설치한 경우 extension 또한 업그레이드 하셔야 합니다. Extension 에 대한 사소한 업그레이드는 어렵지 않습니다.

```
ALTER EXTENSION postgis UPDATE TO "2.0.5SVN";
ALTER EXTENSION postgis_topology UPDATE TO "2.0.5SVN";
```

만약 다음과 같은 에러 메시지가 보인다면:

```
No migration path defined for ... to 2.0.5SVN
```

먼저 데이터베이스를 백업하고, 2.6 단락에서 설명한대로 새로운 데이터베이스에 복원합니다. postgis extension 이 이미 설치되었다는 메시지를 보실 수도 있지만 무시하셔도 무방합니다.

특정 버전 없이 PostGIS 를 설치한 경우, 백업을 복구하기 전에 postgis extension 재설치를 무시하고 넘어갈 수 있습니다. 왜냐하면 백업 하면서 CREATE EXTENSION postgis 을 생성하고, 복구시에 최신버전을 찾아내기 때문입니다.

2.9.2. 하드 업그레이드

HARD UPGRADE 는 full dump/reload 를 의미합니다. HARD UPGRADE 는 PostGIS 객체의 내부 스토리지가 변화하지만 SOFT UPGRADE 는 그렇지 않습니다. Release Notes 부록(appendix)에서 덤프/리로드(하드 업그레이드)가 필요한 버전에 대해 알려줍니다.

덤프/리로드 프로세스는 PostGIS(구 버전들 포함)의 덤프를 건너뛸 수 있도록 하는 postgis_restore.pl script 의 지원을 받습니다. 이것은 심볼 등 중복 파일로 인한 문제나 더 이상 사용되지 않는 객체들을 불러오지 않고 설치된 PostGIS 의 데이터베이스에 스키마(schemas)와 데이터가 복구할 수 있게 합니다.

윈도우 사용자들을 위한 보충설명들은 Windows Hard upgrade 에서 이용 가능합니다.

절차는 다음과 같습니다:

1. binary blobs 이 있는 경우 (-b) 및 진행 상세정보를 원하면 (-v) 옵션을 포함하여 업그레이드를 원하는 데이터베이스(olddb 라 부름)의 "custom-format" 덤프를 생성하십시오. 사용자는 데이터베이스의 소유자이면 되며, postgres 수퍼 계정일 필요는 없습니다.

   ```
   pg_dump -h localhost -p 5432 -U postgres -Fc -b -v -f
     "/somepath/olddb.backup" olddb
   ```

2. newdb 라 부를 새로운 데이터베이스를 만드십시오. 2.5 와 2.6 단락을 참조하십시오. 덤프에서 발견되는 spatial_ref_sys 목록은 복구되지만 중복으로 겹쳐쓰지는 않을 것입니다. 만약 어떤

이유로 겹쳐쓰기를 원하신다면 새 DB 를 생성할 때 spatial_ref_sys.sql 을 로드 하지 마십시오. 가지고 계신 데이터베이스가 아주 구식이거나 가지고 계신 뷰나 기능들의 기능들이 더 이상 사용되지 않는다면 legacy.sql 로딩을 필요로 할 수도 있습니다. 정말로 필요할 경우에만 이를 실행하십시오. 가능하다면 덤핑(dumping)을 하기 전에 뷰 및 기능들을 업그레이드 할 수 있는지 먼저 확인하십시오. `uninstall_legacy.sql` 을 통해 더 이상 사용되지 않고 앞으로는 사라지게 될 기능들을 제거할 수 있습니다.

3. postgis_restore.pl 을 사용하여 새로운 `newdb` 데이터베이스에 백업을 복구시키십시오. 예상치 못한 에러들이 발생하면 psql 에 의해 표시될 것입니다. 그 결과 로그를 보관하십시오.

```
perl utils/postgis_restore.pl "/somepath/olddb.backup" | psql -h
   localhost -p 5432 -U postgres newdb 2> errors.txt
```

다음과 같은 경우에 에러들이 발생할 수 있습니다:

1. 일부 뷰 또는 기능에서 더 이상 사용되지 않고 사라지게 될 PosgtGIS 객체들을 사용하는 경우입니다. 이러한 문제를 해결하기 위해서 복구에 앞서 `legacy.sql` 를 로딩하거나 해당 객체들을 PostGIS 에 복구한 다음 마이그레이션을 다시 시도해 보십시오. legacy.sql 방법이 효과적이라면, 해당 객체를 기억하고 `uninstall_legacy.sql` 을 로딩하는 것을 중단하십시오.

2. 덤프 파일 중 spatial_ref_sys 의 사용자 레코드에 잘못된 SRID 값이 있는 경우입니다. 올바른 SRID 값은 0 보다 크며 999000 보다 작습니다. 999000.999999 범위 내 값들은 내부 사용을 위한 것이고, 999999 보다 큰 값들은 사용될 수 없습니다. 인식 불가능한 SIRD 의 모든 커스텀 기록들은 유지됩니다. 999999 보다 큰 모든 기록들은 보관되는 범위로 이동되지만 spatial_ref_sys 테이블을 변함없이 유지하지 위한 체크 제약 가이드와 기본 키(primary key)는 유실될 수 있습니다. (다수의 인식 불가능한 SRIDS 가 똑같은 예약 SRID 값으로 전환되는 경우). 이 문제를 해결하기 위해선 사용자 SRS 를 유효한 값의 SRID 로 복사하고(아마도 910000..910999 범위), 모든 테이블들을 새로운 SRID 로 전환하셔야 합니다. (UpdateGeometrySRID 참조), 또한 spatial_ref_sys 의 인식 불가능한 목록을 지우고 아래와 같이 변경하십시오:

```
ALTER TABLE spatial_ref_sys ADD CONSTRAINT
spatial_ref_sys_srid_check check (srid > 0 AND srid < 999000 );
```

```
ALTER TABLE spatial_ref_sys ADD PRIMARY KEY(srid));
```

2.10. 일반적인 문제들

인스톨이나 업그레이드 시 잘 되지 않을 경우 확인해야 할 몇 가지 사항은 아래와 같습니다.

1. PostgreSQL 8,4 또는 그 보다 최신 버전을 설치하셨는지 체크하십시오. 현재 작동하고 계시는 PostgreSQL 의 버전과 동일한 PostgreSQL 소스 버전을 바탕으로 컴파일하였는지 확인해주십시오. Linux 에서 이미 PostgreSQL 이 설치된 경우 중복 설치가 될 수 있고, 또는 이전에 설치한 사실을 잊어버렸을 수도 있습니다. PostGIS 는 PostgreSQL 8.4 또는 그 이상

버전에서 작동되므로 구 버전을 이용 시 예상하지 못한 에러들이 발생할 수 있습니다. 어떤 PostgreSQL 버전이 작동하고 있는지 확인하기 위해서는 psql 을 통하여 데이터베이스에 접속해 다음의 쿼리를 실행하십시오:

```
SELECT version();
```

RPM 방식이라면 **rpm -qa | grep postgresql** 로 이전 설치 패키지의 존재를 확인할 수 있습니다.

2. 만약 업그레이드가 실패하면 기존 PostGIS 설치 버전으로 복원하셔야 합니다.

```
SELECT postgis_full_version();
```

또한 PostreSQL, Proj4 라이브러리 및 GEOS 라이브러리에 대한 버전과 경로에 대한 설정이 올바른지 확인하십시오.

1. 설정은 `postgis_config.h` 파일을 생성하기 위해 사용됩니다. `POSTGIS_PGSQL_VERSION`, `POSTGIS_PROJ_VERSION` 및 `POSTGIS_GEOS_VERSION` 변수가 정확한지 확인하십시오.

2.11. JDBC

JDBC extension 은 PostGIS 타입들에 상응하는 자바 객체를 제공합니다. 이 객체들은 PostGIS 데이터베이스 쿼리, 그리기, 계산 등을 하는 자바 클라이언트에 쓰입니다.

1. `java/jdbc` 경로로 이동하십시오.
2. `ant` 명령어를 실행하고 `postgis.jar` 파일을 java 라이브러리들이 있는 곳으로 복사하십시오.

JDBC extension 은 빌드 프로세스 중 CLASSPATH 에 PostgreSQL JDBC 를 필요로 합니다. 만약 PostgreSQL JDBC 드라이버가 다른 경로에 있다면 아래와 같은 -D 파라미터(-D parameter)를 이용하여 JDBC driver JAR 의 경로를 연결시켜 줍니다:

```
# ant -Dclasspath=/path/to/postgresql-jdbc.jar
```

PostgreSQL JDBC 드라이버들은 http://jdbc.postgresql.org 에서 다운로드 받으실 수 있습니다.

2.12. 로더/덤퍼(Loader/Dumper)

데이터 로더와 덤퍼는 PostGIS 의 한 부분으로서 자동적으로 설치 및 생성됩니다. 로더와 덤퍼를 수동으로 설치하기 위해서는:

```
# cd postgis-2.0.5SVN/loader
# make
# make install
```

로더는 shp2pgsql 이라고 불리며 ESRI Shape 파일을 PostGIS/PostgreSQL 다루기 알맞도록 전환합니다. 덤퍼는 pgsql2shp 이라 불리며 PostGIS 테이블(또는 쿼리들)을 ESRI Shape 파일로 전환합니다. 보다 더 상세한 설명을 원하신다면 온라인 도움말(help)과 매뉴얼을 참조하십시오.

Chapter 3.PostGIS 자주 묻는 질문들

1. **애플리케이션들과 데스크탑 툴들은 PostGIS1.5 와 연동이 잘되는 반면 PostGIS2.0 과는 연동이 잘 되지 않습니다. 이 문제를 어떻게 해결 할 수 있나요?**

중요도가 떨어져 더 이상 사용되지 않고 앞으로 사라지게 될 많은 기능들이 PostGIS 2.0 기반의 코드에서 제거되었습니다. 이것은 애플리케이션뿐만아니라 Geoserver, Mapserver, QuantumGIS, OpenJump 와 같은 제 3 의 툴들에도 영향을 끼쳤으나 문제 해결방법에는 몇가지가 있습니다. 문제 해결을 위해 이들의 버전을 최신 버전으로 업그레이드하시면 많은 이슈들이 해결됩니다. 코드의 경우 제거된 기능들을 사용하지 않도록 코드를 변화시킬 수 있습니다. 이러한 기능의 대부분은 non ST_ aliases of ST_Union, ST_Length 등입니다. 더불어 최후의 수단으로서 `legacy.sql` 전체를 설치하시거나 필요로하는 `legacy.sql` 의 일부를 설치하십시오.

The legacy.sql f 파일은 postgis.sql. 과 같은 폴더에 위치하고 있습니다. 제거되었던 200 개의 기함수를 얻기 위해 postgis.sql 그리고 spatial_ref_sys.sq 설치하신 뒤 해당 파일을 설치하시면 됩니다.

2. **osm2pgsql 와 함께 OpenStreetMap 데이터를 로딩할 때 다음과 같은 실패 에러 메세지가 나타납니다: ERROR: operator class "gist_geometry_ops" does not exist for access method "gist" Error occurred. PostGIS 1.5 에서는 잘 운용됩니다.**

PostGIS2 에서 디폴트 geometry operator class gist_geometry_ops 는 gist_geometry_ops_2d 로 변경되었으며 the gist_geometry_ops 은 완전히 제거되었습니다. 이는 PostGIS 2.0 또한 3D 를 지원하기 위해 Nd 공간 인덱스를 도입하였는데 옛 이름이 혼동을 주어 적절하지 못하다고 판단되었기 때문입니다.

프로세스의 한 일부로써 일부 이전 애플리케이션들 테이블들과 인덱스를 생성하며 오퍼레이터 클래스 명을 명쾌하게 참조 표시합니다. 2D 인덱스를 디폴트 하고 싶으실 경우에는 위와 같은 작업이 필요 없습니다. 그러므로 만약 Good 이라고 나타난다면 인덱스 크리에이션을 BAD 에서 Good 으로 바꾸십시오: BAD:

```
CREATE INDEX idx_my_table_geom ON my_table USING gist(geom gist_geometry_ops);
```

To GOOD:

```
CREATE INDEX idx_my_table_geom ON my_table USING gist(geom);
```

다음과 같은 3D 공간 인덱스가 필요한 경우에만 오퍼레이터 클래스를 명시화 하십시오:

```
CREATE INDEX idx_my_super3d_geom ON my_super3d USING gist(geom
gist_geometry_ops_nd);
```

만약 안타깝게도 이전 gist_geometry_ops hard-coded 을 가지고 있는 변화 시킬 수 없는 컴파일 된 코드에 갇혀있다면 PostGIS 2.02+ 안 패키지된 `legacy_gist.sql` 을 이용하여 구 클래스를 생성시킬 수 있습니다. 그러나 만약 이 픽스를 사용하신다면 나중에 인덱스를 중단하고 오퍼레이터 클래스 없이 다시 재생성 하라고 권고 받을 것입니다. 이로 인해 훗날 다시 업그레이드를 하실 때 불편함을 덜 수 있을 것입니다.

3. PostgreSQL 9.0 을 운용 중이며 OpenJump, SafeFME 와 그리고 몇 몇 다른 툴들에서 지오메트리들을 더는 읽거나 볼 수 없습니다.

PostgreSQL 9.0+에서, bytea 데이터를 위한 디폴트 인코딩은 hex 로 바뀌었고 예전 JDBC 드라이브는 여전히 escape format 을 취합니다. 이것은 예전 JDBC 드라이버를 사용한 Java 어플리케이션이나 오래된 ST_AsBinary 의 작동을 요하는 예전 npgsql 드라이버를 사용하는 .NET 어플리케이션 같은 몇몇 어플리케이션에 영향을 끼칩니다. 이것을 다시 작동시키기 위한 두 가지 접근 방법이 있습니다.

JDBC driver 를 최신 PostgreSQL 9.0 버전으로 업그레이드 시킬 수 있습니다. 최신 PosgtreSQL 버전은 http://jdbc.postgresql.org/download.html 에서 다운받으실 수 있습니다.

만약 .NET app 을 실행중이라면 Npgsql 2.0.11 또는 그 이상의 버전을 사용할 수 있습니다. 이는 Francisco Figueiredo's NpgSQL 2.0.11 released blog entry 에 설명되어 있는 것과 같이 http://pgfoundry.org/frs/?group_id=1000140 에서 다운받으실 수 있습니다.

만약 PostgreSQL driver 를 업그레이드 하는 것이 옵션사항이 아니라면 아래를 입력함으로써 이전방식으로 디폴트 설정을 할 수 있습니다:

```
ALTER DATABASE mypostgisdb SET bytea_output='escape';
```

4. 기하학 행(geometry column)을 보기 위해 PgAdmin 을 사용하려고 했으나 비어있습니다. 왜 그런지요?

```
-- this should return no records if all your geom fields are filled in
SELECT somefield FROM mytable WHERE geom IS NULL;
```

```
-- To tell just how large your geometry is do a query of the form
--which will tell you the most number of points you have in any of your
geometry columns
SELECT MAX(ST_NPoints(geom)) FROM sometable;
```

5. 어떠한 종류의 지오메트리 오브젝트들을 저장할 수 있습니까?

점, 선, 다각형, 기하, 다중점, 다중선, 다중다각형 그리고 지오메트리 컬렉션들을 저장할 수 있습니다. PostGIS 2.0 과 그 이상 버전에서는 기본 지오메트리 타입의 TINS 그리고 다면체 표면 또한 저장할 수 있습니다. 이러한 것들은 잘 유명한 Open GIS 텍스터 포맷에 명시되어 있습니다(XYZ, XYM, XYZM 확장자들과 함께). 현재 세가지 데이터타입들이 지원 받고 있습니다. 측량을 위한 평면 좌표계를 사용하는 표준 OGC 지오메트리 데이터 타입, 측지학 좌표계(OGC 가 아닌 Microsoft SQL Server 2008+에서 비슷한 타입을 발견하실 수 있습니다.)를 사용하는 측지학 데이터 타입. 오직 WGS 84 long lat (SRID:4326)만이 지형 데이터 타입에 의해 지원됩니다. PostGIS 공간타입에서 최신의 형태는 래스터 데이터를 저장하기 위한 래스터 입니다. 래스터는 매우독특한 FAQ 를 가지고 있습니다. 보다 자세한 사항을 위해 Chapter 10 자주묻는 질문들 그리고 Chapter 9 래스터 그리고 래스터 레퍼런스를 참조하십시오.

6. 혼동되네요. 제가 지오메트리(geometry) 또는 지형(geography) 중 어떤 데이터를 사용하여 저장해야 합니까?

짧은 답변: 지형(geography)는 원거리 거리 측량을 지원하는 새로운 데이터 타입입니다. 그러나 기하에 대한 대부분의 계산들은 geometry 에서 하는 것보다 현재 느립니다. 만약 지형을 사용하실 경우 평면 좌표계에 대해서 배우실 필요가 거의 없습니다. 만약 거리나 길이를 측량하는 것을 중요시 생각하시거나 전세계로부터의 데이터를 보유하고 계실 경우 지형(geography)이 기하보다 빠릅니다. Geometry 형식은 이를 지원하는 많은 기능들을 가지고 있는 오래된 데이터 형태입니다. 지오메트리 데이터 타입은 타상의 툴들로부터 많은 지원을 받으며 일반적으로 이것을 처리하는데 있어서 더 빠릅니다. 더 큰 기하적 구조들의 경우 10 배까지 더 빠른 경우도 종종 있습니다. 공간 레퍼런스 시스템에 더 수월함을 느끼거나 단일 공간 spatial reference system(SRID)에 맞는 국지적 데이터를 다루고 있거나 또는 많은 양의 공간적 프로세싱을 수행해야 하는 상황이라면 Geometry 가 최고의 선택일 것입니다. 참고: 두 타입 모두의 혜택들을 얻기 위해 두 타입 사이의 일회성 전환은 어렵지 않습니다. 현재 무엇이 지원되고 지원되지 않는 지를 확인하기 위해서 13.10 단락, "PostGIS Function Support Matrix" 을 참조하십시오.

긴 답변: 보다 더 긴 답변을 원하신다면 4.2.2 단락, "When to use Geography Data type 그리고 function type matrix.를 참조하십시오.

7. 지리적 지역이 얼마나 큰지와 같이 더 복잡하고 심오한 질문이 있습니다. geography 행에 삽입하여 타당한 답들을 얻을 수 있나요? 예를 들어 남북극과 같은 제한사항이 있나요? 모든 필드는 반구(SQL Server 2008 가 가지고 있는 것처럼), 스피드 등에 맞아 떨어져야 하나요?

이 섹션에서 답변하기에는 질문이 너무 깊고 복잡합니다. 4.2.3 단락을 참조하십시오.

8. 어떻게 GIS 오브젝트를 데이터베이스에 삽입할 수 있나요?

First, 첫째로 GIS 데이터를 보유하기 위해 "geometry" 또는 "geography"의 행을 가진 테이블을 생성하셔야 합니다. geography 타입 데이터를 저장하는 것은 geometry 를 저장하는 것과는 조금

다릅니다. Geography 를 저장하는 것에 관한 보다 자세한 설명은 4.2.1 단락을 참조하십시오. Geometry 사용을 위해: psql 과 함께 데이터베이스에 접속하시고 다음의 SQL 을 시도하십시오.

```
CREATE TABLE gtest ( gid serial primary key, name varchar(20)
       , geom geometry(LINESTRING) );
```

만약 geometry 행 정의가 실패한다면 이는 아마도 PostGIS 기능들과 오브젝트들을 데이터베이스에서 로딩하지 않았거나 PostGIS 의 pre-2.0 버전을 사용하고 있기 때문일 것입니다. 2.4 단락을 참조하십시오. 그러고 난 뒤, SQL insert statement 를 사용함으로서 geometry 를 테이블에 삽입할 수 있습니다. OpenGIS Consortium "well-knows text" 포맷을 사용함으로서 GIS 오브젝트를 포맷할 수 있습니다:

```
INSERT INTO gtest (ID, NAME, GEOM)
VALUES (
  1,
  'First Geometry',
  ST_GeomFromText('LINESTRING(2 3,4 5,6 5,7 8)')
);
```

GIS 오브젝트들에 관한 보다 많은 정보들을 원한다면 object reference 를 참조하십시오. 테이블 속 GIS 데이터를 보고자 할 때:

```
SELECT id, name, ST_AsText(geom) AS geom FROM gtest;
```

반환값(return value)는 아래와 같이 나타나야 합니다:

```
id | name            | geom
---+-----------------+----------------------------
 1 | First Geometry | LINESTRING(2 3,4 5,6 5,7 8)
(1 row)
```

9. 어떻게 공간 쿼리(spatial query)를 구축할 수 있나요?

반환값, 기능들 그리고 불 방식(boolean) 테스트들의 SQL 컴피네이션과 같은 다른 모든 데이터베이스 쿼리를 구축하는 것과 똑같은 방식으로 구축할 수 있습니다.

공간 쿼리들의 경우, 쿼리를 구축할 시 염두해두어야 할 두가지 중요한 사항들이 있습니다. 활용할 수 있는 공간인덱스가 있는가? 그리고 geometries 의 많은 수의 고급계산들을 하고 있는가? 가 바로 그 중요한 두가지 사항들 입니다.

일반적으로 피쳐의 바운딩박스(features of the bounding boxes)가 교차하는지 하지 않는지에 대해 테스트 하는 "교차 오퍼레이터(intersects operator)"(&&)"를 사용하길 원할 것입니다. && 오퍼레이터가 유용한 이유는 공간 인덱스가 테스트 속드를 높일 수 있는 상황일 때 이를 활용하기 때문입니다. 이는 쿼리는 매우 모두 빠르게 할 것입니다.

그 중에서도 Distance(), ST_Intersects(), ST_Contains() and ST_Within()와 같은 공간 기능들을 검색 결과를 좁히기 위해 활용하게 될 것입니다. 대부분의 공간 쿼리들은 색인 붙은 테스트 그리고 공간 기능 테스트를 포함합니다. 인덱스 테스트는 그상태를 충족할 수도 있는 리턴 투플들(returen tuples)의

숫자를 제한하는 역할을 합니다. 그러고는 공간 기능들은 컨디션을 정확히 테스트 하기 위해 사용됩니다.

```
SELECT id, the_geom
FROM thetable
WHERE
  ST_Contains(the_geom,'POLYGON((0 0, 0 10, 10 10, 10 0, 0 0))');
```

10. 어떻게 큰 테이블 위에서 공간 쿼리 속도를 높일 수 있나요?

큰 테이블의 빠른 쿼리는 공간 데이터베이스의 존재이유입니다 (트랜잭션 지원와 함께) 그러므로 좋은 인데스(good index)를 가지는 것은 중요합니다.

Geometry 행과 함께 테이블 위 공간 인덱스를 구축하기 위해 다음과 같은 "CREATE INDEX" 기능을 사용하십시오

```
CREATE INDEX [indexname] ON [tablename] USING GIST ( [geometrycolumn] );
```

"USING GIST"옵션은 GiST(Generalized Search Tree) 인덱스를 사용하도록 서버를 알려줍니다.

GiST 인덱스들은 손실된다고 추정됩니다. 손실 인덱스들은 인덱스 구축을 위해 proxy 오브젝트을 사용합니다.

인덱스에 관한 충분한 정보를 가진 PosgreSQL 쿼리 플래너가 언제 이것을 사용할지에 합당한 결정을 할 수 있도록 보장해야 합니다. 이것을 하기 위해서는 geometry 테이블에서 “gather statistics”을 반드시 해야 합니다.

PostgreSQL 8.0.x 그리고 그 이상 버전의 경우, **VACUUM ANALYZE** 명령어을 실행하십시오.

PostgreSQL 7.4.x 그 이하의 버전의 경우, **SELECT UPDATE_GEOMETRY_STATS()**명령어를 실행하십시오.

11. 왜 PostgreSQL R-Tree indexes 는 지원되지 않나요?

PostGIS 의 초기 버전들은 PostgreSQL R-Tree 인덱스들을 사용했습니다. 그러나 PostgreSQL R-Trees 버전 0.6 이후 완전히 폐기되었고 공간인덱싱은 R-Tree-over-GiST scheme 와 함께 제공됩니다.

저희 테스트들은 native R-Tree and GiST 를 위한 검색 속도가 비슷하다는 것을 보여주었습니다. Native PostgreSQL R-Trees 는 GIS 피쳐들과 함께 사용하는 것이 바람직하지 않게 만드는 두개의 한계를 가지고 있습니다.(이러한 한계들은 일반적으로 R-Tree concept 때문이 아닌 현재의 PostgreSQL native R-Tree PostgreSQL native R-Tree implementation 때문이라는 점을 주의하십시오:

- R-Tree indexes in PostgreSQL 의 R-Tree 인덱스들은 8K 보다 큰 사이즈 피쳐들을 처리할 수 없습니다. GiST indexes 피쳐를 스스로를 위한 바운딩 박스를 대체하는 "손실(lossy)" 트릭을 사용할 수 있습니다.

- R-Tree indexes in PostgreSQL 의 R-Tree 인덱스들은 "null safe"가 아닙니다. 그러므로 null geomtmeries 를 함유하는 geometry 행 위 인덱스를 구축하지 못합니다.

12. 왜 `AddGeometryColumn()` 기능을 사용해야 하나요? 그리고 다른 모든 OpenGIS 들도 사용해야만 하나요?

만약 OpenGIS 지원 기능들을 사용하길 원하지 않으신다면 사용할 필요는 없습니다. 단순히 구 버전의 테이블들을 생성하고 CREATE 문의 geometry 행들을 정의하십시오. 모든 gemmetiries 는 -1 의 SRID 를 가질 것이며, OpenGIS meta-data 테이블은 적절하게 채워지지 않을 것입니다. 그러나, 이것은 PostGIS 기반의 대부분의 애플리케이션들이 실패하도록 조장할 것 입니다. 그리고 geometry 테이블들을 생성하기 위해서 일반적으로 `AddGeometryColumn()`을 사용토록 권장될 것입니다.

MapServer 는 `geometry_columns` meta-data 를 활용하는 하나의 애플리케이션 입니다. 구체적으로 말하면 MapServer 는 피쳐의 on-the-fly reprojection 을 correct map projection 에 Geometry 행의 SRID 를 사용할 수 있도록 합니다.

13. 다른 오브젝트의 반경 이내 모든 오브젝트들을 찾을 수 있는 가장 좋은 방법은 무엇인가요?

데이터베이스를 가장 효율적으로 사용하기 위해서는 바운딩박스 테스트와 반경 테스트를 결합하는 반경 쿼리를 하는 것이 가장 좋습니다: 바운딩박스 테스트(bounding box test)는 공간 인덱스를 사용하며 반경 테스트가 뒤에 적용되는 데이터의 부분집합에 대한 빠른 엑세스를 제공합니다. `ST_DWithin(geometry, geometry, distance)` 기능은 인덱스된 거리 검색을 수행하는데 유용한 방법입니다. 반경을 둘러싸기에 충분한 크기의 검색 직사각형을 생성하며 작동합니다. 그런 뒤 검색에 인덱스된 부분집합의 정확한 거리를 수행합니다.

예를 들어 POINT(1000 1000)의 100 미터의 모든 오브젝트들을 찾기 위해서는 다음의 쿼리를 통해 원활히 수행할 수 있습니다.

```
SELECT * FROM geotable
WHERE ST_DWithin(geocolumn, 'POINT(1000 1000)', 100.0);
```

14. 어떻게 쿼리의 일부로 좌표 reprojection 을 수행합니까?

reprojection 을 수행하기 위해서는 소스 그리고 목적지 좌표계 둘 모두 SPATIAL_REF_SYS 테이블에 정의되어야 합니다. 그리고 reprojection 되는 geometries 는 반드시 SRID 를 먼저 가지고 있어야 합니다. 이것이 수행된 후 reprojection 은 희망 목적지 SRID(desired destination SRID)관련 만큼이나 간단합니다. the_geom 이 -1 이 아닌 경우에만 아래와 같이 작동됩니다

```
SELECT ST_Transform(the_geom,4269) FROM geotable;
```

15. 제법 큰 지오메트리 위에서 ST_AsEWKT 와 ST_AsText 을 하였습니다. 그랬더니 이것이 빈 필드로 변해버렸습니다. 왜 이런 것인가요?

아마 큰 텍스트를 출력하지 않는 PgAdmin 혹은 어떤 툴들을 사용하고 있기 때문일 것입니다. 만약 geometry 가 충분히 크다면 이러한 툴들에서 빈 필드가 나타날 것입니다. 정말 해당사항을 보기 원하거나 WKT 에서 이것의 출력을 원하신다면 PSQL 을 사용하십시오

```
- - To check number of geometries are really blank
SELECT count(gid) FROM geotable WHERE the_geom IS NULL;
```

16. ST_Intersects 할 때 두 개의 기하들이 교차하지 않는다는 메세지를 받습니다. 교차가 가능하다고 알고 있는데 왜 이런 것인가요?

이것은 일반적으로 두가지 경우에서 발생합니다. Geometry 가 잘못된 경우 -- ST_IsValid 를 확인하거나, ST_AsText 숫자들의 길이를 줄이고 또 보이지 않은 뒤 소수를 가지고 있기 때문에 이것들이 교차한다고 추정하고 있습니다.

17. 저는 PostIGIS 를 사용하여 소프트웨어를 개발 출시하였습니다. PostGIS 와 같이 GPL 을 사용하는 제 소프트웨어는 라이선스를 획득해야 하나요? 만약 PostGIS 를 사용할 경우 제 코드를 공개해야만 하나요?

그렇지 않습니다. 예를 들어, 리눅스에 오라클 데이터베이스를 생각해보십시오. 리눅스는 GPL 이나 오라클은 그렇지 않습니다. 리눅스에서 운용되는 오라클은 반드시 GPL 을 사용하면서 배포되어야 하나요? 그렇지 않습니다. 그러므로 보유하신 소프트웨어는 어떤 라이선스 아래에서라도 원하시는 만큼 PostgreSQL/PostGIS 데이터베이스를 사용하실 수 있습니다.

하나의 예외를 더 들자면 본인께서 PostGIS 소스코드에 수정을 가하고 그것을 임의적으로 수정하신 PostGIS 를 배포하는 경우일 것입니다. 이러한 경우에는 수정하신 PostGIS 의 코드를 공유하셔야만 하실 수도 있습니다. (그러나 이것 외에서 작동하는 애플리케이션을 제하고)심지어 이렇게 제한된 경우에서도 binaries 를 배포하는 사람들에게만 소스 코드를 베포하시면 됩니다. GPL 은 binariaes 를 제공하는 사람들을 제외하고는 소스 코드 공개를 요구하지 않습니다

Chapter 4. PostGIS 사용하기: 데이터 매니지먼트 및 쿼리

4.1. GIS Objects

PostGIS가 지원하는 GIS 오브젝트들은 OpenGIS 컨소시엄(OGC)에 의해 정의되는 "간단한 피쳐(simple features)"들의 확대집합입니다. 0.9 버전 기준, PostGIS는 OGC "SQL을 위한 간단한 피쳐들" 설명서에 명시되어 있는 기능들 그리고 오브젝트들을 지원합니다.

PostGIS는 3DZ, 3DM and 4D 지원 표준을 확장합니다.

4.1.1. OpenGIS WKB 및 WKT

OpenGIS사양서는 공간 오브젝트들을 나타내는 두 가지 표준 방법을 정의합니다: (잘 알려진 Text(Well-Known Text (WKT)) 형태와 잘 알려진(Well-Known Binary (WKB)) 형태. 두WKT 와WKB 모두 오브젝트 타입과 오브젝트를 구성하는 좌표들에 대한 정보를 포함하고 있습니다.

피쳐들의 공간 오브젝트들의 텍스트 문자열표현들(WKT) 의 예들로는 다음과 같은 것들이 있습니다:

- POINT(0 0)
- LINESTRING(0 0,1 1,1 2)
- POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
- MULTIPOINT(0 0,1 2)
- MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
- MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))
- GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))

The OpenGIS 사양서는 또한 공간 오브젝트들의 내부 저장 포맷이 공간 관계 시스템 식별자(spatial referencing system identifier (SRID)). 데이터로의 삽입을 위한 공간 오브젝트들을 생성할 때 SRID은 필요합니다.

이러한 포맷들의 입/출력은 아래의 인터페이스들을 사용함으로써 이용 가능합니다:

```
bytea WKB = ST_AsBinary(geometry);
text WKT = ST_AsText(geometry);
geometry = ST_GeomFromWKB(bytea WKB, SRID);
geometry = ST_GeometryFromText(text WKT, SRID);
```

예를 들어, OGC 공간 오브젝트를 생성, 삽입하기 위한 유효 삽입문(valid insert statement)은 다음과 같습니다:

```
INSERT INTO geotable ( the_geom, the_name )
  VALUES ( ST_GeomFromText('POINT(-126.4 45.32)', 312), 'A Place');
```

4.1.2. PostGIS EWKB, EWKT 및 표준형(Canonical Forms)

OGC 포맷들은 2d 지오메트리들만을 지원하며 연합 SRID (associated SRID)는 입/출력 representations에 "절대" 보관되지 않습니다.

PostGIS은 확장된 포맷들은 현재 OGC one(모든 유효 WKB/WKT은 유효 EWKB/EWKT)의 확대집합이나 앞으로는 각기 다를 수 있습니다. 특히 OGC가 확장자들과 충돌되는 새로운 포맷으로 나올 경우를 예로 들 수 있습니다. 그러므로 이 피쳐에 의존하시면 안됩니다!

PostGIS EWKB/EWKT은 3dm, 3dz, 4d 좌표 지원 그리고 내장 SRID 정보를 첨가합니다.

피쳐들의 확장된 공간 오브젝트들의 문자열 표현(text representations (EWKT))의 예들은 다음과 같습니다. * 은 이 PostGIS 버전의 새로운 것을 나타냅니다:

- POINT(0 0 0) -- XYZ
- SRID=32632;POINT(0 0) -- XY with SRID
- POINTM(0 0 0) -- XYM
- POINT(0 0 0 0) -- XYZM
- SRID=4326;MULTIPOINTM(0 0 0,1 2 1) -- XYM with SRID
- MULTILINESTRING((0 0 0,1 1 0,1 2 1),(2 3 1,3 2 1,5 4 1))
- POLYGON((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1 0))
- MULTIPOLYGON(((0 0 0,4 0 0,4 4 0,0 4 0,0 0 0),(1 1 0,2 1 0,2 2 0,1 2 0,1 1 0)),((-1 -1 0,-1 -2 0,-2 -2 0,-2 -1 0,-1 -1 0)))
- GEOMETRYCOLLECTIONM(POINTM(2 3 9), LINESTRINGM(2 3 4, 3 4 5))
- MULTICURVE((0 0, 5 5), CIRCULARSTRING(4 0, 4 4, 8 4))
- POLYHEDRALSURFACE(((0 0 0, 0 0 1, 0 1 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)), ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)), ((1 1 0, 1 1 1, 1 0 1, 1 0 0, 1 1 0)), ((0 1 0, 0 1 1, 1 1 1, 1 1 0, 0 1 0)), ((0 0 1, 1 0 1, 1 1 1, 0 1 1, 0 0 1)))

- TRIANGLE ((0 0, 0 9, 9 0, 0 0))
- TIN(((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 0 0 0)))

이러한 포맷들의 입/출력은 다음의 인터페이스를 사용함으로써 이용할 수 있습니다:

```
bytea EWKB = ST_AsEWKB(geometry);
text EWKT = ST_AsEWKT(geometry);
geometry = ST_GeomFromEWKB(bytea EWKB);
geometry = ST_GeomFromEWKT(text EWKT);
```

예를 들어, OGC 공간 오브젝트를 생성, 삽입하기 위한 유효 삽입문(valid insert statement)은 다음과 같습니다:

```
INSERT INTO geotable ( the_geom, the_name )
  VALUES   (   ST_GeomFromEWKT('SRID=312;POINTM(-126.4   45.32   15)'),   'A
Place' )
```

The PostgreSQL 타입의 "표준형(canonical forms)"은 간단한 쿼리(어떠한 함수 호출 없이)를 통해 얻을 수 있는 representations입니다. 그리고 간단한 삽입, 업데이트 또는 복사를 통해 허용됩니다. Postgis '지오메트리(geometry)' 을 위해서 아래 내용을 입력하십시오.:

```
- Output
  - binary: EWKB
       ascii: HEXEWKB (EWKB in hex form)
- Input
  - binary: EWKB
       ascii: HEXEWKB|EWKT
```

예를 들어 이 statement는 EWKT을 읽으며 표준 아스키코드(canonical ascii)의 프로세스 내 HEXEWKB을 되돌립니다:

```
=# SELECT 'SRID=4;POINT(0 0)'::geometry;

geometry
----------------------------------------------------
0101000020040000000000000000000000000000000000000000
(1 row)
```

4.1.3. SQL-MM Part 3

SQL 멀티미디어 애플리케이션 공간 사양서는 SQL 스펙을 위한 간단한 피쳐들을 순환적으로 삽입된 곡선들을 정의함으로써 확장시킵니다.

SQL-MM 정의들은3dm, 3dz and 4d 좌표를(coordinates) 포함하나 SRID 정보 포함(임베딩, embedding)을 허용하지 않습니다.

잘 알려진 텍스트 확장자들(well-known text extensions)은 아직 완전히 지원되진 않습니다. 간단한 곡선의 지오메트리의 예들은 아래와 같습니다:

- CIRCULARSTRING(0 0, 1 1, 1 0)
  CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0)
  CIRCULARSTRING(원형의 스트링)은 직선의 세상의 linestring과 비슷한 것으로 간단한 곡선 타입입니다. 단일 활꼴은 세가지 점들(시작과 끝점;첫번째와 세번째, 그리고 호 위의 점)이 필요합니다. 이것에 대한 예외는 시작과 끝점이 같은 닫힌 원(closed circle)입니다. 이 경우에는 두번째 점은 반드시 호의 중앙에 위치해야 합니다. 즉 원의 반대 방향입니다. 호들을 함께 묶기 위해서는 linestring과 같이 앞의 호의 마지막 점이 다음 호의 첫번째 점이 되어야 합니다. 이것은 유효 원형 스트링은 반드시 1보다 홍수 점들을 가지고 있어야 함을 의미합니다.

- COMPOUNDCURVE(CIRCULARSTRING(0 0, 1 1, 1 0),(1 0, 0 1))
  compound curve(복심 곡선)은 곡선(circular) 활꼴들과 직선 활꼴들 모두를 가지고 있는 단일 연속 곡선입니다. 이는 잘 짜여진 구성요소(component)에 더하여 모든 구성요소(마지막을 제외하고)의 종점이 반드시 뒤에 따르는 부분의 시작 점과 일치해야 한다는 것을 의미합니다.

- CURVEPOLYGON(CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0),(1 1, 3 3, 3 1, 1 1))
  곡선 다각형 안의 복심 곡선의 예: CURVEPOLYGON(COMPOUNDCURVE(CIRCULARSTRING(0 0,2 0, 2 1, 2 3, 4 3),(4 3, 4 5, 1 4, 0 0)), CIRCULARSTRING(1.7 1, 1.4 0.4, 1.6 0.4, 1.6 0.5, 1.7 1))
  CURVEPOLYGON은 외륜과 0 또는 더 많은 내륜들로 이루어진 다각형과 같습니다. 차이점이 있다면 링은 원형 스트링, 직선 스트링 또는 합성 스트링의 형태를 가질 수 있다는 것입니다. PostGIS 1.4기준 PostGIS는 곡선 다각형의 합성 곡선들(compound curves)을 지원합니다.

- MULTICURVE((0 0, 5 5),CIRCULARSTRING(4 0, 4 4, 8 4))
  MULTICURVE은 직선 스트링들, 원형 스트링들 또는 합성 스트링들의 컬렉션입니다.

- MULTISURFACE(CURVEPOLYGON(CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0),(1 1, 3 3, 3 1, 1 1)),((10 10, 14 12, 11 10, 10 10),(11 11, 11.5 11, 11 11.5, 11 11)))
  이것은 다각형들 또는 곡선 다각형들이 될 수 있는 표면의 컬렉션입니다.

PostGIS 1.4. 이전 버전은 곡선 다각형의 합성 곡선들을 지원하지 않습니다. 그러나 PostGIS 1.4. 그리고 그 이상의 버전에서는 곡선 다각형의 합성 곡선들의 사용을 지원합니다.

SQL-MM 수행 내에서의 모든 부동소수점 비교들은 현재 1E-8 인 특정한 허용오차로 수행됩니다.

4.2. PostGIS Geography Type

지오그래피 타입은 “지오그래픽" 좌표들(때때로 “기하" 좌표들 혹은 “lat/lon” 또는 “lon/lat”이라고 불림)을 위해 표현되는 공간 피쳐들을 위한 native support를 제공합니다. 지오그래픽 좌표들은 angular units(degrees)에 표현된 구(spherical)좌표들입니다.

PostGIS 지오메트리를 위한 기반은 평면입니다. 평면 위 뒤 점들 사이의 가장 짧은 길은 직선입니다. 이는 지오메트리(지역, 거리, 길이, 교차 지점, 등)위 계산들이 데카르트(cargesian) 수학 및 직선 벡터들을 이용하여 산출될 수 있다는 것을 의미합니다.

PostGIS지오그래피 타입을 위한 토대는 구(sphere)입니다. 구 위 두 점 사이의 가장 짧은 길은 큰 원호입니다. 이는 지오그래픽 (지역, 거리, 길이, 교차점, 등)위 계산들은 보다 복잡한 수학을 이용하여 반드시 구 위에서 산출되어야 함을 의미합니다. 보다 정교한 계산을 위해서는 실제 세계의 실제 회전 타원체 형태를 계산에 반드시 참작하여야 하며 수학은 아주 복잡해 집니다.

기본적 수학은 보다 매우 복잡하기 때문에 지오그래피 타입을 위해 정의된 기능들이 지오메트리 타입을 위해 정의된 기능 보다 그 수가 적습니다. 시간이 흐르면서 새로운 알고리즘들이 첨가되고 지오그래피 타입의 능력들이 확장될 것입니다.

한 가지 제약사항은 이것은 오직 WGS 84 long lat (SRID:4326)만 지원한다는 것입니다. 지오그래피라고 불리는 새로운 데이터 타입을 사용합니다. 그 어떤 GEOS 기능들도 이 새로운 타입을 지원하지 않습니다. 회피방법은 지오메트리 타입과 지오그래피 타입 사이를 왔다 갔다 전환이 가능합니다.

새로운 지오그래피 타입은 PostgreSQL 8.3+ typmod 정의 포맷을 사용함으로서 지오그래피 필드 테이블은 단일 단계 안에서 첨가될 수 있습니다. 곡선들을 제외한 모든 표준 OGC 포맷들은 지원됩니다.

4.2.1. Geography Basics

지오그래피 타입은 오직 아주 간단한 피쳐들만을 지원합니다. 만약 이것이 SRID 4326이라면 표준 지오메트리 타입 데이터는 지오그래피는 위해서 오토캐스트 할 것입니다. 데이터를 삽입하기 위해서 EWKT 그리고 EWKB 규약들(conventions) 또한 사용할 수 있습니다.

- POINT: 2d 점과 함께 지오메트리 테이블을 생성:
```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINT,4326) );
```
- Z 좌표 점과 함께 테이블 생성
```
CREATE TABLE testgeog(gid serial PRIMARY KEY, the_geog geography(POINTZ,4326) );
```
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

새로운 지오그래피 필드들은 geometry_columns에 등록되지 않습니다. 새로운 지오그래피 필드들은 geography_columns 이라고 불리는 새로운 뷰에 등록됩니다. geography_columns 은 시스템 카탈로그 배경의 뷰로써 AddGeom와 같은 함수가 없어도 자동적으로 업데이트 됩니다.

자 이제 "geography_columns" 뷰를 체크하시고 어떤 테이블이 나열되었는지 살펴보십시오.

CREATE TABLE syntax 사용함으로써 지오그래피 행을 가진 테이블을 생성하실 수 있습니다. 지오메트리와는 다르게 메타데이터(metadata)에 행을 등록하기 위해 독립된 AddGeometryColumns()프로세스를 실행할 필요가 없습니다.

```
CREATE TABLE global_points (
   id SERIAL PRIMARY KEY,
   name VARCHAR(64),
   location GEOGRAPHY(POINT,4326)
  );
```

위치 행은 타입 지오그래피를 가지고 있으며 해당 지오그래피 타입은 두 개의 선택적 수식자(optional modifiers) 지원합니다: 타입 수신자는 행(column)안에서 허용된 면적들과 형태들의 종류들을 제한합니다; SRID 수신자는 좌표 레퍼런스 수신자(coordinate reference identifier)를 특정 숫자로 제한합니다.

타입 수신자를 위해 허용되는 값들은 다음과 같습니다: POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON. 수신자는 또한 접미사들(suffixes)을 통해 차원수 규제들을 지원합니다: Z, M 그리고 ZM. 예를 들어 'LINESTRINGM'의 수신자는 오직 3차원 평면들 내 오직 line strings만 허용하며 제3차원을 척도로 여깁니다. 유사하게 Similarly, 'POINTZM'는 사차원 평면 데이터를 요구합니다.

SRID 수신사는 현재 제한적인 유용성을 가지고 있습니다: 오직 4326 (WGS84) 만이 값으로서 허용됩니다. SRID를 명시하지 않으실 경우에는 값 0 (정의되지 않는 회전타원체, undefined spheroid)가 사용될 것이며 모든 계산들은 WGS84를 사용함으로서 진행될 것입니다.

향후에는 번갈아 생기는 SIRDs가 WGS84 아닌 회전타원체 위에서의 계산들을 허용할 것입니다.

테이블을 생성한 뒤 GEOGRAPHY_COLUMNS table 내에서 이를 확인할 수 있습니다:

```
-- See the contents of the metadata view
SELECT * FROM geography_columns;
```

지오메트리 행을 사용한 것과 같이 데이터를 테이블에 삽입할 수 있습니다:

```
-- Add some data into the test table
INSERT    INTO    global_points    (name,    location)    VALUES    ('Town',
ST_GeographyFromText('SRID=4326;POINT(-110 30)') );
INSERT    INTO    global_points    (name,    location)    VALUES    ('Forest',
ST_GeographyFromText('SRID=4326;POINT(-109 29)') );
INSERT    INTO    global_points    (name,    location)    VALUES    ('London',
ST_GeographyFromText('SRID=4326;POINT(0 49)') );
```

인덱스를 생성하는 작업은 지오메트리 경우와 같습니다. PostGIS는 행 타입이 지오그래피라는 것을 인지하고 적절한 구(sphere) 기반 인덱스를 (지오메트리를 위해 사용되는) 보통 평면 인덱스 대신에 생성합니다.

```
-- Index the test table with a spherical index
```

```
CREATE INDEX global_points_gix ON global_points USING GIST ( location );
```

쿼리 그리고 계측 기능은 미터 단위들을 사용합니다. 그러므로 거리 파리미터들은 반드시 미터로 표현되어야 하며 반환값들은 반드시 미터로 예상되어야 합니다(또는 면적을 위한 평방 미터)

```
-- Show a distance query and note, London is outside the 1000km tolerance
  SELECT    name    FROM    global_points    WHERE    ST_DWithin(location,
ST_GeographyFromText('SRID=4326;POINT(-110 29)'), 1000000);
```

비행기로 시애틀에서 런던까지 얼마나 가까운지 계산해봄으로써 작동하는 지오그래피의 파워를 직접 보실 수 있습니다. (LINESTRING(-122.33 47.606, 0.0 51.5))이 Reykjavik (POINT(-21.96 64.15)으로 됩니다.)

```
-- Distance calculation using GEOGRAPHY (122.2km)
  SELECT  ST_Distance('LINESTRING(-122.33  47.606,  0.0  51.5)'::geography,
'POINT(-21.96 64.15)':: geography);
```

```
-- Distance calculation using GEOMETRY (13.3 "degrees")
  SELECT  ST_Distance('LINESTRING(-122.33  47.606,  0.0  51.5)'::geometry,
'POINT(-21.96 64.15)':: geometry);
```

GEOGRAPHY 타입은 Reykjavik와 시애틀과 런던 사이의 대권 비행 경로 사이 구 위 실제로 가장 짧은 거리를 계산합니다.

Great Circle mapper 지오메트리 타입은 Reykjavik 와 편평한 세계지도 위 시애틀에서 런던까지의 직선 거리 경로 사이의 의미 없는 데카르트 거리를 계산합니다. 결과의 아주 작은 단위들은 “도(degrees)”라고 합니다. 그러나 결과는 점들 간의 그 어떤 실제 각 차이에 상응하지 않습니다. 그러므로 “도”라고 부르는 것은 잘못된 것입니다.

4.2.2. 지오메트리 데이터 타입 보다 지오그래피 데이터 타입을 사용하여야 할 경우

새로운 지오그래피 타입은 경도/위도 좌표들 위에 데이터를 저장할 수 있게 허용합니다. 그러나 비용이 듭니다: 몇 기능들은 지오메트리 위에 정의 되기 보다는 지오그래피 위에 정의되어 있습니다; 정의된 이러한 기능들은 실행되는데 더 많은 GPU 시간을 소요합니다.

선택하시는 타입은 구축하고 계신 애플리케이션의 예상되는 작업 영역 위에 반드시 길들여져야 합니다. 본인 데이터는 전세계 또는 큰 대륙 영역을 포괄하고 있습니까? 혹은 로컬에서 주— 국가 또는 지방자치제 입니까?

- 만약 본인의 데이터가 작은 영역에 포괄되어 있다면 적절한 투사도를 선택하고 지오메트리를 사용하는 것이 퍼포먼스와 이용 가능한 기능적인 측면에서 최선의 솔루션일 것입니다.

- 만약 본인의 데이터가 전세계의 데이터 이거나 대륙 영역들을 포함한다면 투사 세부 사항들 없이 시스템을 지오그래피를 통해 설치할 수 있음을 발견할 수 있습니다. 적도/위도에 본인은 데이터를 저장하고 지오그래피에 정의된 기능들을 사용합니다.

- 투사도를 이해하지 못하고 또 배우기를 원하지 않을 경우 더하여 지오그래피 기능들에 존재하는 한계에 개의치 않다면 지오메트리 보다 지오그래피를 사용하시는 것이 오히려 본인을 위해 더 쉬울 수 있습니다. 간단히 적도/위도로 데이터를 로딩하시면서 시작하십시오.

지오그래피 vs 지오메트리 각 각을 위해 서포트되는 것들을 비교 하기 위해선 13.10 단락을 참조하십시오. 지오그래피 기능들의 간단한 목록과 기술을 위해서는 13.3 단락을 참조하십시오.

4.2.3. 지오그래피 고급 FAQ

1. 구(sphere) 위에서 계산합니까 아니면 회전타원체 위에서 계산합니까?

디폴트에서 모든 거리와 지역 계산들은 구 위에서 이루어 집니다. 좋은 로컬 투사도 내 로컬 평면 결과들과 로컬 지역 내 계산 결과들이 일치하는지 찾으셔야 합니다. 더 넓은 지역에 타원체 계산들은 투영 평면 위에 이루어진 그 어떤 계산보다 정확합니다.

모든 지오그래피 기능들은 최종 불 파라미터를 "FALSE"로 설정함으로써 구(sphere) 계산을 이용하는 옵션을 가지고 있습니다. 이렇게 함으로써 계산 속도가 향상됩니다. 특히 지오메트리가 가장 단순한 경우들에 의해서 계산 속도가 향상됩니다.

2. 날짜변경선과 남/북극은 어떻게 되는 것인가요?

모든 계산들은 날짜변경선 또는 남/북극의 개념을 가지고 있지 않습니다. 좌표들은 구 모양입니다(적도/위도) 그러므로 날짜변경선을 가로지르는 형태는 연산 관점에서 그 어떤 형태들과 다르지 않습니다.

3. 처리할 수 있는 가장 긴 호는 무엇입니까?

대원호는 두 개의 점들 사이의 "보간 선(interpolation line)"처럼 사용됩니다. 이 는 대원을 따라 어떤 방향으로 이동하냐에 따라 모든 두 개의 점들은 실제 두 가지 방법으로 연합된다는 것을 의미합니다. 모든 코드는 대원을 따라 두 개의 길들이 "shroter"에 따라 점들이 연결된다고 가정합니다. 결과적으로, 180 도 이상의 호들은 바르게 만들어지지 않습니다.

4. 유럽/러시아 지역을 계산하고 큰 지리적 영역을 삽입할 때 왜 이렇게 느린가요?

왜냐하면 다각형은 너무 거대하기 때문입니다! 큰 지역들은 두 가지 이유로 좋지 않습니다: 한도가 너무 크고 인덱스는 어떠한 쿼리를 실행하든지 피쳐를 풀(full)하는 경향이 있습니다; 꼭지점의 수는

거대하며 테스트들(거리, 억제) 은 꼭지점 명단을 반드시 한 번은 또는 N 번은 가로질러야 합니다(다른 후보 피쳐에서의 꼭지점 수가 N 일 경우와 함께)

지오메트리와 마찬가지, 매우 큰 다각형을 가지고 계시지만 작은 지역들에서 쿼리들 실행하고 계실 때 기하 데이터를 작은 덩어리들로 "비정규화(denormalize)"하십시오. 그를 통해 인덱스는 효율적으로 오브젝트의 부분들을 하위 쿼리(subquery)를 수행할 수 있으며 쿼리들은 전체 오브젝트들을 매 순간 빼낼 (pull out)필요가 없습니다. 모든 유럽을 하나의 다각형에 저장할 수 있다는 사실은 반드시 그렇게 해야 된다는 것을 의미하는 것은 아닙니다.

4.3. OpenGIS 표준 이용

OpenGIS "SQL을 위한 간단한 피쳐 사양서(Simple Features Specification for SQL)"은 표준 GIS 오브젝트 타입들, 그 타입들을 처리하기 위해 요구되는 기능들 그리고 메타데이터 테이블 세트를 정의합니다. 메타 데이터가 일관되게 남는 것을 보장하기 위해 공간 행을 생성 및 제거하는 작업들이 OpenGIS에 의해 정의된 공간 절차들 진행되는 과정 중 수행됩니다.

두 가지 메타-데이터 테이블들이 존재합니다: `SPATIAL_REF_SYS` 그리고 `GEOMETRY_COLUMNS`. `SPATIAL_REF_SYS` 테이블은 수치 ID 들 그리고 공간 데이터베이스에 사용되는 좌표 시스템의 원문 기술을 보유하고 있습니다.

4.3.1. The SPATIAL_REF_SYS 테이블 및 공간 레퍼런스 시스템들

spatial_ref_sys 테이블은 PostGIS로써 3000개가 넘는 공간 레퍼런스 시스템들과 그것들 사이를 변형/재투영하기 위해 필요한 정보들을 열거하는 OGC 인식 데이터베이스 테이블을 포함하고 있습니다.

비록 PostGIS spatial_ref_sys 테이블은 Proj 라이브러리로 처리될 수 있는 가장 일반적으로 사용되는 공간 레퍼런스 시스템 정의들을 3000여개 넘게 보유하고 있지만 알려진 모든 것을 보유하고 있는 것은 아니며 Proj 구조들에 익숙하시다면 심지어 자신만의 맞춤 재투영을 정의할 수 있습니다. 대부분의 공간 레퍼런스 시스템들은 지방별이며 정해진 지방 밖에 관해서는 그 사용이 무의미함을 인지하십시오.

코어 세트에 정의되지 않는 공간 레퍼런스 시스템들을 찾기 위한 아주 좋은 소스는 http://spatialreference.org/입니다.

보다 더 널리 사용되어지는 공간 레퍼런스 시스템들은 다음의 것들이 있습니다: 4326 - WGS 84 Long Lat, 4269 - NAD 83 Long Lat, 3395 - WGS 84 World Mercator, 2163 - US National Atlas Equal Area, 각 공간 NAD 83, WGS 84 UTM zone - UTM zones들을 위한 공간 레퍼런스 시스템들은 측정을 위해 가장 이상적인 것들 중 하나 이지만 오직 6도 지역들만 다룹니다.

다양한 US 주 평면 공간 레퍼런스 시스템들(미터 혹은 피트 기준) – 주로 US 주당 하나 혹은 2 출구들이 존재합니다. 대부분 미터가 기준인 것들은 코어 세트(core set) 안에 있습니다. 하지만 대다수의 피트가 기준인 것들 혹은 ESRI가 생성한 것들은 spatialreference.org로부터 끌어오셔야 합니다.

관심있는 지역을 위해 어떤 UTM 존을 사용해야 하는지에 대한 결정에 관한 보다 많은 정보를 원하신다면 utmzone PostGIS plpgsql helper function 을 체크하십시오.

SPATIAL_REF_SYS 테이블 정의는 다음과 같습니다:

```
CREATE TABLE spatial_ref_sys (
  srid      INTEGER NOT NULL PRIMARY KEY,
  auth_name  VARCHAR(256),
  auth_srid  INTEGER,
  srtext    VARCHAR(2048),
  proj4text  VARCHAR(2048)
)
```

SPATIAL_REF_SYS 행들(columns)은 다음과 같습니다:

SRID 특유의 형태로 데이터베이스 이내 공간 레퍼런싱 시스템(SRS)를 발견하는 정수 값.

AUTH_NAME 이 레퍼런스 시스템을 위해 소환되는 표준 또는 표준화 바디(body). 예를 들어 "EPSG" 는 유요한 AUTH_NAME 입니다.

AUTH_SRID AUTH_NAME 에 소환된 권환에 의해 정의된 공간 레퍼런스 시스템의 ID. EPSG 의 경우 이것이 바로 EPSG 투영 코드가 갈 곳 입니다.

SRTEXT The 공간 레퍼런스 시스템의 익히 알려진 문자열 표현(Well-Known Text representation). WKT SRS representation 의 예는:

```
PROJCS["NAD83 / UTM Zone 10N",
  GEOGCS["NAD83",
        DATUM["North_American_Datum_1983",
          SPHEROID["GRS 1980",6378137,298.257222101]
        ],
        PRIMEM["Greenwich",0],
        UNIT["degree",0.0174532925199433]
  ],
  PROJECTION["Transverse_Mercator"],
  PARAMETER["latitude_of_origin",0],
  PARAMETER["central_meridian",-123],
  PARAMETER["scale_factor",0.9996],
  PARAMETER["false_easting",500000],
  PARAMETER["false_northing",0],
  UNIT["metre",1]
]
```

EPSG 코드들의 리스트와 그 코드들에 상응하는 WKT 표현들은 위해서는 http://www.opengeospatial.org/을 살펴보십시오. WKT 에 관한 일반적 토론을 보려면 http://www.opengeospatial.org/standards 의 OpenGIS "좌표 변환 서비스 실행 설명서(Coordinate Transformation Services Implementation Specification)"를 참고 하십시오. European Petroleum Survey Group (EPSG)와 EPSG 의 공간 레퍼런스 시스템들에 관한 보다 많은 정보를 위해서는 http://www.epsg.org 를 참고하십시오.

PROJ4TEXT 좌표 변환 능력들을 제공하기 위해 PostGIS 는 Proj 라이브러리를 사용합니다. PROJ4TEXT 행은 특정 SRID 를 위한 PROJ4 좌표 정의 문자열 (coordinate definition string)을 포함합니다. 예를 들어:

```
+proj=utm +zone=10 +ellps=clrk66 +datum=NAD27 +units=m
```

보다 많은 정보를 원하신다면 Proj4 웹사이트 http://trac.osgeo.org/proj/를 참조하십시오. `spatial_ref_sys.sql` 파일은 모든 EPSG 투영들을 위한 `SRTEXT`와 `PROJ4TEXT` 양쪽의 정의를 포함합니다.

4.3.2. 지오메트리 행 뷰(THE GEOMETRY_COLUMNS VIEW)

2.0. 이전 버전 PostGIS에서 geometry_columns은 직접적으로 바로 편집될 수 있는 테이블이었습니다. 더하여 때때로 지오메트리 행들의 실제 기능들과 싱크되지 않기도 하였습니다. In PostGIS 2.0.0에서는, GEOMETRY_COLUMNS는 이전 버전과 동일한 앞면 구조를 가진 뷰이지만 데이터베이스 시스템 카탈로그에서 읽어옵니다. 구조는 다음과 같습니다:

```
\d geometry_columns
```

```
View "public.geometry_columns"
     Column      |         Type          | Modifiers
-----------------+-----------------------+-----------
 f_table_catalog   | character varying(256) |
 f_table_schema    | character varying(256) |
 f_table_name      | character varying(256) |
 f_geometry_column | character varying(256) |
 coord_dimension   | integer                |
 srid              | integer                |
 type              | character varying(30)  |
```

행 의미들(column meanings)은 이전 버전들과 동일하면 다음과 같습니다:

F_TABLE_CATALOG, F_TABLE_SCHEMA, F_TABLE_NAME 지오메트리 행을 포함하는 피쳐 테이블의 정규화된 이름. "카탈로그(catalog)" 그리고 "스키마(schema)" 이 용어들은 오라클화 된 용어들임을 주목하십시오. "카탈로그"의 PostgreSQL 아날로그는 없으며 따라서 행은 비어있습니다 – "스키마"의 경우 PostgreSQL 스키마 이름은 사용됩니다 (public이 디폴트입니다).

F_GEOMETRY_COLUMN 피쳐 테이블 geometry 행의 이름

COORD_DIMENSION 행의 공간 차원 (2, 3 또는 4 차원적)

SRID 이 테이블의 좌표 지오메트리를 위해 사용되는 공간 레퍼런스 시스템의 ID. 이것은 SPATIAL_REF_SYS에게 외래 키 레퍼런스(foreign key reference)입니다.

TYPE 공간 오브젝트의 타입. 단열형에 공간 행을 제한시키기 위해 POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION 중 하나를 사용하거나 상응하는 XYM 버전들 POINTM, LINESTRINGM, POLYGONM, MULTIPOINTM, MULTILINESTRINGM, MULTIPOLYGONM, GEOMETRYCOLLECTIONM을 사용하십시오. 이기종(혼합형; mixed-type) 컬렉션들을 위해서는 "GEOMETRY" 는 타입처럼 사용하실 수 있습니다.

이 속성은(아마도) OpenGIS 사양의 한 부분이 아닐 것입니다. 그러나 타입 동질성을 위해서 요구됩니다.

4.3.3. 공간테이블 생성하기

공간 데이터와 함께 테이블을 생성하는 것은 한 번에 이루어질 수 있습니다. WGS84 long lat내 2D line string을 가진 로드 테이블(roads table)을 생성하는 지오메트리 행의 예시는 아래의 예시와 같습니다.

```
CREATE TABLE ROADS ( ID int4
             , ROAD_NAME varchar(25), geom geometry(LINESTRING,4326) );
```

3-D Line string을 추가하는 다음의 예시와 같이 표준 ALTER 테이블 명령어를 사용함으로써 추가적 행들을 첨부할 수 있습니다.

```
ALTER TABLE roads ADD COLUMN geom2 geometry(LINESTRINGZ,4326);
```

이전 기존 호환성을 위해 데스크 관리 기능을 사용함으로써 두 가지 스테이지들 내 공간 테이블을 여전히 생성할 수 있습니다.

- 보통 비공간 테이블을 생성.
  예: **CREATE TABLE ROADS (ID int4, ROAD_NAME varchar(25))**

- OpenGIS “AddGeometryColumn" 기능을 사용하여 해당테이블에 공간 행을 추가하십시오. 보다 많은 정보를 위해서는 AddGeometryColumn 을 참고하십시오.

syntax는:

```
AddGeometryColumn(
  <schema_name>,
  <table_name>,
  <column_name>,
  <srid>,
  <type>,
  <dimension>
)
```

또는 현재 스키마를 사용하십시오

```
AddGeometryColumn(
  <table_name>,
  <column_name>,
  <srid>,
  <type>,
  <dimension>
)
```

예1: **SELECT AddGeometryColumn('public', 'roads', 'geom', 423, 'LINESTRING', 2)**

예2: **SELECT AddGeometryColumn('roads', 'geom', 423, 'LINESTRING', 2)**

테이블을 생성하고 공간 행을 추가하기 위해 사용된 SQL의 예시는 다음과 같습니다. (SRID of 128 이 이미 존재한다고 가정한다.):

```
CREATE TABLE parks (
  park_id   INTEGER,
  park_name  VARCHAR,
  park_date  DATE,
  park_type  VARCHAR
);
SELECT AddGeometryColumn('parks', 'park_geom', 128, 'MULTIPOLYGON', 2 );
```

총칭 "지오메트리"타입과 정의되지 않는 0의 SRID 값을 사용하는 또 다른 예는 다음과 같습니다:

```
CREATE TABLE roads (
  road_id INTEGER,
  road_name VARCHAR
);
SELECT AddGeometryColumn( 'roads', 'roads_geom', 0, 'GEOMETRY', 3 );
```

4.3.4. 수동으로 지오메트리 행들(geometry_columns)에 지오메트리 등록하기

AddGeometryColumn() 처리법은 지오메트리 행을 생성하며 geometry_columns 테이블의 새로운 테이블을 등록합니다. 만약 본인의 소프트웨어가 geometry_columns을 활용한다면, then 쿼리가 필요하신 모든 geometry columns은 반드시 뷰에 등록되어야 합니다. PostGIS 2.0을 시작으로 geometry_columns은 더 이상 편집 가능하지 않으며 모든 지오메트리 행들은 자동 등록됩니다.

그러나 만약 생성과정에서 행이 특정 타입으로 정의되지 않는 다면 포괄적 지오메트리로서 등록될 수도 있습니다.

이러한 경우가 발생할 수 있는 두 가지 경우는 (그러나 AddGeometryColumn 을 사용할 수 없는) SQL뷰들 그리고 벌크(Bulk) 삽입이 존재합니다. 이러한 경우들 위해서 지오메트리 행 테이블 내 등록을 수정 할 수 있습니다. Note in PostGIS 2.0+ 에서는 행은 typmod 기반이며 생성 프로세스는 올바르게 이것을 등록시킬 것이므로 다른 어떤 작업할 필요가 없습니다.

```
--Lets say you have a view created like this
CREATE VIEW  public.vwmytablemercator AS
       SELECT gid, ST_Transform(geom,3395) As geom, f_name
```

```
       FROM public.mytable;

-- For it to register correctly in PostGIS 2.0+
-- You need to cast the geometry
--
DROP VIEW public.vwmytablemercator;
CREATE VIEW  public.vwmytablemercator AS
       SELECT  gid,  ST_Transform(geom,3395)::geometry(Geometry,  3395)  As
geom, f_name
       FROM public.mytable;

-- If you know the geometry type for sure is a 2D POLYGON then you could
do
DROP VIEW public.vwmytablemercator;
CREATE VIEW  public.vwmytablemercator AS
       SELECT  gid,  ST_Transform(geom,3395)::geometry(Polygon,  3395)  As
geom, f_name
       FROM public.mytable;
```

```
--Lets say you created a derivative table by doing a bulk insert
SELECT poi.gid, poi.geom, citybounds.city_name
INTO myschema.my_special_pois
FROM   poi   INNER   JOIN   citybounds   ON   ST_Intersects(citybounds.geom,
poi.geom);

--Create 2d index on new table
CREATE INDEX idx_myschema_myspecialpois_geom_gist
  ON myschema.my_special_pois USING gist(geom);

-- If your points are 3D points or 3M points,
-- then you might want to create an nd index instead of a 2d index
-- like so
CREATE INDEX my_special_pois_geom_gist_nd
       ON my_special_pois USING gist(geom gist_geometry_ops_nd);

--To   manually   register   this   new   table's   geometry   column   in
geometry_columns
-- Note that this approach will work for both PostGIS 2.0+ and PostGIS
1.4+
-- For PostGIS 2.0 it will also change the underlying structure of the
table to
-- to make the column typmod based.
-- For PostGIS prior to 2.0, this technique can also be used to register
views
SELECT populate_geometry_columns('myschema.my_special_pois'::regclass);

--If you are using PostGIS 2.0 and for whatever reason, you
-- you need the old constraint based definition behavior
-- (such as case of inherited tables where all children do not have the
same type and srid)
-- set new optional  use_typmod argument to false
SELECT     populate_geometry_columns('myschema.my_special_pois'::regclass,
false);
```

비록 예전 구속 기반 방법(the old-constraint based method)가 여전히 지원되더라도 직접적으로 뷰 안에서 사용되는 구속 기반 지오메트리 행은 typmod 처럼 올바르게 geometry_columns안에 등록되지 않을 것입니다. 이 예시에서 typmod와 다른 구속 제약들을 사용함으로써 행을 정의해 볼 것입니다.

```
CREATE TABLE pois_ny(gid SERIAL PRIMARY KEY
   , poi_name text, cat varchar(20)
   , geom geometry(POINT,4326) );
SELECT   AddGeometryColumn('pois_ny',   'geom_2160',   2160,   'POINT',   2,
false);
```

psql에서 실행할 경우

```
\d pois_ny;
```

다르게 정의되는 것을 확인할 수 있습니다 --하나는 typmod이고, 다른 하나는 제약(constraint)입니다.

```
Table "public.pois_ny"
 Column    |     Type            |            Modifiers
-----------+---------------------+-----------------------------------------------------
 gid       | integer             | not null default nextval('pois_ny_gid_seq'::regclass)
 poi_name  | text                |
 cat       | character varying(20)|
 geom      | geometry(Point,4326) |
 geom_2160| geometry             |
Indexes:
   "pois_ny_pkey" PRIMARY KEY, btree (gid)
Check constraints:
   "enforce_dims_geom_2160" CHECK (st_ndims(geom_2160) = 2)
   "enforce_geotype_geom_2160" CHECK (geometrytype(geom_2160) = 'POINT'::text
       OR geom_2160 IS NULL)
   "enforce_srid_geom_2160" CHECK (st_srid(geom_2160) = 2160)
```

geometry_columns에서는 두 개 모두 올바르게 등록됩니다.

```
SELECT f_table_name, f_geometry_column, srid, type
       FROM geometry_columns
       WHERE f_table_name = 'pois_ny';
```

```
f_table_name | f_geometry_column | srid | type
-------------+-------------------+------+-------
pois_ny      | geom              | 4326 | POINT
pois_ny      | geom_2160         | 2160 | POINT
```

그러나 만약 다음과 같은 뷰를 생성할 경우

```
CREATE VIEW vw_pois_ny_parks AS
SELECT *
  FROM pois_ny
  WHERE cat='park';
```

```
SELECT f_table_name, f_geometry_column, srid, type
       FROM geometry_columns
       WHERE f_table_name = 'vw_pois_ny_parks';
```

The typmod 기반 geom 뷰 행(geom view column)은 올바르게 등록되지만 구속 기간은 그렇지 못합니다.

```
f_table_name      | f_geometry_column | srid |   type
------------------+-------------------+------+----------
 vw_pois_ny_parks | geom              | 4326 | POINT
 vw_pois_ny_parks | geom_2160         |    0 | GEOMETRY
```

PostGIS 후속 버전에서 변화가 있을 수도 있지만 현재 구속 기반 뷰 행을 올바르게 강제 등록 하기 위해선 아래의 것을 행하셔야 합니다:

```
DROP VIEW vw_pois_ny_parks;
CREATE VIEW vw_pois_ny_parks AS
SELECT gid, poi_name, cat
  , geom
  , geom_2160::geometry(POINT,2160) As geom_2160
  FROM pois_ny
  WHERE cat='park';
```

```
SELECT f_table_name, f_geometry_column, srid, type
       FROM geometry_columns
       WHERE f_table_name = 'vw_pois_ny_parks';
```

```
    f_table_name   | f_geometry_column | srid | type
------------------+-------------------+------+-------
 vw_pois_ny_parks | geom              | 4326 | POINT
 vw_pois_ny_parks | geom_2160         | 2160 | POINT
```

4.3.5. 지오메트리의 OpenGIS 법규 준수 보장하기

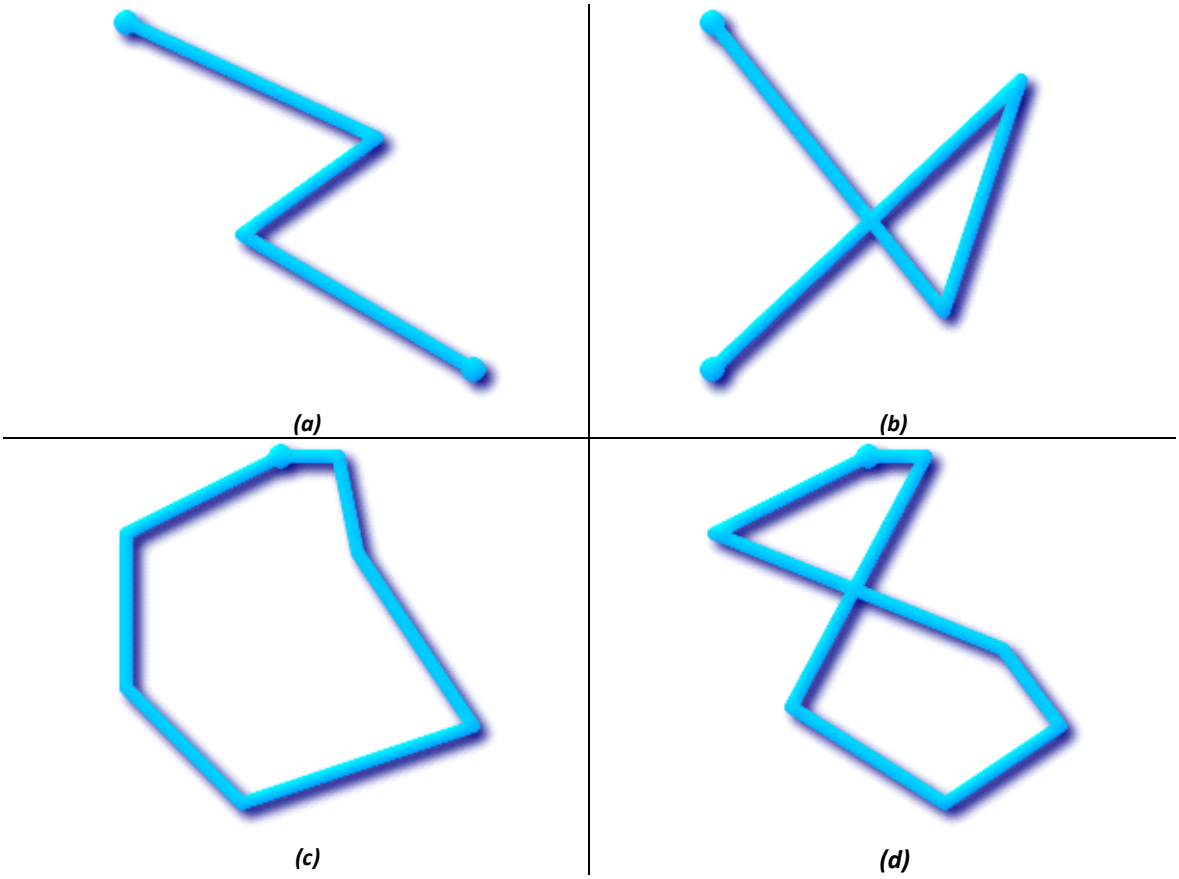
PostGIS 은 Open Geospatial Consortium의 (OGC) OpenGIS 사양에 부응합니다. 많은 PostGIS 방식들은 요구하거나 혹은 보다 더 정확하게 간단하고 유효하게 운용되는 지오메트리들을 추정합니다. 예를 들어 다각형 밖에 윤곽을 드러낸 구멍을 가진 다각형 영역을 계산하거나 혹은 간단하지 않는 경계선으로부터 다각형을 그리는 것은 이치에 맞지 않습니다.

