Руководство пользователя PostGIS

перевод официальной версии документации 1.3.4, версия перевода от 16.08.2008

Резюме

PostGIS - расширение объектно-реляционной СУБД PostgreSQL предназначенное для хранения в базе географических данных. PostGIS включает поддержку пространственных индексов R-Tree/GiST и функции обработки геоданных.

Это руководство актуально для версии 1.3.4 (оригинал пособия). Последняя версия руководства: 1.4SVN

О переводе

Основная работа по переводу выполнена Дмитрием Скоробогатовым (http://xbb.uz). Правка, редакция и обновление: Максим Дубинин (sim@gis-lab.info). Перевод максимально приближен к оригиналу. В процессе перевода и редакции был также составлен небольшой вспомогательный словарь терминов (http://gis-lab.info/docs/postgis/manual/terms.html).

Содержание

1. Предисловие 1
1.1. Благодарности 1
1.2. Дополнительная информация 1
2. Установка 2
2.1. Требования 2
2.2. PostGIS 2
2.2.1.Создание пространственных баз данных, совместимых с PostGIS, с помощью шаблона разработки 3
2.2.2. Обновление 3
2.2.3. Обычные проблемы 4
2.3. JDBC 5
2.4. Загрузчик/дампер 5
3. Часто задаваемые вопросы 6
4. Использование PostGIS 8
4.1. Объекты ГИС 8
4.1.1. OpenGIS WKB и WKT 8
4.1.2. PostGIS EWKB, EWKT и Канонические формы 8
4.1.3. SQL-MM 9
4.2. Использование стандартов OpenGIS 10
4.2.1. Таблица SPATIAL_REF_SYS 10
4.2.2. Таблица GEOMETRY_COLUMNS 11
4.2.3. Создание пространственной таблицы 11
4.2.4. Обеспечение совместимости геометрий с OpenGIS 12
4.3. Загрузка данных ГИС 13
4.3.1. Использование SQL 13
4.3.2. Использование загрузчика 14
4.4. Получение данных ГИС 15
4.4.1. Использование SQL 15
4.4.2. Использование дампера 16
4.5. Построение индексов 17
4.5.1. Индексы GiST 17
4.5.2. Использование индексов 17
4.6. Сложные запросы 18
4.6.1. Преимущества индексов 18
4.6.2. Примеры пространственного SQL 19
4.7. Использование Mapserver 21
4.7.1. Основы использования 21
4.7.2. Часто задаваемые вопросы 23
4.7.3. Продвинутое использование 24
4.7.4. Примеры 24
4.8. Клиенты Java (JDBC) 26
4.9. Клиенты C (libpq) 28
4.9.1. Текстовые указатели 28
4.9.2. Бинарные указатели 28
5. Советы по производительности 29
5.1. Маленькие таблицы больших геометрий 29
5.1.1. Описание проблемы 29
5.1.2. Как обойти проблему 29
5.2. CLUSTER-изация геометрических индексов 29
5.3. Избегайте изменений размерности 30
6. Справочник PostGIS 31
6.1. Функции OpenGIS 31
6.1.1. Функции управления 31
6.1.2. Функции геометрической связи 31
6.1.3. Функции обработки геометрии 33
6.1.4. Геометрические способы доступа 34
6.1.5. Геометрические конструкторы 36
6.2.1. Функции управления 37
6.2.2. Операторы 39
6.2.3. Функции измерения 39

6.2.4. Геометрический вывод 40
6.2.5. Геометрические конструкторы 41
6.2.6. Геометрические редакторы 42
6.2.7. Линейные ссылки 44
6.2.8. Разное 45
6.2.9. Поддержка долгих транзакций 46
6.3. Функции SQL-MM 46
6.4. Функции ArcSDE 50
7. Сообщения о ошибках 51
7.1. Сообщения об ошибках в программном обеспечении 51
7.2. Сообщения об ошибках в документации 51
A.1. Примечания к релизам 52

1. Предисловие

PostGIS разрабатывается консалтинговой компанией Refractions Research Inc как проект в области изучения технологий пространственных баз данных. Refractions Research Inc специализируется на интеграции данных и разработке программного обеспечения на заказ и находится в Виктории, Британская Колумбия, Канада. Мы планируем сопровождение и дальнейшую разработку PostGIS в направлениях включения важной функциональности ГИС, полной поддержки OpenGIS, продвинутого топологического конструирования (покрытия, поверхности, сети), создания пользовательского интерфейса для просмотра и редактирования данных ГИС и веб средств доступа.

1.1. Благодарности

Sandro Santilli <strk@refractions.net>
Координация всех исправлений багов и усилий по поддержке, интеграция новой функциональности GEOS, разработка новых функций.

Mark Leslie <mleslie@refractions.net>
Постоянное сопровождение и разработка функций ядра.

Chris Hodgson <chodgson@refractions.net>
Сопровождение новых функций и привязка индексов 7.2.

Paul Ramsey <pramsey@refractions.net>
Поддержка документации и пакетирование.

Jeff Lounsbury <jeffloun@refractions.net>
Первоначальная разработка загрузчика/дампера shape-файлов.

Dave Blasby <dblasby@gmail.com>
Первоначальный разработчик PostGIS. Dave написал серверные объекты, привязку индексов и много серверных аналитических функций.

Другие участники
В алфавитном порядке: Alex Bodnaru, Alex Mayrhofer, Bruce Rindahl, Bernhard Reiter, Bruno Wolff III, Carl Anderson, Charlie Savage, David Skea, David Techer, IIDA Tetsushi, Geographic Data BC, Gerald Fenoy, Gino Lucrezi, Klaus Foerster, Kris Jurka, Mark Cave-Ayland, Mark Sondheim, Markus Schaber, Michael Fuhr, Nikita Shulga, Norman Vine, Olivier Courtin, Ralph Mason, Steffen Macke.

Важные библиотеки поддержки
Библиотека геометрических операций и алгоритмов GEOS (http://geos.refractions.net/) целиком разработана Martin Davis из Vivid Solutions.

Библиотека картографического проецирования Proj4 (http://trac.osgeo.org/proj/) разработана и сопровождается Gerald Evenden и Frank Warmerdam.

1.2. Дополнительная информация

Последнее ПО, документация и новости доступны на вебсайте PostGIS http://postgis.refractions.net.
Дополнительная информация о библиотеке геометрических операций GEOS доступна по адресу http://geos.refractions.net.
Дополнительная информация о библиотеке проецирования Proj4 доступна по адресу http://trac.osgeo.org/proj.
Дополнительная информация о сервере баз данных PostgreSQL доступна на главном сайте PostgreSQL http://www.postgresql.org.
Дополнительная информация об индексировании GiST доступна на сайте разработки PostgreSQL GiST http://www.sai.msu.su/~megera/postgres/gist.
Дополнительная информация о картографическом веб-сервере Mapserver доступна по адресу http://mapserver.gis.umn.edu.
"Simple Features for Specification for SQL" доступны на сайте Консорциума OpenGIS: http://www.opengis.org.

2. Установка

2.1. Требования

PostGIS имеет следующие требования для сборки и использования:
Полная инсталляция PostgreSQL (включая серверные заголовки). PostgreSQL можно взять на http://www.postgresql.org. Необходима версия 7.2 или выше.
Компилятор GNU C (`gcc`). Некоторые другие компиляторы ANSI C тоже могут быть использованы для компиляции PostGIS, но у вас будет гораздо меньше проблем, если будете использовать `gcc`.
GNU Make (`gmake` или `make`). Для большинства систем GNU `make` является версией make по умолчанию. Проверьте версию вызовом `make -v`. Другие версии `make` могут не обработать свойства `Makefile` PostGIS.
(Рекомендуется) Библиотека проекционных преобразований Proj4. Библиотека Proj4 используется в PostGIS для поддержки работы с проекционными преобразованиями координат. Proj4 можно скачать с http://www.remotesensing.org/proj.
(Рекомендуется) Геометрическая библиотека GEOS. Библиотека GEOS используется в PostGIS для проведения геометрических тестов (ST_Touches(), ST_Contains(), ST_Intersects()) и операций (ST_Buffer(), ST_Union(), ST_Difference()). GEOS можно скачать с http://geos.refractions.net.

2.2. PostGIS

Модуль PostGIS является расширением серверной части PostgreSQL. Поэтому для компиляции PostGIS 1.3.3 *необходим* полный доступ к заголовками сервера PostgreSQL. Исходный код PostgreSQL можно скачать с http://www.postgresql.org.

PostGIS 1.3.3 может быть собран только с PostgreSQL версии 7.2.0 или более старшей. Предыдущие версии PostgreSQL *не* поддерживаются.

Перед компиляцией серверных модулей PostGIS вы должны скомпилировать и установить пакет PostgreSQL.

Замечание: Если вы планируете использовать функциональность GEOS, то вы можете попробовать связать PostgreSQL со стандартной библиотекой C++:

```
LDFLAGS=-lstdc++ ./configure [ЗДЕСЬ ВАШИ ОПЦИИ]
```

Это позволяет обойти мнимые ошибки C++ в старых средствах разработки. Если вы столкнулись с непонятными проблемами (неожидано закрывается сервер или что-то подобное), попробуйте этот подход. Разумеется, он подразумевает повторную компиляцию вашего PostgreSQL.

Скачайте архив исходных кодов PostGIS: http://postgis.refractions.net/postgis-1.3.3SVN.tar.gz. Распакуйте архив:

```
# gzip -d -c postgis-1.3.3SVN.tar.gz | tar xvf -
```

Перейдите в директорию postgis-1.3.3SVN и выполните:

```
# ./configure
```

Если вам нужна поддержка проекционных преобразований координат, у вас должна быть установлена библиотека Proj4. Если ./configure ее не находит, используйте --with-proj=PATH с указанием директории, куда установлен Proj4.
Если вам нужна функциональность GEOS, у вас должна быть установлена библиотека GEOS. Если ./configure ее не находит, используйте --with-geos=PATH с указанием полного пути, по которому находится программа geos-config.

Выполните команды компиляции и установки.

```
# make # make install
```

Все файлы устанавливаются с использованием информации, предоставленной `pg_config`.
Библиотеки устанавливаются в `[pkglibdir]/lib/contrib`.
Важные файлы поддержки, такие как `lwpostgis.sql`, устанавливаются в `[prefix]/share/contrib`.
Бинарные файлы загрузчика и дампера устанавливаются в `[bindir]/`.

PostGIS требует расширения процедурного языка PL/pgSQL. До загрузки `lwpostgis.sql` вы должны включить PL/pgSQL. Для этого следует использовать команду `createlang`. Если по какой-то причине вам нужно сделать это вручную, обратитесь к Руководству программиста PostgreSQL.

```
# createlang plpgsql [yourdatabase]
```

Теперь загрузите описания объектов и функций PostGIS в вашу базу данных с помощью файла определений `lwpostgis.sql`.

```
# psql -d [yourdatabase] -f lwpostgis.sql
```

Теперь серверные расширения PostGIS загружены и готовы к использованию.

Установить полную базу определений систем координат EPSG можно загрузив файл определений `spatial_ref_sys.sql` который заполнит таблицу `SPATIAL_REF_SYS`.

```
# psql -d [yourdatabase] -f spatial_ref_sys.sql
```

2.2.1.Создание пространственных баз данных, совместимых с PostGIS, с помощью шаблона разработки

Некоторые пакетные дистрибутивы PostGIS (особенно инсталляторы под Win32 для PostGIS >= 1.1.5) загружают функции PostGIS в шаблон базы данных `template_postgis`. Если в вашей установке PostgreSQL содержится база данных `template_postgis`, то пользователи и/или приложения могут создавать пространственно-совместимые базы данных с помощью единственной команды. Заметим, что в обоих случаях пользователь базы данных должен иметь права на создание новых баз данных.
С помощью shell:

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

С помощью SQL:

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

2.2.2. Обновление

Обновление существующих пространственных баз данных может оказаться непростой задачей, требующей замены или введения новых определений объектов PostGIS.

К несчастью, не все определения могут быть легко заменены в существующей базе данных, поэтому лучше использовать dump/reload.

PostGIS предоставляет процедуру SOFT UPGRADE для незначительных релизов и исправлений ошибок, и процедуру HARD UPGRADE для значительных релизов.

Перед попыткой обновления PostGIS необходимо сделать резервное копирование данных. Если вы используете pg_dump с флагом -Fc, вы должны уметь восстанавливать дамп с помощью HARD UPGRADE.

2.2.2.1. SOFT-обновление

SOFT-обновление заключается в выполнении скрипта lwpostgis_upgrade.sql для вашей пространственной базе данных:

```
$ psql -f lwpostgis_upgrade.sql -d your_spatial_database
```

Не стесняйтесь сначала использовать SOFT-обновление, так как если скрипт не может его выполнить, он будет отменен и вы будете оповещены о необходимости выполнения HARD-обновления.

Замечание: Если вы не нашли файл `lwpostgis_upgrade.sql`, вероятно, вы используете версию ниже 1.1 и должны создать этот файл самостоятельно. Это делается следующей командой:

```
$ utils/postgis_proc_upgrade.pl lwpostgis.sql > lwpostgis_upgrade.sql
```

2.2.2.2. HARD-обновление

Под HARD-обновлением мы имеем в виду полный dump/reload баз данный использующих PostGIS. HARD-обновление необходимо, когда изменяются встроенные объекты PostGIS или когда невозможно SOFT-обновление. В приложении Примечания к релизам сообщается, необходим ли вам dump/reload (HARD-обновление) для перехода на тот или иной релиз.

PostGIS предоставляет полезный скрипт для востановления дампа, созданного командой pg_dump -Fc. Он является экспериментальным и переназначение его вывода в файл может помочь в разрешении проблем. Это делается так:
Пусть база данных, которую вы хотите обновить, называется "olddb". Создайте ее "custom-format" дамп.

```
$ pg_dump -Fc olddb > olddb.dump
```

Восстановим дамп после обновления PostGIS в новую базу данных. Новая база данных не обязательно должна существовать. Скрипт postgis_restore принимает параметры createdb после имени файла с дампом, и это может быть использовано, например, если вы используете нестандартную кодировку символов в своей базе данных. Давайте назовем эту базу "newdb" и зададим на ней кодировку символов UNICODE:

```
$ sh utils/postgis_restore.pl lwpostgis.sql newdb olddb.dump -E=UNICODE >
restore.log
```

Убедитесь, что все объекты восстанавливаемого дампа были реально восстановлены и не конфликтуют с определениями из lwpostgis.sql.

```
$ grep ^KEEPING restore.log | less
```

Если производится обновление PostgreSQL < 8.0 на >= 8.0, вы можете удалить столбцы attrelid, varattnum и stats в таблице geometry_columns, которые больше не нужны. Но их сохранение также безвредно. ИХ УДАЛЕНИЕ, КОГДА ОНИ ДЕЙСТВИТЕЛЬНО НУЖНЫ, ВЕСЬМА БОЛЕЗНЕННО!

```
$ psql newdb -c "ALTER TABLE geometry_columns DROP attrelid"
$ psql newdb -c "ALTER TABLE geometry_columns DROP varattnum"
$ psql newdb -c "ALTER TABLE geometry_columns DROP stats"
```

Таблица spatial_ref_sys восстанавливается из дампа для обеспечения сохранности пользовательских дополнений, но новый дистрибутив, возможно, модифицирует ее. Поэтому вам следует сделать резервную копию ее содержимого, удалить таблицу и заново ее создать. Если вы создавали дополнения, предполагается, что вы знаете как их сохранить перед обновлением таблицы. Их замена на новые совершается так:

```
$ psql newdb
newdb=> drop spatial_ref_sys;
DROP
newdb=> \i spatial_ref_sys.sql
```

2.2.3. Обычные проблемы

Есть несколько вещей, которые могут неожиданно помешать вашей установке или обновлению.
Проще всего распаковать дистрибутив PostGIS в директорию contrib в дереве исходников PostgreSQL. Однако, если это невозможно, вы можете задать переменную окружения `PGSQL_SRC`, указав путь к директории исходников PostgreSQL. Это позволит вам скомпилировать PostGIS, но если **make install** не сработает, будьте готовы самостоятельно разнести по соответствующим местам библиотеки PostGIS и исполняемые файлы.
Проверьте, что у вас установлен PostgreSQL 7.2 или старше, и что вы компилируете используя исходный код той же самой версии PostgreSQL, что и запущена. Путаница может возникнуть, если в вашем дистрибутиве (Linux) уже установлен PostgreSQL, или, если вы имеете другую, давно забытую версию установленного PostgreSQL. PostGIS будет работать только с PostgreSQL 7.2 или старше. Если вы попытаетесь использовать его со старой версией, то в результате получите сообщение об ошибке. Проверить версию запущенного PostgreSQL, можно подсоединившись к базе посредством psql и выполнив запрос:

```
SELECT version();
```

Если у вас RPM-дистрибутив, вы можете проверить наличие установленных пакетов с помощью команды **rpm** следующим способом: **rpm -qa | grep postgresql**

Также проверьте, что необходимые изменения внесены в начало Makefile.config. Они включают следующие изменения:
Если вы хотите работать с проекционными преобразованиями, вы должны установить библиотеку Proj4, в Makefile.config установить переменную `USE_PROJ` в 1, а пременной `PROJ_DIR` присвоить ваш префикс установки.
Если вы хотите использовать функции GEOS, вы должны установить библиотеку GEOS , в Makefile.config установить переменную `USE_GEOS` в 1, а переменной `USE_GEOS` присвоить ваш префикс установки.

2.3. JDBC

Расширения JDBC предоставляют соответствующие объекты Java внутренним типам PostGIS. Эти объекты могут быть использованы для написания Java-клиентов, совершающих запросы к базе данных PostGIS, получающих или обрабатывающих данные ГИС.
Перейдите в поддиректорию `java/jdbc` дистрибутива PostGIS.
Запустите команду `ant`. Скопируйте файл postgis.jar туда, где хранятся библиотеки java.
Расширения JDBC требуют чтобы драйвер PostgreSQL JDBC располагался в директории указанной в CLASSPATH в процессе сборки. Если драйвер PostgreSQL JDBC расположен в другом месте, можно передать его местоположение его файла JAR отдельно, используя параметр -D:

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

Драйверы PostgreSQL JDBC могут быть загружены с http://jdbc.postgresql.org.

2.4. Загрузчик/дампер

Загрузчик данных и дампер автоматически собирается и устанавливается как часть сборки PostGIS. Собрать и установить его вручную можно так:

```
# cd postgis-1.3.3SVN/loader
# make
# make install
```

Загрузчик вызывается как shp2pgsql и конвертирует ESRI Shape-файлы в SQL подходящий для загрузки в PostGIS/PostgreSQL. Дампер вызывается как pgsql2shp и конвертирует таблицы (или запросы) в ESRI Shape-файлы. Более подробная документация доступна в онлайн-справке и руководстве.

3. Часто задаваемые вопросы

Какие виды геометрических объектов имеются в моем распоряжении?

В вашем распоряжении point, line, polygon, multipoint, multiline, multipolygon, and geometrycollections (точка, линия, полигон, мультиточка, мультилиния, мультиполигон и геометрическая коллекция). Они определены в формате Well Known Text Open GIS (с расширениями XYZ, XYM, XYZM).

Как построить пространственный запрос?

Сначала вы должны создать таблицу со столбцом типа "geometry", который будет содержать ваши ГИС-данные. Соединитесь с вашей базой данных с помощью `psql` и выполните SQL:

```
CREATE TABLE gtest ( ID int4, NAME varchar(20) );
SELECT AddGeometryColumn('', 'gtest','geom',-1,'LINESTRING',2);
```

Если не получилось добавить столбец геометрии, то, вероятно, вы не загрузили функции и объекты PostGIS в свою базу данных. Смотрите инструкцию по установке.
Далее, вы можете вставлять геометрию в таблицу с помощью SQL-команды insert. Объект ГИС будет форматирован согласно формату "well-known text" Консорциума OpenGIS:

```
INSERT INTO gtest (ID, NAME, GEOM)
VALUES (
1,
'Первая геометрия',
GeomFromText('LINESTRING(2 3,4 5,6 5,7 8)', -1)
);
```

Подробную информацию о других объектах ГИС можно посмотреть в справочнике объектов.
Просмотр ваших ГИС-данных в таблице:

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

Результат должен выглядеть примерно так:

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

Как вставить объект ГИС в базу данных?