OGC 설명서에 따르면 간단한 지오메트리는 자기 교착 혹은 자기 접촉 같은 변칙 기하 점들을 가지고 있지 않으며 주로 0 또는 1 차원적 지오메트리을 나타냅니다 (i.e. `[MULTI]POINT`, `[MULTI]LINESTRING`). 지오메트리 유효성(Geometry validity)은 반면에 주로 2 –차원적 지오메트리를 나타내며(i.e. `[MULTI]POLYGON`) 유효 다각형을 특징 짓는 assertions 세트를 정의합니다. 각 기하 클래스(geometric class)의 기술은 더 자세하게 기하학적 유사성 및 유효성을 열거하는 특정 조건들을 포함합니다.

`POINT` 은 본래 것으로 0 차원적 기하 오브젝트 같이 **간단**합니다.

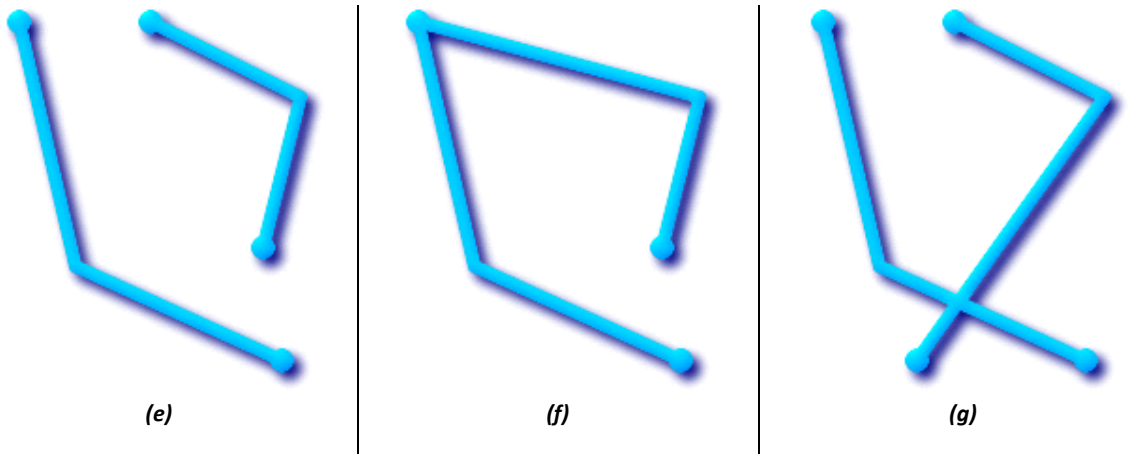
MULTIPOINTs 은 두 좌표들이 (POINTs)동일 하지 않을 때 **심플**합니다(동일한 좌표 값을 가진).

LINESTRING 은 같은 POINT 을 두 번 지나치질 않을 때 **심플**합니다(linear ring 이라 여겨지고 더하여 닫혔다고 여겨지는 종점들을 제하고).



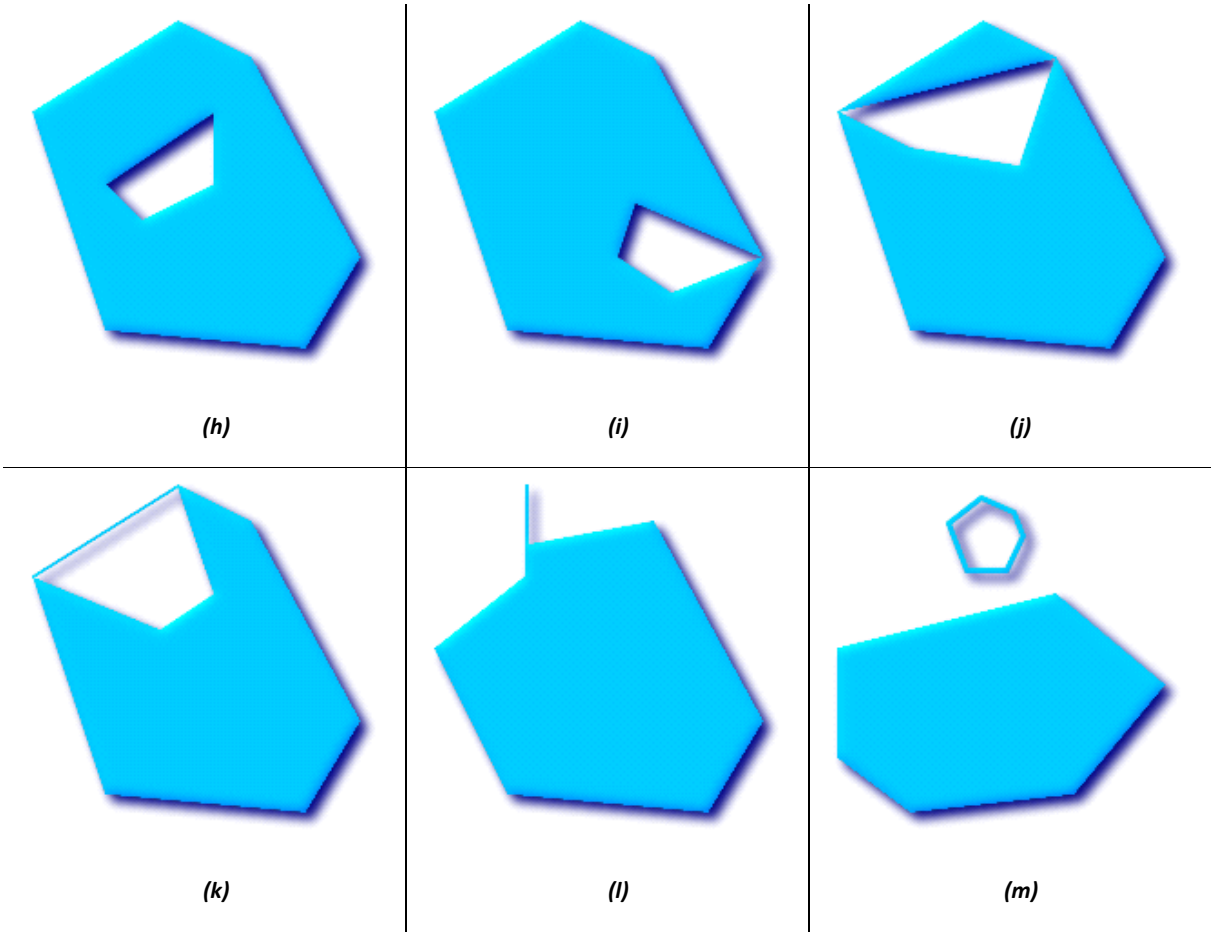
(a) 및 **(c)** 는 심플한 LINESTRING들이나, **(b)** 와 **(d)** 는 그렇지 않습니다.

MULTILINESTRING은 이것의 모든 부분들이 심플하고 두 부분들의 경계선들 위 POINTs들 사이에 교차가 발생할 시에만 **심플**합니다.



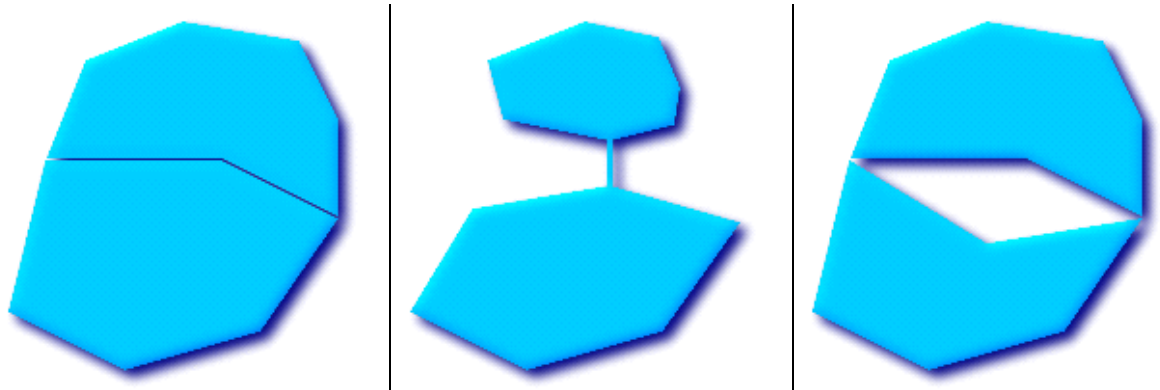
(e) 및 **(f)** 는 심플한 MULTILINESTRING들이나, **(g)** 는 그렇지 않습니다.

정의상, POLYGON은 언제나 심플합니다. 경계선 위 (외부 고리와 내부 고리들로 만들어진) 교차하는 두 개의 고리들이 없는 한 유효합니다. POLYGON 경계선은 한 POINT에서 교차할 수 있으나 오직 접선으로서만 교차합니다. (즉, 선 위가 아닌). POLYGON은 선들 또는 스파이크(spikes)들을 교차하지 않을 수 도 있으며 내부 고리들은 반드시 외부 교리 내에 완전히 함유되어야 합니다.



(h) 와 **(i)** 는 유효한 다각형들(valid POLYGONs)이며 **(j-m)** 은 단일 다각형들로 표현될 수 없습니다. 그러나 **(j)**와**(m)은 유요한** MULTIPOLYGON으로 표현될 수 있습니다.

MULTIPOLYGON 은 모든 부분들이 유효하고 내부에 있는 두 개의 부분들이 교차하지 않는다면 유효합니다. 두 개의 부분들은 경계선들은 접촉할 수 있으나 점들이 한정된 수일 때만 가능합니다.



(n) *(o)* *(p)*

(n) 와 **(o)**은 유효한 MULTIPOLYGON들이 아닙니다. 그러나 **(p)**은 유효합니다.

GEOS 라이브러리에 의해 시행된 대부분의 기능들은 지오메트리들이 OpenGIS 심플 피쳐 사양에 따라 규정된 것처럼 유효하다는 가정에 입각하고 있습니다. 지오메트리의 단순성과 유효성을 체크하기 위해서 ST_IsSimple() 및 ST_IsValid()을 사용하실 수 있습니다.

```
-- Typically, it doesn't make sense to check
-- for validity on linear features since it will always return TRUE.
-- But in this example, PostGIS extends the definition of the OGC IsValid
--  by returning false if a LineString has less than 2 *distinct*
vertices.
    gisdb=# SELECT
       ST_IsValid('LINESTRING(0 0, 1 1)'),
       ST_IsValid('LINESTRING(0 0, 0 0, 0 0)');

 st_isvalid | st_isvalid
------------+-----------
     t      |     f
```

디폴트에서 PostGIS은 지오메트리 입력에 이 유효성 체크를 적용하지 않습니다. 왜냐하면 유효성 체크는 복잡한 지오메트리에 의해(특히 다각형들) 많은 CPU 시간을 필요로 하기 때문입니다. 만약 본인의 데이터 소스를 신뢰하지 않으신다면 수동으로 제약조건(check constraint)를 첨가함으로써 본인은 테이블들을 체크하실 수 있습니다:

```
ALTER TABLE mytable
  ADD CONSTRAINT geometry_valid_check
       CHECK (ST_IsValid(the_geom));
```

유효 입력 지오메트리로 PostGIS 기능들을 호출 할 때 만약 GEOS Intersection() threw an error!" 혹은 "JTS Intersection() threw an error!"와 같은 이상한 에러 메세지들을 접하게 되신다면 이것이 사용하는 하나의 라이브러리 또는 PostGIS에서 에러를 찾으실 수 있을 것입니다. PostGIS 개발자들에게 연락을 취하셔야 합니다. PostGIS 기능이 유효 입력을 위한 인식 불가능한 지오메트리를 되돌릴 경우에도 마찬가지 입니다.

엄격하게 준수하는 OGC 지오메트리는 Z 또는 M 값들을 가질 수 없습니다. ST_IsValid() 기능은 고차원 지오메트리를 인식 불가능하다고 여기지 않을 것입니다. AddGeometryColumn() 의 발동은 제약 확인 지오메트리 차원들을 첨가하며 따라서 2를 규정하기에 충분합니다.

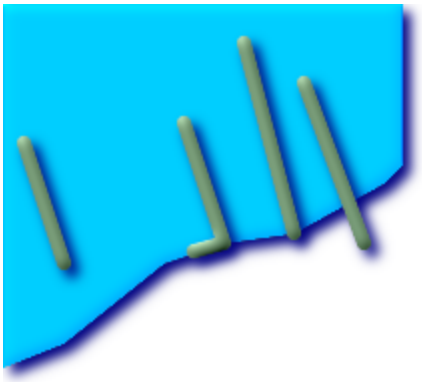
4.3.6. 차원적으로 확장된 9 교차 모델(DE-9IM)

전형적인 공간 predicates(ST_Contains, ST_Crosses, ST_Intersects, ST_Touches, ...)은 때로는 원하는 공간 필터를 적절하게 제공하는 것이 부족한 경우가 있습니다.

예를 들어 도로망을 나타내는 직선 데이터셋을 생각해보십시오. 한 점이 아니라 한 선 위에서 서로 교차하는 모든 도로 부분들(아마도 어떤 비즈니스 룰을 무효화하는)을 파악하는 일은 GIS 분석가의 몫일 수도 있습니다. 이 경우에 있어서 ST_Crosses 는 오직 한 점에서 교차할 때에만 정확하기 때문에 선 피쳐을 위한 필요한 공간 필터를 충분히 제공하지 않습니다.

원 투-스텝 솔루션은 (One two-step solution) 아마 공간적으로 교차하는(ST_Intersects) 도로망 짝들의 실제 교차지점을 수행할 것입니다. 그리고 난 후 교차지점의 ST_GeometryType 을 'LINESTRING' 와 비교합니다. (아마 [MULTI]POINTs, [MULTI]LINESTRINGs, 등등의 GEOMETRYCOLLECTIONs 을 되돌리는 케이스들을 처리할 것입니다.)

더 우아하고/더 빠른 솔루션은 정말 가치 있습니다.



두 번째 [이론적] 예시는 아마도 선 위 호수의 경계선과 오직 부두의 한 끝이 육지에 있는 곳을 교차시키는 모든 부두들과 선창들의 정확한 위치를 찾는 것을 노력하는 GIS 분석가 일 것입니다. 즉, 부두가 호수 내 있으나 완전히 안에 있지 않고 선 위 호수의 경계선과 부두의 종점들이 호수 내와 호수의 경계선 위에 있는 곳을 가로지르는 곳을 말합니다. 분석가는 아마도 수요가 많은 피쳐들을 고립시키기 위한 공간 술부들의 조합을 사용할 필요가 있을 것입니다:

- ST_Contains(lake, wharf) = TRUE
- ST_ContainsProperly(lake, wharf) = FALSE
- ST_GeometryType(ST_Intersection(wharf, lake)) = 'LINESTRING'
- ST_NumGeometries(ST_Multi(ST_Intersection(ST_Boundary(wharf), ST_Boundary(lake)))) = 1
  ... (needless to say, this could get quite complicated)

그러므로 Dimensionally Extended 9 Intersection Model, 또는 DE-9IM for short에 들어가십시오.

4.3.6.1. 이론

OpenGIS Simple Features Implementation Specification for SQL 에 따르면 "두 개의 지오메트리를 비교하는 기초 접근법은 내부들, 경계선들 그리고 두 개의 지오메트리의 외부들(exteriors) 사이의 교차점들의 쌍방식 테스트들을 만듭니다. 그리고 “교차점”매트리스의 엔트리 기반 두 개의 지오메트리 사이의 관계를 분류합니다.

경계선(Boundary)

지오메트리의 경계선은 다음 저차원의 지오메트리 집합입니다. 0 의 차원을 가진 `POINT` 들의 경계선은 공집합입니다. `LINESTRING` 의 경계선은 두 개의 종점들(endpoints)입니다. `POLYGON` 들의 경계선은 내부 그리고 외부 고리들을 형성하는 경계선입니다.

내부(Interior)

지오메트리의 내부는 경계선을 제거할 때 남는 지오메트리의 점들입니다. `POINT`s 경우 내부는 `POINT` 그 자체입니다. `LINESTRING` 내부는 종점들(endpoints) 사이의 실제 점들의 집합입니다. `POLYGON` 들의 내부는 다각형 안쪽 영역의 표면입니다.

외부(Exterior)

지오메트리의 외부는 유니버스(universe)이며 면적 표면이나 지오메트리의 경계면 혹은 외부에 있지 않습니다.

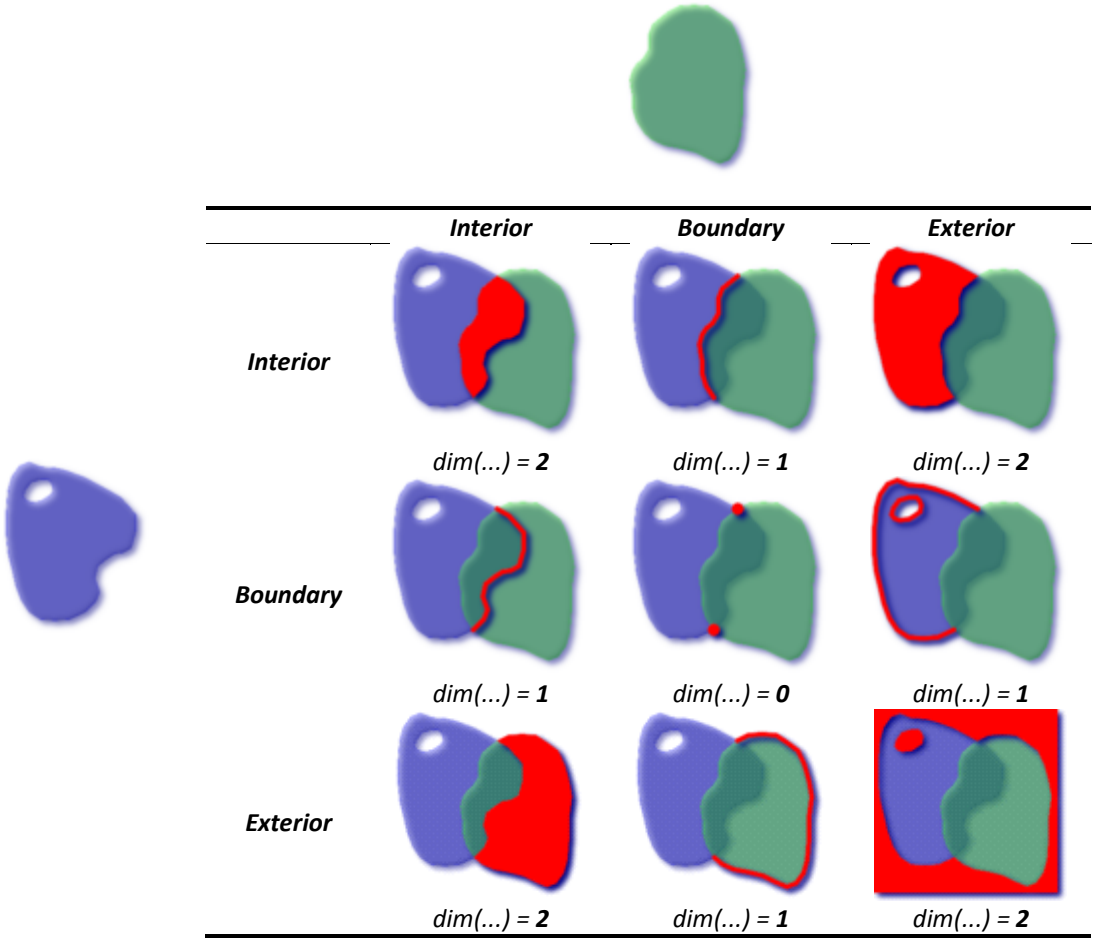
주어진 지오메트리 a, *I(a)*, *B(a)*, and *E(a)*가 외부인 곳, 경계선, 그리고 a의 외부, 매트릭스의 수학적 표현은 아래와 같습니다:

	Interior	**Boundary**	**Exterior**
Interior	*dim(I(a) ∩ I(b))*	*dim(I(a) ∩ B(b))*	*dim(I(a) ∩ E(b))*
Boundary	*dim(B(a) ∩ I(b))*	*dim(B(a) ∩ B(b))*	*dim(B(a) ∩ E(b))*
Exterior	*dim(E(a) ∩ I(b))*	*dim(E(a) ∩ B(b))*	*dim(E(a) ∩ E(b))*

Where *dim(a)* is이 ST_Dimension 에 의해 a의 차원으로 규정되지만 {0,1,2,T,F,*}의 도메인을 가지고 있는 곳

- 0 => point
- 1 => line
- 2 => area
- T => {0,1,2}
- F => empty set
- * => don't care

시각적으로 두 개의 겹치는 다각형 지오메트리들을 위해 다음과 같은 형태를 보입니다:



왼쪽에서 오른쪽 그리고 위에서 아래로 읽으십시오. 차원적 매트릭스는 '**212101212**'로 표현됩니다.

한 선 위 교차하는 두 선들의 첫 번째 예를 표본이 되는 relate 매트릭스는 다음과 같습니다: '**1*1***1****'

```
-- Identify road segments that cross on a line
SELECT a.id
FROM roads a, roads b
WHERE a.id != b.id
AND a.geom && b.geom
AND ST_Relate(a.geom, b.geom, '1*1***1**');
```

A 호수의 해안선 위 부분적으로 있는 부두들의 두 번째 예시를 표본이 되는 relate 매트릭스는 다음과 같습니다: '**102101FF2**'

```
-- Identify wharfs partly on a lake's shoreline
SELECT a.lake_id, b.wharf_id
FROM lakes a, wharfs b
WHERE a.geom && b.geom
AND ST_Relate(a.geom, b.geom, '102101FF2');
```

더 많은 정보와 읽을거리를 원하신다면 아래를 참조하십시오:

- OpenGIS Simple Features Implementation Specification for SQL (version 1.1, section 2.1.13.2)
- Dimensionally Extended Nine-Intersection Model (DE-9IM) by Christian Strobl
- GeoTools: Point Set Theory and the DE-9IM Matrix
- *Encyclopedia of GIS* By Hui Xiong

4.4. GIS 데이터 로딩

일단 공간 테이블을 생성하셨다면 데이터베이스에 GIS 데이터를 업로드 할 준비가 되신 것입니다. 형식화 된 SQL statement 또는 shape 파일 로더/덤퍼를 사용함으로써 데이터를 PostGIS/PostgreSQL 데이터베이스에 넣을 수 있는 방법에는 현재 2가지가 있습니다.

4.4.1. SQL 사용하기

만약 문자열 표현으로 본인 데이터를 전환하실 수 있다면 형식화된 SQL을 사용하는 것이 PostGIS에 데이터를 넣기 위한 가장 쉬운 방법일 것입니다. 오라클 그리고 다른 SQL 데이터베이스로 SQL 터미널 모니터(terminal monitor)에 SQL "INSERT" 명령문들의 큰 텍스트 파일을 송신함으로써 데이터는 대량 로딩될 수 있습니다.

데이터 업로드 파일(예를 들어 `roads.sql`)은 다음과 같습니다:

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

"psql" SQL 터미널 모니터를 사용함으로써 데이터파일은 쉽게 PostgreSQL으로 전송될 수 있습니다:

```
psql -d [database] -f roads.sql
```

4.4.2. 로더 사용하기

The `shp2pgsql` 데이터 로더는 ESRI Shape 파일들을 SQL로 전환하여 지오메트리 혹은 지오그래피 포맷의 PostGIS/PostgreSQL 데이터베이스에 삽입하기 적당하도록 만듭니다. 로더는 명령 행 플래그들(command line flags)에 의해 구별되는 몇 몇의 운용 모드들을 가지고 있습니다:

shp2pgsql 명령 행 로더에 더하여 명령 행 로더와 같이 대부분의 옵션들을 가진 그래픽 인터페이스 `shp2pgsql-gui`가 있으나 일회성이며 스크립 되지 않는 로딩 혹은 PostGIS에 대해 초보이실 때 사용하시기에 더 쉬울 것입니다. 이는 또한 PgAdminIII의 플러그인으로 환경 설정 될 수 있습니다.

(c|a|d|p) 아래의 것들은 상호 배타적인 옵션들 입니다:

-c 새로운 테이블을 생성하고 이것을 shape파일로부터 덧붙입니다. 이것은 디폴트 모드입니다.

-a shape파일로부터의 데이터를 데이터베이스 테이블로 덧붙입니다. 다량의 파일들을 로드하는 이 옵션을 사용하기 위해서는 해당 파일들은 반드시 같은 속성과 데이터 타입을 가지고 있어야 한다는 점에 주목하십시오.

-d shape 파일의 데이터로 새로운 테이블을 생성하기 이전에 데이터베이스 테이블을 중단 시킵니다.

-p 그 어떤 실제 데이터 추가 없이 오직 테이블 작성 SQL 코드만 생성합니다. 독립된 테이블 생성 및 데이터 로딩 스텝들이 전적으로 필요 시 사용될 수 있습니다.

-? 헬프 스크린을 보여줍니다.

-D 출력데이터를 위해서 PostgreSQL "덤프" 포맷을 사용합니다. 이것은 -a, -c 그리고 –d와 함께 결합될 수 있습니다. 디폴트 "삽입" SQL 포맷보다 훨씬 빨리 로드 됩니다. 매우 큰 데이터 집합들 경우 이것을 사용하십시오.

-s [<FROM_SRID%gt;:]<SRID> 해당 옵션은 명시된 SIRD을 가진 지오메트리 테이블을 생성하고 덧붙입니다. 해당 옵션은 선택적으로 입력shape파일이 타겟 SRID(target SRID)에 지오메트리들이 재투영되는 경우 주어진 FROM_SRID을 사용한다는 것을 명시합니다. FROM_SRID 는 –D와 함께 규정될 수 없습니다.

-k 해당 옵션은 식별자 케이스(행, 스키마 그리고 속성)를 보관합니다. shape파일의 속성들은 모두 대문자임에 주목하십시오.

-i 비록 DBF SIGNATURE이 이것을 보장하는 것처럼 보여도 모든 정수들을 표준 32-bit 정수들로 강제합니다. 또한 64-bit bigints 생성하지 않게 강제합니다.

-I 지오메트리 행 위 GiST 인덱스를 생성합니다.

-S 멀티 지오메트리 대신 심플 지오메트리를 생성합니다. 모든 지오메트리들이 실제로 심플할 경우에만 성공 가능합니다. (예. 싱글 쉘(single shell) 모델을 가진 MULTIPOLYGON 또는 단일 꼭지점을 가진 MULTIPOINT).

-t <dimensionality> 출력 지오메트리가 규정된 차원수를 가지도록 강요합니다. 차원수를 나타내기 위해 다음의 strings을 사용합니다: 2D, 3DZ, 3DM, 4D.

만약 입력이 가진 차원들이 규정된 것 보다 더 적을 경우, 출력은 해당 차원들을 0들로 채웁니다. 만약 입력이 가진 차원들이 규정된 것 보다 많은 경우에는, 필요 없는 차원들은 제거 됩니다.

-w WKB 대신 WKT 포맷울 출력해 냅니다. 정밀성 상실로 인해 좌표 표류를 시작할 수 있음에 주의하십시오.

-e 처리(transaction)사용 없이 스스로의 각 명령어를 실행합니다. 이는 에러를 생성하는 일부 나쁜 지오메트리들이 있을 시 좋은 데이터를 대부분 로딩하는 것을 가능케 합니다. "덤프" 포맷이 항상 처리(transaction)을 사용하기 때문에 이 옵션은 the -D flag 와 함께 사용될 수 없음을 주의하십시오.

-W <encoding> 입력 데이터의 코드화를 명시화 합니다(dbf파일). 사용될 시 모든 dbf의 속성들은 규정된 코드화(encoding, 인코딩)으로 부터 UTF8로 전환됩니다. 결과로 초래된 SQL 출력은 `UTF8` 커맨더를 하기 위해 `SET CLIENT_ENCODING`을 포함하며 이로 인해 백엔드는 UTF8로 부터 내부적으로 사용하기 위해 환경설정 된 인코딩 데이터베이스로 재전환 할 수 있게 됩니다.

-N <policy> NULL 지오메트리 처리 정책 (insert*, skip, abort)

-n -n은 오직 DBF파일만 임포트(IMPORT)합니다. 만약 본인은 데이터가 상응하는 shape파일이 없다면 자동적으로 이 모드로 전환한 뒤 dbf만 로딩 할 것 입니다. 그러므로 모든 shape파일 세트를 가지고 있을 때와 오직 속성 데이터만 필요하고 지오메트리는 필요치 않을 시에 이 플래그(flag) 세팅이 요구됩니다.

-G WGS84 long lat (SRID=4326)의 지오메트리 대신에 지오그래피 타입을 사용.

-T <tablespace> 새로운 테이블을 위한 테이블스페이스(tablespace)를 명시합니다. –X 파라미터가 사용되지 않는 한 인덱스들은 여전히 디폴트 테이블스페이스를 사용할 것 입니다. PostgreSQL 다큐멘테이션은 언제 커스텀 테이블스페이스를 사용하는지에 관한 좋은 설명을 제공합니다.

-X <tablespace> 새로운 테이블의 인덱스들을 위한 테이블스페이스를 명시합니다. 이것은 주요 키 인덱스에 적용됩니다. 또한 -I이 사용될 시 GIST에도 이것은 적용됩니다.

입력 파일을 생성하기 위해 로더를 사용하고 이것을 업로딩 하는 예시 세션(session)은 다음과 같은 형태처럼 나타납니다:

```
# shp2pgsql -c -D -s 4269 -i -I shaperoads.shp myschema.roadstable >
roads.sql
# psql -d roadsdb -f roads.sql
```

전환 및 업로드는 UPNIC 파이프(pipes)를 사용함으로써 한번에 수행될 수 있습니다:

```
# shp2pgsql shaperoads.shp myschema.roadstable | psql -d roadsdb
```

4.5. GIS 데이터 추출하기

SQL 혹은 Shape 파일 로더/덤퍼를 사용함으로써 데이터베이스로부터 데이터를 추출할 수 있습니다. SQL 섹션에서 공간 테이블 위 비교와 쿼리를 가능케 하는 일부 오퍼레이터들에 대한 논의가 있을 것입니다.

4.5.1. SQL 사용하기

데이터를 데이터베이스로부터 끌어오는 것에 관한 가장 쉬운 의미는 리턴되는 행들과 기록들의 숫자를 줄이기 위해 SQL 선택 권리를 사용하기 위함입니다. 그리고 결과로 초래된 행들을 parsable 텍스트 파일(parsable text file)로 덤프하기 위함입니다:

```
db=# SELECT road_id, ST_AsText(road_geom) AS geom, road_name FROM roads;

road_id | geom                                | road_name
--------+-------------------------------------+-----------
       1 | LINESTRING(191232 243118,191108 243242) | Jeff Rd
       2 | LINESTRING(189141 244158,189265 244817) | Geordie Rd
       3 | LINESTRING(192783 228138,192612 229814) | Paul St
       4 | LINESTRING(189412 252431,189631 259122) | Graeme Ave
       5 | LINESTRING(190131 224148,190871 228134) | Phil Tce
       6 | LINESTRING(198231 263418,198213 268322) | Dave Cres
       7 | LINESTRING(218421 284121,224123 241231) | Chris Way
(6 rows)
```

그러나 리턴되는 필드들의 숫자를 줄이기 위해 일종의 규제들이 필요할 경우들이 있습니다. 속성기반 규제들의 경우 똑같은 SQL 신택스(syntax) 비공간 테이블과 함께 정상적으로 사용합니다. 공간 규제들의 경우에는 아래의 오퍼레이트들이 이용가능하며 유효합니다:

&& 이 오퍼레이터는 하나의 지오메트리 바운딩 박스가 다른 바운딩 박스와 교차하는지에 관해 말해줍니다.

ST_OrderingEquals 이 오퍼레이터는 두 지오메트리들이 기하하적으로 동일한지를 테스트 합니다. 예를 들어 'POLYGON((0 0,1 1,1 0,0 0))' 이 'POLYGON((0 0,1 1,1 0,0 0))' (it is)와 같은지를 테스트 합니다.

= 이 오퍼레이트는 조금 더 나이브합니다. 이 오퍼레이트는 두 지오메트리들의 바운딩 박스들이 서로 같지만 테스트 합니다.

다음으로 이 해당 오퍼레이터들을 쿼리에서 사용하실 수 있습니다. SQL 커맨드 라인(command line) 위 박스들과 지오메트리들을 명시화 할 시 string 표현들을 "ST_GeomFromText()" 기능을 사용하여 반드시 분명하게 지오메트리들로 변화시켜야 합니다. 312 은 데이터를 일치시키는 허구의 공간 레퍼런스 시스템입니다. 그러므로 예를 들어:

```
SELECT road_id, road_name
  FROM roads
  WHERE ST_OrderingEquals(roads_geom , ST_GeomFromText('LINESTRING(191232
243118,191108 243242)',312) ) ;
```

위 쿼리는 지오메트리가 그 값과 동일했던 "ROADS_GEOM" 테이블로부터 단일 레코드를 리턴합니다.

"&&" 오퍼레이터 사용시 BOX3D를 비교 피쳐 혹은 지오메트리로써 명시할 수 있습니다. 하지만 지오메트리를 명시화 할 때 이것의 바운딩 박스는 비교를 위해 사용될 것입니다.

```
SELECT road_id, road_name
FROM roads
WHERE roads_geom && ST_GeomFromText('POLYGON((...))',312);
```

위 쿼리는 비교 목적을 위한 다각형의 바운딩 박스를 사용할 것입니다.

가장 일반적인 공간 쿼리는 아마도 데이터 브라우저들 및 웹 맵퍼들 같이 디스플레이를 위한 “맵 프레임(map frame) 가치를 움켜잡는 데이터 클라이언트 소프트웨어가 사용하는 “프레임기반(frame-based)” 쿼리일 것입니다. 쿼리 같은 프레임을 위한 "BOX3D" 오브젝트는 다음과 같은 형태를 보입니다:

```
SELECT ST_AsText(roads_geom) AS geom
FROM roads
WHERE
  roads_geom && ST_MakeEnvelope(191232, 243117,191232, 243119,312);
```

envelope의 투영을 명시화 하는 SRID 312의 용도에 주목하십시오.

4.5.2. 덤퍼 사용하기

pgsql2shp 테이블 덤퍼는 직접적으로 데이터베이스에 연결되며 테이블(쿼리에 정의될 수 있는)을 shape파일로 전환시킵니다. 베이직 신택스(basic syntax)는 다음과 같습니다:

```
pgsql2shp [<options>] <database> [<schema>.]<table>
```

```
pgsql2shp [<options>] <database> <query>
```

명령행 옵션들은 다음과 같습니다:

-f <filename> 특정 파일 이름에 출력을 기입합니다.

-h <host> 연결하기 위한 데이터베이스 호스트

-p <port> 데이터베이스 호스트에 연결되는 포트

-P <password> 데이터베이스에 접속할 때 사용하는 비밀번호

-u <user> 데이터베이스에 접속할 때 사용하는 사용자 이름

-g <geometry column> 다중 지오메트리 행들을 가진 테이블의 경우 shape파일을 기입할 때 지오메트리 행이 사용하는 옵션

-b binary 커서(cursor) 사용. 이 옵션은 운용을 더 빠르게 만들 것입니다. 그러나 테이블의 비기하학 속성이 문자를 위해 cast가 부족할 경우 이는 작동되지 않습니다.

-r 미처리 모드. Do not drop the `gid` 필드를 정지하거나 행 이름들을 벗어나지 않습니다.

-d 백워드 호환성을 위해: 구(pre-1.0.0) postgis 데이터베이스들(기본값은 이 경우에는 2차원 shape을 입력할 때입니다) 파일로부터 덤핑 할 경우 3 차원 shape 파일을 기록합니다. postgis-1.0.0+을 기점으로 차원들은 완전히 암호화 되었습니다.

-m filename 10개의 캐릭터 이름으로 식별자들을 재배치합니다. 파일의 컨텐츠는 단일의 흰 공간으로 분리된 두 상징들의 선들이며 뒤쳐지거나 선두적인 공간이 아닙니다: VERYLONGSYMBOL SHORTONE ANOTHERVERYLONGSYMBOL SHORTER 등.

4.6. 인덱스 구축하기

인덱스들은 큰 데이터 집합들을 위한 공간 데이터베이스 사용을 가능케 하는 것들입니다. 인덱싱이 없이 피쳐를 위한 그 어떤 검색도 데이터베이스 내 모든 레코드의 "순차 스캔(sequential scan)" 을 요구합니다. 인덱싱은 특정 레코드를 재빨리 가로지를 수 있는 검색 트리에 데이터를 체계화 시키면서 검색 속도를 빠르게 합니다. PostgreSQL은 인덱스들의 세가지 종류들을 기본으로 지원합니다: B-Tree indexes, R-Tree indexes, GiST indexes.

- B-Trees은 한 축을 따라 분류될 수 있는 데이터를 위해 사용됩니다; 예를 들어 숫자들, 편지들, 날짜들 입니다. GIS data은 하나의 축을 따라 합리적으로 분류되지 않습니다 (더 큰, (0,0) or (0,1) or (1,0)?) 그러므로 so B-Tree 인덱싱은 사용용도가 무의미합니다.

- R-Trees는 데이터를 직사각형들 그리고 부분 직사각형들(sub-rectangles)및 부분의 부분 직사각형들(sub- sub-rectangles), 등으로 분해합니다. R-Trees는 GIS 데이터를 인덱스 하기 위해 일부 공간 데이터베이스들로부터 사용됩니다. 그러나 PostgreSQL R-Tree 구현(implementation is)은 Gist 구현(implementation)만큼 견고하지 않습니다.

- GiST (일반화된 검색 트리, Generalized Search Trees) 인덱스들은 "한쪽에 있는 것들(things to one side)", "겹치는 것들(things which overlap)", "안쪽에 있는 것들(things which are inside)" 로 데이터를 분해하며 GIS 데이터 포함 데이터 타입들의 넓은 범위에서 사용될 수 있습니다. PostGIS은 인덱스 GIS 데이터 위에 구현된 R-Tree 인덱스를 사용합니다.

4.6.1. GiST Indexes

GiST는 "일반화된 서치 트리(Generalized Search Tree)"를 의미하며 인덱싱의 포괄적인 형태입니다. GIS 인덱싱에 더하여 GiST는 보통 B-Tree 인덱싱에 처리할 수 있지 않은 불규칙적인 데이터 구조들(정수 배열들, 스펙트럼 데이터, 등)의 모든 종류들 위 검색 속도를 높입니다.

GIS 데이터테이블이 몇 천 개의 열들을 넘었을 때 데이터 공간 검색들의 속도를 높이기 위해 인덱스 구축이 필요할 것입니다. (본인의 검색들이 속성 필드들 위에 보통 인덱스를 구축해야하는 경우인 속성들을 기반이 아니라면)

"지오메트리" 행 위 GiST 인덱스를 구축하기 위한 syntax는 다음과 같습니다:

```
CREATE INDEX [indexname] ON [tablename] USING GIST ( [geometryfield] );
```

위 syntax는 언제나 2D-index을 구축할 것입니다. 지오메트리 타입을 위한 PostGIS 2.0+에서 지원되는 n-차원 인덱스를 가지기 위해서는 다음의 syntax를 사용함으로써 생성하실 수 있습니다.

```
CREATE INDEX [indexname] ON [tablename] USING GIST ([geometryfield]
gist_geometry_ops_nd);
```

공간 인덱스를 구축하는 것은 계산적인 측면에서 많은 주의를 요하는 활동입니다: 백 만여개의 열들의 테이블들 위, 300MHz Solaris machine 위, GiST 인덱스를 구축하는 것은 약 1시간 정도 소요된다는 사실을 발견하였습니다. 인덱스를 구축한 뒤 query plans들을 최대한 활용할 수 있도록 사용 되는 테이블 통계를 모으도록 PostgreSQL을 강제하여야 합니다:

```
VACUUM ANALYZE [table_name] [(column_name)];
-- This is only needed for PostgreSQL 7.4 installations and below
SELECT UPDATE_GEOMETRY_STATS([table_name], [column_name]);
```

GiST 인덱스들은 PostgreSQL의 R-Tree 인덱스들 비해 우월한 두 가지 장점이 있습니다. 첫째로, GiST 인덱스들은 "null safe"이며 이는 null 값들을 포함하는 행들을 인덱스 할 수 있다는 걸 의미합니다. 둘째로, GiST 인덱스들은 PostgreSQL 8K 페이지 사이즈 보다 더 큰 GIS 오브젝트들을 다룰 때 중요한 "lossiness"컨셉을 지원합니다. Lossiness는 allows PostgreSQL이 인덱스 안 오브젝트의 오직 "중요한" 부분만 저장할 수 있도록 허용합니다--GIS 오브젝틀의 경우에는 단지 바운딩 박스뿐 입니다. 8k보다 큰 GIS 오브젝트들은 R-Tree 인덱스들이 구축되고 있는 프로세스 중 실패가 되는 원인이 됩니다.

4.6.2. 인덱스들 사용하기

원칙적으로 인덱스들은 데이터 엑세스 속도를 높여 줍니다: 일단 인덱스가 구축되면 질의 플래너(query planner)는 질의 계획(query plan)의 속도를 높이게 위해 언제 인덱스 정보를 사용해야 되는지 명료하게 결정합니다. 안타깝게도 PostgreSQL 쿼리 플래너는 GiST 인덱스의 용도를 최적화 하지 않습니다. 그러므로 검색들은 때때로 전체 테이블의 시퀀스 스캔을 디폴드 하는 대신 공간 인덱스를 사용해야만 합니다.

만약 본인의 공간 인덱스들이 사용되지 않다는 것을 발견(또는 본인의 속성 인덱스들)했을 때 몇 가지 조치를 취하실 수 있는 방법들이 있습니다:

- 첫째로 쿼리 플래너가 인덱스 사용에 관한 더 나은 결정들을 할 수 있도록 테이블 내 값들의 분포들 및 숫자에 관한 통계가 반드시 모이도록 하십시오. For PostgreSQL 7.4 설치들과 아래의 사항들은 **update_geometry_stats([table_name, column_name])**(compute distribution) 및 **VACUUM ANALYZE [table_name] [column_name]** (compute number of values)을 실행함으로서 이행될 수 있습니다. Starting with PostgreSQL 8.0 기점으로 **VACUUM ANALYZE 실행함으로써 두 가지 운용 모두를 이행합니다.** 어쨌든 정기적으로 본인의 데이터베이스들을 청소하셔야 합니다 – 많은 PostgreSQL DBAs 는 **VACUUM을 정기적 으로 off peak cron job으로써 실행합니다.**

- vacuuming이 작동하지 않을 시 **SET ENABLE_SEQSCAN=OFF** 명령어를 통해 플래너가 인덱스 정보를 사용하도록 강제할 수 있습니다. 이 명령어는 반드시 드물게 사용하셔야 하며 오직 공간적으로 인덱스된 질의들(spatially indexed queries) 위에서만 사용하셔야합니다: 대체로 플래너는 언제 보통 B-Tree 인덱스들을 사용해야 하는 지에 대해서 본인보다 더 잘알고 있습니다. 한번 본인의 쿼리를 실행했을 시 `ENABLE_SEQSCAN`을 다시 세팅하는 것을 고려하여만 합니다. 이를 통해 다른 쿼리들은 플래너를 평상치 처럼 플래너 활용할 것입니다.

버전 0.6 기준, `ENABLE_SEQSCAN` 와 함께 인덱스 사용하도록 플래너를 강제하는 것은 필요치 않습니다.

- 만약 플래너가 순차적 vs 인덱스 스캔들에 관해 틀렸다는 것을 발견하셨다면 postgresql.conf 내 random_page_cost 의 값을 줄이거나 SET random_page_cost=#을 사용하십시오. 파라미터를 위한 디폴트 값은 4입니다. 이것을 1 혹은 2로 세팅하도록 시도하십시오. 값을 감소하는 것은 플래너가 인덱스 스캔을 보다 사용하도록 만듭니다.

4.7. 복잡한 쿼리(질의)들

The 공간 데이터베이스 기능의 *raison d'etre*은 대개 데스크탑 GIS 기능을 요구하는 데이터베이스 내 쿼리들을 수행합니다. PostGIS를 효율적으로 사용하기 위해선 어떤 공간 기능들이 이용가능한지를 알아야하며 적절한 인덱스들이 좋은 퍼포먼스를 제공하기 위해 제자리에 있는지를 확인하여야 합니다. 이러한 예시들에 사용된312의 SRID은 순전히 데모(demonstration)를 위한 것입니다. 반드시 spatial_ref_sys table 에 나열되어 있는 READ SRID를 사용하시거나 본인 데이터의 투영도 일치하는 것을 사용하셔야 합니다. 만약 본인의 데이터가 명시된 공간 레퍼런스 시트템을 가지고 있지 않다면 왜 없는지에 대해 신중히 생각하셔야 합니다. 왜냐하면 가지고 있어야 되는 부분이기 때문입니다. 만약 그 이유가 분자의 내부구조 혹은 핵 대참사 사건이 발생할 경우 인류를 수송하기 위한 좋은 위치와 같이 정의된 지라적 공간 레퍼런스 시스템이 없는 어떤 것을 만들고 계신 때문이라면 간단히 SRID를 생략하시거나 하나를 만드시 뒤 그것을 `spatial_ref_sys` 테이블에 삽입하십시오.

4.7.1. 인덱스 장점 활용하기

쿼리 구축 시 && 같은 바운딩 박스 기반 오퍼레이터들이 GiST 공간 인덱스를 이용할 수 있다는 점을 기억하는 것은 중요합니다. Functions such as ST_Distance()와 같은 기능들은 자신들의 오퍼레이션을 최적화 하기 위해 인덱스를 사용할 수 없습니다. 예를 들어 큰 테이블 위에서 다음의 쿼리는 상당히 느릴 것입니다:

```
SELECT the_geom
FROM geom_table
WHERE ST_Distance(the_geom, ST_GeomFromText('POINT(100000 200000)', 312))
< 100
```

이 쿼리는 점의 100 단 위(100 units of the point (100000, 200000))에 있는 geom_table 의 모든 지오메트리들을 선택합니다. 규정된 점과 테이블 내 각 점들 사이의 거리를 계산하기 때문에 이는 느리게 수행될 것입니다. 테이블 내 각 열을 위한 ST_Distance() 계산이 그 예입니다. 필요한 거리 계산들의 숫자를 줄이기 위해 &&오퍼레이트 사용을 통해 이를 방지하실 수 있습니다:

```
SELECT the_geom
FROM geom_table
WHERE   ST_DWithin(the_geom,     ST_MakeEnvelope(90900,   190900,   100100,
200100,312), 100)
```

이 쿼리는 똑같은 지오메트리들을 선택합니다. 하지만 보다 더 효율적인 방법으로 이를 선택합니다. the_geom 위 GiST 인데스가 있다고 가정한다며 쿼리 플래너는 ST_distance() 기능의 결과 계산에 앞서 열들의 숫자을 줄이기 위해 인덱스를 사용할 수 있다는 점을 인지 할 것입니다. && 오퍼레이션에 사용되는 ST_MakeEnvelope 지오메트리는 기점 중앙에 위치한 200 평방 단위 사각 박스이라는 점에 주의하십시오 – 이것은 "질의 박사(query box)"입니다. && operator는 “쿼리 박스”를 겹치는 바운딩 박스를 가진 지오메트리들에게만 항목화 되는 결과들을 재빠르게 줄이는 인덱스를 사용합니다. 전체 지오메트리 테이블의 내용보다 쿼리 박스가 훨씬 작다고 가정한다면 이는 수행되어야 할 거리 계산들의 숫자를 기하급수적으로 줄일 것입니다.

행동에서의 변화(Change in Behavior)

PostGIS 1.3.0 기준 대부분의 지오메트리 관련 기능들(ST_Disjoint and ST_Relate 의 주목할 만한 예외들과 함께)은 내포된 바운딩 박스 중첩 오퍼레이터들을 포함합니다.

4.7.2. 공간 SQL 의 예시

이 섹션의 이 예시들은 두 개의 테이블, 직선 도로들의 테이블 그리고 다각형의 지방자치제 경계선들을 활용할 것입니다. bc_roads 테이블을 위한 테이블 정의들은 아래와 같습니다:

```
Column       | Type               | Description
-------------+--------------------+-------------------
gid          | integer            | Unique ID
name         | character varying  | Road Name
the_geom     | geometry           | Location Geometry (Linestring)
```

bc_municipality 테이블을 위한 테이블 정의는 아래와 같습니다:

```
Column       | Type               | Description
-------------+--------------------+-------------------
gid          | integer            | Unique ID
code         | integer            | Unique ID
name         | character varying  | City / Town Name
the_geom     | geometry           | Location Geometry (Polygon)
```

1. 모든 도로들의 총 길이는 km 로 얼마입니까?

매우 간단한 SQL 조각으로 이 질문에 대한 답을 할 수 있습니다:

```
SELECT sum(ST_Length(the_geom))/1000 AS km_roads FROM bc_roads;

km_roads
------------------
70842.1243039643
(1 row)
```

2. 프린스 조지 시는 헥타르로 얼마나 큽니까?

이 쿼리는 속성 조건(지방자치제 이름 위)와 공간 계산(지역의)을 결합시킵니다:

```
SELECT
  ST_Area(the_geom)/10000 AS hectares
FROM bc_municipality
WHERE name = 'PRINCE GEORGE';

hectares
------------------
32657.9103824927
 (1 row)
```

3. 주에서 가장 큰 지방자치구역는 영역별 기준 어디 입니까?

이 쿼리는 공간 측정을 쿼리 조건 내로 가져옵니다. 이 문제에 접근할 수 있는 몇 가지 방법들이 있습니다만 가장 효율적인 방법은 아래와 같습니다:

```
SELECT
  name,
  ST_Area(the_geom)/10000 AS hectares
FROM
  bc_municipality
ORDER BY hectares DESC
LIMIT 1;

name          | hectares
--------------+-----------------
TUMBLER RIDGE  | 155020.02556131
(1 row)
```

이 쿼리에 답하기 위해서는 반드시 모든 다각형의 영역을 계산해야만 한다는 점에 주의하십시오. 만약 이 활동을 많이 하게 된다면 퍼포먼스를 위해 독립적으로 인덱스하는 테이블에 지역 행을 삽입하는 것이 상식에 맞는 것일 것입니다. By ordering the results i 결과들을 내림 정렬하고 PostSQL "LIMIT' 명령어를 사용함으로써 max()같은 총합 기능을 사용하지 않고 가장 큰 값을 제거할 수 있습니다.

4. 각 지방자치구역 이내 완전히 포함되는 도로들의 길이는 얼마 입니까?

이것은 "공간 결합(spatial join)"의 한 예입니다. 왜냐하면 두 개의 테이블들로부터 데이터를 함께 가져오지만(결합을 통해) 보통 관계적 접근보다 결합 조건이 공통 키(common key)에 합류하면서 공간 상호 작용 조건 ("contained")을 사용합니다:

```
SELECT
  m.name,
  sum(ST_Length(r.the_geom))/1000 as roads_km
FROM
  bc_roads AS r,
  bc_municipality AS m
WHERE
  ST_Contains(m.the_geom,r.the_geom)
GROUP BY m.name
ORDER BY roads_km;

name                      | roads_km
--------------------------+------------------
SURREY                    | 1539.47553551242
VANCOUVER                 | 1450.33093486576
LANGLEY DISTRICT          | 833.793392535662
BURNABY                   | 773.769091404338
PRINCE GEORGE             | 694.37554369147
...
```

이 쿼리는 시간이 조금 걸립니다. 왜냐하면 테이블 내 모든 도로는 최종 결과(특정 예시 테이블을 위한 약 250K 도로들)로 요약되기 때문입니다. 더 작은 오버레이들(overlays)을의 경우 반응은 더 빠를 수도 있습니다.

5. 프린스 조지 시 이내 모든 도로들을 가진 새로운 테이블을 생성하십시오

이것은 "오버레이(overlay)"의 한 예입니다. 오버레이는 두 개의 테이블을 포함하며 공간적으로 오려지거나 잘라진 형들로 구성된 새로운 테이블을 출력합니다. 위에서 보여진 "공간 결합(spatial join)"과는 다르게 이 쿼리는 실제로 새로운 지오메트리들을 생성합니다. 오버레이는 터보차저가 달린 공간 결합과 같습니다. 그리고 오버레이는 정확한 분석 작업에 유용합니다:

```
CREATE TABLE pg_roads as
SELECT
  ST_Intersection(r.the_geom, m.the_geom) AS intersection_geom,
  ST_Length(r.the_geom) AS rd_orig_length,
```

```
  r.*
FROM
  bc_roads AS r,
  bc_municipality AS m
WHERE  m.name = 'PRINCE GEORGE' AND ST_Intersects(r.the_geom, m.the_geom);
```

6. 빅토리아의 "더글라스 거리"의 킬로미터 기준 길이는 무엇입니까?

```
SELECT
  sum(ST_Length(r.the_geom))/1000 AS kilometers
FROM
  bc_roads r,
  bc_municipality m
WHERE  r.name = 'Douglas St' AND m.name = 'VICTORIA'
       AND ST_Contains(m.the_geom, r.the_geom) ;

kilometers
------------------
4.89151904172838
(1 row)
```

7. 하나의 구멍을 가진 가장 큰 지방자치구역 다각형은 무엇입니까?

```
SELECT gid, name, ST_Area(the_geom) AS area
FROM bc_municipality
WHERE ST_NRings(the_geom) > 1
ORDER BY area DESC LIMIT 1;

gid  | name        | area
-----+-------------+------------------
12   | SPALLUMCHEEN | 257374619.430216
(1 row)
```

Chapter 5.래스터 데이터 관리, 쿼리, 응용프로그램

5.1. 래스터 생성 및 로딩

대부분의 경우, 패키지 된 래스터 로더(raster2pgsql)를 이용하여 이미 존재하는 래스터를 로딩하고 PostGIS 래스터들을 생성할 수 있습니다.

5.1.1. 래스터를 로딩하기 위한 raster2pgsql 사용하기

raster2pgsql 은 GDAL 에서 지원하는 래스터 포맷들을 PostGIS 래스터 테이블에 로딩하기 위한 sql 을 생성하는 실행 파일입니다. raster2pgsql 은 개별 래스터 파일 및 폴더에 존재하는 여러 개의 래스터 파일들을 로딩할 수 있을 뿐만 아니라 래스터들의 오버뷰를 생성할 수 있습니다.

raster2pgsql 은 PostGIS 의 일부로 컴파일 되기 때문에 (별도로 GDAL 라이브러리를 컴파일 하지 않는 한) 지원되는 래스터 타입들은 설치 된 GDAL 종속 라이브러리에서 컴파일 된 것과 동일합니다. raster2pgsql 의 -G flag 를 사용하면 지원되는 래스터 타입들의 리스트를 확인할 수 있습니다. 만약 두 가지 모두를 위해 같은 GDAL 라이브러리를 사용하고자 한다면 PostGIS 의 ST_GDALDrivers 문서에서 제공되는 것과 동일해야 합니다.

이 도구의 구 버전은 python 스크립트였으며, rsater2pgsql 실행 파일이 python 스크립트를 대체했습니다. 만약 Python 스크립트가 필요하다면 python 의 예시들은 GDAL PostGIS Raster Driver Usage 에서 찾을 수 있습니다. PostGIS 의 향후 버전에서는 raster2pgsql python 스크립트가 더 이상 지원되지 않을 수 있음을 주의하십시오.

특정 요소(레벨)의 래스터 오버뷰를 생성하면 생성된 오버뷰들의 same_alignment 값은 Ture 가 되지만, same_alignment 값이 False 가 되는 경우가 있습니다. 이러한 경우에 대한 내용은 http://trac.osgeo.org/postgis/ticket/1764 에서 확인할 수 있습니다.

EXAMPLE USAGE:

```
raster2pgsql raster_options_go_here raster_file someschema.sometable > out.sql
```

-? 도움말을 표시합니다. 아무런 옵션이 입력되지 않는 경우에도 도움말을 표시합니다.

-G 지원되는 래스터 포맷의 리스트를 표시합니다.

(c|a|d|p) 공통으로 적용되는 전용 옵션 :

-c 새로운 테이블을 생성하고 생성된 테이블에 래스터(들)를 추가합니다. (디폴트 모드)

-a 이미 존재하는 테이블에 래스터(들)를 추가합니다.

-d 이미 존재하는 테이블을 삭제하고, 새로운 테이블을 생성하여 래스터(들)를 추가합니다.

-p 준비 모드, 래스터(들)를 추가하지 않고 테이블만 생성합니다.

Raster processing : 래스터 카탈로그에 제약사항을 등록

-C srid, pixelsize 등 래스터의 제약사항을 `raster_columns` 뷰에 등록합니다.

-x 래스터의 공간적 최대범위(Max Extent) 설정을 해제합니다. (`-C` flag 가 사용된 경우에만 적용)

-r 일반 차단(regular blocking) 제약 조건을 설정 합니다. (`-C` flag 가 사용된 경우에만 적용)

Raster processing : 입력 래스터 데이터셋을 조작하기 위해 사용되는 선택적 파라미터

-s <SRID> 래스터의 SIRD 를 설정합니다.

-b BAND 래스터로 부터 추출하기 위한 Band 인덱스(1-based)를 설정합니다. 만약 두 개 이상의 밴드를 추출하기 위해서는 인덱스를 콤마(,)로 구분하십시오. 명시되어 있지 않을 경우 래스터의 모든 밴드가 추출됩니다.

-t TILE_SIZE 래스터를 테이블의 Row 당 하나의 삽입하기 위해 타일로 잘라 넣습니다. `TILE_SIZE` 는 WIDTH(넓이) x HEIGHT(높이)로 표현됩니다.

-R, --register 파일 시스템(out-db) 래스터로 래스터를 등록합니다. 데이터베이스에는 오직 래스터의 메타데이터 정보와 경로가 저장됩니다. (래스터 픽셀의 값이 저장되지 않음)

-l OVERVIEW_FACTOR 래스터의 오버뷰를 생성합니다. 하나 이상의 factor(레벨) 정보는 콤마(,)로 구분됩니다. 오버뷰 테이블의 명칭은 `o_overview factor_table` 의 패턴을 따르며, 패턴에서 `overview factor` 는 숫자로 된 factor(레벨)를 대신하는 텍스트 이며, `table` 은 원본 래스터의 테이블 명칭으로 변경됩니다. 생성된 오버뷰 정보는 데이터베이스에 저장되며 `-R` 옵션에 영향을 받지 않습니다. 생성된 sql 파일에는 기본 데이블 및 오버뷰 테이블에 대한 내용이 모두 포함되어 있습니다.

-N NODATA NODATA 값을 설정합니다.

데이터 오브젝트를 처리하기 위해 사용되는 선택적 파라미터

-q PostgreSQL 의 ID(Identifiers)를 따옴표로 두릅니다.

-f COLUMN 래스터 COLUMN 의 이름을 명시합니다. 기본은 'rast' 입니다.

-F 파일 명칭이 저장될 COLUMN 을 추가합니다.

-I 래스터 COLUMN 에 GIST 인덱스를 생성합니다.

-M 래스터 테이블에 대해 VACUUM ANALYZE 를 수행합니다.

-T tablespace 새로운 테이블이 사용 할 테이블스페이스를 지정합니다. 인덱스들(Primary Key 포함)은 `-X` flag 가 사용되지 않으면 기본 테이블스페이스를 사용합니다.

-X tablespace 새로운 테이블의 인덱스가 사용 할 테이블스페이스를 지정합니다. `-I` flag 가 사용될 때 기본키 (Primary Key) 및 공간 인덱스(Spatial Index)에 적용됩니다.

-Y 삽입(Insert) statement 대신 복사(Copy) statement 를 사용합니다.

-e 트랜잭션(transaction)을 사용하지 않고 각 명령문을 개별적으로 실행합니다.

-E ENDIAN 생성된 래스터의 binary 가 메모리상에 저장되는 순서(endianness)를 설정합니다. 0 은 XDR, 1 은 NDR 을 나타내며 NDR 을 기본으로 사용합니다. 현재 버전에서는 NDR 만 지원합니다.

※ XDR : 높은 우선순위 바이트(Most Significant Byte, MSB)를 메모리의 낮은 주소부터 먼저 저장

※ NDR : 낮은 우선순위 바이트(Least Significant Bytes, LSB)를 메모리의 낮은 주소부터 먼저 저장

-V version 출력 포맷의 버전을 명시합니다. 디폴트는 0 이며, 현재 버전에서는 0 만 지원합니다.

`raster2pgsql` 로더를 사용하여 래스터를 100x100 크기의 타일로 나누고 데이터베이스에 업로드하는 예시는 다음과 같습니다.

데이터베이스의 Schema 명칭을 생략할 수 있습니다.
예) 명령문에서 `public.demelevation` 대신 Schema 명칭을 생략하고 `demelevation` 을 사용한 경우 래스터 테이블은 Database 또는 User 의 기본 스키마 명칭으로 생성됩니다.

```
raster2pgsql -s 4236 -I -C -M *.tif -F -t 100x100 public.demelevation >
elev.sql
psql -d gisdb -f elev.sql
```

변환과 업로드는 UNIX pipes(|)를 사용하여 모두 한번에 수행할 수 있습니다:

```
raster2pgsql -s 4236 -I -C -M *.tif -F -t 100x100 public.demelevation |
psql -d gisdb
```

다음은 메사추세츠 주의 항공사진 래스터 타일(*.jpg)들을 `aerials` 스키마에 `boston` 이라는 테이블 명칭으로 로드하고 풀 뷰(Full View), 2, 4 Level 의 오버뷰를 생성하는 예시입니다. 데이터의 입력모드는 복사모드(Copy Statement : 중간 파일 없이 DB 에 바로 입력)를 사용하고, 기다릴 필요 없이 바로 테이블을 확인하기 위해 `-e` flag 를 사용하여 모든 트랜잭션을 사용하지 않습니다. 타일의 크기는

128x128 픽셀크기로 설정합니다. 마지막으로 -F flag 를 사용하여 filename 이라는 필드에 나누어진 타일의 파일명을 함께 저장합니다.

```
raster2pgsql -I -C -e -Y -F -s 26986 -t 128x128 -l 2,4
bostonaerials2008/*.jpg aerials.boston | psql -U postgres -d gisdb -h
localhost -p 5432
--get a list of raster types supported:
raster2pgsql -G
```

다음과 같이 -G 명령어는 GDAL 에서 지원하는 래스터 포맷의 리스트를 출력합니다

```
Available GDAL raster formats:
  Virtual Raster
  GeoTIFF
  National Imagery Transmission Format
  Raster Product Format TOC format
  ECRG TOC format
  Erdas Imagine Images (.img)
  CEOS SAR Image
  CEOS Image
  JAXA PALSAR Product Reader (Level 1.1/1.5)
  Ground-based SAR Applications Testbed File Format (.gff)
  ELAS
  Arc/Info Binary Grid
  Arc/Info ASCII Grid
  GRASS ASCII Grid
  SDTS Raster
  DTED Elevation Raster
  Portable Network Graphics
  JPEG JFIF
  In Memory Raster
  Japanese DEM (.mem)
  Graphics Interchange Format (.gif)
  Graphics Interchange Format (.gif)
  Envisat Image Format
  Maptech BSB Nautical Charts
  X11 PixMap Format
  MS Windows Device Independent Bitmap
  SPOT DIMAP
  AirSAR Polarimetric Image
  RadarSat 2 XML Product
  PCIDSK Database File
  PCRaster Raster File
  ILWIS Raster Map
  SGI Image File Format 1.0
  SRTMHGT File Format
  Leveller heightfield
  Terragen heightfield
  USGS Astrogeology ISIS cube (Version 3)
  USGS Astrogeology ISIS cube (Version 2)
  NASA Planetary Data System
  EarthWatch .TIL
  ERMapper .ers Labelled
  NOAA Polar Orbiter Level 1b Data Set
  FIT Image
```

```
GRIdded Binary (.grb)
Raster Matrix Format
EUMETSAT Archive native (.nat)
Idrisi Raster A.1
Intergraph Raster
Golden Software ASCII Grid (.grd)
Golden Software Binary Grid (.grd)
Golden Software 7 Binary Grid (.grd)
COSAR Annotated Binary Matrix (TerraSAR-X)
TerraSAR-X Product
DRDC COASP SAR Processor Raster
R Object Data Store
Portable Pixmap Format (netpbm)
USGS DOQ (Old Style)
USGS DOQ (New Style)
ENVI .hdr Labelled
ESRI .hdr Labelled
Generic Binary (.hdr Labelled)
PCI .aux Labelled
Vexcel MFF Raster
Vexcel MFF2 (HKV) Raster
Fuji BAS Scanner Image
GSC Geogrid
EOSAT FAST Format
VTP .bt (Binary Terrain) 1.3 Format
Erdas .LAN/.GIS
Convair PolGASP
Image Data and Analysis
NLAPS Data Format
Erdas Imagine Raw
DIPEx
FARSITE v.4 Landscape File (.lcp)
NOAA Vertical Datum .GTX
NADCON .los/.las Datum Grid Shift
NTv2 Datum Grid Shift
ACE2
Snow Data Assimilation System
Swedish Grid RIK (.rik)
USGS Optional ASCII DEM (and CDED)
GeoSoft Grid Exchange Format
Northwood Numeric Grid Format .grd/.tab
Northwood Classified Grid Format .grc/.tab
ARC Digitized Raster Graphics
Standard Raster Product (ASRP/USRP)
Magellan topo (.blx)
SAGA GIS Binary Grid (.sdat)
Kml Super Overlay
ASCII Gridded XYZ
HF2/HFZ heightfield raster
OziExplorer Image File
USGS LULC Composite Theme Grid
Arc/Info Export E00 GRID
ZMap Plus Grid
NOAA NGS Geoid Height Grids
```

5.1.2. PostGIS 래스터 기능들을 사용하여 래스터 생성하기

많은 경우 데이터베이스에 직접 래스터 및 래스터 테이블을 생성하기를 원할 때가 있습니다. PostGIS 에서는 이러한 경우를 위해 많은 기능들을 제공합니다. 다음과 같은 단계를 통해 래스터 및 래스터 테이블을 직접 생성할 수 있습니다.

1. 아래와 같이 래스터 레코드를 저장할 수 있는 래스터 Column 을 가지는 테이블을 생성합니다.

```
CREATE TABLE myrasters(rid serial primary key, rast raster);
```

2. 당신이 원하는 목표를 달성할 수 있는 많은 기능들이 있습니다. 만약 생성하고자 하는 래스터가 다른 래스터로부터 파생되는 래스터가 아닌 경우 ST_AddBand 와 ST_MakeEmptyRaster 를 통해 새로운 래스터를 생성할 수 있습니다.
   또한 지오메트리를 통해 래스터를 생성할 수 있습니다. ST_Union 또는 ST_MapAlgebraFct 등과 같은 Map Algebra(지도대수) 기능과 함께 ST_AsRaster 기능을 활용하여 분석된 결과 지오메트리를 래스터로 생성할 수 있습니다.
   이미 존재하는 래스터 테이블들로부터 새로운 래스터 테이블을 생성하는 방법은 더 많은 기능을 제공합니다. 예를 들어 ST_Transform 기능을 사용하여 이미 존재하는 테이블로부터 다른 좌표체계로 투영된 새로운 래스터 테이블을 생성할 수 있습니다.

3. 일단 래스터 테이블을 생성한 후 래스터 Column 에 공간인덱스를 생성하기를 원하는 경우에는 아래와 같이 공간인덱스를 생성할 수 있습니다.

```
CREATE INDEX myrasters_rast_st_convexhull_idx ON myrasters USING
gist( ST_ConvexHull(rast) );
```

   대부분의 래스터 오퍼레이터들이 래스터의 공간인덱스를 생성할 때 기준이 되는 영역을 Envelope 기반에서 Convex-hull 기반으로 변경된 이후부터 ST_ConvexHull 을 사용합니다.

PostGIS 2.0 이전 버전 래스터는 Convex-hull 보다는 Envelope 에 기반을 두었습니다. Spatial Index(공간인덱스)가 올바르게 적용되기 위해서는 기존 Envelope 기반의 인덱스를 Convex-hull 기반으로 대체해야 합니다.

4. AddRasterConstraints 를 사용하여 래스터의 제약사항들을 적용합니다.

5.2. 래스터 카탈로그

PostGIS 패키지에는 두 개의 래스터 카탈로그 뷰(View)가 있습니다. 두 개의 뷰(View) 모두 래스터 테이블의 제약들을 포함한 정보를 활용합니다. 결과적으로 래스터에 대한 제약 조건들이 활용되기 때문에 카탈로그 뷰(View)의 내용은 항상 래스터 데이터와 일치합니다.

1. `raster_columns` 데이터베이스 내의 모든 래스터 테이블들의 Column 정보를 가지고 있습니다.

2. `raster_overviews` 데이터베이스 내의 모든 래스터 테이블들의 Column 에 대한 자세한 오버뷰 정보를 포함합니다. 래스터 데이터를 로딩할 때, `-l` flag 를 사용하여 오버뷰를 생성한 경우 이 테이블에 생성된 오버뷰의 정보가 생성됩니다.

5.2.1. 래스터 Column 카탈로그 (Raster Columns Catalog)

`raster_columns` 는 데이터베이스의 모든 래스터 테이블의 Column(래스터 타입의 Column) 들의 명칭을 저장하는 카탈로그 입니다. 이 뷰는 래스터 테이블의 제약정보를 활용하기 때문에 하나의 래스터를 다른 데이터베이스에서 백업하여 다시 저장했다 하더라도 항상 정보들이 일관성을 유지합니다. 아래에 설명하는 Column 들이 `raster_columns` 카탈로그 내에 존재합니다.

만약 래스터 테이블을 로더(loader)를 통해 생성하지 않았거나, 로딩 도중 `-C` flag 를 명시하지 않았다면 AddRasterConstraints 를 활용하여 제약들을 생성할 수 있습니다. 이런 과정을 통해 `raster_columns` 카탈로그에 래스터 타일에 대한 일반적인 정보를 등록할 수 있습니다.

- `r_table_catalog` 래스터 테이블이 속한 데이터베이스. 항상 현재 데이터베이스를 읽습니다.
- `r_table_schema` 래스터 테이블이 속한 데이터베이스의 스키마
- `r_table_name` 래스터 테이블의 명칭
- `r_raster_column` `r_table_name` 으로 저장된 래스터 테이블의 래스터 타입 Column 명칭. PostGIS 는 하나의 테이블이 여러 개의 래스터 Column 을 가지는 것이 가능함으로 여러 개의 래스터 Column 이 저장될 수 있습니다.
- `srid` 래스터의 공간 레퍼런스 식별자. 4.3.1 단락 참고.
- `scale_x` 기하학적 공간 좌표와 픽셀 사이의 스케일을 명시. 래스터 Column 에 있는 모든 타일이 같은 `scale_x` 를 가지고 `scale_x` 제약 조건이 적용되는 경우에만 사용할 수 있습니다. 자세한 내용은 ST_ScaleX 를 참조하십시오.
- `scale_y` 기하학적 공간 좌표와 픽셀 사이의 스케일을 명시. 래스터 Column 에 있는 모든 타일이 같은 `scale_y` 를 가지고 `scale_y` 제약 조건이 적용되는 경우에만 사용할 수 있습니다. 자세한 내용은 ST_ScaleY 를 참조하십시오.

- `blocksize_x` 각 래스터 타일의 폭 (가로 픽셀의 수). 자세한 내용은 ST_Width 을 참조하십시오.
- `blocksize_y` 각 래스터 타일의 폭 (세로 픽셀 수). 자세한 내용은 ST_Height 을 참조하십시오.
- `same_alignment` 모든 래스터 타일의 기울기, 픽셀 사이즈, srid 등이 동일한 경우 True 입니다. 자세한 내용은 ST_SameAlignment 를 참조하십시오.
- `regular_blocking` 동일한 기울기, 픽셀 사이즈, srid 등을 가지는 타일을 겹치지 않도록 나타내기 위해 테이블에 설정된 True/False 제약사항 입니다. 이 제약사항은 별도의 검증과정이 없이 설정된 데로 받아들여지기 때문에 참고하는 측면으로 사용되어야 합니다. 향후 이 제약사항이 제대로 동작할 수 있도록 할 계획이며 이를 통해 해당 정보가 정확함을 보장하게 할 것입니다.
- `num_bands` 래스터 데이터세트 타일의 밴드의 수. ST_NumBands 에 의해 제공되는 것과 동일한 정보입니다.
- `pixel_types` 각 밴드의 픽셀 형식을 정의하는 배열. 이 배열의 수는 래스터 밴드의 수와 동일해야 합니다. `pixel_types` 은 ST_BandPixelType 에 정의된 타입들 중 하나 입니다.
- `nodata_values` 각 밴드의 `nodata_value` 를 나타내는 배정 밀도(double precision) 숫자의 배열. 이 배열의 수는 래스터 밴드의 수와 동일해야 합니다. `nodata_value` 로 정의된 숫자들은 대부분의 오퍼레이션에서 무시 되어야 하는 각 밴드의 픽셀 값을 정의합니다. ST_BandNoDataValue 에서 제공하는 정보와 유사합니다.
- `extent` 래스터 데이터 집합에 있는 모든 래스터 행의 공간적 범위. 만약 해당 래스터에 더 많은 데이터를 로드 하여 공간적 범위가 변경되어야 하는 경우에는 로드 하기 전에 DropRasterConstraints 기능을 실행하고, 데이터를 로드 한 후 AddRasterConstraints 기능을 실행하여 제약조건을 다시 적용 할 수 있습니다.

5.2.2. 래스터 오버뷰

래스터 오버뷰는 래스터 테이블 Column 에 대한 `raster_overviews` 카탈로그 정보 및 이와 관련된 추가적인 정보가 필요할 때 알아두면 편리합니다. 오버뷰 테이블들은 `raster_columns` 및 `raster_overviews` 양쪽 모두에 카탈로그화 되어 있습니다. 왜냐하면, 오버뷰 테이블은 테이블 자체로 래스터를 표현하는 목적을 가지는 동시에 고해상도 테이블의 저해상도 표현과 같은 추가적인 특수 목적으로 제공되기 때문입니다. 오버뷰 테이블은 래스터를 로딩 할 때 `-l` flag 를 사용하여 생성할 수 있습니다.

오버뷰 테이블은 다른 래스터 테이블과 같은 제약 조건뿐만 아니라 구체적인 제약들에 관한 추가 정보까지 포함합니다.

raster_overviews 의 정보는 raster_columns 에 있는 정보를 복제하지 않습니다. 만약 raster_columns 에 존재하는 오버뷰 테이블에 대한 정보가 필요한 경우에는 rater_overview 테이블과 raster_columns 테이블을 조인(Join)하여 모든 정보를 확인할 수 있습니다.

오버뷰가 존재하는 두 가지 주된 이유 :

1. 고해상도의 래스터를 낮은 해상도로 표현하여 줌 아웃 맵핑을 빠르게 합니다.
2. 낮은 해상도는 높은 해상도에 비해 레코드의 수가 적고, 많은 지역을 커버할 수 있음으로 계산의 속도가 빠릅니다. 계산의 결과는 높은 해상도의 테이블만큼 정확하지 않지만 경험과 상식에 입각한 추정이 가능하기 때문에 충분히 활용될 수 있습니다.

raster_overviews 카탈로그는 다음의 Column 정보를 포함합니다.

- o_table_catalog 오버뷰 테이블이 속한 데이터베이스. 항상 현재 데이터베이스를 읽습니다.
- o_table_schema 오버뷰 테이블이 속한 데이터베이스의 스키마.
- o_table_name 래스터 오버뷰 테이블의 명칭.
- o_raster_column 오버뷰 테이블의 래스터 Column 명칭.
- r_table_catalog 오버뷰 서비스를 하는 래스터 테이블이 속한 데이터베이스. 항상 현재 데이터베이스를 읽습니다.
- r_table_schema 오버뷰 서비스를 하는 래스터 테이블이 속한 데이터베이스의 스키마.
- r_table_name 오버뷰 서비스를 하는 래스터 테이블 명칭.
- r_raster_column 오버뷰 서비스를 하는 래스터 테이블의 래스터 Column 명칭.
- overview_factor – 오버뷰 테이블의 피라미드 레벨(Level). 숫자가 높을 수록 래스터 테이블의 해상도는 더 낮습니다. raster2pgsql 에서 이미지들의 폴더가 주어지면 각 이미지 파일의 크기를 계산하고 각각 로드 합니다. 레벨 1 은 항상 원본 파일입니다. 레벨 2 는 원래 타일의 개수를 4 로 나눈 만큼의 타일을 가지고 있습니다. 예로, 5,000x5,000 픽셀의 해상도를 가지는 이미지 파일을 125x125 사이즈의 타일로 나누어 로드 할 경우, 기본 테이블(Level 1)은 (5,000*5,000) / (125*125) = 1,600 개의 레코드를 저장하며, o_2 테이블(Level 2)은 ceiling(1,600 / Power(2, 2)) = 400 개의 레코드를 저장합니다. o_3 테이블(Level 3)은 ceiling(1,600 / Power(2, 3)) = 200 개의 레코드를 저장합니다. 만약 픽셀이 타일의 크기에 따라 정확하게 나누어지지 않는 경우 완전히 채워지지 않은 조각난 타일을 얻게 될 것입니다. raster2pgsql 에 의해 생성된 각각의 오버뷰 타일의 픽셀 수는 원본 파일과 동일하지만 낮은 해상도 에서는 피라미드 레벨에 따라 생성된 픽셀(Power(2,overview_factor) pixels of the original) 이 대신한다는 점을 주의하십시오.

5.3. PostGIS 래스터를 활용한 사용자 응용프로그램 구축

PostGIS 의 래스터는 잘 알려진 이미지 형식으로 래스터를 렌더링하기 위해 SQL 함수를 제공하며 더불어 많은 옵션을 제공합니다. 예를 들어 Rendering PostGIS Raster graphics with LibreOffice Base Reports 문서에서 보여준 바와 같이 렌더링을 위해 OpenOffice / LibreOffice 를 사용할 수 있습니다. 또한, 이 섹션에 설명하는 것처럼 다양한 언어를 활용할 수 있습니다.

5.3.1. ST_AsPNG 내보내기 기능과 다른 래스터 기능들을 활용한 PHP 예시

이 섹션에서는 PHP PostgreSQL driver 를 사용하는 방법과 img src html tag 에 포함할 수 있는 PHP 요청 스트림에 ST_AsGDALRaster 제품군 기능을 활용하여 래스터의 1, 2, 3 밴드를 출력하는 방법을 보여줍니다.

샘플 쿼리는 다음과 같이 많은 래스터의 기능들을 어떻게 결합 하는지를 보여줍니다. 특정 wgs84 바운딩 박스에 교차하는 모든 래스터의 타일을 찾아내고 ST_Union 으로 찾아낸 타일의 모든 밴드를 병합합니다. 그 다음 ST_Transform 을 활용하여 사용자가 지정한 투영정보(EPSG:26986)로 변환하고 ST_AsPNG 로 결과 래스터를 png 형식으로 출력합니다.

아래와 같이 호출하여 사용할 수 있습니다.

```
http://mywebserver/test_raster.php?srid=2249
```

메사추세츠 주 평면 피트의 래스터 이미지를 얻기 위해 다음의 소스코드를 사용합니다.

```
<?php
/** contents of test_raster.php **/
$conn_str    ='dbname=mydb    host=localhost    port=5432    user=myuser
password=mypwd';
$dbconn = pg_connect($conn_str);
header('Content-Type: image/png');
/**If a particular projection was requested use it otherwise use mass
state plane meters **/
if (!empty( $_REQUEST['srid'] ) && is_numeric( $_REQUEST['srid']) ){
             $input_srid = intval($_REQUEST['srid']);
}
else { $input_srid = 26986; }
/** The set bytea_output may be needed for PostgreSQL 9.0+, but not for
8.4 **/
$sql = "set bytea_output='escape';
SELECT ST_AsPNG(ST_Transform(
                    ST_AddBand(ST_Union(rast,1),
ARRAY[ST_Union(rast,2),ST_Union(rast,3)])
                          ,$input_srid) ) As new_rast
 FROM aerials.boston
       WHERE
        ST_Intersects(rast, ST_Transform(ST_MakeEnvelope(-71.1217, 42.227,
-71.1210, 42.218,4326),26986) )";
$result = pg_query($sql);
```

```
$row = pg_fetch_row($result);
pg_free_result($result);
if ($row === false) return;
echo pg_unescape_bytea($row[0]);
?>
```

5.3.2. ST_AsPNG 내보내기 기능과 다른 래스터 기능들을 활용한 ASP.NET C# 예시

이 섹션에서는 Npgsql PostgreSQL .NET driver 를 사용하는 방법과 img src html tag 에 포함할 수 있는 PHP 요청 스트림에 ST_AsGDALRaster 제품군 기능을 활용하여 래스터의 1, 2, 3 밴드를 출력하는 방법을 보여줍니다.

아래 예시를 실행하기 위해서는 Npgsql PostgreSQL .NET driver 가 필요하며 해당 드라이버는 http://npgsql.projects.postgresql.org/에서 제공합니다. 최신 버전을 다운 받고 ASP.NET 이 설치된 경로의 bin 폴더에 복사하면 해당 드라이버를 사용할 수 있습니다.

샘플 쿼리는 다음과 같이 많은 래스터의 기능들을 어떻게 결합 하는지를 보여줍니다. 특정 wgs84 바운딩 박스에 교차하는 모든 래스터의 타일을 찾아내고 ST_Union 으로 찾아낸 타일의 모든 밴드를 병합합니다. 그 다음 ST_Transform 을 활용하여 사용자가 지정한 투영정보(EPSG:26986)로 변환하고 ST_AsPNG 로 결과 래스터를 png 형식으로 출력합니다.

아래 예시는 C#으로 구현된 부분을 제외하고 5.3.1 단락의 예시와 동일합니다.

아래와 같이 호출하여 사용할 수 있습니다.

```
http://mywebserver/TestRaster.ashx?srid=2249
```

메사추세츠 주 평면 피트의 래스터 이미지를 얻기 위해 다음의 소스코드를 사용합니다.

```
-- web.config connection string section --
<connectionStrings>
    <add name="DSN"
        connectionString="server=localhost;database=mydb;Port=5432;User
Id=myuser;password=mypwd"/>
</connectionStrings>
// Code for TestRaster.ashx
<%@ WebHandler Language="C#" Class="TestRaster" %>
using System;
using System.Data;
using System.Web;
using Npgsql;

public class TestRaster : IHttpHandler
{
        public void ProcessRequest(HttpContext context)
        {

                context.Response.ContentType = "image/png";
                context.Response.BinaryWrite(GetResults(context));
```

```
        }

        public bool IsReusable {
                get { return false; }
        }

        public byte[] GetResults(HttpContext context)
        {
                byte[] result = null;
                NpgsqlCommand command;
                string sql = null;
                int input_srid = 26986;
        try {
                    using       (NpgsqlConnection       conn       =       new
NpgsqlConnection(System.Configuration.ConfigurationManager.ConnectionStrin
gs["DSN"].ConnectionString)) {
                          conn.Open();

                if (context.Request["srid"] != null)
                {
                   input_srid = Convert.ToInt32(context.Request["srid"]);
                }
                sql = @"SELECT ST_AsPNG(
                          ST_Transform(
                                    ST_AddBand(
                             ST_Union(rast,1),
ARRAY[ST_Union(rast,2),ST_Union(rast,3)])
                                                ,:input_srid) ) As new_rast
                      FROM aerials.boston
                            WHERE
                               ST_Intersects(rast,
                               ST_Transform(ST_MakeEnvelope(-71.1217,
42.227, -71.1210, 42.218,4326),26986) )";
                          command = new NpgsqlCommand(sql, conn);
                command.Parameters.Add(new       NpgsqlParameter("input_srid",
input_srid));


                          result = (byte[]) command.ExecuteScalar();
                conn.Close();
                       }

                }
        catch (Exception ex)
        {
           result = null;
           context.Response.Write(ex.Message.Trim());
        }
                return result;
        }
}
```

5.3.3. 래스터 쿼리를 통해 이미지 파일로 출력하는 자바 콘솔 응용프로그램

아래 예시는 쿼리를 활용하여 하나의 이미지를 지정한 파일형식으로 출력하는 간단한 자바 콘솔 응용 프로그램입니다. 최신 PostgreSQL JDBC 드라이버는 http://jdbc.postgresql.org/download.html 에서 다운받을 수 있습니다.

아래 코드의 명령어를 사용하여 컴파일 할 수 있습니다.

```
set env CLASSPATH .:..\postgresql-9.0-801.jdbc4.jar
javac SaveQueryImage.java
jar cfm SaveQueryImage.jar Manifest.txt *.class
```

그리고 아래와 같이 Command-line 에서 호출하여 사용할 수 있습니다.

```
java -jar SaveQueryImage.jar "SELECT
ST_AsPNG(ST_AsRaster(ST_Buffer(ST_Point(1,5),10, 'quad_segs=2'),150, 150,
'8BUI',100));" "test.png"
```

```
-- Manifest.txt --
Class-Path: postgresql-9.0-801.jdbc4.jar
Main-Class: SaveQueryImage
// Code for SaveQueryImage.java
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.io.*;

public class SaveQueryImage {
  public static void main(String[] argv) {
    System.out.println("Checking    if    Driver    is    registered    with
DriverManager.");

    try {
      //java.sql.DriverManager.registerDriver                          (new
org.postgresql.Driver());
      Class.forName("org.postgresql.Driver");
    }
    catch (ClassNotFoundException cnfe) {
      System.out.println("Couldn't find the driver!");
      cnfe.printStackTrace();
      System.exit(1);
    }

    Connection conn = null;

    try {
      conn                                                                =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/mydb","myuse
r", "mypwd");
      conn.setAutoCommit(false);

      PreparedStatement sGetImg = conn.prepareStatement(argv[0]);
```

```
      ResultSet rs = sGetImg.executeQuery();

             FileOutputStream fout;
             try
             {
                    rs.next();
                    /** Output to file name requested by user **/
                    fout = new FileOutputStream(new File(argv[1]) );
                    fout.write(rs.getBytes(1));
                    fout.close();
             }
             catch(Exception e)
             {
                    System.out.println("Can't create file");
                    e.printStackTrace();
             }

      rs.close();
             sGetImg.close();
      conn.close();
     }
     catch (SQLException se) {
      System.out.println("Couldn't connect: print out a stack trace and
exit.");
      se.printStackTrace();
      System.exit(1);
     }
  }
}
```

5.3.4. SQL 을 통해 이미지를 덤프하기 위한 PLPython

서버의 디렉토리에 각 레코드들을 파일로 생성하는 PLPython 의 저장 기능입니다.

```
//plpython postgresql stored proc.  Requires you have plpython installed
CREATE OR REPLACE FUNCTION write_file (param_bytes bytea, param_filepath
text)
RETURNS text
AS $$
f = open(param_filepath, 'wb+')
f.write(param_bytes)
return param_filepath
$$ LANGUAGE plpythonu;
```

```
--write out 5 images to the PostgreSQL server in varying sizes
--  note the postgresql daemon account needs to have write access to
folder
-- this echos back the file names created;
 SELECT write_file(ST_AsPNG(
       ST_AsRaster(ST_Buffer(ST_Point(1,5),j*5,      'quad_segs=2'),150*j,
150*j, '8BUI',100)),
        'C:/temp/slices'|| j || '.png')
        FROM generate_series(1,5) As j;
```

```
    write_file
----------------------
 C:/temp/slices1.png
 C:/temp/slices2.png
 C:/temp/slices3.png
 C:/temp/slices4.png
 C:/temp/slices5.png
```

5.3.5. PSQL 의 래스터 출력하기

애석하게도 PSQL 은 바이너리를 출력을 위한 빌트인 기능의 활용이 쉽지 않습니다. 이 방법은 다소 편법(hack)의 일부이며, PostgreSQL 의 피기백(Piggy back) 방식으로 기존 대용량 객체를 지원하는 Clever Trick Challenge -- Outputting bytea with psql 문서에 서술된 제안들 중 하나에 기반합니다. psql 명령창에서 해당 기능을 사용하기 위해서는 데이터베이스에 먼저 연결이 되어 있어야 합니다.

이 방법은 Python 방식과는 달리 로컬 컴퓨터에 파일을 직접 만듭니다.

```
SELECT oid, lowrite(lo_open(oid, 131072), png) As num_bytes
 FROM
 ( VALUES (lo_create(0),
   ST_AsPNG( (SELECT rast FROM aerials.boston WHERE rid=1) )
  ) ) As v(oid,png);
-- you'll get an output something like --
   oid   | num_bytes
---------+-----------
 2630819 |    74860

-- next note the oid and do this replacing the c:/test.png to file path
location
-- on your local computer
 \lo_export 2630819 'C:/temp/aerial_samp.png'

-- this deletes the file from large object storage on db
SELECT lo_unlink(2630819);
```

Chapter 6.PostGIS 지오메트리 사용하기: 어플리케이션 만들기

6.1. Using MapServer

미네소타 MapServer 는 OpenGIS 웹매핑 서버사양을 준수하는 인터넷 웹매핑 서버입니다.

- MapServer 홈페이지는 http://mapserver.org.
- OpenGIS 웹맴사양은 http://www.opengeospatial.org/standards/wms 에서 찾을 수 있습니다.

6.1.1. 기본 사용법

MapServer 와 PostGIS 를 사용하려면, 이 문서의 범위를 벗어나는 MapServer 를 구성하는 방법에 대해 알아야 합니다. 이 섹션에서는 특정 PostGIS 문제 및 구성 세부사항을 다룰 것 입니다.

MapServer 와 PostGIS 를 사용하려면, 다음이 필요합니다:

- PostGIS 0.6 버전이상.
- MapServer 3.5 버전이상.

MapServer 는 다른 PostgreSQL 의 클라이언트처럼 PostGIS / PostgreSQL 의 데이터에 libpq 라이브러리 인터페이스를 사용하여 엑세스합니다. 이는 PostGIS 서버에 접근 가능한 네트워크를 가진 그 어떤 기계 위에서도 MapServer 는 설치 가능하다는 것을 의미하며 데이터 소스로서 PostGIS 를 사용한다는 것을 의미합니다. 연결이 빠를수록 더 좋습니다

1. 설정 옵션을 사용하여 MapServer 를 컴파일하고 설치하십시오 ("--with-postgis" 구성 옵션)
2. MapServer 맵파일안에 PostGIS 레이어를 추가합니다. 예를 들면:

```
LAYER
  CONNECTIONTYPE postgis
  NAME "widehighways"
  # Connect to a remote spatial database
  CONNECTION "user=dbuser dbname=gisdatabase host=bigserver"
  PROCESSING "CLOSE_CONNECTION=DEFER"
```

```
  # Get the lines from the 'geom' column of the 'roads' table
  DATA "geom from roads using srid=4326 using unique gid"
  STATUS ON
  TYPE LINE
  # Of the lines in the extents, only render the wide highways
  FILTER "type = 'highway' and numlanes >= 4"
  CLASS
    # Make the superhighways brighter and 2 pixels wide
    EXPRESSION ([numlanes] >= 6)
    STYLE
      COLOR 255 22 22
      WIDTH 2
    END
  END
  CLASS
    # All the rest are darker and only 1 pixel wide
    EXPRESSION ([numlanes] < 6)
    STYLE
      COLOR 205 92 82
    END
  END
END
```

위의 예에서, PostGIS 특정 지침은 다음과 같습니다:

연결유형(CONNECTIONTYPE) PostGIS 레이어는 항상 "postgis 입니다

연결(CONNECTION) 데이터베이스 연결은 키와 값의 표준집합인 '연결문자열(connection string)'에 의해 제어됩니다 (<>안 기본값과 함께):
user=<username> password=<password> dbname=<username> hostname=<server> port=<5432>
빈 연결문자열은 유효하며, 키/값 쌍은 생략할 수 있습니다. 최소한 연결을 위해서는 dbname 과 username 을 제공해야 합니다.

데이터(DATA) 이 매개변수의 형식은 공간컬럼이 맵에 렌더링 될 수 있도록 하는 "<geocolumn> from <tablename> using srid=<srid> using unique <primary key>입니다.
"using SRID"과 "using unique" 절들을 생략하면 MapServer 가 자동으로 올바른 값을 결정합니다. 하지만 각 맵을 그리기 위한 서버에 몇 가지 추가 쿼리를 실행하는 비용이 존재합니다.

프로세싱(PROCESSING) 만약 여러 레이어들을 가지고 있으시다면 클로징하는 대신 존재하는 연결을 재사용하는 옵션인 CLOSE_CONNECTION = DEFER 를 사용하세요. 이것은 속도를 향상 시킵니다.더 자세한 설명은 Mapserver PostGIS 성능에 관한 팁을 참조하십시오.

필터(FILTER) 필터는 SQL 쿼리의 “WHERE” 키워드에 따르는 로직에 상응하는 유효한 SQL 문자열 이어야 만 합니다. 예를 들어, 6 개 이상의 차선만 도로를 렌더링하기 위해선 "num_lanes> = 6" 의 필터를 사용합니다.

3. 본인의 공간 데이터베이스에서 그리게 될 레이어를 위한 공간(GiST)인덱스를 가지고 있는 지 확인 하십시오.

```
CREATE INDEX [indexname] ON [tablename] USING GIST
( [geometrycolumn] );
```

4. 이 MapServer 를 사용하여 계층을 쿼리 할 경우 본인의 데이터 명령문에서 " using unique" 절을 사용하셔야 합니다.
   쿼리를 수행할 때 MapServer 는 각각의 공간 레코드에 대한 고유 식별자를 필요로 하고, MapServer 의 PostGIS 모듈은 이러한 고유식별자를 제공하기 위해 지정한 유일한 값을 사용합니다. 테이블의 기본키를 사용하는 것이 좋습니다.

6.1.2. 자주 묻는 질문들

1. 맵파일에 EXPRESSION 을 사용하는 경우 테이블에 존재하는 값들을 알고 있음에도 불구하고 조건은 결코 true 로서 반환되지 않습니다.

shape 파일과는 달리, PostGIS 필드 이름은 소문자를 사용하여 EXPRESSION 에서 참조할 수 있습니다.

```
EXPRESSION ([numlanes] >= 6)
```

2. shape 파일을 위해 사용하는 필터가 동일한 데이터 내 PostGIS 테이블에 대해 작동하지 않습니다.

Shape 파일과는 달리, PostGIS 레이어에 대한 필터는 SQL syntax 를 사용합니다(그것들은 MapServer 에 레이어를 그리기 위해 PostGIS 커넥터가 생성하는 SQL 명령문에 추가됩니다).

```
FILTER "type = 'highway' and numlanes >= 4"
```

3. PostGIS 레이어가 shape 파일 레이어보다 훨씬 늦게 그립니다 정상입니까?

일반적으로, 더 많은 기능을 지정된 맵에 그릴 경우, shape 파일보다 PostGIS 가 느려질 가능성이 짙어집니다. 상대적으로 몇가지 피쳐(100S)만 있는 맵의 경우, PostGIS 는 빠릅니다. 높은 피쳐밀도를 가진맵(1000s)의 경우 PostGIS 은 언제나 느립니다.

상당한 드로잉 성능 문제를 발견하는 경우 그 원인은 테이블에 공간 인덱스를 구축하지 않았기 때문일 가능성이 있습니다.

```
postgis# CREATE INDEX geotable_gix ON geotable USING GIST ( geocolumn );
postgis# VACUUM ANALYZE;
```

4. 내 PostGIS 레이어는 그리는데 문제가 전혀 없으나 쿼리는 너무 느립니다. 무엇이 문제입니까?

쿼리의 속도가 빨라지기 위해선 공간테이블을 위한 고유키를 반드시 가지고 있어야 하며 그 고유키는 인덱스를 가지고 있어야 합니다.

DATA 라인에서 USING UNIQUE 절과 함께 MapServer 를 위한 고유키를 명시해줍니다.

```
DATA "geom FROM geotable USING UNIQUE gid"
```

5. **MapServer 레이어에 대한 소스로 "geography" 컬럼(PostGIS 1.5 의 새로운 기능)을 사용할 수 있습니까**

가능합니다. MapServer 는 지오메트리 컬럼과 동일한것으로 geography 컬럼을 이해합니다. 그러나 항상 4326 의 SRID 를 사용하여. 당신의 DATA 구문에서 "using srid=4326"절을 반드시 포함 하여야 합니다. 나머지 다른 모든 것은 정확히 지오메트리와 동일하게 작동합니다.

```
DATA "geog FROM geogtable USING SRID=4326 USING UNIQUE gid"
```

6.1.3. 고급 사용법

USING pseudo-SQL 절은 MapServer 가 보다 복잡한 쿼리 결과를 이해하는데 도움을 주기 위해 사용됩니다. 보다 명확하게 (DATA 정의의 "FROM" 의 오른쪽에 명시한) 뷰 또는 부속 질의(subselect)가 소스테이블로 사용될 때 MapServer 는 자동으로 각행에 대한 고유식별자 또한 테이블의 SRID 를 결정하는데 더 어려움을 느낍니다. USING 절은 다음과 같은 두 가지 정보를 MapServer 에게 제공할 수 있습니다:

```
DATA "geom FROM (
                  SELECT
table1.geom AS geom,
table1.gid AS gid,
table2.data AS data
  FROM table1 7
                          LEFT JOIN table2
                          ON table1.id = table2.id
) AS new_table USING UNIQUE gid USING SRID=4326"
```

USING UNIQUE <uniqueid> MapServer 는 맵쿼리를 수행할 때 행을 식별하기 위해 각 행에 대한 고유 ID 를 요구합니다. 일반적으로 시스템 테이블로부터 주요키를 식별합니다. 그러나 뷰와 부속질의는 알려진 고유행을 자동적으로 가지고 있지 않습니다. MapServer 의 쿼리기능을 사용하려는 경우 뷰 또는 부속질의가 고유값 열을 포함하고 이것을 USING UNIQUE 와 함께 선언해야 합니다. 예를 들어 이러한 목적을 위한 주요키 값들의 nee 혹은 결과값의 고유함을 보장하기 위한 추가 컬럼을 명시적으로 선언할 수 있습니다.

"맵 쿼리하기 (Quering a Map)"는 지도 위치에 있는 정보를 요청할 수 있는 지도를 클릭하는 동작입니다. DATA 정의의 SQL 질의를 "맵 쿼리(map queries)"와 혼동하지 마십시오.

USING SRID=<srid> PostGIS 은 MapServer 로 올바른 데이터를 반환하기 위해 지오메트리에 의해 어떤 공간 레퍼런싱 시스템이 사용되고 있는지에 대해 알아야 합니다. 일반적으로 PostGIS 데이터베이스에 "geometry_columns"테이블에서 해당 정보를 찾을 수 있습니다. 그러나 부속질의 및 뷰와 같이 즉시 만들어진 테이블에겐 불가능한 일입니다. 그러므로 USING SRID= 옵션은 올바른 SRID 가 데이터정의에 지정될 수 있게 허용합니다.

6.1.4. 예시

간단한 예시부터 시작합시다. 다음의 MapServer 레이어 정의를 고려해 봅시다:

```
LAYER
  CONNECTIONTYPE postgis
  NAME "roads"
  CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"
  DATA "geom from roads"
  STATUS ON
  TYPE LINE
  CLASS
    STYLE
      COLOR 0 0 0
    END
  END
END
```

이 레이어는 검은색 선으로 도로테이블의 모든 도로형상을 표시합니다.

자 이제 우리가 적어도 1:100000 축척으로 확대하기 전까지 오직 고속도로만 보여주고 싶다고 가정합시다 – 다음 두개의 레어어들이 이 효과를 보여줄 것 입니다:

```
LAYER
  CONNECTIONTYPE postgis
  CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"
  PROCESSING "CLOSE_CONNECTION=DEFER"
  DATA "geom from roads"
  MINSCALE 100000
  STATUS ON
  TYPE LINE
  FILTER "road_type = 'highway'"
  CLASS
    COLOR 0 0 0
  END
END
LAYER
  CONNECTIONTYPE postgis
  CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"
  PROCESSING "CLOSE_CONNECTION=DEFER"
  DATA "geom from roads"
  MAXSCALE 100000
  STATUS ON
  TYPE LINE
  CLASSITEM road_type
  CLASS
    EXPRESSION "highway"
    STYLE
      WIDTH 2
      COLOR 255 0 0
    END
  END
  CLASS
    STYLE
```

```
      COLOR 0 0 0
    END
  END
END
```

첫번째 레이어는 축척 1:100000 보다 큰 경우 사용되며, 검은색 선으로 유형 "고속도로"의 도로들만을 표시합니다. FILTER 옵션은 오직 타입 “hightway”의 도로들 만이 표시 되도록 설정합니다.

두번째 레이어는 축척이 1:100000 보다 적을 때 사용되며, 고속도로는 두꺼운 붉은선으로 표현되고 다른 도로들은 검은색으로 표시합니다.

그러므로 오직 MapServer 기능을 사용하여 몇 가지 흥미로운 일을 했지만 DATA SQL 문장은 여전히 간단합니다. (어떤 이유에서 든 간에) 도로의 이름이 다른 테이블 에 저장되어 있고 우리는 그것을 연결하여 도로들에게 라벨을 붙여야 해야 한다고 가정 해 봅시다.

```
LAYER
  CONNECTIONTYPE postgis
  CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"
  DATA "geom FROM (SELECT roads.gid AS gid, roads.geom AS geom,
road_names.name as name FROM roads LEFT JOIN road_names ON
       roads.road_name_id = road_names.road_name_id)
       AS named_roads USING UNIQUE gid USING SRID=4326"
  MAXSCALE 20000
  STATUS ON
  TYPE ANNOTATION
  LABELITEM name
  CLASS
    LABEL
      ANGLE auto
      SIZE 8
      COLOR 0 192 0
      TYPE truetype
      FONT arial
    END
  END
END
```

축척 1:20000 이하로 내려오면 주석 레이어의 모든 도로에 녹색 레이블을 추가합니다. 이는 또한 DATA 정의의 SQL join 을 어떻게 사용하는지를 보여줍니다.

6.2. 자바 클라이언트 (JDBC)

Java 클라이언트는 텍스트 문자열로서 직접적으로 혹은 PostGIS 와 함께 제공되는 JDBC 확장개체를 사용함으로써 PosrgreSQL 테이터베이스에 있는 PostGIS “지오메트리” 오브젝트에 접근할 수 있습니다. 확장 개체들을 이용하기 위해서는 "postgis.jar" 파일이 반드시 "postgresql.jar" JDBC 드라이버 패키지와 함께 본인의 CLASSPATH 에 있어야 합니다.

```
import java.sql.*;
import java.util.*;
```

```
import java.lang.*;
import org.postgis.*;

public class JavaGIS {

public static void main(String[] args) {

java.sql.Connection conn;

try {
   /*
   * Load the JDBC driver and establish a connection.
   */
Class.forName("org.postgresql.Driver");
   String url = "jdbc:postgresql://localhost:5432/database";
conn = DriverManager.getConnection(url, "postgres", "");
   /*
   * Add the geometry types to the connection. Note that you
   * must cast the connection to the pgsql-specific connection
   * implementation before calling the addDataType() method.
   */

((org.postgresql.PGConnection)conn).addDataType("geometry",Class.forName("
org.postgis.PGgeometry"));

((org.postgresql.PGConnection)conn).addDataType("box3d",Class.forName("org
.postgis.PGbox3d"));
   /*
   * Create a statement and execute a select query.
   */
   Statement s = conn.createStatement();
   ResultSet r = s.executeQuery("select geom,id from geomtable");
while( r.next() ) {
     /*
     * Retrieve the geometry as an object then cast it to the geometry
type.
     * Print things out.
     */
     PGgeometry geom = (PGgeometry)r.getObject(1);
int id = r.getInt(2);
System.out.println("Row " + id + ":");
System.out.println(geom.toString());
   }
s.close();
conn.close();
  }
catch( Exception e ) {
e.printStackTrace();
  }
}
}
```

"PGgeometry" 객체는 point, Linestring, Polygon, MultiPoint, MultiLineString, MultiPolygon 타입과 같은 특정 위상구조를 갖는 래퍼(wrapper) 개체(추상클래스 "Geometry"의 서브클래스)입니다:

```
PGgeometry geom = (PGgeometry)r.getObject(1);
if( geom.getType() == Geometry.POLYGON ) {
  Polygon pl = (Polygon)geom.getGeometry();
for( int r = 0; r < pl.numRings(); r++) {
   LinearRing rng = pl.getRing(r);
System.out.println("Ring: " + r);
for( int p = 0; p < rng.numPoints(); p++ ) {
     Point pt = rng.getPoint(p);
System.out.println("Point: " + p);
System.out.println(pt.toString());
   }
  }
}
```

확장 개체를 위한 Javadoc 은 지오메트리 객체의 다양한 데이터 접근 기능에 대한 레퍼런스를 제공합니다.

6.3. C 클라이언트 (libpq)

...

6.3.1. Text Cursors

...

6.3.2. Binary Cursors

...

Chapter 7.Performance tips

7.1. 큰 지오메트리의 작은 테이블들

7.1.1. 문제 설명

현재의 PostgreSQL 버전 (8.0 포함) TOAST 테이블에 대한 쿼리 최적화 약점으로 어려움을 겪고 있습니다. TOAST 테이블은 일반 데이터 페이지 (많은 꼭지점을 가진 복잡한 지오메트리 또는 긴 텍스트, 이미지 같은) 에 맞지 않는 큰 값을 저장하는데(데이터크기의 의미에서) 사용 되는 "확장자룸"의 일종입니다. 보다 더 많은 정보를 위해서 the PostgreSQL Documentation for TOAST 보십시오.

너무 큰 지오메트리를 가진 테이블을 가지고 있으나 테이블의 열들이 너무 많진 않을 때(고해상도의 모든 유럽 국가들의 경계선 들을 포함하고 있는 테이블 같이) 문제는 발생합니다. 그러면 테이블 자체는 작습니다, 그러나 많은 TOAST 공간을 사용합니다. 우리 예제의 경우, 테이블 자체가 80 행이 있고 단지 3 데이터 페이지를 사용하지만, TOAST 테이블은 8225 페이지를 사용했습니다. 이제 오직 그 행들중 몇 개만 일치하는 바운딩 박스를 검색하기 위해 지오메트리 오퍼레이터&&을 사용하는 쿼리를 실행합니다. 이제 쿼리 최적화 프로그램은 테이블 오직 3 페이지와 80 열들만 가진다고 간주합니다. 그는 작은 테이블에 순차적 스캔이 인덱스를 사용하는 것 보다훨씬 빠른 것 으로 추정하고 있습니다. 고로 그는 GIST 인덱스를 무시하기로 결정합니다. 일반적으로,이 예측은 정확합니다. 그러나 우리의 경우, &&오퍼레이터(연산자)는 경계상자를 비교하기 위해 디스크에서 모든 지오메트리를 가져와야 만 합니다. 이 버그로 어려움을 겪고 있는지 여부를확인 하려면 "EXPLAIN ANALYZE "PostgreSQL 명령어을 사용하십시오. Postgres 성능 메일링리스트의 스레드(thread)를 읽으실 수 있습니다(참조하실 수 있습니다): http://archives.postgresql.org/pgsql-performance/2005-02/msg00030.php

7.1.2. 해결 방법

The PostgreSQL 사람들은 쿼리추정 TOAST-aware 을 만들면서 해당 문제를 해결하기 위해 노력하고 있습니다. 현재 두 가지 해결 방법이 있습니다:

첫번째 해결 방법은 쿼리 계획이 인덱스를 사용하도록 강제하는 것 입니다. 서버에 쿼리를 실행하기 전에, "SET enable_seqscan TO off"을 보낼 수 있습니다. 이것은 기본적으로 순차적 스캔을 가능한 피하기 위해 쿼리 계획을 강제합니다. 그리하여 평소처럼 GIST 인덱스를 사용합니다. 하지만 이

플래그(flag)는 모든 연결에 설정 되어야 하고, 다른 경우에 오추정을 만들기 위해 쿼리 계획을 야기합니다. 그러므로 쿼리 후 "SET enable_seqscan TO on;" 하셔야 만 합니다.

두번째 해결 방법은 쿼리 플래너가 생각하는 것과 같이 최대한 빨리 빨리 순차적 스캔을 만드는 것입니다. Bbox 를 "캐시(catches)"하는 추가적인 행을 생성하고 이것에 매칭함으로써 이는 달성될 수 있습니다. 명령어는 다음과 같습니다:

```
SELECT
AddGeometryColumn('myschema','mytable','bbox','4326','GEOMETRY','2');
UPDATE mytable SET bbox = ST_Envelope(ST_Force_2d(the_geom));
```

이제 geom_column 대신 bbox 에 &&오퍼레이터를 사용하기 위해 본인의 쿼리를 바꾸십시오:

```
SELECT geom_column
FROM mytable
WHERE bbox && ST_SetSRID('BOX3D(0 0,1 1)'::box3d,4326);
```

변경하거나 MYTABLE 에 행을 추가하는 경우 물론, 당신은 "동기화(in sync)" BBOX 을 유지해야 합니다. 이 작업을 수행하는 가장 투명한 방법은 트리거(triggers) 될 것이다, 그러나 또한 BBOX 열을 최신 상태로 유지하거나 모든 수정 후위의 UPDATE 쿼리를 실행하는 응용프로그램을 수정할 수 있습니다.

7.2. 지오메트리 인덱스에 CLUSTERing

대부분의 단일 인덱스를 가진 읽기전용 테이블을 위해, PostgreSQL 은 CLUSTER 명령어를 제공합니다. 이 명령어는 두 가지 성능 이점을 산출하면서 인덱스 크리테리아(index criteria)와 같은 순서로 모든 데이터 열들을 물리적으로 재정렬 합니다: 첫째, 인덱스 범위 스캔에 대한 데이터 테이블의 탐색수는 크게 감소 됩니다. 둘째, 본인의 작업집합이 인덱스의 몇 작은 간격에 집중한다면 보다 효율적인 캐싱을 가질 수 있습니다. 왜냐하면 데이터 열들이 일부 적은 데이터 페이지들 위해 펼쳐지기 때문입니다. (이 시점에서 PostgreSQL 의 사용설명서의 CLUSTER 명령 문서를 읽으십시오)

GIST 인덱스는 단순히 NULL 값을 무시하기 때문에, 현재 PostgreSQL 의 이 PostGIS 의 GIST 인덱스에 대한 클러스터링을 허용하지 않습니다, .디음과 같은 오류메시지를 보시게 될 것입니다:

```
lwgeom=# CLUSTER my_geom_index ON my_table;
ERROR: cannot cluster when index access method does not handle null
values
HINT: You may be able to work around this by marking column "the_geom"
NOT NULL.
```

HINT 메시지가 알려주는것과 같이 하나의 테이블에 "null 이 아닌"제약조건을 추가하여 이 결핍 문제를 해결할 수 있습니다:

```
lwgeom=# ALTER TABLE my_table ALTER COLUMN the_geom SET not null;
ALTER TABLE
```

본인의 지오메트리열에 NULL 값을 필요로 하는 경우에, 물론 이는 작동하지 않습니다. 또한 "ALTER TABLE blubb ADD CHECK (지오메트리가 null 이 아닌)"와 같은 CHECK 제약조건을 사용하여 제약조건을 추가하려면 위의 방법을 사용해야 합니다.

7.3. 차원변환 방지하기

때때로 테이블에 3D 또는 4D 데이터를 가질 경우가 생길 것입니다. 하지만 항상 2D 지오메트리만 출력하는 ST_AsBinary()와 OpenGIS compliant ST_AsText()을 사용하여 엑세스할 수 있습니다. Geometry 에 대해 상당한 오버헤드를 발생하는 ST_Force_2d() 기능을 내부적으로 호출함으로서 이것을 수행합니다. 이 오버헤드를 피하기 위해서 이 추가차원을 일회 그리고 영원히 사전중단 할 수 있습니다:

```
UPDATE mytable SET the_geom = ST_Force_2d(the_geom);
VACUUM FULL ANALYZE mytable;
```

AddGeometryColumn() 사용하여 지오메트리 컬럼을 추가한 경우 지오메트리 차원에 제약 조건이 있을거라는 점을 주의하십시오. 이를 우회 하려면 제약조건을 삭제해야 합니다. geometry_columns 테이블에서 항목을 업데이트 하기 위해선 이후 제약조건을 다시 생성해야 한다는 점을 기억하십시오.

대형 테이블의 경우, WHERE 절 및 기본키 또는 다른가능한 기준을 통해 UPDATE 가 테이블의 일부가 되는 것을 제한하고 업테이트들간의 간단한 VACUUM 을 실행 함으로써 이 UPDATE 를 작은 부분으로 분할하는 것이 현명할 수 있습니다. 이는 임시 디스크공간을 위한 필요가 현저히 줄어듭니다. 더하여 만약 혼합 차원 지오메트리을 가지고 계신다면 "WHERE dimension(the_geom)>2" 이 이미 2d 에 있는 지오메트리를 스킵함으로써 UPDATE 를 제한할 수 있습니다.

7.4. configuration 조정

이 팁은 FOSS4G 2007 컨퍼런스 Kevin Neufeld 의 프리젠테이션 "Tips for the PostGIS Power User"에서 발췌된 것입니다. PostGIS 의 사용에 따라 (예를 들어, 정적 데이터와 복잡한 분석 vs 자주 업데이트 되는 데이터 및 많은 사용자) 이러한 변화는 쿼리에 상당한 속도 향상을 제공할 수 있습니다.

더 많은 팁(더 나은 서식)을 원하면 http://2007.foss4g.org/presentations/view.php?abstract_id=117 에서 원본 프리젠테이션을 찾으십시오.

7.4.1. 시작

이 설정들은 postgresql.conf 에 구성됩니다:

checkpoint_segments

- 자동적인 WAL 체크포인트 사이의 로그파일 세그먼트의 최대 숫자 (각 세그먼트는 일반적으로 16MB); 기본값 3

- 무거운 쓰기 행위 혹은 큰 데이터베이스를 로딩하기 위한 데이터베이스들을 위해 적어도 10 또는 30 으로 설정. 더 많은 추천 읽을거리; Greg Smith: Checkpoint and Background writer
- xlog 를 독립적인 디스크 디바이스에 저장 가능

constraint_exclusion

- 기본값 : off (PostgreSQL 를 8.4 이전 및 PostgreSQL 를 8.4 버전에 대한 파티션 설정)
- 이것은 주로 테이블 분할을 위해 사용 니다. 만약 PostgreSQL 8.4 보다 낮은 버전을 실행 중이라면 원하는 대로 쿼리 플래너가 최적화 할 수 있도록 "on" 설정 하십시오. PostgreSQL 8.4 기준, 이것을 위한 기본값은 "파티션"으로 설정되어 있으며 이것은 상속 계층 구조에 있고 그렇지 않으면 플래너 패널티를 지불하지 않은 경우 제약 고려를 위한 테이블들 만을 분석하기 위한 플래너를 강제하기 때문에 이는 PosgreSQL 8.4.에 이상적입니다.

shared_buffers

- 기본값 : ~32MB
- 사용 가능한 RAM 의 약 1/3 ~ 3/4 로 설정

7.4.2. 런타임

work_mem (정렬 작업과 복잡한 쿼리를 위해 사용되는 메모리)

- 기본값 : 1MB
- 큰 dbs, 복잡한 쿼리, 많은 RAM 을 위한 상향조절.
- 많은 동시 사용자와 낮은 RAM 을 위한 하향조절.
- 만약 많은 RAM 과 적은 개발자들을 가지고 계신다면:

```
SET work_mem TO 1200000;
```

maintenance_work_mem (VACUUM, CREATE INDEX, 등을 위해 사용.)

- 기본값 : 16MB
- 일반적으로 매우 낮음– 메모리를 교환하는 동안에 I/O 가 묶이고 서브오브젝트들이 잠김
- 많은 RAM 을 가진 프로덕션 서버에서 32MB 에서 256MB 를 권장. 그러나 동시 사용자 의 수에 따라 달라짐, 만약 많은 RAM 과 적은 개발자 들을 가지고 계시다면:

```
SET maintainence_work_mem TO 1200000;
```

Chapter 8.PostGIS Reference

아래의 함수들은 PostGIS 사용자가 필요로 하는 함수들이며, 일반 사용자가 쓰지 않는 PostGIS 객체에 필요한 다른 지원 함수들도 있습니다.

PostGIS 는 기존 명명 규칙에서 SQL-MM-중심 명명 규칙으로 전환하기 시작했습니다. 그 결과로, 여러분이 알고 있고 좋아하는 대부분의 함수 이름이 표준 Spatial Type (ST) 접두어를 사용한 이름으로 변경되었습니다. 갱신된 함수들이 이전 함수들과 같은 기능을 제공하며, 비록 이 문서에는 나열되어 있지는 않지만 이전 함수도 사용할 수 있습니다. 다음 버전 출시에는 이 문서에 없는 비 ST_ 함수들이 더 이상 사용되지 않을 것이며 제외될 예정이므로 **이들 함수의 사용을 중단해 주십시오**.

8.1. PostgreSQL PostGIS Geometry/Geography/Box 유형

이 장에서는 PostGIS 에 의해 설치된 PostgreSQL 의 데이터 유형을 나열합니다. 다음은 우리가 자신의 함수를 설계할 때 특히 중요한 이들의 캐스팅 동작을 설명합니다.

캐스트란 한 유형을 다른 유형으로 강제 변환하는 때를 말합니다. PostgreSQL 은 사용자 정의 유형을 위한 캐스팅 동작 및 캐스팅에 사용되는 함수들을 정의할 수 있도록 한다는 점에서 대부분의 데이터베이스보다 특별합니다. otherfootype 유형만 처리하거나 자동 캐스트가 포함된 함수에 사용할 때처럼, CAST(myfoo As otherfootype) 또는 myfoo::otherfootype 를 명시적으로 수행할 필요가 없는 상황에서 캐스트는 자동으로 설정될 수 있습니다.

자동 캐스트 동작을 사용할 때 발생할 수 있는 위험은 하나는 Box2D 를 사용하고 또 하나는 지오메트리가 없는 Box3D 를 사용하는 오버로드 된 함수를 사용하는 경우 입니다. 지오메트리가 두 함수 모두를 위한 자동캐스트를 가지고 있기 때문에 만약 두 함수 모두 지오메트리와 함께 사용한다면 어떤 일이 발생할까요? -- 결국 모호한 함수 오류가 발생할 것입니다. 이런 경우 PostgreSQL 이 받아들일 수 있도록 CAST(mygeom As box3d) 또는 mygeom::box3d 캐스트를 사용해야 합니다.

적어도 PostgreSQL 8.3 기준에서 모든 객체는 텍스트로 캐스트 될 수 있습니다. 따라서 객체를 텍스트로 캐스트 하기 위해 별도로 정의된 캐스트는 없습니다.

box2d — xmin, ymin, xmax, ymax 로 구성된 박스. 지오메트리 또는 지오메트리 컬렉션의 2 차원 공간범위(2D Extent)를 반환하는데 주로 사용됩니다.

box3d — xmin, ymin, zmin, xmax, ymax, zmax 로 구성된 박스. 지오메트리 또는 지오메트리 컬렉션의 3 차원 공간범위(3D Extent)를 반환하는데 주로 사용됩니다.

geometry — 평면 좌표계 중심(Planar) 공간 데이터 유형.

geometry_dump — 두 개의 필드를 가진 공간 데이터 유형- geom (지오메트리 객체 포함) 및 path[](덤프된 객체 내 지오메트리의 위치를 유지하는 1 차원 배열).

geography — 타원체 중심(Ellipsoidal) 공간 데이터 유형.

PostgreSQL PostGIS Geometry/Geography/Box 유형과 관련된 이 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/reference.html#PostGIS_Types

8.2. 관리 함수

AddGeometryColumn — 기존 속성 테이블에 지오메트리 컬럼을 추가합니다. 기본적으로 컬럼을 정의하기 위해 제약 조건보다 오히려 유형 변경자를 사용합니다.

DropGeometryColumn — 공간 테이블에서 지오메트리 컬럼을 제거합니다.

DropGeometryTable — 테이블 및 geometry_columns 의 모든 참조를 삭제합니다.

PostGIS_Full_Version — PostGIS 버전과 빌드 환경 정보를 반환합니다.

PostGIS_GEOS_Version — GEOS 라이브러리의 버전 번호를 반환합니다.

PostGIS_LibXML_Version — libxml2 라이브러리의 버전 번호를 반환합니다.

PostGIS_Lib_Build_Date — PostGIS 라이브러리의 빌드 날짜를 반환합니다.

PostGIS_Lib_Version — PostGIS 라이브러리의 버전 번호를 반환합니다.

PostGIS_PROJ_Version — PROJ4 라이브러리의 버전 번호를 반환합니다.

PostGIS_Scripts_Build_Date — PostGIS 스크립트의 빌드 날짜를 반환합니다.

PostGIS_Scripts_Installed — 이 데이터베이스에 설치된 PostGIS 스크립트의 버전을 반환합니다.

PostGIS_Scripts_Released — 설치된 PostGIS 라이브러리와 함께 출시된 postgis.sql 스크립트의 버전 번호를 반환합니다.

PostGIS_Version — PostGIS 버전 번호 및 컴파일 시간 옵션을 반환합니다.

Populate_Geometry_Columns — 지오메트리 컬럼이 타입 변경자로 정의되거나 적절한 공간 제약을 가지고 있는지 확인합니다. 이 함수는 공간 관련 테이블들이 geometry_columns 뷰에 올바르게 등록되도록 합니다. 기본적으로 유형 변경자를 가지지 않는 모든 지오메트리 컬럼들을 유형 변경자를 가진 지오메트리 컬럼들로 변환시킵니다.

UpdateGeometrySRID — 지오메트리 컬럼, geometry_columns 메타데이터 및 srid 를 대상으로 모든 피처들의 SRID 를 갱신합니다. 만약 제약조건으로 설정된 경우 새 SRID 제약조건으로 갱신되고, 유형 정의로 설정된 경우 유형 정의가 변경됩니다.

이 관리 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/reference.html#Management_Functions

8.3. 지오메트리 생성자

ST_BdPolyFromText — MultiLineString Well-Known Text(WKT) 포맷을 입력하여 임의의 폐합된 LineString 으로 컬렉션으로 구성된 Polygon 지오메트리를 생성합니다. WKT 는 반드시 MultiLineString 이어야 하며, 결과가 MultiPolygon 이면 오류가 발생하므로 이때는 ST_BdMPolyFromText 함수를 사용하십시오.

ST_BdMPolyFromText — MultiLineString Well-Known Text(WKT) 포맷을 입력하여 임의의 폐합된 LineString 으로 컬렉션으로 구성된 MultiPolygon 지오메트리를 생성합니다. WKT 는 반드시 MultiLineString 이어야 하며, 결과가 단일 Polygon 이더라도 MultiPolygon 으로 반환합니다.

ST_GeogFromText — WKT 또는 EWKT 에서 지정된 Geography 객체를 반환합니다. 이 함수는 ST_GeographyFromText 의 별칭입니다.

ST_GeographyFromText — WKT 또는 EWKT 에서 지정된 Geography 객체를 반환합니다. WGS84 경위도(EPSG 4326) 좌표계를 가정합니다.

ST_GeogFromWKB — WKB 또는 EWKB 에서 Geography 객체를 반환합니다. SRID 가 지정되지 않으면 WGS 84 경위도(EPSG 4326) 좌표계를 기본값으로 사용합니다.

ST_GeomCollFromText — 지정된 SRID 의 컬렉션 WKT 에서 컬렉션 지오메트리를 생성합니다. SRID 가 지정되지 않은 경우 -1 을 기본값으로 사용합니다.

ST_GeomFromEWKB — EWKB 에서 지정된 ST_Geometry 값을 반환합니다.

ST_GeomFromEWKT — EWKT 에서 지정된 ST_Geometry 값을 반환합니다.

ST_GeometryFromText — WKT 에서 지정된 ST_Geometry 값을 반환합니다. 이 함수는 ST_GeomFromText 의 별칭입니다

ST_GeomFromGML — GML 지오메트리를 입력 받아 PostGIS 지오메트리 객체를 반환합니다.

ST_GeomFromGeoJSON — GeoJSON 지오메트리를 입력 받아 PostGIS 지오메트리 객체를 반환합니다.

ST_GeomFromKML — KML 지오메트리를 입력 받아 PostGIS 지오메트리 객체를 반환합니다.

ST_GMLToSQL — GML 표현에서 지정된 ST_Geometry 값을 반환합니다. 이 함수는 ST_GeomFromGML 의 별칭입니다.

ST_GeomFromText — WKT 에서 지정된 ST_Geometry 값을 반환합니다.

ST_GeomFromWKB — WKB 에서 지정된 ST_Geometry 값을 반환합니다.

ST_LineFromMultiPoint — MultiPoint 지오메트리에서 LineString 을 생성합니다.

ST_LineFromText — 지정된 SRID 및 WKT 에서 LineString 지오메트리를 생성합니다. SRID 가 지정되지 않은 경우 -1 을 기본값으로 사용합니다.

ST_LineFromWKB — 지정된 SRID 및 WKB 에서 LineString 을 생성합니다.

ST_LinestringFromWKB — 지정된 SRID 및 WKB 에서 LineString 을 생성합니다. 이 함수는 ST_LineFromWKB 의 별칭입니다.

ST_MakeBox2D — 주어진 좌하단 및 우상단 Point 지오메트리에 의해 정의된 BOX2D 를 생성합니다.

ST_3DMakeBox — 주어진 좌하단 및 우상단 3 차원 Point 지오메트리에 의해 정의된 BOX3D 를 생성합니다.

ST_MakeLine — Point 또는 Line 지오메트리를 이용하여 LineString 을 생성합니다.

ST_MakeEnvelope — 지정된 최소 및 최대값으로 구성된 직사각형의 Polygon 을 생성합니다. 입력 값은 SRID 에 의해 지정된 좌표계여야 합니다.

ST_MakePolygon — 주어진 쉘(Shell)로 구성된 Polygon 을 생성합니다. 입력 지오메트리는 닫힌 LineString 이어야 합니다.

ST_MakePoint — X, Y, Z, M 등의 값을 이용하여 2D, 3DZ 또는 4D Point 지오메트리를 생성합니다.

ST_MakePointM — XY 및 M 좌표값을 이용하여 Point 지오메트리를 생성합니다.

ST_MLineFromText — WKT 에서 지정된 ST_MultiLineString 값을 반환합니다.

ST_MPointFromText — 지정된 SRID 와 WKT 에서 지오메트리를 생성합니다. SRID 가 지정되지 않은 경우 -1 을 기본값으로 사용합니다.

ST_MPolyFromText — 지정된 SRID 와 WKT 에서 MultiPolygon 지오메트리를 생성합니다. SRID 가 지정되지 않은 경우 -1 을 기본값으로 사용합니다.

ST_Point — 주어진 좌표 값을 이용하여 ST_Point 값을 반환합니다. ST_MakePoint 의 OGC 별칭입니다.

ST_PointFromText — 지정된 SRID 와 WKT 에서 Point 지오메트리를 생성합니다. SRID 가 지정되지 않은 경우 기본값은 unknown 입니다.

ST_PointFromWKB — 지정된 SRID 의 WKB 에서 지오메트리를 생성합니다.

ST_Polygon — 지정된 LineString 및 SRID 를 이용해서 생성한 Polygon 을 반환합니다.

ST_PolygonFromText — 지정된 SRID 와 WKT 에서 지오메트리를 생성합니다. SRID 가 지정되지 않은 경우 -1 을 기본값으로 사용합니다.

ST_WKBToSQL — WKB 에서 지정된 ST_Geometry 값을 반환합니다. 이 함수는 SRID 를 사용하지 않은 ST_GeomFromWKB 의 별칭입니다.

ST_WKTToSQL — WKT 에서 지정된 ST_Geometry 값을 반환합니다. 이 함수는 ST_GeomFromText 의 별칭입니다.

이 지오메트리 생성자와 관련한 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/reference.html#Geometry_Constructors

8.4. 지오메트리 접근자

GeometryType —지오메트리의 유형을 문자열로 반환합니다. 예: 'LINESTRING', 'POLYGON', 'MULTIPOINT' 등.

ST_Boundary — 주어진 지오메트리에 대한 폐쇄된 조합 경계를 반환합니다.

ST_CoordDim — ST_Geometry 값에 대한 지오메트리의 좌표 차수(XY 는 2, XYZ 는 3 등)를 반환합니다. 이 함수는 SWL/MM 사양 중 ST_NDims 의 별칭입니다.

ST_Dimension — 좌표 차수보다 작거나 반드시 같아야 하는 이 지오메트리 객체의 고유 차원을 반환합니다. Point 는 0, LineString 은 1, Polygon 은 2 를 반환합니다.

ST_EndPoint — LineString 지오메트리의 끝점을 Point 로 반환합니다.

ST_Envelope — 제공된 지오메트리의 double precision(float8) 최소경계영역 사각형(bounding box)을 나타내는 지오메트리를 반환합니다.

ST_ExteriorRing — Polygon 지오메트리의 외부 링(Exterior Ring)을 나타내는 LineString 을 반환합니다. 지오메트리가 Polygon 이 아닌 경우 NULL 값을 반환합니다. MultiPolygon 은 작동하지 않습니다.

ST_GeometryN — 지오메트리가 GEOMETRYCOLLECTION, (MULTI) POINT, (MULTI) LINESTRING, MULTICURVE 또는 (MULTI) POLYGON, POLYHEDRALSURFACE 인 경우 처음을 1 부터 시작하는 N 번째 지오메트리를, 그렇지 않으면 NULL 값을 반환합니다.

ST_GeometryType — ST_Geometry 값의 지오메트리 유형을 반환합니다. 예: 'ST_Linestring', 'ST_Polygon', 'ST_MultiPolygon' 등. 이 함수는 GeometryType 함수와는 다릅니다.

ST_InteriorRingN — Polygon 지오메트리의 N 번째 내부 링(Interior Ring)을 LineString 으로 반환합니다. 지오메트리가 Polygon 이 아니거나 주어진 N 값이 범위를 벗어나면 NULL 값을 반환합니다.

ST_IsClosed — LineString 의 시작점과 끝점이 일치하는 경우 TRUE 를 반환합니다.

ST_IsCollection — 지오메트리가 GEOMETRYCOLLECTION, MULTI{POINT, POLYGON, LINESTRING, CURVE, SURFACE}, COMPOUNDCURVE 일 경우 TURE 를 반환합니다:

ST_IsEmpty — 지오메트리가 빈 GeometryCollection, Polygon, LineString, Point 등일 경우 TRUE 를 반환합니다.

ST_IsRing — 이 LineString 지오메트리가 닫혀(ST_IsClosed) 있고 단순(ST_IsSimple)한 경우 TRUE 를 반환합니다.

ST_IsSimple — 이 지오메트리가 자기 교차 또는 자기 접선과 같이 변칙적인 지오메트리 포인트를 가지고 있지 않은 경우 TRUE 를 반환합니다.

ST_IsValid — ST_Geometry 가 잘 구성되어 있는 경우에 TRUE 를 반환합니다.

ST_IsValidReason — 지오메트리가 유효한지 아닌지, 유효하지 않으면 그 이유는 무엇인지에 관해 설명하는 문자열을 반환합니다.

ST_IsValidDetail — 지오메트리가 유효한지 아닌지, 유효하지 않다면 그 이유는 무엇이고 위치가 어디인지에 관해 설명하는 valid_detail (valid, reason, location) 컬럼을 반환합니다.

ST_M — Point 의 M 값을 반환하거나 없는 경우에는 NULL 값을 반환합니다. 입력은 반드시 Point 이어야 합니다.

ST_NDims — 지오메트리의 좌표 차수(XY 는 2, XYZ 는 3 등)를 정수(small int) 값으로 반환합니다.

ST_NPoints — 지오메트리를 구성하는 점(정점)의 개수를 반환합니다.

ST_NRings — 지오메트리가 Polygon 또는 MultiPolygon 인 경우, 링(Ring)의 개수를 반환합니다.

ST_NumGeometries — 지오메트리가 GEOMETRYCOLLECTION (또는 MULTI *) 인 경우, 지오메트리의 수를 반환합니다. 단일 지오메트리는 1 을 반환, 없는 경우는 NULL 값을 반환합니다.

ST_NumInteriorRings — 지오메트리에서 첫 번째 다각형의 내부 링(Interior Ring)의 개수를 반환합니다. Polygon 및 MultiPolygon 두 가지 유형 모두 사용 가능하지만 지오메트리의 첫 번째 Polygon 을 사용합니다. 지오메트리에 Polygon 이 없는 경우 NULL 값을 반환합니다.

ST_NumInteriorRing — 지오메트리에서 첫 번째 다각형의 내부 링(Interior Ring)의 개수를 반환합니다. 이 함수는 ST_NumInteriorRings 의 별칭입니다.

ST_NumPatches — PolyhedralSurface 의 면(faces)의 개수를 반환합니다. Polyhedral 지오메트리가 아닌 경우 NULL 값을 반환합니다.

ST_NumPoints — ST_LineString 또는 ST_CircularString 객체의 점(정점)의 개수를 반환합니다. ST_NPoints 의 별칭이며, LineString 지오메트리가 아닌 경우 ST_NPoints 함수를 사용하십시오.

ST_PatchN — PolyhedralSurface 또는 PolyhedralSurfaceM 지오메트리인 경우 처음을 1 부터 시작하는 N 번째 지오메트리(Face)를, 그렇지 않으면 NULL 값을 반환합니다.

ST_PointN — LineString 또는 CircularLineString 지오메트리인 경우 처음을 1 부터 시작하는 N 번째 지오메트리를, 그렇지 않으면 NULL 값을 반환합니다.

ST_SRID — spatial_ref_sys 테이블에 정의된 ST_Geometry 객체의 공간 참조 식별자(SRID)를 정수값으로 반환합니다.

ST_StartPoint — LineString 지오메트리의 첫 번째 점을 Point 지오메트리로 반환합니다.

ST_Summary — 지오메트리의 정보를 문자열로 요약해서 반환합니다.

ST_X — Point 지오메트리의 X 좌표 또는 사용할 수 없는 경우 NULL 값을 반환합니다. 입력은 반드시 Point 이어야 합니다.

ST_XMax — 경계 상자 2D, 3D 또는 지오메트리의 X 최대 값을 반환합니다.

ST_XMin — 경계 상자 2D, 3D 또는 지오메트리의 X 최소 값을 반환합니다.

ST_Y — Point 지오메트리의 Y 좌표 또는 사용할 수 없는 경우 NULL 값을 반환합니다. 입력은 반드시 Point 이어야 합니다.

ST_YMax — 경계 상자 2D, 3D 또는 지오메트리의 Y 최대 값을 반환합니다.

ST_YMin — 경계 상자 2D, 3D 또는 지오메트리의 Y 최소 값을 반환합니다.

ST_Z — Point 지오메트리의 Z 좌표 또는 사용할 수 없는 경우 NULL 값을 반환합니다. 입력은 반드시 Point 이어야 합니다.

ST_ZMax — 경계 상자 2D, 3D 또는 지오메트리의 Z 최대 값을 반환합니다.

ST_Zmflag — 지오메트리의 ZM (치수 의미) 플래그 값을 정수(small int) 값으로 반환합니다. 0 = 2D, 1=3DM, 2 = 3DZ, 3 = 4D 값입니다.

ST_ZMin — 경계 상자 2D, 3D 또는 지오메트리의 Z 최소 값을 반환합니다.

이 지오메트리 접근자와 관련한 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/reference.html#Geometry_Accessors

8.5. 지오메트리 편집기

ST_AddPoint — LineString 의 Point<position>(0 기반 인덱스) 앞에 Point 를 추가합니다. position 값이 생략되면 -1 값이 기본값이며 LineString 의 마지막 위치에 Point 가 추가됩니다.

ST_Affine — 한번에 변환(translate), 회전(rotate), 크기(scale)와 같은 작업들을 수행하기 위해 지오메트리에 3 차원 아핀 변환(Affine transformation)을 적용합니다.

ST_Force_2D — 출력 지오메트리가 X 및 Y 좌표를 갖도록 "2 차원 모드"로 전환합니다.

ST_Force_3D — 지오메트리를 XYZ 모드로 전환합니다. 이 함수는 ST_Force_3DZ 의 별칭입니다.

ST_Force_3DZ — 지오메트리를 XYZ 모드로 전환합니다. 이 함수는 ST_Force_3D 의 별칭입니다

ST_Force_3DM — 지오메트리를 XYZ 모드로 전환합니다.

ST_Force_4D — 지오메트리를 XYZM 모드로 전환합니다.

ST_Force_Collection — 지오메트리를 GeometryCollection 으로 변환합니다.

ST_ForceRHR — 오른손 규칙(Right-Hand-Rule)을 따르도록 Polygon 정점의 방향을 구성합니다. Polygon 에서 외부 링(Exterior Ring)은 시계방향, 내부 링(Interior Ring)은 반 시계 방향으로 정점의 순서가 구성됩니다.

ST_LineMerge — MultiLineString 을 병합하여 LineString(또는 LineString 집합)으로 반환합니다. MultiLineString 이 병합되지 않으면 원본 MultiLineString 을 반환합니다.

ST_CollectionExtract — 주어진 (다중)지오메트리에서 지정된 유형의 요소만으로 구성된 (다중) 지오메트리를 반환합니다. 유형 매개변수에서 1 = POINT, 2 = LINESTRING, 3 = POLYGON 입니다.

ST_CollectionHomogenize — 주어진 GeometryCollection 에서 "가장 단순한(simplest)" 형태의 지오메트리(같은 유형이 여러 개일 경우 Multi-*타입)를 반환합니다.

ST_Multi — 지오메트리를 MULTI * 지오메트리로 반환합니다. 지오메트리가 이미 MULTI * 인 경우에는 변경 없이 그대로 반환합니다.

ST_RemovePoint — LineString 에서 Point 를 제거합니다. 오프셋(offset)은 0 값을 기준으로 합니다.

ST_Reverse — 정점 순서가 역으로 구성된 지오메트리를 반환합니다.

ST_Rotate — 원점 축을 기준으로 시계 반대 방향으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_RotateX — X 축을 기준으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_RotateY — Y 축을 기준으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_RotateZ — Z 축을 기준으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_Scale — 매개변수와 원본 지오메트리의 좌표값을 곱하여 새로운 크기의 지오메트리를 생성 후 반환합니다. 예: ST_Scale(geom, Xfactor, Yfactor, Zfactor).

ST_Segmentize — 주어진 거리보다 길지 않은 세그먼트를 가진 수정된 지오메트리를 반환합니다. 거리 계산은 2D 에서만 수행됩니다.

ST_SetPoint — LineString 의 N 번째 Point 를 주어진 Point 로 교체합니다. 인덱스는 0 을 기준으로 합니다.

ST_SetSRID — 지정된 정수 값(SRID 코드)으로 지오메트리의 SRID 를 설정합니다.

ST_SnapToGrid — 입력 지오메트리의 모든 점을 정규 그리드(Regular Grid, 원점과 셀 크기로 정의된 그리드)에 스냅 합니다. 이 함수는 같은 셀에 포함된 포인트는 제거하며, 지오메트리 좌표의 정밀도를 줄이는데 유용합니다.