Так же, как вы строите другие запросы к базе данных, используя SQL-запрос для получения значений, функций, логических тестов.
Есть две важных вещи, которые следует учитывать при построении пространственных запросов: существует ли пространственный индекс, которым можно воспользовать и нужно ли произвести сложные вычисления на большом числе геометрий.
Чаще всего вам будет нужен "оператор пересечения" (&&), который проверяет пересекаются ли границы объектов. Польза оператора && заключается в том, что он может использовать пространственный индекс, если он существует. Это ускорит выполнения запроса.
Кроме того, вы можете использовать пространственные функции, такие как Distance(), ST_Intersects(), ST_Contains() and ST_Within() и другие для сужения результатов поиска. Большинство пространственных запросов включают тест на индекс и тест на пространственную функцию. Тест индекса полезен тем, что ограничивает число проверок значений теми, которые *могут* попасть в требуемый набор. Далее используются пространственные функции используются для окончательной проверки условия.

```
SELECT id, the_geom FROM thetable
WHERE
the_geom && 'POLYGON((0 0, 0 10, 10 10, 10 0, 0 0))'
AND
_ST_Contains(the_geom,'POLYGON((0 0, 0 10, 10 10, 10 0, 0 0))');
```

Как повысить скорость пространственных запросов в больших таблицах?

Быстрые запросы в больших таблицах, вместе с поддержкой транзакций, являются *назначением* пространственной базы данных. Поэтому очень важно иметь хорошие индексы.

Для создания пространственного индекса таблицы со столбцом `geometry`, используйте запрос "CREATE INDEX", как показано ниже:

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

Опция "USING GIST" означает, что сервер должен использовать индекс GiST (Generalized Search Tree - обобщенное поисковое дерево).
Замечание: Индексы GiST допускают потери. Индексы с потерями используют замещающие объекты (в пространственном случае - охват объекта) для создания индекса.
Вы должны также обеспечить планировщик запросов PostgreSQL достаточной информацией о вашем индексе, чтобы он мог выбирать когда их использовать. Чтобы сделать это, вам нужно "собрать статистику" по вашим таблицам с геометрией.
Для PostgreSQL 8.0.x и старше, выполните команду **VACUUM ANALYZE**.
Для PostgreSQL 7.4.x и младше, выполните команду **SELECT UPDATE_GEOMETRY_STATS()**

Почему не поддерживаются R-Tree индексы PostgreSQL?

Ранние версии PostGIS использовали R-Tree индексы PostgreSQL. Однако индексы стали ненужны начиная с версии 0.6, когда появились пространственные индексы со схемой R-Tree-через-GiST.
Наши тесты показали, что скорости поиска с использованием R-Tree и GiST индексов сопоставимы. Исходные R-Tree индексы PostgreSQL имеют два ограничения, которые делают их нежелательными для использования с геоданными (заметим, что эти ограничения присущи текущей реализации R-Tree в PostgreSQL, а не самой концепции R-Tree):
R-Tree индексы в PostgreSQL не умеют работать со объектами, размер которых превышает 8K. Индексы GiST умеют, "теряя" часть информации, заменяя собственно объекты на их охват.
R-Tree индексы в PostgreSQL не являются "null-безопасными", т.е. индекс на столбце геометрии, в котором содержатся геометрии null создать нельзя.

Почему я должен использовать функцию AddGeometryColumn() и все прочие функции совместимости с OpenGIS?

Если у вас нет необходимости использовать функции поддержки OpenGIS, не используйте их. Просто создайте таблицы как в старых версиях, определите столбцы геометрии с помощью CREATE. Вся ваша геометрия будет иметь SRID, равный -1, а таблицы мета-данных OpenGIS не будут должным образом заполнены. Однако большинство приложений, основанных на PostGIS, не смогут с этим работать, и мы советуем вам использовать `AddGeometryColumn()` при создания геометрических таблиц.
Одним из приложений, которое использует мета-данные `geometry_columns`, является Mapserver. Mapserver может использовать в SRID столбцов геометрии для перепроектирования объектов на лету в текущую проекцию карты.

Какой способ лучше использовать для поиска всех объектов в радиусе от другого объекта?

Чтобы использовать базу данных наиболее эффективно, лучше создать запрос, которые будет совмешать тест на попадание объекта в радиус и охват: тест на попадание в охват использует пространственный индекс, который предоставляет быстрый доступ к подмножеству данных, далее используется тест на нахождение в заданном радиусе от нужного объекта.
Для нахождения индексированных расстояний удобна функция `ST_DWithin(geometry, geometry, distance)`. Она создает поисковый прямоугольник, достаточно большой, чтобы включить все на расстояние радиуса поиска. Затем прямоугольник используется для объектов на необходимом расстоянии в результирующем проиндексированном подмножестве.
Например, чтобы найти все объекты в 100 метрах от POINT(1000 1000), хорошо работает следующий запрос:

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

Как включить в запрос перепроектирование координат?

Для перепроектирования, исходная и конечная система координат должны быть определены в таблице SPATIAL_REF_SYS, а геометрии должны иметь установленный SRID. Если это сделано, то перепроектирование просто осуществляется обращением к нужному SRID.

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

4. Использование PostGIS

4.1. Объекты ГИС

Объекты ГИС, поддерживаемые PostGIS, являются надмножествами "Simple Features", определенных Консорциумом OpenGIS (OGC). Начиная с версии 0.9, PostGIS поддерживает все объекты и функции, определенные OGC в спецификации "Simple Features SQL".

PostGIS расширяет стандарт поддержкой координат 3DZ, 3DM и 4D.

4.1.1. OpenGIS WKB и WKT

Спецификация OpenGIS определяет два стандартных способа определения пространственных объектов: в форме Well-Known Text (WKT) и в форме Well-Known Binary (WKB). WKT и WKB включают информацию о типе объекта и координаты, составляющие объект.
Примеры текстового представления (WKT) пространственных объектов приведены ниже:

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)))

Кроме этого, спецификация OpenGIS требует, чтобы внутренний формат хранения пространственных объектов включал идентификатор системы координат (spatial referencing system identifier - SRID). SRID необходим для добавления обекта в базу данных.

Ввод/вывод в этих форматах доступен с использованием следующих интерфейсов:

```
bytea WKB = asBinary(geometry);
text WKT = asText(geometry);
geometry = GeomFromWKB(bytea WKB, SRID);
geometry = GeometryFromText(text WKT, SRID);
```

Например, правильный запрос insert для создания и вставки пространственного объекта OGC может быть таким:

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

4.1.2. PostGIS EWKB, EWKT и Канонические формы

Формат OGC поддерживает только 2d геометрии, и соответствующие SRID никогда не вставляются в вводимые/выводимые представления.

PostGIS расширяет форматы текущего набора OGC (всякий валидный WKB/WKT является валидным EWKB/EWKT), но это может измениться в будущем, если OGC выпустит новый формат, противоречащий нашим. Таким образом, вам НЕ СЛЕДУЕТ полагаться на эту возможность!

В EWKB/EWKT PostGIS добавлена поддержка координат 3dm, 3dz, 4d и встроена информация SRID.

Примеры текстовых представлений (EWKT) пространственных объектов, расширенных описанными возможностями:

POINT(0 0 0) -- XYZ
SRID=32632;POINT(0 0) -- XY c SRID
POINTM(0 0 0) -- XM
POINT(0 0 0 0) -- XYZM
SRID=4326;MULTIPOINTM(0 0 0,1 2 1) -- XYM c 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))

Ввод/вывод в этих форматах возможен с использованием следующих интерфейсов:

```
bytea EWKB = asEWKB(geometry);
text EWKT = asEWKT(geometry);
geometry = GeomFromEWKB(bytea EWKB);
geometry = GeomFromEWKT(text EWKT);
```

Пример правильного запроса insert для создания и вставки пространственного объекта PostGIS:

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

"Канонические формы" типов в PostgreSQL позволяют вам получить представления с помощью простых запросов (без вызова каких либо функций) и гарантируют выполнение простых операций insert, update или копирования. Эти формы для PostGIS-овского типа "geometry":

```
- Вывод
- binary: EWKB
ascii: HEXEWKB (EWKB в 16-ричной форме)
- Ввод
- binary: EWKB
ascii: HEXEWKB|EWKT
```

Например, следующий запрос читает EWKT и возвращает HEXEWKB в процессе канонического ввода/вывода ascii:

```
=# SELECT 'SRID=4;POINT(0 0)'::geometry;
geometry
----------------------------------------------------
0101000020040000000000000000000000000000000000000000
(1 row)
```

4.1.3. SQL-MM

Пространственная спецификация мультимедийных приложений SQL расширяет простые возможности спецификации SQL, определяя несколько кривых, интерполированных дугами.

Определения SQL-MM определяют координаты 3dm, 3dz и 4d, но не позволяют включать информацию о SRID.

WKT расширения еще поддерживаются не полностью. Ниже показаны примеры некоторых простых криволинейных геометрий:

CIRCULARSTRING(0 0, 1 1, 1 0)
COMPOUNDCURVE(CIRCULARSTRING(0 0, 1 1, 1 0),(1 0, 0 1))
CURVEPOLYGON(CIRCULARSTRING(0 0, 4 0, 4 4, 0 4, 0 0),(1 1, 3 3, 3 1, 1 1))
MULTICURVE((0 0, 5 5),CIRCULARSTRING(4 0, 4 4, 8 4))
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)))

Замечание: В настоящее время PostGIS не поддерживает использование составных кривых в криволинейном полигоне.

Замечание: Все числовые сравнения с плавающей точкой в SQL-MM выполняются в пределах допустимой точности, в настоящее время - 1E-8.

4.2. Использование стандартов OpenGIS

Спецификация "Simple Features Specification for SQL" OpenGIS определяет стандартные типы объектов ГИС, функции для манипуляции ими, и набор таблиц метаданных. В целях сохранения корректности метаданных, такие операции как создание и удаление столбцов с пространственными данными осуществляются с помощью специальных процедур, определенных OpenGIS.

Существуют две таблицы метаданных OpenGIS: `SPATIAL_REF_SYS` и `GEOMETRY_COLUMNS`. Таблица `SPATIAL_REF_SYS` содержит числовые ID и текстовые описания систем координат, используемых в пространственной базе данных.

4.2.1. Таблица SPATIAL_REF_SYS

Таблица `SPATIAL_REF_SYS` определяется следующим образом:

```
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 имеет следующие столбцы:

SRID
Целое число - уникальный идентификатор системы координат (Spatial Referencing System, SRS) в пределах базы данных.

AUTH_NAME
Название стандарта или стандартизирующего организации, являющейся источником информации о данной системе координат. Например, правильным значением `AUTH_NAME` будет "EPSG".

AUTH_SRID
Идентификатор системы координат, так как он определяется организацией указанной в `AUTH_NAME`. В случае EPSG, это должен быть код проекции EPSG.

SRTEXT
WKT представление системы координат. Пример WKT SRS представления:

```
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]
]
```

Список кодов проекций EPSG и соответствующих им представлений WKT можно найти на http://www.opengis.org/techno/interop/EPSG2WKT.TXT. Общее обсуждение WKT можно прочитать в документе Open GIS "Coordinate Transformation Services Implementation Specification": http://www.opengis.org/techno/specs.htm. Информацию о European Petroleum Survey Group (EPSG) и их базе данных систем координат можно найти на http://epsg.org.

PROJ4TEXT
PostGIS использует библиотеку Proj4 для преобразований систем координат. Столбец PROJ4TEXT содержит строку определение координат Proj4 для данного SRID. Например:

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

Смотрите подробности на сайте Proj4: http://www.remotesensing.org/proj. Файл spatial_ref_sys.sql содержит определения SRTEXT и PROJ4TEXT для всех проекций EPSG.

4.2.2. Таблица GEOMETRY_COLUMNS

Таблица GEOMETRY_COLUMNS определяется следующим образом:

```
CREATE TABLE geometry_columns (
        f_table_catalog    VARRCHAR(256) NOT NULL,
        f_table_schema     VARCHAR(256) NOT NULL,
        f_table_nam        VARCHAR(256) NOT NULL,
        f_geometry_column  VARCHAR(256) NOT NULL,
        coord_dimension    INTEGER NOT NULL,
        srid               INTEGER NOT NULL,
        type               VARCHAR(30) NOT NULL
)
```

Она имеет следующие столбцы:

F_TABLE_CATALOG, F_TABLE_SCHEMA, F_TABLE_NAME
Все составляющие имени таблицы, содержащей столбец геометрии. Заметим, что термины "catalog" и "scheme" заимствованы из Oracle. В PostgreSQL нет аналога для "catalog", поэтому этот столбец остается пустым. Для "scheme" используется имя схемы PostgreSQL (по умолчанию public).

F_GEOMETRY_COLUMN
Имя столбца геометрии в таблице объектов.

COORD_DIMENSION
Пространственная размерность столбца (2, 3 или 4 измерения).

SRID
Идентификатор системы координат, используемой для геометрии в этой таблице. Он является внешним ключом для таблицы SPATIAL_REF_SYS.

TYPE
Тип пространственного объекта. Можно использовать один из следующих: POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION или соответствующим XYM-версиям POINTM, LINESTRINGM, POLYGONM, MULTIPOINTM, MULTILINESTRINGM, MULTIPOLYGONM, GEOMETRYCOLLECTIONM. Для разнородных коллекций смешанных типов можно использовать "GEOMETRY" как тип.

Замечание: Этот атрибут, возможно, не является частью спецификации OpenGIS, но необходим для обеспечения типового единообразия.

4.2.3. Создание пространственной таблицы

Создание таблицы с пространственными данными выполняется в два шага:
Создайте обычную непространственную таблицу.
Например: CREATE TABLE ROADS_GEOM (ID int4, NAME varchar(25))
Добавьте в таблицу пространственный столбец с помощью функции OpenGIS "AddGeometryColumn".
Ее синтаксис:

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

```
        <dimension>
)
```

Или с использованием текущей схемы:

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

Пример 1: SELECT AddGeometryColumn('public', 'roads_geom', 'geom', 423, 'LINESTRING', 2)
Пример 2: SELECT AddGeometryColumn('roads_geom', 'geom', 423, 'LINESTRING', 2)

Приведем пример использования SQL для создания таблицы и добавления пространственного столбца (при условии, что SRID 128 уже существует):

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

Приведем другой пример, использующий общий тип "geometry" и неопределенный SRID -1:

```
CREATE TABLE roads (
        road_id INTEGER,
        road_name VARCHAR
);
SELECT AddGeometryColumn( 'roads', 'roads_geom', -1, 'GEOMETRY', 3 );
```

4.2.4. Обеспечение совместимости геометрий с OpenGIS

Большинство функций, предоставленных библиотекой GEOS, основаны на предположении, что ваша геометрии соответствуют спецификации Simple Feature Specification OpenGIS. Вы можете проверить соответствие геометрий функцией IsValid():

```
gisdb=# select isvalid('LINESTRING(0 0, 1 1)'),
      isvalid('LINESTRING(0 0,0 0)');


 isvalid | isvalid
---------+---------
       t |       f
```

По умолчанию PostGIS не применяет проверку соответствия вводимой геометрии, так как проверка требует дополнительного процессорного времени для сложных геометрий, особенно для полигонов. Если вы не доверяете вашим исходным данным, вы можете вручную включить такую проверку для своей таблицы, добавив ограничение "CHECK":

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

Если вы столкнулись с непонятной ошибкой, такой как например "GEOS Intersection() threw an error!" или "JTS Intersection() threw an error!", которые генерируются функциями PostGIS на правильных геометриях, то, возможно, вы обнаружили ошибку в PostGIS или в одной из используемых им библиотек, и вам следует обратиться к разработчикам PostGIS. Это касается и тех случаев, когда функция PostGIS возвращает неправильную геометрию при правильном вводе.

Замечание: Геометрии, строго соответствующие OGC, не должны иметь значений Z или M. Функция IsValid() не считает многомерные геометрии неправильными! Вызывая AddGeometryColumn(), вы можете указать ограничение числа измерений геометрии. Для строгого соответствия ограничьте число измерений двумя.

4.3. Загрузка данных ГИС

После того как пространственная таблица создана, вы можете загружать данные ГИС в базу. В настоящее время существуют два способа загрузить данные в базу PostGIS/PostgreSQL: использование команд SQL или использование загрузчика/дампера шейп-файлов.

4.3.1. Использование SQL

Если вы можете конвертировать ваши данные в текстовое представление, то для загрузки данных в PostGIS проще всего использовать формат SQL. Как и в случае Oracle и других баз SQL данные могут быть загружены через терминал SQL из текстового файла, содержащего SQL-запросы "INSERT".

Файл (например `roads.sql`) с загружаемыми данными может выглядеть так:

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

Данные из файла могут быть легко загружены в PostgreSQL с помощью терминала SQL "psql":

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

4.3.2. Использование загрузчика

Загрузчик данных `shp2pgsql` конвертирует шейп-файлы ESRI в соответствующий SQL для вставки в базу данных PostGIS/PostgreSQL. Загрузчик имеет несколько рабочих режимов, задаваемых флагами командной строки:

-d
Удалить таблицу базы данных перед созданием новой таблицы с данными из шейп-файла.

-a
Добавить данные из шейп-файла в таблицу базы данных. Заметим, что для использования этой опции при загрузке множества файлов, эти файлы должны иметь одинаковый набор атрибутов и одинаковые типы данных.

-c
Создать новую таблицу и заполнить ее из шейп-файла. *Этот режим включен по умолчанию.*

-p
Произвести только код SQL для создания таблицы без добавления данных. Может быть использована, если вам нужно разделить шаги по созданию таблицы и загрузке данных.

-D
Использовать формат PostgreSQL "dump" для вывода данных. Эта опция может быть скомбинирована с -a, -c и -d. Это позволит ускорить загрузку по сравнению с форматом SQL по умолчанию "insert". Используйте ее для очень больших наборов данных.

-s <SRID>
Создает и заполняет таблицы геометрии указанными SRID.

-k
Сохранять регистр идентификаторов (столбец, схема и атрибуты). Заметим, что все атрибуты в шейп-файлах имеют ВЕРХНИЙ регистр.

-i
Привести все целые числа к стандартным 32-битным целочисленными, не создавать 64-битных чисел формата bigint, даже если заголовок DBF это позволяет.

-I
Создает индекс GiST для столбца геометрии.

-w
Вывести формат WKT для использования в старых (0.x) версиях PostGIS. Заметим, что это приведет к изменению координат и удалению значений M из шейп-файла.

-W <кодировка>
Указывает кодировку вводимых данных (в файле DBF). При ее использовании все атрибуты в DBF будут конвертироваться из указанной кодировки в UTF8. Результат SQL будет содержать команду `SET CLIENT_ENCODING to UTF8`. Таким образом, данные будут конвертированы из UTF8 в любую внутреннюю кодировку, с которой вы сконфигурировали базу данных.

Заметим, что -a, -c, -d и -p взаимно исключают друг друга.

Посмотрите пример сессии использования загрузчика для создания файла ввода и его загрузки:

```
# shp2pgsql shaperoads myschema.roadstable > roads.sql
# psql -d roadsdb -f roads.sql
```

В UNIX конвертация и загрузка могут буть выполнены за один шаг с помощью команды:

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

4.4. Получение данных ГИС

Данные могут быть извлечены из базы с помощью SQL или загрузчика/дампера шейп-файлов. В разделе об SQL мы обсудим некоторые запросы к пространственным таблицам и операторы для сравнения данных.

4.4.1. Использование SQL

Самый простой способ получить данные из базы - использовать SQL-запрос "SELECT" с сохранением результирующих столбцов в форматированный текстовый файл:

```
db=# SELECT road_id, 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)
```

Иногда необходимо сократить число возвращаемых полей. В случае ограничений на основе атрибутов, просто используйте тот же синтаксис SQL, как в случае обычной, непространственной таблицы. В случае пространственных ограничений можно использовать следующие операторы:

&&
Этот оператор используется для проверки, пересекаются ли границы одной геометрии с границами другой.

~=
Этот оператор проверяет, являются ли две геометрии геометрически идентичными. Например, 'POLYGON((0 0,1 1,1 0,0 0))' совпадает с 'POLYGON((0 0,1 1,1 0,0 0))'.

=
Более простой оператор. Он проверяет только совпадение границ геометрий.

Вы можете использовать эти операторы в запросах. Заметим, что если геометрии и охваты одновременно указываются в строке команды SQL, вы должны явно указывать необходимость конвертации строки в геометрию с помощью функции "GeomFromText()". Например:

```
SELECT road_id, road_name
FROM roads
WHERE roads_geom ~= GeomFromText('LINESTRING(191232 243118,191108
243242)',-1);
```

Этот запрос возвратит единственную запись таблицы "ROADS_GEOM", геометрия которой равна указанному значению.

Когда используется оператор "&&", вы можете для сравнения указывать как BOX3D так и GEOMETRY. Разумеется, если вы указываете GEOMETRY, для сравнения будет использоваться охват объекта.