ST_Snap — 참조 지오메트리의 정점에 입력 지오메트리의 정점 및 세그먼트를 스냅 합니다.

ST_Transform — 정수 매개변수에 의해 참조되는 SRID 값으로 좌표변환을 수행한 지오메트리를 반환합니다.

ST_Translate — 숫자 매개변수 오프셋을 사용하여 새 위치로 지오메트리를 변환하여 반환합니다. 예: ST_Translate(geom, X, Y) or ST_Translate(geom, X, Y,Z).

ST_TransScale — deltaX, deltaY 매개변수를 사용하여 지오메트리를 변환한 뒤 XFactor, YFactor 매개변수를 사용하여 크기를 조정한 지오메트리를 반환합니다. 2D 에서만 수행됩니다.

이 지오메트리 편집기와 관련한 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/reference.html#Geometry_Editors

8.6. 지오메트리 출력

ST_AsBinary — Geometry/Geography 객체를 SRID 메타데이터를 포함하지 않은 Well-Known Binary(WKB) 포맷으로 반환합니다. 두 번째 매개변수는 little-endian ('NDR') 또는 big-endian ('XDR') 인코딩 옵션입니다.

ST_AsEWKB — Geometry/Geography 객체를 SRID 메타데이터를 포함한 Well-Known Binary(WKB) 포맷으로 반환합니다. 두 번째 매개변수는 little-endian ('NDR') 또는 big-endian ('XDR') 인코딩 옵션입니다.

ST_AsEWKT — 지오메트리를 SRID 메타데이터를 포함한 Well-Known Text(WKT) 포맷으로 반환합니다.

ST_AsGeoJSON — 지오메트리를 GeoJSON(Geometry Javascript Object Notation) 포맷으로 반환합니다. 옵션 매개변수를 이용하여 GeoJSON 의 버전, CRS 표시 유형 및 소수점 최대 자리수(기본값 15) 등을 설정할 수 있습니다.

ST_AsGML — 지오메트리를 GML(Geography Markup Language) 버전 2(2.1.2, 기본값) 또는 버전 3(3.1.1) 포맷으로 반환합니다. 옵션 매개변수를 이용하여 GML 의 버전, CRS 표시 유형 및 소수점 최대 자리수(기본값 15) 등을 설정할 수 있습니다.

ST_AsHEXEWKB — 지오메트리를 little-endian (NDR) 또는 big-endian (XDR) 인코딩을 사용하여 HEXEWKB 포맷(텍스트)으로 반환합니다.

ST_AsKML — 지오메트리를 KML(Keyhole Markup Language) 포맷으로 반환합니다. 옵션 매개변수를 이용하여 GML 의 버전(기본값 버전 2), CRS 표시 유형 및 소수점 최대 자리수(기본값 15) 등을 설정할 수 있습니다.

ST_AsSVG — 주어진 Geometry 또는 Geography 객체를 SVG 경로 데이터 포맷의 지오메트리로 반환합니다.

ST_AsX3D — X3D XML 노드 요소 포맷으로 지오메트리를 반환합니다: ISO-IEC-19776-1.2-X3DEncodings-XML. 옵션 매개변수를 이용하여 유형 및 소수점 최대 자리수(기본값 15) 등을 설정할 수 있습니다.

ST_GeoHash — 지오메트리를 GeoHash(geohash.org) 포맷으로 변환합니다 옵션 매개변수를 이용하여 좌표 값의 정밀도를 설정할 수 있습니다.

ST_AsText — Geometry/Geography 객체를 SRID 메타데이터를 포함하지 않은 Well-Known Text(WKT) 포맷으로 반환합니다.

ST_AsLatLonText — 주어진 Point 지오메트리를 도, 분, 초 형식으로 반환하며, 출력 형식을 설정할 수 있습니다.

이 지오메트리 출력과 관련한 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/reference.html#Geometry_Outputs

8.7. 연산자

&& — A 의 2D 바운딩 박스와 B 의 2D 바운딩 박스가 교차하는 경우 TRUE 를 반환합니다.

&&& — A 의 3D 바운딩 박스와 B 의 3D 바운딩 박스가 교차하는 경우 TRUE 를 반환합니다.

&< — A 의 바운딩 박스가 B 의 바운딩 박스와 겹치거나 B 의 왼쪽에 있으면 TRUE 를 반환합니다.

&<| — A 의 바운딩 박스가 B 의 바운딩 박스와 겹치거나 B 의 아래에 있으면 TRUE 를 반환합니다.

&> — A 의 바운딩 박스가 B 의 바운딩 박스와 겹치거나 B 의 오른쪽에 있으면 TRUE 를 반환합니다.

<< — A 의 바운딩 박스가 정확히 B 의 바운딩 박스 왼쪽에 있으면 TRUE 를 반환합니다.

<<| — A 의 바운딩 박스가 정확히 B 의 바운딩 박스 아래에 있으면 TRUE 를 반환합니다.

= — A 의 바운딩 박스가 B 의 바운딩 박스와 일치하면 TRUE 를 반환합니다. double precision 바운딩 박스를 사용합니다.

>> — A 의 바운딩 박스가 정확히 B 의 바운딩 박스 오른쪽에 있으면 TRUE 를 반환합니다.

@ — A 의 바운딩 박스가 정확히 B 의 바운딩 박스에 포함되면 TRUE 를 반환합니다.

|&> — A 의 바운딩 박스가 B 의 바운딩 박스와 겹치거나 B 의 위 쪽에 있으면 TRUE 를 반환합니다.

|>> — A 의 바운딩 박스가 정확하게 B 의 위 쪽에 있으면 TRUE 를 반환합니다.

~ — A 의 바운딩 박스가 B 의 바운딩 박스를 포함하면 TRUE 를 반환합니다.

~= — A 의 바운딩 박스가 B 의 바운딩 박스와 일치하면 TRUE 를 반환합니다.

<-> — 두 점 사이의 거리를 반환합니다. 포인트/포인트 거리 측정에 부동 소수점 정확도(기본적인 Point 지오메트리의 double precision 에 반하는)를 사용합니다. 다른 지오메트리 유형들에 대해서는 부동 소수점 바운딩 박스 중심간의 거리가 반환됩니다. KNN gist 기능을 사용하여 최근린 검색 제한과 거리 정렬에 유용합니다.

<#> — 두 지오메트리 바운딩 박스 사이의 거리를 반환합니다. 포인트/포인트 거리 측정은 지오메트리 거리 측정과 거의 똑같습니다. (그러나 지오메트리가 double precision 이고 바운딩 박스가 부동 소수 정확도에 있기 때문에 조금 다를 수도 있습니다) KNN gist 기능을 사용하여 최근린 검색 제한과 거리 정렬에 유용합니다.

이 연산자들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/reference.html#Operators

8.8. 공간 관계와 측정

ST_3DClosestPoint — g2 와 가장 가까운 g1 의 3 차원 Point 를 반환합니다. 3 차원 공간에서 최단거리 직선의 첫 번째 Point 입니다.

ST_3DDistance — Geometry 유형에 대해서 두 지오메트리 사이의 3 차원 직교 최단거리(공간 좌표계를 기준으로 한 단위)를 반환합니다.

ST_3DDWithin — 3D(z) Geometry 유형에 대해서 두 지오메트리의 3 차원 공간에서의 거리가 지정된 거리 내에 있는 경우 TRUE 를 반환합니다.

ST_3DDFullyWithin — 모든 3D 지오메트리들이 지정된 거리 내에 있는 경우 TRUE 를 반환합니다.

ST_3DIntersects — 두 지오메트리가 3 차원 공간에서 "공간적으로 교차"하는 경우 TRUE 를 반환합니다. Point 및 LineString 만을 대상으로 합니다.

ST_3DLongestLine — 두 지오메트리간의 3 차원 공간에서 최장거리 직선을 LineString 으로 반환합니다.

ST_3DMaxDistance — Geometry 유형에 대해서 두 지오메트리 사이의 3 차원 직교 최장거리(공간 좌표계를 기준으로 한 단위)를 반환합니다.

ST_3DShortestLine — 두 지오메트리간의 3 차원 공간에서 최단거리 직선을 LineString 으로 반환합니다.

ST_Area — 지오메트리가 Polygon 또는 MultiPolygon 인 경우 표면의 면적을 반환합니다. "Geometry" 유형은 SRID 단위, "Geography" 유형은 평방 미터 단위입니다.

ST_Azimuth — pointA 에서 poinB 로 수직으로 시계 방향으로 측정한 북쪽 기반의 방위각을 라디안 단위의 각도로 반환합니다.

ST_Centroid — 지오메트리의 기하학적 중심(무게 중심)을 반환합니다.

ST_ClosestPoint — g2 와 가장 가까운 g1 의 2 차원 Point 를 반환합니다. 2 차원 공간에서 최단거리 직선의 첫 번째 Point 입니다.

ST_Contains — A 지오메트리 외부에 B 지오메트리의 그 어떤 점도 놓여있지 않거나, 적어도 B 지오메트리 내부에서 하나의 점만이 A 지오메트리 내부에 놓여 있을 때 TRUE 를 반환합니다.

ST_ContainsProperly — B 지오메트리가 A 지오메트리 내부와 교차하지만 경계선(또는 외부)과 교차하지 않을 때 TRUE 를 반환합니다.

ST_Covers — B 지오메트리의 그 어떤 점도 A 지오메트리의 외부에 있지 않을 때 TRUE 를 반환합니다.

ST_CoveredBy — A Geometry/Geography 유형 지오메트리의 그 어떤 점도 B Geometry/Geography 유형 지오메트리의 외부에 있지 않을 때 TRUE 를 반환합니다.

ST_Crosses — 제공된 지오메트리가 공통의 내부(interior) 점들을 일부(전부는 아닌) 가지고 있을 경우 TRUE 를 반환합니다.

ST_LineCrossingDirection — 주어진 2 개의 LineString 에서 어떤 종류의 교차 동작을 나타내는 -3 과 3 사이의 숫자를 반환합니다. 0 은 교차하지 않음을 의미합니다.

ST_Disjoint — 지오메트리가 서로 "공간적으로 교차"하지 않을 때, 어떤 공간도 함께 공유하지 않는다면 TRUE 를 반환합니다.

ST_Distance — Geometry 유형에 대해서 두 지오메트리 사이의 2 차원 직교 최단거리(공간 좌표계를 기준으로 한 단위)를 반환합니다. Geography 유형에 대해서는 기본적으로 미터로 표현된 두 Geography 사이의 타원체 최단거리를 반환합니다.

ST_HausdorffDistance — 두 지오메트리 사이의 Hausdorff 거리를 반환합니다. 기본적으로 두 지오메트리가 서로 얼마나 비슷한지 혹은 서로 다른지에 대한 측정입니다. 단위는 지오메트리의 공간 좌표계를 기준으로 합니다.

ST_MaxDistance — 두 지오메트리 사이의 2 차원 직교 최장거리(공간 좌표계를 기준으로 한 단위)를 반환합니다.

ST_Distance_Sphere — 두 개의 경/위도 지오메트리 사이의 최단거리를 반환합니다. 반지름 6370986 미터의 구면 좌표계를 사용하며 ST_Distance_Spheroid 함수보다 빠르나 정확성이 떨어집니다. 1.5 버전 이전 PostGIS 는 포인트만 지원합니다.

ST_Distance_Spheroid — 주어진 회전 타원체를 이용하여 두 개의 경/위도 지오메트리 사이의 최단거리를 반환합니다. 1.5 버전 이전 PostGIS 는 포인트만 지원합니다.

ST_DFullyWithin — 모든 지오메트리가 서로간에 지정된 거리 내에 있는 경우에 TRUE 를 반환합니다.

ST_DWithin —지오메트리가 서로간에 지정된 거리 내에 있는 경우에 TRUE 를 반환합니다. Geometry 유형일 경우 단위는 공간 좌표계를 기준으로 하고, Geography 일 경우 단위는 미터를 사용하며 측정은 use_spheroid=true(회전 타원체 사용)가 기본값이고, 빠른 처리를 위해서 use_spheroid=false 옵션을 사용하면 됩니다.

ST_Equals — 주어진 지오메트리가 동일한 지오메트리를 나타내는 경우에 TRUE 를 반환하고 방향성은 무시합니다.

ST_HasArc — 지오메트리 또는 지오메트리 컬렉션이 Circular String 을 포함하는 경우 TRUE 를 반환합니다.

ST_Intersects — Geometry/Geography 객체가 "2 차원 상에서 공간적으로 교차"(공간의 어느 부분을 공유)할 경우 TRUE 를 반환하고 그렇지 않을 경우(Disjoint) FALSE 를 반환합니다. Geography 유형에 대해서 0.00001 미터의 허용 오차(오차 내에 포함되면 서로 교차하는 것으로 간주)를 사용합니다.

ST_Length — 지오메트리가 LineString 또는 MultiLineString 일 경우 2 차원 상에서의 길이를 반환합니다. Geometry 유형은 공간 좌표계 단위를, Geography 유형은 미터 단위(회전 타원체 사용 기본값)를 사용합니다.

ST_Length2D — 지오메트리가 LineString 또는 MultiLineString 일 경우 2 차원 상에서의 길이를 반환합니다. 이 함수는 ST_Length 의 별칭입니다.

ST_3DLength — 지오메트리가 LineString 또는 MultiLineString 일 경우 2 차원 또는 3 차원 상에서의 길이를 반환합니다. Z 값이 있는 경우 3 차원, 그렇지 않으면 ST_Length(또는 ST_Lenght2D)와 동일합니다.

ST_Length_Spheroid — LineString 또는 MultiLineString 지오메트리의 2 차원 또는 3 차원 상에서의 길이를 타원체에 기반하여 계산합니다. 지오메트리의 좌표가 경/위도이고 투영 없이 길이를 측정하고자 할 때 유용합니다.

ST_Length2D_Spheroid — LineString 또는 MultiLineString 지오메트리의 2 차원 상에서의 길이를 타원체에 기반하여 계산합니다. 지오메트리의 좌표가 경/위도이고 투영 없이 길이를 측정하고자 할 때 유용합니다.

ST_3DLength_Spheroid — 타원체에 기반하여 고도를 고려한 지오메트리의 길이를 계산합니다. 이 함수는 ST_Length_Spheroid 의 별칭입니다.

ST_LongestLine — 두 지오메트리 간에 2 차원 공간에서의 최장 직선의 두 점을 LineString 으로 반환합니다. 두 개 이상의 선이 발견된 경우 첫 번째 직선을 반환합니다. 직선은 항상 g1 에서 시작하여 g2 에서 끝납니다. 이 함수가 반환하는 직선의 길이는 ST_MaxDistance 와 동일합니다.

ST_OrderingEquals — 주어진 지오메트리가 동일한 지오메트리(ST_Equals)를 나타내고 구성하는 점들의 순서가 동일한 경우 TRUE 를 반환합니다.

ST_Overlaps — 주어진 지오메트리가 공간을 공유하고 동일한 차원이지만 서로 완전하게 포함되지 않을 경우 TRUE 를 반환합니다.

ST_Perimeter — ST_Surface 또는 ST_MultiSurface 지오메트리(Polygon, MultiPolygon)의 둘레 길이를 반환합니다. Geometry 유형은 공간 좌표계 단위를, Geography 유형은 미터 단위를 사용합니다.

ST_Perimeter2D — 지오메트리가 Polygon 또는 MultiPolygon 일 경우 2 차원 상에서의 둘레 길이를 반환합니다. 이 함수는 현재 ST_Perimeter 의 별칭입니다.

ST_3DPerimeter — 지오메트리가 Polygon 또는 MultiPolygon 일 경우 3 차원 상에서의 둘레 길이를 반환합니다.

ST_PointOnSurface — 표면에 반드시 위치하는 Point 지오메트리를 반환합니다.

ST_Project — 미터 단위의 거리 및 라디안 단위의 베어링(방위각) 각도를 사용하여 시작 지점에서 예상되는 Point 지오메트리를 반환합니다.

ST_Relate — 지정된 intersectionMatrixPattern 값에 의한 두 지오메트리의 내부(Interior), 경계(Boundary), 외부(Exterior) 사이의 교차점 테스트를 통해 이 지오메트리가 anotherGeometry 와 공간적으로 관련이 있는 경우 TRUE 를 반환합니다. 만약 intersectionMatrixPattern 값이 지정되지 않은 경우에는 두 지오메트리와 관련된 intersectionMatrixPattern 값을 문자열로 반환합니다.

ST_RelateMatch — intersectionMattrixPattern1 이 intersectionMatrixPattern2 을 의미하는 경우 TRUE 를 반환합니다.

ST_ShortestLine — 두 지오메트리 간에 2 차원 공간에서의 최단 직선의 두 점을 LineString 으로 반환합니다.

ST_Touches — 두 지오메트리간에 최소한 하나의 공통된 포인트를 가지고 있으나 그들의 내부(Interior)들이 서로 교차하지 않는 경우 TRUE 를 반환합니다.

ST_Within — A 지오메트리가 완전하게 B 지오메트리 안에 있을 경우 TRUE 를 반환합니다.

이 공간관계와 측정에 관련한 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/reference.html#Spatial_Relationships_Measurements

8.9. 지오메트리 처리

ST_Buffer — Geometry 유형일 경우: 이 지오메트리로부터의 거리가 같거나 작은 모든 점들을 포함하는 지오메트리를 반환합니다. 계산은 지오메트리의 공간 좌표계를 따릅니다. Geography 유형일 경우: 평면 변환 래퍼를 사용합니다. 1.5 버전 이후부터 버퍼 옵션으로 끝점, 꼭지점 처리에 대한 스타일을 설정할 수 있습니다.

ST_BuildArea — 주어진 지오메트리의 모든 선을 이용하여 면적을 가진 Polygon 지오메트리를 생성합니다. 반환되는 유형은 입력 지오메트리에 따라 Polygon 또는 MultiPolygon 이며, 입력된 지오메트리가 Polygon 을 구성하지 못하면 NULL 값을 반환합니다. 이 함수는 모든 내부 지오메트리를 홀(Hole)로 처리합니다. 입력 지오메트리는 LineString, MultiLineString, Polygon, MultiPolygon, GeometryCollection 을 사용할 수 있습니다.

ST_Collect — 다른 지오메트리 집합에서 지정된 ST_Geometry 값을 반환합니다. 반환되는 유형은 Multi* 또는 GeometryCollection 지오메트리입니다.

ST_ConcaveHull — 지정된 지오메트리 집합을 모두 둘러싸는 최소 오목 폐곡선(Concave Hull)을 Polygon 지오메트리로 반환합니다.

ST_ConvexHull — 지정된 지오메트리 집합을 모두 둘러싸는 최소 볼록 폐곡선(Convex Hull)을 Polygon 지오메트리로 반환합니다.

ST_CurveToLine — CircularString/CurvedPolygon 지오메트리를 LineString/Polygon 지오메트리로 변환합니다.

ST_Difference — 지오메트리 B 와 교차하지 않는 지오메트리의 A 의 일부를 나타내는(차집합) 지오메트리를 반환합니다.

ST_Dump — g1 지오메트리를 구성하는 geometry_dump(geom, path, 지오메트리와 정수 배열로 구성된) 행의 집합으로 반환합니다. 이 함수는 집합을 반환하는 함수(Set-Returning Function - SRF)입니다.

ST_DumpPoints — 지오메트리를 구성하는 모든 점들을 geometry_dump(geom, path, Point 와 정수 배열로 구성된) 행의 집합으로 반환합니다. 이 함수는 집합을 반환하는 함수(Set-Returning Function - SRF)입니다.

ST_DumpRings — Polygon 을 구성하는 외부(Exterior) 및 내부(Interior) 링(Ring)을 geometry_dump(Polygon 과 정수 배열로 구성된) 행의 집합으로 반환합니다. 이 함수는 집합을 반환하는 함수(Set-Returning Function - SRF)입니다.

ST_FlipCoordinates — 주어진 지오메트리의 X 및 Y 축을 반전한 지오메트리를 반환합니다. 위/경도 객체를 구축해 본 적이 있거나 수정을 필요로 하는 사람들에게 유용합니다.

ST_Intersection — geomA 와 geomB 가 공유하는 부분(교집합)을 나타내는 지오메트리를 반환합니다. Geography 유형에서의 구현은 교차 수행을 위해 UTM 등으로 투영하고 처리 후 다시 WGS84 로 변환하여 반환합니다.

ST_LineToCurve — LineString/Polygon 지오메트리를 CircularString, CurvedPolygon 지오메트리로 변환합니다.

ST_MakeValid — 정점을 유지한 채 유효하지 않는 지오메트리를 유효하게 만듭니다.

ST_MemUnion — ST_Union 함수와 같은 결과를 반환하지만, 단지 메모리 친화적(더 작은 메모리와 더 많은 프로세서 시간을 사용)입니다.

ST_MinimumBoundingCircle — 주어진 지오메트리를 완전히 포함하는 가장 작은 원형 Polygon 지오메트리를 반환합니다. 기본값은 사분면 당 48 개의 세그먼트를 사용합니다.

ST_Polygonize — 주어진 지오메트리 집합의 모든 선을 이용하여 형성한 Polygon 들을 포함하는 GeometryCollection 을 생성합니다.

ST_Node — 주어진 LineString 지오메트리들로부터 생성 가능한 모든 노드를 추가한 지오메트리를 반환합니다.

ST_OffsetCurve — 입력한 선 지오메트리에서 주어진 거리와 측면에 해당하는 오프셋 선을 반환합니다. 중심 선에 대해 평행한 선을 생성하는 데 유용합니다.

ST_RemoveRepeatedPoints — 주어진 지오메트리에서 중복된 점을 제거한 지오메트리를 반환합니다.

ST_SharedPaths — 두 개의 입력 LineString/MultiLineString 지오메트리가 공유하는 경로(Path)를 포함한 집합(GeometryCollectoin)을 반환합니다.

ST_Shift_Longitude — 지오메트리의 모든 점/정점을 읽은 후, 경도 좌표가 0 보다 작은 경우 360 을 더합니다. 그 결과는 180 도 중심 지도에 그려진 데이터의 0-360 도 중심의 버전이 됩니다. 이 함수는 경위도 좌표계(WGS 84 경위도)에만 유용합니다.

ST_Simplify — 주어진 지오메트리를 Douglas-Peucker 알고리즘을 사용하여 "단순화"한 지오메트리를 반환합니다.

ST_SimplifyPreserveTopology — 주어진 지오메트리를 Douglas-Peucker 알고리즘을 사용하여 "단순화"한 지오메트리를 반환합니다. 이 함수는 유효하지 않는 파생된 지오메트리(특히 Polygon 지오메트리 단순화에서 발생할 수 있는)를 생성하지 않도록 합니다.

ST_Split — 지오메트리를 분할한 결과 지오메트리 집합(GeometryCollection)을 반환합니다.

ST_SymDifference — 지오메트리 A 와 지오메트리 B 가 교차하지 않는 부분의 지오메트리를 반환합니다. 이것을 대칭 차집합(Symmetric Difference) 이라고 합니다. 왜냐하면 ST_SymDifference (A, B) = ST_SymDifference (B, A)이기 때문입니다.

ST_Union — 입력된 지오메트리의 합집합 지오메트리를 반환하며, 결과 지오메트리는 Multi* 지오메트리, 단일 지오메트리 또는 GeometryCollection 일 수 있습니다.

ST_UnaryUnion — ST_Union 같지만 지오메트리 구성 요소 수준에서 작동합니다. 즉 ST_Union 과는 다르게 ST_UnaryUnion 은 MultiPolygon(유효하지 않은) 구성 요소 간의 경계를 합치며, GeometryCollection 구성 요소 간의 유니언을 수행합니다.

이 지오메트리 처리에 관련한 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/reference.html#Geometry_Processing

8.10. 선형 참조

ST_Line_Interpolate_Point — 선을 따라 보간된 지점을 Point 지오메트리로 반환합니다. 두 번째 매개변수는 LineString 에 대한 길이의 비율을 나타내는 부동소수점(Float8) 값이며 0 과 1 사이의 값을 사용해야 합니다. 입력 지오메트리는 반드시 LineString 이어야 합니다.

ST_Line_Locate_Point — 주어진 Point 지오메트리와 가장 가까운 LineString 상의 위치를 나타내는 0 과 1 사이의 부동 소수점(2 차원 상에서 선의 총 길이에 대한 비율)을 반환합니다.

ST_Line_Substring — 주어진 LineString 지오메트리에서 2 차원 상의 총 길이에 대한 시작 및 종료 비율에 해당하는 LineString 지오메트리를 반환합니다. 두 번째 및 세 번째 인수는 0 과 1 사이의 부동 소수점(Float8) 값입니다.

ST_LocateAlong — 지정된 측정치(Measure 및 Offset)와 일치하는 요소로 파생된 지오메트리 컬렉션 값을 반환합니다. Polygon 지오메트리 유형은 지원하지 않습니다.

ST_LocateBetween — 포괄적으로 지정된 범위의 측정치(Measure 및 Offset)와 일치하는 요소로 파생된 지오메트리 컬렉션 값을 반환합니다. Polygon 지오메트리 유형은 지원하지 않습니다.

ST_LocateBetweenElevations — 포괄적으로 지정된 범위의 고도와 교차하는 요소들을 가진 파생된 지오메트리(컬렉션) 값을 반환합니다. 3D, 4D LineString 및 MultiLineString 지오메트리만 지원합니다.

ST_InterpolatePoint — 지정된 점과 가까운 LineString 지오메트리 지점의 측정(M) 값을 반환합니다.

ST_AddMeasure — 시작점과 끝점 사이에 선형 보간 처리된 측정 요소로 파생된 지오메트리를 반환합니다. 지오메트리가 측정치를 가지지 않을 경우, 하나가 추가됩니다. 지오메트리가

측정치를 가지고 있을 경우 새 값으로 덮어씁니다. LineString 및 MultiLineString 지오메트리만 지원합니다.

이 선형 참조와 관련한 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/reference.html#Linear_Referencing

8.11. 장기 트랜잭션 지원

AddAuth — 현재의 트랜잭션에 사용되는 인증 토큰을 추가합니다.

CheckAuth — 인증 토큰 기반으로 행의 갱신 및 삭제를 방지/허용하는 테이블에 대해 트리거를 생성합니다.

DisableLongTransactions — 장기 트랜잭션 지원을 비활성화합니다. 이 함수는 장기 트랜잭션 지원 메타데이터 테이블을 제거하고, 잠금이 체크된 테이블과 연결된 모든 트리거를 삭제합니다.

EnableLongTransactions — 장기 트랜잭션 지원을 활성화합니다. 이 함수는 필요한 메데이터 테이블을 생성하며, 이 섹션의 다른 함수를 사용하기 전에 한번 호출해야 합니다. 호출을 두 번 해도 상관없습니다.

LockRow — 테이블의 특정 행에 대한 잠금/인증을 설정합니다.

UnlockRows — 지정된 인증 ID 가 가진 모든 잠금을 제거합니다. 해제된 잠금 수를 반환합니다.
이 모듈과 관련된 pl/pgsql 함수는 OGC Web Feature Service 표준에 필요한 장기 잠금 지원을 제공하기 위해 구현되었습니다.

사용자는 직렬화 가능 트랜잭션 레벨(serializable transaction level)을 사용해야 하며, 그렇지 않으면 잠금 메커니즘이 파손될 수 있습니다.

이 장기 트랜잭션과 관련한 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/reference.html#Long_Transactions_Support

8.12. 기타 함수

ST_Accum — 집계. 지오메트리의 배열을 구축합니다.

Box2D — 지오메트리의 최대 범위를 나타내는 BOX2D 를 반환합니다.

Box3D — 지오메트리의 최대 범위를 나타내는 BOX3D 를 반환합니다.

ST_Estimated_Extent — 주어진 공간 테이블의 '추정'된 공감범위를 반환합니다. 추정은 지오메트리 컬럼의 통계정보에서 가져온 것입니다. 지정하지 않으면 현재 스키마가 사용됩니다.

ST_Expand — 입력된 지오메트리의 바운딩 박스에서 모든 방향으로 확장된 바운딩 박스를 반환합니다. double precision 를 사용합니다.

ST_Extent — 지오메트리 행의 영역을 포함하는 바운딩 박스를 반환하는 집계 함수입니다.

ST_3DExtent — 행의 영역을 제한하는 Box3D 바운딩 박스를 반환하는 집계 함수입니다.

Find_SRID — 함수의 구문은 Find_SRID(<db/schema>, <table>, <column>) 이며, 해당 함수는 GEOMETRY_COLUMNS 테이블을 검색하여 지정된 컬럼의 정수 SRID 를 반환합니다.

ST_Mem_Size — 지오메트리가 차지하는 총 메모리 공간의 크기(바이트 단위)를 반환합니다.

ST_Point_Inside_Circle — 주어진 Point 지오메트리가 중심점 x, y 및 반경으로 구성된 원 안에 포함되면 TRUE 를 반환합니다.

기타 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/reference.html#Miscellaneous_Functions

8.13. 예외적인 함수

다음 함수들은 거의 사용되지 않으며 데이터가 어떤 식으로 손상된 경우에만 사용되어야 합니다. 이 함수들은 손상된 문제들을 해결하고 정상적인 상황에서는 절대 발생하지 않는 비정상적인 상황의 문제들을 해결하는데 사용합니다.

PostGIS_AddBBox — 지오메트리에 바운딩 박스 캐시를 추가합니다. 바운딩 박스 기반의 조회 성능을 높일 수 있으나 지오메트리 저장 공간을 증가시킵니다.

PostGIS_DropBBox — 지오메트리에서 바운딩 박스 캐시를 삭제합니다.

PostGIS_HasBBox — 지오메트리의 바운딩 박스가 캐시되어 있다면 TRUE 를, 그렇지 않으면 FALSE 를 반환합니다.

예외적인 함수들의 Synopsi, 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/reference.html#Exceptional_Functions

Chapter 9.Raster Reference

아래의 함수는 PostGIS 래스터(편집자 주: 래스터 데이터란 격자형 형태의 공간데이터)기능을 사용하는 사용자가 필요로하는 것입니다 현재 PostGIS 기능에서는 래스터기능을 추가 사용할 수 있습니다. 이 기능은 이전의 postgis 일반 사용자들에게는 다소 생소한 래스터 데이터를 처리하기 위한 기능들이 추가 된 것입니다. 이 기능을 통하여 공간데이터를 래스터 데이터 형태로 저장하고 분석이 가능합니다.

앞서서 래스터 파일에서 postgis 래스터 데이터 베이스로드하기 위한 5.1 절 내용을 여러분은 기억하실것입니다.

이 장에서는 실습을 위해 래스터 테이블을 덤프 된 SQL 코드로 생성하여 사용하게 될 것입니다. –

(다음 코드를 통해 생성합니다.)

```
CREATE TABLE dummy_rast(rid integer, rast raster);
INSERT INTO dummy_rast(rid, rast)
VALUES (1,
('01' -- little endian (uint8 ndr)
||
'0000' -- version (uint16 0)
||
'0000' -- nBands (uint16 0)
||
'0000000000000040' -- scaleX (float64 2)
||
'0000000000000840' -- scaleY (float64 3)
||
'000000000000E03F' -- ipX (float64 0.5)
||
'000000000000E03F' -- ipY (float64 0.5)
||
'0000000000000000' -- skewX (float64 0)
||
'0000000000000000' -- skewY (float64 0)
||
'00000000' -- SRID (int32 0)
||
'0A00' -- width (uint16 10)
||
'1400' -- height (uint16 20)
)::raster
),
```

```
-- Raster: 5 x 5 pixels, 3 bands, PT_8BUI pixel type, NODATA = 0
(2,  ('01000003009A9999999999A93F9A9999999999A9BF000000E02B274A' ||
'4100000000771956410000000000000000000000000000000000FFFFFFFF050005000400FD
FEFDFEFEFDFEFEFDF9FAFEF' ||
'EFCF9FBFDFEFEFDFCFAFEFEFE04004E627AADD16076B4F9FE6370A9F5FE59637AB0E54F5
8617087040046566487A1506CA2E3FA5A6CAFFBFE4D566DA4CB3E454C5665')::raster);
```

9.1. 래스터 지원 데이터 유형들

이 장에서는 PostgreSQL 의 지원 하는 래스터 데이터 유형들을 설명 합니다.

geomval — 원본 공간도형객체를 정의하는 두 개의 필드와 공간 데이터 형태를 제공합니다. –공간 객체 (원본 Geometry 개체)과 픽셀 변수(래스터 밴드필드에 저장된 Double 변수형의 픽셀 값)

histogram — 히스토그램 정보는 ST_Histogram 및 ST_ApproxHistogram 함수를 실행하여 알아낼 수 있습니다.

raster — 래스터 공간 데이터. 격자형 이미지 데이터

reclassarg — 래스터 데이터의 레이어 픽셀 값을 재분류를 하기 위해 ST_Reclass 함수사용됩니다.(편집자 주: 벡터 데이터를 사용자 정의에 의해 데이터구간을 나누어 지도에 표현 하는 것과 같음)

summarystats — 래스터 데이터의 요약정보는 ST_SummaryStats 함수를 통하여 알 수 있습니다.

래스터 지원 데이터 유형들의 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#Raster_Types

9.2. 래스터 데이터 관리

AddRasterConstraints — 공간 참조정보, 확장정보, 크기, 정렬, 밴드, 밴드 타입 및 래스터 에 대하여 데이터 무결성 확보를 위해 제약 조건을 추가합니다. 이 경우 반드시 제약조건이 래스터 데이터 테이블에 반영되어 나타나야 합니다. 제약조건이 반영되면 true 를 반환합니다.

DropRasterConstraints — 래스터 테이블 열을 참조하는 PostGIS 래스터 제약 조건을 삭제합니다. 데이터를 다시 로드하거나 래스터 열 데이터를 업데이트 할 시 유용합니다.

PostGIS_Raster_Lib_Build_Date — postgis 에서 구동하는 래스터 라이브러리가 구축된 날짜를 조회 합니다.

PostGIS_Raster_Lib_Version — postgis 에서 구동하는 래스터 라이브러리 버전 및 빌드 구성 정보를 조회 합니다,

ST_GDALDrivers — postgis 가 설치된 경로의 lib 디렉토리 내의 GDAL 라이브러리 에서 지원하는 래스터 데이터 종류를 보여줍니다. 조회되는 정보는 ST_AsGDALRaster 기능을 사용하여도 확인가능합니다.

래스터 데이터 관리에 관련한 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#Raster_Management_Functions

9.3. 래스터 생성자

ST_AddBand — 지정된 인덱스 위치에 지정된 초기 값으로 추가 지정된 형식의 새 밴드 (들)과 래스터를 반환합니다. 인덱스를 지정하지 않은 경우, 새롭게 추가된 밴드는 데이터 테이블의 레코드 끝에 추가됩니다.

ST_AsRaster — PostGIS geometry(편집자주: 벡터형태의 공간도형)를 PostGIS raster 로 변환합니다.

ST_Band — 하나 또는 그 이상의 기존 래스터 밴드(들)를 새로운 래스터데이터로 변환되어 재구성합니다. 기존에 존재하는 래스터정보들을 재구성하여 새로운 래스터로 구축하는데 유용합니다.

ST_MakeEmptyRaster — 좌상단 X 및 Y 픽셀 크기 및 이미지 내부의 화소 상의 기울기 (scaleX, scaley, skewx & skewy) 및 좌표참조 시스템 (SRID 가 부여된 좌표계정보), 주어진 치수(폭 및 높이) 등 기본적인 래스터 데이터를 구성할 수 있는 데이터를 입력하면 새로운 Raster 데이터 레이어가 postgis 에 생성됩니다. SRID 가 생략되는 경우, 공간 좌표참조 값은 기본값 (0)으로 설정됩니다.(편집자 주: 이것은 실제 데이터가 없는 래스터 레이어를 초기 생성하기 위한 기능입니다.)

래스터 생성자 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#Raster_Constructors

9.4. 래스터 접근자

ST_GeoReference 일반적으로 세계 지도파일(world file)에서 볼 수 있는 것처럼 GDAL 또는 ESRI 형식의 공간 참조 메타 데이터를 조회하여 반환합니다. 기본값은 GDAL 입니다.

ST_Height — 픽셀 래스터의 높이를 조회하여 반환합니다.

ST_MetaData — 왼쪽 아래, 픽셀 크기, 회전 (기울이기), 상부와 같은 래스터 오브젝트에 대한 기본 메타 데이터 등을 반환

ST_NumBands — 래스터 객체를 구성하고 있는 밴드의 수를 반환.

ST_PixelHeight — 좌표를 참조하고 있는 래스터 단위 픽셀의 높이를 반환합니다

ST_PixelWidth — 좌표를 참조하고 있는 래스터 단위 픽셀 너비를 반환합니다.

ST_ScaleX — 좌표를 참조하고 있는 단위 픽셀 너비의 X 구성 요소를 반환합니다

ST_ScaleY — 좌표를 참조하고 있는 단위 픽셀 높이의 Y 구성 요소를 반환합니다.

ST_Raster2WorldCoordX — 래스터, 열 및 행의 기하학적 X 좌표 반환합니다. 열 및 행의 번호는 1 부터 시작합니다.

ST_Raster2WorldCoordY — 래스터, 열 및 행의 기하학적 Y 좌표 반환합니다. 열 및 행의 번호는 1 부터 시작합니다.

ST_Rotation — 라디안 단위의 래스터 이미지의 회전각을 반환 합니다.

ST_SkewX — (또는 회전 매개 변수) X 왜곡 지리 참조(georeference)를 반환합니다.

ST_SkewY — (또는 회전 매개 변수) Y 왜곡 지리 참조(georeference)를 반환합니다.

ST_SRID — 래스터 데이터 가 참조하고 있는 좌표계 코드를 반환 합니다.

ST_UpperLeftX — 투영 된 래스터 데이터의 상단 좌측 X 좌표를 반환합니다.

ST_UpperLeftY — 투영 된 래스터 데이터의 왼쪽 상단 Y 좌표를 반환합니다..

ST_Width — 래스터의 픽셀 너비를 반환 합니다.

ST_World2RasterCoordX —래스터 데이터 가로 행에 해당하는 X 좌표값을 래스터의 세계 공간 참조 시스템(world spatial reference system)형태로 조회되어 표시됩니다.

ST_World2RasterCoordY — 래스터 데이터 세로 열에 해당하는 Y 좌표값을 래스터의 세계 공간 참조 시스템(world spatial reference system)형태로 조회되어 표시됩니다.

ST_IsEmpty — 래스터가 데이터 값의 유효성 검사 시 데이터가 없는경우 true 를 반환합니다. 그렇지 않으면 false 는 반환합니다.

래스터 접근자 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#Raster_Accessors

9.5. 래스터 밴드 접근자

ST_BandMetaData — 특정 래스터 밴드에 대한 기본 메타 데이터를 반환합니다. 아무것도-지정되지 않은 경우 대역 숫자 1 로 간주됩니다.

ST_BandNoDataValue — 아무런 데이터도 나타내지 않는 특정 래스터 밴드 값을 반환합니다. 밴드가 없다면 숫자 1 로 간주합니다.

ST_BandIsNoData — 밴드가 오직 nodate 값으로만 채워진 경우 true 를 반환합니다.

ST_BandPath — 파일 시스템에 저장된 밴드 정보를 담고 있는 파일 경로를 반환합니다. 더 band 넘버가 지정되어 있지 않으면 1 로 간주됩니다.

ST_BandPixelType — 지정된 밴드의 픽셀 형식을 반환합니다. band 넘버가 지정되어 있지 않으면 1 로 간주됩니다.

ST_HasNoBand — 주어진 밴드 수를 가진 밴드가 없는 경우에 true 를 반환합니다. 아무 밴드 번호가 지정되지 않은 경우, 밴드 번호 1 로 간주됩니다.

래스터 밴드 접근자 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#RasterBand_Accessors

9.6. 래스터 픽셀 접근자 및 설정

ST_PixelAsPolygon — 특정 행과 열의 픽셀을 제약하는 래스터 값을 지오메트리 정보형태로 반환합니다.

ST_PixelAsPolygons — 각 픽셀의 X 와 Y 래스터 좌표, 값과 함께 래스터 밴드의 모든 픽셀을 제약하는 지오메트리를 반환합니다.

ST_Value — 주어진 행 X, 열 y 에서 픽셀 또는 특정 기하학적 점에서의 특정 대역의 값을 반환합니다. 밴드 번호는 1 에서 시작하여 지정되지 않은 경우 1 로 간주합니다. exclude_nodata_value 가 false 로 설정되어있는 경우, 모든 픽셀 (NODATA 포함)의 픽셀 값은 교차하고 값을 반환하는 것으로 간주됩니다. 조건을 제약 nodata 가 전달되지 않은 경우, 래스터의 메타 데이터에서 읽습니다.

ST_SetValue — 주어진 COLUMNX, Rowy 픽셀 또는 특정 점을 교차하는 픽셀의 지정된 밴드의 값을 설정으로 함으로써 수정되어있는 래스터데이터를 반환합니다. 밴드 번호는 1 에서 시작하며 지정되지 않은 경우 1 로 간주됩니다.

래스터 픽셀 접근자 및 설정 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#Raster_Pixel_Accessors

9.7. 래스터 편집기

ST_SetGeoReference — 단일한 단위의 지리 참조번호형식인 6 지리 매개 변수를 설정합니다. 번호는 공백으로 구분됩니다. GDAL 또는 ESRI 데이터형식의 입력유형을 받아들입니다. 기본값은 GDAL 입니다.

ST_SetRotation — 래스터 이미지 픽셀의 회전 값을 라디안 단위로 설정합니다.

ST_SetScale — 좌표 참조 시스템의 단위로 X 와 픽셀의 Y 크기를 설정합니다. 단위 / 픽셀 너비 / 높이 번호를 매깁니다.

ST_SetSkew — 지리 참조 X (또는 회전 매개 변수) 그리고 Y 기울기를 설정합니다. 하나가 전달되는 경우, 같은 값으로 X 와 Y 를 설정합니다.

ST_SetSRID — spatial_ref_sys(편집자주: 공간좌표참조) 테이블에 정의된 특정 정수로 래스터의 SRID 를 설정합니다.

ST_SetUpperLeft — 예상 X 및 Y 좌표로 픽셀의 좌상단 의 값을 설정합니다.

ST_Resample — 특정 리샘플링 알고리즘을이용하여 원본의 래스터를 재구성하여 생성함.

최근린 내삽법 혹은 Bilinear, Cubic, CubicSpline 또는 Lanczos 같은 알고리즘을 선택하여 래스터 데이터를 재분류하여 구성 합니다. 초기 값은 최근린 내삽법 (NearestNeighbor.)으로 설정 되어 있습니다.

ST_Rescale — 원본 픽셀의 축척를 조절하여 래스터를 재분류. 새로운 픽셀 값은 최근린 내삽법 (NearestNeighbor.), 이중 선형, 큐빅, CubicSpline 또는 Lanczos 리샘플링 알고리즘을 사용하여 계산됩니다. 기본값은 NearestNeighbor 입니다.

ST_Reskew — 원본 픽셀의 기울기를 조정하여 래스터를 재분류. 새로운 픽셀 값은 최근린 내삽법 (NearestNeighbor.), 이중 선형, 큐빅, CubicSpline 또는 란초스 리샘플링 알고리즘을 사용하여 계산됩니다. 기본값은 NearestNeighbor 입니다.

ST_SnapToGrid — 격자형데이터 영역을 선택하여 선택 된 래스터영역에 대하여 리샘플링 합니다. 새로운 픽셀 값은 NearestNeighbor (영어 또는 미국 철자), 이중 선형, 큐빅, CubicSpline 또는 Lanczos 리샘플링 알고리즘을 사용하여 계산됩니다. 기본값은 NearestNeighbor 입니다.

ST_Transform — 원본 래스터 데이터가 갖고 있는 좌표계를 다른좌표계로 투영하여 래스터데이터의 공간적위치를 변환 합니다.

래스터 편집기 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#Raster_Editors

9.8. 래스터 밴드 편집기

ST_SetBandNoDataValue — no data 값을 다른 밴드값으로 치환하여 설정합니다.

ST_SetBandIsNoData — nodata 여부를 설정하여 밴드 값을 구성한다.

래스터 밴드 편집기 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#RasterBand_Editors

9.9. 래스터 밴드 통계 및 분석

ST_Count — 래스터 또는 래스터 범위의 지정된 밴드의 픽셀 수를 반환합니다. 아무 밴드가 지정되지 않았다면 기본값은 1 입니다. 미리 정의되어 있는 밴드 정보가 true 로 설정되어 있는 NODATA 값과 같지 않은 픽셀만 계산합니다.

ST_Histogram — 래스터 또는 래스터 밴드 그룹 범위를 요약 히스토그램의 집합을 반환합니다. 지정하지 않으면 그룹의 수가 자동으로 계산됩니다.

ST_Quantile —예를 들어 인구의 통계를 래스터 데이터로표현한다면, 래스터 또는 래스터 테이블의 범위에 대한 분위수를 계산합니다. 따라서, 값은 래스터의 25 %, 50 %, 75 %의 백분위로 검사할 수 있습니다.

ST_SummaryStats — 래스터 범위 혹은 래스터의 주어진 래스터 밴드에 대한 수, 합, 평균, 분산, 최소값, 최대값으로 구성되는 요약 통계를 반환합니다. 밴드 1 은 더 밴드가 지정되지 않은 것입니다 가정합니다.

ST_ValueCount —픽셀 값 밴드를 포함하는 레코드의 집합을 변환하고 값들의 주어진 집합을 가진 래스터의 주어진 밴드의 픽셀 수를 계산합니다. 아무런 밴드가 지정되지 않을 경우 디폴트는 밴드 1. 기본적으로 NODATA 값 픽셀은 계산되지 않습니다. 그리고 픽셀의 다른 모든 값은 출력이며 픽셀 밴드 값은 가장 가까운 정수로 반올림됩니다.

래스터 밴드 통계 및 분석과 관련한 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#RasterBand_Stats

9.10. 래스터 출력

ST_AsBinary — SRID 메타 데이터가 없는 래스터의 바이너리 정보 (WKB: WELL KNOWN binary 형식)를 반환하여 나타냅니다.

ST_AsGDALRaster — 지정된 GDAL 래스터 형식으로 타일형태의 래스터정보를 변환합니다. 타일형태 래스터 형식도 컴파일 된 라이브러리에서 지원하는 형식 중 하나입니다. 본인의 라이브러리에서 지원하는 형식 목록을 얻기 위해 ST_GDALRasters()를 사용하십시오.

ST_AsJPEG — JPEG 이미지 (바이트 배열)로 래스터 타일을 선택 밴드를 변환합니다. 아무런 밴드가 지정되지 않고 1 개 또는 3 개 이상 대역의 경우, 첫 번째 밴드가 사용됩니다. 만약 오직 3 개의 밴드일 경우 3 개의 밴드가 모두 사용되며 변환된 이미지는 RGB 값을 가진 컬러이미지 입니다.

ST_AsPNG — PNG 이미지 (바이트 배열)로 래스터 타일 선택된 밴드를 변환합니다. 래스터의 1, 3 또는 4 밴드가 지정되고 아무런 밴드가 지정되지 않으며, 모든 밴드가 사용됩니다. 만약 2 또는 4 개이상의 밴드가 지정되거나 아무런 밴드가 지정되지 않으면 밴드 1 만이 오직 사용됩니다. 밴드 RGB 또는 RGBA 형태로 이미지가 분류 됩니다.

ST_AsTIFF —래스터 선택된 밴드를 TIFF 이미지 (바이트 배열)로서 변환합니다. 아무런 밴드도 지정되지 않는다면 모든 밴드 정보를 이용합니다.

래스터 출력과 관련한 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#Raster_Outputs

9.11. 래스터 처리

Box3D — 래스터의 폐합 된 도형정보를 3 차원 데이터 형태로 변환합니다

ST_Clip — 공간연산작업을 이용하여 입력된 벡터 도형공간 의해 절취하여 (편집자 주: 클리핑처리- 하나의 래스터를 일부 벡터도형이 점유하고 있는 경우 공동점유되는 교집합부분으로 잘라 냄) 래스터를 변환합니다. 아무런 밴드도 지정되지 않았을 경우 모든 밴드가 반환됩니다. 만일 생성된래스터가 지정되어 있지않는 경우, true 는 출력래스터가 절취되었다고 의미하는 것으로 간주됩니다

ST_ConvexHull — 래스터(점 선 면 등 가장 바깥)의 외곽선을 추출하여 변환합니다.. 결과물은 볼록다각형과 같은 형태입니다. 규칙 적인 모양을지니거나 경사가 없는 래스터들의 경우 ST_Envelope 와 같은 결과를 줄 것입니다, 그러므로 비규칙적이거나 경사가 있는 래스터들의 경우에만 유용합니다.

ST_DumpAsPolygons — 주어진 래스터 밴드, geomval (기하, 발)의 행 집합을 반환합니다. 아무런 밴드 넘버가 지정되지 않은 경우 밴드 숫자 디폴트는 1 입니다.

ST_Envelope — 래스터 픽셀 단위의 좌표범위를 반환합니다.

ST_HillShade — 래스터의 음영기복을 나타냅니다.. 지형을 시각화하는 데 유용합니다.

ST_Aspect — 고도 래스터 밴드의 지형의 방향성정보를 반환합니다. 지형을 분석하는 데 유용합니다.

ST_Slope — 고도 래스터 밴드의 지형 경사정보를 반환합니다. 지형을 분석하는 데 유용합니다.

ST_Intersection — 래스터 또는 두개의 래스터가 공유하는 부분 또는 래스터 및 지오메트리의벡터화의 기하하적 교차를 나타내는 지오메트리 pixelvalue 쌍의 집합을 반환합니다.

ST_MapAlgebraExpr — 1 래스터 밴드 버전: 제공된 입력 래스터 밴드와 pixeltype 의 유효한 PostgreSQL 의 대수 연산을 적용하여 형성된 밴드 값을 갖는 신규 Raster 를 생성합니다. 아무런 밴드가 지정되지 않은 경우 대역 1 로 간주됩니다.

ST_MapAlgebraExpr — 2 래스터 밴드 버전: 제공되는 두 개의 입력 래스터 밴드에와 pixeltype 의 유효한 PostgreSQL 의 대수 연산을 적용하여 밴드 값을 갖는 신규 Raster 를 생성합니다 합니다. 아무 밴드 번호가 지정되지 않은 경우, 각 래스터 밴드 1 로 간주됩니다. 결과로 초래된 래스터는 첫번째 래스터에 의해 정의된 그리드(눈금, 기울기 및 픽셀 코너)에 의해 정렬됩니다. 그리고 “extenttype” 파라미터에 의해 정의된 범위를 가집니다. "extenttype"를 위한 값들은 다음과 같을 수 있습니다: INTERSECTION, UNION, FIRST, SECOND.

ST_MapAlgebraFct — 1 밴드 버전- 입력 래스터 밴드와 제공된 pixeltype 이 유효한 PostgreSQL 의 기능을 적용하여 형성된 새로운 밴드 Raster 를 작성합니다. 어떤 밴드가 지정되지 않은 경우 대역 1 로 간주됩니다.

ST_MapAlgebraFct — 2 밴드 버전- 2 입력 래스터 밴드와 제공된 pixeltype 의 유효한 PostgreSQL 의 기능을 적용하여 형성된 새로운 밴드 래스터 prodived 생성. 어떤 밴드가 지정되지 않은 경우 대역 1 로 간주됩니다. 지정되지 않은 경우 범위 유형(Extent type) 디폴트는 INTERSECTION.

ST_MapAlgebraFctNgb — 1 밴드 버전: 사용자 정의 PostgreSQL 의 기능을 사용하여 최근접 밀집 대수를 보여줍니다. 입력 래스터 밴드로부터 이웃의 값들을 포함하는 PLPGSQL 사용자 함수의 결과는 평가하는 래스터 반환합니다.

ST_Polygon — 아무 데이터 값이 아닌 픽셀 값을 가진 픽셀의 조합으로 형성된 다각형 지오메트리를 반환합니다. 1 에 아무 밴드 번호가 지정되지 않은 경우, 밴드 숫자 기본값으로 사용합니다.

ST_Reclass — 원본으로부터 재분류된 밴드 형식으로 구성된 새로운 래스터. Nband 는 변경되는밴드입니다. nband 는 지정되지 않았을 경우 1 로 간주됩니다. 모든 다른 밴드들은 변하지 않은 상태에서 반환됩니다. 케이스 사용: 16BUI band 를 8BUI 로 전환 그리고 볼 수 있는 형식으로 간단하게 렌더링합니다.

ST_Union — 1 밴드로 구성된 하나의 래스터로 래스터 타일의 집합의 합집합을 반환합니다. 그 어떤 밴드 unioning 을 지정하지 않은 경우, 대역 숫자 1 로 간주됩니다. 결과로 초래된 래스터의 범위는 전체 집합의 범위이다. 교차로의 경우, 결과 값은 다음 중 하나인 p_expression 에 의해 정의됩니다: LAST – 디폴트 아무것도 지정되지 않았을 경우, MEAN, SUM, FIRST, MAX, MIN.

래스터 처리와 관련한 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#Raster_Processing

9.12. 래스터 처리 내장 함수

ST_Min4ma — 이웃의 최소 픽셀 값을 계산하는 래스터 처리 기능.

ST_Max4ma — 이웃의 최대 픽셀 값을 계산 래스터 처리 기능.

ST_Sum4ma — 이웃의 모든 픽셀 값들의 합을 계산하는 래스터 처리 기능.

ST_Mean4ma — 이웃의 픽셀 값들의 평균을 계산하는 래스터 처리 기능.

ST_Range4ma — 이웃의 픽셀 값들의 범위을 계산하는 래스터 처리 기능.

ST_Distinct4ma — 이웃의 고유 픽셀 값들의 수를 계산하는 래스터 처리 기능.

ST_StdDev4ma — 이웃의 픽셀 값들의 표준편차를 계산하는 래스터 처리 기능.

래스터 처리 내장 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#Raster_Processing_Builtin_Functions

9.13. 래스터 연산자들

&& — A 의 바운딩박스가 B 의 바운딩박스가 겹쳐질 때 TRUE 가 반환

&< — A 의 바운딩박스가 B 의 바운딩박스 왼쪽에 있을 때 TRUE 가 반환

&> — A 의 바운딩박스가 B 의 바운딩박스가 오른쪽에 있을 때 TRUE 가 반환

래스터 연산자들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/RT_reference.html#RT_Operators

9.14. 래스터 및 래스터 밴드 공간적 관계

ST_Intersects — 래스터 공간적으로 분리 된 래스터 또는 지오메트리를 교차하는 경우 true 를 반환합니다. 밴드 수가 (또는 NULL 로 설정)를 없으면, 래스터의 기복 형태를 두고 기능이 실행됩니다. 대역 번호가 제공되는 경우, 값 (NODATA 아니라)을 가진 오직 다른 픽셀들이 대상정보로 간주되어 실행 됩니다.

ST_SameAlignment — 만약 래스터밴드가 같은 경사, 축척, 공간적 좌표계를 가질 때 true 반환합니다. 같은 기울기, 축척, 공간적 좌표계를 가지지 않을 때 false 값를 반환합니다. 래스터 데이터에 대하여 이와 같은 내용을 시스템 상에 표출 합니다.

래스터 및 래스터 밴드 공간적 관계와 관련한 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오 : http://postgis.net/docs/manual-2.0/RT_reference.html#Raster_Relationships

Chapter 10. PostGIS 래스터 자주 묻는 질문

1. PostGIS 래스터 프로젝트에 관한 더 많은 내용을 어디서 찾을 수 있습니까?

PostGIS Raster 홈페이지를 참조하십시오.

2. 이 소프트웨어 기능을 익히기 위한 책 또는 사용지침서가 없습니까?

PostGIS 에서 래스터 기능과 벡터 기능을 모두 공통적으로 사용하는 초보자를 위한 사용지침서가 있습니다. Postgis 개발자인 조지는 래스터 데이터를 로드하여 오라클 GeoRaster 에서 동일한 기능과 비교하는 것에 대한 블로그 기사 시리즈를 연재하고 있습니다. 연재내용 확인을 위해 조지의 PostGIS 래스터/오라클 GeoRaster 시리즈를 확인하십시오. 프리 코드와 데이터 다운로드에 관련한 내용으로 구성된 전체 35 개 이상의 페이지로 이 구성된 PostGIS in Action - Raster chapter 에서 이용가능합니다. 메닝(manning)에서 하드 카피로 된 "PostGIS in Auction" 서적을 구입할 수 있습니다(대량 구매 시 상당한 할인이 제공됩니다). 혹은 그냥 E-book 포맷도(pdf 파일) 구입 가능합니다. 아마존 혹은 다른 서적 배급자들로에서도 구입가능 합니다. 모든 하드 카피 서적들은 E-book 버전을 다운받기 위한 프리쿠폰이 함께 제공됩니다.

PostGIS Raster 유저들로부터의 리뷰는 PostGIS raster applied to land classification urban forestry 에서 찾으실 수 있습니다.

3. 어떻게 PostGIS 데이터베이스에 래스터 기능을 설치할 수 있습니까?

윈도우와 Mac OSX 에서 PostGIS 와 래스터 기능을 구동하기 위하여 최신버전의 바이너리 형태의 실행파일을 다운받습니다. 그 후에 먼저 작업 PostGIS 2.0.0 이상이 필요 PostgreSQL 를 8.4, 9.0 또는 9.1 로 실행해야합니다. PostGIS 2.0 에서 PostGIS Raste 는 완전히 통합되므로 PostGIS 자체적으로 컴파일 됩니다.

윈도우 체제에서 인스톨 하고 실행하는 것에 대한 설명은 How to Install and Configure PostGIS raster on windows 를 참조하십시오.

윈도우 OS 를 사용중이시라면 Stand alone 환경 하에서 컴파일 할 수 있습니다. 혹은 pre-compiled PostGIS windows binaries 를 사용하십시오. Mac OSX 레오파드 또는 스노우 레오파드를 사용 중이시라면 이용 가능한 바이너리는 Kyng Chaos Mac OSX PostgreSQL/GIS binaries 입니다.

데이터베이스에서 래스터 데이터 기능을 사용하기 위해서는 설치된 데이터베이스 에서 tpostgis.sql 파일을 실행하십시오. 현존하는 인스톨을 업그레이드하기 위해서는 rtpostgis.sql 대신 rtpostgis_upgrade_minor..sql 사용하십시오.

다른 플랫폼들의 경우에서도 일반적으로 컴파일을 수행 하셔야 합니다. 컴파일 시 모듈 종속성 해결을 위해 PostGIS 그리고 GDAL 모듈 경로를 설정합니다. 소스 컴파일링하는 것에 대한 보다 많은 정보를 위해서는 Installing PostGIS Raster from source (PostGIS 의 이전 버전에서)을 참조하십시오.

4. 라이브러리 "C:/Program Files/PostgreSQL/8.4/lib/rtpostgis.dll": The specified module could not be found. 로드할 수 있다는 에러 메시지를 받습니다. 혹은 rtpostgis.sql 을 실행하려 할 때 리눅스에서 라이브러리를 로드할 수 없습니다.

rtpostgis.so/dll 은 libgdal.dll/so 의 종속성(dependency)과 함께 설치됩니다. 윈도우 용일 경우 본인 pc 의 PostgreSQL 가 설치된 휴지통에 libgdal-1.dll 있는지를 확인하십시오. 리눅스에 경우 libgdal 은 본인의 경로 또는 휴지통에 있어야 합니다.

만약 데이터베이스에 PostGIS 를 설치하지 않으셨다면 다른 종류의 에러들이 나타나게 될 것 입니다. 래스터 지원을 설치하기 앞서 본인의 데이터베이스에 PostGIS 설치하는 것을 잊지 마십시오.

5. 어떻게 PostGIS 에 래스터 데이터를 로드합니까?

PostGIS 최신 버전은 다양한 종류의 래스터를 로딩하고 다른 추가 소프트웨어 없이 생성할 수 있는 `raster2pgsql` 래스터 로더실행모듈과 함께 패키지로 제공 됩니다. 더 많은 정보를 위해 5.1.1 단락을 참조하십시오. 2.0 versions 이전 버전은 numpy 와 GDAL 가 있는 python 를 요하는 `raster2pgsql.py` 함께 있었습니다. 하지만 이제는 더 이상 이 모듈기능은 쓰이지 않습니다.

6. 어떤 종류의 레스터 파일 포맷들을 내 데이터베이스에 로드할 수 있나요?

GDAL 라이브러리가 지원하는 모든 포맷이 가능합니다. GDAL 지원 포맷은 관련 문서에 기록된 파일 포맷들을 참조하십시오.

단, 특정 GDAL 설치버전은 모든 포맷을 지원하지 않을 수도 있습니다. 특정 GDAL 인스톨에 의해 지원 되는 포맷들을 확인하시기 위해서 다음 명령어을 사용하십시오.

```
raster2pgsql -G
```

7. 다른 래스터 포맷들로 PostGIS 래스터 데이터를 변환 할 수 있습니까?

네 가능합니다.

GDAL 1.7+은 PostGIS 래스터 드라이버를 가지고 있으나 PostgreSQL 지원과 컴파일을 선택했을 시에만 컴파일 될 수 있습니다.

비록 PostGIS 래스터 데이터 타입 내 지원 되지않는 레스터데이터 타입들을 저장할 순 있지만 해당 드라이버 현재 이들을 지원하지 못합니다.

만약 소스로부터 컴파일 중이시라면 본인의 개발환경설정내용으로 아래의 내용이 포함되어야 합니다.

```
--with-pg=path/to/pg_config
```

다양한 OS platforms 에 GDAL 를 구동하는 것과 드라이버 구동에 대한 팁은 GDAL Build Hints 참조하십시오..

만약 설치된 GDAL 버전과 PostGIS 래스터 드라이버를 컴파일 된다면 이를 진행하실 때 PostGIS 포맷 목록을 반드시 보셔야 합니다.

```
gdalinfo --formats
```

gdalinfo 을 통하여 설치된 레스터 기능에 관한 요약을 얻기 위해서 다음의 명령어를 수행 합니다.

```
gdalinfo "PG:host=localhost port=5432 dbname='mygisdb' user='postgres'
password='whatever' schema='someschema' table=sometable"
```

다른 레스터 포맷들의 데이터를 임포트하기 위해서는 gdal_translate 을 사용하십시오, 아래는 테이블의 모든 데이터를 PNG 파일에 10%사이즈로 임포트될 것입니다.

작업하는 레스터 데이터의 픽셀 밴드 유형에 따라 해당 변환된 포맷이 픽셀 타입을 지원하지 않는다면 일부 변환이 되지 않을 수 있습니다.예를 들어 부동소수점 밴드 유형과 32 비트 부호없는 정수 는 JPG 또는 다른 데이터 타입으로 변형되지 않는 경우입니다.

아래는 변형에 관한 간단한 예제입니다.

```
gdal_translate -of PNG -outsize 10% 10% "PG:host=localhost
dbname='mygisdb' user='postgres' password=whatever' schema='someschema'
table=sometable" C:\somefile.png
```

귀하는 임포트가 되는 절에 where=…구문을 사용하여 SQL 문을 실행할 수 있습니다.

위의 내용은 드라이버 연결 문자열 입니다. 아래의 일부는 where 절을 사용하는 예입니다.

```
gdal_translate -of PNG -outsize 10% 10% "PG:host=localhost
dbname='mygisdb' user='postgres' password=whatever' schema='someschema'
table=sometable" C:\somefile.png
```

```
gdal_translate -of PNG -outsize 10% 10% "PG:host=localhost
dbname='mygisdb' user=' -
postgres' password=whatever' schema='someschema' table=sometable where='
-
ST_Intersects(rast, ST_SetSRID(ST_Point(-71.032,42.3793),4326) )' " C:\ -
intersectregion.png
```

보다 많은 예제와 구문을 원하시면 Reading Raster Data of PostGIS Raster section 를 참조하십시오.

8. 이미 설치 된 PostGIS 래스터 기능에 컴파일된 GDAL 바이너리가 이용가능 합니까?

네. GDAL Binaries 페이지를 확인하십시오. Any compiled with PostgreSQL support 와 컴파일 된 어떤 것이라도 반드시 PostGIS 래스터 모듈이 설치 pc 안에 존재해야합니다.

다음의 윈도우 바이너리는 빌드 된 PostGIS 래스터모듈을 포함 있다는 것을 알고 있습니다.

Windows 를 위한 FWTools 최신 버전은 래스터 지원이 컴파일 되었습니다.

PostGIS 레스터는 많은 변화를 하고 있습니다. 만약 윈도우 용 최신 버전을 원하신다면 GDAL trunk, Python 바인딩 그리고 맵서버 익테큐터블 그리고 빌드인 PostGIS 래스터 드라이버를 포함하는 비쥬얼 스투디오와 함께 설치된 Tamas Szekeres nightly 빌드를 체크하십시오. SDK bat 부터 실행하십시오(http://vbkto.dyndns.org/sdk/). AVS project 파일들 또한 이용 가능합니다.

9. PostGIS 래스터 데이터를 뷰하기 위해 어떤 툴들을 사용가능합니까?

GDAL 1.7+이 있는 MapServer 로 컴파일 된 맵서버를 사용하실 수 있으며 래스터 데이터를 뷰하기 위한 PostGIS Raster 드라이버 지원을 사용하실 수 있습니다.. 만약 PostGIS 래스터 드라이버가 설치되어 있을 경우 QuantumGIS (QGIS) 은 이제 PostGIS 래스터 뷰잉(viewing)을 지원합니다.

이론상 GDAL 를 사용하여 데이터를 렌더링하는 모든 도구는 PostGIS 래스터 데이터를 지원하거나 아주 최소한의 범위에서 지원합니다. 윈도우 사용자를 위해 다시 말하자면, Tamas 의 바이너리 http://vbkto.dyndns.org/sdk/ 실행은 혼자서 컴파일하여 설치를 하실 때 겪을 수 있는 어려움을 피하기 위한 좋은 선택입니다.

10. PostGIS 래스터 레이어를 어떻게 my MapServer 맵에 추가할 수 있습니까?

첫째 GDAL 1.7 이상 버전을 PostGIS raster support 와 컴파일 시켜야 합니다. 많은 이슈들이 1.8 버전에서 해결되었고 더 많은 PostGIS 래스터 이슈들이 trunk 버전에서 고쳐졌기 때문에 GDAL 1.8 이상을 권장 합니다.

맵서버 래스터 레이어들과 함께 사용할 수 있는 다양한 처리 함수들의 실행방법을 알기 위해서는 MapServer Raster processing options 를 참조하십시오. PostGIS 래스터 데이터가 특별히 흥미롭게 만드는 것은 각 타일이 다양한 표준 데이터베이스 구조에서 쓰일수 있으며 데이터 소스 내 분할된 형태로 가지고 있을 수 있다는 사실입니다, 아래는 맵서버에서 어떻게 PostGIS 래스터 레이어를 정의하실 수 있는지에 대한 예시입니다.

`mode=2` 는 타일래스터를 위해 요구되어지며 `PostGIS 2.0` 와 `GDAL 1.8` 드라이버에 추가됩니다. 이것은 `GDAL 1.7` 드라이버에 존재하지 않습니다.

```
-- displaying raster with standard raster options
LAYER
       NAME coolwktraster
       TYPE raster
       STATUS ON
       DATA "PG:host=localhost port=5432 dbname='somedb' user='someuser'
password='whatever'
              schema='someschema' table='cooltable' mode='2'"
       PROCESSING "NODATA=0"
       PROCESSING "SCALE=AUTO"
       #... other standard raster processing functions here
       #... classes are optional but useful for 1 band data
       CLASS
              NAME "boring"
              EXPRESSION ([pixel] < 20)
              COLOR 250 250 250
       END
       CLASS
              NAME "mildly interesting"
              EXPRESSION ([pixel] > 20 AND [pixel] < 1000)
              COLOR 255 0 0
       END
       CLASS
              NAME "very interesting"
              EXPRESSION ([pixel] >= 1000)
              COLOR 0 255 0
       END
END

-- displaying raster with standard raster options and a where clause
LAYER
       NAME soil_survey2009
       TYPE raster
       STATUS ON
       DATA "PG:host=localhost port=5432 dbname='somedb' user='someuser'
password='whatever'
              schema='someschema'                          table='cooltable'
where='survey_year=2009' mode='2'"
       PROCESSING "NODATA=0"
       #... other standard raster processing functions here
       #... classes are optional but useful for 1 band data
END
```

11. 래스터 데이터 기능을 무엇을 어디까지 지원하고 있나요?

9 장의 리스트를 참조하십시오. 더욱 많은 내용이 지원될 예정이지만 아직 진행 중입니다.

지원 예정인 많은 사항에 대한 정보를 원하신다면 PostGIS Raster roadmap page 를 참조하십시오.

12. 해당 에러를 갖습니다. ERROR: function st_intersects(raster, unknown) is not unique or st_union(geometry,text) is not unique. 어떻게 고칠 수 있나요?

만약 실행 중인 인수중 하나가 공간객체 대신 공간객체을 문자열로 표현한 것이라면 에러를 발생합니다. 이러한 경우에는 PostgreSQL 을 알 수없는 형식으로 텍스트 표현을 표시합니다, 이는 ST_Intersects(래스터, 지오메트리) 또는 t_intersects(raster,raster)를 상황에 맞지않게 쓸 수 있다는 것을 의미합니다. 그러므로 이론상 수행이 가능한 두 함수들은 귀하의 요구를 지원할 수 있으므로 시스템 상에서 오류 상황을 야기합니다. 이를 방지하기 위해서, 공간 객체형태로 정의할 필요가 있습니다.

예를 들어 만약 귀하의 코드가 아래와 같이 구성 되었다면:

```
SELECT rast
 FROM my_raster
   WHERE ST_Intersects(rast, 'SRID=4326;POINT(-10 10)');
```

레스터 공간 도형은 공간 객체로 다음과 같이 변환이 가능합니다.

```
SELECT rast
 FROM my_raster
   WHERE ST_Intersects(rast, 'SRID=4326;POINT(-10 10)'::geometry);
```

13. 어떻게 PostGIS Raster Oracle GeoRaster (SDO_GEORASTER) 그리고 SDO_RASTER types 과 다릅니까?

이 주제에 관한 더 많은 토론을 원하신다면 Jorge Arévalo 의 Oracle GeoRaster and PostGIS Raster: First impressions 를 체크 하십시오.

Tone-georeference-by-raster 가 단일좌표참조 레스터 보다 더 성능적으로 우수한 측면은 단일좌표참조 레스터 가 경계범위가 더 크기 때문입니다: * 경계영역이 직사각형일 필요가 없습니다. (큰 범위를 커버하기 위한 레스터 케이스 문서에 있는 가능한 래스터 방식들 보십시오(편집자 주: MBR 정보를 갖는 가상의 경계 영역이 꼭 직사각형일 이유는 없습니다.) * rasters to overlaps(겹침) (래스터로 변환할시 벡터 데이터의 손실없이 실행하는 데 필요합니다) 이러한 방식들은 오라클에서도 가능합니다. 하지만 그들은 많은 SDO_RASTER 테이블로 연결된 여러 SDO_GEORASTER 개체의 저장소를 커져야함을 의미 합니다. 복잡한 형태로 이루어진 공간적 범위는 데이터베이스의 수백개의 테이블로도 구성 될수 있습니다. PostGIS 래스터를 사용하면 고유 한 테이블에 유사한 레스터 배열을 저장할 수 있습니다. 이것은 PostGIS 가 자동적으로 오직 전체 사각형 MBR 벡터범위를 저장하도록 하는 것으로 보일수 있습니다. 일부 애플리케이션에서는 매우 유용하지만 대부분의 경우 실행하였을 때 이는 기술적 유용성이 없고 대부분 지리적 경계를 처리 하는 데에 바람직하지 않다는 사실을 발견되었습니다. 벡터 구조는 연속 및 비 직사각형 경계영역를 저장하기 위한 유연성이 필요합니다. 레스터 구조에도 이 방식은 큰 장점이라 생각됩니다.

14. N 바이트의문자열 인코딩 변환을 한 대용량 파일의 raster2pgsql 명령 실행은 왜 이렇게 오랜시간이 걸립니까?

로드하기 위한 파일을 생성할 때 raster2pgsq 는 설치 된 본인의 데이터베이스에 그 어떤 연결도 만들지 않습니다. 설치 된 데이터베이스가 데이베이스 인코딩과는 다른 명시적 클라이언트 인코딩을 설정하고 난 후 비교적 대용량의 레스터 파일들(30MB 이상의 크기)을 로딩할 경우, 변환하여 인코딩 하는데 수행시간이 지나치게 오래 걸리는 상황을 초래할 수도 있습니다.

UTF8 에서 데이터베이스가 있는 경우에 발생하지만 윈도우 애플리케이션을 지원하기 위해, 클라이언트 인코딩이 `WIN1252` 으로 설정되어 있습니다.

이 문제를 해결하려면 로드하는 해당 클라이언트 인코딩은 귀하의 데이터베이스 인코딩과 동일해야합니다. 만일 윈도우 운영체제를 사용중이시라면 로드 스크립트에 명확하게 해당 인코딩을 설정함으로써 해결하실 수 있습니다.

```
set PGCLIENTENCODING=UTF8
```

Unix/Linux 체제 사용 중이라면 다음과 같이 설정 합니다.

```
export PGCLIENTENCODING=UTF8
```

더 자세한 정보는 http://trac.osgeo.org/postgis/ticket/2209 에서 찾을 수 있습니다.

Chapter 11. 토폴로지

PostGIS 토폴로지 타입들과 함수들은 면(Face), 에지(Edge), 노드(Node)와 같은 오브젝트들을 관리하기 위해 사용됩니다.

파리에서 열린 2011 년 PostGIS day 회의에서의 산드로 틸리의 프리젠테이션인 Topology with PostGIS 2.0 slide deck 는 바람직한 PostGIS 토폴로지에 대한 개요 및 방향성에 대한 좋은 정보를 제공합니다.

Vincent Picave 은 State of the art of FOSS4G for topology and network analysis 에서 토폴로지란 무엇인지, 이를 어떻게 이용하는지, 이를 지원하는 다양한 FOSS4G 툴에 대한 좋은 개요와 개략적인 설명을 제공합니다.

토폴로지적인 GIS 데이터베이스의 한 예는 US Census Topologically Integrated Geographic Encoding and Reference System (TIGER) 데이터베이스 입니다. 만약 사용자가 PostGIS 토폴로지를 실험하기 위해 데이터가 좀 필요하신 경우에는 Topology_Load_Tiger 를 확인해 보십시오.

PostGIS 토폴로지 모듈은 PostGIS 의 이전 버전에도 존재했지만, PostGIS 공식문서의 일부였던 적은 없습니다. PostGIS 2.0.0 에서의 주로 정리되고 있는 부분들은, 더 이상 불필요한 모든 기능의 사용 제거, 알려진 사용성 이슈 수정, 기능과 함수들에 대한 더 나은 문서화, 새로운 기능 추가, SQL-MM 표준 준수 향상에 대한 것입니다.

이 프로젝트에 관한 보다 자세한 사항은 PostGIS Topology Wiki 에서 찾으실 수 있습니다.

이 모듈과 관련된 모든 기능과 테이블은 `topology` 라 불리는 스키마에 설치됩니다.

SQL/MM 표준에서 정의 된 함수들 앞에는 ST_라는 접두사가 붙어있고 PostGIS 에 특화된 기능들은 접두사가 붙어 있지 않습니다.

토폴로지가 지원되도록 PostGIS 2.0 를 빌드하기 위해서는 2 장에서 설명된 것처럼 `--with-topology` 옵션과 함께 컴파일 하십시오. 일부 함수들은 GEOS 3.3+에 의존성이 있습니다. 그러므로 토폴로지 지원을 완전히 활용하기 위해서는 GEOS 3.3+와 함께 컴파일 하셔야 합니다.

11.1. 토폴로지 타입

이 섹션은 PostGIS 토폴로지에 의해 설치되는 PostgreSQL 데이터 유형들의 목록입니다. 우리는 형변환에 따른 행동을 설명하고 있고, 이는 특히 사용자가 스스로 함수들을 설계할 때 매우 중요함을 명심하십시오.

getfaceedges_returntype — 일련번호화와 에지번호로 구성된 복합형. 이것은 `ST_GetFaceEdges` 을 위한 반환값입니다.

TopoGeometry — 토폴로지적으로 정의된 지오메트리를 표현하는 복합형.

validatetopology_returntype — 에러의 위치를 나타내기 위한 에러 메시지와 id1, id2 로 구성된 복합형. `ValidateTopology` 을 위한 반환값입니다.

토폴로지 타입들의 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/Topology.html#Topology_Types

11.2. 토폴로지 도메인

이 단락은 PostGIS 토폴로지에 의해 설치된 PostgreSQL 도메인의 목록입니다. 도메인은 함수들이나 테이블 컬럼 반환 객체와 같은 객체 유형들처럼 사용할 수 있습니다. 도메인과 유형의 차이점은, 도메인은 이것의 범위에 대한 검사 조건(Check constraint)을 가지고 있는 타입이라는 것입니다.

TopoElement — 두 정수의 배열로, 일반적으로 TopoGeometry 구성 요소를 식별하는 데 사용됩니다..

TopoElementArray — TopoElement 오브젝트들의 배열.

토폴로지 도메인들의 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/Topology.html#Topology_Domains

11.3. 토폴로지 및 TopoGeometry 관리

이 단락은 새로운 토폴로지 스키마를 구축하고, 토폴로지를 검증하며, TopoGeometry 컬럼들을 관리하는 토폴로지 함수들의 목록입니다.

AddTopoGeometryColumn — 기존 테이블에 topoGeometry 컬럼을 추가, topology.layer 의 레이어로 새로운 컬럼을 등록하고 새 layer_id 를 반환합니다.

DropTopology — 사용시 주의: 토폴로지 스키마를 드랍하고, 토폴로지의 해당 스키마 참조를 topology.topology 테이블과 geometry_columns 테이블의 스키마에 있는 참조 테이블에서 삭제합니다.

DropTopoGeometryColumn — `schema_name` 스키마의 table_name 이라는 이름을 가진 테이블에서 topogeometry 컬럼을 드랍하고, 이 컬럼을 topology.layer 테이블에서 등록해제 합니다.

TopologySummary — 토폴로지 이름을 받아 토폴로지에 있는 객체 유형의 요약 합계를 제공합니다.

ValidateTopology — 토폴로지 문제를 자세히 설명하는 validatetopology_returntype 객체들의 집합을 반환합니다.

토폴로지 및 TopoGeometry 관리에 관한 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/Topology.html#Topology_ManagementFunctions

11.4. 토폴로지 생성자

이 단락에서는 새로운 토폴로지를 생성하기 위한 토폴로지 함수들을 설명합니다.

CreateTopology — 새 토폴로지 스키마를 만들고 topology.topology 테이블에 새 스키마를 등록합니다.

CopyTopology — 토폴로지 구조 (노드, 에지, 면, 레이어, TopoGeometry 들)의 복사본을 만듭니다.

ST_InitTopoGeo — 새 토폴로지 스키마를 만들고 topology.topology 테이블에 등록하고, 처리과정의 세부 요약을 알려줍니다.

ST_CreateTopoGeo — 주어진 빈 토폴로지에 지오메트리의 컬렉션을 추가하고 성공시 자세한 설명 메시지를 반환합니다.

TopoGeo_AddPoint — 허용오차(tolerance)를 이용, 기존의 에지를 분할하여 기존의 토폴로지에 점을 추가합니다.

TopoGeo_AddLineString — 허용오차(tolerance)를 이용, 기존의 에지/면을 분할하여 기존 토폴로지에 스트링을 추가합니다.

TopoGeo_AddPolygon — 허용오차(tolerance)를 이용, 기존의 에지/면을 분할하여 기존 토폴로지에 다각형을 추가합니다.

토폴로지 생성자 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오: http://postgis.net/docs/manual-2.0/Topology.html#Topology_Constructors

11.5. 토폴로지 편집기

이 단락은 추가, 이동, 삭제와 에지, 면, 노드를 분할하는 토폴로지 함수들을 설명합니다. 이러한 기능은 모두 ISO SQL/MM 에 의해 정의됩니다.

ST_AddIsoNode — 토폴로지의 면에 고립된 노드를 추가하고 새 노드의 nodeid 를 반환합니다. 면이 null 이라도 노드는 여전히 생성됩니다.

ST_AddIsoEdge — alinestring 라는 지오메트리에 의해 정의되는 독립된 에지를 기존에 존재하는 두 개의 고립된 노드 anode 와 anothernode 를 연결하는 토폴로지에 추가합니다. 그리고 새로운 에지의 에지 id 를 반환합니다.

ST_AddEdgeNewFaces — 새로운 에지를 추가하고, 이로 인해 면이 분할되는 경우, 원래의 면을 삭제하고 두 개의 새로운 면들로 대체합니다.

ST_AddEdgeModFace — 새로운 에지를 추가하고, 이로 인해 면이 분할되는 경우, 원래의 면을 수정하고 새로운 면을 하나 추가합니다.

ST_RemEdgeNewFace — 에지를 제거하고, 만약 제거된 에지가 두 면을 분리하고 있었다면, 원래의 면들을 삭제하고 새로운 한 면으로 대체합니다.

ST_RemEdgeModFace — 에지를 제거하고, 만약 제거된 에지가 두 면을 분리하고 있었다면, 면 중 하나를 삭제하고 다른 하나의 면을 나머지 영역을 포함하도록 수정합니다.

ST_ChangeEdgeGeom — 토폴로지 구조에 영향을 주지 않으며 에지의 모양을 변경합니다.

ST_ModEdgeSplit — 기존의 에지 위에 새 노드를 만들고, 원래의 에지를 수정하고 새로운 에지를 추가하여 에지를 분할합니다.

ST_ModEdgeHeal — 두 개의 에지를 연결하는 노드를 삭제하여 치료합니다. 첫 번째 에지를 수정하고 두 번째 에지를 삭제합니다. 삭제된 노드의 ID 를 반환합니다.

ST_NewEdgeHeal — 두 개의 에지를 연결하는 노드를 삭제하여 치료합니다. 두 에지를 모두 삭제하고 삭제된 에지들을 첫 번째 주어졌던 에지와 같은 방향의 에지로 대체합니다.

ST_MoveIsoNode — 한 점에서 다른 점으로 토폴로지에 고립 된 노드를 이동합니다. 새로운 apoint 라는 지오메트리가 노드로 존재하는 경우 오류를 발생시킵니다. 이동에 대한 설명을 반환합니다.

ST_NewEdgesSplit — 기존의 에지 위에 새 노드를 만들고, 원래의 에지를 삭제한 뒤 이를 두 개의 새로운 에지로 교체하여 에지를 분할합니다. 두 에지가 합쳐진 새로운 에지의 ID 를 반환합니다.

ST_RemoveIsoNode — 고립 된 노드를 제거하고 동작에 대한 설명을 반환합니다. 노드가 고립되지 않는 경우 (노드의 시작 또는 끝), 예외가 발생됩니다.

토폴로지 편집기 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/Topology.html#Topology_Editing

11.6. 토폴로지 접근자

GetEdgeByPoint — 주어진 점에서 교차하는 에지의 에지 ID 를 찾습니다.

GetFaceByPoint — 주어진 점에서 교차하는 면의 면 ID 를 찾습니다.

GetNodeByPoint — 점 위치에 있는 노드의 ID 를 찾습니다.

GetTopologyID — 주어진 토폴로지의 이름을 `topology.topology` 테이블에서 찾아 토폴로지의 ID 를 반환합니다.

GetTopologySRID — 주어진 토폴로지의 이름을 `topology.topology` 테이블에서 찾아 토폴로지의 SRID 를 반환합니다.

GetTopologyName —주어진 토폴로지 ID 에 해당하는 토폴로지 이름(스키마)을 반환합니다.

ST_GetFaceEdges — aface 라는 면을 구성하는 순차적인 에지들의 집합을 순번과 함께 반환합니다.

ST_GetFaceGeometry — 주어진 토폴로지 내의 폴리곤을 면 ID 와 함께 반환합니다.

GetRingEdges — 주어진 에지와 함께 고리를 형성하는 에지들의 순차적인 집합을 반환합니다.

GetNodeEdges — 주어진 노드가 들어있는 에지들의 순차적인 집합을 반환합니다.

토폴로지 접근자 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/Topology.html#Topology_Accessors

11.7. 토폴로지 처리

이 단락은 비 표준적인 방법으로 토폴로지를 처리하는 함수에 대한 설명을 합니다.

Polygonize — 토폴로지 에지들에 의해 정의된 모든 면들을 찾아서 등록.

AddNode — 지정된 토폴로지 스키마의 노드 테이블에 점 노드를 추가하고 새 노드의 nodeid 를 반환합니다. 만약 점이 노드로써 이미 존재한다면, 존재하고 있는 nodeid 가 반환됩니다.

AddEdge — 라인스트링 에지를 에지 테이블에 추가하고 연관된 시작 및 끝 점들을 지정된 라인스트링 지오메트리를 사용하는 지정된 토폴로지 스키마의 포인트 노드 테이블에 추가합니다. 그리고 새로운(또는 존재하는) 에지의 edgeid 를 반환합니다.

AddFace — 면 프리미티브를 토폴로지에 등록하고 이것의 식별자를 얻습니다.

토폴로지 처리와 관련한 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/Topology.html#Topology_Processing

11.8. TopoGeometry 생성자

이 단락에서는 새로운 topogeometires 를 생성하기 위한 토폴로지 함수들을 설명합니다.

CreateTopoGeom — TOPO 요소 배열에서 새 topo 지오메트리 객체를 생성 - tg_type:1 [multi] point, 2:[multi] line, 3:[multi] poly, 4:collection

toTopoGeom — 단순 지오메트리에서 새로운 topo 지오메트리를 생성합니다.

TopoElementArray_Agg — element_id, 형식 배열 (topoelements)의 집합에 대한 topoelementarray 를 반환합니다.

TopoGeometry 생성자 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/Topology.html#TopoGeometry_Constructors

11.9. TopoGeometry 접근자

GetTopoGeomElementArray — 주어진 TopoGeometry(기본 요소)의 위상적 요소와 유형을 포함하는 topoelementarray (topoelements 의 배열)을 반환.

GetTopoGeomElements — 주어진 TopoGeometry(기본 요소)의 위상적 element_id, element_type 을 포함하는 topoelement 객체를 반환.

TopoGeometry 접근자 함수들의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/Topology.html#TopoGeom_Accessors

11.10. TopoGeometry 출력

AsGML — topogeometry 의 GML 표현을 반환.

TopoGeometry 출력과 관련한 함수의 Synopsis, 설명, 예시를 보려면 다음을 참조하십시오:
http://postgis.net/docs/manual-2.0/Topology.html#TopoGeometry_Outputs

Chapter 12. PostGIS 부가기능

이 장은 PostGIS 소스 타르볼(tarballs) 및 소스 저장소(repository)의 Extras 폴더에서 찾은 객체를 기록합니다. 이 부가기능은 PostGIS 바이너리 릴리스와 함께 항상 제공되지 않지만 일반적으로 실행할 수 있는 plpgsql 기반 또는 표준 Shell 스크립트입니다.

12.1. Tiger Geocoder

PostGIS 에서 인기를 얻고 국제적으로 사용이 더 적합한 또 다른 지오코더가 있습니다. 그것은 Nominatim 이라고 하며 OpenStreetMap gazeteer 형식의 데이터를 사용합니다. 이는 데이터를 로딩하기 위하여 PostgreSQL8.4 + 및 PostGIS 1.5 +의 가능인 osm2pgsql 을 필요로 합니다. 이것은 웹서비스 인터페이스로 패키징되어 있으며, 웹서비스로 호출되도록 설계된 것으로 보입니다. Tiger Geocoder 처럼 이것 역시 지오코더, 리버스지오코더 두 가지 요소로 구성되어있습니다. 문서에 따르면 이것이 Tiger 지오코더 처럼 순수한 SQL 로 이루어진 것인지 다량의 로직이 웹 인터페이스에서 구현된 것인지 명확하지 않습니다.

12.1.1. Drop_Indexes_Generate_Script

Drop_Indexes_Generate_Script - tiger 스키마 상의 모든 non-unique 인덱스와 non-primary 키 그리고 사용자 정의 스키마를 삭제하는 스크립트를 생성합니다. 스키마가 정의되지 않았을 경우 기본값으로 `tiger_data` 로 정의됩니다.

Synopsis

text **Drop_Indexes_Generate_Script**(text param_schema=tiger_data);

Description

tiger 스키마 상의 모든 non-unique 인덱스와 non-primary 키 그리고 사용자 정의 스키마를 삭제하는 스크립트를 생성합니다. 스키마가 정의되지 않았을 경우 기본값으로 `tiger_data` 로 정의됩니다.

이것은 쿼리 플래너의 혼란과 불필요한 공간을 취할 인덱스의 팽창을 최소화 하는데 유용합니다. 지오코더에 의해 사용되는 인덱스를 추가 할 때 Install_Missing_Indexes 와 함께 사용하십시오.

Availability: 2.0.0

Examples

```
SELECT drop_indexes_generate_script() As actionsql;
actionsql
-----------------------------------------------------------
DROP INDEX tiger.idx_tiger_countysub_lookup_lower_name;
DROP INDEX tiger.idx_tiger_edges_countyfp;
DROP INDEX tiger.idx_tiger_faces_countyfp;
DROP INDEX tiger.tiger_place_the_geom_gist;
DROP INDEX tiger.tiger_edges_the_geom_gist;
DROP INDEX tiger.tiger_state_the_geom_gist;
DROP INDEX tiger.idx_tiger_addr_least_address;
DROP INDEX tiger.idx_tiger_addr_tlid;
DROP INDEX tiger.idx_tiger_addr_zip;
DROP INDEX tiger.idx_tiger_county_countyfp;
DROP INDEX tiger.idx_tiger_county_lookup_lower_name;
DROP INDEX tiger.idx_tiger_county_lookup_snd_name;
DROP INDEX tiger.idx_tiger_county_lower_name;
DROP INDEX tiger.idx_tiger_county_snd_name;
DROP INDEX tiger.idx_tiger_county_the_geom_gist;
DROP INDEX tiger.idx_tiger_countysub_lookup_snd_name;
DROP INDEX tiger.idx_tiger_cousub_countyfp;
DROP INDEX tiger.idx_tiger_cousub_cousubfp;
DROP INDEX tiger.idx_tiger_cousub_lower_name;
DROP INDEX tiger.idx_tiger_cousub_snd_name;
DROP INDEX tiger.idx_tiger_cousub_the_geom_gist;
DROP INDEX tiger_data.idx_tiger_data_ma_addr_least_address;
DROP INDEX tiger_data.idx_tiger_data_ma_addr_tlid;
DROP INDEX tiger_data.idx_tiger_data_ma_addr_zip;
DROP INDEX tiger_data.idx_tiger_data_ma_county_countyfp;
DROP INDEX tiger_data.idx_tiger_data_ma_county_lookup_lower_name;
DROP INDEX tiger_data.idx_tiger_data_ma_county_lookup_snd_name;
DROP INDEX tiger_data.idx_tiger_data_ma_county_lower_name;
DROP INDEX tiger_data.idx_tiger_data_ma_county_snd_name;
:
:
```

See Also

Install_Missing_Indexes, Missing_Indexes_Generate_Script

12.1.2. Drop_State_Tables_Generate_Script

Drop_State_Tables_Generate_Script — 상태 약어로 시작되는 지정된 스키마의 모든 테이블을 삭제하는 스크립트를 생성합니다. 스키마가 정의되지 않았을 경우 기본값으로 `tiger_data` 로 정의됩니다.

Synopsis

text **Drop_State_Tables_Generate_Script**(text param_state, text param_schema=tiger_data);

Description

상태 약어로 시작되는 지정된 스키마의 모든 테이블을 삭제하는 스크립트를 생성합니다. 스키마가 정의되지 않았을 경우 기본값으로 tiger_data 로 정의됩니다. 이 기능은 이전에 로드하는 중 잘못된 경우가 발생하였을 때 로드직전 상태로 되돌리기 위해 테이블을 드롭 할 때 유용합니다.

Availability: 2.0.0

Examples

```
SELECT drop_state_tables_generate_script('PA');
DROP TABLE tiger_data.pa_addr;
DROP TABLE tiger_data.pa_county;
DROP TABLE tiger_data.pa_county_lookup;
DROP TABLE tiger_data.pa_cousub;
DROP TABLE tiger_data.pa_edges;
DROP TABLE tiger_data.pa_faces;
DROP TABLE tiger_data.pa_featnames;
DROP TABLE tiger_data.pa_place;
DROP TABLE tiger_data.pa_state;
DROP TABLE tiger_data.pa_zip_lookup_base;
DROP TABLE tiger_data.pa_zip_state;
DROP TABLE tiger_data.pa_zip_state_loc;
```

See Also

Loader_Generate_Script

12.1.3. Geocode

Geocode — 문자열 (또는 다른 정규화 된 주소)로 주소를 받아 **NAD 83 long lat** 의 포인트 지오메트리, 각각에 대한 정규화 된 주소, 그리고 순위를 포함한 가능한 위치의 집합을 출력합니다. 순위의 번호가 낮을수록 더 많이 일치합니다. 결과들은 가장 낮은 순위를 첫째로 나열합니다. 최대 출력 결과 개수(기본값은 10 개) 와 restrict_region (디폴트 NULL)을 선택적으로 제출할 수 있습니다.

Synopsis

setof record **geocode**(varchar address, integer max_results=10, geometry restrict_region=NULL, norm_addy OUT addy, geometry OUT geomout, integer OUT rating);

setof record **geocode**(norm_addy in_addy, integer max_results=10, geometry restrict_region=NULL, norm_addy OUT addy,geometry OUT geomout, integer OUT rating);

Descroption

문자열 (또는 이미 정규화 된 주소)로 주소를 받아 NAD 83 long lat 의 포인트 지오메트리, 각각에 대한 nomalized_address(addy), 그리고 순위를 포함한 가능한 위치의 집합을 출력합니다. 순위의 번호가 낮을수록 더 많이 일치합니다. 결과들은 가장 낮은 순위를 첫째로 나열합니다. Tiger Data(Edges, faces, addr)는 PostgreSQL 의 퍼지 문자열 매칭(soundex, levenshtein)과 PostGIS 의 라인 보간함수를 사용하여 주소를 보간합니다. 높은 번호의 순위는 지오코드의 정확성이 작습니다. 지오코딩된 포인트는 기본적으로 도로의 중앙선으로부터 10m 떨어진 도로 주소의 측면(L/R)에 위치 합니다.

고급: 2.0.0 은 Tiger 1020 구조화된 데이터를 지원하고 지오코딩의 속도 개선, 정확성을 개선, 그리고 도로의 중심선으로부터 도로 주소의 측면에 위치한 포인트의 차이를 상쇄하기 위해 몇 가지 로직을 개정하였습니다. 새로운 파라미터인 max_results 는 최상의 결과만을 리턴 받는데 사용하기에 유용합니다.

Examples: Basic

아래의 예는 MA,MIN,CA,RI 주의 Tiger data 가 로드된 PostgreSQL 9.1rc1/PostGIS 2.0 이 실행되고, 2GB ram 이 탑재된 3.0 GHZ single Processor Windows 7 머신에서 측정된 시간입니다.

정확히 일치하는 계산 시간은 더 빠릅니다(61ms)

```
SELECT g.rating, ST_X(g.geomout) As lon, ST_Y(g.geomout) As lat,
(addy).address As stno, (addy).streetname As street,
(addy).streettypeabbrev    As    styp,    (addy).location    As    city,
(addy).stateabbrev As st,(addy)  -
.zip
FROM geocode('75 State Street, Boston MA 02109') As g;
rating | lon            | lat           |stno|street|styp| city | st | zip
-------+----------------+---------------+---+------+----+-------+---+----
0      | -71.0556722990239| 42.3589914927049 | 75 | State  | St  | Boston | MA | 02109
```

우편번호가 지오코더에 전달되지 않은 경우에도 추측할 수 있습니다(약 122-15- ms 소요됨).

```
SELECT    g.rating,    ST_AsText(ST_SnapToGrid(g.geomout,0.00001))    As
wktlonlat,
(addy).address As stno, (addy).streetname As street,
(addy).streettypeabbrev    As    styp,    (addy).location    As    city,
(addy).stateabbrev As st,(addy)  -
.zip
FROM geocode('226 Hanover Street, Boston, MA',1) As g;
rating | wktlonlat            |stno| street | styp | city | st | zip
-------+----------------------+----+--------+------+------+----+------
1      | POINT(-71.05528 42.36316) | 226 | Hanover | St    | Boston | MA | 02113
```

철자오류를 처리하고 하나 이상의 순위를 가진 결과를 제공하면 더 오랜 시간이 걸립니다(500ms).

```
SELECT    g.rating,    ST_AsText(ST_SnapToGrid(g.geomout,0.00001))    As
wktlonlat,
```

```
(addy).address As stno, (addy).streetname As street,
(addy).streettypeabbrev    As    styp,    (addy).location    As    city,
(addy).stateabbrev As st,(addy)  -
.zip
FROM geocode('31 - 37 Stewart Street, Boston, MA 02116') As g;
rating | wktlonlat            | stno| street | styp | city | st | zip
-------+----------------------+-----+--------+------+------+----+-----
70     | POINT(-71.06459 42.35113) | 31   | Stuart   | St    | Boston | MA  | 02116
```

주소의 지오코드를 일괄 처리. 가장 쉬운 설정 set max_result=1. 아직 지오코드가 되지 않은 것만 처리(순위를 가지지 않음).

```
CREATE TABLE addresses_to_geocode(addid serial PRIMARY KEY, address text,
    lon numeric, lat numeric, new_address text, rating integer);
INSERT INTO addresses_to_geocode(address)
VALUES ('529 Main Street, Boston MA, 02129'),
  ('77 Massachusetts Avenue, Cambridge, MA 02139'),
  ('25 Wizard of Oz, Walaford, KS 99912323'),
  ('26 Capen Street, Medford, MA'),
  ('124 Mount Auburn St, Cambridge, Massachusetts 02138'),
  ('950 Main Street, Worcester, MA 01610');

-- only update the first 3 addresses (323-704 ms - there are caching and
shared memory  -
effects so first geocode you do is always slower) --
-- for large numbers of addresses you don't want to update all at once
-- since the whole geocode must commit at once
-- For this example we rejoin with LEFT JOIN
-- and set to rating to -1 rating if no match
-- to ensure we don't regeocode a bad address
UPDATE addresses_to_geocode
  SET (rating, new_address, lon, lat)
  = ( COALESCE((g.geo).rating,-1), pprint_addy((g.geo).addy),
    ST_X((g.geo).geomout)::numeric(8,5),
ST_Y((g.geo).geomout)::numeric(8,5) )
FROM (SELECT addid
  FROM addresses_to_geocode
  WHERE rating IS NULL ORDER BY addid LIMIT 3) As a
  LEFT JOIN (SELECT addid, (geocode(address,1)) As geo
    FROM addresses_to_geocode As ag
  WHERE ag.rating IS NULL ORDER BY addid LIMIT 3) As g ON a.addid =
g.addid
WHERE a.addid = addresses_to_geocode.addid;

result
-----
Query returned successfully: 3 rows affected, 480 ms execution time.

SELECT * FROM addresses_to_geocode WHERE rating is not null;

addid | address | lon | lat |  -
new_address | rating
-------+-----------------------------------------------+-----------+------
----+-------------------------------------------+-----
1 | 529 Main Street, Boston MA, 02129 | -71.07181 | 42.38359 | 529 Main
```

```
St,  -
Boston, MA 02129 | 0
2 | 77 Massachusetts Avenue, Cambridge, MA 02139 | -71.09428 | 42.35988 |
77  -
Massachusetts Ave, Cambridge, MA 02139 | 0
3 | 25 Wizard of Oz, Walaford, KS 99912323 | | |  -
| -1
```

Examples: Geometry filter 사용

```
SELECT     g.rating,     ST_AsText(ST_SnapToGrid(g.geomout,0.00001))     As
wktlonlat,
(addy).address As stno, (addy).streetname As street,
(addy).streettypeabbrev As styp,
(addy).location As city, (addy).stateabbrev As st,(addy).zip
FROM geocode('100 Federal Street, MA',
3,
(SELECT ST_Union(the_geom)
FROM place WHERE statefp = '25' AND name = 'Lynn')::geometry
) As g;

rating | wktlonlat                | stno | street |styp| city | st | zip
-------+--------------------------+------+--------+----+------+----+-----
8      | POINT(-70.96796 42.4659) | 100  | Federal | St | Lynn | MA | 01905
Total query runtime: 245 ms.
```

See Also

Normalize_Address, Pprint_Addy, ST_AsText, ST_SnapToGrid, ST_X, ST_Y

12.1.4. Geocode_Intersection

Geocode_Intersection — 교차하는 두 개의 도로와 주, 도시, 우편번호를 입력으로 취하여 교차로의 첫 번째 교차점의 위치를 출력합니다. 이 위치 역시 NAD 83 long lat 으로 되어있는 포인트 지오메트리, 각각의 위치에 대한 정규화된 주소 그리고 순위를 포함 합니다. 순위의 번호가 낮을수록 더 일치 됩니다. 낮은 번호의 순위가 첫 번째로 나오는 순서로 정렬됩니다. 선택적으로 최대 결과 수를 설정할 수 있습니다. 기본값은 10 입니다.

Synopsis

setof record **geocode_intersection**(text roadway1, text roadway2, text in_state, text in_city, text in_zip, integer max_results=10, norm_addy OUT addy, geometry OUT geomout, integer OUT rating);

Description

교차하는 두 개의 도로와 주, 도시, 우편번호를 입력으로 취하여 교차로의 첫 번째 교차점의 위치를 출력합니다. 이 위치 역시 NAD 83 long lat 으로 되어있는 포인트 지오메트리, 각각의 위치에 대한 정규화된 주소 그리고 순위를 포함 합니다. 순위의 번호가 낮을수록 더 일치 됩니다. 낮은 번호의 순위가 첫 번째로 나오는 순서로 정렬됩니다. 선택적으로 최대 결과 수를 설정할 수 있습니다. 기본값은 10 입니다. 반환 되는 `nomalized_address`(addy)는 각각 NAD 83 long lat 으로 된 포인트 위치 지오메트리와 순위를 포함합니다.

Availability: 2.0.0

Examples: Basic

아래의 예는 MA 주의 Tiger data 가 로드 된 PostgreSQL 9.1rc1/PostGIS 1.5 이 실행되고, 2GB ram 이 탑재된 3.0 GHZ single Processor Windows 7 에서 측정된 시간입니다. 현재 (3000 ms)보다 조금 느립니다.

Windows 2003 64 bit, 8GB PostGIS 2.0 PostgreSQL 64bit , Tiger 2011 데이터 로드에서 테스트 - (41ms)

```
SELECT pprint_addy(addy), st_astext(geomout),rating
  FROM       geocode_intersection(       'Haverford       St','Germania
St', 'MA', 'Boston', '02130',1);

pprint_addy                        | st_astext                   | rating
-----------------------------------+-----------------------------+--------
98 Haverford St, Boston, MA 02130 | POINT(-71.101375 42.31376) | 0
```

우편번호가 전달되지 않은 경우에 지오코더가 추측할 수 있습니다. windows 2003 64-bit 에서 741ms (Windows 7 에서 약 3500ms 가 걸림)

```
SELECT pprint_addy(addy), st_astext(geomout),rating
    FROM geocode_intersection('Weld', 'School', 'MA', 'Boston');

pprint_addy                    | st_astext                  | rating
-------------------------------+----------------------------+--------
98 Weld Ave, Boston, MA 02119 | POINT(-71.099 42.314234) | 3
99 Weld Ave, Boston, MA 02119 | POINT(-71.099 42.314234) | 3
```

See Also

Geocode, Pprint_Addy, ST_AsText

12.1.5. Get_Tract

Get_Tract — 지오메트리가 위치한 곳의 지역 테이블에서의 필드 또는 인구 조사 지역을 반환합니다. 기본값은 지역의 짧은 이름을 반환합니다.

Synopsis

text **get_tract**(geometry loc_geom, text output_field='name');

Description

주어진 지오메트리는 그 지오메트리의 인구조사 지역의 위치를 반환합니다. 공간참조모델이 정의되지 않았을 경우 NAD 83 long lat 으로 간주 합니다.

Availability: 2.0.0

Examples: Basic

```
SELECT get_tract(ST_Point(-71.101375, 42.31376) ) As tract_name;
tract_name
---------
1203.01
```

```
--this one returns the tiger geoid
SELECT   get_tract(ST_Point(-71.101375,   42.31376),   'tract_id'   )   As
tract_id;

tract_id
---------
25025120301
```

See Also

Geocode

12.1.6. Install_Mssing_Indexes

Install_Missing_Indexes — 지오코더의 조인에 사용되는 키 컬럼을 가지는 모든 테이블과 인덱스 사용이 누락된 필터 조건의 컬럼을 찾고 더합니다.

Synopsis

boolean **Install_Missing_Indexes**();

Description

지오코더의 조인에 사용되는 키 컬럼을 가지는 모든 tiger 와 tiger_data 스키마의 테이블과 인덱스 사용이 누락된 필터 조건의 컬럼을 찾고 인덱스가 정의된 SQL DDL 을 출력하고 이를 실행 합니다.

이것은 프로세스를 로드하는 동안 누락되었을 수 있는 새로운 인덱스를 추가하여 쿼리속도를 더 빠르게 합니다. 이 기능은 인덱스를 생성하는 스크립트를 만드는것 이외에 Missing_Indexes_Generate_Script 과 함께 수행됩니다. 이것은 `update_geocode.sql` 갱신 스크립트의 일부 입니다.

Availability: 2.0.0

Examples

```
SELECT install_missing_indexes();

install_missing_indexes
-------------------------
t
```

See Also

Loader_Generate_Script, Missing_Indexes_Generate_Script

12.1.7. Loader_Generate_Script

Loader_Generate_Script — 지정된 플랫폼에 대한 Tiger 데이터를 내려 받고, `Tiger_data` 스키마에 단계별로 로드하기 위한 지정된 상태 스크립트를 생성합니다. 각각의 상태 스크립트는 별도의 레코드로 반환됩니다. 최신 버전은 Tiger 2010 의 구조적인 변화를 지원하고, 인구조사 구역, 블록 그룹, 블록 테이블을 로드합니다.

Synopsis

setof text **loader_generate_script**(text[] param_states, text os);

Description

지정된 플랫폼에 대한 Tiger 데이터를 내려 받고, `Tiger_data` 스키마에 단계별로 로드하기 위한 상태 별 스크립트를 생성합니다. 각각의 상태 스크립트는 별도의 레코드로 반환됩니다.

Linux 에서는 unzip(Windows 에서는 7-zip 을 사용)을 사용하고 wget 으로 다운로드 합니다. 4.4.2 에서 데이터 로드를 다룹니다. 참고로 최소단위가 전체상태 이지만 파일을 사용자가 직접 다운로드 하여 덮어쓸 수 있습니다. 준비와 임시 폴더 안에서 파일을 처리할 것입니다.

프로세스 제어와 다른 OS 별 Shell 문법차이를 다음 제어 테이블을 사용합니다.

1. **`loader_variables`** 인구조사 사이트, 년도, 데이터와 단계별 스키마와 같은 다양한 변수를 추적합니다.

2. **loader_platform** 다양한 플랫폼과 다양한 실행파일의 위치를 정의합니다. windows 와 Linux 에 제공되며 더 더할 수 있습니다.

3. **loader_lookuptables** 각각의 레코드는 테이블의 종류(states, county) 및 레코드를 처리할지 여부, 어떻게 로드할지를 정의합니다. 데이터와 단계 데이터를 가져오고, 각각의 컬럼, 인덱스 그리고 제약조건을 더하거나 제거하기 위한 절차를 정의 합니다. 각 테이블은 tiger 스키마의 테이블의 상태와 상속으로 시작됩니다. 예를 들어 `tiger.faces` 로 부터 상속 받아 `tiger_data.ma_faces` 를 생성합니다.

Availability: 2.0.0 은 구조적으로 변화된 Tiger 2010 데이터와 인구조사 구역(Tract), Block Group(bg), blocks (tabblocks)의 데이터를 로드하는 것을 지원합니다.

Examples

Windows Shell 스크립트로 2 개의 주를 찾는 스크립트를 생성합니다.

```
SELECT loader_generate_script(ARRAY['MA','RI'], 'windows') AS result;
-- result --
set
STATEDIR="\gisdata\www2.census.gov\geo\pvs\tiger2010st\44_Rhode_Island"
set TMPDIR=\gisdata\temp\
set UNZIPTOOL="C:\Program Files\7-Zip\7z.exe"
set WGETTOOL="C:\wget\wget.exe"
set PGBIN=C:\Program Files\PostgreSQL\8.4\bin\
set PGPORT=5432
set PGHOST=localhost
set PGUSER=postgres
set PGPASSWORD=yourpasswordhere
set PGDATABASE=geocoder
set PSQL="%PGBIN%psql"
set SHP2PGSQL="%PGBIN%shp2pgsql"
%WGETTOOL% http://www2.census.gov/geo/pvs/tiger2010st/44_Rhode_Island/ --
no-parent --  -
relative --recursive --level=2 --accept=zip,txt --mirror --reject=html
:
:
```

sh 스크립트 생성

```
SELECT loader_generate_script(ARRAY['MA','RI'], 'sh') AS result;
-- result --
STATEDIR="/gisdata/www2.census.gov/geo/pvs/tiger2010st/44_Rhode_Island"
TMPDIR="/gisdata/temp/"
UNZIPTOOL=unzip
PGPORT=5432
PGHOST=localhost
PGUSER=postgres
PGPASSWORD=yourpasswordhere
PGDATABASE=geocoder
PSQL=psql
SHP2PGSQ=shp2pgsql
wget   http://www2.census.gov/geo/pvs/tiger2010st/44_Rhode_Island/   --no-
parent --relative --  -
```

```
recursive --level=2 --accept=zip,txt --mirror --reject=html
:
:
```

12.1.8. Loader_Generate_Census_Script

Loader_Generate_Census_Script — Tieger 인구 조사 구역 상태, bg 그리고 tabblocks 데이터 테이블, 단계를 다운로드 하고 tiger_data 스키마로 로드하는 지정된 상태를 위한 특정 플랫폼 Shell 스크립트를 생성합니다. 각 상태 스크립트는 별도의 레코드로 반환됩니다.

Synopsis

setof text **loader_generate_census_script**(text[] param_states, text os);

Description

Tiger 인구 조사 구역 상태 tract, 블록그룹 bg, 그리고 tabblocks 데이터 테이블, 단계를 다운로드 하고 tiger_data 스키마로 로드하는 지정된 상태를 위한 특정 플랫폼 Shell 스크립트를 생성합니다. 각 상태 스크립트는 별도의 레코드로 반환됩니다.

Linux 에서는 unzip(Windows 에서는 7-zip 을 사용)을 사용하고 wget 으로 다운로드 합니다. 4.4.2 에서 데이터 로드를 다룹니다. 참고로 최소단위가 전체상태 이지만 파일을 사용자가 직접 다운로드 하여 덮어쓸 수 있습니다. 준비와 임시 폴더 안에서 파일을 처리할 것입니다.

프로세스 제어와 다른 OS 별 Shell 문법차이를 다음 제어 테이블을 사용합니다.

1. **loader_variables** 인구조사 사이트, 년도, 데이터와 단계별 스키마와 같은 다양한 변수를 추적합니다.
2. **loader_platform** 다양한 플랫폼과 다양한 실행파일의 위치를 정의합니다. windows 와 Linux 에 제공되며 더 더할 수 있습니다.
3. **loader_lookuptables** 각각의 레코드는 테이블의 종류(states, county) 및 레코드를 처리할지 여부, 어떻게 로드할지를 정의합니다. 데이터와 단계 데이터를 가져오고, 각각의 컬럼, 인덱스 그리고 제약조건을 더하거나 제거하기 위한 절차를 정의 합니다. 각 테이블은 tiger 스키마의 테이블의 상태와 상속으로 시작됩니다. 예를 들어 tiger.faces 로 부터 상속 받아 tiger_data.ma_faces 를 생성합니다.

Availability: 2.0.0

이전에 PostGIS 2.0.0 alpha5 의 Tiger 지오코더를 설치 하였다면 이러한 추가 테이블을 얻기 위해서 이 작업을 수행하여야 합니다. Loader_Generate_Script 는 이러한 로직이 포함되어 있습니다.

Examples

Windows Shell 스크립트로 선택한 상태에 대한 데이터를 찾는 스크립트를 생성합니다.

```
SELECT loader_generate_census_script(ARRAY['MA'], 'windows');
-- result --
set
STATEDIR="\gisdata\www2.census.gov\geo\pvs\tiger2010st\25_Massachusetts"
set TMPDIR=\gisdata\temp\
set UNZIPTOOL="C:\Program Files\7-Zip\7z.exe"
set WGETTOOL="C:\wget\wget.exe"
set PGBIN=C:\projects\pg\pg91win\bin\
set PGPORT=5432
set PGHOST=localhost
set PGUSER=postgres
set PGPASSWORD=yourpasswordhere
set PGDATABASE=tiger_postgis20
set PSQL="%PGBIN%psql"
set SHP2PGSQL="%PGBIN%shp2pgsql"
cd \gisdata
%WGETTOOL%
http://www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts/25/      --no-
parent --  -
relative   --accept=*bg10.zip,*tract10.zip,*tabblock10.zip   --mirror   --
reject=html
del %TMPDIR%\*.* /Q
%PSQL% -c "DROP SCHEMA tiger_staging CASCADE;"
%PSQL% -c "CREATE SCHEMA tiger_staging;"
cd %STATEDIR%
for /r %%z in (*.zip) do %UNZIPTOOL% e %%z -o%TMPDIR%
cd %TMPDIR%
%PSQL%   -c   "CREATE   TABLE   tiger_data.MA_tract(CONSTRAINT   pk_MA_tract
PRIMARY KEY (tract_id) )  -
INHERITS(tiger.tract); "
%SHP2PGSQL%  -c  -s  4269  -g  the_geom  -W  "latin1"  tl_2010_25_tract10.dbf
tiger_staging.  -
ma_tract10 | %PSQL%
%PSQL%   -c   "ALTER   TABLE   tiger_staging.MA_tract10   RENAME   geoid10   TO
tract_id; SELECT  -
loader_load_staged_data(lower('MA_tract10'), lower('MA_tract')); "
%PSQL%    -c    "CREATE    INDEX    tiger_data_MA_tract_the_geom_gist    ON
tiger_data.MA_tract USING gist  -
(the_geom);"
%PSQL% -c "VACUUM ANALYZE tiger_data.MA_tract;"
%PSQL%  -c  "ALTER  TABLE  tiger_data.MA_tract  ADD  CONSTRAINT  chk_statefp
CHECK (statefp =  -
'25');"
:
```

sh 스크립트 생성

```
STATEDIR="/gisdata/www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts"
TMPDIR="/gisdata/temp/"
UNZIPTOOL=unzip
WGETTOOL="/usr/bin/wget"
export PGBIN=/usr/pgsql-9.0/bin
export PGPORT=5432
```

```
export PGHOST=localhost
export PGUSER=postgres
export PGPASSWORD=yourpasswordhere
export PGDATABASE=geocoder
PSQL=${PGBIN}/psql
SHP2PGSQL=${PGBIN}/shp2pgsql
cd /gisdata
wget  http://www2.census.gov/geo/pvs/tiger2010st/25_Massachusetts/25/  --
no-parent --relative  -
--accept=*bg10.zip,*tract10.zip,*tabblock10.zip --mirror --reject=html
rm -f ${TMPDIR}/*.*
${PSQL} -c "DROP SCHEMA tiger_staging CASCADE;"
${PSQL} -c "CREATE SCHEMA tiger_staging;"

cd $STATEDIR
for z in *.zip; do $UNZIPTOOL -o -d $TMPDIR $z; done
:
:
```

See Also

Loader_Generate_Script

12.1.9. Missing_Indexes_Generate_Script

Missing_Indexes_Generate_Script — 해당 열에서 사용되는 누락된 인덱스를 추가 할 geocoder 조인(geocoder joins)에 사용되는 키 행들을 가진 테이블들을 찾아내고 해당 테이블들을 위한 인덱스를 정의한 SQL DDL 을 출력합니다.

Synopsis

text **Missing_Indexes_Generate_Script**();

Description

해당 열에서 사용되는 누락된 인덱스를 추가 할 지오코더 조인(geocoder joins)에 사용되는 키 행들을 가진 모든 `Tiger` 와 `tiger_data` 테이블들을 찾아내고 해당 테이블들을 위한 인덱스를 정의한 SQL DDL 을 출력합니다. 프로세스를 로드 하는 동안 누락 됐을 수도 있는 쿼리 속도를 향상시키기 위한 새로운 인덱스를 추가하는 기능입니다. 지오코더가 향상된 이 기능은 새롭게 사용되는 인덱스를 수용하기 위하여 업데이트 됩니다.

Availability: 2.0.0

Examples

```
SELECT missing_indexes_generate_script();
-- output: This was run on a database that was created before many
corrections were made to -
the loading script ---
CREATE INDEX idx_tiger_county_countyfp ON tiger.county USING
btree(countyfp);
CREATE INDEX idx_tiger_cousub_countyfp ON tiger.cousub USING
btree(countyfp);
CREATE INDEX idx_tiger_edges_tfidr ON tiger.edges USING btree(tfidr);
CREATE INDEX idx_tiger_edges_tfidl ON tiger.edges USING btree(tfidl);
CREATE INDEX idx_tiger_zip_lookup_all_zip ON tiger.zip_lookup_all USING
btree(zip);
CREATE INDEX idx_tiger_data_ma_county_countyfp ON tiger_data.ma_county
USING btree(countyfp -
);
CREATE INDEX idx_tiger_data_ma_cousub_countyfp ON tiger_data.ma_cousub
USING btree(countyfp -
);
CREATE INDEX idx_tiger_data_ma_edges_countyfp ON tiger_data.ma_edges
USING btree(countyfp);
CREATE INDEX idx_tiger_data_ma_faces_countyfp ON tiger_data.ma_faces
USING btree(countyfp);
```

See Also

Loader_Generate_Script, Install_Missing_Indexes

12.1.10. Normalize_Address

Normalize_Address — 주어진 도로명 주소 문자를 받아 도로의 접미사, 접두사 그리고 표준화된 형태의 도로, 도로명 등을 가지는 별도의 필드로 분리된 복합적인 nom_addy 를 반환합니다. 이 기능은 Tiger_geocoder 의 조회 데이터(tiger 인수조사 데이터는 필요치 않습니다)와 함께 동작합니다.

Synopsis

norm_addy **normalize_address**(varchar in_address);

Description

주어진 도로명 주소 문자를 받아 도로의 접미사, 접두사 그리고 표준화된 형태의 도로, 도로명 등을 가지는 별도의 필드로 분리된 복합적인 norm_addy 를 반환합니다. 정규화된 우편주소 형태로 모든 주소를 얻는 지오코더의 첫 번째 단계입니다.

지오코더 패키지 이외의 다른 데이타는 요구되지 않습니다.

이 기능은 미리 로드된 tiger 지오코더와 tiger 스키마에 위치한 조회 테이블의 다양한 방향/상태/접미사를 사용합니다. 그래서 tiger 인구조사 데이터나 다른 추가 데이터를 필요로 하지 않습니다.

tiger 스키마의 다양한 조회 테이블에 더 많은 약어 또는 대체 이름을 추가하여 찾을 수도 있습니다.

입력 주소의 정규화를 위해 tiger 스키마에 있는 다양한 조회 테이블 제어를 사용합니다.

이 기능은 norm_addy 형태의 객체로 다음과 같은 순서로 반환 됩니다.

() 는 지오코더가 필요로하는 필드입니다. [] 선택적인 필드를 나타냅니다:
(address) [predirAbbrev] (streetName) [streetTypeAbbrev] [postdirAbbrev] [internal] [location] [stateAbbrev] [zip]

1. **address** 는 정수입니다. 도로번호
2. **predirAbbrev** 는 Varchar 입니다. N,S,E,W 와 같은 도로의 방향 전치사 입니다. `direction_lookup` 테이블로 사용이 제어 됩니다.
3. **streetNumber** Varchar
4. **streetTypeAbbrev** 도로 형태의 축약 버전: e.g. St, Ave, Cir. street_type_lookup 테이블로 사용이 제어됩니다.
5. **postdirAbbrev** Varchar N,S,E,W 등과 같은 도로 표면의 방향성 약어. direction_lookup 테이블로 사용이 제어됩니다.
6. **internal** Varchar 아파트 또는 좌석번호 같은 내부 주소입니다.
7. **location** Varchar 일반적으로 도시 또는 지방도시
8. **stateAbbrev** Varchar 두 개의 문자로 된 미연방 주. e.g. MA, NY, ML state_lookup 테이블로 제어됩니다.
9. **zip** varchar 다섯 개의 숫자로 된 우편번호. e.g. 02109
10. **parsed** boolean - 정규화 프로세스로부터 정규화 되었다면 나타냅니다. nomalize_address 기능은 주소를 반환 하기 전에 true 로 설정합니다.

Examples

출력 필드를 선택합니다. 고급 텍스트 출력을 원한다면 Pprint_Addy 기능을 사용합니다.

```
SELECT address As orig, (g.na).streetname, (g.na).streettypeabbrev
FROM (SELECT address, normalize_address(address) As na
FROM addresses_to_geocode) As g;
orig | streetname | streettypeabbrev
```

```
----------------------------------------------------+--------------+---
---------------
28 Capen Street, Medford, MA | Capen | St
124 Mount Auburn St, Cambridge, Massachusetts 02138 | Mount Auburn | St
950 Main Street, Worcester, MA 01610 | Main | St
529 Main Street, Boston MA, 02129 | Main | St
77 Massachusetts Avenue, Cambridge, MA 02139 | Massachusetts | Ave
25 Wizard of Oz, Walaford, KS 99912323 | Wizard of Oz |
```

See Also

Geocode, Pprint_Addy

12.1.11. Pprint_Addy

Pprint_Addy — norm_addy 형태의 복합적인 객체를 입력 받아 고급 인쇄 표현을 반환 합니다. 보통 normalize_address 와 함께 사용됩니다.

Synopsis

varchar **pprint_addy**(norm_addy in_addy);

Description

norm_addy 형태의 복합적인 객체를 입력 받아 인쇄 표현을 반환 합니다. 지오코더 패키지와 함께 제공된 데이터 이외의 것은 필요로 하지 않습니다. Normalize_address 와 함께 사용됩니다.

Examples

단일 주소 인쇄

```
SELECT pprint_addy(normalize_address('202 East Fremont Street, Las Vegas,
Nevada 89101')) -
As pretty_address;
pretty_address
----------------------------------------
202 E Fremont St, Las Vegas, NV 89101
```

주소 테이블의 주소인쇄

```
SELECT address As orig, pprint_addy(normalize_address(address)) As
pretty_address
FROM addresses_to_geocode;
orig | pretty_address
-----------------------------------------------------+------------------
----------------------  -
529 Main Street, Boston MA, 02129 | 529 Main St, Boston MA, 02129
```

```
77 Massachusetts Avenue, Cambridge, MA 02139 | 77 Massachusetts Ave,
Cambridge, MA -
02139
28 Capen Street, Medford, MA | 28 Capen St, Medford, MA
124 Mount Auburn St, Cambridge, Massachusetts 02138 | 124 Mount Auburn
St, Cambridge, MA -
02138
950 Main Street, Worcester, MA 01610 | 950 Main St, Worcester, MA 01610
```

See Also

Normalize_Address

12.1.12. Reverse_Geocode

Reverse_Geocode — 알려진 공간 참조 모델(known spatial ref sys)의 지오메트리 포인트을 받아 이론적으로 가능한 주소의 배열과 교차로의 배열을 포함하는 레코드를 반환합니다. include_strnum_range = true 이면 도로간 교차점의 도로 범위를 포함합니다.

Synopsis

record **Reverse_Geocode**(geometry pt, boolean include_strnum_range=false, geometry[] OUT intpt, norm_addy[] OUT addy, varchar[] OUT street);

Description

알려진 공간 참조 모델(known spatial ref sys)의 지오메트리 포인트을 받아 이론적으로 가능한 주소의 배열과 교차로의 배열을 포함하는 레코드를 반환합니다. include_strnum_range = true 이면 도로간 교차점의 도로 범위를 포함합니다. include_strnum_range 의 기본값은 false 입니다. 첫 번째 주소와 가장 가까운 도로의 포인트 순서에 따라 정렬됩니다.

왜 우리가 실제 주소 대신 이론적인 주소라 말할까요? Tiger 데이터는 실제 주소를 가지고 있지 않고 도로의 범위를 가지고 있습니다. 그래서 이론적 주소는 도로 범위에 기반을 둔 보간 주소입니다. 보간된 주소를 예를 들면 26 Court St. 과 26 Court Sq. 26 Court Sq 와 같은 주소는 존재하지 않습니다. 그 이유는 두 개의 도로의 코너에 위치한 점과 두 도로의 보간 논리 때문입니다. 이 논리는 도시건물을 선택할 때 건물이 도로범위의 한 부분을 차지하고 나머지 부분이 끝에 군집될 때 잘못된 절차가 될 수 있어 도로에 위치한 동등한 공간범위로 주소를 추측합니다.

Note : 이 기능은 Tiger 데이터에 의존합니다. 만약 이 지점을 커버하는 데이터를 로드 하지 않을 경우 NULL 값을 반환 받을 것입니다.

반환되는 요소는 다음과 같습니다:

1. **intpt** 포인트의 배열: 입력 포인트에서 가장 가까운 도로의 중심선 포인트입니다. 주소 만큼 많은 포인트가 존재합니다.

2. **addy** 는 norm_addy(정규화된 주소)의 배열입니다. : 입력 포인트에 맞는 가능한 주소의 배열입니다. 배열의 첫 번째 것이 가장 정확성이 높습니다. 일반적으로 두세 개의 도로 코너에 있는 포인트 또는 도로의 가장자리나 끝이 아닌 곳에 위치하는 포인트를 제외하고는 하나만 존재합니다.

3. **street** 은 varchar 의 배열입니다. : 교차로 입니다.(또는 도로) (겹치는 도로 도는 포인트가 위치한 도로)

Availability: 2.0.0

Examples

두 개의 도로의 코너에 위치하지만 하나의 도로에 가장 가까운 포인트의 예 입니다.

이것은 MIT 의 대략적인 위치입니다. : 77 Massachusetts Ave, Cambridge, MA 02139

3 개의 도로를 가지고 있지 않음에도 불구하고 PostgreSQL 은 사용 안정성의 상한선을 넘지 않고 NULL 을 반환합니다.

```
SELECT pprint_addy(r.addy[1]) As st1, pprint_addy(r.addy[2]) As st2,
pprint_addy(r.addy[3]) As st3,
            array_to_string(r.street, ',') As cross_streets
FROM                    reverse_geocode(ST_GeomFromText('POINT(-71.093902
42.359446)',4269),true) As r
;
result
------
st1                                        |st2|st3| cross_streets
-------------------------------------------+---+---+-----------------
67 Massachusetts Ave, Cambridge, MA 02139|   |   | 67 - 127 Massachusetts
Ave,32 - 88 Vassar St
```

교차하는 도로의 주소범위를 포함하지 않고 실제로 두 개의 도로가 끝나는 곳을 취할 경우 두 개의 서로 다른 주소를 알 수 있습니다.

```
SELECT pprint_addy(r.addy[1]) As st1, pprint_addy(r.addy[2]) As st2,
pprint_addy(r.addy[3])   As   st3,   array_to_string(r.street,   ',')   As
cross_str
FROM  reverse_geocode(ST_GeomFromText('POINT(-71.06941   42.34225)',4269))
As r;
result
--------
st1                                 | st2                          | st3 |
cross_str
------------------------------------+------------------------------+-----
+------------------------  -
5 Bradford St, Boston, MA 02118 | 49 Waltham St, Boston, MA 02118 | |
Waltham St
```

이번 예는 Geocode를 통해 지오코드된 예제를 재사용하고 두 개 도로의 교차로와 주 주소만을 원할 경우 입니다.

```
SELECT actual_addr, lon, lat, pprint_addy((rg).addy[1]) As int_addr1,
    (rg).street[1] As cross1, (rg).street[2] As cross2
  FROM (SELECT address As actual_addr, lon, lat,
    reverse_geocode( ST_SetSRID(ST_Point(lon,lat),4326) ) As rg
    FROM addresses_to_geocode WHERE rating > -1) As foo;

actual_addr | lon | lat | int_addr1 | cross1 | cross2
------------------------------------------------------+-----------+-------
---+-----------------------------------------+------
529 Main Street, Boston MA, 02129 | -71.07181 | 42.38359 | 527 Main St,
Boston, MA 02129 | Medford St |
77 Massachusetts Avenue, Cambridge, MA 02139 | -71.09428 | 42.35988 | 77
Massachusetts Ave, Cambridge, MA 02139 | Vassar St |
26 Capen Street, Medford, MA | -71.12377 | 42.41101 | 9 Edison Ave,
Medford, MA 02155 | Capen St | Tesla Ave
124 Mount Auburn St, Cambridge, Massachusetts 02138 | -71.12304 |
42.37328 | 3 University Rd, Cambridge, MA 02138 | Mount Auburn St |
950 Main Street, Worcester, MA 01610 | -71.82368 | 42.24956 | 3 Maywood
St, Worcester, MA 01603 | Main St | Maywood Pl
```

See Also

Pprint_Addy, Geocode

12.1.13. Topology_Load_Tiger

Topology_Load_Tiger — PostGIS Topology 로 Tiger 데이터의 정의된 지역을 로드, 토폴로지의 좌표 참조 모델로 Tiger 데이터를 좌표 변환하고 토폴로지의 허용오차 정밀도로 스내핑 할 때 사용합니다.

Synopsis

text **Topology_Load_Tiger**(varchar topo_name, varchar region_type, varchar region_id);

Description

PostGIS Topology 로 Tiger 데이터의 정의된 지역을 로드합니다. faces, nodes, edges 를 대상 토폴로지의 좌표 참조 모델로 좌표 변환하고 포인트는 대상 토폴로지의 허용오차 내로 스내핑 합니다. 토폴로지의 좌표 참조 모델로 Tiger 데이터를 좌표 변환하고 토폴로지의 허용오차 정밀도로 스내핑 할 때 사용합니다.

Tiger 데이터의 faces, nodes, edges 와 같은 Ids 를 유지하여 faces, nodes, edges 를 생성 함으로서 추후에 Tiger 데이터와 더욱 쉽게 일치합니다. 프로세스에 대한 요약정보를 반환합니다.

새롭게 도로의 중심선을 따라 폴리곤을 형성하거나 중첩 없는 폴리곤 결과를 형성하기 위한 예로서 유용합니다.

이 기능은 Tiger 데이터뿐만 아니라 PostGIS Topology 설치에 의존합니다. 더 자세한 내용은 11 장과 2.4.1 단락을 참조 하십시요. 만약 관심 영역의 데이터가 로드 되지 않았다면 토폴로지 레코드는 생성되지 않을 것 입니다. 토폴로지 기능을 사용하여 토폴로지를 생성하지 않았다면 이 기능은 역시 실패 할 것입니다.

대부분의 토폴로지 유효성 검사 오류는 Edges Point 의 좌표변환 후 line 화 되지 않았거나 중복오류의 허용 오차 문제 입니다.

Required arguments:

1. **topo_name** 기존 PostGIS 토폴로지 이름에 데이터를 로드 합니다.
2. **region_type** 지역 경계의 유형입니다. 현재 place 와 county 를 지원하고 있습니다. 앞으로의 몇 개의 경우를 더 지원할 계획입니다. 정의된 지역의 경계를 볼 수 있는 테이블입니다. e.g. tiger.place, tiger.county
3. **region_id Tiger** 지오이드를 호출하는 기능입니다. 테이블에 존재하는 지역의 고유 식별자 입니다. 지역은 `tiger.place` 의 `plcidfp` 컬럼입니다. county 의 경우 `tiger.county` 에서 cntyidfp 컬럼입니다.

Availability: 2.0.0

Example: Boston, Massachusetts Topology

Boston, Massachusetts in Mass State Plane Feet (2249)를 0.25feet 의 허용 오차로 토폴로지를 생성하고 Boston city tiger faces, edges, nodes 로 로드합니다.

```
SELECT topology.CreateTopology('topo_boston', 2249, 0.25);
createtopology
--------------
15
-- 60,902 ms ~ 1 minute on windows 7 desktop running 9.1 (with 5 states
tiger data loaded)
SELECT tiger.topology_load_tiger('topo_boston', 'place', '2507000');
-- topology_loader_tiger --
29722 edges holding in temporary. 11108 faces added. 1875 edges of faces
added. 20576 nodes added.
19962 nodes contained in a face. 0 edge start end corrected. 31597 edges
added.

-- 41 ms --
SELECT topology.TopologySummary('topo_boston');
-- topologysummary--
```

```
Topology topo_boston (15), SRID 2249, precision 0.25
20576 nodes, 31597 edges, 11109 faces, 0 topogeoms in 0 layers

-- 28,797 ms to validate yeh returned no errors --
SELECT * FROM
       topology.ValidateTopology('topo_boston');
error               | id1      | id2
--------------------+----------+-----------
```

Example: Suffolk, Massachusetts Topology

Suffolk, Massachusetts in Mass State Plane Meters (26986) 를 0.25feet 의 허용 오차로 토폴로지를 생성하고 Suffolk county tiger faces, edges, nodes 로 로드합니다.

```
SELECT topology.CreateTopology('topo_suffolk', 26986, 0.25);
-- this took 56,275 ms ~ 1 minute on Windows 7 32-bit with 5 states of
tiger loaded
-- must have been warmed up after loading boston
SELECT tiger.topology_load_tiger('topo_suffolk', 'county', '25025');
-- topology_loader_tiger --
36003 edges holding in temporary. 13518 faces added. 2172 edges of faces
added.
24761 nodes added. 24075 nodes contained in a face. 0 edge start end
corrected. 38175 edges added.
-- 31 ms --
SELECT topology.TopologySummary('topo_suffolk');
-- topologysummary--
Topology topo_suffolk (14), SRID 26986, precision 0.25
24761 nodes, 38175 edges, 13519 faces, 0 topogeoms in 0 layers
-- 33,606 ms to validate --
SELECT * FROM
topology.ValidateTopology('topo_suffolk');
error               | id1       | id2
--------------------+----------+-----------
coincident nodes   | 81045651 | 81064553
edge crosses node | 81045651 | 85737793
edge crosses node | 81045651 | 85742215
edge crosses node | 81045651 | 620628939
edge crosses node | 81064553 | 85697815
edge crosses node | 81064553 | 85728168
edge crosses node | 81064553 | 85733413
```

See Also

CreateTopology, CreateTopoGeom, TopologySummary, ValidateTopology

Chapter 13. PostGIS 의 특별 기능 인덱스

13.1. PostGIS 의 집계함수

아래에 주어진 함수들은 PostGIS 에서 제공하는 공간 집계함수입니다. 이 함수들은 다른 sql 집계함수 – 합, 평균 등- 처럼 쓰일 수 있습니다.

ST_3DExtent — 지오메트리의 행을 제한하는 box3D 바운딩박스를 반환하는 집계함수입니다.

ST_Accum — 집계. 지오메트리의 배열을 구축합니다.

ST_Collect – 다른 지오메트리 집합에서 지정된 ST_Geometry 값을 반환합니다. 반환되는 유형은 Multi* 또는 GeometryCollection 지오메트리입니다.

ST_Extent — 지오메트리의 행을 제한하는 바운딩박스를 반환하는 집계함수입니다.

ST_MakeLine — Point 또는 Line 지오메트리를 이용하여 LineString 을 생성합니다.

ST_MemUnion — ST_Union 함수와 같은 결과를 반환하지만, 단지 메모리 친화적(더 작은 메모리와 더 많은 프로세서 시간을 사용)입니다.

ST_Polygonize — 주어진 지오메트리 집합의 모든 선을 이용하여 형성한 Polygon 들을 포함하는 GeometryCollection 을 생성합니다.

ST_Union — 입력된 지오메트리의 합집합 지오메트리를 반환하며, 결과 지오메트리는 Multi* 지오메트리, 단일 지오메트리 또는 GeometryCollection 일 수 있습니다.

ST_Union — 1 밴드로 구성된 하나의 래스터로 래스터 타일의 집합의 합집합을 반환합니다. 만약 어떤 밴드도 결합하기 위한 명시가 되어있지 않으면, num 1 밴드라고 가정합니다. 그 결과 래스터의 범위는 전체 집합의 범위입니다. 인터섹션의 경우에, 그 결과값은 p_expression 에 의해 정의되는데, 다음 중에 하나입니다: LAST - 아무것도 명시되어있지 않을 경우, MEAN, SUM, FIRST, MAX, MIN.

TopoElementArray_Agg — element_id 와 type arrays(topoelements)집합을 위한 topoelementarray 를 반환합니다.

13.2. PostGIS SQL-MM 준수 함수

아래에 주어진 함수들은 SQL/MM 3 의 기준을 준수하는 PostGIS 함수입니다.

SQL-MM 은 지오메트리 생성자의 SRID 기본값을 0 으로 정의합니다. PostGIS 는 SRID 의 기본값을 -1 로 합니다.

ST_3DDWithin – 3D(z) Geometry 유형에 대해서 두 지오메트리의 3 차원 공간에서의 거리가 지정된 거리 내에 있는 경우 TRUE 를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL—MM ?

ST_3DDistance — Geometry 유형에 대해서 두 지오메트리 사이의 3 차원 직교 최단거리(공간 좌표계를 기준으로 한 단위)를 반환합니다(spatial ref 참조). 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM ?

ST_3DIntersects — 두 지오메트리가 3 차원 공간에서 "공간적으로 교차"하는 경우 TRUE 를 반환합니다. Point 및 LineString 만을 대상으로 합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3: ?

ST_AddEdgeModFace — 새로운 edge 를 추가하고, 이로 인해 면이 분할되는 경우, 원래의 면을 수정하고 새로운 면을 하나 추가합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.13

ST_AddEdgeNewFaces – 새로운 edge 를 더하고, 이로 인해 면이 분할되는 경우, 원래의 면을 삭제하고 두 개의 새로운 면들로 대체합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.12

ST_AddIsoEdge — alinestring 라는 지오메트리에 의해 정의되는 독립된 에지를 기존에 존재하는 두 개의 고립된 노드 anode 와 anothernode 를 연결하는 토폴로지에 추가합니다. 그리고 새로운 에지의 에지 id 를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details: X.3.4

ST_AddIsoNode — 토폴로지의 면에 고립된 노드를 추가하고 새 노드의 nodeid 를 반환합니다. 면이 null 이라도 노드는 여전히 생성됩니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM: Topo-Net Routines: X+1.3.1

ST_Area — 지오메트리가 Polygon 또는 MultiPolygon 인 경우 표면의 면적을 반환합니다. "Geometry" 유형은 SRID 단위, "Geography" 유형은 평방 미터 단위입니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 8.1.2, 9.5.3

ST_AsBinary — SRID 메타 데이터가 없는 래스터의 바이너리 정보 (WKB: WELL KNOWN binary 형식)를 반환하여 나타냅니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.37

ST_AsText — Geometry/Geography 객체를 SRID 메타데이터를 포함하지 않은 Well-Known Text(WKT) 포맷으로 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.25

ST_Boundary — 주어진 지오메트리에 대한 폐쇄된 조합 경계를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.14

ST_Buffer — (T) Geometry 유형일 경우: 이 지오메트리로부터의 거리가 같거나 작은 모든 점들을 포함하는 지오메트리를 반환합니다. 계산은 지오메트리의 공간 좌표계를 따릅니다. Geography 유형일 경우: 평면 변환 래퍼를 사용합니다. 1.5 버전 이후부터 버퍼 옵션으로 끝점, 꼭지점 처리에 대한 스타일을 설정할 수 있습니다.
buffer_style options: quad_segs=#, endcap=round|flat|square, join=round|mitre|bevel, mitre_limit=#.# 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.17

ST_Centroid — 지오메트리의 기하학적 중심(무게 중심)을 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 8.1.4, 9.5.5

ST_ChangeEdgeGeom — 토폴로지 구조에 영향을 주지 않으며 에지의 모양을 변경합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details X.3.6

ST_Contains — A 지오메트리 외부에 B 지오메트리의 그 어떤 점도 놓여있지 않거나, 적어도 B 지오메트리 내부에서 하나의 점만이 A 지오메트리 내부에 놓여 있을 때 TRUE 를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.31

ST_ConvexHull — 지정된 지오메트리 집합을 모두 둘러싸는 최소 볼록 폐곡선(Convex Hull)을 Polygon 지오메트리로 반환합니다.

ST_CoordDim — ST_Geometry 값에 대한 지오메트리의 좌표 차수(XY 는 2, XYZ 는 3 등)를 반환합니다. 이 함수는 SWL/MM 사양 중 ST_NDims 의 별칭입니다 SQL-MM 3: 5.1.3

ST_CreateTopoGeo — 주어진 빈 토폴로지에 지오메트리의 컬렉션을 추가하고 성공시 자세한 설명 메시지를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM: Topo-Geo and Topo-Net 3: Routine Details -- X.3.18

ST_Crosses — 제공된 지오메트리가 공통의 내부(interior) 점들을 일부(전부는 아닌) 가지고 있을 경우 TRUE 를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.29

ST_CurveToLine — CIRCULARSTRING/CURVEDPOLYGON 지오메트리를 LINESTRING/POLYGON 로 변환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 7.1.7

ST_Difference — 지오메트리 B 와 교차하지 않는 지오메트리의 A 의 일부를 나타내는(차집합) 지오메트리를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.20

ST_Dimension — 좌표 차수보다 작거나 반드시 같아야 하는 이 지오메트리 객체의 고유 차원을 반환합니다. Point 는 0, LineString 은 1, Polygon 은 2 를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.2

ST_Disjoint — 지오메트리가 서로 "공간적으로 교차"하지 않을 때, 어떤 공간도 함께 공유하지 않는다면 TRUE 를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.26

ST_Distance — Geometry 유형에 대해서 두 지오메트리 사이의 2 차원 직교 최단거리(공간 좌표계를 기준으로 한 단위)를 반환합니다. Geography 유형에 대해서는 기본적으로 미터로 표현된 두 Geography 사이의 타원체 최단거리를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.23

ST_EndPoint — LineString 지오메트리의 끝점을 Point 로 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 7.1.4

ST_Envelope — 제공된 지오메트리의 double precision(float8) 최소경계영역 사각형(bounding box)을 나타내는 지오메트리를 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.15

ST_Equals — 주어진 지오메트리가 동일한 지오메트리를 나타내는 경우에 TRUE 를 반환하고 방향성은 무시합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.24

ST_ExteriorRing — Polygon 지오메트리의 외부 링(Exterior Ring)을 나타내는 LineString 을 반환합니다. 지오메트리가 Polygon 이 아닌 경우 NULL 값을 반환합니다. MultiPolygon 은 작동하지 않습니다. MULTIPOLYGON 과는 작동하지 않을 것입니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 8.2.3, 8.3.3

ST_GMLToSQL — GML 표현에서 지정된 ST_Geometry 값을 반환합니다. 이 함수는 ST_GeomFromGML 의 별칭입니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.50 (곡선 지원).