```
SELECT road_id, road_name
FROM roads
WHERE roads_geom && GeomFromText('POLYGON((...))',-1);
```

Этот запрос будет использовать для сравнения охват полигона.

Большинство простых пространственных запросов, вероятно, являются "фрейм-ориентированными" ("frame-based") запросами, используемыми в клиентском ПО, таком, как просмотровщики данных и приложения для веб картографии, захватывающие "фрейм" данных для показа. Используйте для фрейма объект "BOX3D", как в запросе ниже:

```
SELECT AsText(roads_geom) AS geom
FROM roads
WHERE
 roads_geom && SetSRID('BOX3D(191232 243117,191232 243119)'::box3d,-1);
```

Отметьте, использование SRID для определения проекции BOX3D. Значение -1 используется для указания неопределенного SRID.

4.4.2. Использование дампера

Дампер таблиц `pgsql2shp` соединяется с базой данных и конвертирует таблицу (возможно, заданную запросом) в шейп-файл. Основной синтаксис:

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

Опции командной строки:

-f <имя_файла>
Записать вывод в указанный файл.

-h <хост>
Хост базы данных, с которой осуществляется соединение.

-p <порт>
Порт базы данных, с которой осуществляется соединение.

-P <пароль>
Пароль, используемый для соединения с базой данных.

-u <пользователь>
Имя пользователя, используемое для соединения с базой данных.

-g <геометрический столбец>
Эта опция указывает столбец геометрии, используемый для записи в шейп-файл. Используется для таблиц с несколькими столбцами геометрии.

-b
Использовать бинарный курсор. Это ускорит операцию, но не будет работать, если какой-то из НЕ-геометрических атрибутов таблицы не может быть приведен к тесту.

-r
raw-режим. Не удаляйте поле `gid`, и не эскейпируйте имена столбцов.

-d
Для обратной совместимости: записывает дамп в 3-мерный шейп-файл для старых (до 1.0.0) баз данных PostGIS (по умолчанию в этом случае записывается 2-мерный шейп-файл). Начиная с PostGIS-1.0.0+, измерения кодируются полностью.

4.5. Построение индексов

Индексы делают возможным использование пространственной базы данных для больших наборов данных. Без индексации, любой поиск приводил бы к "последовательному сканированию" каждой записи в базе данных. Индексация организовывает данные в поисковое дерево, по которому можно быстро перемещаться, чтобы быстро найти конкретную запись. PostgreSQL по умолчанию поддерживает три вида индексов: индексы B-Tree, индексы R-Tree и индексы GiST.
B-Tree (B-деревья) используются, когда данные могут быть отсортированы вдоль одной оси; например, числа, символы, даты. Данные ГИС не могут быть рациональным способом отсортированы вдоль одной оси (что больше: (0,0) или (0,1) или (1,0)?), а потому для их индексирования B-Tree не помогут.
R-Tree (R-деревья) разбивают данные на прямоугольники, под-прямоугольники, под-под-прямоугольники и т.д. R-Tree используются в некоторых пространственных базах данных для индексации данных ГИС, но в PostgreSQL реализация R-Tree не столь надежна, как реализация GiST.
Индексы GiST (Generalized Search Trees - обобщенные деревья поиска) разделяют данные на "объекты по одну сторону " ("things to one side"), "пересекающиеся объекты " ("things which overlap"), "объекты внутри" ("things which are inside") и могут быть использованы для многих типов данных, включая данные ГИС. PostGIS использует реализацию R-Tree поверх GiST для индексации данных ГИС.

4.5.1. Индексы GiST

GiST означает "обобщенное поисковое дерево" ("Generalized Search Tree") и является общей формой индексации. Кроме индексации ГИС, GiST используется для ускорения поиска для всех видов нерегулярных структур данных (целочисленные массивы, спектральные данные и т.д.), к которым неприменимо обычное индексирование B-Tree.
Когда таблица данных ГИС вырастает до нескольких тысяч записей, создание индексов необходимо для ускорения поиска пространственных данных (кроме случаев, когда весь ваш поиск основывается атрибутах. Тогда вам достаточно обычных индексов полей атрибутов).

Ниже описан синтаксис запроса для создания GiST-индекса столбца "geometry":

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

Создание пространственного индекса требует интенсивных вычислений: для таблицы размером 1 миллион строк на машине Solaris 300MHz создание индекса заняло около 1 часа. После создания индекса важно заставить PostgreSQL собрать табличную статистику, которая используется для оптимизации запросов:

```
VACUUM ANALYZE [table_name] [column_name];
-- Это необходимо только для PostgreSQL 7.4 и более старых версий
SELECT UPDATE_GEOMETRY_STATS([table_name], [column_name]);
```

В PostgreSQL индексы GiST имеют два преимущества перед R-Tree. Во-первых, индексы GiST являются "null-безопасными" ("null safe"). Это означает, что они могут индексировать столбцы, содержащие значения null. Во-вторых, индексы GiST поддерживают концепцию "потерь" ("lossiness"), которая имеет значение, когда объекты ГИС занимают больше 8K (размер страницы PostgreSQL). Потери позволяют PostgreSQL сохранять в индексе только "значимую" часть объекта. В случае объектов ГИС, ими являются охваты. R-Tree не может быть создан для объектов ГИС, занимающих больше 8K.

4.5.2. Использование индексов

Как правило, индексы ускоряют доступ к данным оставаясь при этом прозрачными: как только индекс создан, планировщик запросов самостоятельно решает, использовать ли индексную информацию для создания быстрого плана запроса. К сожалению, планировщик запросов в PostgreSQL неоптимально использует индексы GiST. Так, иногда, там, где при поиске должен использоваться пространственный индекс, используется последовательное сканирование всей таблицы выполняемое по умолчанию.

Если вы обнаружили, что ваши пространственные индексы не используются (впрочем, как и индексы атрибутов), есть пара вещей, которые вы можете сделать:

Во-первых, убедитесь, что собрана статистика о числе и распределении значений в таблице, для предоставления планировщику запросов лучшей информации при принятии решений об использовании индекса. Для PostgreSQL 7.4 и более ранних это делается запуском **update_geometry_stats([table_name, column_name])** (подсчет распределения) и **VACUUM ANALYZE [table_name] [column_name]** (подсчет числа значений). В PostgreSQL 8.0 запуск **VACUUM ANALYZE** выполнит обе операции. Вам следует регулярно производить вакуумизацию базы данных. Большинство DBA PostgreSQL позволяют регулярно выполнять **VACUUM**, как задачу cron во время слабой загрузки базы.
Если вакуумизация не работает, вы можете заставить планировщик использовать индекс с помощью команды **SET ENABLE_SEQSCAN=OFF**. Вам следует осторожно использовать эту команду, и только для запросов с пространственными индексами: как правило, планировщик лучше вас знает, когда следует использовать B-

Tree. После выполнения запроса, вам следует восстановить прежнее значение ENABLE_SEQSCAN, чтобы другие запросы обрабатывались планировщиком как обычно.
Замечание: Начиная с версии 0.6 нет необходимости заставлять планировщик использовать индекс с ENABLE_SEQSCAN.

Если вы считаете, что планировщик неправильно оценивает расходы на последовательное сканирование по сравнению с использованием индекса, можете уменьшить величину random_page_cost в postgresql.conf или использовать SET random_page_cost=#. По умолчанию этот параметр равен 4. Попробуйте установить его в 1 или 2. Уменьшение значения сделает планировщик более склонным к использованию индексов.

4.6. Сложные запросы

Смысл функциональности пространственных баз данных заключается в выполнении запросов к базе данных, для которых иначе потребовалась бы настольная ГИС. Эффективное использование PostGIS требует знания доступных пространственных функций и умения создавать индексы, обеспечивающие эффективную работу.

4.6.1. Преимущества индексов

Конструируя запрос важно помнить, что только операторы работающие с охватами (bounding-box-based), типа && могут использовать пространственные индексы GiST. Такие функции, как distance() не могут использовать индекс для оптимизации своих операций. Например, следующий запрос был бы очень медленным на большой таблице:

```
SELECT the_geom
FROM geom_table
WHERE ST_Distance(the_geom, GeomFromText('POINT(100000 200000)', -1)) <
100
```

Этот запрос выбирает все геометрии из geom_table, которые находятся на расстоянии в 100 единиц от точки (100000, 200000). Он будет выполняться медленно, так как влечет за собой вычисления расстояний между каждой точкой в таблице и нашей точкой, т.е. одно вычисление ST_Distance() для каждой строки таблицы. Мы можем избежать этого с помощью оператора &&, уменьшающего число необходимых вычислений:

```
SELECT the_geom
FROM geom_table
WHERE the_geom && 'BOX3D(90900 190900, 100100 200100)'::box3d
  AND
ST_Distance(the_geom, GeomFromText('POINT(100000 200000)', -1)) < 100
```

Этот запрос выбирает те-же геометрии, но делает это более эффективно. При условии, что для the_geom существует индекс GiST, планировщик будет считать, что использование этого индекса уменьшит число строк, на которых неоходимо вычислять функцию distance(). Заметим, что геометрия BOX3D, которая используется с оператором &&, является прямоугольником со стороной в 200 единиц, центрированным в нужной точке - это наш "прямоугольник запроса" ("query box"). Оператор && использует индекс, чтобы быстро уменьшить число рассматриваемых записей, выбирая только те геометрии, которые пересекаются с "прямоугольником запроса". Если предположить, что наш прямоугольник запросов намного меньше, чем все геометрии в таблице, то это позволит резко сократить число вычислений расстояний, которые должны быть сделаны.
Изменение поведения: Начиная с PostGIS 1.3.0 большинство функций геометрических отношений (Geometry Relationship Functions), за исключением ST_Disjoint и ST_Relate, скрыто включают операторы определения охвата.

4.6.2. Примеры пространственного SQL

Примеры этого раздела будут использовать две таблицы: таблицу улиц (линии) и таблицу границ муниципалитетов (полигоны). Таблица `bc_roads` будет определена так:

```
Column      | Type              | Description
------------+-------------------+-------------------
gid         | integer           | Уникальный ID
name        | character varying | Имя улицы
the_geom    | geometry          | Геометрия расположения (Linestring)
```

Таблица `bc_municipality` будет определена так:

```
Column     | Type              | Description
-----------+-------------------+-------------------
gid        | integer           | Уникальный ID
code       | integer           | Уникальный ID
name       | character varying | Название муниципалитета / города
the_geom   | geometry          | Геометрия расположения (Polygon)
```

Какова общая длина всех улиц в километрах?
Ответ на этот вопрос может дать очень простой SQL-запрос:

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

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

Какова площадь города "Prince George" в гектарах?

В этом запросе скомбинированы атрибутивное условие (имя города) и пространственное вычисление (площадь):

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

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

Какой муниципалитет является самым большим в провинции по площади?
Этот запрос производит пространственные измерения согласно условиям вопроса. Есть несколько путей решения этой проблемы, но наиболее эффективным является следующий:

```
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)
```

Заметим, что для ответа на данный вопрос, мы должны вычислить площадь каждого полигона. Чтобы не делать этого постоянно, имеет смысл создать в таблице столбец площадей с отдельным индексом для повышения производительности. Отсортировав результаты по убыванию и использовав команду PostgreSQL "LIMIT" мы сможем легко выбрать наибольшее значение без помощи таких агрегирующих функций, как max().

Какова длина улиц, полностью находящихся в пределах своего муниципалитета?
Это - пример "пространственного объединения " (spatial join) , в котором объединяются данные из двух таблиц (join), но вместо обычного реляционного подхода к объединению (по внешнему ключу), в качестве условия объединения используется пространственное условие взаимоположения ("содержится в"):

```
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
...
```

Этот запрос выполняется заметное время, так как все дороги в таблице влияют на конечный результат (для нашего конкретного примера таблицы - это около 250K дорог). Для небольших покрытий (от нескольких сотен, до нескольких тысяч записей) ответ может быть очень быстрым.

Создать новую таблицу со всеми улицами в городе "Prince George".
Это - пример "наложения" (overlay) , которое использует пространственные данные из двух таблиц и заносит результат в новую таблицу. В отличие от показанного выше "пространственного объединения ", запрос создает новые геометрии. Оверлей похож на пространственный join с турбонаддувом, и полезен для более точной аналитической работы:

```
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 ST_Intersects(r.the_geom, m.the_geom)
  AND m.name = 'PRINCE GEORGE';
```

Какова длина в километрах улицы "Douglas St" в городе "Victoria"?

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

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

Какой муниципалитет является наибольшим из тех, чей полигон содержит дырку?

```
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)
```

4.7. Использование Mapserver

Mapserver - картографический веб-сервер, который поддерживает спецификацию OpenGIS Web Mapping Server.
Домашняя страница Mapserver: http://mapserver.gis.umn.edu.
Спецификация OpenGIS для веб-карт: http://www.opengis.org/techno/specs/01-047r2.pdf.

4.7.1. Основы использования

Для использования PostGIS с Mapserver, вы должны знать, как конфигурировать Mapserver, это выходит за рамки данной документации. В этом разделе описывается специфика использования PostGIS и детали конфигурирования.

Для использования PostGIS с Mapserver вам понадобятся:
PostGIS версии 0.6 или выше.
Mapserver версии 3.5 или выше.

Mapserver получает доступ к данным PostGIS/PostgreSQL, как и любой другой клиент PostgreSQL, с помощью `libpq`. Это означает, что Mapserver может быть установлен на любую машину с сетевым доступом к серверу PostGIS, если только в системе есть клиентские библиотеки PostgreSQL `libpq`.

Скомпилируйте и установите Mapserver с любыми нужными вам опциями включив опцию конфигурации "--with-postgis".
В map-файл вашего Mapserver-а добавьте слой PostGIS. Например:

```
LAYER
  CONNECTIONTYPE postgis
  NAME "widehighways"
  # Соединение с удаленной пространственной базой данных
  CONNECTION "user=dbuser dbname=gisdatabase host=bigserver"
  # Получение линий из столбца 'geom' таблицы 'roads'
  DATA "geom from roads"
  STATUS ON
  TYPE LINE
  # Из имеющихся линий выбрать только широкие дороги
  FILTER "type = 'highway' and numlanes >= 4"
  CLASS
        # Супермагистрали сделать более яркими и установить их ширину в
2 пикселя
      EXPRESSION ([numlanes] >= 6)
        COLOR 255 22 22
        SYMBOL "solid"
      SIZE 2
  END
  CLASS
      # Все остальные - более темные и широной в 1 пиксель
      EXPRESSION ([numlanes] < 6)
      COLOR 205 92 82
   END
END
```

Приведенный пример содержит следующие специфические директивы относящиеся к PostGIS:

CONNECTIONTYPE
Для слоев PostGIS всегда должен быть равен "postgis".

CONNECTION
Соединение с базой данных управляется "строкой соединения ", которая содержит стандартный набор указанных ниже ключей и значений (значения по умолчанию заключены в скобки <>):
user=<имя пользователя> password=<пароль> dbname=<имя базы> hostname=<имя сервера> port=<5432>
Пустая строка коннекта считается валидной, а любая пара ключ/значение может быть опущена. Как минимум вы должны указать имя базы и имя пользователя для соединения с ней.

DATA
Этот параметр указывается в форме "<столбец> from <имя_таблицы>", где столбец подразумевает пространственный столбец, представленный на карте.

FILTER
Фильтр должен быть правильной строкой SQL, соответствующей логике, обычно следующей после ключевого слова "WHERE" в SQL-запросе. Так, например, для выбора только 6-рядных (или более) улиц, используется фильтр "num_lanes >= 6".

Постройте пространственные (GiST) индексы на тех данных, с помощью которых будете рисовать какие-либо слои.

```
CREATE INDEX [имя_индекса] ON [имя_таблицы] USING GIST (
[столбец_геометрии] );
```

Если вы собираетесь запрашивать свои слои с помощью Mapserver, вам понадобится "индекс oid".
При запросах Mapserver требует уникальные идентификаторы для каждого пространственного объекта и модуль Mapserver PostGIS использует значения `oid` PostgreSQL в качестве таких идентификаторов. Побочным эффектом этого является то, что для быстрого доступа к произвольной записи в запросе требуется индекс по `oid`.

Для создания "индекса oid" используйте следующий запрос SQL:

```
CREATE INDEX [имя_индекса] ON [имя_таблицы] ( oid );
```

4.7.2. Часто задаваемые вопросы

Когда я использую **EXPRESSION** в своем map-файле, условие никогда не возвращает истину, даже когда я знаю, что в моей таблице существует нужное значение.
В отличие от шейп-файлов, имена полей PostGIS, встречающиеся в EXPRESSIONS, должны быть написаны в *нижнем регистре*.

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

FILTER, который я использую для моих шейп-файлов, не работает для моей таблицы PostGIS с теми же самыми данными.
В отличие от шейп-файлов, фильтры для слоев PostGIS используют синтаксис SQL (они присоединяются к предложению SQL, которое создает коннектор PostGIS для рисования слоев в Mapserver).

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

Мой слой PostGIS выдается медленнее, чем мой слой из шейп-файла, это нормально?
Как правило, ожидание слоев PostGIS на 10% дольше, чем эквивалентных слоев шейп-файлов, из-за дополнительных накладных расходов: соединение с базой данных, преобразование данных и передача данных между базой и Mapserver.
Если вы столкнулись с серьезной проблемой производительности, то, вероятно, вы не создали пространственного индекса для своей таблицы.

```
postgis# CREATE INDEX geotable_gix ON geotable USING GIST ( geocolumn );
postgis# SELECT update_geometry_stats(); -- For PGSQL < 8.0
postgis# VACUUM ANALYZE; -- For PGSQL >= 8.0
```

Мой слой PostGIS выдается хорошо, но запросы реально тормозят. Что неправильно?

Для ускорения запросов вы должны иметь уникальный ключ в своей пространственной таблице и индекс на нем.

С помощью пункта `USING UNIQUE` в своей строке `DATA` вы можете указать mapserver-у какой уникальный ключ использовать:

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

Если ваша таблица не имеет определенного уникального столбца, PostgreSQL позволяет вам "подделать" уникальный столбец с помощью "oid" в качестве такого столбца. "oid" является уникальным столбцом по умолчанию на случай, если вы таковой не декларировали. Вы сможете увеличить скорость выполнения запросов, создав индекс на значении oid вашей пространственной таблицы.