ST_GeomCollFromText — 지정된 SRID 의 컬렉션 WKT 에서 컬렉션 지오메트리를 생성합니다. SRID 가 지정되지 않은 경우 -1 을 기본값으로 사용합니다. 이 방법은 SQL/MM 사양을 구현합니다.

ST_GeomFromText — Well-Known Text (WKT)에서 지정된 ST_Geometry 값을 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.40

ST_GeomFromWKB — Well-Known Binary (WKB)와 옵션으로 SRID 에서 geometry 인스턴스를 생성합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.41

ST_GeometryFromText — Well-Known Text (WKT)포맷에서 지정된 ST_Geometry 값을 반환합니다. 이것은이 방법은 MM / SQL 사양을 구현 ST_GeomFromText 의 별칭입니다. SQL-MM 3 : 5.1.40

ST_GeometryN — 지오메트리가 GEOMETRYCOLLECTION, (MULTI) POINT, (MULTI) LINE, MULTICURVE, 또는 (MULTI) POLYGON, POLYHEDRALSURFACE 경우 1 부터 N 번째 도형을 반환합니다. 그렇지 않으면 NULL 을 반환합니다 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 9.1.5

ST_GeometryType — ST_Geometry 값의 지오메트리 타입을 반환합니다. 예: 'ST_Linestring', 'ST_Polygon', 'ST_MultiPolygon' 등. 이 함수는 GeometryType 함수와는 다릅니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.4

ST_GetFaceEdges — aface 라는 면을 구성하는 순차적인 에지들의 집합을 순번과 함께 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.5

ST_GetFaceGeometry — 주어진 토폴로지 내의 폴리곤을 면 ID 와 함께 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.16

ST_InitTopoGeo — 신규 토폴로지 스키마를 작성하고 topology.topology 테이블과 프로세스의 세부 요약에 새 스키마를 등록합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 Topo-Geo and Topo-Net 3: Routine Details: X.3.17

ST_InteriorRingN — 폴리곤 지오메트리의 N 번째 내부 linestring 링을 LineString 으로 반환합니다. 지오메트리가 폴리곤이 아니거나 주어진 N 이 범위를 벗어나면 NULL 을 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3: 8.2.6, 8.3.5

ST_Intersection — (T) geomA 와 geomB 가 공유하는 부분(교집합)을 나타내는 지오메트리를 반환합니다. Geography 유형에서의 구현은 교차 수행을 위해 UTM 등으로 투영하고 처리 후 다시 WGS84 로 변환하여 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.18

ST_Intersects — Geometry/Geography 객체가 "2 차원 상에서 공간적으로 교차"(공간의 어느 부분을 공유)할 경우 TRUE 를 반환하고 그렇지 않을 경우(Disjoint) FALSE 를 반환합니다. Geography 유형에 대해서 0.00001 미터의 허용 오차(오차 내에 포함되면 서로 교차하는 것으로 간주)를 사용합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.27

ST_IsClosed — LineString 의 시작점과 끝점이 일치하는 경우 TRUE 를 반환합니다. 다면체 표면은 닫혀있습니다(volumetric). 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 7.1.5, 9.3.3

ST_IsEmpty — 지오메트리가 빈 GeometryCollection, Polygon, LineString, Point 등일 경우 TRUE 를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.7

ST_IsRing — 이 LineString 지오메트리가 닫혀(ST_IsClosed) 있고 단순(ST_IsSimple)한 경우 TRUE 를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 7.1.

ST_IsSimple — 이 지오메트리가 자기 교차 또는 자기 접선과 같이 변칙적인 지오메트리 포인트를 가지고 있지 않은 경우 TRUE 를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.8

ST_IsValid — ST_Geometry 가 잘 구성되었을 경우 ture 를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.9

ST_Length – 지오메트리가 LineString 또는 MultiLineString 일 경우 2 차원 상에서의 길이를 반환합니다. Geometry 유형은 공간 좌표계 단위를, Geography 유형은 미터 단위(회전 타원체 사용 기본값)를 사용합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 7.1.2, 9.3.4

ST_LineFromText — 지정된 SRID 및 WKT 에서 LineString 지오메트리를 생성합니다. SRID 가 지정되지 않은 경우 -1 을 기본값으로 사용합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 7.2.8

ST_LineFromWKB — 지정된 SRID 및 WKB 에서 LineString 을 생성합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 7.2.9

ST_LinestringFromWKB — 지정된 SRID 및 WKB 에서 LineString 을 생성합니다. 이 함수는 ST_LineFromWKB 의 별칭입니다. SQL-MM 3 : 7.2.9

ST_M — Point 의 M 값을 반환하거나 없는 경우에는 NULL 값을 반환합니다. 입력은 반드시 Point 이어야 합니다. 이 방법은 MM / SQL 사양을 구현합니다.

ST_MLineFromText — WKT 표현에서 지정된 ST_MultiLineString 값을 반환합니다. 9.4.4 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3: 9.4.4

ST_MPointFromText — 지정된 SRID 와 WKT 로부터 지오메트리을 만듭니다. SRID 가 주어지지 않은 경우, -1 을 기본값으로 사용합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 9.2.4

ST_MPolyFromText — 지정된 SRID 와 WKT 로부터 멀티폴리곤 지오메트리을 만듭니다. SRID 이 주어지지 않은 경우 -1 을 기본값으로 사용합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 9.6.4

ST_ModEdgeHeal — 두 개의 에지를 연결하는 노드를 삭제하여 치료합니다. 첫 번째 에지를 수정하고 두 번째 에지를 삭제합니다. 삭제된 노드의 ID 를 반환합니다. 이것은 SQL/MM 사양을 구현합니다. SQL-MM: Topo-Geo 그리고 Topo-Net 3: Routine Details: X.3.9

ST_ModEdgeSplit — 기존의 에지를 따라 새 노드를 만들고 원래의 에지를 수정하며, 새로운 에지를 추가하여 에지를 분할합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM: Topo-Geo 및 Topo-Net 3: Routine Details: X.3.9

ST_MoveIsoNode — 한 점에서 다른 점으로 토폴로지에 고립 된 노드를 이동합니다. 새로운 apoint 라는 지오메트리가 노드로 존재하는 경우 오류를 발생시킵니다. 이동에 대한 설명을 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다.. SQL-MM: Topo-Net Routines: X.3.2

ST_NewEdgeHeal — 두 개의 에지를 연결하는 노드를 삭제하여 치료합니다. 두 에지를 모두 삭제하고 삭제된 에지들을 첫 번째 주어졌던 에지와 같은 방향의 에지로 대체합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM: Topo-Geo 및 Topo-Net 3: Routine Details: X.3.9

ST_NewEdgesSplit — 기존의 에지 위에 새 노드를 만들고, 원래의 에지를 삭제한 뒤 이를 두 개의 새로운 에지로 교체하여 에지를 분할합니다. 두 에지가 합쳐진 새로운 에지의 ID 를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM: Topo-Net Routines: X.3.8

ST_NumGeometries — 지오메트리가 GEOMETRYCOLLECTION (또는 MULTI *) 인 경우, 지오메트리의 수를 반환합니다. 단일 지오메트리는 1 을 반환, 없는 경우는 NULL 값을 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 9.1.4

ST_NumInteriorRing — 지오메트리에서 첫 번째 다각형의 내부 링(Interior Ring)의 개수를 반환합니다. 이 함수는 ST_NumInteriorRings 의 별칭입니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3

ST_NumInteriorRings — 지오메트리에서 첫 번째 다각형의 내부 링(Interior Ring)의 개수를 반환합니다. Polygon 및 MultiPolygon 두 가지 유형 모두 사용 가능하지만 지오메트리의 첫 번째 Polygon 을 사용합니다. 지오메트리에 Polygon 이 없는 경우 NULL 값을 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 8.2.5

ST_NumPatches — PolyhedralSurface 의 면(faces)의 개수를 반환합니다. Polyhedral 지오메트리가 아닌 경우 NULL 값을 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 :?

ST_NumPoints — ST_LineString 또는 ST_CircularString 객체의 점(정점)의 개수를 반환합니다. ST_NPoints 의 별칭이며, LineString 지오메트리가 아닌 경우 ST_NPoints 함수를 사용하십시오. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 7.2.4

ST_OrderingEquals — 주어진 지오메트리가 동일한 지오메트리(ST_Equals)를 나타내고 구성하는 점들의 순서가 동일한 경우 TRUE 를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.43

ST_Overlaps — 주어진 지오메트리가 공간을 공유하고 동일한 차원이지만 서로 완전하게 포함되지 않을 경우 TRUE 를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.32

ST_PatchN — PolyhedralSurface 또는 PolyhedralSurfaceM 지오메트리인 경우 처음을 1 부터 시작하는 N 번째 지오메트리(Face)를, 그렇지 않으면 NULL 값을 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 :?

ST_Perimeter — ST_Surface 또는 ST_MultiSurface 지오메트리(Polygon, MultiPolygon)의 둘레 길이를 반환합니다. Geometry 유형은 공간 좌표계 단위를, Geography 유형은 미터 단위를 사용합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 8.1.3, 9.5.4

ST_Point — 주어진 좌표 값을 이용하여 ST_Point 값을 반환합니다. ST_MakePoint 의 OGC 별칭입니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 6.1.2

ST_PointFromText — 지정된 SRID 와 WKT 에서 포인트 지오메트리를 만듭니다. SRID 가 지정되지 않은 경우 기본값은 unknown 입니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 6.1.8

ST_PointFromWKB — 지정된 SRID 의 WKT 에서 지오메트리를 만듭니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 6.1.9

ST_PointN — LineString 또는 CircularLineString 지오메트리인 경우 처음을 1 부터 시작하는 N 번째 지오메트리를, 그렇지 않으면 NULL 값을 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 7.2.5, 7.3.5

ST_PointOnSurface — 표면에 반드시 위치하는 Point 지오메트리를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3: 8.1.5, 9.5.6. 사양에 따르면, ST_PointOnSurface 는 표면 지오메트리 (POLYGON, MULTYPOLYGONS, CURVED POLYGONS)에 대해 작동합니다. 따라서 PostGIS 은 스펙이 허용하는 것을 확장 할 것으로 보입니다. 대부분의 데이터베이스 Oracle, DB II, ESRI SDE 는 오직 surface 를 위한 기능만을 지원하는 것처럼 보입니다. PostGIS 와 같은 SQL Server 2008 은 모든 일반적인 지오메트리에 대해 지원합니다.

ST_Polygon — 아무 데이터 값이 아닌 픽셀 값을 가진 픽셀의 조합으로 형성된 다각형 지오메트리를 반환합니다. 1 에 아무 밴드 번호가 지정되지 않은 경우, 밴드 숫자 기본값으로 사용합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 8.3.2

ST_PolygonFromText — 지정된 SRID 와 WKT 에서 지오메트리를 생성합니다. SRID 가 지정되지 않은 경우 -1 을 기본값으로 사용합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 8.3.6

ST_Relate — 이 지오메트리가 공간적으로 intersectionMatrixPattern 의 값에 의해 지정된 두 기하학의 내부, 경계와 외부 사이의 교차점에 대한 테스팅을 통해, anotherGeometry 에 관련되어있는 경우 True 를 반환합니다. 아무 intersectionMatrixPattern 가 전달되지 않은 경우 2 지오메트리에 관련된 최대 intersectionMatrixPattern 을 반환합니다. 이 방법은 SQL/MM 사양을 구현합니다. SQL-MM 3: 5.1.25

ST_RemEdgeModFace — 에지를 제거하고, 만약 제거된 에지가 두 면을 분리하고 있었다면, 면 중 하나를 삭제하고 다른 하나의 면을 나머지 영역을 포함하도록 수정합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM : Topo-Geo 그리고 Topo-Net 3: Routine Details: X.3.15

ST_RemEdgeNewFace — 에지를 제거하고, 만약 제거된 에지가 두 면을 분리하고 있었다면, 원래의 면들을 삭제하고 새로운 한 면으로 대체합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM : 토포 - : Topo-Geo 그리고 Topo-Net 3: Routine Details: X.3.15

ST_RemoveIsoNode — 고립 된 노드를 제거하고 동작에 대한 설명을 반환합니다. 노드가 고립되지 않는 경우 (노드의 시작 또는 끝), 예외가 발생됩니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM : Topo-Geo 및 Topo-Net 3: Routine Details: X+1.3.3

ST_SRID — spatial_ref_sys 테이블에 정의 된 ST_Geometry 를위한 공간 레퍼런스 식별자를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.5

ST_StartPoint — LINESTRING 지오메트리의 첫 번째 점을 POINT 형태로 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 7.1.3

ST_SymDifference — 지오메트리 A 와 지오메트리 B 가 교차하지 않는 부분의 지오메트리를 반환합니다. ST_SymDifference (A, B) = ST_SymDifference (B, A) 이기 때문에 이는 대칭차(symmetric difference)라고 불립니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.21

ST_Touches — 지오메트리가 최소한 하나 이상의 공통 지점을 가지고 있고 그 내부가 서로 교차하지 않는 경우 TRUE 를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.28

ST_Transform — 정수 매개변수에 의해 참조되는 SRID 값으로 좌표변환을 수행한 지오메트리를 반환합니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.6

ST_Union — 입력된 지오메트리의 합집합 지오메트리를 반환하며, 결과 지오메트리는 Multi* 지오메트리, 단일 지오메트리 또는 GeometryCollection 일 수 있습니다. 이 방법은 MM / SQL 사양을 구현합니다. 다각형들이 포함될 때 SQL-MM 3: 5.1.19 the z—index (고도).

ST_WKBToSQL – Well-Known Binary (WKB)에서 지정된 ST_Geometry 를 값을 반환합니다. 이것은 SRID 를 갖지 않은 ST_GeomFromWKB 의 별칭입니다. 이 방법은 MM / SQL 사양을 구현합니다. SQL-MM 3 : 5.1.36.

ST_WKTToSQL — WKT 에서 지정된 ST_Geometry 값을 반환합니다. 이것은 ST_GeomFromText 의 별칭 이름입니다. 이 방법은 MM / SQL 사양을 구합니다. SQL-MM 3 : 5.1.34.

ST_Within — A 지오메트리가 완전하게 B 지오메트리 안에 있을 경우 TRUE 를 반환합니다. This method implements the SQL/MM specification. SQL-MM 3: 5.1.30

ST_X – Point 지오메트리의 X 좌표 또는 사용할 수 없는 경우 NULL 값을 반환합니다. 입력은 반드시 Point 이어야 합니다. 이 방법은 SQL / MM 사양을 구현합니다. SQL-MM 3 : 6.1.3

ST_Y — Point 지오메트리의 Y 좌표 또는 사용할 수 없는 경우 NULL 값을 반환합니다. 입력은 반드시 Point 이어야 합니다. 이 방법은 SQL / MM 사양을 구현합니다. SQL-MM 3: 6.1.4

ST_Z — Point 지오메트리의 Z 좌표 또는 사용할 수 없는 경우 NULL 값을 반환합니다. 입력은 반드시 Point 이어야 합니다. 이 방법은 SQL / MM 사양을 구현합니다.

13.3. PostGIS 지오그래피 지원 함수

아래의 함수 그리고 연산자는 geography data type 오브젝트를 입력으로 받거나 또는 출력으로 반환하는 PostGIS 함수/연산자 입니다.

(T)의 함수는 기본 측지 기능(geodetic functions)을 하지 않습니다, 그리고 작업을 수행하는 지오메트리에서 ST_Transform 호출을 사용합니다. 그 결과 날짜 변경선, 남/북극으로 갈 때 그리고 큰 지오메트리 또는 두개 이상의 UTM 존(ZONE)을 커버하는 지오메트리 쌍의 경우 예상대로 작동하지 않을 수도 있습니다. 기본 tranform - (UTM, 램버트 Azimuthal (북 / 남)을 선호하고 최악의 경우 메르카토르)

ST_Area — 지오메트리가 Polygon 또는 MultiPolygon 인 경우 표면의 면적을 반환합니다. "Geometry" 유형은 SRID 단위, "Geography" 유형은 평방 미터 단위입니다.

ST_AsBinary — Geometry/Geography 객체를 SRID 메타데이터를 포함하지 않은 Well-Known Binary(WKB) 포맷으로 반환합니다.

ST_AsEWKT — SRID 메타 데이터를 포함한 Well-known Text (WKT) 포맷으로 지오메트리를 반환합니다.

ST_AsGML — 지오메트리를 GML(Geography Markup Language) 버전 2(2.1.2, 기본값) 또는 버전 3(3.1.1) 포맷으로 반환합니다.

ST_AsGeoJSON — 지오메트리를 GeoJSON(Geometry Javascript Object Notation) 포맷으로 반환합니다.

ST_AsKML — 지오메트리를 KML(Keyhole Markup Language) 포맷으로 반환합니다. 몇 가지 변형입니다. Default version=2, default precision=15

ST_AsSVG — geometry 또는 geography 개체에 주어진 SVG 경로 데이터의 지오메트리를 반환합니다.

ST_AsText — SRID 메타 데이터가 없는 Well-known Text (WKT) 포맷으로 지오메트리 / 지오그래피를 반환합니다.

ST_Azimuth — pointA 에서 poinB 로 수직으로 시계 방향으로 측정한 북쪽 기반의 방위각을 라디안 단위의 각도로 반환합니다.

ST_Buffer — (T) Geometry 유형일 경우: 이 지오메트리로부터의 거리가 같거나 작은 모든 점들을 포함하는 지오메트리를 반환합니다. 계산은 지오메트리의 공간 좌표계를 따릅니다. Geography 유형일 경우: 평면 변환 래퍼를 사용합니다. 1.5 버전 이후부터 버퍼 옵션으로 끝점, 꼭지점 처리에 대한 스타일을 설정할 수 있습니다. buffer_style 옵션들: quad_segs=#,endcap=round|flat|square,join=round|mitre|bevel,mitre_limit=#.#

ST_CoveredBy – 지오메트리/지오그래픽 A 의 포인트가 Geometry/Geography B 밖에 없는 경우 1 (TRUE)을 반환합니다.

ST_Covers – 지오메트리 B 의 포인트가 지오메트리 A 의 밖에 없을 경우 1 (TRUE)을 반환합니다.

ST_DWithin - 지오메트리가 서로간에 지정된 거리 내에 있는 경우에 TRUE 를 반환합니다. Geometry 유형일 경우 단위는 공간 좌표계를 기준으로 하고, Geography 일 경우 단위는 미터를 사용하며 측정은 use_spheroid=true(회전 타원체 사용)가 기본값이고, 빠른 처리를 위해서 use_spheroid=false 옵션을 사용하면 됩니다.

ST_Distance — Geometry 유형에 대해서 두 지오메트리 사이의 2 차원 직교 최단거리(공간 좌표계를 기준으로 한 단위)를 반환합니다. Geography 유형에 대해서는 기본적으로 미터로 표현된 두 Geography 사이의 타원체 최단거리를 반환합니다.

ST_GeogFromText — WKT 또는 EWKT 에서 지정된 Geography 객체를 반환합니다. 이 함수는 ST_GeographyFromText 의 별칭입니다..

ST_GeogFromWKB — Well-Known Binary geometry (WKB) 포맷 또는 extended Well-Known Binary (EWKB) 포맷에서 geography 인스턴스를 만듭니다.

ST_GeographyFromText — WKT 또는 EWKT 에서 지정된 Geography 객체를 반환합니다. WGS84 경위도(EPSG 4326) 좌표계를 가정합니다.

= — A 의 바운딩 박스가 B 의 바운딩 박스와 같을 경우 TRUE 을 반환합니다. 바운딩 박스로 double precision 을 사용합니다.

ST_Intersection — (T) geomA 와 geomB 가 공유하는 부분(교집합)을 나타내는 지오메트리를 반환합니다. Geography 유형에서의 구현은 교차 수행을 위해 UTM 등으로 투영하고 처리 후 다시 WGS84 로 변환하여 반환합니다.

ST_Intersects — Geometry/Geography 객체가 "2 차원 상에서 공간적으로 교차"(공간의 어느 부분을 공유)할 경우 TRUE 를 반환하고 그렇지 않을 경우(Disjoint) FALSE 를 반환합니다. Geography 유형에 대해서 0.00001 미터의 허용 오차(오차 내에 포함되면 서로 교차하는 것으로 간주)를 사용합니다.

ST_Length — 지오메트리가 LineString 또는 MultiLineString 일 경우 2 차원 상에서의 길이를 반환합니다. Geometry 유형은 공간 좌표계 단위를, Geography 유형은 미터 단위(회전 타원체 사용 기본값)를 사용합니다.

ST_Perimeter — ST_Surface 또는 ST_MultiSurface 지오메트리(Polygon, MultiPolygon)의 둘레 길이를 반환합니다. Geometry 유형은 공간 좌표계 단위를, Geography 유형은 미터 단위를 사용합니다.

ST_Project — 미터 단위의 거리 및 라디안 단위의 베어링(방위각) 각도를 사용하여 시작 지점에서 예상되는 Point 지오메트리를 반환합니다.

ST_Summary — 지오메트리의 정보를 문자열로 요약해서 반환합니다.

&& — — A 의 2D 바운딩 박스와 B 의 2D 바운딩 박스가 교차하는 경우 TRUE 를 반환합니다.

13.4. PostGIS 래스터 지원 함수

아래의 함수 그리고 연산자는 입력으로 받아지거나 또는 출력으로 raster 유형 오브젝트를 반환하는 PostGIS 함수/연산자 입니다. 알파렛 순서로 나열됩니다.

Box3D — 래스터의 폐합 된 도형정보를 3 차원 데이터 형태로 변환합니다

&< — A 의 바운딩박스가 B 의 바운딩박스 왼쪽에 있을 때 TRUE 를 반환합니다.

&> — A 의 바운딩박스가 B 의 바운딩 박스의 오른쪽에 있을 때 TRUE 을 반환합니다.

&& — A 의 바운딩박스가 B 의 바운딩 박스와 겹칠 때 TRUE 을 반환합니다.

ST_AddBand — 지정된 인덱스 위치에 지정된 초기 값으로 추가 지정된 형식의 새 밴드 (들)과 래스터를 반환합니다. 인덱스를 지정하지 않은 경우, 새롭게 추가된 밴드는 데이터 테이블의 레코드 끝에 추가됩니다.

ST_AsBinary — SRID 메타 데이터가 없는 Well-Known Binary (WKB)포맷으로 래스터를 반환합니다.

ST_AsGDALRaster — 지정된 GDAL 래스터 형식으로 래스터 타일을 반환합니다. 래스터 형식은 컴파일 된 라이브러리에서 지원하는 형식 중 하나입니다. ST_GDALRasters ()를 사용하여 라이브러리가 지원하는 형식의 목록을 가져올 수 있습니다.

ST_AsJPEG — JPEG 이미지 (바이트 배열)로 래스터 타일을 선택 밴드를 변환합니다. 아무런 밴드가 지정되지 않고 1 개 또는 3 개 이상 대역의 경우, 첫 번째 밴드가 사용됩니다. 만약 오직 3 개의 밴드일 경우 3 개의 밴드가 모두 사용되며 변환된 이미지는 RGB 값을 가진 컬러이미지 입니다.

ST_AsPNG — PNG 이미지 (바이트 배열)로 래스터 타일 선택된 밴드를 변환합니다. 래스터의 1, 3 또는 4 밴드가 지정되고 아무런 밴드가 지정되지 않으며, 모든 밴드가 사용됩니다. 만약 2 또는 4 개이상의 밴드가 지정되거나 아무런 밴드가 지정되지 않으면 밴드 1 만이 오직 사용됩니다. 밴드 RGB 또는 RGBA 형태로 이미지가 분류 됩니다.

ST_AsRaster — PostGIS geometry(편집자주: 벡터형태의 공간도형)를 PostGIS raster 로 변환합니다.

ST_AsTIFF — 래스터 선택된 밴드를 TIFF 이미지 (바이트 배열)로서 변환합니다. 아무런 밴드도 지정되지 않는다면 모든 밴드 정보를 이용합니다.

ST_Aspect — 고도 래스터 밴드의 지형의 방향성정보를 반환합니다. 지형을 분석하는 데 유용합니다.

ST_Band — 하나 또는 그 이상의 기존 래스터 밴드(들)를 새로운 래스터데이터로 변환되어 재구성합니다. 기존에 존재하는 래스터 정보들을 재구성하여 새로운 래스터로 구축하는데 유용합니다.

ST_BandIsNoData — 밴드가 nodata 값으로 채워질 경우 true 를 반환합니다.

ST_BandMetaData — 지정된 raster 밴드를 위한 기본 메타 데이터를 반환합니다. 지정되지 않은 경우 band num 1 로 간주됩니다.

ST_BandNoDataValue — 아무런 데이터도 나타내지 않는 특정 래스터 밴드 값을 반환합니다. 밴드가 없다면 숫자 1 로 간주합니다.

ST_BandPath — 파일 시스템에 저장된 밴드 시스템 파일 경로를 반환합니다. bandnum 이 지정되지 않으면 1 로 간주됩니다.

ST_BandPixelType — 지정된 밴드의 픽셀 형식을 반환합니다. Bandnum 이 지정되지 않으면 1 로 간주됩니다.

ST_Clip — 공간연산작업을 이용하여 입력된 벡터 도형공간 의해 절취하여 (편집자 주: 클리핑처리- 하나의 래스터를 일부 벡터도형이 점유하고 있는 경우 공동점유되는 교집합부분으로 잘라냄) 래스터를 변환합니다. 아무런 밴드도 지정되지 않았을 경우 모든 밴드가 반환됩니다. 만일 생성된래스터가 지정되어 있지않는 경우, true 는 출력래스터가 절취되었다고 의미하는 것으로 간주됩니다.

ST_ConvexHull – 래스터(점 선 면 등 가장 바깥)의 외곽선을 추출하여 변환합니다. 결과물은 볼록다각형과 같은 형태입니다. 규칙 적인 모양을 지니거나 경사가 없는 래스터들의 경우 ST_Envelope 와 같은 결과를 줄 것입니다, 그러므로 비규칙적이거나 경사가 있는 래스터들의 경우에만 유용합니다.

ST_Count — 래스터 또는 래스터 범위의 지정된 밴드의 픽셀 수를 반환합니다. 아무 밴드가 지정되지 않았다면 기본값은 1 입니다. 미리 정의되어 있는 밴드 정보가 true 로 설정되어 있는 NODATA 값과 같지 않은 픽셀만 계산합니다.

ST_DumpAsPolygons — 주어진 래스터 밴드의 geomval (geom, val) rows 의 집합을 반환합니다. 아무런 band num 이 지정되지 않은 경우 밴드 넘버는 1 로 디폴트 됩니다.

ST_Envelope — 래스터 픽셀 단위의 좌표범위를 반환합니다.

ST_GeoReference — 일반적으로 세계 지도파일(world file)에서 볼 수 있는 것처럼 GDAL 또는 ESRI 형식의 공간 참조 메타 데이터를 조회하여 반환합니다. 기본값은 GDAL 입니다.

ST_HasNoBand — 주어진 밴드 수를 가진 밴드가없는 경우에 true 를 돌려줍니다. 아무런 band num 이 지정되지 않은 경우, 밴드 번호 1 로 간주됩니다.

ST_Height — 픽셀 래스터의 높이를 반환합니다.

ST_HillShade — 제공 방위각, 고도, 밝기와 고도 가늠자 입력을 사용하여 고도 래스터 밴드의 가상 조명을 반환합니다. 지형을 시각화하는 데 유용합니다.

ST_Histogram — 래스터 또는 래스터 커버리지 데이터 분배 별도 빈 범위(raster coverage data distribution separate bin ranges)를 요약하는 히스토그램의 집합을 반환합니다. 지정되지 않을시 경우 빈은 숫자들은 자동적으로 계산됩니다.

ST_Intersection – 래스터 또는 두개의 래스터가 공유하는 부분 또는 래스터 및 지오메트리의벡터화의 기하하적 교차를 나타내는 지오메트리 pixelvalue 쌍의 집합을 반환합니다.

ST_Intersects — 래스터 공간적으로 분리 된 래스터 또는 지오메트리를 교차하는 경우 true 를 반환합니다. 밴드 수가 (또는 NULL 로 설정)를 없으면, 래스터의 기복 형태를 두고 기능이 실행됩니다. 대역 번호가 제공되는 경우, 값 (NODATA 아니라)을 가진 오직 다른 픽셀들이 대상정보로 간주되어 실행 됩니다.

ST_IsEmpty – 지오메트리가 빈 GeometryCollection, Polygon, LineString, Point 등일 경우 TRUE 를 반환합니다.

ST_MakeEmptyRaster — 좌상단 X 및 Y 픽셀 크기 및 이미지 내부의 화소 상의 기울기 (scaleX, scaley, skewx & skewy) 및 좌표참조 시스템 (SRID 가 부여된 좌표계정보), 주어진 치수(폭 및 높이) 등 기본적인 래스터 데이터를 구성할 수 있는 데이터를 입력하면 새로운 Raster 데이터 레이어가 postgis 에 생성됩니다. SRID 가 생략되는 경우, 공간 좌표참조 값은 기본값 (0)으로 설정됩니다.(편집자 주: 이것은 실제 데이터가 없는 래스터 레이어를 초기 생성하기 위한 기능입니다.)

ST_MapAlgebraExpr — 1 래스터 밴드 버전: 제공된 입력 래스터 밴드와 pixeltype 의 유효한 PostgreSQL 의 대수 연산을 적용하여 형성된 새로운 one band Raster 를 작성합니다. 아무런 밴드가 지정되지 않은 경우 band 1 로 간주됩니다.

ST_MapAlgebraExpr — 2 래스터 밴드 버전: 제공된 2 개 입력 래스터 밴드와 pixeltype 의 유효한 PostgreSQL 의 대수 연산을 적용하여 형성된 새로운 one band Raster 를 작성합니다. 아무런 밴드가 지정되지 않은 경우 대역 1 로 간주됩니다. 그 결과로 생긴 raster 는 첫번째 raster 에 의해 정의된 grid 위 에서 정렬되며 (눈금, 기울이기 및 픽셀 코너) "extenttype"매개 변수에 의해 정의된 범위를 가집니다. "extenttype"의 값은: INTERSECTION, UNION, FIRST, SECOND.

ST_MapAlgebraFct — 1 밴드 버전: 제공된 입력 래스터 밴드와 pixeltype 의 유효한 PostgreSQL 의 대수 연산을 적용하여 형성된 새로운 one band Raster 를 작성합니다. 아무런 밴드가 지정되지 않은 경우 band 1 로 간주됩니다

ST_MapAlgebraFct — 2 밴드 버전: 제공된 2 개 입력 래스터 밴드와 pixeltype 의 유효한 PostgreSQL 의 대수 연산을 적용하여 형성된 새로운 one band Raster 를 작성합니다. 아무런

밴드가 지정되지 않은 경우 band 1 로 간주됩니다. 만약 지정되지 않으면 Extent type 은 INTERSETION 으로 디폴트 됩니다.

ST_MapAlgebraFctNgb — 1 밴드 버전: Map Algebra Nearest Neighbor 사용자 정의 PostgreSQL 의 기능을 사용하는 입력 Raster 밴드의 값들의 이웃을 포함하는 PLPGSQL 유저 기능의 결과인 값들인 Raster 를 반환합니다.

ST_MetaData — pixel size, rotation(skew), upper, lower, left 등과 같은 raster 오브젝트에 관한 기본 메타 데이터를 반환합니다.

ST_NumBands — 래스터 객체를 구성하고 있는 밴드의 수를 반환합니다.

ST_PixelAsPolygon — 특정 행과 열의 픽셀을 제약하는 래스터 값을 지오메트리 정보형태로 반환합니다.

ST_PixelAsPolygons — 각 픽셀의 X 그리고 Y raster 좌표들과 값에 따른 raster 밴드의 모든 픽셀을 제한하는 지오메트리를 반환합니다.

ST_PixelHeight — 좌표를 참조하고 있는 래스터 단위 픽셀의 높이를 반환합니다

ST_PixelWidth — 좌표를 참조하고 있는 래스터 단위 픽셀 너비를 반환합니다.

ST_Polygon — 아무 데이터 값이 아닌 픽셀 값을 가진 픽셀의 조합으로 형성된 다각형 지오메트리를 반환합니다. 1 에 아무 밴드 번호가 지정되지 않은 경우, 밴드 숫자 기본값으로 사용합니다.

ST_Quantile — 견본 또는 인구의 맥락에서 래스터 또는 래스터 테이블 커버리지 대한 분위수를 계산합니다. 따라서, 값은 래스터의 25 %, 50 %, 75 %의 백분위에서 검사 될 수 있습니다.

ST_Raster2WorldCoordX — 래스터, 열 및 행의 기하학적 좌상단 X 좌표를 반환합니다. 열 및 행의 번호는 1 부터 시작합니다.

ST_Raster2WorldCoordY — 래스터, 열 및 행의 기하학적 좌상단 Y 좌표를 반환합니다. 열 및 행의 번호는 1 부터 시작합니다..

ST_Reclass — 원본으로부터 재분류된 밴드 형식으로 구성된 새로운 래스터. Nband 는 변경되는밴드입니다. nband 는 지정되지 않았을 경우 1 로 간주됩니다. 모든 다른 밴드들은 변하지 않은 상태에서 반환됩니다. 케이스 사용: 16BUI band 를 8BUI 로 전환 그리고 볼 수 있는 형식으로 간단하게 렌더링합니다.

ST_Resample – 다른 raster 에서 정의되거나 빌려진 raster geoferecing 속성의 집합 그리고 임의의 그리드 코너, new dimensions, 지정된 리샘플링 알고리즘을 이용하여 raster 를 리샘플합니다. 새로운 픽셀 값은 NearestNeighbor (영어 또는 미국 철자), 이중 선형, 큐빅, CubicSpline 또는 랑크조스 리샘플링 알고리즘을 사용하여 계산됩니다. 디폴트는 NearestNeighbor 입니다.

ST_Rescale — scale 의 규모 (또는 픽셀 크기)를 조절하여 래스터를 리샘플링합니다. 새로운 픽셀 값은 NearestNeighbor (영어 또는 미국 철자), Bilinear, Cubic, CubicSpline 또는 Lanczos reasmpling 알고리즘을 사용하여 계산됩니다. 기본값은 NearestNeighbor 입니다.

ST_Reskew — 기울기 (또는 회전 매개 변수)를 조정하여 래스터를 리샘플링합니다. 새로운 픽셀 값은 NearestNeighbor (영어 또는 미국 철자), Bilinear, Cubic, CubicSpline 또는 Lanczos reasmpling 알고리즘을 사용하여 계산됩니다. 기본값은 NearestNeighbor 입니다.

ST_Rotation — 라디안 래스터의 회전을 반환합니다.

ST_SRID — spatial_ref_sys 테이블에 정의 된 래스터의 공간 레퍼런스 식별자를 반환합니다.

ST_SameAlignment – raster 가 같은 skew, scale, spatial ref 를 가지고 있으면 true 를 반환하고, 문제에 관한 자세한 설명 없이 가지고 있지 않으면 false 를 반환합니다.

ST_ScaleX — 좌표 레퍼런스 시스템의 단위로 픽셀 너비의 X 구성 요소를 반환합니다.

ST_ScaleY — 좌표 레퍼런스 시스템의 단위로 픽셀 높이의 Y 구성 요소를 반환합니다.

ST_SetBandIsNoData — TRUE 로 밴드의 isnodata 플래그를 설정합니다.

ST_SetBandNoDataValue — no 데이터를 나타내는 주어진 배드를 위한 값을 설정합니다. 아무런 밴드도 지정되어 있지 않으면 band 1 로 간주됩니다. Nodata 값으로 마크하기 위해서는 nodata value = Null 로 설정합니다.

ST_SetGeoReference — 단일한 단위의 지리 참조번호형식인 6 지리 매개 변수를 설정합니다. 번호는 공백으로 구분됩니다. GDAL 또는 ESRI 데이터형식의 입력유형을 받아들입니다. 기본값은 GDAL 입니다.

ST_SetRotation — 래스터 이미지 픽셀의 회전 값을 라디안 단위로 설정합니다.

ST_SetSRID — spatial_ref_sys(편집자주: 공간좌표참조) 테이블에 정의된 특정 정수로 래스터의 SRID 를 설정합니다.

ST_SetScale — 좌표 참조 시스템의 단위로 X 와 픽셀의 Y 크기를 설정합니다. 단위 / 픽셀 너비 / 높이 번호를 매깁니다. Number units / pixel width / height.

ST_SetSkew — georeference X 와 Y 기울기를 설정합니다(또는 회전 파라미터). 오직 하나만 전달되는 경우, 같은 값으로 X 와 Y 를 설정합니다.

ST_SetUpperLeft – 투영된 X, Y 좌표로 픽셀의 좌상단 모서리의 값을 설정합니다.

ST_SetValue — 주어진 COLUMNX, Rowy 픽셀 또는 특정 점을 교차하는 픽셀의 지정된 밴드의 값을 설정으로 함으로써 수정되어있는 래스터데이터를 반환합니다. 밴드 번호는 1 에서 시작하며 지정되지 않은 경우 1 로 간주합니다.

ST_SkewX — (또는 회전 매개 변수) 지오레퍼런스 X 를 반환합니다.

ST_SkewY — (또는 회전 매개 변수) 지오레퍼런스 X 를 반환합니다.

ST_Slope — 고도 래스터 밴드의 표면 기울기를 반환합니다. 지형을 분석하는 데 유용합니다.

ST_SnapToGrid — 격자형데이터 영역을 선택하여 선택 된 래스터영역에 대하여 리샘플링 합니다. 새로운 픽셀 값은 NearestNeighbor (영어 또는 미국 철자), 이중 선형, 큐빅, CubicSpline 또는 Lanczos 리샘플링 알고리즘을 사용하여 계산됩니다. 기본값은 NearestNeighbor 입니다.

ST_SummaryStats — 래스터 또는 래스터 커버리지의 주어진 raster 밴드의 셈, 합, 평균, stddev, 최소값, 최대값을 구성하는 요약 통계를 반환합니다. 아무런 밴드가 지정되지 않은 경우 band1 로 간주됩니다.

ST_Transform — 지정된 리샘플링 알고리즘을 사용하여, 알려진 공간 레퍼런스 시스템으로 알려진 레퍼런스 시스템의 raster 를 재투영합니다. 옵션으로는 NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos 이 있으며 디폴트는 NearestNeighbor 입니다.

ST_Union — 1 밴드로 구성된 하나의 래스터로 래스터 타일의 집합의 합집합을 반환합니다. 아무런 밴드 unioning 을 지정하지 않은 경우, band num 1 로 간주됩니다. 그 결과 래스터의 범위는 전체 집합의 범위입니다. 교차로의 경우, 결과 값은 p_expression 에 의해 정의된 다음 중 하나입니다: LAST – 아무것도 지정되지 않았을 때 디폴트, MEAN, SUM, FIRST, MAX, MIN.

ST_UpperLeftX — 투영된 공간 참조의 래스터 좌상단 X 좌표를 반환합니다.

ST_UpperLeftY — 투영된 공간 참조의 래스터 좌상단 Y 좌표를 반환합니다.

ST_Value — 주어진 행 X, 열 y 에서 픽셀 또는 특정 기하학적 점에서의 특정 대역의 값을 반환합니다. 밴드 번호는 1 에서 시작하여 지정되지 않은 경우 1 로 간주합니다. exclude_nodata_value 가 false 로 설정되어있는 경우, 모든 픽셀 (NODATA 포함)의 픽셀 값은 교차하고 값을 반환하는 것으로 간주됩니다. 조건을 제약 nodata 가 전달되지 않은 경우, 래스터의 메타 데이터에서 읽습니다.

ST_ValueCount 픽셀 값 밴드를 포함하는 레코드의 집합을 변환하고 값들의 주어진 집합을 가진 래스터의 주어진 밴드의 픽셀 수를 계산합니다. 아무런 밴드가 지정되지 않을 경우 디폴트는 밴드 1. 기본적으로 NODATA 값 픽셀은 계산되지 않습니다. 그리고 픽셀의 다른 모든 값은 출력이며 픽셀 밴드 값은 가장 가까운 정수로 반올림됩니다.

ST_Width – 래스터 픽셀의 너비를 반환합니다.

ST_World2RasterCoordX – 포인트 지오메트리(pt)의 래스터 행을 반환하거나 래스터의 세계 공간 시스템에 나타나는 x 그리고 Y 세계 좌표(xw, yw)을 반환합니다.

ST_World2RasterCoordY — 포인트 지오메트리(pt)의 래스터 열을 반환하거나 래스터의 세계 공간 시스템에 나타나는 x 그리고 Y 세계 좌표(xw, yw)을 반환합니다.

13.5. PostGIS 지오메트리/지오그래피/래스터 덤프 함수들

아래에 주어진 함수들은 입력을 취하거나 geometry_dump 또는 geomval 데이터 유형 오브젝트 의 단일 또는 집합을 출력하는 PostGIS 함수들 입니다.

ST_DumpAsPolygons — 주어진 래스터 밴드, geomval (geom, val) rows 의 집합을 반환합니다. 아무 band num 이 지정되지 않은 경우, band num 은 1 을 기본값으로 합니다.

ST_Intersection — 래스터 또는 두개의 래스터가 공유하는 부분 또는 래스터 및 지오메트리의 vectorization 의 geometrical intersection 을 나타내는 geometry pixelvalue 쌍의 집합을 반환합니다.

ST_Dump — 지오메트리 G1 을 만드는 geometry_dump (geom, path) rows 의 세트를 반환합니다.

ST_DumpPoints — 지오메트리를 구성하는 모든 점들을 geometry_dump(geom, path, Point 와 정수 배열로 구성된) 행의 집합으로 반환합니다. 이 함수는 집합을 반환하는 함수(Set-Returning Function - SRF)입니다.

ST_DumpRings – Polygon 을 구성하는 외부(Exterior) 및 내부(Interior) 링(Ring)을 geometry_dump(Polygon 과 정수 배열로 구성된) 행의 집합으로 반환합니다. 이 함수는 집합을 반환하는 함수(Set-Returning Function - SRF)입니다.

13.6. PostGIS 상자 함수들

아래에 주어진 함수들은 입력을 취하거나 PostGIS spatial types 의 box* family 을 입출력하는 PostGIS 함수들 입니다. 유형들의 box family 는 box2d, 와 box3d 로 구성됩니다.

Box2D — 지오메트리의 최대 범위를 나타내는 BOX2D 을 반환.

Box3D — 지오메트리의 최대 범위를 나타내는 BOX3D 을 반환합니다.

Box3D — 래스터의 elclosing box 의 3 차원 표현을 반환합니다.

ST_3DExtent — 지오메트리의 열들을 제한하는 box3D 바운딩 박스를 반환하는 집계 함수.

ST_3DMakeBox – 주어진 3d 포인트 지오메트리에 의해 정의되는 BOX3D 를 생성합니다.

ST_Estimated_Extent — 주어진 공간 테이블의 '추정된' extent 를 반환합니다. 추정은 지오메트리 컬럼의 통계에서 가져온 것입니다. 만약 지정되지 않는다면 현재 스키마가 사용될 것입니다.

ST_Expand — 입력 지오메트리의 바운딩 박스로부터 모든 방향으로 확장된 바운딩 박스를 반환합니다. Double precision 을 사용합니다.

ST_Extent — 지오메트리의 열들을 제한하는 바운딩박스를 반환하는 집계함수입니다.

ST_MakeBox2D – 주어진 포인트 지오메트리에 의해 정의된 BOX2D 를 생성합니다.

ST_XMax — 2D, 3D 혹은 지오메트리 바운딩 박스의 X 최대 값을 반환합니다.

ST_XMin — 2D, 3D 혹은 지오메트리 바운딩 박스의 X 최소 값을 반환합니다.

ST_YMax — 2D, 3D 혹은 지오메트리 바운딩 박스의 Y 최대 값을 반환합니다.

ST_YMin — 2D, 3D 혹은 지오메트리 바운딩 박스의 Y 최소 값을 반환합니다.

ST_ZMax — 2D, 3D 혹은 지오메트리 바운딩 박스의 Z 최대 값을 반환합니다.

ST_ZMin — 2D, 3D 혹은 지오메트리 바운딩 박스의 Z 최소 값을 반환합니다.

13.7. 3D 지원 PostGIS 함수들

아래에 주어진 함수들은 Z-Index 를 무시하지 않고 3D 를 지원하는 PostGIS 함수들 입니다.

AddGeometryColumn — 기존 속성 테이블에 지오메트리 컬럼을 추가합니다. 기본적으로 정의하기 위해서 제약 조건보다는 type modifier 를 사용합니다. 행동에 기반한 오래된 check 제약 얻기 위한 use_typmod 에 대한 false 을 전달합니다.

Box3D — 지오메트리의 최대 규모를 표현하는 BOX3D 를 반환합니다.

DropGeometryColumn — 공간 테이블로부터 지오메트리 행을 제거합니다.

GeometryType — 지오메트리의 유형을 문자열로 반환합니다. 예: 'LINESTRING', 'POLYGON', 'MULTIPOINT' 등

ST_3DClosestPoint — G2 에 가장 가까운 G1 에서 3 차원 POINT 를 반환합니다. 이는 3D 짧은 라인의 첫 번째 점입니다.

ST_3DDFullyWithin — 3D Geometry 의 모든 것이 서로의 지정된 거리 내에있는 경우에 true 를 반환합니다.

ST_3DDWithin — 3D(z) Geometry 유형에 대해서 두 지오메트리의 3 차원 공간에서의 거리가 지정된 거리 내에 있는 경우 TRUE 를 반환합니다.

ST_3DDistance — Geometry 유형에 대해서 두 지오메트리 사이의 3 차원 직교 최단거리(공간 좌표계를 기준으로 한 단위)를 반환합니다.

ST_3DExtent — 지오메트리 열들을 제한하는 box3d 바운딩 박스를 반환하는 집계 함수입니다.

ST_3DIntersects — 두 지오메트리가 3 차원 공간에서 "공간적으로 교차"하는 경우 TRUE 를 반환합니다. Point 및 LineString 만을 대상으로 합니다.

ST_3DLength — 지오메트리가 LineString 또는 MultiLineString 일 경우 2 차원 또는 3 차원 상에서의 길이를 반환합니다. Z 값이 있는 경우 3 차원, 그렇지 않으면 ST_Length(또는 ST_Lenght2D)와 동일합니다.

ST_3DLength_Spheroid — 타원체에 기반하여 고도를 고려한 지오메트리의 길이를 계산합니다. 이 함수는 ST_Length_Spheroid 의 별칭입니다.

ST_3DLongestLine — 두 지오메트리 사이의 3 차원 가장 긴 라인을 돌려줍니다

ST_3DMakeBox — 주어진 좌하단 및 우상단 3 차원 Point 지오메트리에 의해 정의된 BOX3D 를 생성합니다.

ST_3DMaxDistance — Geometry 유형에 대해서 두 지오메트리 사이의 3 차원 직교 최장거리(공간 좌표계를 기준으로 한 단위)를 반환합니다.

ST_3DPerimeter — 지오메트리가 Polygon 또는 MultiPolygon 일 경우 3 차원 상에서의 둘레 길이를 반환합니다.

ST_3DShortestLine — 두 지오메트리 사이의 3 차원 가장 짧은 라인을 돌려줍니다.

ST_Accum — 집계. 지오메트리의 배열을 구축합니다.

ST_AddMeasure — 시작과 끝 점 사이 선형적인 보간 측정 요소와 파생 된 형상을 반환합니다. 지오메트리가 측정 치수를 가지지 않는 경우, 하나가 추가됩니다. Geometry 가 측정 치수가 있으면 새 값으로 덮어 쓰기 됩니다. 오직 LINESTRINGS 와 MULTILINESTRINGS 만이 지원됩니다.

ST_AddPoint — LineString 의 Point<position>(0 기반 인덱스) 앞에 Point 를 추가합니다. position 값이 생략되면 -1 값이 기본값이며 LineString 의 마지막 위치에 Point 가 추가됩니다.

ST_Affine — 한번에 변환(translate), 회전(rotate), 크기(scale)와 같은 작업들을 수행하기 위해 지오메트리에 3 차원 아핀 변환(Affine transformation)을 적용합니다.

ST_AsBinary — Geometry/Geography 객체를 SRID 메타데이터를 포함하지 않은 Well-Known Binary(WKB) 포맷으로 반환합니다.

ST_AsEWKB — Geometry/Geography 객체를 SRID 메타데이터를 포함한 Well-Known Binary(WKB) 포맷으로 반환합니다.

ST_AsEWKT — 지오메트리를 SRID 메타데이터를 포함한 Well-Known Text(WKT) 포맷으로 반환합니다.

ST_AsGML — 지오메트리를 GML(Geography Markup Language) 버전 2(2.1.2, 기본값) 또는 버전 3(3.1.1) 포맷으로 반환합니다.

ST_AsGeoJSON — 지오메트리를 GeoJSON(Geometry Javascript Object Notation) 포맷으로 반환합니다.

ST_AsHEXEWKB — 지오메트리를 little-endian (NDR) 또는 big-endian (XDR) 인코팅을 사용하여 HEXEWKB 포맷(텍스트)으로 반환합니다.

ST_AsKML — 지오메트리를 KML(Keyhole Markup Language) 포맷으로 반환합니다. 옵션 매개변수를 이용하여 GML 의 버전(기본값 버전 2), CRS 표시 유형 및 소수점 최대 자리수(기본값 15) 등을 설정할 수 있습니다.

ST_AsX3D — X3D XML 노드 요소 형식으로 지오메트리를 반환합니다. ISO-IEC-19776-1.2-X3DEncodings-XML

ST_Boundary — 이 지오메트리의 조합 경계의 폐쇄를 반환합니다.

ST_Collect — 다른 지오메트리의 컬렉션에서 지정된 ST_Geometry 를 값을 반환합니다.

ST_ConvexHull — 지정된 지오메트리 집합을 모두 둘러싸는 최소 볼록 폐곡선(Convex Hull)을 Polygon 지오메트리로 반환합니다.

ST_CoordDim — ST_Geometry 값에 대한 지오메트리의 좌표 차수(XY 는 2, XYZ 는 3 등)를 반환합니다. 이 함수는 SWL/MM 사양 중 ST_NDims 의 별칭입니다.

ST_CurveToLine — CIRCULARSTRING/CURVEDPOLYGON 을 LINESTRING/POLYGON 로 변환합니다.

ST_Difference — 지오메트리 B 와 교차하지 않는 지오메트리의 A 의 일부를 나타내는(차집합) 지오메트리를 반환합니다.

ST_Dump — 지오메트리 g1 을 구성하는 geometry_dump (geom,path) rows 의 집합을 반환합니다 .

ST_DumpPoints — 지오메트리를 구성하는 모든 점들의 geometry_dump (geom,path) rows 의 집합을 반환합니다.

ST_DumpRings — 폴리곤의 내부링, 외부링을 나타내는 geometry_dump rows 를 반환합니다.

ST_EndPoint — LineString 지오메트리의 끝점을 Point 로 반환합니다.

ST_ExteriorRing — 폴리곤 지오메트리의 외부 링을 나타내는 선 스트링을 반환합니다. 지오메트리가 폴리곤이 아닐 경우 NULL 을 반환합니다. MULTIPOLYGON 과 는 동작하지 않습니다.

ST_FlipCoordinates — X 및 Y 축 대칭으로 주어진 지오메트리의 버전을 반환합니다. 경위도 피쳐들을 개발하거나 수정하는 사람들에게 유용합니다.

ST_ForceRHR — 오른손 규칙(Right-Hand-Rule)을 따르도록 Polygon 정점의 방향을 구성합니다. Polygon 에서 외부 링(Exterior Ring)은 시계방향, 내부 링(Interior Ring)은 반 시계 방향으로 정점의 순서가 구성됩니다.

ST_Force_2D — 모든 출력 문자열들이 오직 X, Y 좌표들을 가지기 위해 지오메트리를 “2 차원 모드”로 전환합니다.

ST_Force_3D — XYZ 모드로 지오메트리를 전환합니다. 이 함수는 ST_Force_3DZ 의 별칭입니다.

ST_Force_3DZ — XYZ 모드로 지오메트리를 전환합니다. 이 함수는 ST_Force_3D 의 별칭입니다.

ST_Force_4D — XYZM 모드로 지오메트리를 전환합니다.

ST_Force_Collection — 지오메트리를 GEOMETRYCOLLECTION 로 변환합니다..

ST_GeomFromEWKB — Extended Well-Known Binary(EWKB)에서 지정된 ST_Geometry 를 값을 반환합니다.

ST_GeomFromEWKT — Extended Well-Known Text (EWKT)에서 지정된 ST_Geometry 를 값을 반환합니다.

ST_GeomFromGML — GML 표현으로 지오메트리를 입력 받아 PostGIS 지오메트리 오브젝트를 출력합니다.

ST_GeomFromGeoJSON — GeoJSON 지오메트리를 입력 받아 PostGIS 지오메트리 객체를 반환합니다.

ST_GeomFromKML — KML 표현으로 지오메트리를 입력 받아 PostGIS 지오메트리 오브젝트를 출력합니다.

ST_GeometryN — 지오메트리가 GEOMETRYCOLLECTION, (MULTI) POINT, (MULTI) LINE, MULTICURVE 또는 (MULTI) POLYGON, POLYHEDRALSURFACE 경우 1 부터 N 번째 도형을 반환합니다. 그렇지 않으면, NULL 을 반환합니다.

ST_GeometryType — ST_Geometry 값의 지오메트리 유형을 반환합니다.

ST_HasArc – 지오메트리 또는 지오메트리 컬렉션이 circular string 을 포함하는 경우 true 를 반환합니다.

ST_InteriorRingN — 다각형 지오메트리의 N 번째 내부 라인스트링을 반환합니다. 지오메트리가 다각형이 아니거나 주어진 N 이 범위 밖이라면 NULL 을 반환합니다.

ST_InterpolatePoint — 주어진 포인트로 마감된 시점의 지오메트리 측정 치수 값을 반환합니다.

ST_IsClosed — 선 스트링의 시작과 끝 점이 일치하는 경우 TRUE 를 반환합니다. Polyhedral 경우 표면은 닫혀있습니다. (volumetric).

ST_IsCollection — 지오메트리가 GEOMETRYCOLLECTION, MULTI{POINT, POLYGON, LINESTRING, CURVE, SURFACE}, COMPOUNDCURVE 일 경우 TURE 를 반환합니다:

ST_IsSimple — 이 지오메트리가 자기 교차 또는 자기 접선과 같이 변칙적인 지오메트리 포인트를 가지고 있지 않은 경우 TRUE 를 반환합니다.

ST_Length_Spheroid — LineString 또는 MultiLineString 지오메트리의 2 차원 또는 3 차원 상에서의 길이를 타원체에 기반하여 계산합니다. 지오메트리의 좌표가 경/위도이고 투영 없이 길이를 측정하고자 할 때 유용합니다.

ST_LineFromMultiPoint — MultiPoint 지오메트리에서 LineString 을 생성합니다.

ST_LineToCurve - LINESTRING/POLYGON 을 CIRCULARSTRING, CURVED POLYGON 으로 변환합니다.

ST_Line_Interpolate_Point — 선을 따라 보간된 지점을 Point 지오메트리로 반환합니다. 두 번째 매개변수는 LineString 에 대한 길이의 비율을 나타내는 부동소수점(Float8) 값이며 0 과 1 사이의 값을 사용해야 합니다. 입력 지오메트리는 반드시 LineString 이어야 합니다.

ST_Line_Substring — 주어진 LineString 지오메트리에서 2 차원 상의 총 길이에 대한 시작 및 종료 비율에 해당하는 LineString 지오메트리를 반환합니다. 두 번째 및 세 번째 인수는 0 과 1 사이의 부동 소수점(Float8) 값입니다.

ST_LocateBetweenElevations — 포괄적으로 지정된 범위의 고도와 교차하는 요소들을 가진 파생된 지오메트리(컬렉션) 값을 반환합니다. 3D, 4D LineString 및 MultiLineString 지오메트리만 지원합니다.

ST_M — Point 의 M 값을 반환하거나 없는 경우에는 NULL 값을 반환합니다. 입력은 반드시 Point 이어야 합니다.

ST_MakeLine — Point 또는 Line 지오메트리를 이용하여 LineString 을 생성합니다.

ST_MakePoint — X, Y, Z, M 등의 값을 이용하여 2D, 3DZ 또는 4D Point 지오메트리를 생성합니다.

ST_MakePolygon — 주어진 쉘에 의해 형성된 폴리곤을 생성합니다. 입력 geometries 는 반드시 닫혀진 라인스트링이어야 합니다.

ST_MakeValid — vertice 를 잃지 않고 유효하지 않는 지오메트리을 유효하게 만드는 시도를 합니다.

ST_MemUnion — ST_Union 함수와 같은 결과를 반환하지만, 단지 메모리 친화적(더 작은 메모리와 더 많은 프로세서 시간을 사용)입니다.

ST_Mem_Size — Geometry 가 소요하는 총 메모리 공간 (바이트)의 크기를 반환합니다.

ST_NDims — 지오메트리의 좌표 차수(XY 는 2, XYZ 는 3 등)를 정수(small int) 값으로 반환합니다. 값들은: 2,3 or 4.

ST_NPoints — 지오메트리에서 점의 개수 (vertexes)를 반환합니다.

ST_NRings – 지오메트리가 폴리곤 또는 멀티 폴리곤일 경우 링의 개수를 반환합니다.

ST_Node — 라인스트링 집합의 노드입니다.

ST_NumGeometries — 만약 지오메트리가 GEOMETRYCOLLECTION (or MULTI*)일 경우 지오메트리의 수를 반환합니다. 단일 지오메트리의 경우 1 을, 그렇지 않는 경우 NULL 을 반환합니다.

ST_NumPatches — PolyhedralSurface 의 면(faces)의 개수를 반환합니다. Polyhedral 지오메트리가 아닌 경우 NULL 값을 반환합니다.

ST_PatchN — PolyhedralSurface 또는 PolyhedralSurfaceM 지오메트리인 경우 처음을 1 부터 시작하는 N 번째 지오메트리(Face)를, 그렇지 않으면 NULL 값을 반환합니다.

ST_PointFromWKB — 주어진 SRID 와 WKB 로부터 지오메트리를 만듭니다.

ST_PointN — LineString 또는 CircularLineString 지오메트리인 경우 처음을 1 부터 시작하는 N 번째 지오메트리를, 그렇지 않으면 NULL 값을 반환합니다.

ST_PointOnSurface — 표면에 반드시 위치하는 Point 지오메트리를 반환합니다.

ST_Polygon — 지정된 LineString 및 SRID 를 이용해서 생성한 Polygon 을 반환합니다.

ST_RemovePoint — LineString 에서 Point 를 제거합니다. 오프셋 (offset)는 0 을 기준으로합니다.

ST_RemoveRepeatedPoints — 제거된 중복 포인트의 주어진 지오메트리 버전을 반환합니다.

ST_Rotate — 원점 축을 기준으로 시계 반대 방향으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_RotateX — X 축을 기준으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_RotateY — Y 축을 기준으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_RotateZ — Z 축을 기준으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_Scale — 매개변수와 원본 지오메트리의 좌표값을 곱하여 새로운 크기의 지오메트리를 생성 후 반환합니다. 예: ST_Scale(geom, Xfactor, Yfactor, Zfactor).

ST_SetPoint — 주어진 점과 라인 스트링의 포인트 N 를 교체합니다. 지수는 0 을 기준으로합니다.

ST_Shift_Longitude — 지오메트리 모든 기능의 모든 구성 요소의 point/vertex 를 읽고, 경도 좌표 <0 인 경우, 360 추가합니다. 결과는 180 중심의지도에 그려질 데이터의 0-360 버전이 될 것입니다.

ST_SnapToGrid — 입력 지오메트리의 모든 점을 정규 그리드(Regular Grid, 원점과 셀 크기로 정의된 그리드)에 스냅 합니다. 이 함수는 같은 셀에 포함된 포인트는 제거하며, 지오메트리 좌표의 정밀도를 줄이는데 유용합니다.

ST_StartPoint — LineString 지오메트리의 첫 번째 점을 Point 지오메트리로 반환합니다.

ST_SymDifference — 지오메트리 A 와 지오메트리 B 가 교차하지 않는 부분의 지오메트리를 반환합니다. 이것을 대칭 차집합(Symmetric Difference) 이라고 합니다. 왜냐하면 ST_SymDifference (A, B) = ST_SymDifference (B, A)이기 때문입니다.

ST_TransScale — deltaX, deltaY 매개변수를 사용하여 지오메트리를 변환한 뒤 XFactor, YFactor 매개변수를 사용하여 크기를 조정한 지오메트리를 반환합니다. 2D 에서만 수행됩니다.

ST_Translate — 오프셋으로 숫자 매개 변수를 사용하여 새 위치로 지오메트리를 변환합니다. Ie: ST_Translate(geom, X, Y) or ST_Translate(geom, X, Y,Z).

ST_UnaryUnion — ST_Union 과 같지만 지오메트리 구성 요소 수준에서 작동합니다..

ST_X — 포인트의 X 좌표를 반환합니다. 사용 가능하지 않을 경우 NULL 을 반환합니다. 입력은 반드시 포인트여야 합니다.

ST_XMax — 바운딩 박스 2D 또는 3D 또는 지오메트리의 X 최대 값을 반환합니다.

ST_XMin — 바운딩 박스 2D 또는 3D 또는 지오메트리의 X 최소 값을 반환합니다.

ST_Y — 포인트의 Y 좌표를 반환합니다. 사용가능하지 않을 경우 NULL 을 반환합니다. 입력은 반드시 포인트여야 합니다.

ST_YMax — 바운딩 박스 2D 또는 3D 또는 지오메트리의 Y 최대 값을 반환합니다.

ST_YMin — 바운딩 박스 2D 또는 3D 또는 지오메트리의 Y 최소 값을 반환합니다.

ST_Z — 포인트의 Z 좌표를 반환합니다. 사용가능하지 않을 경우 NULL 을 반환합니다. 입력은 반드시 포인트여야 합니다.

ST_ZMax — 바운딩 박스 2D 또는 3D 또는 지오메트리의 Z 최대 값을 반환합니다.

ST_ZMin — 바운딩 박스 2D 또는 3D 또는 지오메트리의 Z 최소 값을 반환합니다

ST_Zmflag — 지오메트리의 ZM (치수 의미) 플래그 값을 정수(small int) 값으로 반환합니다. 0 = 2D, 1=3DM, 2 = 3DZ, 3 = 4D 값입니다.

UpdateGeometrySRID — 지오메트리 컬럼, geometry_columns 메타 데이터와 SRID 의 모든 피쳐의 SRID 를 업데이트합니다. 이 제약 조건을 시행 한 경우 제약 조건이 새 SRID 제약과 함께 업데이트됩니다. 기존의 것이(The old) 유형 정의에 의해 시행 된 경우, 타입 정의가 변경됩니다.

geometry_overlaps_nd — A 의 3D bounding box 가 B 의 3D bounding box 와 교차하는 경우 TRUE 을 반환합니다.

13.8. PostGIS 곡선 지오메트리 지원 함수

아래 주어진 함수들은 CIRCULARSTRING, CURVEDPOLYGON, 그리고 다른 곡선 지오메트리 유형등을 사용할 수 있는 PostGIS 함수들입니다.

AddGeometryColumn — 기존의 속성 테이블에 지오메트리 컬럼을 추가합니다. 기본적으로 오히려 제약 조건보다 정의 type modidfier 를 사용합니다. 기본적으로 정의하기위해서는 제약 조건보다 정의 유형 한정자(type modifier)를 사용합니다. 행동을 기반으로하는 구 check 제약조건을 가지기 위한 use_typmod 시 false 를 전달합니다.

Box2D — 지오메트리 최대 범위를 나타내는 BOX2D 을 반환합니다.

Box3D — 지오메트리 최대 범위를 나타내는 BOX3D 을 반환합니다

DropGeometryColumn — 공간 테이블로부터 지오메트리 행을 제거합니다.

GeometryType — 문자열로 지오메트리의 형식을 반환합니다. 예: 'LINESTRING', 'POLYGON', 'MULTIPOINT', etc.

PostGIS_AddBBox — 지오메트리에 바운딩 박스를 추가합니다.

PostGIS_DropBBox — 지오메트리부터 바운딩 박스 캐쉬를 삭제합니다.

PostGIS_HasBBox — 이 지오메트리의 BBOX 이 캐시되었다면 TRUE 을 반환, 그렇지 않으면 FALSE 를 반환합니다.

ST_3DExtent — 행의 영역을 포함하는 Box3D 바운딩 박스를 반환하는 집계 함수입니다.

ST_Accum — 집계. 지오메트리의 배열을 구축합니다.

ST_Affine — 한번에 변환(translate), 회전(rotate), 크기(scale)와 같은 작업들을 수행하기 위해 지오메트리에 3 차원 아핀 변환(Affine transformation)을 적용합니다.

ST_AsBinary — Geometry/Geography 객체를 SRID 메타데이터를 포함하지 않은 Well-Known Binary(WKB) 포맷으로 반환합니다.

ST_AsEWKB — Geometry/Geography 객체를 SRID 메타데이터를 포함한 Well-Known Binary(WKB) 포맷으로 반환합니다.

ST_AsEWKT — 지오메트리를 SRID 메타데이터를 포함한 Well-Known Text(WKT) 포맷으로 반환합니다.

ST_AsHEXEWKB — 지오메트리를 little-endian (NDR) 또는 big-endian (XDR) 인코딩을 사용하여 HEXEWKB 포맷(텍스트)으로 반환합니다.

ST_AsText — Geometry/Geography 객체를 SRID 메타데이터를 포함하지 않은 Well-Known Text(WKT) 포맷으로 반환합니다.

ST_Collect — 다른 지오메트리 집합에서 지정된 ST_Geometry 값을 반환합니다. 반환되는 유형은 Multi* 또는 GeometryCollection 지오메트리입니다.

ST_CoordDim — ST_Geometry 값에 대한 지오메트리의 좌표 차수(XY 는 2, XYZ 는 3 등)를 반환합니다. 이 함수는 SWL/MM 사양 중 ST_NDims 의 별칭입니다.

ST_CurveToLine — CIRCULARSTRING/CURVEDPOLYGON 을 LINESTRING/POLYGON 으로 변환합니다.

ST_Dump — g1 지오메트리를 구성하는 geometry_dump(geom, path, 지오메트리와 정수 배열로 구성된) 행의 집합으로 반환합니다. 이 함수는 집합을 반환하는 함수(Set-Returning Function - SRF)입니다.

ST_Estimated_Extent — 주어진 공간 테이블의 '추정된' extent 를 반환합니다. 추정은 지오메트리 컬럼의 통계에서 가져온 것입니다. 지정하지 않으면 현재 스키마가 사용됩니다

ST_FlipCoordinates — 주어진 지오메트리의 X 및 Y 축을 반전한 지오메트리를 반환합니다. 위/경도 객체를 구축해 본 적이 있거나 수정을 필요로 하는 사람들에게 유용합니다.

ST_Force_2D — 모든 출력 문자열들이 오직 X 그리고 Y 좌표들을 가지기 위해 지오메트리를 "2 차원 모드"로 전환합니다

ST_Force_3D — 지오메트리를 XYZ 모드로 전환합니다. 이 함수는 ST_Force_3DZ 의 별칭 이름입니다.

ST_Force_3DM – 지오메트리를 XYZ 모드로 전환합니다.

ST_Force_3DZ — 지오메트리를 XYZ 모드로 전환합니다. 이 함수는 ST_Force_3D 의 별칭 이름입니다

ST_Force_4D — 지오메트리를 XYZM 모드로 전환합니다

ST_Force_Collection — geometry 를 GEOMETRYCOLLECTION 로 변환합니다..

ST_GeoHash — 지오메트리를 GeoHash(geohash.org) 포맷으로 변환합니다 옵션 매개변수를 이용하여 좌표 값의 정밀도를 설정할 수 있습니다.

ST_GeogFromWKB — WKT 또는 EWKT 에서 지정된 Geography 객체를 반환합니다. WGS84 경위도(EPSG 4326) 좌표계를 가정합니다.

ST_GeomFromEWKB — EWKB 에서 지정된 ST_Geometry 값을 반환합니다.

ST_GeomFromEWKT — EWKT 에서 지정된 ST_Geometry 값을 반환합니다.

ST_GeomFromText — WKT 에서 지정된 ST_Geometry 값을 반환합니다.

ST_GeomFromWKB — WKB 에서 지정된 ST_Geometry 값을 반환합니다.

ST_GeometryN — 지오메트리가 GEOMETRYCOLLECTION, (MULTI) POINT, (MULTI) LINESTRING, MULTICURVE 또는 (MULTI) POLYGON, POLYHEDRALSURFACE 인 경우 처음을 1 부터 시작하는 N 번째 지오메트리를, 그렇지 않으면 NULL 값을 반환합니다.

= — A 의 바운딩 박스가 B 의 바운딩 박스와 일치하면 TRUE 를 반환합니다. 배 정밀도 바운딩 박스를 사용합니다.

&<| — A 의 바운딩 박스가 B 의 바운딩 박스와 겹치거나 B 의 아래에 있으면 TRUE 를 반환합니다.

ST_HasArc — 지오메트리 또는 지오메트리 컬렉션이 Circular String 을 포함하는 경우 TRUE 를 반환합니다.

ST_IsClosed — LineString 의 시작점과 끝점이 일치하는 경우 TRUE 를 반환합니다. 다면체 표면은 닫힙니다. (volumetric)

ST_IsCollection — 지오메트리가 GEOMETRYCOLLECTION, MULTI{POINT, POLYGON, LINESTRING, CURVE, SURFACE}, COMPOUNDCURVE 일 경우 TURE 를 반환합니다:

ST_IsEmpty — 지오메트리가 빈 GeometryCollection, Polygon, LineString, Point 등일 경우 TRUE 를 반환합니다.

ST_LineToCurve — LINESTRING/POLYGON 을 CIRCULARSTRING, CURVED POLYGON 로 변환합니다.

ST_Mem_Size — 지오메트리가 차지하는 총 메모리 공간의 크기(바이트 단위)를 반환합니다.

ST_NPoints — 지오메트리에서 점의 수 (vertexes)를 반환합니다.

ST_NRings — 지오메트리가 Polygon 또는 MultiPolygon 인 경우, 링(Ring)의 개수를 반환합니다.

ST_PointFromWKB — 지정된 SRID 의 WKB 에서 지오메트리를 생성합니다.

ST_PointN — LineString 또는 CircularLineString 지오메트리인 경우 처음을 1 부터 시작하는 N 번째 지오메트리를, 그렇지 않으면 NULL 값을 반환합니다.

ST_Rotate — 원점 축을 기준으로 시계 반대 방향으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_RotateZ — Z 축을 기준으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_SRID — spatial_ref_sys 테이블에 정의 된 ST_Geometry 를위한 공간 레퍼런스 식별자를 반환합니다.

ST_Scale — 매개변수와 원본 지오메트리의 좌표값을 곱하여 새로운 크기의 지오메트리를 생성 후 반환합니다. 예: ST_Scale(geom, Xfactor, Yfactor, Zfactor).

ST_SetSRID — 지정된 정수 값(SRID 코드)으로 지오메트리의 SRID 를 설정합니다.

ST_TransScale — deltaX, deltaY 매개변수를 사용하여 지오메트리를 변환한 뒤 XFactor, YFactor 매개변수를 사용하여 크기를 조정한 지오메트리를 반환합니다. 2D 에서만 수행됩니다.

ST_Transform — 정수 매개 변수에 의해 참조되는 SRID 으로 변환된 좌표를 가진 새 지오메트리를 반환합니다.

ST_Translate — 숫자 매개변수 오프셋을 사용하여 새 위치로 지오메트리를 변환하여 반환합니다. 예: ST_Translate(geom, X, Y) or ST_Translate(geom, X, Y,Z).

ST_XMax — 바운딩 박스 2D 또는 3D 또는 지오메트리의 X 최대 값을 반환합니다.

ST_XMin — 바운딩 박스 2D 또는 3D 또는 지오메트리의 X 최소 값을 반환합니다.

ST_YMax — 바운딩 박스 2D 또는 3D 또는 지오메트리의 Y 최대 값을 반환합니다.

ST_YMin — 바운딩 박스 2D 또는 3D 또는 지오메트리의 Y 최소 값을 반환합니다.

ST_ZMax — 바운딩 박스 2D 또는 3D 또는 지오메트리의 Z 최데 값을 반환합니다.

ST_ZMin — 바운딩 박스 2D 또는 3D 또는 지오메트리의 Z 최소값을 반환합니다.

ST_Zmflag — Returns 작은 int 로 지오메트르의 ZM (dimension semantic) 플래그를 반환합니다. 값들은: 0=2d, 1=3dm, 2=3dz, 3=4d.

UpdateGeometrySRID — 지오메트리 열, geometry_columns 메타 데이터와 SRID 의 모든 피쳐의 SRID 를 업데이트합니다. 이 제약 조건을 시행 한 경우 제약 조건이 새 SRID 제약과 함께 업데이트됩니다. 기존의 것이(The old) 타입 정의에 의해 시행 된 경우, 타입 정의가 변경됩니다.

&& — A 의 2D 바운딩 박스가 B 의 2D 바운딩 박스와 교차할 때 TRUE 를 반환합니다.

&&& — A 의 3D 바운딩 박스가 B 의 3D 바운딩 박스와 교차할 때 TRUE 를 반환합니다

13.9. PostGIS 다면체 표면 지원 함수들

아래 주어진 함수들은 POLYHEDRALSURFACE, POLYHEDRALSURFACEM 지오메트리를 사용하는 PostGIS 함수들입니다.

Box2D — 지오메트리의 최대 범위를 나타내는 BOX2D 를 반환합니다.

Box3D — 지오메트리의 최대 범위를 나타내는 BOX3D 를 반환합니다.

GeometryType —지오메트리의 유형을 문자열로 반환합니다. 예: 'LINESTRING', 'POLYGON', 'MULTIPOINT' 등.

ST_3DClosestPoint — g2 와 가장 가까운 g1 의 3 차원 Point 를 반환합니다. 3 차원 공간에서 최단거리 직선의 첫 번째 Point 입니다.

ST_3DDFullyWithin — 모든 3D 지오메트리들이 지정된 거리 내에 있는 경우 TRUE 를 반환합니다.

ST_3DDWithin — 3D(z) Geometry 유형에 대해서 두 지오메트리의 3 차원 공간에서의 거리가 지정된 거리 내에 있는 경우 TRUE 를 반환합니다.

ST_3DDistance – Geometry 유형에 대해서 두 지오메트리 사이의 3 차원 직교 최단거리(공간 좌표계를 기준으로 한 단위)를 반환합니다.

ST_3DExtent – 행의 영역을 포함하는 Box3D 바운딩 박스를 반환하는 집계 함수입니다.

ST_3DIntersects — 두 지오메트리가 3 차원 공간에서 "공간적으로 교차"하는 경우 TRUE 를 반환합니다. Point 및 LineString 만을 대상으로 합니다.

ST_3DLongestLine – 두개의 지오메트리 사이의 가장 긴 3 차원 선을 반환합니다.

ST_3DMaxDistance – 지오메트리 타입의 경우, 투영된 단위의 두개의 지오메트리 사이 3 차원 직교 최소 거리(공간 ref 기반)를 반환합니다.

ST_3DShortestLine – 두 지오메트리 사이의 가장 짧은 3 차원 선을 반환합니다.

ST_Accum – 집계. 지오메트리의 배열을 구축합니다.

ST_Affine — 한번에 변환(translate), 회전(rotate), 크기(scale)와 같은 작업들을 수행하기 위해 지오메트리에 3 차원 아핀 변환(Affine transformation)을 적용합니다.

ST_Area — 지오메트리가 Polygon 또는 MultiPolygon 인 경우 표면의 면적을 반환합니다. "Geometry" 유형은 SRID 단위, "Geography" 유형은 평방 미터 단위입니다.

ST_AsBinary — SRID 메타 데이터가 없는 래스터의 바이너리 정보 (WKB: WELL KNOWN binary 형식)를 반환하여 나타냅니다.

ST_AsEWKB — Geometry/Geography 객체를 SRID 메타데이터를 포함한 Well-Known Binary(WKB) 포맷으로 반환합니다. 두 번째 매개변수는 little-endian ('NDR') 또는 big-endian ('XDR') 인코딩 옵션입니다.

ST_AsEWKT — 지오메트리를 SRID 메타데이터를 포함한 Well-Known Text(WKT) 포맷으로 반환합니다.

ST_AsGML – 지오메트리를 GML(Geography Markup Language) 버전 2(2.1.2, 기본값) 또는 버전 3(3.1.1) 포맷으로 반환합니다. 옵션 매개변수를 이용하여 GML 의 버전, CRS 표시 유형 및 소수점 최대 자리수(기본값 15) 등을 설정할 수 있습니다.

ST_AsX3D — X3D XML 노드 요소 포맷으로 지오메트리를 반환합니다: ISO-IEC-19776-1.2-X3DEncodings-XML. 옵션 매개변수를 이용하여 유형 및 소수점 최대 자리수(기본값 15) 등을 설정할 수 있습니다.

ST_CoordDim — ST_Geometry 값에 대한 지오메트리의 좌표 차수(XY 는 2, XYZ 는 3 등)를 반환합니다. 이 함수는 SWL/MM 사양 중 ST_NDims 의 별칭입니다.

ST_Dimension – 좌표 차수보다 작거나 반드시 같아야 하는 이 지오메트리 객체의 고유 차원을 반환합니다. Point 는 0, LineString 은 1, Polygon 은 2 를 반환합니다.

ST_Dump – g1 지오메트리를 구성하는 geometry_dump(geom, path, 지오메트리와 정수 배열로 구성된) 행의 집합으로 반환합니다. 이 함수는 집합을 반환하는 함수(Set-Returning Function - SRF)입니다.

ST_DumpPoints — 지오메트리를 구성하는 모든 점들을 geometry_dump(geom, path, Point 와 정수 배열로 구성된) 행의 집합으로 반환합니다. 이 함수는 집합을 반환하는 함수(Set-Returning Function - SRF)입니다.

ST_Expand — 입력된 지오메트리의 바운딩 박스에서 모든 방향으로 확장된 바운딩 박스를 반환합니다. Double precision 을 사용합니다.

ST_Extent – 지오메트리 행의 영역을 포함하는 바운딩 박스를 반환하는 집계 함수입니다.

ST_FlipCoordinates — 주어진 지오메트리의 X 및 Y 축을 반전한 지오메트리를 반환합니다. 위/경도 객체를 구축해 본 적이 있거나 수정을 필요로 하는 사람들에게 유용합니다.

ST_ForceRHR – 오른손 규칙(Right-Hand-Rule)을 따르도록 Polygon 정점의 방향을 구성합니다. Polygon 에서 외부 링(Exterior Ring)은 시계방향, 내부 링(Interior Ring)은 반 시계 방향으로 정점의 순서가 구성됩니다.

ST_Force_2D — 출력 지오메트리가 X 및 Y 좌표를 갖도록 "2 차원 모드"로 전환합니다.

ST_Force_3D — 지오메트리를 XYZ 모드로 전환합니다. 이 함수는 ST_Force_3DZ 의 별칭입니다.

ST_Force_3DZ — 지오메트리를 XYZ 모드로 전환합니다. 이 함수는 ST_Force_3D 의 별칭입니다

ST_Force_Collection — 지오메트리를 GeometryCollection 으로 변환합니다.

ST_GeomFromEWKB — Extended Well-Known Binary (EWKB)에서 지정된 ST_Geometry 값을 반환합니다.

ST_GeomFromEWKT — Extended Well-Known Text (EWKT)에서 지정된 ST_Geometry 값을 반환합니다.

ST_GeomFromGML — GML 지오메트리를 입력 받아 PostGIS 지오메트리 객체를 반환합니다.

ST_GeometryN — 지오메트리가 GEOMETRYCOLLECTION (MULTI) POINT (MULTI) LINESTRING, MULTICURVE 또는 (MULTI) POLYGON, POLYHEDRALSURFACE 경우 1 부터 N 번째 지오메트리를 반환. 그렇지 않은 경우 NULL 을 반환합니다.

ST_GeometryType — ST_Geometry 값의 지오메트리 유형을 반환합니다. 예: 'ST_Linestring', 'ST_Polygon', 'ST_MultiPolygon' 등. 이 함수는 GeometryType 함수와는 다릅니다.

= — A 의 바운딩 박스가 B 의 바운딩 박스와 같으면 TRUE 를 반환합니다. Double precision 바운딩 박스 사용.

&<| — A 의 바운딩 박스가 B 의 바운딩 박스와 겹치거나 그 아래에 있을 때 TRUE 를 반환합니다.

~= — A 의 바운딩 박스가 B 의 바운딩 박스와 같으면 TRUE 를 반환합니다.

ST_IsClosed — 라인 스트링의 시작과 끝 점이 일치하는 경우 TRUE 를 반환합니다. 다면체의 경우 표면 이 닫힙니다. (volumetric)

ST_Mem_Size — Geometry 가 소요하는 공간 (바이트)의 양을 반환합니다..

ST_NPoints — 지오메트리에서 점의 수 (vertexes)를 반환합니다.

ST_NumGeometries — 지오메트리가 GEOMETRYCOLLECTION (또는 MULTI *) 인 경우, 지오메트리의 수를 반환합니다. 단일 지오메트리는 1 을 반환, 없는 경우는 NULL 값을 반환합니다.

ST_NumPatches — PolyhedralSurface 의 면(faces)의 개수를 반환합니다. Polyhedral 지오메트리가 아닌 경우 NULL 값을 반환합니다.

ST_PatchN — PolyhedralSurface 또는 PolyhedralSurfaceM 지오메트리인 경우 처음을 1 부터 시작하는 N 번째 지오메트리(Face)를, 그렇지 않으면 NULL 값을 반환합니다.

ST_RemoveRepeatedPoints — 주어진 지오메트리에서 중복된 점을 제거한 지오메트리를 반환합니다.

ST_Rotate — 원점 축을 기준으로 시계 반대 방향으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_RotateX — X 축을 기준으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_RotateY — Y 축을 기준으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_RotateZ — Z 축을 기준으로 rotRadians(라디안 단위의 회전 각도)만큼을 회전한 지오메트리를 반환합니다.

ST_Scale — 매개변수와 함께 좌표를 곱하여 지오메트리를 새로운 크기로 조정합니다.예: ST_Scale(geom, Xfactor, Yfactor, Zfactor).

ST_Shift_Longitude — 지오메트리의 모든 점/정점을 읽은 후, 경도 좌표가 0 보다 작은 경우 360 을 더합니다. 그 결과는 180 도 중심 지도에 그려진 데이터의 0-360 도 중심의 버전이 됩니다. 이 함수는 경위도 좌표계(WGS 84 경위도)에만 유용합니다.

ST_Transform — 정수 매개변수에 의해 참조되는 SRID 값으로 좌표변환을 수행한 지오메트리를 반환합니다.

&& — A 의 2D 바운딩 박스가 B 의 2D 바운딩 박스와 교차할 때 TRUE 을 반환합니다.

&&& — A 의 3D 바운딩 박스가 B 의 3D 바운딩 박스와 교차할 때 TRUE 을 반환합니다.

13.10. 매트릭스 지원 PostGIS 함수

아래는 PostGIS 의 공간 특정 함수들의 알파펫 리스트와 함께 작동하는 공간 유형들의 종류 또는 변환하려는 OGC/SQL compliance 입니다.

✓ 는 본래적으로 타입 또는 substype 과 함께 작동하는 함수를 의미합니다.

이는 작동은 하지만 지오메트리에 캐스트를 사용하여 내장 변환 캐스트와 함께 “best sird” 공간 ref 로 변환한 뒤 다시 캐스트 back 하는 것을 의미합니다. 큰 영역들과 남/북근 영역에 대한 결과들은 예상과 다를 수 있으며 floating point junk 를 축적할 수 있습니다.

직접 입력 지원 보다 오히려 box3d 와 같은 것에 자동 캐스트 되기 때문에 함수는 유형에 따라 작동한다는 것을 의미합니다.

geom – 기본 2D 지오메트리 지원 (x,y).

geog – 기본 2D 지오그래피 지원 (x,y).

2.5D - 3 D/4D space 의 기본 2D 지오메트리(Z or M coord 를 가집니다).

PS – 다면체 표면(Polyhedral surfaces)

T – 삼각형들과 삼각 불규칙 네트워크 표면들(Triangles and Triangulated Irregular Network surfaces (TIN))

Function	*geom*	*geog*	*2.5D*	*Curves*	*SQL MM*	*PS*	*T*
Box2D	✓			✓		✓	✓
Box3D	✓		✓	✓		✓	✓
Find_SRID							
GeometryType	✓		✓	✓		✓	✓
ST_3DClosestPoint	✓		✓			✓	
ST_3DDFullyWithin	✓		✓			✓	
ST_3DDWithin	✓		✓		✓	✓	
ST_3DDistance	✓		✓		✓	✓	
ST_3DExtent	✓		✓	✓		✓	✓
ST_3DIntersects	✓		✓		✓	✓	
ST_3DLength	✓		✓				
ST_3DLength_Spheroid	✓		✓				
ST_3DLongestLine	✓		✓			✓	
ST_3DMakeBox	✓		✓				
ST_3DMaxDistance	✓		✓			✓	
ST_3DPerimeter	✓		✓				
ST_3DShortestLine	✓		✓			✓	
ST_Accum	✓		✓	✓		✓	✓
ST_AddMeasure	✓		✓				
ST_AddPoint	✓		✓				
ST_Affine	✓		✓	✓		✓	✓
ST_Area	✓	✓			✓	✓	
ST_AsBinary	✓	✓	✓	✓	✓	✓	✓
ST_AsEWKB	✓		✓	✓		✓	✓
ST_AsEWKT	✓	✓	✓	✓		✓	✓
ST_AsGML	✓	✓	✓			✓	✓
ST_AsGeoJSON	✓	✓	✓				
ST_AsHEXEWKB	✓		✓	✓			
ST_AsKML	✓	✓	✓				
ST_AsLatLonText	✓						
ST_AsSVG	✓	✓					
ST_AsText	✓	✓		✓	✓		
ST_AsX3D	✓		✓			✓	✓
ST_Azimuth	✓	✓					
ST_BdMPolyFromText	✓						
ST_BdPolyFromText	✓						
ST_Boundary	✓		✓		✓		
ST_Buffer	✓	😁			✓		
ST_BuildArea	✓						
ST_Centroid	✓				✓		
ST_ClosestPoint	✓						
ST_Collect	✓		✓	✓			
ST_CollectionExtract	✓						

Function	*geom*	*geog*	*2.5D*	*Curves*	*SQL MM*	*PS*	*T*
ST_CollectionHomogenize	✓						
ST_ConcaveHull	✓						
ST_Contains	✓				✓		
ST_ContainsProperly	✓						
ST_ConvexHull	✓		✓		✓		
ST_CoordDim	✓		✓	✓	✓	✓	✓
ST_CoveredBy	✓	✓					
ST_Covers	✓	✓					
ST_Crosses	✓				✓		
ST_CurveToLine	✓		✓	✓	✓		
ST_DFullyWithin	✓						
ST_DWithin	✓	✓					
ST_Difference	✓		✓		✓		
ST_Dimension	✓				✓	✓	✓
ST_Disjoint	✓				✓		
ST_Distance	✓	✓			✓		
ST_Distance_Sphere	✓						
ST_Distance_Spheroid	✓						
ST_Dump	✓		✓	✓		✓	✓
ST_DumpPoints	✓		✓			✓	✓
ST_DumpRings	✓		✓				
ST_EndPoint	✓		✓		✓		
ST_Envelope	✓				✓		
ST_Equals	✓				✓		
ST_Estimated_Extent	✓			✓			
ST_Expand	✓					✓	✓
ST_Extent	✓					✓	✓
ST_ExteriorRing	✓		✓		✓		
ST_FlipCoordinates	✓		✓	✓		✓	✓
ST_ForceRHR	✓		✓			✓	
ST_Force_2D	✓		✓	✓		✓	
ST_Force_3D	✓		✓	✓		✓	
ST_Force_3DM	✓			✓			
ST_Force_3DZ	✓		✓	✓		✓	
ST_Force_4D	✓		✓	✓			
ST_Force_Collection	✓		✓	✓		✓	
ST_GMLToSQL	✓				✓	✓	
ST_GeoHash	✓			✓			
ST_GeogFromText		✓					
ST_GeogFromWKB		✓		✓			
ST_GeographyFromText		✓					
ST_GeomCollFromText	✓				✓		
ST_GeomFromEWKB	✓		✓	✓		✓	✓
ST_GeomFromEWKT	✓		✓	✓		✓	✓

Function	*geom*	*geog*	*2.5D*	*Curves*	*SQL MM*	*PS*	*T*
ST_GeomFromGML	✓		✓			✓	✓
ST_GeomFromGeoJSON	✓		✓				
ST_GeomFromKML	✓		✓				
ST_GeomFromText	✓			✓	✓		
ST_GeomFromWKB	✓			✓	✓		
ST_GeometryFromText	✓				✓		
ST_GeometryN	✓		✓	✓	✓	✓	✓
ST_GeometryType	✓		✓		✓	✓	
|>>	✓						
<<|	✓						
~	✓						
@	✓						
=	✓	✓		✓		✓	
<<	✓						
|&>	✓						
&<|	✓			✓		✓	
&<	✓						
&>	✓						
>>	✓						
~=	✓					✓	
ST_HasArc	✓		✓	✓			
ST_HausdorffDistance	✓						
ST_InteriorRingN	✓		✓		✓		
ST_InterpolatePoint	✓		✓				
ST_Intersection	✓	😁			✓		
ST_Intersects	✓	✓			✓		
ST_IsClosed	✓		✓	✓	✓	✓	
ST_IsCollection	✓		✓	✓			
ST_IsEmpty	✓			✓	✓		
ST_IsRing	✓				✓		
ST_IsSimple	✓		✓		✓		
ST_IsValid	✓				✓		
ST_IsValidDetail	✓						
ST_IsValidReason	✓						
ST_Length	✓	✓			✓		
ST_Length2D	✓						
ST_Length2D_Spheroid	✓						
ST_Length_Spheroid	✓		✓				
ST_LineCrossingDirection	✓						
ST_LineFromMultiPoint	✓		✓				
ST_LineFromText	✓				✓		
ST_LineFromWKB	✓				✓		
ST_LineMerge	✓						

Function	*geom*	*geog*	*2.5D*	*Curves*	*SQL MM*	*PS*	*T*
ST_LineToCurve	✓		✓	✓			
ST_Line_Interpolate_Point	✓		✓				
ST_Line_Locate_Point	✓						
ST_Line_Substring	✓		✓				
ST_LinestringFromWKB	✓				✓		
ST_LocateAlong	✓						
ST_LocateBetween	✓						
ST_LocateBetweenElevations	✓		✓				
ST_LongestLine	✓						
ST_M	✓		✓		✓		
ST_MLineFromText	✓				✓		
ST_MPointFromText	✓				✓		
ST_MPolyFromText	✓				✓		
ST_MakeBox2D	✓						
ST_MakeEnvelope	✓						
ST_MakeLine	✓		✓				
ST_MakePoint	✓		✓				
ST_MakePointM	✓						
ST_MakePolygon	✓		✓				
ST_MakeValid	✓		✓				
ST_MaxDistance	✓						
ST_MemUnion	✓		✓				
ST_Mem_Size	✓		✓	✓		✓	✓
ST_MinimumBoundingCircle	✓						
ST_Multi	✓						
ST_NDims	✓		✓				
ST_NPoints	✓		✓	✓		✓	
ST_NRings	✓		✓	✓			
ST_Node	✓		✓				
ST_NumGeometries	✓		✓		✓	✓	✓
ST_NumInteriorRing	✓				✓		
ST_NumInteriorRings	✓				✓		
ST_NumPatches	✓		✓		✓	✓	
ST_NumPoints	✓				✓		
ST_OffsetCurve	✓						
ST_OrderingEquals	✓				✓		
ST_Overlaps	✓				✓		
ST_PatchN	✓		✓		✓	✓	
ST_Perimeter	✓	✓			✓		
ST_Perimeter2D	✓						
ST_Point	✓				✓		
ST_PointFromText	✓				✓		
ST_PointFromWKB	✓		✓	✓	✓		
ST_PointN	✓		✓	✓	✓		

Function	*geom*	*geog*	*2.5D*	*Curves*	*SQL MM*	*PS*	*T*
ST_PointOnSurface	✓		✓		✓		
ST_Point_Inside_Circle	✓						
ST_Polygon	✓		✓		✓		
ST_PolygonFromText	✓				✓		
ST_Polygonize	✓						
ST_Project		✓					
ST_Relate	✓				✓		
ST_RelateMatch							
ST_RemovePoint	✓		✓				
ST_RemoveRepeatedPoints	✓		✓			✓	
ST_Reverse	✓						
ST_Rotate	✓		✓	✓		✓	✓
ST_RotateX	✓		✓			✓	✓
ST_RotateY	✓		✓			✓	✓
ST_RotateZ	✓		✓	✓		✓	✓
ST_SRID	✓			✓	✓		
ST_Scale	✓		✓	✓		✓	✓
ST_Segmentize	✓						
ST_SetPoint	✓		✓				
ST_SetSRID	✓			✓			
ST_SharedPaths	✓						
ST_Shift_Longitude	✓		✓			✓	✓
ST_ShortestLine	✓						
ST_Simplify	✓						
ST_SimplifyPreserveTopology	✓						
ST_Snap	✓						
ST_SnapToGrid	✓		✓				
ST_Split	✓						
ST_StartPoint	✓		✓		✓		
ST_Summary	✓	✓					
ST_SymDifference	✓		✓		✓		
ST_Touches	✓				✓		
ST_TransScale	✓		✓	✓			
ST_Transform	✓			✓	✓	✓	
ST_Translate	✓		✓	✓			
ST_UnaryUnion	✓		✓				
ST_Union	✓				✓		
ST_WKBToSQL	✓				✓		
ST_WKTToSQL	✓				✓		
ST_Within	✓				✓		
ST_X	✓		✓		✓		
ST_XMax	✓		✓	✓			
ST_XMin	✓		✓	✓			
ST_Y	✓		✓		✓		

Function	geom	geog	2.5D	Curves	SQL MM	PS	T
ST_YMax	☑		✓	✓			
ST_YMin	☑		✓	✓			
ST_Z	✓		✓		✓		
ST_ZMax	☑		✓	✓			
ST_ZMin	☑		✓	✓			
ST_Zmflag	✓		✓	✓			
<#>	✓						
<->	✓						
&&	✓	✓		✓		✓	
&&&	✓		✓	✓		✓	✓

13.11. PostGIS 의 신규기능 및 개선되거나 변경된 기능들

13.11.1. 신규 및 PostGis 2.0 릴리즈에서 변경 또는 개선된 함수

아래의 함수들은 2.0. 출시판에 추가되고, 개선되거나 혹은 13.11.2 단락의 주요 변경내용들 입니다.

새로운 지오메트리 유형들: 2.0. 버전에서는 TIN(Triangular Irregular Networks)과 다면체 표면(Polyhedral surfaces)이 도입되었습니다.

토폴로지를 위해 매우 향상된 지원의 보다 자세한 정보를 위해서는 Chapter 11, Topology 를 참조하십시오.

PostGIS 2.0 에서는 래스터 형식 및 래스터 함수가 통합되었습니다. 여기서 열거하기에는 너무 많은 래스터 함수들이 존재함으로써 이용가능한 래스터 기능들에 관한 더 많은 정보를 보기 위해서는 Chapter 9, Raster Reference 를 참조하십시오. 초기 2.0. 이전 버전들은 실제 테이블 같은 raster_columns/raster_overviews 를 가지고 있었습니다. 이들은 출시 전에 views 로 변경되었습니다. ST_AddRasterColumn 와 같은 함수들은 삭제되고 AddRasterConstraints, DropRasterConstraints 들로 대체되었습니다. 결과적으로 래스터 테이블들을 생성하였던 일부 프로그램들은 변경이 필요할 수 있습니다.

Tiger Geocoder 는 TIGER 2012 인구 조사 데이터와 함께 작동하기 위해 업그레이드 되었으며, 이제 core PostGIS documentation 포함되었습니다. 역 geocoder 함수도 추가되었습니다. 보다 자세한 정보를 원하시면 Section 12.1, “Tiger Geocoder”를 참조하십시오.

래스터 밴드는 오직 아웃 DB 래스터의 첫 번째 256 밴드를 참조합니다.

&&& — AVAILABILITY: 2.0.0 은 A 의 3D 바운딩 박스가 B 의 3D 바운딩 박스와 교차할 시 true 를 반환합니다.

<#> — AVAILABILITY: 2.0.0 은 PostgreSQL 9.1+에 대해서만 오직 이용 가능합니다. 2 개의 지오메트리들의 바운딩 박스 사이 거리를 반환합니다. 포인트 / 포인트 체크를 위해 부동 소수점 정확도를 사용합니다. (기본이 되는 점 지오메트리의 배정 밀도 정확도(double)의 반대). 다른 지오메트리 유형들에 대한 부동 소수점 바운딩 박스 중심들 사이의 거리는 반환됩니다. KNN gist 기능을 사용하여 가장 가까운 이웃 한계와 distance ordering 을 하는데 유용합니다.

<-> — Availability: 2.0.0 은 PostgreSQL 9.1+에 대해서만 오직 이용가능합니다. 2 개의 점들사이의 거리를 반환합니다. 포인트 / 포인트 체크를 위해 부동 소수점 정확도를 사용합니다. (기본이 되는 점 지오메트리의 배정 밀도 정확도에의 반대). 다른 지오메트리 유형들에 대한 부동 소수점 바운딩 박스 중심들 사이의 거리는 반환됩니다. KNN gist 기능을 사용하여 가장 가까운 이웃 한계와 distance ordering 을 하는데 유용합니다.

AddEdge — AVAILABILITY: 2.0.0 GEOS >= 3.3.0 을 요구합니다. 지정된 선 스트링 지오메트리를 사용하는 특정 토폴로지 스키마 포인트 노드 테이블에 관련 시작 그리고 끝 포인틀를 추가하고 선 스트링에지를 에지 테이블에 추가합니다. 더하여 새로운(또는 기존의) 에지의 edge id 를 반환합니다.

AddFace — AVAILABILITY: 2.0.0 토폴로지에 기본 면을 등록하고 그것의 식별자를 얻습니다.

AddNode — AVAILABILITY: 2.0.0 은 특정 토폴로지 스키마의 노드에 포인트 노드를 추가하고 새로운 노드의 node id 를 반환합니다. 만약 포인트가 노드로서 이미 존재하고 있다면 존재하는 node id 는 반환됩니다.

AddRasterConstraints — AVAILABILITY: 2.0.0 래스터 행이 정기적으로 차단된 경우 나타내기 위한 플래그와 밴드 유형, 밴드, 정력, 블럭 크기, 확장, 공간 REF 에 대한 로드된 래스터 테이블에 래스터 제약 조건을 추가합니다. 유추할 수 있는 제약 조건에 대한 테이블과 함께 테이블은 반드시 로드되어야 합니다. 제약 조건 세팅이 완수되고 통보를 발포한 경우 true 을 반환합니다.

AsGML — AVAILABILITY: 2.0.0 topogeometry 의 GML 표현을 반환합니다.

CopyTopology — AVAILABILITY: 2.0.0 토폴로지 구조(노드, 에지, 면, 레이어 그리고 TopoGeometries)의 복사본을 만듭니다.

DropRasterConstraints — AVAILABILITY: 2.0.0 래스터 테이블 행을 참조하는 PostGIS 래스터 제약 조건들을 중단합니다. 데이터를 다시 로드할 필요가 있거나 래스터 행 데이터를 업데이크 할 필요가 있을 실 때 유용합니다.

Drop Indexes Generate Script — AVAILABILITY: 2.0.0 tiger 스키마와 유저 지정 스키마에서 모든 비 기본키와 비 고유 인덱스들을 중단하는 스크립트를 생성합니다. 스키마가 지정되어 있지 않은 경우 tiger_data 로 스키마가 디폴트됩니다.

Drop_State_Tables_Generate_Script —Availabilit: 2.0.0 상태 약어로 시작되는 지정된 스키마의 모든 테이블을 삭제하는 스크립트를 생성합니다. 아무런 스키마가 지정되지 않은 경우, tiger_data 로 스키마를 디폴트합니다.

Geocode_Intersection — Availability: 2.0.0 문자열 (또는 다른 정규화 된 주소)로 주소를 받아 NAD 83 long lat(EPSG:4269)의 포인트 지오메트리, 각각에 대한 정규화 된 주소, 그리고 rating 을 포함 가능한 위치의 집합을 출력합니다. Rating 이 낮을수록 match 일 가능성이 높아집니다. 결과들은 가장 낮은 rating 을 첫째로 분류됩니다. 최대 결과, 10 디폴트, 그리고 restrict_region (디폴트 NULL)을 선택적으로 제출할 수 있습니다.

GetEdgeByPoint — Availability: 2.0.0 GEOS >= 3.3.0 을 요구합니다. 주어진 점을 교차하는 에지의 edge-id 를 찾습니다.

GetFaceByPoint — Availability: 2.0.0 GEOS >= 3.3.0 을 요구합니다. 주어진 면을 교차하는 에지의 face-id 를 찾습니다

GetNodeByPoint — Availability: 2.0.0 - GEOS >= 3.3.0 을 요구합니다. 포인트 위치의 노드의 id 를 찾습니다.

GetNodeEdges – Availability:2.0 지정된 노드에 에지들의 주문 집합 사건을 반환합니다.

GetRingEdges — Availability: 2.0 지정된 에지와 링을 형성하는 에지들의 주문 집합을 반환합니다.

GetTopologySRID — Availability: 토폴로지의 이름이 주어진 topology.topology 테이블에서 토폴로지의 ID 를 반환합니다.

Get_Tract — Availability: 2.0.0 지오메트리가 위치되어 있는 tract 테이블의 필드 또는 인구 조사 tract 을 반환합니다. Tract 의 짧은 이름을 반환하는 것을 디폴트로 합니다.

Install_Missing_Indexes — Availability: 2.0.0 해당 열에서 사용되는 인덱스를 누락하고 추가 할 geocoder 조인(geocoder joins)와 필터 조건들(filter conditions)에 사용되는 키 행들을 가진 테이블들을 찾습니다.

Loader_Generate_Census_Script — Availability: 2.0.0 tiger_data 스키마에 Tiger 데이터, 스테이지 그리고 로드를 다운로드 하는 지정된 상태들을 위한 특정 플랫품 쉘 스크립트(shell script)을 생성합니다. 각 상태 스크립트는 독립적 레코드로 반환됩니다. 최신 버전은 Tiger 2010 구조적 변화를 지원하며 또한 인구 조사 요로, 블락 그룹(block groups) 그리고 블락 테이블을 로드합니다.

Loader_Generate_Script — Availability: 2.0.0 Tiger 2010 구조 데이터를 지원하고 인구 조사 tract(tract), 블락 그룹(bg) 그리고 블락들(tabblocks) 테이블들을 로드합니다. Tiger data, 스테이지 그리고 다운로드 하고 tiger_date 스키마로 로드하는 지정된 상태를 위한 특정 플랫폼 쉘 스크립트를 생성합니다. 각 상태 스크립트는 독립적 레코드로 반환됩니다. 최신 버전은 Tiger 2010 구조 변화들을 지원하며 인구 조사 tract, 블락 그룹 그리고 블락 테이블들을 로드합니다.

Missing_Indexes_Generate_Script – Availability: 2.0.0 행들 위 인덱스를 누락하는 geocoder 조인(joins)에 사용되는 주요 행들을 가진 모든 테이블을 찾으며 해당 테이블을 위한 인덱스를 정의하기 위해 SQL DDL 을 출력합니다.

Polygonize – Availabilit): 2.0.0 토폴로지 에지들에 의해 정의된 모든 면들을 찾고 등록합니다.

Reverse_Geocode – Availabilit): 2.0.0 알려진 공간 레퍼런스 SYS(known spatial ref sys)의 지오메트리 포인트을 받아 이론적으로 가능한 주소의 배열과 상호 거리의 배열을 포함하는 레코드를 반환합니다. include_strnum_range = true 인 경우, 크로스 거리에서 거리 범위를 포함합니다.

ST_3DClosestPoint – Availability: 2.0.0 G2 에 가장 가까운 G1 에서 3 차원 점을 반환합니다. 이는 3D 가장 짧은 라인의 첫 번째 점입니다. 이는 3D 의 가장 짧은 선의 첫번째 점입니다.

ST_3DDFullyWithin – Availability 2.0.0 모든 3D 지오메트리들이 지정된 자신들의 거리 이내에 있을 때 true 을 반환합니다.

ST_3DDWithin – Availability: 2.0.0 3d (z) geometry type 의 경우, 두 개의 지오메트리 3d 거리가 단위의 수내에 있을 시 true 를 반환합니다.

ST_3DDistance — Availability: 2.0.0 지오메트리 형식에 대한 전망된 단위의 두 개의 지오메트리 사이 3 차원 직교 최소 거리(공간 ref 기반)를 반환합니다.

ST_3DIntersects — Availability: 2.0.0 두 지오메트리가 3 차원 공간에서 "공간적으로 교차"하는 경우 TRUE 를 반환합니다. Point 및 LineString 만을 대상으로 합니다.

ST_3DLongestLine — Availability: 2.0.0 두 개의 지오메트리들 사이의 가장 긴 3 차원 선들을 반환합니다.

ST_3DMaxDistance — Availability: 2.0.0 지오메트리 형식에 대한 전망된 단위의 두 개의 지오메트리 사이 3 차원 직교 최소 거리(공간 ref 기반)를 반환합니다.

ST_3DShortestLine — Availability: 2.0.0 두 개의 지오메트리들 사이의 가장 짧은 3 차원 선들을 반환합니다.

ST_AddEdgeModFace — Availability: 2.0 새로운 에지(edge)를 추가하고, 이로 인해 면이 분할되는 경우, 원래의 면을 수정하고 새로운 면을 하나 추가합니다.

ST_AddEdgeNewFaces — Availability: 2.0 새로운 에지(edge)를 추가하고, 그렇게 함으로써 면 분할하는 경우, 원래의 면을 수정하고 두 개의 새로운 면을 추가합니다.

ST_AsGDALRaster — Availability: 2.0.0 – GDAL >= 1.6.0 을 요구합니다. Return 지정된 GDAL 래스터 포맷의 래스터 타일을 반환합니다. 래스터 포맷은 귀하의 컴파일된 라이브러리에 의해 지원되는 것 중 하나입니다. 귀하의 라이브러리에 의해 지원되는 포맷의 리스트를 얻기 위해서 ST_GDALRasters()을 사용하십시오.

ST_AsJPEG — Availability: 2.0.0 – 하나의 제이펙 (JPEG) 이미지 (바이트 배열)로 래스터 타일을 선택 밴드를 반환합니다. 아무런 밴드가 지정되지 않고 1 개 또는 3 개 이상 대역의 경우, 첫 번째 밴드가 사용됩니다. 만약 오직 3 개의 밴드일 경우 3 개의 밴드는 사용되며 RGB 에 매핑됩니다.

ST_AsLatLonText — Availability: 2.0 주어진 포인트의 도, 분, 초 표현을 반환합니다.

ST_AsPNG — Availability: 2.0.0 GDAL >= 1.6.0 을 요구합니다. 하나의 PNG 이미지 (바이트 배열)로 래스터 타일 선택된 밴드를 반환합니다. 래스터의 1, 3 또는 4 밴드가 지정되고 아무런 밴드가 지정되지 않으며, 모든 밴드가 사용됩니다. 만약 2 또는 4 개 이상의 밴드가 지정되거나 아무런 밴드가 지정되지 않으면 밴드 1 만이 오직 사용됩니다. 밴드 RGB 또는 RGBA 공간에 매핑됩니다

ST_AsRaster — Availability: 2.0.0 - GDAL >= 1.6.0 을 요구합니다. PostGIS 지오메트리를 PostGIS raster 로 변환합니다.

ST_AsTIFF — Availability: 2.0.0 - GDAL >= 1.6.0 을 요구합니다. 래스터 선택된 밴드를 하나의 TIFF 이미지 (바이트 배열)로서 반환합니다. 아무런 밴드도 지정되지 않는다면 모든 대역을 사용하려고 합니다.

ST_AsX3D — Availability: 2.0.0: ISO-IEC-19776-1.2-X3DEncodings-XML 은 X3D xml node element format: ISO-IEC-19776-1.2-X3DEncodings-XML 의 지오메트리를 반환합니다.

ST_Aspect — Availability: 2.0.0 제공되는 방위각, 고도, 밝기와 고도 가늠자 입력을 사용하여 고도 래스터 밴드의 가상 조명을 반환합니다. 지형을 시각화하는 데 유용합니다

ST_Band — Availability: 2.0.0 새로운 래스터로서 기존의 래스터의 하나이상의 밴드를 반환합니다. 기존의 래스터들에서 새로운 래스터들을 만드는데 유용합니다.

ST_BandIsNoData — Availability: 2.0.0 밴드가 nodate 값들로 채워져 있을 때 true 를 반환합니다.

ST_Clip — Availability: 2.0.0 입력 지오메트리에 의해 잘린 래스터를 반환합니다. 아무런 밴드도 지정되지 않았을 경우 모든 밴드가 반환됩니다. If crop 이 지정되지 않는 경우, true 는 출력 래스터가 잘렸다고 의미하는 것으로 간주됩니다.

ST_CollectionHomogenize — Availability: 2.0.0 주어진 지오메트리 켈렉션을 고려, 컨텐츠의 "가장 간단한" 표현을 반환합니다.

ST_ConcaveHull — Availability: 2.0.0 지오메트리의 오목한 덮개(hull)는 집합 내 모든 지오메트리들을 둘러싸는 오목 지오메트리를 나타냅니다. 수축되는 포장 재료를 떠오르실 수 있을 것입니다.

ST_Count — Availability: 2.0.0 래스터 범위 또는 래스터의 주어진 밴드의 픽셀을 수를 반환합니다. 아무런 밴드가 지정되지 않을 시 밴드는 1 디폴트. exclude_nodata_value 가 true 로 지정되었을 시 nodate 값과 동일하지 않는 픽셀들만 셀 것입니다.

ST_CreateTopoGeo — Availability: 2.0 지오메트리들의 컬렉션을 주어진 빈 토폴로지에 추가하고 성공을 설명하는 메시지를 반환합니다.

ST_Distinct4ma — Availability: 2.0.0 이웃의 최소 픽셀 값을 계산하는 래스터 처리 기능.

ST_FlipCoordinates — Availability: 2.0.0 X 및 Y 축 대칭으로 주어진 지오메트리의 버전을 반환합니다. 경도/적도 피쳐를 구축해본 적이 있거나 수정을 하는 사람들에게 아주 유용합니다.

ST_GDALDrivers — Availability: 2.0.0 - GDAL >= 1.6.0 을 요구합니다. 귀하의 lib gdal 에 의해 지원되는 래스터 포맷들의 리스트를 반환합니다. ST_AsGDALRaster 을 사용함으로써 귀하의 래스터를 출력할 수 있는 포맷들입니다.

ST_GeomFromGeoJSON — Availability: 2.0.0 - JSON-C >= 0.9 을 요구합니다. 지오메트리의 입력 geojson 표현으로 받아 PostGIS 기하학 오브젝트를 출력.

ST_GetFaceEdges — Availability: 2.0 일련 번호를 포함하는 aface 를 제한하는 주문 에지들의 집합을 반환합니다.

ST_HasNoBand — Availability: 2.0.0 주어진 밴드 숫자가 없는 경우 true 을 반환합니다. 아무런 밴드 숫자가 지정되지 않았을 경우 밴드 숫자 1 이 간주됩니다.

ST_HillShade — Availability: - 2.0.0 제공 방위각, 고도, 밝기와 고도 가늠자 입력을 사용하여 고도 래스터 밴드의 가상 조명을 반환합니다. 지형을 시각화하는 데 유용합니다.

ST_Histogram — Availability: 2.0.0 래스터 또는 래스터 커버리지 데이터 분배 별도의 빈 범위를 요약 히스토그램의 집합을 반환합니다. 지정하지 않으면 빈의 수가 자동으로 계산됩니다.

ST_InterpolatePoint — Availability: 2.0.0 제공된 포인트에 가까운 포인트의 지오메트리 측정 치수의 값을 반환합니다.

ST_IsValidDetail — Availability: 2.0.0 – GEOS >= 3.3.0 을 요구합니다. 지오메트리가 유효한지 아닌지 그리고 유효한지 않다면 그 이유는 무엇이고 위치가 어디인지에 관해 설명하는 valid_detail (valid,reason,location) 열을 반환합니다.

ST_IsValidReason — Availability: 2.0 – 플래그를 취하는 버전을 위한 GEOS >= 3.3.0 을 요구합니다. 지오메트리가 유효한지 안하지, 유효하지 않다면 그 이유는 무엇인지에 대한 텍스트를 반환합니다.

ST_MakeLine — Availability: 2.0.0 – 선 스트링 입력 요소들을 위한 지원을 소개합니다. 점 또는 선 지오메트리으로부터 선 스트링을 작성합니다.

ST_MakeValid — Availability: 2.0.0, GEOS-3.3.0 을 요구합니다. 정점을 잃지 않고 유요하지 않는 지오메트리를 유효하게 만드는 시도를 합니다.

ST_MapAlgebraExpr — Availability: 2.0.0 1 raster band 버전: 제공된 입력 래스터 밴드와 pixeltype 의 유효한 PostgreSQL 의 대수 연산을 적용하여 형성된 새로운 밴드 Raster 를 작성합니다. 아무런 밴드가 지정되지 않은 경우 대역 1 로 간주됩니다.

ST_MapAlgebraExpr — Availability: 2.0.0 2 raster band 버전: 제공된 입력 래스터 밴드와 pixeltype 의 유효한 PostgreSQL 의 대수 연산을 적용하여 형성된 새로운 밴드 Raster 를 작성합니다. 아무런 밴드가 지정되지 않은 경우 대역 1 로 간주됩니다. 그 결과로 생긴 raster 는 첫번째 raster 에 의해 정의된 그리드(grid)위 에서 정렬되며 (눈금, 기울이기 및 픽셀 코너) "extenttype"매개 변수에 의해 정의된 범위를 가집니다. "extenttype"의 값은: INTERSECTION, UNION, FIRST, SECOND.

ST_MapAlgebraFct — Availability: 2.0.0 1 밴드 버전- 제공된 입력 래스터 밴드와 pixeltype 의 유효한 PostgreSQL 의 대수 연산을 적용하여 형성된 새로운 밴드 Raster 를 작성합니다. 아무런 밴드가 지정되지 않은 경우 대역 1 로 간주됩니다..

ST_MapAlgebraFct — Availability: 2.0.0 2 band 버전 - 제공된 2 입력 래스터 밴드와 pixeltype 의 유효한 PostgreSQL 의 대수 연산을 적용하여 형성된 새로운 밴드 Raster 를 작성합니다. 아무런 밴드가 지정되지 않은 경우 대역 1 로 간주됩니다. 지정되지 않은 경우 범위 유형 디폴트는 INTERSECTION 입니다.

ST_MapAlgebraFctNgb — Availability: 2.0.0 1-band 버전: 사용자 정의 PostgreSQL 의 기능을 사용하여 지도 대수 가장 가까운 이웃. 입력 Raster 밴드의 값들의 이웃을 포함하는 PLPGSQL 유저 기능의 결과인 값들인 Raster 를 반환합니다.

ST_Max4ma — Availability: 2.0.0 이웃의 최대 픽셀 값들을 계산하는 래스터 처리 기능

ST_Mean4ma — Availability: 2.0.0 이웃의 픽셀 값들의 평균을 계산하는 래스터 처리 기능

ST_Min4ma — Availability: 2.0.0 이웃의 최소 픽셀 값들을 계산하는 래스터 처리 기능

ST_ModEdgeHeal — Availability: 두 개의 에지를 연결하는 노드를 삭제하고 첫번째 에지를 수정, 그리고 두번째 에지를 삭제함으로써 두개의 에지를 치유합니다. 지워진 노드의 id 는 반환됩니다.

ST_NewEdgeHeal — Availability: 2.0 두개의 에지를 연결하는 노드를 삭제하고 삭제된 에지들을 첫번째 주어졌던 에지와 같은 방향의 에지로 대체함으로써 두개의 에지를 대체합니다. 삭제된 노드의 id 를 반환합니다.

ST_Node — Availability: 2.0.0 – GEOS >= 3.3.0 요구합니다. 선 스트링의 집합을 노드합니다.

ST_NumPatches — Availability: 2.0.0 다면체 표면의 면들의 수를 반환합니다. 비 다면체 지오메트리의 경우 null 을 반환합니다.

ST_OffsetCurve — Availability: 2.0 – 은 GEOS >= 3.2 을 요구합니다, GEOS >= 3.3 와 함께 개선됩니다. 입력 행으로부터 주어진 거리와 측면에서의 오프셋 라인을 돌려줍니다. 센터 라인에 대해 평행 한 선을 계산하는 데 유용합니다.

ST_PatchN — Availability: 2.0.0 지오메트리가 POLYHEDRALSURFACE, POLYHEDRALSURFACEM 일 경우 1 부터 N 번째 (면) 반환합니다. 그렇지 않으면 NULL 을 반환합니다.

ST_Project — Availability: 2.0.0 라디안 미터 베어링 거리 (방위각)를 사용하여 시작 지점에서 투사된 POINT 를 반환합니다.

ST_Quantile — Availability: 2.0.0 샘플이나 인구의 경우에, 래스터 또는 래스터 테이블의 범위에 대한 분위수를 계산합니다. 따라서, 값은 래스터의 25 %, 50 %, 75 %의 백분위로 검사 할 수 있습니다.

ST_Range4ma — Availability: 2.0.0 이웃의 픽셀 값들의 범위를 계산한 래스터 처리 기능

ST_Reclass — Availability: 2.0.0 원본으로부터 재분류된 밴드 형식으로 구성된 새로운 래스터. Nband 는 변경되는밴드입니다. nband 는 지정되지 않았을 경우 1 로 간주됩니다. 모든 다른 밴드들은 변하지 않은 상태에서 반환됩니다. 케이스 사용: 16BUI band 를 8BUI 로 전환 그리고 볼 수 있는 형식으로 간단하게 렌더링합니다.

ST_RelateMatch — Availability: 2.0.0 - GEOS >= 3.3.0 을 요구합니다. intersectionMattrixPattern1 가 intersectionMatrixPattern2 암시할 경우 true 을 반환합니다.

ST_RemEdgeModFace — Availability: 2.0 에지를 제거하고 제거된 에지가 두개의 면을 분할할 경우 두개의 면 중 하나를 삭제하며 다른 하나가 두면의 공간 모두를 차지하도록 수정합니다.

ST_RemEdgeNewFace — Availability: 2.0 에지를 제거하고 제거된 에지가 두개의 면을 분할할 경우, 원본 면들을 삭제하고 새로운 면으로 그것들을 대체합니다.

ST_RemoveRepeatedPoints — Availability: 2.0.0 제거된 중복 포인트의 주어진 지오메트리 버전을 반환합니다.

ST_Resample — Availability: 2.0.0 은 GDAL 1.6.1+ 을 요구합니다. 특정 리샘플링 알고리즘. 새로운 차원들, 임의의 그리드 코너를 이용하 래스터와다른 래스터에 의해 주어졌거나 정의된 래스터 지오레퍼렌싱 속성들의 집합을 리샘플링 합니다. New pixel values are computed using NearestNeighbor (영어나 미국 스펠링), Bilinear, Cubic, CubicSpline 또는 Lanczos 리샘플링을 사용하여 새로운 픽셀을 계산합니다. 디폴트는 NearestNeighbor

ST_Rescale — Availability: 2.0.0 GDAL 1.6.1+ 요구합니다. 단지의 규모 (또는 픽셀 크기)를 조절하여 래스터를 리샘플링. 새로운 픽셀 값은 NearestNeighbor (영어 또는 미국 철자), 이중 선형, 큐빅, CubicSpline 또는 란초스 리샘플링 알고리즘을 사용하여 계산됩니다. 기본값은 NearestNeighbor 입니다.

ST_Reskew — Availability: 2.0.0 GDAL 1.6.1+ 요구합니다. 단지 그 기울기 (또는 회전 매개 변수)를 조정하여 래스터를 리샘플링. 새로운 픽셀 값은 NearestNeighbor (영어 또는 미국 철자), 이중 선형, 큐빅, CubicSpline 또는 란초스 리샘플링 알고리즘을 사용하여 계산됩니다. 기본값은 NearestNeighbor 입니다.

ST_SameAlignment — Availability: 2.0.0 래스터가 같은 기울기, 크기, 규모, 공간 ref 를 가지고 있을 때 true 을 반환합니다. 아무러 통지 없이 같은 위에 것들을 가지고 있지 않을 때 false 를 반환합니다.

ST_SetBandIsNoData — Availability: 2.0.0 밴드의 innodate 을 TRUE 로 반환합니다.

ST_SharedPaths — Availability: 2.0.0 requires GEOS >= 3.3.0. 두입력 선스트링/다중스트링에 의해 공유되는 부분을 포함하는 컬렉션을 반환합니다.

ST_Slope — Availability: 2.0.0 고도 래스터 밴드의 표면 경사를 반환합니다. 영역을 분석하는데 유용합니다.

ST_Snap — Availability: 2.0.0 GEOS >= 3.3.0 을 요구합니다. 레퍼런스 지오메트리의 정점에 입력 지오메트리의 세그먼트와 정점을 끼웁니다.

ST_SnapToGrid — Availability: 2.0.0 GDAL 1.6.1+을 요구합니다. 특정 리샘플링 알고리즘. 새로운 차원들, 임의의 그리드 코너를 이용하 래스터와다른 래스터에 의해 주어졌거나 정의된 래스터 지오레퍼렌싱 속성들의 집합을 리샘플링 합니다. New pixel values are computed using NearestNeighbor(영어나 미국 스펠링), Bilinear, Cubic, CubicSpline 또는 Lanczos 리샘플링을 사용하여 새로운 픽셀을 계산합니다. 디폴트는 NearestNeighbor.

ST_Split — Availability: 2.0.0 지오메트리를 나눔으로써 생기는 지오메트리 컬렉션을 반환합니다.

ST_StdDev4ma — Availability: 2.0.0 이웃의 픽셀 값들의 표줌 편차를 계산하는 함수를 처리하는 래스터.

ST_Sum4ma — Availability: 2.0.0 이웃의 픽셀 값들의 합을 계산하는 함수를 처리하는 래스터.

ST_SummaryStats — Availability: 2.0.0 raster 또는 raster 범위의 주어진 raster 밴드의 셈, 합, 평균, stddev, 최소값, 최대값을 구성하는 요약 통계를 반홥합니다. 아무런 밴드가 지정되지 않은 경우 밴드 1 으로 간주됩니다.

ST_Transform — Availability: 2.0.0 GDAL 1.6.1+을 요구합니다. 지정된 리샘플링 알고리즘을 사용하여 알려진 다른 공간 레퍼런스 시스템으로 알려진 레퍼런스 시스템의 래스터를 reprojection 합니다. 옵셥으로는 NearestNeighbor, Bilinear, Cubic, CubicSpline, Lanczos 이 있으며 디폴트는 NearestNeighbor 입니다.

ST_UnaryUnion — Availability: 2.0.0 - GEOS >= 3.3.0 을 요구합니다. ST_Union 와 같지만 지오메트리의 구성 요소 레벨에서 작동합니다.

ST_Union — Availability: 2.0.0 1 밴드로 구성된 하나의 래스터로 래스터 타일의 집합의 합집합을 반환합니다. 아무 밴드 unioning 을 지정하지 않은 경우, 대역 숫자 1 로 간주됩니다. 그 결과 래스터의 범위는 전체 집합의 범위입니다. 교차로의 경우, 결과 값은 p_expression 에 의해 정의된 다음 중 하나입니다: LAST – 아무것도 지정되지 않았을 때 디폴트, MEAN, SUM, FIRST, MAX, MIN.

ST_ValueCount — Availability: 2.0.0 값의 주어진 세트를 가진 래스터 (또는 래스터 범위)의 지정된 밴드의 픽셀 수의 픽셀 밴드 값 수를 포함하는 레코드 집합을 반환합니다. 아무런 밴드가 지정되지 않았을 경우에는 디폴트는 밴드 1 입니다. 기본적으로 값 픽셀은 계산되지 않습니다, 기르고 픽셀의 모든 값들은 출력이며 픽셀 밴드 값들은 가장 가까운 정수 값으로 반올림됩니다.

TopoElementArray_Agg — Availability: 2.0.0 타입 배열(topoelements), element_id 의 집합을 위한 topoelementarrary 를 반환합니다.

TopoGeo_AddLineString — Availability: 2.0.0 허용 오차 및 가능한 기존의 에지/면을 분할 함으로써 기존의 토폴로지에 선 스트링을 추가합니다.

TopoGeo_AddPoint — Availability: 2.0.0 2.0.0 허용 오차 및 가능한 기존의 에지/면을 분할 함으로써 기존의 토폴로지에 포인트를 추가합니다.

TopoGeo_AddPolygon — Availability: 2.0.0 2.0.0 허용 오차 및 가능한 기존의 에지/면을 분할 함으로써 기존의 토폴로지에 다각형을 추가합니다.

TopologySummary — Availability: 2.0.0 은 토폴로지 이름을 사용하고 토폴로지의 개체 유형의 요약 합계를 제공합니다.

Topology_Load_Tiger — Availability: 2.0.0 PostGIS 토폴로지에 tiger 데이터의 정의된 영역을 로드하고 토폴로지의 공간 레퍼런스로 tiger 데이터를 변환하며 토폴로지의 정밀도 허용 오차에 스냅을 합니다.

toTopoGeom — Availability: 2.0 은 단일 지오메트리로부터 새로운 topo 지오메트리를 생성합니다.

아래에 주어진 함수들은 PostGIS 2.0 에서 강화된 PostGIS 함수들입니다.

AddGeometryColumn – 강화(Enhanced): 2.0.0 use_typmod argument 인수가 소개됨. 제약 조건 기반 대신 typmod 지오메트리 행 생성에 기본 설정됩니다.

Box2D — Enhanced: 2.0.0 다면체 표면들을 위한 지원, Triangles 와 TIN 이 소개됨.

Box3D — Enhanced: 2.0.0 다면체 표면들을 위한 지원, Triangles 와 TIN 이 소개됨.

Geocode — Enhanced: 2.0.0 Tiger 2012 구조 데이터와 속도와 인코딩의 정확도를 향상시키기 위해 수정된 일부 로직을 서포트하고 중심선에서 거리 주소의 측면에 지점을 상쇄하기 위해 위치하고 있습니다. 최고의 결과를 규정하거나 반환하기 위해 새로운 파리미터 max_results 는 유용합니다.

GeometryType –Enhanced: 2.0.0 다면체 표면들을 위한 지원, Triangles 과 TIN 이 도입됨.

Populate_Geometry_Columns — Enhanced: 2.0.0 행들이 typmodifiers 또는 체크 제약조건과 함께 생성되었을 때 컨트롤하는 것을 허용하는 use_typmod 선택적 인수가 도입됨.

ST_Intersection — Enhanced: 2.0.0 - 래스터 공간에서 Intersection 이 도입되었습니다. 초기 2.0.0 이전 버전에서는 벡터 공간에서 수행된 Intersection 만이 지원되었습니다.

ST_Intersects — Enhanced: 2.0.0 지원 raster/raster intersects 가 도입.

ST_Value — Enhanced: 2.0.0 exclude_nodata_value 선택적 인수가 추가됨.

ST_3DExtent — Enhanced: 2.0.0 다면체 표면들을 위한 지원, Triangles 와 TIN 이 도입됨.

ST_Accum — Enhanced: 2.0.0 다면체 표면들을 위한 지원, Triangles 와 TIN 이 도입됨.

ST_Affine — Enhanced: 2.0.0 다면체 표면들을 위한 지원, Triangles 와 TIN 이 도입됨.

ST_Area — Enhanced: 2.0.0 – 2D 다면체 표면들을 위한 지원이 도입됨.

ST_AsBinary — Enhanced: 2.0.0 다면체 표면들을 위한 지원, Triangles 와 TIN 이 도입됨.

ST_AsBinary — Enhanced: 2.0.0 높은 좌표 차원에 대한 지원이 도입되었습니다.

ST_AsBinary — Enhanced: 지오그래피의 특정 엔디안(endian) 지정 2.0.0 지원이 도입되었습니다.

ST_AsEWKB — 2.0.0 다면체 표면들을 위한 지원, Triangles 와 TIN 이 도입됨.

ST_AsEWKT — Enhanced: 2.0.0 2.0.0 다면체 표면들을 위한 지원, Triangles 와 TIN 이 도입됨.

ST_AsGML — Enhanced: 2.0.0 prefix 가 도입되었습니다. 선들을 위한 Curve tag 대신 선스트링을 이용하는 것을 허용하기 위해 GML3 를 위한 옵션 4 가 도입되었습니다. 박스를 출력하기 위해 옵션 32 가 도입되었습니다.

ST_AsKML — Enhanced: 2.0.0 - prefix 네임 스페이스를 추가합니다. 기본값은 prefix 입니다.

ST_Azimuth — Enhanced: 2.0.0 지오그래피를 위한 지원이 도입되어습니다. support for geography was introduced.

ST_ChangeEdgeGeom — Enhanced: 2.0.0 topological consistency enforcement 을 추가합니다.

ST_Dimension — Enhanced: 2.0.0 다면체 표면들과 TINs 를 위한 지원이 도입되었습니다.비어있는 지오메트리가 주어진다면 더 이상 예외는 발생하지 않습니다.

ST_Dump — Enhanced: 2.0.0 다면체 표면들과 TIN 를 위한 지원이 도입되었습니다.

ST_DumpPoints — Enhanced: 2.0.0 삼각형들과 TIN 를 위한 지원이 도입되었습니다.

ST_Expand — Enhanced: 2.0.0 삼각형들과 TIN 를 위한 지원이 도입되었습니다.

ST_Extent — Enhanced: 2.0.0 다면체 표면들과 TINs 를 위한 지원이 도입되었습니다.

ST_ForceRHR — Enhanced: 2.0.0 다면체 표면들과 TINs 를 위한 지원이 도입되었습니다.

ST_Force_2D — Enhanced: 2.0.0 다면체 표면들과 TINs 를 위한 지원이 도입되었습니다.

ST_Force_3D — Enhanced: 2.0.0 다면체 표면들과 TINs 를 위한 지원이 도입되었습니다.

ST_Force_3DZ — Enhanced: 2.0.0 다면체 표면들을 위한 지원이 도입되었습니다.

ST_Force_Collection — Enhanced: 2.0.0 다면체 표면들을 위한 지원이 도입되었습니다.

ST_GMLToSQL — Enhanced: 2.0.0 다면체 표면들과 TINs 를 위한 지원이 도입되었습니다

ST_GMLToSQL — Enhanced: 2.0.0 디폴트 srid 선택적 파라미터가 추가되었습니다.

ST_GeomFromEWKB — Enhanced: 2.0.0 다면체 표면들과 TIN 를 위한 지원이 도입되었습니다.

ST_GeomFromEWKT — 2.0.0 다면체 표면들과 TIN 를 위한 지원이 도입되었습니다.

ST_GeomFromGML — Enhanced: 2.0.0 다면체 표면들과 TIN 를 위한 지원이 도입되었습니다.

ST_GeomFromGML — Enhanced: 2.0.0 디폴트 srid 선택적 파라미터가 추가되습니다.

ST_GeometryN — Enhanced: 2.0.0 삼각형들과 TIN 를 위한 지원이 도입되었습니다.

ST_GeometryType — Enhanced: 2.0.0 다면체 표면들을 위한 지원이 도입되었습니다.

ST_IsClosed — Enhanced: 2.0.0 다면체 표면들을 위한 지원이 도입되었습니다.

ST_MakeEnvelope — Enhanced: 2.0: SRID 를 지정하지 지정하지 않고 evelope 를 지정할 수 있는 능력이 도입되었습니다.

ST_MakeValid — Enhanced: 2.0.1, 속도 향상은 GEOS-3.3.4 을 요구합니다.

ST_NPoints — Enhanced: 2.0.0 다면체 표면들을 위한 지원이 도입되었습니다.

ST_NumGeometries — Enhanced: 2.0.0 다면체 표면들, 삼각형들 그리고 TIN 을 위한 지원이 도입되었습니다.

ST_Relate — Enhanced: 2.0.0 - 경계 노드 규칙을 지정하기 위한 지원이 추가되었습니다 (GEOS >= 3.0 이 요구됨).

ST_Rotate — Enhanced: 2.0.0 다면체 표면들, 삼각형들 그리고 TIN 을 위한 지원이 도입되었습니다.

ST_Rotate — Enhanced: 2.0.0 회전의 원점을 지정하는 추가 매개 변수가 추가되었습니다.

ST_RotateX — 2.0.0 다면체 표면들, 삼각형들 그리고 TIN 을 위한 지원이 도입되었습니다.

ST_RotateY — 2.0.0 다면체 표면들, 삼각형들 그리고 TIN 을 위한 지원이 도입되었습니다.

ST_RotateZ — Enhanced: 2.0.0 다면체 표면들, 삼각형들 그리고 TIN 을 위한 지원이 도입되었습니다.

ST_Scale — Enhanced: 2.0.0 다면체 표면들, 삼각형들 그리고 TIN 을 위한 지원이 도입되었습니다.

ST_Shift_Longitude — Enhanced: 2.0.0 다면체 표면들, 삼각형들 그리고 TIN 을 위한 지원이 도입되었습니다.

ST_Transform — Enhanced: 2.0.0 다면체 표면들을 위한 지원이 도입되었습니다.

ValidateTopology — Enhanced: 2.0.0 이전 버전에 존재했던 false positives 를 위한 보다 효율적인 에지 감지 및 수정.

&& — Enhanced: 2.0.0 다면체 표면들을 위한 지원이 도입되었습니다.

13.11.2. PostGIS 2.0 에서 방식이 변경된 함수

아래의 함수들은 PostGIS 2.0.에서 방식의 변화를 준 PostGIS 함수들이며, 애플리케이션 변경을 요구할 수도 있습니다.

대부분 사용되지 않는 함수는 제거되었습니다. 이들은 1.2 이후 일부 문서화되지 않은 내부 기능 때문에 문서화되지 않은 함수들입니다. 문서화 되지 않는 함수를 사용한다면 그 함수는 아마도 사용되지 않거나, 사용되지 않을 기능 또는 내부기능이므로 반드시 피해할

기능들입니다. 만약 사용되지 않는 함수들에 의존하는 애플리케이션 혹은 툴들을 사용한다면 보다 많은 정보를 위해 자세한 기능들을 참조하십시오.

지오메트리의 바운딩 상자가 float4 에서 배정 밀도 (float8)로 변경되었습니다. 이는 바운딩 박스 오퍼레이터와 바운딩 박스를 지오메트리에 캐스팅하여 얻는 값에 영향을 끼칠 것입니다. 예) ST_SetSRID(abbox) 는 view port 쿼리을 변화시켜 이전 버전 보다 PostGIS 2.0.+에서는 더욱 더 정확한 값을 반환할 것입니다.

인수 hasnodata 와 같은 의미를 가지나, 목적에서 더 명확한 exclude_nodate_value 로 인수 hasnodata 를 대체합니다.

AddGeometryColumn – 변경(Changed): 2.0.0 geometry_columns 은 시스템 카탈로그로부터 읽는 뷰이기 때문에 이 함수는 더 이상 geometry_columns 을 업데이트하지 못합니다. 기본적으로 이 또한 제약 조건을 생성하지 않습니다. 하지만 대신에 PostgreSQL 의 유형 식별자 행동을 사용합니다. 따라서 예를 들어 함수와 함께 wgs84 POINT column 을 구축하는 것은: ALTER TABLE some_table ADD COLUMN geom geometry(Point,4326); 과 같습니다.

AddGeometryColumn — Changed: 2.0.0 제약 조건의 이전 방식을 요구한다면 the default use_typmod 을 사용하십시오. 그러나 false 로 설정하십시오, .

AddGeometryColumn — Changed: 2.0.0 뷰들은 더 이상 수동으로 geometry_columns 에 등록될 수 없습니다. 하지만 geometry typmod tables geometries 에 설치되고 랩퍼(wrapper) 함수들 없이 사용된 뷰들은 그들의 부모 테이블 행의 행동에 의해 스스로 올바르게 등록할 것입니다. 다른 지오메트리들을 출력하는 지오메트리 함수들을 사용하는 뷰들은 이러한 뷰 지오메트리 행들이 geometry_columns 에 올바르게 등록되기 위해 typmod 지오메트리에 캐스트 될 필요가 있습니다. 참조하십시오.

DropGeometryColumn — Changed: 2.0.0 이 기능은 이전 버전과의 호환성을 위해 제공됩니다. geometry_columns 은 이제 시스템 카탈로그에 대한 뷰이기 때문에, 귀하는 ALTER 테이블을 사용하여 다른 테이블 컬럼과 같은 지오메트리 열을 삭제할 수 있습니다.

DropGeometryTable — Changed: 이 기능은 이전 버전과의 호환성을 위해 제공됩니다. 지금 geometry_columns 이제 시스템 카탈로그에 대한 뷰이기 때문에, 귀하는 DROP 테이블을 사용하여 다른 테이블 컬럼과 같은 지오메트리 열을 삭제 수 있습니다다.

Populate Geometry Columns — Changed: 2.0.0 디폴트로 이제는 체크 제약 조건 대신 유형 식별자를 사용하여 지에메트리 유형들을 제약할 수 있습니다. 새로운 use_typmod 와 이것을 false 로 세팅함으로써 여전히 check constraint behavior 를 사용하실 수 있습니다..

Box3D — Changed: 2.0 이전 버전에서는 box3d 대신 box2d 를 였습니다. Box2d 는 사용되지 않는 기능이므로 box3d 로 변경되었습니다.

ST_ScaleX — Changed: 2.0.0. WKTRaster 버전에서 이는 ST_PixelSizeX 이라 불렸습니다.

ST_ScaleY — Changed: 2.0.0 WKTRaster 버전에서 이는 ST_PixelSizeY 이라 불렸습니다.

ST_SetScale — Changed: 2.0.0 2.0.0 WKTRaster 버전에서 이는 ST_PixelSize 이라 불렸습니다. 2.0.0.에서 이는 변화되었습니다.

ST_SetValue — Changed: 2.0.2 ST_SetValue ()의 지오메트리 변형는 이제 입력 래스터 및 지오메트리이 동일한 SRID 가 있는지 확인하기 위해 체크합니다.

ST_3DExtent — Changed: 2.0.0 이전 버전에서는 ST_Extent3D 로 불리곤 하였습니다.

ST_3DLength — Changed: 2.0.0 이전 버전에서는 ST_3DLength 로 알려져 있습니다.

ST_3DLength_Spheroid — Changed: 2.0.0 In prior versions 이전 버전에서는 다중 스트링이 아니거나 선 스트링이 아닌 모든 것을 위해 0 을 반환하곤 하였습니다. 그리고 2.0.0.에서는 다각형이 주어진다면 경계를 반환합니다.

ST_3DLength_Spheroid — Changed: 2.0.0 이전 버전에서는 ST_Length3d_Spheroid 이라 불리곤 하였습니다.

ST_3DMakeBox — Changed: 2.0.0 이전 버전에서는 ST_MakeBox3D 라 불리곤 하였습니다.

ST_3DPerimeter — Changed: 2.0.0 이전 버전에서는 ST_Perimeter3D 라 불리곤 하였습니다.

ST_AsGML — Changed: 2.0.0 은 args 라 이름 붙여진 디폴트를 사용합니다.

ST_AsGeoJSON — Changed: 2.0.0 기본 args 와 명명된 args 를 지원합니다.

ST_AsKML — Changed: 2.0.0 - 기본 args 와 명명된 args 를 사용합니다.

ST_AsSVG — Changed: 2.0.0 기본 args 와 명명된 args 를 사용하기 위함입니다.

ST_EndPoint — Changed: 2.0.0 더 이상 단일 지오메트리 다중선스트링과 함께 작동하지 않습니다. PostGIS 의 구 버전에서는—단일 선 다중선스트링이 이 함수와 아주 잘 작동하고 시작점을 반환하였습니다. 2.0.0 에서는 다른 다중선스트링과 마찬가지로 NULL 을 반환합니다. 구 행동은 문서화 되지 않는 피쳐였습니다. 그러나 선스트링으로 자신들의 데이터가 가정되었다고 생각했던 사람들은 2.0.0. NULL 반환을 경험하게 될 것입니다.

ST_GeomFromText — Changed: 2.0.0 PostGIS 의 이전 버전. ST_GeomFromText('GEOMETRYCOLLECTION(EMPTY)')가 허용되었습니다. 이것은 이제 PostGIS 2.0.0 에서 SQL/MM standards 함께 더 잘 합치합니다. 이것은 이제 반드시 ST_GeomFromText('GEOMETRYCOLLECTION EMPTY')로 적혀야 합니다.

ST_GeometryN — Changed: 2.0.0 단일 지오메트리에 대해 이전 버전들은 NULL 을 반환할 것입니다. 이는 geometry for ST_GeometryN(..,1) case 을 반환하기 위해 변경되었습니다.

ST_IsEmpty — Changed: 2.0.0 PostGIS 의 이전 버전들. ST_GeomFromText('GEOMETRYCOLLECTION(EMPTY)'가 허용되었습니다. 이제 이것은 PostGIS 2.0.0 에서 불법이며 SQL/MM standards 와 더 잘 합치됩니다.

ST_Length — Changed: 2.0.0 Breaking change – 이전 번전에서는 유형 지오그래피의 다중/다각에 이것을 적용하는 것이 다각형/다중다각형의 경계를 가져다 주었습니다. 2.0.0 에서는 지오메트리 행동과 같은 선상으로 0 을 반환하도록 변경되었습니다. 다각형은 경계를 원하신다면 ST_Perimeter 을 이용하십시오.

ST_LocateAlong — Changed: 2.0.0 이전 버전에서 이것은 ST_Locate_Along_Measure 라 불리곤 하였습니다. 이 구 이름은 반대되고 향후에 삭제되어 질 것입니다만 현재 계속 사용가능합니다.

ST_LocateBetween — Changed: 2.0.0 – 이전 버전에서 ST_Locate_Between_Measures 이라 불리곤 하였습니다. 이 구 이름은 반대되고 향후에 삭제되어 질 것입니다만 현재 계속 이전 버전과 호환성을 위해 사용 가능합니다.

ST_ModEdgeSplit — Changed: 2.0 – 이전 버전에서는 ST_ModEdgesSplit 이라 잘못 이름 지어졌습니다.