```
postgis# CREATE INDEX geotable_oid_idx ON geotable (oid);
```

4.7.3. Продвинутое использование

Функция USING предложений псевдо-SQL используется для того, чтобы предоставить MapServer некоторую информацию, которая поможет ему понять результаты более сложных запросов. Более конкретно: если вид или вложенный SELECT используются в качестве исходной таблицы (справа от "FROM" в определении DATA), то mapserver затруднительно автоматически определить уникальный идентификатор для каждой строки и SRID для таблицы. Функция USING может предоставить эту информацию MapServer следующим образом:

```
DATA "the_geom FROM (
      SELECT
            table1.the_geom AS the_geom,
            table1.oid AS oid,
            table2.data AS data
      FROM table1
      LEFT JOIN table2
      ON table1.id = table2.id
      ) AS new_table USING UNIQUE oid USING SRID=-1"
```

USING UNIQUE <уникальный_id>
Mapserver требует уникального идентификатора для каждой строки, чтобы определить строку при создании карты запроса. Он, как правило, использует oid в качестве уникального идентификатора, но вилы и вложенные SELECT не имеют автоматически создаваемого столбца oid. Если нужно использовать функциональность запросов Mapserver, вам следует добавлять уникальные столбцы в свои виды и вложенные SELECT, и объявлять их в USING UNIQUE. Для этого вы можете, например, выбрать oid одной из таблиц, или любой другой столбец, который будет гарантированно уникальным в результирующем множестве.

Когда вы создаете карту запросов, предложение USING может быть полезным также и для простых предложений DATA. Ранее было рекомендовано создавать в используемых таблицах индекс по столбцу oid для ускорения выполнения карты запросов. Но с помощью пункта USING можно указать Mapserver-у использовать первичный ключ вашей таблицы в качестве идентификатора для карты запросов, и тогда отпадает необходимость создавать дополнительный индекс.
Замечание: "Запрос карты" ("Querying a Map") - действие, возникающее при щелчке на карту для запроса информации о данном месте. Не следует путать его с "картой запроса" для SQL-запроса в определении DATA.

USING SRID=<srid>
Чтобы вернуть Mapserver правильные данные, PostGIS должен знать, какая пространственная ссылочная система исопльзуется в геометриях. Обычно, эту информацию можно найти в базе данных PostGIS в таблице "geometry_columns", однако, это невозможно для таблиц, создающихся "на лету" вложенными SELECT и видами. Поэтому лучше использовать опцию USING SRID=, которая позволяет указывать корректный SRID в определении DATA.
Предупреждение: Парсер Mapserver для слоев PostGIS довольно примитивен и в некоторых местах является чувствительным к регистру. Проследите, чтобы все ключевые слова SQL и все ваши пункты USING были в верхнем регистре, а также, чтобы пункт USING UNIQUE предшествовал пункту USING SRID.

4.7.4. Примеры

Давайте начнем с простого примера работы с описанными выше таблицами. Рассмотрим следующее определение слоя Mapserver:

```
LAYER
        CONNECTIONTYPE postgis
        NAME "roads"
        CONNECTION "user=theuser password=thepass dbname=thedb
host=theserver"
        DATA "the_geom FROM roads"
        STATUS ON
        TYPE LINE
```

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

Этот слой отобразит все геометрии улиц из таблицы улиц как черные линии.

Теперь, при приближении к масштабу 1:100000, будем показывать только шоссе. Следующие два слоя позволят добиться этого эффекта:

```
LAYER
        CONNECTION "user=theuser password=thepass dbname=thedb
host=theserver"
        DATA "the_geom FROM roads"
        MINSCALE 100000
        STATUS ON
        TYPE LINE
        FILTER "road_type = 'highway'"
        CLASS
            COLOR 0 0 0
        END
      END
      LAYER
        CONNECTION "user=theuser password=thepass dbname=thedb
host=theserver"
        DATA "the_geom FROM roads"
        MAXSCALE 100000
        STATUS ON
        TYPE LINE
        CLASSITEM road_type
        CLASS
            EXPRESSION "highway"
            SIZE 2
            COLOR 255 0 0
        END
        CLASS
            COLOR 0 0 0
        END
END
```

Первый слой используется, когда запрашиваемый масштаб больше, чем 1:100000, и отображает только дороги типа "шоссе", как черные линии. Опция `FILTER` отфильтровывает для показа только дороги типа "шоссе" ("highway").

Второй слой используется, когда масштаб меньше 1:100000, и отобразит шоссе как две красные линии, а другие дороги, - как сплошные черные линии.

Итак, мы получили интересный результат, используя только функции MapServer, с довольно простым предложением `DATA`. Теперь предположим, что названия улиц по какой-то причине хранятся в другой таблице, и нам нужно присоединить ее, чтобы пометить улицы.

```
LAYER
CONNECTION "user=theuser password=thepass dbname=thedb host=theserver"
      DATA "the_geom FROM (SELECT roads.oid AS oid, roads.the_geom AS
the_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 oid USING SRID=-1"
        MAXSCALE 20000
        STATUS ON
        TYPE ANNOTATION
        LABELITEM name
        CLASS
                LABEL
                        ANGLE auto
                        SIZE 8
                        COLOR 0 192 0
                        TYPE truetype
                        FONT arial
                ENDl
        END
END
```

Этот слой примечаний добавляет зеленые подписи на все улицы в масштабе 1:20000 или меньше. Также здесь демонстрируется использование SQL вместе с определением `DATA`.

4.8. Клиенты Java (JDBC)

Java-клиенты имеют доступ к объектам "geometry" PostGIS в базе данных PostgreSQL одним из двух способов: непосредственно, в тестовом представлении, или с использованием расширения объектов JDBC, связанного с PostGIS. В порядке использования расширенных объектов, файл "postgis.jar" должен находиться в вашем CLASSPATH вместе с с пакетом драйвера JDBC "postgresql.jar".

```
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 {
                /*
                * Загрузка драйвера JDBC и установление соединения.
                */
                Class.forName("org.postgresql.Driver");
                String url = "jdbc:postgresql://localhost:5432/database";
                conn = DriverManager.getConnection(url, "postgres", "");
                /*
                * Добавляем геометрические типы в соединение.
                * Заметим, что вы должны установить специальное соединение с
pgsql
                * до вызова метода addDataType().
                */
```

```
        ((org.postgresql.Connection)conn).addDataType("geometry","org.post
gis.PGgeometry")
        ;

        ((org.postgresql.Connection)conn).addDataType("box3d","org.postgis
.PGbox3d");
                /*
                * Создаем объект запроса и выполняем запрос select.
                */
                Statement s = conn.createStatement();
                ResultSet r = s.executeQuery("select AsText(geom) as geom,id
from geomtable");
                while( r.next() ) {
                  /*
                  * Восстанавливаем геометрию как объект, какого то из
геометрических типов.
                  * Выводим его.
                  */
                  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();
          }
        }
        }
```

Объект "PGgeometry" является оберткой объекта, который содержит объект топологической геометрии (подкласс абстрактного класса "Geometry") одного из следующих типов: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon.

```
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());
                  }
          }
        }
```

Для расширенных объектов JavaDoc предоставляет справку по различным функциям доступа к данным в геометрических объектах.

4.9. Клиенты C (libpq)

...

4.9.1. Текстовые указатели

...

4.9.2. Бинарные указатели

...

5. Советы по производительности

5.1. Маленькие таблицы больших геометрий

5.1.1. Описание проблемы

Текущие версии PostgreSQL (включая 8.0) имеют слабый оптимизатор запросов для таблиц TOAST. Таблицы TOAST являются разновидностью "расширения" используемые для хранения больших значений (в смысле размера данных), которых нет в обычных данных (например длинные тексты, изображения или сложные геометрии, с множеством вершин). Подробную информацию можно получить здесь http://www.postgresql.org/docs/8.0/static/storage-toast.html.

Проблема появляется, если у вас таблица с большими геометриями, но с небольшим числом строк (как, например, в таблице, содержащей границы всех европейских стран в высоком разрешении). Такая таблица сама по себе мала, но использует много места в TOAST. В нашем примере таблица имеет всего 80 строк и всего 3 страницы данных, но таблица TOAST использует 8225 страниц.

Теперь рассмотрим запрос, в котором вы используете геометрический оператор && для поиска границы, которой соответствует только очень немногие из этих строк. Оптимизатор запроса видит, что таблица имеет всего 3 страницы и 80 строк. Он вычисляет, что последовательное сканирование по всей маленькой таблице быстрее использования индекса. Поэтому он принимает решение игнорировать индекс GIST. Обычно такое суждение правильно. Но не в нашем случае, так как оператор && будет получать с диска каждую геометрию для сравнения границ и, таким образом, считает все страницы TOAST.

Посмотреть, не в этом ли причина ошибки, можно с помощью команды postgresql "EXPLAIN ANALYZE". Более подробную информацию и технические детали вы можете прочитать в ветке листа рассылки postgres: http://archives.postgresql.org/pgsql-performance/2005-02/msg00030.php

5.1.2. Как обойти проблему

Пользователи PostgreSQL пытаются найти решение с помощью создания запросов, учитывающих TOAST. Вот два способа:

Первый способ - насильно заставить планировщик запросов использовать индекс. Перед выполнением запроса сообщите серверу "SET enable_seqscan TO off;". Эта команда насильно отменяет план запроса с возможным полным сканированием. Теперь индекс GIST будет использоваться как обычно. Но этот флаг будет действовать в течение всего коннекта и может стать причиной того, что планировщик запросов будет неверно работать в других случаях. Поэтому рекомендуем выполнить "SET enable_seqscan TO on;" после запроса.

Второй способ заключается в том, чтобы выполнить сканирование с той скоростью, на которую расчитывает планировщик. Этого можно добиться, создав дополнительный "кэширующий" столбец bbox, равносильный исходному. В нашем примере команды могут быть такими:

```
SELECT
addGeometryColumn('myschema','mytable','bbox','4326','GEOMETRY','2');

UPDATE mytable set bbox = Envelope(Force_2d(the_geom));
```

Теперь изменим запрос, использующий оператор && для bbox вместо geom_column:

```
SELECT geom_column

FROM mytable

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

Разумеется, что если вы изменяете или добавляете строки в mytable, вы должны синхронизировать bbox. Наиболее простой способ делать это - триггеры. Кроме того, вы можете изменить свое приложение так чтобы оно следило за корректностью столбца bbox или выполняло указанный выше запрос UPDATE после каждой модификации.

5.2. CLUSTER-изация геометрических индексов

Если таблица, обычно, только читается, и большинство запросов к ней используют единственный индекс, то к ней можно применить команду CLUSTER, предлагаемую PostgreSQL. Эта команда физически сортирует все строки данных в порядке, обусловленном индексом, что дает два преимущества в производительности. Во-первых, резко сокращается число обращений к данным таблицы при сканировании по диапазону индекса. Во-вторых, если вы чаще работаете с каким-то небольшим интервалом индексов, вы будете иметь более

эффективное кэширование, потому что строки данных распределены между немногими страницами данных. (Рекомендуем прочитать документацию по команде CLUSTER в мануале PostgreSQL.)

Однако, в настоящее время PostgreSQL не позволяет кластеризовать индексы PostGIS GIST, так как индексы GIST просто игнорируют значения NULL. При попытке вы получите соббщение об ошибке вроде этого:

```
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.
```

Если вы получили такое сообщение HINT, вы все же можете заставить это выражение работать, добавив в таблицу ограничение "not null":

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

ALTER TABLE
```

Разумеется, это не сработает, если вам необходимы значения NULL в геометрическом столбце. Учтите, что для создания ограничения вы должны использовать описанный выше метод. Использование ограничения CHECK, вроде такого: "ALTER TABLE blubb ADD CHECK (geometry is not null);", - работать не будет.

5.3. Избегайте изменений размерности

Иногда таблица содержит 3D- или 4D-данные, но для совместимости с OpenGIS следует всегда обращаться к ним с помощью функций asText() или asBinary(), которые возвращают только 2D-геометрии. Они делают это с помощью вызовов функции force_2d(), которая влечет существенные накладные расходы для больших геометрий. Во избежание этих накладных расходов стоит удалить эти лишние измерения раз и навсегда:

```
UPDATE mytable SET the_geom = force_2d(the_geom);

VACUUM FULL ANALYZE mytable;
```

Заметим, что если вы создаете свой столбец геометрии с помощью AddGeometryColumn(), будет создано ограничение на геометрические измерения. При необходимости вы можете удалить это ограничение. Но не забудьте, что после обновления записей в геометрических столбцах следует пересоздать ограничение.

В случае больших таблиц будет разумно осуществлять UPDATE небольшими порциями, ограничив UPDATE выражением WHERE с использованием первичного ключа или другого выполнимого условия. И запускать "VACUUM;" между UPDATE. Это существенно снижает потребность во временном дисковом пространстве. Кроме того, если вы имеете геометрии смешанной размерности, ограничение UPDATE посредством "WHERE dimension(the_geom)>2" приведет к неперезаписи геометрий, которые уже являются 2D.

6. Справочник PostGIS

Функции, приведенные ниже, скорее всего понадобятся пользователям PostGIS. Другие функции поддержки объектов PostGIS не используются повсеместно.

Замечание: PostGIS переходит на наименования по системе ориентированной на SQL-MM. В результате, большинство функций, которые вы знаете и любите, были переименованы с использованием стандартного пространственного префикса типа (ST - spatial type). Предыдущие функции остаются доступными, но не перечисляются в этом документе вместе с эквивалентными измененными функциями. Их использование в будущих релизах будет приостановлено.

6.1. Функции OpenGIS

6.1.1. Функции управления

AddGeometryColumn(varchar, varchar, varchar, integer, varchar, integer)
Синтаксис: AddGeometryColumn(<имя_схемы>, <имя_таблицы>, <имя_столбца>, <srid>, <тип>, <размерность>). Добавляет столбец геометрии в существующую таблицу атрибутов. `Имя_схемы` - название схемы таблицы (не используется для пре-схемы инсталляции PostgreSQL). `srid` должен быть целым числом совпадающим с одним из значений в таблице SPATIAL_REF_SYS. `Тип` должен быть строкой в верхнем регистре, сообщающей тип геометрии, такой как 'POLYGON' или 'MULTILINESTRING'.

DropGeometryColumn(varchar, varchar, varchar)
Синтаксис: DropGeometryColumn(<имя_схемы>, <имя_таблицы>, <имя_столбца>). Удаляет столбец геометрии указанной таблицы. Заметим, что имя_схемы должно совпадать с полем f_schema_name таблицы geometry_columns.

ST_SetSRID(geometry, integer)
Устанавливает SRID для геометрии в определенное целочисленное значение. Используется при построении охватов для запросов.

6.1.2. Функции геометрической связи

ST_Distance(geometry, geometry)
Возвращает декартово расстояние между двумя геометриями в заданных единицах. Не использует индексы.

ST_DWithin(geometry, geometry, float)
Возвращает истину, если геометрии находятся в пределах указанного расстояния одна от другой. Использует индексы если они есть.

ST_Equals(geometry, geometry)
Возвращает 1 (TRUE), если данные геометрии "пространственно равны". Использование этой функции предпочтительнее, чем использование '='. Выражение ST_Equals('LINESTRING(0 0, 10 10)','LINESTRING(0 0, 5 5, 10 10)') возвращает истину.
Выполняется с помощью модуля GEOS.
OGC SPEC s2.1.1.2

ST_Disjoint(geometry, geometry)
Возвращает 1 (TRUE), если геометрии "пространственно разделены".
Выполняется с помощью модуля GEOS.
Нельзя вызывать с GeometryCollection в качестве аргумента.
Замечание: это - "допустимая" версия, которая возвращает boolean, а не integer.
OGC SPEC s2.1.1.2 //s2.1.13.3 - a.Relate(b, 'FF*FF****')

ST_Intersects(geometry, geometry)
Возвращает 1 (TRUE), если геометрии "пространственно пересекаются".
Выполняется модулем GEOS.
Нельзя вызывать с GeometryCollection в качестве аргумента.
Вызов этой функции автоматически включает сравнение охватов, которое будет использовать любые индексы на указанных геометриях. Чтобы избежать использования индексов, пользуйте функцию _ST_Intersects.
Замечание: это - "допустимая" версия, которая возвращает boolean а не integer.
OGC SPEC s2.1.1.2 //s2.1.13.3 - Intersects(g1, g2) --> Not (Disjoint(g1, g2))

ST_Touches(geometry, geometry)
Возвращает 1 (TRUE), если геометрии "пространственно соприкасаются".
Выполняется модулем GEOS.
Нельзя вызывать с GeometryCollection в качестве аргумента.

Вызов этой функции автоматически включает сравнение охватов, которое будет использовать любые индексы на указанных геометриях. Чтобы избежать использования индексов, пользуйте функцию _ST_Touches.
Замечание: это - "допустимая" версия, которая возвращает boolean а не integer.
OGC SPEC s2.1.1.2 // s2.1.13.3- a.Touches(b) -> (I(a) intersection I(b) = {empty set}) and (a intersection b) not empty

ST_Crosses(geometry, geometry)
Возвращает 1 (TRUE), если геометрии "пространственно пересекаются".
Выполняется модулем GEOS.
Нельзя вызывать с GeometryCollection в качестве аргумента.
Вызов этой функции автоматически включает сравнение охватов, которое будет использовать любые индексы на указанных геометриях. Чтобы избежать использования индексов, пользуйте функцию _ST_Crosses.
Замечание: это - "допустимая" версия, которая возвращает boolean а не integer.
OGC SPEC s2.1.1.2 // s2.1.13.3 - a.Relate(b, 'T*T******')

ST_Within(geometry A, geometry B)
Возвращает 1 (TRUE), если геометрия A находится "пространственно внутри" геометрии B.
Выполняется модулем GEOS.
Нельзя вызывать с GeometryCollection в качестве аргумента.
Вызов этой функции автоматически включает сравнение охватов, которое будет использовать любые индексы на указанных геометриях. Чтобы избежать использования индексов, пользуйте функцию _ST_Within.
Замечание: это - "допустимая" версия, которая возвращает boolean а не integer.
OGC SPEC s2.1.1.2 // s2.1.13.3 - a.Relate(b, 'T*F**F***')

ST_Overlaps(geometry, geometry)
Возвращает 1 (TRUE), если геометрии "частично пространственно перекрываются".
Выполняется модулем GEOS.
Нельзя вызывать с GeometryCollection в качестве аргумента.
Вызов этой функции автоматически включает сравнение охватов, которое будет использовать любые индексы на указанных геометриях. Чтобы избежать использования индексов, пользуйте функцию _ST_Overlaps.
Замечание: это - "допустимая" версия, которая возвращает boolean а не integer.
OGC SPEC s2.1.1.2 // s2.1.13.3

ST_Contains(geometry A, geometry B)
Возвращает 1 (TRUE), если геометрия A "пространственно содержит" геометрию B.
Выполняется модулем GEOS.
Нельзя вызывать с GeometryCollection в качестве аргумента.
Вызов этой функции автоматически включает сравнение охватов, которое будет использовать любые индексы на указанных геометриях. Чтобы избежать использования индексов, пользуйте функцию _ST_Contains.
Замечание: это - "допустимая" версия, которая возвращает boolean а не integer.
OGC SPEC s2.1.1.2 // s2.1.13.3 - так же как within(geometry B, geometry A)

ST_Covers(geometry A, geometry B)
Возвращает 1 (TRUE), если ни одна точка геометрии B не находится вне геометрии A.
Объяснение, зачем нужна эта функция, можно прочесть здесь: http://lin-ear-th-inking.blogspot.com/2007/06/subtleties-of-ogc-covers-spatial.html
Вызов этой функции автоматически включает сравнение охватов, которое будет использовать любые индексы на указанных геометриях. Чтобы избежать использования индексов, пользуйте функциюn _ST_Covers.

ST_CoveredBy(geometry A, geometry B)
Возвращает 1 (TRUE), если ни одна точка геометрии A не находится вне геометрии B.
Объяснение, зачем нужна эта функция, можно прочесть здесь: http://lin-ear-th-inking.blogspot.com/2007/06/subtleties-of-ogc-covers-spatial.html

ST_Intersects(geometry, geometry)
Возвращает 1 (TRUE), если геометрии "пространственно пересекаются".
Выполняется модулем GEOS.
Нельзя вызывать с GeometryCollection в качестве аргумента.
Замечание: это - "допустимая" версия, которая возвращает boolean, а не integer.
OGC SPEC s2.1.1.2 // s2.1.13.3 - NOT disjoint(geometry, geometry)

ST_Relate(geometry, geometry, intersectionPatternMatrix)
Возвращает 1 (TRUE), если указанные геометрии пространственно связаны с другой геометрией, с помощью проверки пересечения между Interior, Boundary and Exterior двух геометрий, как указано в значении intersectionPatternMatrix.
Выполняется модулем GEOS.
Нельзя вызывать с GeometryCollection в качестве аргумента.
Замечание: это - "допустимая" версия, которая возвращает, boolean а не integer.

OGC SPEC s2.1.1.2 // s2.1.13.3

ST_Relate(geometry, geometry)
Возвращает DE-9IM - матрица девяти пересечений с расширенной размерностью (dimensionally extended nine-intersection matrix).
Выполняется модулем GEOS.
Нельзя вызывать с GeometryCollection в качестве аргумента.
Не описана в спецификации OGC, но подразумевается. Смотрите s2.1.13.2

6.1.3. Функции обработки геометрии

ST_Centroid(geometry)
Возвращает точку - центроид геометрии.
Вычисления будут более точными, если выполняются модулем GEOS (включается во время компиляции).

ST_Area(geometry)
Возвращает площадь геометрии, если она является полигоном или мультиполигоном.

ST_Length(geometry)
Возвращает длину кривой в соответствующей ей системе координат.
Синоним для length2d()
OGC SPEC 2.1.5.1

ST_PointOnSurface(geometry)
Возвращает точку, гарантированно лежащую на поверхности.
Выполняется с помощью модуля GEOS
OGC SPEC 3.2.14.2 и 3.2.18.2 -

ST_Boundary(geometry)
Возвращает замыкание комбинированной границы геометрии. Комбинированная граница (combinatorial boundary) определяется в разделе 3.12.3.2 спецификации OGC. Поскольку результат этой функции - замыкание и, следовательно, топологически замкнут, то результирующая граница может быть представлена с помощью геометрических примитивов, как обсуждается в спецификации OGC, в резделе 3.12.2.
Выполняется с помощью модуля GEOS
OGC SPEC s2.1.1.1

ST_Buffer(geometry, double, [integer])
Возвращает геометрию, все точки которой находятся на меньшем или равном расстоянии, чем заданное, от заданной геометрии. Расчеты производятся в пространственной системе координат заданной геометрии. Опциональный третий параметр задает число сегментов, используемых для аппроксимации четверти окружности (по умолчанию - 8).
Выполняется с помощью модуля GEOS
Нельзя вызывать с GeometryCollection в качестве аргумента.
OGC SPEC s2.1.1.3

ST_ConvexHull(geometry)
Возвращает геометрию, которая представляет собой конвексный полигон заданной геометрии.
Выполняется с помощью модуля GEOS
OGC SPEC s2.1.1.3

ST_Intersection(geometry, geometry)
Возвращает геометрию, которая представляет собой множество точек пересечения заданных геометрий.
Выполняется с помощью модуля GEOS
Нельзя вызывать с GeometryCollection в качестве аргумента.
OGC SPEC s2.1.1.3

ST_Shift_Longitude(geometry)
Считывает каждую точку/узел в каждом компоненте каждого объетк и, если ее долгота <0, добавляет к ней 360. Результат будет версией данных с диапазоном координат 0-360 вместо -180..180.

ST_SymDifference(geometry A, geometry B)
Возвращает геометрию, которая представляет собой множество точек геометрий A и B, которые не пересекаются. Это называется симметричной разностью потому что: ST_SymDifference(A,B) = ST_SymDifference(B,A).
Выполняется с помощью модуля GEOS
Нельзя вызывать с GeometryCollection в качестве аргумента.
OGC SPEC s2.1.1.3

ST_Difference(geometry A, geometry B)
Возвращает геометрию, которая представляет собой множество точек геометрии A не пересекающихся с геометрией B.
Выполняется с помощью модуля GEOS
Нельзя вызывать с GeometryCollection в качестве аргумента.
OGC SPEC s2.1.1.3

ST_Union(geometry, geometry)
Возвращает геометрию, которая представляет собой множество точек объединения геометрий.
Выполняется с помощью модуля GEOS
Нельзя вызывать с GeometryCollection в качестве аргумента.
Замечание: эта функция была переименована с "GeomUnion", так как "union" является зарезервированным словом SQL.
OGC SPEC s2.1.1.3

ST_Union(geometry set)
Возвращает геометрию, которая представляет собой множество точек объединения всех геометрий из указанного набора.
Выполняется с помощью модуля GEOS
Нельзя вызывать с GeometryCollection в качестве аргумента.
Явным образом не описана в спецификации OGC.

ST_MemUnion(geometry set)
То же, что и выше, но использует меньше памяти и больше процессорного времени.

6.1.4. Геометрические способы доступа

ST_AsText(geometry)
Возвращает WKT представление геометрии. Например: POLYGON(0 0,0 1,1 1,1 0,0 0)
OGC SPEC s2.1.1.1

ST_AsBinary(geometry)
Возвращает геометрию в "известном бинарном" ("well-known-binary") формате OGC, используя кодировку памяти сервера (endian encoding of the server), на котором запущена база данных. Это полезно в бинарном курсоре выбранных из базы данных, без конвертации их в строковое представление.
OGC SPEC s2.1.1.1 - смотрите также asBinary(<geometry>,'XDR') и asBinary(<geometry>,'NDR')

ST_SRID(geometry)
Возвращает целочисленный номер SRID системы пространственных координат данной геометрии.
OGC SPEC s2.1.1.1

ST_Dimension(geometry)
Размерность геометрического объекта, меньшая или равная размерности координат. Согласно OGC SPEC s2.1.1.1 возвращает 0 для точек, 1 - для линий, 2 - для полигонов и наибольшую размерность компонентов для GEOMETRYCOLLECTION.