ST_NumGeometries — Changed: 2.0.0 이전 버전에서는 지오메트리가 컬렉션/다중 유형이 아닐 경우 NULL 을 반환했습니다. 2.0.0+에서는 이제 단일 지오메트리를 위해 1 을 반환합니다. 예)POLYGON, LINESTRING, POINT.

ST_PointN — Changed: 2.0.0 단일 지오메트리 다중스트링과 더 이상 함께 작동하지 않습니다. PostGIS 의 이전 버전에서는 – 단일 선 다중선스트링이 이 함수와 아주 잘 작동하고 시작점을 반환했었습니다. 2.0.0 에서는 다른 다중선스트링과 마찬가지로 NULL 을 반환합니다.

ST_Simplify — Changed: 2.0.0 – 이 함수의 행동은 보다 더 일정하게 되었습니다. 이로 인해 일부 사람들은 기쁘지 않습니다.

ST_Simplify — Changed: 2.0.2 – 이 함수의 행동은 1.5 버전의 방식으로 되돌아갔습니다. 이로 인해 최근 보다 더 일정한 행동의 버전에 만족하던 사람들에게 긴 논란과 고통을 던져주었습니다. 그러므로 아마 새로운 방식으로 변화될 가능성이 있습니다.

ST_StartPoint — Changed: 단일 지오메트리 다중스트링과 더 이상 함께 작동하지 않습니다. PostGIS 의 이전 버전에서는 – 단일 선, 다중선스트링이 이 함수와 아주 잘 작동하고 시작점을 반환했었습니다. 2.0.0 에서는 다른 다중선스트링과 마찬가지로 NULL 을 반환합니다.이전 방식은 문서화 되지 않는 피쳐였습니다. 하지만 자신들의 데이터가 선 스트링으로 저장되었다고 생각했던 사람들은 2.0 에서는 NULL 반환을 경험하게 될 것입니다.

13.11.3. PostGIS 1.5 에서 신규 또는 개선, 변경된 함수

아래의 함수들은 minor release 에서 도입되고 기능이 향상된 PostGIS 함수들입니다.

PostGIS_LibXML_Version — Availability: 1.5 Returns the version number of the libxml2 library.

ST_AddMeasure — Availability: 1.5.0 선형 시작과 끝 점 사이의 보간 된 측정 요소와 파생 된 형상을 반환합니다. 지오메트리가 측정 치수를 가지지 않을 경우, 하나가 추가됩니다.

지오메트리가 측정 치수를 가지고 있을 경우 새 값으로 덮어 쓰기됩니다. 오직 선 스트링 과 다중 선 스트링이 지원됩니다.

ST_AsBinary — Availability: 1.5.0 지오그래피 지원이 도입되었습니다. SRID 메타 데이터 없이 지오메트리/지오그래피의 잘 알려진 바이너리 (well—known binary)를 표현을 반환합니다.

ST_AsGML — Availability: 1.5.0 지오그래피 지원이 도입되었습니다. GML 2 또는 3 형식으로 지오메트리를 반환합니다.

ST_AsGeoJSON — Availability: 1.5.0 지오그래피 지원이 도입되었습니다. GeoJSON 형식으로 지오메트리를 반환합니다.

ST_AsText — Availability: 1.5 – 지오그래피를 위한 지원이 도입되었습니다. SRID 메타 데이터 없이 지오메트리/지오그래피의 잘 알려진 텍스트 (WKT) 표현을 반환합니다.

ST_Buffer — Availability: 1.5 - ST_Buffer 는 다른 엔드캡(endcaps)과 조인유형을 지원하기 위해 강화되었습니다. 선스트링을 둥근 가장자리가 아닌 사각형 가장자리와 평평한 다각형 길로 변환하는 것과 같이 경우에 아주 유용합니다. 지오그래피를 위한 얇은 래퍼(wrapper)가 추가되었습니다. 이는 고급 지오메트리 기능의 장점을 취하기 위해 - requires GEOS >= 3.2 을 요구합니다. (T): Geometry 유형일 경우: 이 지오메트리로부터의 거리가 같거나 작은 모든 점들을 포함하는 지오메트리를 반환합니다. 계산은 지오메트리의 공간 좌표계를 따릅니다. Geography 유형일 경우: 평면 변환 래퍼를 사용합니다. 1.5 버전 이후부터 버퍼 옵션으로 끝점, 꼭지점 처리에 대한 스타일을 설정할 수 있습니다. buffer_style 옵션들: quad_segs=#, endcap=round|flat|square, join=round|mitre|bevel, mitre_limit=#.#

ST_ClosestPoint — Availability: g2 에 가까운 g2 의 2 차원 포인트를 반환합니다.가장 짧은 선의 첫번째 점입니다.

ST_CollectionExtract — Availability: 1.5.0 (다중)지오메트리를 고려, returns a 지정된 유형의 요소들만으로 구성된 (다중)지오메트리를 반환합니다.

ST_Covers — Availability: 1.5 – 지오그래피를 위한 지원이 도입되었습니다.지오메트리 B 의 어떤 점도 지오메트리 A 의 외부에 존재하지 않을 때 1 (TRUE)을 반환합니다.

ST_DFullyWithin — Availability: 1.5.0 모든 지오메트리들이 서로의 지정된 거리 내에 있을 시 true 을 반환합니다.

ST_DWithin — Availability: 1.5.0 지오그래피를 위한 지원이 도입되었습니다. 모든 지오메트리들이 서로의 지정된 거리 내에 있을 시 true 을 반환합니다. 지오메트리 단위는 공간 ref 이고 지오그래피 단위는 미터이며 치수는 use_spheroid=true (회전 타원체 주변)로 디폴트되어 있습니다. 빠른 체크를 위해서는, 회전 타원를 따라 측하기 위한 use_spheroid=false.

ST_Distance — Availability: 1.5.0 지오그래피 지원이 1.5.에 도입되었습니다. 크거나 많은 정점 지오메트리을 더 잘 처리하기 위한 평면을 위한 속도 향상. 지오메트리 타입에 대해 예상된 단위의 두 지오메트리 사이의 가장 짧은 2 차원 직교거리(공간 ref 기반)를 반환합니다. 지오그래피를 타입입의 경우 미터 단위의 두 지오그래피 사이의 가장 짧은 회전 차원체 거리를 반환 .

ST_Distance_Sphere — Availability: 1.5 – 포인트 외 다른 지오메트리 유형들을 위한 지원이 도입되었습니다. 이전 버전은 오직 포인트하고만 작동하였습니다. 두 적/위도 지오메트리 사이의 미터의 최소 거리를 반환합니다. 회전 타원체 지구그리고 6370986 반경 미터를 사용합니다. ST_Distance_Spheroid 보다 빠르지만 덜 정확합니다. PostGIS 1.5 이전 버전은 오직 포인트를 위해서만 구현되었습니다.

ST_Distance_Spheroid — Availability: 1.5 – 포인트 외 다른 지오메트리 유형들을 위한 지원이 도입되었습니다. 이전 버전은 오직 포인틀들 고만 작동하였습니다. 주어직 특정 회전 차원체의 두개의 적/위도 사이의 최소 거리를 반환합니다. 1

ST_DumpPoints — Availability: 1.5.0 지오메트리를 만드는 모든 포인트의 geometry_dump (geom,path) 열들을 반환합니다..

ST_Envelope — Availability: 1.5.0 float4 대신 배정 밀도를 출력하기 위해 행동이 바뀌어습니다. 공급된 지오메트리의 배정 밀도(float8) 바운딩 박스을 나타내는 지오메트리를 반환합니다. a geometry representing the double precision (float8) bounding box of the supplied geometry.

ST_GMLToSQL — Availability: 1.5, 은 libxml2 1.6+ 요구합니다. GML 표현으로 부터의 특정 ST_Geometry 값을 반환합니다. 이것은 ST_GeomFromGML 의 별칭입니다.

ST_GeomFromGML — Availability: 1.5, libxml2 1.6+ 요구합니다. 지오메트리의 입력 GML 표현을 취하고 PostGIS 지오메트리 오브젝트를 출력합니다.

ST_GeomFromKML — Availability: 1.5,libxml2 2.6+ 지오메트리의 KML 표현을 취하고 PostGIS geometry 오브젝트를 출력합니다.

~= — Availability: 1.5.0 changed behavior A 의 바운딩 박스가 B 의 바운딩 박스와 같을 때 True 를 반환합니다.

ST_HausdorffDistance — Availability: 1.5.0 - requires GEOS >= 3.2.0 을 요구합니다. 두 지오메트리 사이의 hausdorff 거리를 반환합니다. 기본적으로 2 지오메트리가 얼마나 서로 다르거나 비슷한가에 대한 측정입니다. 단위들은 지오메트리의 공간 레퍼런스 시스템의 단위에 있습니다.

ST_Intersection — Availability: 1.5 지오그래피 타입 유형을 위한 지원이 도입되었습니다. (T)는 geomA 와 geomB 가 공유하는 포인트를 나타내는 지오메트리를 반환합니다.지오그래피 구현은 교차를 하기 위해 변환을 한 뒤 WGS84 으로 재전환합니다.

ST_Intersects — Availability: 1.5 지오그래피를 위한 지원이 도입되었습니다. 지오메트리/지오그래피가 2D 에서 “공간적으로 교차”-(어떤 공간을 공유할때)할 때 TRUE 을 반환합니다. 그리고 교차하지 않을 때 (분리될때) FALSE 를 반환합니다. Geography 에 허용 오차는 0.00001 미터 입니다 (그러므로 가까운 어떤 점들이라고 서로 교차한다고 간주됩니다.)

ST_Length — Availability: 1.5.0 지오메트리 지원이 1.5 에서 도입되었습니다. 선스트링 이거나 다중 스트링일 경우 지오메트르의 2d 길이를 반환합니다. 지오메트리는 공간 ref 단위이고 지오그래피는 meters 단위입니다 (기본 값 회전반원체).

ST_LongestLine — Availability: 1.5.0 두 지오메트리의 가장 긴 2 차원 선 포인트를 반환합니다.함수가 찾는 것보다 더 많을 경우 함수는 오직 가장 긴 선만 반환합니다.반환된 선은 언제나 g1 에서 시작되고 g2 에서 끝납니다. st_maxdistance 가 g1 그리고 g2 를 위해 반환할 것이므로 이 함수가 반환한 선의 길이는 언제나 같습니다.

ST_MakeEnvelope — Availability: 1.5 주어진 최소 그리고 최대값으로부터 형성된 사각 다각형을 생성합니다.입력값은 반드시 SRID 에 의해 규정된 SRS 이어야 합니다.

ST_MaxDistance — Availability: 1.5.0 예상된 단위의 두 지오메트리 사이의 가장 긴 2 차원 거리를 반환합니다.

ST_ShortestLine — Availability: 두 지오메트리 사이의 가장 짧은 2 차원 선들이 반환됩니다.

&& — Availability: 1.5.0 지오그래피를 위한 지원이 도입되었습니다. A 의 2D 바운딩 박스가 B 의 2D 바운딩 박스와 교차할 때 true 가 반환됩니다.

13.11.4. PostGIS 1.4 에서의 신규 또는 개선, 변경된 함수

아래의 함수들은 1.4.릴리즈에서 도입되거나, 향상된 PostGIS 함수들입니다.

Populate_Geometry_Columns – 지오메트리 행들이 유형 식별자들과 함께 정의되고 적절한 공간 제약조건을 가지기를 보장합니다. 이는 geometry_columns view 에 적절히 등록되도록 보장합니다. 기본적으로 유형 식별자가 없는 모든 지오메트리 행들을 식별자들이 있는 지오메트리 행들로 변환시킵니다. 구 행동을 얻기 위해서는 use_typmod=false Availability: 1.4.0 을 설정하십시오.

ST_AsSVG – 주어진 지오메트리 또는 지오그래피 오브젝트 SVG path 데이터를 반환합니다. (Availability: 1.2.2. Availability: 1.4.0 http://www.w3.org/TR/SVG/paths.html#PathDataBNF 에 준수하기 위한 절대적 path 의 L 명령어를 포함하기 위해 PostGIS 1.4.에서 변경되었습니다.

ST_Collect — 다른 지오메트리들의 컬렉션으로부터 특정 ST_Geometry 값을 반환합니다. Availability: 1.4.0 - ST_Collect(geomarray)가 도입되었습니다. 더 많은 지오메트리를 빠르게 처리하기 위해 ST_Collect 가 도입되었습니다.

ST_ContainsProperly — B 가 A 의 경계(혹은 외부)가 아닌 내부와 교차할 때 true 를 반환합니다. A 는 적절하게 스스로를 포함하지 않지만 스스로를 포함하긴 합니다. Availability: 1.4.0 – 은 GEOS >= 3.1.0 를 요구합니다.

ST_Extent — 지오메트리들의 열들을 제한하는 바운딩 박스를 반환하는 집계 함수. Availability: 1.4.0

ST_GeoHash — 지오메트리의 GeoHash 표현(geohash.org)을 반환합니다. Availability: 1.4.0

ST_IsValidReason — 지오메트리가 유효한지 아닌지에 대해 나타내고, 유효하지 않은 이유에 대해 설명하는 텍스트를 반환합니다. Availability: 1.4 - 은 GEOS >= 3.1.0 을 요구합니다.

ST_LineCrossingDirection — 2 선 스트링을 고려, 어떤 종류의 crossing behavior 인지 나타내는 -3 과 3 사이의 숫자를 반환합니다. Availability: 1.4

ST_LocateBetweenElevations — 지정된 범위를 교차하는 요소와 파생된 지오메트리 (컬렉션) 값을 반환합니다. 오직 3D, 4D 의 선 스트링 과 다중 선 스트링이 지원됩니다. Availability: 1.4.0

ST_MakeLine – 포인트 또는 선 지오메트리로부터 선스트링을 생성합니다. Availability: 1.4.0 - ST_MakeLine(geomarray)가 도입되었습니다. 더 많은 포인트를 빠르게 처리하기 위해 ST_MakeLine 집합함수가 강화되었습니다.

ST_MinimumBoundingCircle — 지오메트리를 완전하게 포함할 수 있는 가장 작은 원형 다각형을 반환합니다. 디폴트는 부채꼴 마다 48 세그먼트를 사용합니다. Availability: 1.4.0 – 은 GEOS 을 요구합니다.

ST_Union — 지오메트리의 포인트 집합 연합을 나타내는 지오메트리를 반환합니다. Availability: 1.4.0 - ST_Union 가 강화되었습니다. ST_Union(geomarray) 와 더 빠른 집계 컬렉션이 PostgreSQL 에 도입되었습니다. 만약 GEOS 3.1.0+ ST_Union 을 사용 중이시라면 http://blog.cleverelephant.ca/2009/01/must-faster-unions-in-postgis-14.html 에 설명된 더 빠른 Cascaded 연합 알고리즘을 사용하게 될 것입니다.

13.11.5. PostGIS 1.3 에서의 새로운 함수

아래의 함수들은 1.3. 릴리즈에서 도입된 PostGIS 함수들 입니다.

ST_AsGML — 지오메트리를 GML(Geography Markup Language) 버전 2(2.1.2, 기본값) 또는 버전 3(3.1.1) 포맷으로 반환합니다. Availability: 1.3.2

ST_AsGeoJSON — 지오메트리를 GeoJSON(Geometry Javascript Object Notation) 포맷으로 반환합니다. Availability: 1.3.4

ST_SimplifyPreserveTopology — 주어진 지오메트리를 Douglas-Peucker 알고리즘을 사용하여 "단순화"한 지오메트리를 반환합니다. 이 함수는 유효하지 않는 파생된 지오메트리(특히 Polygon 지오메트리 단순화에서 발생할 수 있는)를 생성하지 않도록 합니다. Availability: 1.3.3

Chapter 14. 문제 보고

14.1. 소프트웨어 버그 보고

버그를 효율적으로 보고하는 것은 PostGIS 개발을 도울 수 있는 근본적인 방법입니다. PostGIS 개발자들이 이것을 재생산할 수 있도록 하는 가장 효율적인 버그 보고 방식은 문제가 되는 스크립트와 버그가 발견되었을 때의 환경에 대한 모든 정보를 포함하는 것입니다.

`SELECT postgis_full_version()` [for postgis] and `SELECT version() [for postgresql]` 을 실행함으로써 좋은 정보가 추출될 수 있습니다.

최신 버전을 사용하고 있지 않다면 이미 고쳐진 버그가 확인하기 위해 release changelog 를 먼저 살펴보는것도 필요합니다.

PostGIS bug tracker 의 사용으로 귀하의 버그 보고서가 삭제되지 않으며, 처리 과정에 대한 정보가 지속적으로 유지될 수 있습니다. 새로운 버그를 보고하기 전에 이미 알려진 버그인지를 확인하기 위해 데이터베이스를 쿼리하시고 새로운 정보이면 추가해주십시오.

새로운 리포트를 제출하기 앞서 Simon Tatham's 의 How to Report Bugs Effectively 를 살펴보십시오.

14.2. 문서 오류 보고

문서는 정확하게 소프트웨어의 기능 및 동작을 반영해야 합니다. 그렇지 않은 경우에는 소프트웨어 버그 또는 문서의 오류나 결함이 있을 수 있습니다.

PostGIS bug tracker 에도 문서 오류에 대해 보고할 수 있습니다.

시험판을 사용 중이시라면 새로운 Bug Tracker 이슈에 문서의 위치를 구체적으로 설명하십시오.

변경 사항이 광범위한 경우에는, Subversion 에 패치하는 것이 확실히 선호될 수 있습니다. Unix 체계에서는 4 단계 과정이 있습니다. (Subversion 이 이미 설치되었다는 가정하에) :

1. PostGIS' Subversion 의 trunk 를 Check out 합니다.
   svn checkout http://svn.osgeo.org/postgis/trunk/
   ./trunk 디렉토리에 저장될 것입니다.

2. 선호하는 텍스트 에디터로 문서를 변경합니다. (Unix 형식의 예):
   vim trunk/doc/postgis.xml
   문서는 HTML 이 아닌 DocBook XML 에서 작성되었다는 점에 유의하십시오. 만약 이에 익숙하지 않은 경우 문서의 나머지 부분의 예시를 따라 하십시오.

3. 문서의 원본과 차이를 포함한 패치 파일을 만드십시오. (Unix 형식의 예):
   svn diff trunk/doc/postgis.xml > doc.patch

4. Bug Tracker 의 새로운 이슈에 패치를 첨부하십시오.

Appendix A Appendix

A.1 Release 2.0.4

Release date: 2013/09/06

This is a bug fix release, addressing issues that have been filed since the 2.0.3 release. If you are using PostGIS 2.0+ a soft upgrade is required. For users of PostGIS 1.5 or below, a hard upgrade is required.

A.1.1 Bug Fixes

#2110, Equality operator between EMPTY and point on origin

Allow adding points at precision distance with TopoGeo_addPoint

#1968, Fix missing edge from toTopoGeom return

#2165, ST_NumPoints regression failure with CircularString

#2168, ST_Distance is not always commutative

#2186, gui progress bar updates too frequent

#2201, ST_GeoHash wrong on boundaries

#2257, GBOX variables not initialized when testing with empty geometries

#2271, Prevent parallel make of raster

#2267, Server crash from analyze table

#2277, potential segfault removed

#2307, ST_MakeValid outputs invalid geometries

#2351, st_distance between geographies wrong

#2359, Incorrect handling of schema for overview constraints

#2371, Support GEOS versions with more than 1 digit in micro

#2372, Cannot parse space-padded KML coordinates

Fix build with systemwide liblwgeom installed

#2383, Fix unsafe use of \' in warning message

#2410, Fix segmentize of collinear curve

#2412, ST_LineToCurve support for lines with less than 4 vertices

#2415, ST_Multi support for COMPOUNDCURVE and CURVEPOLYGON

#2420, ST_LineToCurve: require at least 8 edges to define a full circle

#2423, ST_LineToCurve: require all arc edges to form the same angle

#2424, ST_CurveToLine: add support for COMPOUNDCURVE in MULTICURVE

#2427, Make sure to retain first point of curves on ST_CurveToLine

A.1.2 Enhancements

#2269, Avoid uselessly detoasting full geometries on ANALYZE

A.1.3 Known Issues

#2111, Raster bands can only reference the first 256 bands of out-db rasters

A.2 Release 2.0.3

Release date: 2013/03/01

This is a bug fix release, addressing issues that have been filed since the 2.0.2 release. If you are using PostGIS 2.0+ a soft upgrade is required. For users of PostGIS 1.5 or below, a hard upgrade is required.

A.2.1 Bug Fixes

#2126, Better handling of empty rasters from ST_ConvexHull()

#2134, Make sure to process SRS before passing it off to GDAL functions

Fix various memory leaks in liblwgeom

#2173, Fix robustness issue in splitting a line with own vertex also affecting topology building (#2172)

#2174, Fix usage of wrong function lwpoly_free()

#2176, Fix robustness issue with ST_ChangeEdgeGeom

#2184, Properly copy topologies with Z value

postgis_restore.pl support for mixed case geometry column name in dumps

#2188, Fix function parameter value overflow that caused problems when copying data from a GDAL dataset

#2216, More memory errors in MultiPolygon GeoJSON parsing (with holes)

Fix Memory leak in GeoJSON parser

A.2.2 Enhancements

#2141, More verbose output when constraints fail to be added to a raster column

Speedup ST_ChangeEdgeGeom

A.3 Release 2.0.2

Release date: 2012/12/03

This is a bug fix release, addressing issues that have been filed since the 2.0.1 release. As soft update is required for PostGIS 2.0.0 and 2.0.1 users.

A.3.1 Bug Fixes

#1287, Drop of "gist_geometry_ops" broke a few clients package of legacy_gist.sql for these cases

#1391, Errors during upgrade from 1.5

#1828, Poor selectivity estimate on ST_DWithin

#1838, error importing tiger/line data

#1869, ST_AsBinary is not unique added to legacy_minor/legacy.sql scripts

#1885, Missing field from tabblock table in tiger2010 census_loader.sql

#1891, Use LDFLAGS environment when building liblwgeom

#1900, Fix pgsql2shp for big-endian systems

#1932, Fix raster2pgsql for invalid syntax for setting index tablespace

#1936, ST_GeomFromGML on CurvePolygon causes server crash

#1955, ST_ModEdgeHeal and ST_NewEdgeHeal for doubly connected edges

#1957, ST_Distance to a one-point LineString returns NULL

#1976, Geography point-in-ring code overhauled for more reliability

#1978, wrong answer calculating length of closed circular arc (circle)

#1981, Remove unused but set variables as found with gcc 4.6+

#1987, Restore 1.5.x behaviour of ST_Simplify

#1989, Preprocess input geometry to just intersection with raster to be clipped

#1991, geocode really slow on PostgreSQL 9.2

#1996, support POINT EMPTY in GeoJSON output

#1998, Fix ST_{Mod,New}EdgeHeal joining edges sharing both endpoints

#2001, ST_CurveToLine has no effect if the geometry doesn't actually contain an arc

#2015, ST_IsEmpty('POLYGON(EMPTY)') returns False

#2019, ST_FlipCoordinates does not update bbox

#2025, Fix side location conflict at TopoGeo_AddLineString

#2026, improve performance of distance calculations

#2033, Fix adding a splitting point into a 2.5d topology

#2051, Fix excess of precision in ST_AsGeoJSON output

#2052, Fix buffer overflow in lwgeom_to_geojson

#2056, Fixed lack of SRID check of raster and geometry in ST_SetValue()

#2057, Fixed linking issue for raster2psql to libpq

#2060, Fix "dimension" check violation by GetTopoGeomElementArray

#2072, Removed outdated checks preventing ST_Intersects(raster) from working on out-db bands

#2077, Fixed incorrect answers from ST_Hillshade(raster)

#2092, Namespace issue with ST_GeomFromKML,ST_GeomFromGML for libxml 2.8+

#2099, Fix double free on exception in ST_OffsetCurve

#2100, ST_AsRaster() may not return raster with specified pixel type

#2108, Ensure ST_Line_Interpolate_Point always returns POINT

#2109, Ensure ST_Centroid always returns POINT

#2117, Ensure ST_PointOnSurface always returns POINT

#2129, Fix SRID in ST_Homogenize output with collection input

#2130, Fix memory error in MultiPolygon GeoJson parsing

Update URL of Maven jar

A.3.2 Enhancements

#1581, ST_Clip(raster, ...) no longer imposes NODATA on a band if the corresponding band from the source raster did not have NODATA

#1928, Accept array properties in GML input multi-geom input (Kashif Rasul and Shoaib Burq / SpacialDB)

#2082, Add indices on start_node and end_node of topology edge tables

#2087, Speedup topology.GetRingEdges using a recursive CTE

A.4 Release 2.0.1

Release date: 2012/06/22

This is a bug fix release, addressing issues that have been filed since the 2.0.0 release.

A.4.1 Bug Fixes

#1264, fix st_dwithin(geog, geog, 0).

#1468 shp2pgsql-gui table column schema get shifted

#1694, fix building with clang. (vince)

#1708, improve restore of pre-PostGIS 2.0 backups.

#1714, more robust handling of high topology tolerance.

#1755, ST_GeographyFromText support for higher dimensions.

#1759, loading transformed shapefiles in raster enabled db.

#1761, handling of subdatasets in NetCDF, HDF4 and HDF5 in raster2pgsql.

#1763, topology.toTopoGeom use with custom search_path.

#1766, don't let ST_RemEdge* destroy peripheral TopoGeometry objects.

#1774, Clearer error on setting an edge geometry to an invalid one.

#1775, ST_ChangeEdgeGeom collision detection with 2-vertex target.

#1776, fix ST_SymDifference(empty, geom) to return geom.

#1779, install SQL comment files.

#1782, fix spatial reference string handling in raster.

#1789, fix false edge-node crossing report in ValidateTopology.

#1790, fix toTopoGeom handling of duplicated primitives.

#1791, fix ST_Azimuth with very close but distinct points.

#1797, fix (ValidateTopology(xxx)).* syntax calls.

#1805, put back the 900913 SRID entry.

#1813, Only show readable relations in metadata tables.

#1819, fix floating point issues with ST_World2RasterCoord and ST_Raster2WorldCoord variants.

#1820 compilation on 9.2beta1.

#1822, topology load on PostgreSQL 9.2beta1.

#1825, fix prepared geometry cache lookup

#1829, fix uninitialized read in GeoJSON parser

#1834, revise postgis extension to only backup user specified spatial_ref_sys

#1839, handling of subdatasets in GeoTIFF in raster2pgsql.

#1840, fix logic of when to compute # of tiles in raster2pgsql.

#1851, fix spatial_ref_system parameters for EPSG:3844

#1857, fix failure to detect endpoint mismatch in ST_AddEdge*Face*

#1865, data loss in postgis_restore.pl when data rows have leading dashes.

#1867, catch invalid topology name passed to topogeo_add*

#1872, fix ST_ApproxSummarystats to prevent division by zero

#1873, fix ptarray_locate_point to return interpolated Z/M values for on-the-line case

#1875, ST_SummaryStats returns NULL for all parameters except count when count is zero

#1881, shp2pgsql-gui -- editing a field sometimes triggers removing row

#1883, Geocoder install fails trying to run create_census_base_tables() (Brian Panulla)

#1870, ST_Intersects doc and behaviour are out of synch

A.4.2 Enhancements

More detailed exception message from topology editing functions.

#1786, improved build dependencies

#1806, speedup of ST_BuildArea, ST_MakeValid and ST_GetFaceGeometry.

#1812, Add lwgeom_normalize in LIBLWGEOM for more stable testing.

A.5 Release 2.0.0

Release date: 2012/04/03

This is a major release. A hard upgrade is required. Yes this means a full dump reload and some special preparations if you are using obsolete functions. Refer to Section 2.9.2, “Hard upgrade” for details on upgrading. Refer to Section 13.11.1, “PostGIS Functions new, behavior changed, or enhanced in 2.0” for more details and changed/new functions.

A.5.1 Testers - Our unsung heroes

We are most indebted to the numerous members in the PostGIS community who were brave enough to test out the new features in this release. No major release can be successful without these folk.

Below are those who have been most valiant, provided very detailed and thorough bug reports, and detailed analysis.

Andrea Peri - Lots of testing on topology, checking for correctness

Andreas Forø Tollefsen - raster testing

Chris English - topology stress testing loader functions

Salvatore Larosa - topology robustness testing

Brian Hamlin - Benchmarking (also experimental experimental branches before they are folded into core) , general testing of various pieces including Tiger and Topology. Testing on various server VMs

Mike Pease - Tiger geocoder testing - very detailed reports of issues

Tom van Tilburg - raster testing

A.5.2 Important / Breaking Changes

#722, #302, Most deprecated functions removed (over 250 functions) (Regina Obe, Paul Ramsey)

Unknown SRID changed from -1 to 0. (Paul Ramsey)

-- (most deprecated in 1.2) removed non-ST variants buffer, length, intersects (and internal functions renamed) etc.

-- If you have been using deprecated functions CHANGE your apps or suffer the consequences. If you don't see a function documented -- it ain't supported or it is an internal function. Some constraints in older tables were built with deprecated functions. If you restore you may need to rebuild table constraints with populate_geometry_columns(). If you have applications or tools that rely on deprecated functions, please refer to ??? for more details.

#944 geometry_columns is now a view instead of a table (Paul Ramsey, Regina Obe) for tables created the old way reads (srid, type, dims) constraints for geometry columns created with type modifiers reads rom column definition

#1081, #1082, #1084, #1088 - Mangement functions support typmod geometry column creation functions now default to typmod creation (Regina Obe)

#1083 probe_geometry_columns(), rename_geometry_table_constraints(), fix_geometry_columns(); removed - now obsolete with geometry_column view (Regina Obe)

#817 Renaming old 3D functions to the convention ST_3D (Nicklas Avén)

#548 (sorta), ST_NumGeometries,ST_GeometryN now returns 1 (or the geometry) instead of null for single geometries (Sandro Santilli, Maxime van Noppen)

A.5.3 New Features

KNN Gist index based centroid (<->) and box (<#>) distance operators (Paul Ramsey / funded by Vizzuality)

Support for TIN and PolyHedralSurface and enhancement of many functions to support 3D (Olivier Courtin / Oslandia)

Raster support integrated and documented (Pierre Racine, Jorge Arévalo, Mateusz Loskot, Sandro Santilli, David Zwarg, Regina Obe, Bborie Park) (Company developer and funding: University Laval, Deimos Space, CadCorp, Michigan Tech Research Institute, Azavea, Paragon Corporation, UC Davis Center for Vectorborne Diseases)

Making spatial indexes 3D aware - in progress (Paul Ramsey, Mark Cave-Ayland)

Topology support improved (more functions), documented, testing (Sandro Santilli / Faunalia for RT-SIGTA), Andrea Peri, Regina Obe, Jose Carlos Martinez Llari

3D relationship and measurement support functions (Nicklas Avén)

ST_3DDistance, ST_3DClosestPoint, ST_3DIntersects, ST_3DShortestLine and more...

N-Dimensional spatial indexes (Paul Ramsey / OpenGeo)

ST_Split (Sandro Santilli / Faunalia for RT-SIGTA)

ST_IsValidDetail (Sandro Santilli / Faunalia for RT-SIGTA)

ST_MakeValid (Sandro Santilli / Faunalia for RT-SIGTA)

ST_RemoveRepeatedPoints (Sandro Santilli / Faunalia for RT-SIGTA)

ST_GeometryN and ST_NumGeometries support for non-collections (Sandro Santilli)

ST_IsCollection (Sandro Santilli, Maxime van Noppen)

ST_SharedPaths (Sandro Santilli / Faunalia for RT-SIGTA)

ST_Snap (Sandro Santilli)

ST_RelateMatch (Sandro Santilli / Faunalia for RT-SIGTA)

ST_ConcaveHull (Regina Obe and Leo Hsu / Paragon Corporation)

ST_UnaryUnion (Sandro Santilli / Faunalia for RT-SIGTA)

ST_AsX3D (Regina Obe / Arrival 3D funding)

ST_OffsetCurve (Sandro Santilli, Rafal Magda)

ST_GeomFromGeoJSON (Kashif Rasul, Paul Ramsey / Vizzuality funding)

A.5.4 Enhancements

Made shape file loader tolerant of truncated multibyte values found in some free worldwide shapefiles (Sandro Santilli)

Lots of bug fixes and enhancements to shp2pgsql Beefing up regression tests for loaders Reproject support for both geometry and geography during import (Jeff Adams / Azavea, Mark Cave-Ayland)

pgsql2shp conversion from predefined list (Loic Dachary / Mark Cave-Ayland)

Shp-pgsql GUI loader - support loading multiple files at a time. (Mark Leslie)

Extras - upgraded tiger_geocoder from using old TIGER format to use new TIGER shp and file structure format (Stephen Frost)

Extras - revised tiger_geocoder to work with TIGER census 2010 data, addition of reverse geocoder function, various bug fixes, accuracy enhancements, limit max result return, speed improvements, loading routines. (Regina Obe, Leo Hsu / Paragon Corporation / funding provided by Hunter Systems Group)

Overall Documentation proofreading and corrections. (Kasif Rasul)

Cleanup PostGIS JDBC classes, revise to use Maven build. (Maria Arias de Reyna, Sandro Santilli)

A.5.5 Bug Fixes

#1335 ST_AddPoint returns incorrect result on Linux (Even Rouault)

A.5.6 Release specific credits

We thank U.S Department of State Human Information Unit (HIU) and Vizzuality for general monetary support to get PostGIS 2.0 out the door.

A.6 Release 1.5.4

Release date: 2012/05/07

This is a bug fix release, addressing issues that have been filed since the 1.5.3 release.

A.6.1 Bug Fixes

#547, ST_Contains memory problems (Sandro Santilli)

#621, Problem finding intersections with geography (Paul Ramsey)

#627, PostGIS/PostgreSQL process die on invalid geometry (Paul Ramsey)

#810, Increase accuracy of area calculation (Paul Ramsey)

#852, improve spatial predicates robustness (Sandro Santilli, Nicklas Avén)

#877, ST_Estimated_Extent returns NULL on empty tables (Sandro Santilli)

#1028, ST_AsSVG kills whole postgres server when fails (Paul Ramsey)

#1056, Fix boxes of arcs and circle stroking code (Paul Ramsey)

#1121, populate_geometry_columns using deprecated functions (Regin Obe, Paul Ramsey)

#1135, improve testsuite predictability (Andreas 'ads' Scherbaum)

#1146, images generator crashes (bronaugh)

#1170, North Pole intersection fails (Paul Ramsey)

#1179, ST_AsText crash with bad value (kjurka)

#1184, honour DESTDIR in documentation Makefile (Bryce L Nordgren)

#1227, server crash on invalid GML

#1252, SRID appearing in WKT (Paul Ramsey)

#1264, st_dwithin(g, g, 0) doesn't work (Paul Ramsey)

#1344, allow exporting tables with invalid geometries (Sandro Santilli)

#1389, wrong proj4text for SRID 31300 and 31370 (Paul Ramsey)

#1406, shp2pgsql crashes when loading into geography (Sandro Santilli)

#1595, fixed SRID redundancy in ST_Line_SubString (Sandro Santilli)

#1596, check SRID in UpdateGeometrySRID (Mike Toews, Sandro Santilli)

#1602, fix ST_Polygonize to retain Z (Sandro Santilli)

#1697, fix crash with EMPTY entries in GiST index (Paul Ramsey)

#1772, fix ST_Line_Locate_Point with collapsed input (Sandro Santilli)

#1799, Protect ST_Segmentize from max_length=0 (Sandro Santilli)

Alter parameter order in 900913 (Paul Ramsey)

Support builds with "gmake" (Greg Troxel)

A.7 Release 1.5.3

Release date: 2011/06/25

This is a bug fix release, addressing issues that have been filed since the 1.5.2 release. If you are running PostGIS 1.3+, a soft upgrade is sufficient otherwise a hard upgrade is recommended.

A.7.1 Bug Fixes

#1056, produce correct bboxes for arc geometries, fixes index errors (Paul Ramsey)

#1007, ST_IsValid crash fix requires GEOS 3.3.0+ or 3.2.3+ (Sandro Santilli, reported by Birgit Laggner)

#940, support for PostgreSQL 9.1 beta 1 (Regina Obe, Paul Ramsey, patch submitted by stl)

#845, ST_Intersects precision error (Sandro Santilli, Nicklas Avén) Reported by cdestigter

#884, Unstable results with ST_Within, ST_Intersects (Chris Hodgson)

#779, shp2pgsql -S option seems to fail on points (Jeff Adams)

#666, ST_DumpPoints is not null safe (Regina Obe)

#631, Update NZ projections for grid transformation support (jpalmer)

#630, Peculiar Null treatment in arrays in ST_Collect (Chris Hodgson) Reported by David Bitner

#624, Memory leak in ST_GeogFromText (ryang, Paul Ramsey)

#609, Bad source code in manual section 5.2 Java Clients (simoc, Regina Obe)

#604, shp2pgsql usage touchups (Mike Toews, Paul Ramsey)

#573 ST_Union fails on a group of linestrings Not a PostGIS bug, fixed in GEOS 3.3.0

#457 ST_CollectionExtract returns non-requested type (Nicklas Avén, Paul Ramsey)

#441 ST_AsGeoJson Bbox on GeometryCollection error (Olivier Courtin)

#411 Ability to backup invalid geometries (Sando Santilli) Reported by Regione Toscana

#409 ST_AsSVG - degraded (Olivier Courtin) Reported by Sdikiy

#373 Documentation syntax error in hard upgrade (Paul Ramsey) Reported by psvensso

A.8 Release 1.5.2

Release date: 2010/09/27

This is a bug fix release, addressing issues that have been filed since the 1.5.1 release. If you are running PostGIS 1.3+, a soft upgrade is sufficient otherwise a hard upgrade is recommended.

A.8.1 Bug Fixes

Loader: fix handling of empty (0-verticed) geometries in shapefiles. (Sandro Santilli)

#536, Geography ST_Intersects, ST_Covers, ST_CoveredBy and Geometry ST_Equals not using spatial index (Regina Obe, Nicklas Aven)

#573, Improvement to ST_Contains geography (Paul Ramsey)

Loader: Add support for command-q shutdown in Mac GTK build (Paul Ramsey)

#393, Loader: Add temporary patch for large DBF files (Maxime Guillaud, Paul Ramsey)

#507, Fix wrong OGC URN in GeoJSON and GML output (Olivier Courtin)

spatial_ref_sys.sql Add datum conversion for projection SRID 3021 (Paul Ramsey)

Geography - remove crash for case when all geographies are out of the estimate (Paul Ramsey)

#469, Fix for array_aggregation error (Greg Stark, Paul Ramsey)

#532, Temporary geography tables showing up in other user sessions (Paul Ramsey)

#562, ST_Dwithin errors for large geographies (Paul Ramsey)

#513, shape loading GUI tries to make spatial index when loading DBF only mode (Paul Ramsey)

#527, shape loading GUI should always append log messages (Mark Cave-Ayland)

#504, shp2pgsql should rename xmin/xmax fields (Sandro Santilli)

#458, postgis_comments being installed in contrib instead of version folder (Mark Cave-Ayland)

#474, Analyzing a table with geography column crashes server (Paul Ramsey)

#581, LWGEOM-expand produces inconsistent results (Mark Cave-Ayland)

#513, Add dbf filter to shp2pgsql-gui and allow uploading dbf only (Paul Ramsey)

Fix further build issues against PostgreSQL 9.0 (Mark Cave-Ayland)

#572, Password whitespace for Shape File (Mark Cave-Ayland)

#603, shp2pgsql: "-w" produces invalid WKT for MULTI* objects. (Mark Cave-Ayland)

A.9 Release 1.5.1

Release date: 2010/03/11

This is a bug fix release, addressing issues that have been filed since the 1.4.1 release. If you are running PostGIS 1.3+, a soft upgrade is sufficient otherwise a hard upgrade is recommended.

A.9.1 Bug Fixes

#410, update embedded bbox when applying ST_SetPoint, ST_AddPoint ST_RemovePoint to a linestring (Paul Ramsey)

#411, allow dumping tables with invalid geometries (Sandro Santilli, for Regione Toscana-SIGTA)

#414, include geography_columns view when running upgrade scripts (Paul Ramsey)

#419, allow support for multilinestring in ST_Line_Substring (Paul Ramsey, for Lidwala Consulting Engineers)

#421, fix computed string length in ST_AsGML() (Olivier Courtin)

#441, fix GML generation with heterogeneous collections (Olivier Courtin)

#443, incorrect coordinate reversal in GML 3 generation (Olivier Courtin)

#450, #451, wrong area calculation for geography features that cross the date line (Paul Ramsey)

Ensure support for upcoming 9.0 PgSQL release (Paul Ramsey)

A.10 Release 1.5.0

Release date: 2010/02/04

This release provides support for geographic coordinates (lat/lon) via a new GEOGRAPHY type. Also performance enhancements, new input format support (GML,KML) and general upkeep.

A.10.1 API Stability

The public API of PostGIS will not change during minor (0.0.X) releases.

The definition of the =~ operator has changed from an exact geometric equality check to a bounding box equality check.

A.10.2 Compatibility

GEOS, Proj4, and LibXML2 are now mandatory dependencies

The library versions below are the minimum requirements for PostGIS 1.5

PostgreSQL 8.3 and higher on all platforms

GEOS 3.1 and higher only (GEOS 3.2+ to take advantage of all features)

LibXML2 2.5+ related to new ST_GeomFromGML/KML functionality

Proj4 4.5 and higher only

A.10.3 New Features

Section 13.11.3, "PostGIS Functions new, behavior changed, or enhanced in 1.5"

Added Hausdorff distance calculations (#209) (Vincent Picavet)

Added parameters argument to ST_Buffer operation to support one-sided buffering and other buffering styles (Sandro Santilli)

Addition of other Distance related visualization and analysis functions (Nicklas Aven)

- ST_ClosestPoint
- ST_DFullyWithin
- ST_LongestLine
- ST_MaxDistance
- ST_ShortestLine

ST_DumpPoints (Maxime van Noppen)

KML, GML input via ST_GeomFromGML and ST_GeomFromKML (Olivier Courtin)

Extract homogeneous collection with ST_CollectionExtract (Paul Ramsey)

Add measure values to an existing linestring with ST_AddMeasure (Paul Ramsey)

History table implementation in utils (George Silva)

Geography type and supporting functions

- Spherical algorithms (Dave Skea)
- Object/index implementation (Paul Ramsey)
- Selectivity implementation (Mark Cave-Ayland)
- Serializations to KML, GML and JSON (Olivier Courtin)

- ST_Area, ST_Distance, ST_DWithin, ST_GeogFromText, ST_GeogFromWKB, ST_Intersects, ST_Covers, ST_Buffer (Paul Ramsey)

A.10.4 Enhancements

Performance improvements to ST_Distance (Nicklas Aven)

Documentation updates and improvements (Regina Obe, Kevin Neufeld)

Testing and quality control (Regina Obe)

PostGIS 1.5 support PostgreSQL 8.5 trunk (Guillaume Lelarge)

Win32 support and improvement of core shp2pgsql-gui (Mark Cave-Ayland)

In place 'make check' support (Paul Ramsey)

A.10.5 Bug fixes

http://trac.osgeo.org/postgis/query?status=closed&milestone=PostGIS+1.5.0&order=priority

A.11 Release 1.4.0

Release date: 2009/07/24

This release provides performance enhancements, improved internal structures and testing, new features, and upgraded documentation. If you are running PostGIS 1.1+, a soft upgrade is sufficient otherwise a hard upgrade is recommended.

A.11.1 API Stability

As of the 1.4 release series, the public API of PostGIS will not change during minor releases.

A.11.2 Compatibility

The versions below are the *minimum* requirements for PostGIS 1.4

PostgreSQL 8.2 and higher on all platforms

GEOS 3.0 and higher only

PROJ4 4.5 and higher only

A.11.3 New Features

ST_Union() uses high-speed cascaded union when compiled against GEOS 3.1+ (Paul Ramsey)

ST_ContainsProperly() requires GEOS 3.1+

ST_Intersects(), ST_Contains(), ST_Within() use high-speed cached prepared geometry against GEOS 3.1+ (Paul Ramsey / funded by Zonar Systems)

Vastly improved documentation and reference manual (Regina Obe & Kevin Neufeld)

Figures and diagram examples in the reference manual (Kevin Neufeld)

ST_IsValidReason() returns readable explanations for validity failures (Paul Ramsey)

ST_GeoHash() returns a geohash.org signature for geometries (Paul Ramsey)

GTK+ multi-platform GUI for shape file loading (Paul Ramsey)

ST_LineCrossingDirection() returns crossing directions (Paul Ramsey)

ST_LocateBetweenElevations() returns sub-string based on Z-ordinate. (Paul Ramsey)

Geometry parser returns explicit error message about location of syntax errors (Mark Cave-Ayland)

ST_AsGeoJSON() return JSON formatted geometry (Olivier Courtin)

Populate_Geometry_Columns() -- automatically add records to geometry_columns for TABLES and VIEWS (Kevin Neufeld)

ST_MinimumBoundingCircle() -- returns the smallest circle polygon that can encompass a geometry (Bruce Rindahl)

A.11.4 Enhancements

Core geometry system moved into independent library, liblwgeom. (Mark Cave-Ayland)

New build system uses PostgreSQL "pgxs" build bootstrapper. (Mark Cave-Ayland)

Debugging framework formalized and simplified. (Mark Cave-Ayland)

All build-time #defines generated at configure time and placed in headers for easier cross-platform support (Mark Cave-Ayland)

Logging framework formalized and simplified (Mark Cave-Ayland)

Expanded and more stable support for CIRCULARSTRING, COMPOUNDCURVE and CURVEPOLYGON, better parsing, wider support in functions (Mark Leslie & Mark Cave-Ayland)

Improved support for OpenSolaris builds (Paul Ramsey)

Improved support for MSVC builds (Mateusz Loskot)

Updated KML support (Olivier Courtin)

Unit testing framework for liblwgeom (Paul Ramsey)

New testing framework to comprehensively exercise every PostGIS function (Regine Obe)

Performance improvements to all geometry aggregate functions (Paul Ramsey)

Support for the upcoming PostgreSQL 8.4 (Mark Cave-Ayland, Talha Bin Rizwan)

Shp2pgsql and pgsql2shp re-worked to depend on the common parsing/unparsing code in liblwgeom (Mark Cave-Ayland)

Use of PDF DbLatex to build PDF docs and preliminary instructions for build (Jean David Techer)

Automated User documentation build (PDF and HTML) and Developer Doxygen Documentation (Kevin Neufeld)

Automated build of document images using ImageMagick from WKT geometry text files (Kevin Neufeld)

More attractive CSS for HTML documentation (Dane Springmeyer)

A.11.5 Bug fixes

http://trac.osgeo.org/postgis/query?status=closed&milestone=PostGIS+1.4.0&order=priority

A.12 Release 1.3.6

Release date: 2009/05/04

If you are running PostGIS 1.1+, a soft upgrade is sufficient otherwise a hard upgrade is recommended. This release adds support for PostgreSQL 8.4, exporting prj files from the database with shape data, some crash fixes for shp2pgsql, and several small bug fixes in the handling of "curve" types, logical error importing dbf only files, improved error handling of AddGeometryColumns.

A.13 Release 1.3.5

Release date: 2008/12/15

If you are running PostGIS 1.1+, a soft upgrade is sufficient otherwise a hard upgrade is recommended. This release is a bug fix release to address a failure in ST_Force_Collection and related functions that critically affects using MapServer with LINE layers.

A.14 Release 1.3.4

Release date: 2008/11/24

This release adds support for GeoJSON output, building with PostgreSQL 8.4, improves documentation quality and output aesthetics, adds function-level SQL documentation, and improves performance for some spatial predicates (point-in-polygon tests).

Bug fixes include removal of crashers in handling circular strings for many functions, some memory leaks removed, a linear referencing failure for measures on vertices, and more. See the NEWS file for details.

A.15 Release 1.3.3

Release date: 2008/04/12

This release fixes bugs shp2pgsql, adds enhancements to SVG and KML support, adds a ST_SimplifyPreserveTopology function, makes the build more sensitive to GEOS versions, and fixes a handful of severe but rare failure cases.

A.16 Release 1.3.2

Release date: 2007/12/01

This release fixes bugs in ST_EndPoint() and ST_Envelope, improves support for JDBC building and OS/X, and adds better support for GML output with ST_AsGML(), including GML3 output.

A.17 Release 1.3.1

Release date: 2007/08/13

This release fixes some oversights in the previous release around version numbering, documentation, and tagging.

A.18 Release 1.3.0

Release date: 2007/08/09

This release provides performance enhancements to the relational functions, adds new relational functions and begins the migration of our function names to the SQL-MM convention, using the spatial type (SP) prefix.

A.18.1 Added Functionality

. JDBC: Added Hibernate Dialect (thanks to Norman Barker)

Added ST_Covers and ST_CoveredBy relational functions. Description and justification of these functions can be found at http://lin-ear-th-inking.blogspot.com/2007/06/subtleties-of-ogc-covers-spatial.html

Added ST_DWithin relational function

A.18.2 Performance Enhancements

Added cached and indexed point-in-polygon short-circuits for the functions ST_Contains, ST_Intersects, ST_Within and ST_Disjoint

Added inline index support for relational functions (except ST_Disjoint)

A.18.3 Other Changes

Extended curved geometry support into the geometry accessor and some processing functions

Began migration of functions to the SQL-MM naming convention; using a spatial type (ST) prefix.

Added initial support for PostgreSQL 8.3

A.19 Release 1.2.1

Release date: 2007/01/11

This release provides bug fixes in PostgreSQL 8.2 support and some small performance enhancements.

A.19.1 Changes

Fixed point-in-polygon shortcut bug in Within().

Fixed PostgreSQL 8.2 NULL handling for indexes.

Updated RPM spec files.

Added short-circuit for Transform() in no-op case.

JDBC: Fixed JTS handling for multi-dimensional geometries (thanks to Thomas Marti for hint and partial patch). Additionally, now JavaDoc is compiled and packaged. Fixed classpath problems with GCJ. Fixed pgjdbc 8.2 compatibility, losing support for jdk 1.3 and older.

A.20 Release 1.2.0

Release date: 2006/12/08

This release provides type definitions along with serialization/deserialization capabilities for SQL-MM defined curved geometries, as well as performance enhancements.

A.20.1 Changes

Added curved geometry type support for serialization/deserialization

Added point-in-polygon shortcircuit to the Contains and Within functions to improve performance for these cases.

A.21 Release 1.1.6

Release date: 2006/11/02

This is a bugfix release, in particular fixing a critical error with GEOS interface in 64bit systems. Includes an updated of the SRS parameters and an improvement in reprojections (take Z in consideration). Upgrade is encouraged.

A.21.1 Upgrading

If you are upgrading from release 1.0.3 or later follow the soft upgrade procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the upgrade section of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an hard upgrade.

A.21.2 Bug fixes

fixed CAPI change that broke 64-bit platforms

loader/dumper: fixed regression tests and usage output

Fixed setSRID() bug in JDBC, thanks to Thomas Marti

A.21.3 Other changes

use Z ordinate in reprojections

spatial_ref_sys.sql updated to EPSG 6.11.1

Simplified Version.config infrastructure to use a single pack of version variables for everything.

Include the Version.config in loader/dumper USAGE messages

Replace hand-made, fragile JDBC version parser with Properties

A.22 Release 1.1.5

Release date: 2006/10/13

This is an bugfix release, including a critical segfault on win32. Upgrade is encouraged.

A.22.1 Upgrading

If you are upgrading from release 1.0.3 or later follow the soft upgrade procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the upgrade section of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an hard upgrade.

A.22.2 Bug fixes

Fixed MingW link error that was causing pgsql2shp to segfault on Win32 when compiled for PostgreSQL 8.2

fixed nullpointer Exception in Geometry.equals() method in Java

Added EJB3Spatial.odt to fulfill the GPL requirement of distributing the "preferred form of modification"

Removed obsolete synchronization from JDBC Jts code.

Updated heavily outdated README files for shp2pgsql/pgsql2shp by merging them with the manpages.

Fixed version tag in jdbc code that still said "1.1.3" in the "1.1.4" release.

A.22.3 New Features

Added -S option for non-multi geometries to shp2pgsql

A.23 Release 1.1.4

Release date: 2006/09/27

This is an bugfix release including some improvements in the Java interface. Upgrade is *encouraged*.

A.23.1 Upgrading

If you are upgrading from release 1.0.3 or later follow the soft upgrade procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the upgrade section of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an hard upgrade.

A.23.2 Bug fixes

Fixed support for PostgreSQL 8.2

Fixed bug in collect() function discarding SRID of input

Added SRID match check in MakeBox2d and MakeBox3d

Fixed regress tests to pass with GEOS-3.0.0

Improved pgsql2shp run concurrency.

A.23.3 Java changes

reworked JTS support to reflect new upstream JTS developers' attitude to SRID handling. Simplifies code and drops build depend on GNU trove.

Added EJB2 support generously donated by the "Geodetix s.r.l. Company" http://www.geodetix.it/

Added EJB3 tutorial / examples donated by Norman Barker <nbarker@ittvis.com>

Reorganized java directory layout a little.

A.24 Release 1.1.3

Release date: 2006/06/30

This is an bugfix release including also some new functionalities (most notably long transaction support) and portability enhancements. Upgrade is *encouraged*.

A.24.1 Upgrading

If you are upgrading from release 1.0.3 or later follow the soft upgrade procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the upgrade section of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an hard upgrade.

A.24.2 Bug fixes / correctness

BUGFIX in distance(poly,poly) giving wrong results.

BUGFIX in pgsql2shp successful return code.

BUGFIX in shp2pgsql handling of MultiLine WKT.

BUGFIX in affine() failing to update bounding box.

WKT parser: forbidden construction of multigeometries with EMPTY elements (still supported for GEOMETRYCOLLECTION).

A.24.3 New functionalities

NEW Long Transactions support.

NEW DumpRings() function.

NEW AsHEXEWKB(geom, XDR|NDR) function.

A.24.4 JDBC changes

Improved regression tests: MultiPoint and scientific ordinates

Fixed some minor bugs in jdbc code

Added proper accessor functions for all fields in preparation of making those fields private later

A.24.5 Other changes

NEW regress test support for loader/dumper.

Added --with-proj-libdir and --with-geos-libdir configure switches.

Support for build Tru64 build.

Use Jade for generating documentation.

Don't link pgsql2shp to more libs then required.

Initial support for PostgreSQL 8.2.

A.25 Release 1.1.2

Release date: 2006/03/30

This is an bugfix release including some new functions and portability enhancements. Upgrade is *encouraged*.

A.25.1 Upgrading

If you are upgrading from release 1.0.3 or later follow the soft upgrade procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the upgrade section of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an hard upgrade.

A.25.2 Bug fixes

BUGFIX in SnapToGrid() computation of output bounding box

BUGFIX in EnforceRHR()

jdbc2 SRID handling fixes in JTS code

Fixed support for 64bit archs

A.25.3 New functionalities

Regress tests can now be run *before* postgis installation

New affine() matrix transformation functions

New rotate{,X,Y,Z}() function

Old translating and scaling functions now use affine() internally

Embedded access control in estimated_extent() for builds against pgsql >= 8.0.0

A.25.4 Other changes

More portable ./configure script

Changed ./run_test script to have more sane default behavior

A.26 Release 1.1.1

Release date: 2006/01/23

This is an important Bugfix release, upgrade is *highly recommended*. Previous version contained a bug in postgis_restore.pl preventing hard upgrade procedure to complete and a bug in GEOS-2.2+ connector preventing GeometryCollection objects to be used in topological operations.

A.26.1 Upgrading

If you are upgrading from release 1.0.3 or later follow the soft upgrade procedure.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the upgrade section of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an hard upgrade.

A.26.2 Bug fixes

Fixed a premature exit in postgis_restore.pl

BUGFIX in geometrycollection handling of GEOS-CAPI connector

Solaris 2.7 and MingW support improvements

BUGFIX in line_locate_point()

Fixed handling of postgresql paths

BUGFIX in line_substring()

Added support for localized cluster in regress tester

A.26.3 New functionalities

New Z and M interpolation in line_substring()

New Z and M interpolation in line_interpolate_point()

added NumInteriorRing() alias due to OpenGIS ambiguity

A.27 Release 1.1.0

Release date: 2005/12/21

This is a Minor release, containing many improvements and new things. Most notably: build procedure greatly simplified; transform() performance drastically improved; more stable GEOS connectivity (CAPI support); lots of new functions; draft topology support.

It is *highly recommended* that you upgrade to GEOS-2.2.x before installing PostGIS, this will ensure future GEOS upgrades won't require a rebuild of the PostGIS library.

A.27.1 Credits

This release includes code from Mark Cave Ayland for caching of proj4 objects. Markus Schaber added many improvements in his JDBC2 code. Alex Bodnaru helped with PostgreSQL source dependency relief and provided Debian specfiles. Michael Fuhr tested new things on Solaris arch. David Techer and Gerald Fenoy helped testing GEOS C-API connector. Hartmut Tschauner provided code for the azimuth() function. Devrim GUNDUZ provided RPM specfiles. Carl Anderson helped with the new area building functions. See the credits section for more names.

A.27.2 Upgrading

If you are upgrading from release 1.0.3 or later you *DO NOT* need a dump/reload. Simply sourcing the new lwpostgis_upgrade.sql script in all your existing databases will work. See the soft upgrade chapter for more information.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the upgrade section of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an hard upgrade.

A.27.3 New functions

scale() and transscale() companion methods to translate()

line_substring()

line_locate_point()

M(point)

LineMerge(geometry)

shift_longitude(geometry)

azimuth(geometry)

locate_along_measure(geometry, float8)

locate_between_measures(geometry, float8, float8)

SnapToGrid by point offset (up to 4d support)

BuildArea(any_geometry)

OGC BdPolyFromText(linestring_wkt, srid)

OGC BdMPolyFromText(linestring_wkt, srid)

RemovePoint(linestring, offset)

ReplacePoint(linestring, offset, point)

A.27.4 Bug fixes

Fixed memory leak in polygonize()

Fixed bug in lwgeom_as_anytype cast functions

Fixed USE_GEOS, USE_PROJ and USE_STATS elements of postgis_version() output to always reflect library state.

A.27.5 Function semantic changes

SnapToGrid doesn't discard higher dimensions

Changed Z() function to return NULL if requested dimension is not available

A.27.6 Performance improvements

Much faster transform() function, caching proj4 objects

Removed automatic call to fix_geometry_columns() in AddGeometryColumns() and update_geometry_stats()

A.27.7 JDBC2 works

Makefile improvements

JTS support improvements

Improved regression test system

Basic consistency check method for geometry collections

Support for (Hex)(E)wkb

Autoprobing DriverWrapper for HexWKB / EWKT switching

fix compile problems in ValueSetter for ancient jdk releases.

fix EWKT constructors to accept SRID=4711; representation

added preliminary read-only support for java2d geometries

A.27.8 Other new things

Full autoconf-based configuration, with PostgreSQL source dependency relief

GEOS C-API support (2.2.0 and higher)

Initial support for topology modelling

Debian and RPM specfiles

New lwpostgis_upgrade.sql script

A.27.9 Other changes

JTS support improvements

Stricter mapping between DBF and SQL integer and string attributes

Wider and cleaner regression test suite

old jdbc code removed from release

obsoleted direct use of postgis_proc_upgrade.pl

scripts version unified with release version

A.28 Release 1.0.6

Release date: 2005/12/06

Contains a few bug fixes and improvements.

A.28.1 Upgrading

If you are upgrading from release 1.0.3 or later you *DO NOT* need a dump/reload.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the upgrade section of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an hard upgrade.

A.28.2 Bug fixes

Fixed palloc(0) call in collection deserializer (only gives problem with --enable-cassert)

Fixed bbox cache handling bugs

Fixed geom_accum(NULL, NULL) segfault

Fixed segfault in addPoint()

Fixed short-allocation in lwcollection_clone()

Fixed bug in segmentize()

Fixed bbox computation of SnapToGrid output

A.28.3 Improvements

Initial support for postgresql 8.2

Added missing SRID mismatch checks in GEOS ops

A.29 Release 1.0.5

Release date: 2005/11/25

Contains memory-alignment fixes in the library, a segfault fix in loader's handling of UTF8 attributes and a few improvements and cleanups.

Return code of shp2pgsql changed from previous releases to conform to unix standards (return 0 on success).

A.29.1 Upgrading

If you are upgrading from release 1.0.3 or later you *DO NOT* need a dump/reload.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the upgrade section of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an hard upgrade.

A.29.2 Library changes

Fixed memory alignment problems

Fixed computation of null values fraction in analyzer

Fixed a small bug in the getPoint4d_p() low-level function

Speedup of serializer functions

Fixed a bug in force_3dm(), force_3dz() and force_4d()

A.29.3 Loader changes

Fixed return code of shp2pgsql

Fixed back-compatibility issue in loader (load of null shapefiles)

Fixed handling of trailing dots in dbf numerical attributes

Segfault fix in shp2pgsql (utf8 encoding)

A.29.4 Other changes

Schema aware postgis_proc_upgrade.pl, support for pgsql 7.2+

New "Reporting Bugs" chapter in manual

A.30 Release 1.0.4

Release date: 2005/09/09

Contains important bug fixes and a few improvements. In particular, it fixes a memory leak preventing successful build of GiST indexes for large spatial tables.

A.30.1 Upgrading

If you are upgrading from release 1.0.3 you *DO NOT* need a dump/reload.

If you are upgrading from a release *between 1.0.0RC6 and 1.0.2* (inclusive) and really want a live upgrade read the upgrade section of the 1.0.3 release notes chapter.

Upgrade from any release prior to 1.0.0RC6 requires an hard upgrade.

A.30.2 Bug fixes

Memory leak plugged in GiST indexing

Segfault fix in transform() handling of proj4 errors

Fixed some proj4 texts in spatial_ref_sys (missing +proj)

Loader: fixed string functions usage, reworked NULL objects check, fixed segfault on MULTILINESTRING input.

Fixed bug in MakeLine dimension handling

Fixed bug in translate() corrupting output bounding box

A.30.3 Improvements

Documentation improvements

More robust selectivity estimator

Minor speedup in distance()

Minor cleanups

GiST indexing cleanup

Looser syntax acceptance in box3d parser

A.31 Release 1.0.3

Release date: 2005/08/08

Contains some bug fixes - *including a severe one affecting correctness of stored geometries* - and a few improvements.

A.31.1 Upgrading

Due to a bug in a bounding box computation routine, the upgrade procedure requires special attention, as bounding boxes cached in the database could be incorrect.

An hard upgrade procedure (dump/reload) will force recomputation of all bounding boxes (not included in dumps). This is *required* if upgrading from releases prior to 1.0.0RC6.

If you are upgrading from versions 1.0.0RC6 or up, this release includes a perl script (utils/rebuild_bbox_caches.pl) to force recomputation of geometries' bounding boxes and invoke all operations required to propagate eventual changes in them (geometry statistics update, reindexing). Invoke the script after a make install (run with no args for syntax help). Optionally run utils/postgis_proc_upgrade.pl to refresh postgis procedures and functions signatures (see Soft upgrade).

A.31.2 Bug fixes

Severe bugfix in lwgeom's 2d bounding box computation

Bugfix in WKT (-w) POINT handling in loader

Bugfix in dumper on 64bit machines

Bugfix in dumper handling of user-defined queries

Bugfix in create_undef.pl script

A.31.3 Improvements

Small performance improvement in canonical input function

Minor cleanups in loader

Support for multibyte field names in loader

Improvement in the postgis_restore.pl script

New rebuild_bbox_caches.pl util script

A.32 Release 1.0.2

Release date: 2005/07/04

Contains a few bug fixes and improvements.

A.32.1 Upgrading

If you are upgrading from release 1.0.0RC6 or up you *DO NOT* need a dump/reload.

Upgrading from older releases requires a dump/reload. See the upgrading chapter for more informations.

A.32.2 Bug fixes

Fault tolerant btree ops

Memory leak plugged in pg_error

Rtree index fix

Cleaner build scripts (avoided mix of CFLAGS and CXXFLAGS)

A.32.3 Improvements

New index creation capabilities in loader (-I switch)

Initial support for postgresql 8.1dev

A.33 Release 1.0.1

Release date: 2005/05/24

Contains a few bug fixes and some improvements.

A.33.1 Upgrading

If you are upgrading from release 1.0.0RC6 or up you *DO NOT* need a dump/reload.

Upgrading from older releases requires a dump/reload. See the upgrading chapter for more informations.

A.33.2 Library changes

BUGFIX in 3d computation of length_spheroid()

BUGFIX in join selectivity estimator

A.33.3 Other changes/additions

BUGFIX in shp2pgsql escape functions

better support for concurrent postgis in multiple schemas

documentation fixes

jdbc2: compile with "-target 1.2 -source 1.2" by default

NEW -k switch for pgsql2shp

NEW support for custom createdb options in postgis_restore.pl

BUGFIX in pgsql2shp attribute names unicity enforcement

BUGFIX in Paris projections definitions

postgis_restore.pl cleanups

A.34 Release 1.0.0

Release date: 2005/04/19

Final 1.0.0 release. Contains a few bug fixes, some improvements in the loader (most notably support for older postgis versions), and more docs.

A.34.1 Upgrading

If you are upgrading from release 1.0.0RC6 you *DO NOT* need a dump/reload.

Upgrading from any other precedent release requires a dump/reload. See the upgrading chapter for more informations.

A.34.2 Library changes

BUGFIX in transform() releasing random memory address

BUGFIX in force_3dm() allocating less memory then required

BUGFIX in join selectivity estimator (defaults, leaks, tuplecount, sd)

A.34.3 Other changes/additions

BUGFIX in shp2pgsql escape of values starting with tab or single-quote

NEW manual pages for loader/dumper

NEW shp2pgsql support for old (HWGEOM) postgis versions

NEW -p (prepare) flag for shp2pgsql

NEW manual chapter about OGC compliancy enforcement

NEW autoconf support for JTS lib

BUGFIX in estimator testers (support for LWGEOM and schema parsing)

A.35 Release 1.0.0RC6

Release date: 2005/03/30

Sixth release candidate for 1.0.0. Contains a few bug fixes and cleanups.

A.35.1 Upgrading

You need a dump/reload to upgrade from precedent releases. See the upgrading chapter for more information.

A.35.2 Library changes

BUGFIX in multi()

early return [when noop] from multi()

A.35.3 Scripts changes

dropped {x,y}{min,max}(box2d) functions

A.35.4 Other changes

BUGFIX in postgis_restore.pl scrip

BUGFIX in dumper's 64bit support

A.36 Release 1.0.0RC5

Release date: 2005/03/25

Fifth release candidate for 1.0.0. Contains a few bug fixes and a improvements.

A.36.1 Upgrading

If you are upgrading from release 1.0.0RC4 you *DO NOT* need a dump/reload.

Upgrading from any other precedent release requires a dump/reload. See the upgrading chapter for more informations.

A.36.2 Library changes

BUGFIX (segfaulting) in box3d computation (yes, another!).

BUGFIX (segfaulting) in estimated_extent().

A.36.3 Other changes

Small build scripts and utilities refinements.

Additional performance tips documented.

A.37 Release 1.0.0RC4

Release date: 2005/03/18

Fourth release candidate for 1.0.0. Contains bug fixes and a few improvements.

A.37.1 Upgrading

You need a dump/reload to upgrade from precedent releases. See the upgrading chapter for more informations.

A.37.2 Library changes

BUGFIX (segfaulting) in geom_accum().

BUGFIX in 64bit architectures support.

BUGFIX in box3d computation function with collections.

NEW subselects support in selectivity estimator.

Early return from force_collection.

Consistency check fix in SnapToGrid().

Box2d output changed back to 15 significant digits.

A.37.3 Scripts changes

NEW distance_sphere() function.

Changed get_proj4_from_srid implementation to use PL/PGSQL instead of SQL.

A.37.4 Other changes

BUGFIX in loader and dumper handling of MultiLine shapes

BUGFIX in loader, skipping all but first hole of polygons.

jdbc2: code cleanups, Makefile improvements

FLEX and YACC variables set *after* pgsql Makefile.global is included and only if the pgsql *stripped* version evaluates to the empty string

Added already generated parser in release

Build scripts refinements

improved version handling, central Version.config

improvements in postgis_restore.pl

A.38 Release 1.0.0RC3

Release date: 2005/02/24

Third release candidate for 1.0.0. Contains many bug fixes and improvements.

A.38.1 Upgrading

You need a dump/reload to upgrade from precedent releases. See the upgrading chapter for more informations.

A.38.2 Library changes

BUGFIX in transform(): missing SRID, better error handling.

BUGFIX in memory alignment handling

BUGFIX in force_collection() causing mapserver connector failures on simple (single) geometry types.

BUGFIX in GeometryFromText() missing to add a bbox cache.

reduced precision of box2d output.

prefixed DEBUG macros with PGIS_ to avoid clash with pgsql one

plugged a leak in GEOS2POSTGIS converter

Reduced memory usage by early releasing query-context palloced one.

A.38.3 Scripts changes

BUGFIX in 72 index bindings.

BUGFIX in probe_geometry_columns() to work with PG72 and support multiple geometry columns in a single table

NEW bool::text cast

Some functions made IMMUTABLE from STABLE, for performance improvement.

A.38.4 JDBC changes

jdbc2: small patches, box2d/3d tests, revised docs and license.

jdbc2: bug fix and testcase in for pgjdbc 8.0 type autoregistration

jdbc2: Removed use of jdk1.4 only features to enable build with older jdk releases.

jdbc2: Added support for building against pg72jdbc2.jar

jdbc2: updated and cleaned makefile

jdbc2: added BETA support for jts geometry classes

jdbc2: Skip known-to-fail tests against older PostGIS servers.

jdbc2: Fixed handling of measured geometries in EWKT.

A.38.5 Other changes

new performance tips chapter in manual

documentation updates: pgsql72 requirement, lwpostgis.sql

few changes in autoconf

BUILDDATE extraction made more portable

fixed spatial_ref_sys.sql to avoid vacuuming the whole database.

spatial_ref_sys: changed Paris entries to match the ones distributed with 0.x.

A.39 Release 1.0.0RC2

Release date: 2005/01/26

Second release candidate for 1.0.0 containing bug fixes and a few improvements.

A.39.1 Upgrading

You need a dump/reload to upgrade from precedent releases. See the upgrading chapter for more information.

A.39.2 Library changes

BUGFIX in pointarray box3d computation

BUGFIX in distance_spheroid definition

BUGFIX in transform() missing to update bbox cache

NEW jdbc driver (jdbc2)

GEOMETRYCOLLECTION(EMPTY) syntax support for backward compatibility

Faster binary outputs

Stricter OGC WKB/WKT constructors

A.39.3 Scripts changes

More correct STABLE, IMMUTABLE, STRICT uses in lwpostgis.sql

stricter OGC WKB/WKT constructors

A.39.4 Other changes

Faster and more robust loader (both i18n and not)

Initial autoconf script

A.40 Release 1.0.0RC1

Release date: 2005/01/13

This is the first candidate of a major postgis release, with internal storage of postgis types redesigned to be smaller and faster on indexed queries.

A.40.1 Upgrading

You need a dump/reload to upgrade from precedent releases. See the upgrading chapter for more informations.

A.40.2 Changes

Faster canonical input parsing.

Lossless canonical output.

EWKB Canonical binary IO with PG>73.

Support for up to 4d coordinates, providing lossless shapefile->postgis->shapefile conversion.

New function: UpdateGeometrySRID(), AsGML(), SnapToGrid(), ForceRHR(), estimated_extent(), accum().

Vertical positioning indexed operators.

JOIN selectivity function.

More geometry constructors / editors.

PostGIS extension API.

UTF8 support in loader.