```
select dimension('GEOMETRYCOLLECTION(LINESTRING(1 1,0 0),POINT(0 0)');
dimension
-----------
1
```

ST_Envelope(geometry)
Возвращает правильную геометрию (POINT, LINESTRING или POLYGON) представляющую охват геометрии. Производит уменьшение размерности (вертикальные линии, точки) с возващением геометрии размерностью ниже чем POLYGON.
OGC SPEC s2.1.1.1 - Минимальная поверхность геометрии, возвращаемая как геометрия. Полигон определяется как множество точек, ограниченных ((MINX, MINY), (MAXX, MINY), (MAXX, MAXY), (MINX, MAXY), (MINX, MINY)).
Замечание: В PostGIS добавлены координаты Zmin/Zmax.

ST_IsEmpty(geometry)
Возвращает 1 (TRUE), если геометрия является пустой. Если true, то геометрия представлена пустым множеством точек, т.е. GEOMETRYCOLLECTION(EMPTY).
OGC SPEC s2.1.1.1

ST_IsSimple(geometry)
Возвращает 1 (TRUE), если геометрия не содержит аномальные геометрические точки. Под аномальными понимаются такие точки, как точки самопересечения или самокасания.
Выполняется с помощью модуля GEOS
OGC SPEC s2.1.1.1

ST_IsClosed(geometry)
Возвращает истину, если начальная и конечная точки геометрии совпадают.

ST_IsRing(geometry)
Возвращает 1 (TRUE), если заданная кривая является замкнутой (StartPoint ( ) = EndPoint ( )) и простой (не проходит через одну точку более одного раза).
Выполняется с помощью GEOS
OGC spec 2.1.5.1

ST_NumGeometries(geometry)
Если геометрия является GEOMETRYCOLLECTION (или MULTI*), возвращает число геометрий. В противном случае возвращает NULL.

ST_GeometryN(geometry,int)
Возвращает N-ную геометрию, если заданная геометрия есть GEOMETRYCOLLECTION, MULTIPOINT, MULTILINESTRING или MULTIPOLYGON. В других случаях возвращает NULL.
Замечание: Нумерация начинается с 1, как это определено спецификацией OGC в версии 0.8.0. В предыдущих версиях, вместо этого, использовалась нумерация начинающаяся с 0.

ST_NumPoints(geometry)
Находит и возвращает число точек в первой линии заданной геометрии. Возвращает NULL, если геометрия не содержит линий.

ST_PointN(geometry,integer)
Возвращает N-ую точку первой линии в заданной геометрии. Возвращает NULL, если геометрия не содержит линий.
Замечание: Нумерация начинается с 1, как это определено спецификацией OGC в версии 0.8.0. В предыдущих версиях, вместо этого, использовалась нумерация начинающаяся с 0.

ST_ExteriorRing(geometry)
Возвращает внешнюю дугу (exterior ring) полигональной геометрии. Возвращает NULL, если геометрия не является полигоном.

ST_NumInteriorRings(geometry)
Возвращает число вписанных дуг (interior rings) в первый полигон данной геометрии. Возвращает NULL, если геометрия не содержит полигонов.

ST_NumInteriorRing(geometry)
Синоним для NumInteriorRings(geometry). Спецификация OpenGIS двусмыслена в плане наименования этой функции. Поэтому мы поддерживаем оба написания.

ST_InteriorRingN(geometry,integer)
Возвращает N-ую вписанную окружность в полигональную геометрию. Возвращает NULL, если геометрия не является полигоном, или имеет меньше N окружностей.
Замечание: Нумерация начинается с 1, как это определено спецификацией OGC в версии 0.8.0. В предыдущих версиях, вместо этого, использовалась нумерация начинающаяся с 0.

ST_EndPoint(geometry)
Возвращает последнюю точку геометрии LineString.

ST_StartPoint(geometry)
Возвращает первую точку геометрии LineString.

GeometryType(geometry)
Возвращает тип геометрии как строку. Например: 'LINESTRING', 'POLYGON', 'MULTIPOINT' и т.п.
OGC SPEC s2.1.1.1 - возвращает название подтипа геометрии, в который установлен данный представитель геометрии. Название подтипа геометрии возвращается как строка.
Замечание: Кроме прочего эта функция указывает, имеет ли геометрия измерение M, возвращая строку с 'POINTM'.

ST_GeometryType(geometry)
Возвращает тип геометрии как строку: Например: 'Linestring', 'Polygon' и т.п. Эта функция отличается от GeometryType(geometry) регистром возвращаемой строки, а также тем, что не показывает, содержит ли геометрия M.

ST_X(geometry)
Возвращает координату X точки. Геометрия должна быть точкой.

ST_Y(geometry)
Возвращает координату Y точки. Геометрия должна быть точкой.

ST_Z(geometry)
Возвращает координату Z точки, или NULL, если этой координаты нет. Геометрия должна быть точкой.

ST_M(geometry)
Возвращает координату M точки, или NULL, если этой координаты нет. Геометрия должна быть точкой.
Замечание: Эта функция (все еще) не является частью спецификации OGC, но необходима для завершения списка функций извлечения координат точки.

6.1.5. Геометрические конструкторы

ST_GeomFromText(text,[<srid>])
Создает геометрию из WKT с указанным SRID.
OGC SPEC 3.2.6.2 - необязательный SRID является отклонением от стандарта.

ST_PointFromText(text,[<srid>])
Создает геометрию из WKT с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.6.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKT не является точкой.

ST_LineFromText(text,[<srid>])
Создает геометрию из WKT с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.6.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKT не является линией.

ST_LinestringFromText(text,[<srid>])
Создает геометрию из WKT с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
Является отклонением от стандарта.
Выдает ошибку, если WKT не является линией.

ST_PolyFromText(text,[<srid>])
Создает геометрию из WKT с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.6.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKT не является полигоном.

ST_PolygonFromText(text,[<srid>])
Создает геометрию из WKT с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
Является отклонением от стандарта.
Выдает ошибку, если WKT не является полигоном.

ST_MPointFromText(text,[<srid>])
Создает геометрию из WKT с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.6.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKT не является MULTIPOINT.

ST_MLineFromText(text,[<srid>])
Создает геометрию из WKT с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.6.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKT не является MULTILINESTRING.

ST_MPolyFromText(text,[<srid>])
Создает геометрию из WKT с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.6.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKT не является MULTIPOLYGON-ом

ST_GeomCollFromText(text,[<srid>])
Создает геометрию из WKT с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.6.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKT не является GEOMETRYCOLLECTION-ом

ST_GeomFromWKB(bytea,[<srid>])
Создает геометрию из WKB с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.6.2 - необязательный SRID является отклонением от стандарта.

ST_GeometryFromWKB(bytea,[<srid>])
Создает геометрию из WKB с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.7.2 - необязательный SRID является отклонением от стандарта.

ST_PointFromWKB(bytea,[<srid>])
Создает геометрию из WKB с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.7.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKB не является POINT.

ST_LineFromWKB(bytea,[<srid>])
Создает геометрию из WKT с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.7.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKB не является LINESTRING.

ST_LinestringFromWKB(bytea,[<srid>])
Создает геометрию из WKB с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
Является отклонением от стандарта.
Выдает ошибку, если WKB не является LINESTRING-ом.

ST_PolyFromWKB(bytea,[<srid>])

Создает геометрию из WKB с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.7.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKB не является POLYGON-ом

ST_PolygonFromWKB(bytea,[<srid>])
Создает геометрию из WKB с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
Является отклонением от стандарта.
Выдает ошибку, если WKB не является POLYGON.

ST_MPointFromWKB(bytea,[<srid>])
Создает геометрию из WKB с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.7.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKB не является MULTIPOINT.

ST_MLineFromWKB(bytea,[<srid>])
Создает геометрию из WKB с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.7.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKB не является MULTILINESTRING.

ST_MPolyFromWKB(bytea,[<srid>])
Создает геометрию из WKB с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.7.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKB не является MULTIPOLYGON.

ST_GeomCollFromWKB(bytea,[<srid>])
Создает геометрию из WKB с указанным SRID. Если SRID не указан, используется -1 по умолчанию.
OGC SPEC 3.2.7.2 - необязательный SRID является отклонением от стандарта.
Выдает ошибку, если WKB не является GEOMETRYCOLLECTION.

ST_BdPolyFromText(text WKT, integer SRID)
Создает полигон, заданный произвольной коллекцией замкнутых кривых, заданных текстовым представлением MultiLineString.
Выдает ошибку, если WKT не является MULTILINESTRING. Выдает ошибку, если вывод является MULTIPOLYGON; используйте, в этом случае, BdMPolyFromText или смотрите BuildArea(), чтобы разобраться в специфике PostGIS.
OGC SFSQL 1.1 - 3.2.6.2
Доступно с 1.1.0. Требует GEOS >= 2.1.0.

ST_BdMPolyFromText(text WKT, integer SRID)
Конструирует мультиполигон, заданный произвольной коллекцией замкнутых кривых, заданных текстовым представлением MultiLineString.
Выдает ошибку, если WKT не является MULTILINESTRING. Выдает MULTIPOLYGON даже если вывод единственный полигон. Если же вам нужен именно POLYGON, то используйте BdPolyFromText или смотрите BuildArea(), чтобы разобраться в специфике PostGIS.
OGC SFSQL 1.1 - 3.2.6.2
Доступно с 1.1.0. Требует GEOS >= 2.1.0.

6.2.1. Функции управления

DropGeometryTable([<schema_name>], <table_name>)
Удаляет таблицу и все связанные с ней столбцы геометрии.
Замечание: если установленный pgsql поддерживает схемы, когда схема не предоставлена, используется current_schema().

UpdateGeometrySRID([<schema_name>], <table_name>, <column_name>, <srid>)
Обновляет SRID всех объектов в столбце геометрии, обновляет ограничения и ссылки в этом столбце.
Замечание: если установленный pgsql поддерживает схемы, когда схема не предоставлена, используется current_schema().

update_geometry_stats([<table_name>, <column_name>])
Обновляет статистику пространственных таблиц для использования планировщиком запросов. Кроме того, вам понадобится "VACUUM ANALYZE [table_name] [column_name]" для завершения процесса сбора статистики.
Замечание: начиная с PostgreSQL 8.0 сбор статистики автоматически выполняется при запуске "VACUUM ANALYZE".

postgis_version()
Возвращает версию PostGIS и опции, с которыми он был скомпилирован.
Замечание: До версии 1.1.0 это была процедурная функция, возможно, возвращающая неправильную информацию (в случае неполного апгрейда базы данных).

postgis_lib_version()
Возвращает номер версии библиотеки PostGIS.
Присутствует с 0.9.0

postgis_lib_build_date()
Возвращает дату сборки библиотеки PostGIS.
Присутствует с 1.0.0RC1

postgis_script_build_date()
Возвращает дату создания скриптов PostGIS.
Присутствует с 1.0.0RC1.

postgis_scripts_installed()
Возвращает версию скриптов PostGIS, установленных в базе данных.
Замечание: Если вывод этой функции не совпадает с выводом postgis_scripts_released(), то, вероятно, вы не выполнили должным образом апгрейд существующей базы. Смотрите подробности в разделе Обновление.
Присутствует с 0.9.0.

postgis_scripts_released()
Возвращает номер версии скрипта lwpostgis.sql, предоставляемого с установкой библиотеки PostGIS.
Замечание: Начиная с версии 1.1.0 эта функция возвращает то же значение, что и postgis_lib_version(). Оставлена для обратной совместимости.
Присутствует с 0.9.0.

postgis_geos_version()
Возвращает номер версии библиотеки GEOS, или NULL, если поддержка GEOS не включена.
Присутствует с 0.9.0

postgis_jts_version()
Возвращает номер версии библиотеки JTS, или NULL, если поддержка JTS не включена.
Присутствует с 1.1.0

postgis_proj_version()
Возвращает номер версии библиотеки PROJ4, или NULL, если поддержка PROJ4 не включена.
Присутствует с 0.9.0

postgis_uses_stats()
Возвращает true, если включено использование STATS, и false в противном случае.
Присутствует с 0.9.0

postgis_full_version()
Сообщает полную версию PostGIS и информацию о конфигурации сборки.
Присутствует с 0.9.0

6.2.2. Операторы

A &< B
Оператор "&<" возвращает true, если охват A частично совпадает или находится слева от охвата B.

A &> B
Оператор "&>" возвращает true, если охват A частично совпадает или находится справа от охвата B.

A << B
Оператор "<<" возвращает true, если охват A находится строго слева от охвата B.

A >> B
Оператор ">>" возвращает true, если охват A находится строго справа от охвата B.

A &<| B
Оператор "&<|" возвращает true, если охват A частично совпадает или находится ниже охвата B.

A |&> B
Оператор "|&>" возвращает true, если охват A частично совпадает или находится выше охвата B.

A <<| B
Оператор "<<|" возвращает true, если охват A находится строго ниже охвата B.

A |>> B
Оператор "|>>" возвращает true, если охват A находится строго выше охвата B.

A ~= B
Оператор "~=" - это оператор "то же самое". Он проверяет, являются ли обе части геометрически эквивалентными. Так, если A и B совпадают "вершина-в-вершину", то оператор вернет true.

A @ B
Оператор "@" возвращает true, если охват A полностью содержит охват B.

A ~ B
Оператор "~" возвращает true, если охват A полностью содержится в охвате B.

A && B
Оператор "&&" является оператором "частичного перекрытия" ("overlaps"). Если охват A частично перекрывается охватом B, то оператор вернет true.

6.2.3. Функции измерения

ST_Area(geometry)
Возвращает площадь геометрии, являющейся полигоном или мультиполигоном.

ST_distance_sphere(point, point)
Возвращает линейное расстояние в метрах между двумя точками (широта/долгота). Вычисление производится для сферы радиусом 6370986 метров (Земля). Работает быстрее, чем distance_spheroid(), но менее точно. Выполняется только для точек.

ST_distance_spheroid(point, point, spheroid)
Возвращает линейное расстояние в метрах между двумя точками (широта/долгота) на заданном сфероиде. Смотрите определение сфероида, данное для length_spheroid(). В настоящее время выполняется только для точек.

ST_length2d(geometry)
Возвращает 2-мерную длину (2-dimensional length) геометрии, если она является линией или мульти-линией.

ST_length3d(geometry)
Возвращает 3-мерную длину (3-dimensional length) геометрии, если она является линией или мульти-линией.

ST_length_spheroid(geometry,spheroid)
Вычисляет длину геометрии на эллипсоиде. Может быть полезна, если координатами геометрии являются широта/долгота и длину желательна вычислить без перепроектирования без репроекции. Эллипсоид является особым тип базы данных и может быть сконструирован следующим способом:

```
SPHEROID[<NAME>,<SEMI-MAJOR AXIS>,<INVERSE FLATTENING>]
```

Например:

```
SPHEROID["GRS_1980",6378137,298.257222101]
```

Пример вычисления должен быть таким:

```
SELECT length_spheroid( geometry_column,
            'SPHEROID["GRS_1980",6378137,298.257222101]' )
            FROM geometry_table;
```

ST_length3d_spheroid(geometry,spheroid)
Вычисляет длину геометрии на эллипсоиде с учетом высоты. Это аналогично length_spheroid, но учитывает вертикальные координаты (выраженные в тех же единицах, что и оси сфероида), и используется для расчета дополнительного расстояния, которое добавляется вертикальным смещением.

ST_distance(geometry, geometry)
Возвращает наименьшее расстояние между двумя геометриями.

ST_max_distance(linestring,linestring)
Возвращает наибольшее расстояние между двумя линиями.

ST_perimeter(geometry)
Возвращает 2-мерный периметр геометрии, если она является полигоном или мульти-полигоном.

ST_perimeter2d(geometry)
Возвращает 2-мерный периметр геометрии, если она является полигоном или мульти-полигоном.

ST_perimeter3d(geometry)
Возвращает 3-мерный периметр геометрии, если она является полигоном или мульти-полигоном.

ST_azimuth(geometry, geometry)
Возвращает азимут сегмента, определенного заданными точечными геометриями, или NULL, если обе точки совпадают. Возвращает значение в радианах.
Присутствует с 1.1.0

6.2.4. Геометрический вывод

ST_AsBinary(geometry,{'NDR'|'XDR'})
Возвращает геометрию в "известном-бинарном" ("well-known-binary") формате OGC, как набор байтов с использованием little-endian (NDR) или big-endian (XDR) порядка байтов. Это используется в бинарном курсоре для вывода данных из базы без преобразования их к строковому представлению.

ST_AsEWKT(geometry)
Возвращает геометрию в формате EWKT (как текст).

ST_AsEWKB(geometry, {'NDR'|'XDR'})
Возвращает геометрию в формате EWKB, как набор байтов с использованием little-endian (NDR) или big-endian (XDR) порядка байтов.

ST_AsHEXEWKB(geometry, {'NDR'|'XDR'})
Возвращает геометрию в формате HEXEWKB, как текстс использованием little-endian (NDR) или big-endian (XDR) порядка байтов.

ST_AsSVG(geometry, [rel], [precision])
Возвращает геометрию как данные SVG. При использовании 1 в качестве второго аргумента, данные выдаются в терминах относительного движения. По умолчанию, (0) выдаются данные в терминах абсолютного движения. Третий аргумент может быть использован для сокращения максимума чисел после запятой в выводе (по умолчанию - 15). Точечные геометрии будут выданы как cx/cy, когда аргумент 'rel' равен 0, и как x/y, когда 'rel' равен 1.

ST_AsGML([version], geometry, [precision])
Возвращает геометрию, как элемент GML. Параметр version, если указан, должен быть либо 2, либо 3. Если параметр version не указан, то используется значение по умолчанию 2. Третий аргумент может быть использован для ограничения максимума числа значащих цифр в выводе (по умолчанию - 15).

ST_AsKML(geometry, [precision])
Возвращает геометрию как элемент KML. Второй аргумент может быть использован для ограничения максимума числа значащих цифр в выводе (по умолчанию - 15).

ST_AsGeoJson([version], geometry, [precision], [options])
Возвращает геометрию как элемент GeoJson. (См. GeoJson specifications 1.0). Поддерживаются и 2D и 3D геометрии. GeoJson поддерживает только тип геометрии SFS 1.1 (например не поддерживаются кривые).

Параметр version если указан должен быть равен 1.

Третий параметр может быть использован для уменьшения количества значимых разрядов чисел (по умолчанию 15).

Последний параметр 'options' может быть использован для добавления Bbox или Crs в вывод GeoJSON:
0: нет опции (значение по умолчанию)
1: GeoJson CRS
2: GeoJson Bbox

3: Both GeoJson Bbox and CRS
Формат GeoJson CRS: auth_name:auth_srid из spatial_ref_sys table (например EPSG:4326).

Присутствует с : 1.3.4

6.2.5. Геометрические конструкторы

ST_GeomFromEWKT(text)
Создает геометрию из EWKT.

ST_GeomFromEWKB(bytea)
Создает геометрию из EWKB.

ST_MakePoint(<x>, <y>, [<z>], [<m>])
Создает точку 2d-, 3dz- или 4d-геометрии.

ST_MakePointM(<x>, <y>, <m>)
Создает точку 3dm-геометрии.

ST_MakeBox2D(<LL>, <UR>)
Создает BOX2D, определенный с помощью заданных точечных геометрий.

ST_MakeBox3D(<LLB>, <URT>)
Создает BOX3D, определенный с помощью заданных точечных геометрий.

ST_MakeLine(geometry set)
Создает линию из множества точечных геометрий. При аггрегировании вы можете использовать вложенный SELECT для упорядочения точек перед созданием линии.

ST_MakeLine(geometry, geometry)
Создает Linestring с помощью двух заданных точечных геометрий.

ST_LineFromMultiPoint(multipoint)
Создает Linestring с помощью геометрии MultiPoint.

ST_MakePolygon(linestring, [linestring[]])
Создает полигон, сформированный с помощью данного остова и массива дыр. Вы можете создать геометрический массив с помощью Accum. Вводимые геометрии должны быть замкнутыми ломаными (смотрите IsClosed и GeometryType).

ST_BuildArea(geometry)
Создает полигональную геометрию, сформированную с помощью линий, составляющих заданную геометрию. Возвращаемый тип может быть полигоном или мультиполигоном в зависимости от ввода. Если вводимые линии не образуют полигона, возвращается NULL.
Смотрите также BdPolyFromText и BdMPolyFromText - обертки этой функции со стандартным интерфейсом OGC.
Поддерживается с 1.1.0. Требует GEOS >= 2.1.0.

ST_Polygonize(geometry set)
Аггрегировать. Создает GeometryCollection, содержащий возможные полигоны, сформированные из составных линий указанного множества геометрий.
Поддерживается с 1.0.0RC1. Требует >= 2.1.0.

ST_Collect(geometry set)
Функция возвращает GEOMETRYCOLLECTION или MULTI-объект для множества геометрий. Функция collect() является "агрегирующей" в терминологии PostgreSQL. Это означает, что она оперирует списком данных, наподобие того, как это делают функции sum() и mean(). Например, "SELECT COLLECT(GEOM) FROM GEOMTABLE GROUP BY ATTRCOLUMN" вернет отдельный GEOMETRYCOLLECTION для каждого уникального значения ATTRCOLUMN.

ST_Collect и ST_Union часто взаимозаменяемы. ST_Collect на обычных запросах на порядок быстрее чем ST_Union потому что он не пытается растворить границы. Он просто сводит единичные геометрии в MULTI и MULTI или смешанные геометрии в Geometry Collections. К сожалению, коллекции не очень хорошо поддерживаются инструментарием ГИС. Чтобы избежать возвращения ST_Collect коллекций геометрии при сборе MULTI геометрий, можно использовать следующий подход, который использует ST_Dump для раскрытия коллекции в единичные геометрии и объединение их в группы:

```
Thread ref: http://postgis.refractions.net/pipermail/postgis-users/2008-June/020331.html

SELECT stusps,
       ST_Multi(ST_Collect(f.the_geom)) as singlegeom
       FROM (SELECT stusps, (ST_Dump(the_geom)).geom As the_geom
```

```
                                    FROM
                                    somestatetable ) As f
        GROUP BY stusps
```

ST_Collect(geometry, geometry)
Эта функция возвращает геометрию, которая является коллекцией двух введенных геометрий. Вывод будет иметь тип MULTI* или GEOMETRYCOLLECTION.

ST_Dump(geometry)
Это функция возвращает множество (SRF - set-returning function). Она возвращает набор строк geometry_dump, сформированных в виде геометрии (geom) и массива целых чисел (path). Если введенная геометрия имеет простой тип (POINT, LINESTRING, POLYGON), будет возвращена единственная запись с пустым массивом "path" и введеной геометрией в качестве "geom". Если введенная геометрия является коллекцией или MULTI, то будет возвращено по записи на каждую компоненту коллекции, а "path" будет указывать позицию компоненты внутри коллекции.
Эта функция полезна для раскрытия геометрий. Она является противоположной функции GROUP BY в том, что она создает новые строки. Например, она может быть использована для раскрытия MULTIPOLYGONS в POLYGONS.

```
SELECT sometable.*, (ST_Dump(the_geom)).geom As the_geom
FROM somestatetable
```

Поддерживается с PostGIS 1.0.0RC1. Требует PostgreSQL 7.3 или выше.

ST_DumpRings(geometry)
Это функция возвращает множество (SRF - set-returning function). Она возвращает набор строк geometry_dump, сформированных в виде геометрии (geom) и массива целых чисел (path). Поле "path" является индексом полигонального круга, содержащим единственный элемент: 0 для остова, номер дырки для дырок. Поле "geom" содержит соответствующую дугу, как полигон.
Поддерживается с PostGIS 1.1.3. Требует PostgreSQL 7.3 или выше.

6.2.6. Геометрические редакторы

ST_AddBBOX(geometry)
Добавляет к геометрии охват. Это приведет к ускорению запросов, основанных на охватах, но увеличивает размер геометрий.

ST_DropBBOX(geometry)
Удаляет из геометрии кэш охватов. Это уменьшает размер геометрии, но замедляет запросы, использующие охваты.

ST_AddPoint(linestring, point, [<position>])
Добавляет точку в линию после точки <pos> (нумерация начинается с 0). Третим параметром можно пренебречь или установить в -1, что одно и то же.

ST_RemovePoint(linestring, offset)
Удаляет точку линии. Offset начинается с 0.
Поддерживается с 1.1.0.

ST_SetPoint(linestring, N, point)
Заменяет точку N линии на заданную точку. Нумерация начинается с 0.
Поддерживается с 1.1.0.

ST_Force_collection(geometry)
Конвертирует геометрию в GEOMETRYCOLLECTION. Может быть использована для упрощения представления WKB.

ST_Force_2d(geometry)
Приводит геометрию в "2-мерный режим", так чтобы весь вывод был представлен только в координатах X и Y. Это может быть полезно для OGC-совместимого вывода (поскольку OGC определяет только 2-D геометрии).

ST_Force_3dz(geometry), ST_Force_3d(geometry)
Переводит геометрии в режим XYZ.

ST_Force_3dm(geometry)
Переводит геометрии в режим XYM.

ST_Force_4d(geometry)
Переводит геометрии в режим XYZM.

ST_Multi(geometry)
Возвращает геометрию как геометрию MULTI*. Если данная геометрия уже является MULTI*, она будет возвращена неизменившейся.

ST_Transform(geometry,integer)

Возвращает новую геометрии, все координаты которой трансформированы соответственно SRID, указанному как целочисленный параметр. Назначенный SRID должен существовать в таблице `SPATIAL_REF_SYS`.

ST_Affine(geometry, float8, float8, float8, float8, float8, float8, float8, float8, float8, float8, float8, float8)
Осуществляет 3d аффинную трансформацию геометрии. Команда:

```
Affine(geom, a, b, c, d, e, f, g, h, i, xoff, yoff, zoff)
```

представляет матрицу трансформации:

```
/ a  b  c  xoff \
| d  e  f  yoff |
| g  h  i  zoff |
\ 0  0  0     1 /
```

Вершины будут трансформированы следующим образом:

```
x' = a*x + b*y + c*z + xoff
y' = d*x + e*y + f*z + yoff
z' = g*x + h*y + i*z + zoff
```

Поддерживается с 1.1.2.

ST_Affine(geometry, float8, float8, float8, float8, float8, float8)
Осуществляет 2d аффинную трансформацию геометрии. Команда:

```
Affine(geom, a, b, d, e, xoff, yoff)
```

представляет матрицу трансформации:

```
/  a  b  0  xoff  \       /  a  b  xoff  \
|  d  e  0  yoff  | rsp.  |  d  e  yoff  |
|  0  0  1     0  |       \  0  0     1  /
\  0  0  0     1  /
```

Вершины будут трансформированы следующим образом:

```
x' = a*x + b*y + xoff
y' = d*x + e*y + yoff
z' = z
```

Этот метод является частностью 3D-метода, описанного выше.
Поддерживается с 1.1.2.

ST_Translate(geometry, float8, float8, float8)
Переводит геометрию в новое местоположение с помощью числовых параметров смещения. Пример: translate(geom, X, Y, Z).

ST_Scale(geometry, float8, float8, float8)
Масштабирует геометрию в новый размер путем умножения координат на соответствующие параметры. Например: scale(geom, Xfactor, Yfactor, Zfactor).
Поддерживается с 1.1.0.

ST_RotateZ(geometry, float8), ST_RotateX(geometry, float8), ST_RotateY(geometry, float8)
Поворачивает геометрию вокруг оси Z, X или Y на угол, заданный в радианах. Согласно правилу правой руки.
Поддерживается с 1.1.2.

ST_TransScale(geometry, float8, float8, float8, float8)
Сначала смещает геометрию с помощью первых двух значений, а потом масштабирует ее с помощью вторых двух чисел. Работает только с 2D. Вызов `transscale(geom, X, Y, XFactor, YFactor)` внутри приводит к вызову `affine(geom, XFactor, 0, 0, 0, YFactor, 0, 0, 0, 1, X*XFactor, Y*YFactor, 0)`.
Поддерживается с 1.1.0.

ST_Reverse(geometry)
Возвращает геометрию с противоположным порядком вершин.

ST_ForceRHR(geometry)
Приводит коллекцию полигонов в соответствие с "правилом правой руки".

ST_Simplify(geometry, tolerance)
Возвращает "упрощенную" версию данной геометрии с помощью алгоритма Дугласа-Пойкера (Douglas-Peuker). На самом деле будет работать только (мульти)линии и (мульти)полигоны, но вы можете безбоязненно применять ее к любым типам геометрий. Так как упрощение осуществляется пообъектно

(object-by-object), вы можете использовать эту функцию и для GeometryCollection. Заметим, что возвращенная геометрия может не быть простой (смотрите IsSimple).

ST_SnapToGrid(geometry, originX, originY, sizeX, sizeY), ST_SnapToGrid(geometry, sizeX, sizeY), ST_SnapToGrid(geometry, size)
Прищелкивает все точки введенной геометрии к гриду, описанному координатами начала и размерами ячеек. Удаляет точки, попадающие в те же ячейки и, в конечном счете, возвращает NULL, если выведенных точек недостаточно, чтобы определить геометрию данного типа. Свернутые геометрии удаляются из коллекции.
Замечание: Возвращаемая геометрия может перестать быть простой (смотрите IsSimple).
Замечание: До релиза 1.1.0 эта функция всегда возвращала 2d-геометрию. Начиная с 1.1.0 возвращаемая геометрия будет иметь ту же размерность, что и введеная. Размерности высших порядков остаются нетронутыми. Используйте версию, берущую второй геометрический аргумент, чтобы определить все измерения грида.
Поддерживается с 1.0.0RC1.

ST_SnapToGrid(geometry, geometry, sizeX, sizeY, sizeZ, sizeM)
Прищелкивает все точки введенной геометрии к гриду, описанному началом (второй аргумент, должен быть точкой) и размерами ячеек. Если вы не хотите осуществлять прищелкивание, укажите любой размер равным 0.
Поддерживается с 1.1.0.

ST_Segmentize(geometry, maxlength)
Возвращает модифицированную геометрию без секторов, с длиной больше указанного размера. Интерполированные точки будут иметь значения Z и M (если они нужны), установленные в 0. Вычисление размеров производится только в 2d.

ST_LineMerge(geometry)
Возвращает линию или линии, сформированную(ые) посредством соединения линий, составляющих ввод.
Поддерживается с 1.1.0. Требует GEOS >= 2.1.0

6.2.7. Линейные ссылки

ST_line_interpolate_point(linestring, location)
Возвращает точки, интерполированные вдоль линии. Первый аргумент должен иметь тип LINESTRING. Второй аргумент является float8 между 0 и 1, представляющим дробную часть всей 2d-длины где должны быть раположены точки.
Смотрите line_locate_point(), чтобы вычислить местоположение линии, самое близкое к точке.
Замечание: Начиная с релиза 1.1.1 эта функция также интерполирует значения M и Z (если они есть), тогда как предыдущие релизы устанавливали их в 0.0.
Поддерживается с 0.8.2.

ST_line_substring(linestring, start, end)
Возвращает линию как часть первого аргумента, начинающегося и заканчивающегося заданными долями. Второй и третий аргумент должны быть значениями float8 между 0 и 1.
Если 'start' и 'end' имеют одинаковые значения, это эквивалентно line_interpolate_point().
Смотрите line_locate_point() про вычисление линии местоположения, ближайшей к точке.
Замечание: Начиная с релиза 1.1.1 эта функция также интерполирует значения M и Z (если они есть), тогда как предыдущие релизы устанавливали их в неопределенное значение.
Поддерживается с 1.1.0.

ST_line_locate_point(LineString, Point)
Возвращается float между 0 и 1, представляющий местонахождение вершины линии ближайшей к указанной точке, как отношение к общей длине 2d-линии.
Вы можете использовать полученное местоположение для извлечения точки (line_interpolate_point) или части линии (line_substring).
Поддерживается с 1.1.0.

ST_locate_along_measure(geometry, float8)
Возвращает значение производной геометрии, элементы которой соответствуют указанной мере. Полигональные элементы не поддерживаются.
Семантика специфицирована в ISO/IEC CD 13249-3:200x(E) - Текст продолжен на CD редакционного заседания.
Поддерживается с 1.1.0.

ST_locate_between_measures(geometry, float8, float8)
Возвращает производную геометрическую коллекцию с элементами, которые включают в себя указанный диапазон мер. Полигональные элементы не поддерживаются.
Семантика определена в ISO/IEC CD 13249-3:200x(E) - Текст продолжен на CD редакционного заседания.
Поддерживается с 1.1.0.

6.2.8. Разное

ST_Summary(geometry)
Возвращает текстовое резюме содержания геометрии.

ST_box2d(geometry)
Возвращает BOX2D, представленный максимальным охватом геометрии.

ST_box3d(geometry)
Возвращает BOX3D, представленный максимальным охватом геометрии.

ST_extent(geometry set)
Функция extent() является "агрегирующей" функцией в терминологии PostgreSQL. Это означает, что она оперирует набором данных, аналогично тому, как это делают функции sum() и mean(). Например, "SELECT EXTENT(GEOM) FROM GEOMTABLE" возвратит BOX3D с максимальными протяженностями для всего содержимого таблицы. Аналогично, "SELECT EXTENT(GEOM) FROM GEOMTABLE GROUP BY CATEGORY" возвратит по одному результату для каждой категории.

ST_zmflag(geometry)
Возвращает флаг ZM (семантическая размерность - dimension semantic) указанной геометрии, как малое целое (small int). Значения: 0=2d, 1=3dm, 2=3dz, 3=4d.

ST_HasBBOX(geometry)
Возвращает TRUE, если охват геометрии является кэшированным. В противном случае - FALSE. Используйте addBBOX() и dropBBOX() для управления кэшированием.

ST_ndims(geometry)
Возвращает число размерностей геометрии, как малое целое (small int). Значения: 2, 3 или 4.

ST_nrings(geometry)
Если геометрия является полигоном или мультиполигоном, возвращает число кругов.

ST_npoints(geometry)
Возвращает число точек в геометрии.

ST_isvalid(geometry)
Возвращает true, если геометрия является валидной.

ST_expand(geometry, float)
Эта функция возвращает охват геометрии, расширенный во всех направлениях на значение, указанное во втором аргументе. Очень удобно использовать эту функцию при запросах distance() для добавления к запросу фильтра по индексу.

ST_estimated_extent([schema], table, geocolumn)
Возвращает "оценочный" охват данной пространственной таблицы. Оценка берется из статистики столбцов геометрии. Используется текущая схема если не определена другая.
В PostgreSQL>=8.0.0 сбор статистики осуществляется с помощью VACUUM ANALYZE, и результат будет составлять около 95% от реального.
В PostgreSQL<8.0.0 сбор статистики осуществляется с помощью update_geometry_stats(), и результат будет точным.

ST_find_srid(varchar,varchar,varchar)
Синтаксис является следующим: find_srid(<БД/схема>, <таблица>, <столбец>). Функция возвращает целочисленный SRID указанного столбца, находящийся в таблице GEOMETRY_COLUMNS. Если геометрический столбец не был должным образом создан с помощью функции AddGeometryColumns(), эта функция будет работать неправильно.

ST_mem_size(geometry)
Возвращает размер (в байтах), занимаемый геометрией.

ST_point_inside_circle(geometry, float, float, float)
Синтаксис этой функции: point_inside_circle(<геометрия>, <центр_круга_x>, <центр_круга_y>, <радиус>). Возвращает true, если данная геометрия является точкой и находится внутри круга. В противном случае возвращает false.

ST_xmin(box3d) ymin(box3d) zmin(box3d)
Возвращает запрошенный минимальный охват.

ST_xmax(box3d) ymax(box3d) zmax(box3d)
Возвращает запрошенный максимальный охват.

ST_Accum(geometry set)
Аггрегация. Строит массив геометрий.

6.2.9. Поддержка долгих транзакций

Этот модуль и ассоциированные функции pl/pgsql может быть использован для организации поддержки долгих блокировок, описанных в спецификации Web Feature Service.
Замечание: Пользователи должны использовать сериализуемый уровень транзакции, иначе механизм блокировок может не работать.

EnableLongTransactions()
Включить поддержку долгих транзакций. Эта функция создает необходимые метаданные таблиц; должна быть вызвана перед использованием других функций этого раздела. Повторный вызов безвреден.
Поддерживается с 1.1.3.

DisableLongTransactions()
Отключить поддержку долгих транзакций. Эта функция удаляет метаданные таблиц, поддерживающие долгие транзакции, и удаляет все триггеры, добавленные к заблокированным таблицам.
Поддерживается с 1.1.3.

CheckAuth([<schema>], <table>, <rowid_col>)
Проверяет UPDATE и DELETE записей данной таблицы на авторизацию. Идентифицирует строки с помощью столбца <rowid_col>.
Поддерживается с 1.1.3.

LockRow([<schema>], <table>, <rowid>, <authid>, [<expires>])
Устанавливает блокировку/авторизацию для указанной записи в таблице. <authid> является текстовым значением, <expires> - временная метка, по умолчанию равная now()+1hour. Возвращает 1, если блокировка прошла успешно, и 0 - в противном случае (уже заблокирована другим пользователем).
Поддерживается с 1.1.3.

UnlockRows(<authid>)
Снять все блокировки, проведенные с указанным идентификатором авторизации. Возвращает число всех снятых блокировок.
Поддерживается с 1.1.3.

AddAuth(<authid>)
Добавляет признак авторизации для использования в текущей транзакции.
Поддерживается с 1.1.3.

6.3. Функции SQL-MM

Здесь приведен список определенных функций SQL-MM, которые корректно поддерживает PostGIS. Реализация этих функций придерживается реализации ArcSDE и, поэтому, несколько отличается от спецификации. Эти отличия будут указаны.
Начиная с версии 1.2.0 эти функции реализованы как обертки над существующими функциями PostGIS. В результате, поддержка криволинейных геометрий реализована для многих функций не полностью .
Замечание: SQL-MM определяет SRID всех геометрических конструкторов по умолчанию как 0. SRID по умолчанию используемый PostGIS равен -1.

ST_Area
Возвращает значение площади ST_Surface или ST_MultiSurface.
SQL-MM 3: 8.1.2, 9.5.3

ST_AsBinary
Возвращает WKB представление значения ST_Geometry.
SQL-MM 3: 5.1.37

ST_AsText
Возвращает WKT представление значения ST_Geometry.
SQL-MM 3: 5.1.25

ST_Boundary
Возвращает границу значения ST_Geometry.
SQL-MM 3: 5.1.14

ST_Buffer
Возвращает буфер вокруг значения ST_Geometry.
SQL-MM 3: 5.1.17

ST_Centroid
Возвращает математический центроид значения ST_Surface или ST_MultiSurface.
SQL-MM 3: 8.1.4, 9.5.5

ST_Contains

Проверяет, является ли значение ST_Geometry пространственно содержащим другое значение ST_Geometry.
SQL-MM 3: 5.1.31

ST_ConvexHull
Конвексный полигон геометрии представляет собой минимальную геометрию, которая описывает все геометрии набора.
Эта функция обычно используется с MULTI геометриями и Geometry Collections. Хотя это не аггрегирование - вы можете использовать ее в паре с ST_Collect чтобы получить конвексный полигон набора точек. ST_ConvexHull(ST_Collect(somepointfield)). Эта функция часто используется для определения зоны влияния определенной набором точек.
SQL-MM 3: 5.1.16

ST_CoordDim
Возвращает размерность координат значения ST_Geometry.
SQL-MM 3: 5.1.3

ST_Crosses
Проверяет, является ли значение ST_Geometry пространственно скрещивающимся с другим значением ST_Geometry.
SQL-MM 3: 5.1.29

ST_Difference
Возвращает значение ST_Geometry, которое представляет множество точек разности двух значений ST_Geometry.
SQL-MM 3: 5.1.20

ST_Dimension
Возвращает размерность значения ST_Geometry.
SQL-MM 3: 5.1.2

ST_Disjoint
Проверяет, является ли значение ST_Geometry пространственно разделенным другим значением ST_Geometry.
SQL-MM 3: 5.1.26

ST_Distance
Возвращает расстояние между двумя геометриями.
SQL-MM 3: 5.1.23

ST_EndPoint
Возвращает значение ST_Point, которое является конечной точкой значения ST_Curve.
SQL-MM 3: 7.1.4

ST_Envelope
Возвращает охват для значения ST_Geometry.
SQL-MM 3: 5.1.15

ST_Equals
Проверяет, является ли значение ST_Geometry пространственно эквивалентным другому значению ST_Geometry.
SQL-MM 3: 5.1.24

ST_ExteriorRing
Возвращает внешнюю дугу для ST_Surface.
SQL-MM 3: 8.2.3, 8.3.3

ST_GeometryN
Возвращает указанное значение ST_Geometry для ST_GeomCollection.
SQL-MM 3: 9.1.5

ST_GeometryType
Возвращает тип геометрии для значения ST_Geometry.
SQL-MM 3: 5.1.4

ST_GeomFromText
Возвращает заданное значение ST_Geometry.
SQL-MM 3: 5.1.40

ST_GeomFromWKB
Возвращает заданное значение ST_Geometry.
SQL-MM 3: 5.1.41

ST_InteriorRingN
Возвращает указанную внутреннюю дугу значения ST_Surface.
SQL-MM 3: 8.2.6, 8.3.5

ST_Intersection
Возвращает значение ST_Geometry, которое представляет множество точек пересечения двух значений ST_Geometry.
Другими словами - часть геометрии A и геометрии B, которая присутсвует в обоих геометриях.
SQL-MM 3: 5.1.18

ST_Intersects
Проверяет, является ли значение ST_Geometry пространственно пересекающимся с другим значением ST_Geometry.
SQL-MM 3: 5.1.27

ST_IsClosed
Проверяет, является ли замкнутым значение ST_Curve или ST_MultiCurve.
Замечание: Согласно SQL-MM результат ST_IsClosed(NULL) должен быть 0, но в PostGIS возвращается NULL.
SQL-MM 3: 7.1.5, 9.3.3

ST_IsEmpty
Проверяет, соответствует ли значение ST_Geometry пустому множеству.
Замечание: Согласно SQL-MM результат ST_IsEmpty(NULL) должен быть 0, но в PostGIS возвращается NULL.
SQL-MM 3: 5.1.7

ST_IsRing
Проверяет, является ли значение ST_Curve дугой.
Замечание: Согласно SQL-MM результат ST_IsRing(NULL) должен быть 0, но в PostGIS возвращается NULL.
SQL-MM 3: 7.1.6

ST_IsSimple
Проверяет, имеет ли значение ST_Geometry особые геометрические точки, такие, как самопересечение или самокасание.
Замечание: Согласно SQL-MM результат ST_IsSimple(NULL) должен быть 0, но в PostGIS возвращается NULL.
SQL-MM 3: 5.1.8

ST_IsValid
Проверяет, является ли значение ST_Geometry правильно сформированным.
Замечание: Согласно SQL-MM результат ST_IsValid(NULL) должен быть 0, но в PostGIS возвращается NULL. Согласно SQL-MM результат ST_IsValid(NULL) должен быть 1.
SQL-MM 3: 5.1.9

ST_Length
Возвращает длину значения ST_Curve или ST_MultiCurve.
SQL-MM 3: 7.1.2, 9.3.4

ST_LineFromText
Возвращает описанное значение ST_LineString.
SQL-MM 3: 7.2.8

ST_LineFromWKB
Возвращает описанное значение ST_LineString.
SQL-MM 3: 7.2.9

ST_MLineFromText
Возвращает описанное значение ST_MultiLineString.
SQL-MM 3: 9.4.4

ST_MLineFromWKB
Возвращает описанное значение ST_MultiLineString.
SQL-MM 3: 9.4.5

ST_MPointFromText
Возвращает описанное значение ST_MultiPoint.
SQL-MM 3: 9.2.4

ST_MPointFromWKB
Возвращает описанное значение ST_MultiPoint.
SQL-MM 3: 9.2.5

ST_MPolyFromText
Возвращает описанное значение ST_MultiPolygon.
SQL-MM 3: 9.6.4

ST_MPolyFromWKB
Возвращает описанное значение ST_MultiPolygon.
SQL-MM 3: 9.6.5

ST_NumGeometries
Возвращает число геометрий в ST_GeomCollection.
SQL-MM 3: 9.1.4

ST_NumInteriorRing
Возвращает число внутренних дуг в ST_Surface.
SQL-MM 3: 8.2.5

ST_NumPoints
Возвращает число точек в значении ST_LineString или ST_CircularString.
SQL-MM 3: 7.2.4

ST_OrderingEquals
ST_OrderingEquals сравнивает две геометрии и возвращает t (TRUE), если эти геометрии эквивалентны и их координаты имеют одинаковый порядок; в противном случае возвращает f (FALSE).
Замечание: Эта функция реализована согласно спецификации ArcSDE SQL, которая отличается от SQL-MM. http://edndoc.esri.com/arcsde/9.1/sql_api/sqlapi3.htm#ST_OrderingEquals
SQL-MM 3: 5.1.43

ST_Overlaps
Проверяет является ли значение ST_Geometry пространственным покрытием другого значения ST_Geometry.
SQL-MM 3: 5.1.32

ST_Perimeter
Возвращает длину границы значения ST_Surface или ST_MultiRSurface.
SQL-MM 3: 8.1.3, 9.5.4

ST_Point
Возвращает ST_Point с заданными значениями координат.
SQL-MM 3: 6.1.2

ST_PointFromText
Возвращает заданное значение ST_Point.
SQL-MM 3: 6.1.8

ST_PointFromWKB
Возвращает заданное значение ST_Point.
SQL-MM 3: 6.1.9

ST_PointN
Возвращает значение указанного ST_Point из ST_LineString или ST_CircularString.
SQL-MM 3: 7.2.5, 7.3.5

ST_PointOnSurface
Возвращает значение ST_Point, гарантированно лежащее на поверхности значения ST_Surface или ST_MultiSurface.
SQL-MM 3: 8.1.5, 9.5.6

ST_PolyFromText
Возвращает заданное значение ST_Polygon.
SQL-MM 3: 8.3.6

ST_PolyFromWKB
Возвращает заданное значение ST_Polygon.
SQL-MM 3: 8.3.7

ST_Polygon
Возвращает полигон, созданный из указанной ломаной и SRID.
SQL-MM 3: 8.3.2

ST_Relate
Проверяет, является ли значение ST_Geometry пространственно связанным с другим значением ST_Geometry.
SQL-MM 3: 5.1.25

ST_SRID
Возвращает идентификатор пространственной системы координат (SRID) значения ST_Geometry.
SQL-MM 3: 5.1.5

ST_StartPoint
Возвращает значение ST_Point, которое является стартовым для значения ST_Curve.
SQL-MM 3: 7.1.3

ST_SymDifference
Возвращает значение ST_Geometry, представленное множеством точек симметрической разности двух значений ST_Geometry.
SQL-MM 3: 5.1.21

ST_Touches
Проверяет, является ли значение ST_Geometry пространственно соприкасающимся с другим значением ST_Geometry.
SQL-MM 3: 5.1.28

ST_Transform
Возвращает значение ST_Geometry, трансформированное в указанную пространственную систему координат.
SQL-MM 3: 5.1.6

ST_Union
Возвращает значение ST_Geometry, которое представлено множеством точек объединения двух значений ST_Geometry.
SQL-MM 3: 5.1.19

ST_Within
Проверяет, является ли лежит значение ST_Geometry пространственно внутри другого значения ST_Geometry.
SQL-MM 3: 5.1.30

ST_WKBToSQL
Возвращает значение ST_Geometry для данного WKB представления.
SQL-MM 3: 5.1.36

ST_WKTToSQL
Возвращает значение ST_Geometry для данного WKT представления.
SQL-MM 3: 5.1.34

ST_X
Возвращает значение координаты x для значения ST_Point.
SQL-MM 3: 6.1.3

ST_Y
Возвращает значение координаты y для значения ST_Point.
SQL-MM 3: 6.1.4

6.4. Функции ArcSDE

Дополнительные функции, добавленные для улучшения поддержки интерфейсов в стиле ArcSDE.

SE_EnvelopesIntersect
Возвращает t (TRUE), если охваты двух геометрий пересекаются; в противном случае возвращает f (FALSE).

SE_Is3d
Проверяет, имеет ли геометрическое значение установленную координату z.

SE_IsMeasured
Проверяет, имеет ли геометрическое значение установленную координату m.

SE_LocateAlong
Возвращает значение полученной геометрической коллекции с элементами, которые равны указанной оценке.

SE_LocateBetween
Возвращает полученные значения наборов геометрий с элементами, которые совпадают с указанным направлением, в содержащихся измерениях.

SE_M
Возвращает значение координаты m для значения ST_Point.

SE_Z
Возвращает значение координаты z для значения ST_Point.

7. Сообщения о ошибках

7.1. Сообщения об ошибках в программном обеспечении

Сообщение об ошибке является существенной помощью разработке PostGIS. Наилучшее сообщение об ошибке должно содержать информацию позволяющую разработчикам PostGIS воспроизвести эту ошибку. Поэтому в идеале сообщение должно содержать скрипт, вызывающий ошибку и всевозможную информацию об окружении, в котором эта ошибка проявилась. Полезная информация может быть получена вызовом `SELECT postgis_full_version()` [для postgis] и `SELECT version()` [для postgresql].

Если вы используете не последний релиз, вам следует сначала посмотреть release changelog, чтобы узнать не исправлена ли уже ваша ошибка.

Использование PostGIS bug tracker гарантирует, что ваше сообщение не будет проигнорировано, и что вы будете информированы о связанном с ним процессе. Перед сообщением о новой ошибке, пожалуйста, посмотрите не присутствует ли она уже в базе, и, если присутствует, пожалуйста, добавьте любую новую информацию о нем.

Перед подготовкой нового сообщения вы можете прочесть статью Simon Tatham о том Как правильно сообщать об ошибке.

7.2. Сообщения об ошибках в документации

Документация должна точно отражать возможности и поведение программного обеспечения. Если это не так, возможно проблема в ошибке программы или в недостаточной или ошибочной документации.

Ошибки документации также могут быть заявлены через PostGIS bug tracker.

Если ваше изменение небольшое, просто опишите его в новом тикете, будучи как можно более конкретным и указав место в документации.

Если изменения более обширные, предпочтительна заплатка к Subversion. Применение заплатки под Unix является процессом из 4 шагов (предполагается, что Subversion установлен):

Загрузите копию PostGIS' Subversion trunk. Если вы используете Unix, введите:
svn checkout http://svn.refractions.net/postgis/trunk/
This will be stored in the directory ./trunk
Сделайте ваши изменения в документации используя подходящий редактор. Если вы используете Unix, введите:
vi trunk/doc/postgis.xml
Отметим, что документация ведется в SGML а не HTML, так что, если вы с ней не знакомы, пожалуйста ознакомьтесь с примером.
Создайте заплатку содержащую изменения относительной главной копии документации. Если вы используете Unix, введите:
svn diff trunk/doc/postgis.xml > doc.patch
Добавьте заплатку к новому тикету в bug tracker.

A.1. Примечания к релизам

A.1.1. Релиз 1.3.3
Дата релиза: 2008/04/12
В релизе исправлены ошибки shp2pgsql, расширена функциональность поддержки SVG и KML, добавлена функция ST_SimplifyPreserveTopology, сборка более чувствительна к версии GEOS, а также исправлены несколько важных, но редких критических ошибок.

A.1.2. Релиз 1.3.2
Дата релиза: 2007/12/01
В релизе исправлены ошибки в ST_EndPoint() и ST_Envelope, улучшена поддержка сборки JDBC и OS/X, добавлена улучшенная поддержка вывода GML с ST_AsGML(), включая вывод GML3.

A.1.3. Релиз 1.3.1
Дата релиза: 2007/08/13
В этом релизе исправлены некоторые оплошности, допущенные в предыдущем релизе, - номера версий, документация, тэги.

A.1.4. Релиз 1.3.0
Дата релиза: 2007/08/09
В этом релизе была повышена производительность реляционных функций, добавлены новые реляционные функции и началось переименование функций в соответствии с соглашением SQL-MM, с использованием префикса пространственного типа (ST).

A.1.4.1. Добавленная функциональность
JDBC: добавлен диалект Hibernate. Спасибо Norman Barker.
Добавлены реляционные функции ST_Covers и ST_CoveredBy. Описания и обоснования этих функций могут быть найдены на http://lin-ear-th-inking.blogspot.com/2007/06/subtleties-of-ogc-covers-spatial.html
Добавлена реляционная функция ST_DWithin

A.1.4.2. Увеличение производительности
Добавлено кэширование и индексирование точек-в-полигонах коротких-обходов для функций ST_Contains, ST_Intersects, ST_Within и ST_Disjoint.
Добавлена поддержка линейных индексов для реляционных функций (исключая ST_Disjoint).

A.1.4.3. Прочие изменения
Расширена поддержка криволинейной геометрии в геометрии доступа и некоторых функций обработки.
Начат перевод функций на соглашение SQL-MM о наименованиях с использованием пространственного префикса (ST) типа.
Добавлена первоначальная поддержка PostgreSQL 8.3.

A.1.5. Release 1.2.1
Дата релиза: 2007/01/11
Этот релиз исправляет ошибки поддержки PostgreSQL 8.2 и незначительно повышает производительность.

A.1.5.1. Изменения
Исправлен мелкая ошибка "точка-в-полигоне" в Within().
Исправлена трактовка NULL для индексов PostgreSQL 8.2.
Обновлены специальные файлы RPM.
В Transform() добавлены короткие-замыкания для случая no-op.
JDBC: Исправлен обработчик JTS для многомерных геометрий (спасибо Томасу Марти за совет и частичный патч). Кроме того, JavaDoc теперь компилируется и пакетируется. Исправлены проблемы classpath с GCJ. Исправлена совместимость с pgjdbc 8.2. Прекращена поддержка jdk 1.3 и более старых.

A.1.6. Release 1.2.0
Дата релиза: 2006/12/08
Этот релиз проводит типовые определения с возможностью сериализации/десериализации криволинейных геометрий, определенных SQL-MM, а также повышает производительность.

A.1.6.1. Изменения
Добавлена поддержка сериализации/десериализации криволинейной геометрии.
В функции Contains и Within добавлены короткие замыкания "точка-в-полигоне" для повышения производительности в соответствующих случаях.

A.1.7. Релиз 1.1.6
Дата релиза: 2006/11/02
В этом релизе исправлены ошибки, в т.ч. исправлена критическая ошибка с интерфейсом GEOS в 64-битных системах. Включены обновление параметров SRS и улучшения перепроектирования (берет Z в рассмотрение). *Рекомендуем* обновиться.

A.1.7.1. Обновление
Если вы обновляетесь с релиза 1.0.3 или старше, следует произвести процедуру SOFT-обновления.
Если вы обновляетесь с релиза *между 1.0.0RC6 и 1.0.2* (включительно), вам следует прочитать раздел "Обновление" заметки о релизе 1.0.3 в этой главе.

Обновление с любого релиза по 1.0.0RC6 осуществляется как HARD-обновление.

A.1.7.2. Исправления ошибок

Исправлено изменение CAPI, которое не работало на 64-битной платформе.
Загрузчик/дампер: исправлены тесты и использование вывода.
Исправлена ошибка setSRID() в JDBC. Спасибо Thomas Marti.

A.1.7.3. Прочие изменения

В перепроектировке используется координата Z.
spatial_ref_sys.sql обновлен EPSG 6.11.1.
Упрощена инфраструктура Version.config. Везде используется простой набор переменных версии.
В сообщения USAGE загрузчика/дампера включен Version.config.
Переписана ручная работа со свойствами парсера версии JDBC.

A.1.8. Релиз 1.1.5

Дата релиза: 2006/10/13
Это релиз исправлений ошибок, включает критические исправления для win32. Рекомендуется *обновиться*.

A.1.8.1. Обновление

Если вы обновляетесь с релиза 1.0.3 или младше, проделайте процедуру SOFT-обновления.
Если вы обновляетесь с релиза *между 1.0.0RC6 и 1.0.2* (включительно), прочтите раздел "Обновление" для релиза 1.0.3 в этой главе.
Обновление с релиза младше чем 1.0.0RC6 выполняется как HARD-обновление.

A.1.8.2. Исправления ошибок

Исправлена ошибка ссылки MingW, которая проявлялась как ошибка сегментации в Win32 с установкой на PostgreSQL 8.2.
Исправлено исключение нулевой точки в методе Geometry.equals() в Java.
Добавлен EJB3Spatial.odt для выполнения требований GPL к дистрибуции "привиллегированных форм модификации".
Удалена устаревшая синхронизация из кода JDBC Jts.
Обновлены сильно устаревшие файлы README для shp2pgsql/pgsql2shp. Которые теперь соответствуют страницам man.
Исправлен тег версии в коде jdbc, который указывал версию "1.1.3" в релизе "1.1.4".

A.1.8.3. Новые возможности

Добавлена опция -S для немножественных геометрий в shp2pgsql.

A.1.9. Релиз 1.1.4

Дата релиза: 2006/09/27
Этот релиз включает исправления ошибок и некоторые улучшения в интерфейсе Java. Рекомендуется *обновление*.

A.1.9.1. Обновление

Если вы обновляетесь с релиза 1.0.3 или старше, выполните процедуру SOFT-обновления.
Если вы обновляетесь с релиза *между 1.0.0RC6 и 1.0.2* (включительно), вам следует прочитать раздел "Обновление" для релиза 1.0.3 в этой главе.
Обновление с релиза младше 1.0.0RC6 требует "HARD-обновления".

A.1.9.2. Исправления ошибок

Исправлена поддержка для PostgreSQL 8.2
Исправлена ошибка в функции collect(), сбрасывающая SRID при вводе.
Добавлен SRID для MakeBox2d и MakeBox3d.
Исправлены тесты работы с GEOS-3.0.0.
Улучшен конкурентный запуск pgsql2shp.

A.1.9.3. Изменения Java

Переработана поддержка JTS согласно новому направлению разработки с обработкой SRID. Упрощен код и используется библиотека GNU Trove.
Добавлена поддержка EJB2, щедро подаренная компанией "Geodetix s.r.l.", - http://www.geodetix.it/
Добавлен учебник / примеры EJB3, подаренные Норманом Баркером (Norman Barker), - <nbarker@ittvis.com>
Реорганизованы директории layout и little.

A.1.10. Релиз 1.1.3

Дата релиза: 2006/06/30
В этом релизе исправлены ошибки, добавлена некоторая новая функциональность (относящейся к поддержке долгих транзакций) и улучшена портируемость. Обновление *рекомендуется*.

A.1.10.1. Обновления

Если вы обновляетесь с релиза 1.0.3 или старше, выполните процедуру SOFT-обновления.
Если вы обновляетесь с релиза *между 1.0.0RC6 и 1.0.2* (включительно), то вам следует прочитать раздел "Обновление" для релиза 1.0.3 в этой главе.
Оьновление с релиза младше 1.0.0RC6 должно производиться как "обновление железа".

A.1.10.2. Исправления ошибок / корректировки

Исправление ошибки в distance(poly,poly), выдающей неправильные результаты.
Исправление ошибки в pgsql2shp для успешного возврата кода.
Исправление ошибки в shp2pgsql в обработке MultiLine WKT.
Исправление ошибки в affine(), неудачном при изменении границ.
Парсер WKT: запрет на построение множества геометрий с пустыми элементами (требуется для поддержки GEOMETRYCOLLECTION).

A.1.10.3. Новая функциональность

НОВШЕСТВО. Поддержка долгих транзакций.
НОВАЯ функция DumpRings().
Новая функция AsHEXEWKB(geom, XDR|NDR).

A.1.10.4. Изменения JDBC

Улучшены тесты: MultiPoint и научные ординаты.
Исправлены некоторые незначительные ошибки в коде jdbc.
Добавлены надлжащие функции доступа для всех полей при подготовке перенесения этих полей в приватный слой.

A.1.10.5. Прочие изменения

Поддержка НОВЫХ тестов для загрузчика/дампера.
Добавлены опции конфигурации --with-proj-libdir и --with-geos-libdir.
Поддержка сборки под Tru64.
Использование Jade для генерации документации.
pgsql2shp больше не ссылается на какие либо библиотеки, кроме необходимых.
Пробная поддержка PostgreSQL 8.2.

A.1.11. Релиз 1.1.2

Дата релиза: 2006/03/30
Релиз содержит исправления ошибок и некоторые новые функции. Улучшена портируемость. Обновление *рекомендуется*.

A.1.11.1. Обновление

Если вы обновляетесь с релиза 1.0.3 или старше, достаточно выполнить процедуру SOFT-обновления.
Если вы обновляетесь с релиза *между 1.0.0RC6 и 1.0.2* (включительно), вам следует прочитать раздел "Обновление" для релиза 1.0.3 в этой главе.
Обновление с релиза до 1.0.0RC6 производится как обновление железа.

A.1.11.2. Исправления ошибок

Исправление ошибки вычислений выводимых границ в SnapToGrid().
Исправление ошибки в EnforceRHR().
Для jdbc2 исправлена обработка SRID в коде JTS.
исправлена поддержка на 64-битных архитектурах.

A.1.11.3. Новая функциональность

Теперь можно запускать тесты до инсталляции PostGIS.
Новая функция матричной трансформации affine().
Новая функция rotate{,X,Y,Z}().
Старые функции перевода и расширения теперь используют встроенную affine().
Встроенный контроль доступа в estimated_extent() для сборки с pgsql >= 8.0.0.

A.1.11.4. Прочие изменения

Более портируемый скрипт ./configure.
Изменен скрипт ./run_test, умолчальное поведение которого теперь более разумно.

A.1.12. Релиз 1.1.1

Дата релиза: 2006/01/23
Этот релиз содержит важные исправления ошибок, *очень рекомендуется* обновиться. Предыдущая версия содержит ошибку в postgis_restore.pl, мешающую выполнять полную процедуру HARD-обновления, и ошибку в коннекторе GEOS-2.2+, мешающую использовать в топологических операциях объекты GeometryCollection.

A.1.12.1. Обновление

Если вы обновляетесь с релиза 1.0.3 или старше, достаточно выполнить процедуру SOFT-обновления.
Если вы обновляетесь с релиза *между 1.0.0RC6 и 1.0.2* (включительно), вам следует прочитать раздел "Обновление" для релиза 1.0.3 в этой главе.
Обновление с релиза до 1.0.0RC6 производится как обновление железа.

A.1.12.2. Исправления ошибок

Исправлено преждевременное завершение postgis_restore.pl.
Исправление ошибки выполнения geometrycollection коннектора GEOS-CAPI.
Улучшена поддержка Solaris 2.7 и MingW.
Исправление ошибки в line_locate_point().
Исправлено управление путями PostgreSQL.
Исправление ошибки в line_substring().

Добавлена поддержка кластера локализации в тестировщике.

A.1.12.3. Новая функциональность

Новая интерполяция Z и M в line_substring().
Новая интерполяция Z и M в line_interpolate_point().
Добавлен алиас NumInteriorRing(), соответствующий двусмысленности OpenGIS.

A.1.13. Релиз 1.1.0

Дата релиза: 2005/12/21
Это - незначительный релиз, содержащий много улучшений и новшеств. Особенно важны: сильное упрощение процедуры сборки; радикальное улучшение выполнения transform(); более стабильное соединение с GEOS (поддержка CAPI); множество новых функций; поддержка топологических планов.
Перед инсталляцией PostGIS *настоятельно рекоммендуется* обновить GEOS до 2.2.x. Это обеспецит обновление возможностей GEOS без пересборки библиотеки PostGIS.

A.1.13.1. Благодарности

Этот релиз содержит код Марка Кэйва Айланда (Mark Cave Ayland) для кэширования объектов proj4. Маркус Шабер (Markus Schaber) добавил много усовершенствований в код JDBC2. Алекс Боднэйру (Alex Bodnaru) помог облегчить исходники, зависящие от PostgreSQL, и предоставил спецфайлы Debian. Мишель Фухр (Michael Fuhr) тестировал новшества на архитектуре Solaris. Давид Течер (David Techer) и Геральд Феной (Gerald Fenoy) помогли тестировать коннектор GEOS C-API. Хартмут Чаунер (Hartmut Tschauner) предоставил код для функции azimuth(). Деврим ГУНДУЗ (Devrim GUNDUZ) предоставил спецификацию RPM. Карл Андерсон помог с новыми функциями построения областей. Другие имена можно посмотреть в разделе Благодарности.

A.1.13.2. Обновление

Если вы обновляетесь с релиза 1.0.3 или старше, вам *НЕ* нужны дамп/восстановление. Просто выполните новый скрипт lwpostgis_upgrade.sql на всех ваших рабочих базах данных. Смотрите подробности в главе Обновление софта.
Если вы обновляетесь с релиза *между 1.0.0RC6 и 1.0.2* (включительно), вам следует прочитать раздел "Обновление" для релиза 1.0.3 в этой главе.
Обновление с любого релиза до 1.0.0RC6 производится как обновление железа.

A.1.13.3. Новые функции

Методы scale() и transscale(), сопутствующие 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 для точек ветвления (поддержка до 4d).
BuildArea(any_geometry)
OGC BdPolyFromText(linestring_wkt, srid)
OGC BdMPolyFromText(linestring_wkt, srid)
RemovePoint(linestring, offset)
ReplacePoint(linestring, offset, point)

A.1.13.4. Исправления ошибок

Исправлена утечка памяти в polygonize()
Исправлена ошибка в функциях расстояний lwgeom_as_anytype.
Исправлены элементы USE_GEOS, USE_PROJ и USE_STATS вывода postgis_version(), отражающих состояние библиотеки.

A.1.13.5. Изменения семантических функций

SnapToGrid не отбрасывает лишние измерения.
Изменена функция Z(), которая теперь возвращает NULL, если требуемое измерение не доступно.

A.1.13.6. Улучшения исполнения

С помощью кэширования объектов proj4 значительна ускорена функция transform().
Отменен автоматический вызов fix_geometry_columns() в AddGeometryColumns() и update_geometry_stats().

A.1.13.7. Работа JDBC2

Усовершенствован Makefile.
Усовершенствована поддержка JTS.
Улучшена система тестов.
Метод проверки основной последовательности для геометрических наборов.
Поддержка (Hex)(E)wkb.
Автопроверка DriverWrapper для переключения HexWKB / EWKT.
Исправлены проблемы компиляции в ValueSetter для старых релизов jdk.
Исправлены конструкторы EWKT, допускавшие представление SRID=4711;.
Добавлена предварительная поддержка толко для чтения для геометрий java2d.

A.1.13.8. Прочие новшества
Конфигурация полностью основана на autoconf, освобожденного от исходников PostgreSQL.
Поддержка GEOS C-API (2.2.0 и выше).
Начальная поддержка топологического моделирования.
Спецфайлы Debian и RPM.
Новый скрипт lwpostgis_upgrade.sql

A.1.13.9. Прочие изменения
Усовершенствована поддержка JTS.
Строгое соответствие между целочисленными и строковыми атрибутами DBF и SQL.
Пополнен и почищен набор тестов.
Из релиза удален старый код jdbc.
Изменено устаревшее использование postgis_proc_upgrade.pl.
Версии скриптов приведены в соответствие с версией релиза.

A.1.14. Релиз 1.0.6
Дата релиза: 2005/12/06
Содержит новые исправления ошибок и усовершенствования.

A.1.14.1. Обновление
Если вы обновляетесь с релиза 1.0.3 или старше, вам *НЕ* нужны дамп/восстановление.
Если вы обновляетесь с релиза *между 1.0.0RC6 и 1.0.2* (включительно), вам следует прочитать раздел "Обновление" для релиза 1.0.3 в этой главе.
Обновление с релиза до 1.0.0RC6 производится как обновление железа.

A.1.14.2. Исправления ошибок
Исправлен вызов palloc(0) в десериализаторе наборов (проблема проявлялась только с --enable-cassert).
Исправлены ошибки в управлении кэшем bbox.
Исправлена ошибка сегментации в geom_accum(NULL, NULL).
Исправлена ошибка сегментации в addPoint().
Исправлено недостаточное распределение в lwcollection_clone().
Исправлена ошибка в segmentize().
Исправлено вычисление вывода SnapToGrid в bbox.

A.1.14.3. Улучшения
Начальная поддержка postgresql 8.2.
В GEOS ops добавлена проверка недостающих SRID.

A.1.15. Релиз 1.0.5
Дата релиза: 2005/11/25
Содержит исправления в управлении памаятью в библиотеке, исправление ошибки сегментации в обработке загрузчиком атрибутов UTF8 и новые усовершенствования и чистки.
Замечание: Код возврата shp2pgsql изменен сравнительно с предыдущими версиями для соответствия стандартам unix (возвращает 0 в случае успеха).

A.1.15.1. Обновление
Если вы обновляетесь с релиза 1.0.3 или старше, вам *НЕ* нужны дамп/восстановление.
Если вы обновляетесь с релиза *между 1.0.0RC6 и 1.0.2* (включительно), вам следует прочитать раздел "Обновление" для релиза 1.0.3 в этой главе.
Обновление с релиза до 1.0.0RC6 производится как обновление железа.

A.1.15.2. Изменения библиотеки
Решены проблемы управления памяти.
Исправлено вычисление несуществующих значений дробей а анализаторе.
Исправлены мелкие ошибки в низкоуровневой функции getPoint4d_p().
Ускорение работы функций сериализатора.
Исправлена ошибка в force_3dm(), force_3dz() и force_4d().

A.1.15.3. Изменения загрузчика
Исправлен код возврата для shp2pgsql.
Исправлена обратная совместимомть для загрузчика (загрузка недопустимых шейп-файлов).
Исправлено обращение с разделительными точками в числовых атрибутах в DBF.
Исправлена ошибка сегментации в shp2pgsql (кодировка utf8).

A.1.15.4. Прочие изменения
Поддержка схем в postgis_proc_upgrade.pl для pgsql 7.2+.
Новая глава руководства "Сообщения об ошибках".

A.1.16. Релиз 1.0.4
Дата релиза: 2005/09/09
Содержит важные исправления ошибок и новые усовершенствования. Как правило, - это исправления утечек памяти, мешавших успешному построению индексов GiST на больших пространственных таблицах.

A.1.16.1. Обновление
Если вы обновляетесь с релиза 1.0.3, вам *НЕ* нужны дамп/восстановление.

Если вы обновляетесь с релиза *между 1.0.0RC6 и 1.0.2* (включительно), вам следует прочитать раздел "Обновление" для релиза 1.0.3 в этой главе.
Обновление с релиза до 1.0.0RC6 производится как обновление железа.

A.1.16.2. Исправления ошибок
Заткнули утечку памяти в индексировании GiST.
Исправлена ошибка сегментации в transform() в управлении ошибками proj4.
Исправлены несколько текстов proj4 в spatial_ref_sys (отсутствие +proj).
Загрузчик: исправлено использование строковых функций, переработана проверка объектов NULL, исправлена ошибка сегментации при вводе MULTILINESTRING.
Исправлена ошибка при работе с размерностями в MakeLine.
Исправлена ошибка в translate(), портящая выводимые границы.

A.1.16.3. Улучшения
Усовершенствована документация.
Более мощный оценщик выборки.
Незначительно увеличение скорости выполнения distance().
Незначительные чистки.
Почищено индексирование GiST.
Упрощение синтаксиса, принятого в парсере box3d.

A.1.17. Релиз 1.0.3
Дата релиза: 2005/08/08
Содержит исправления нескольких ошибок, *включая корректировку запасных геометрий*, и новые усовершенствования.

A.1.17.1. Обновление
Из-за ошибки в вычичлении границ, процедура обновления требует специального внимания. Кэшированные в базе границы могут быть некорректны.
Процедура HARD-обновления (дамп/восстановление) принудительно пересчитывает все границы (не включая дампы). Это *необходимо*, если обновляетесь с релиза до 1.0.0RC6.
Если вы обновляетесь с версии 1.0.0RC6 или выше, этот релиз включает PERL-скрипт (utils/rebuild_bbox_caches.pl), производящие пересчет геометрических границ и выполняющий все прочие необходимые операции (обновление геометрической статистики, переиндексация). Запустите скрипт после 'make install' (запустите без аргументов для получения помощи по синтаксису). Опциональный запуск utils/postgis_proc_upgrade.pl приводит к обновлению процедур PosGIS и подписей функций (смотрите обновление софта).

A.1.17.2. Исправления ошибок
Несколько исправлений ошибок в вычислении 2d границ lwgeom.
Исправление ошибки при обработке POINT в WKT (-w) в загрузчике.
Исправление ошибки в дампере на 64-битных машинах.
Исправление ошибки в дампере при обработке запросов, определенных пользователями.
Исправление ошибки в скрипте create_undef.pl.

A.1.17.3. Усовершенствования
Небольшое улучшение выполнения в функции канонического ввода.
Незначительная чистка загрузчика.
Поддержка многобайтных имен полей в загрузчике.
Усовершенствован скрипт postgis_restore.pl.
Новый полезный скрипт rebuild_bbox_caches.pl.

A.1.18. Релиз 1.0.2
Дата релиза: 2005/07/04
Содержит новые исправления ошибок и усовершенствования.

A.1.18.1. Обновление
Если вы обновляетесь с релиза 1.0.0RC6 или выше, вы *НЕ* нуждаетесь в дампе/восстановлении.
Обновление с более старых релизов требует дампа/перезагрузки. Смотрите подробную информацию в главе Обновление.

A.1.18.2. Исправления ошибок
Неприемлемое ops b-дерева.
Заткнули утечку памяти в pg_error.
Исправлен индекс R-дерева.
Почищены скрипты сборки (во избежание путаницы между CFLAGS и CXXFLAGS).

A.1.18.3. Усовершенствования
Новые возможности создания индекса в загрузчике (переключатель -I).
Начальная поддержка для PostgreSQL 8.1dev.

A.1.19. Релиз 1.0.1
Дата релиза: 2005/05/24
Содержит новые исправления ошибок и некоторые усовершенствования.

A.1.19.1. Обновление
Если вы обновляетесь с релиза 1.0.0RC6 или выше, вы *НЕ* нуждаетесь в дампе/восстановлении.
Обновление с более старых релизов требует дампа/перезагрузки. Смотрите подробную информацию в главе Обновление.

A.1.19.2. Изменения библиотеки
Исправление ошибки 3d-вычислений в length_spheroid().
Исправление ошибки в оценщике присоединенной выборки.

A.1.19.3. Прочие изменения/добавления
Исправление ошибки утечки в функциях shp2pgsql.
Улчшена поддержка конкурнтности PostGIS в нескольких схемах.
Исправления документации.
jdbc2: компиляция с "-target 1.2 -source 1.2" по умолчанию.
НОВЫЙ переключатель -k для pgsql2shp
НОВАЯ поддержка кастомных опций для createdb в postgis_restore.pl.
Исправление ошибки в именах атрибутов в pgsql2shp.
Исправление ошибки в определениях парижских проекций.
Почищен postgis_restore.pl.

A.1.20. Релиз 1.0.0
Дата релиза: 2005/04/19
Финал релиза 1.0.0. Содержит новые исправления ошибок, новые усовершенствования загрузчика (более полная поддержка старых версий PostGIS) и пополненную документацию.

A.1.20.1. Обновление
Если вы обновляетесь с релиза 1.0.0RC6, вы *НЕ* нуждаетесь в дампе/восстановлении.
Обновление с любых других предыдущих релизов требует дампа/перезагрузки. Смотрите подробную информацию в главе Обновление.

A.1.20.2. Изменения библиотеки
Исправление ошибки в transform() со случайным освобождением адресов памяти.
Исправление ошибки в force_3dm() с распределением меньшей. чем необходимо, памяти.
Исправление ошибки в присоединенном избирательном оценщике (defaults, leaks, tuplecount, sd).

A.1.20.3. Прочие изменения/добавления
Исправление ошибки в shp2pgsql с потерей значений, начинающихся с таба или с апострофа.
НОВЫЕ страницы мануала для загрузчика/дампера.
НОВАЯ поддержка старых версий (HWGEOM) PostGIS в shp2pgsql.
НОВЫЙ флаг -p (prepare) для shp2pgsql.
НОВАЯ глава мануала о совместимости с OGC.
НОВАЯ поддержка автоконфигурации для библиотеки JTS.
Исправление ошибки в оценщике тестов (поддержка LWGEOM и парсинга схем).

A.1.21. Релиз 1.0.0RC6
Дата релиза: 2005/03/30
Шестой релиз-кандидат для 1.0.0. Содержит новые исправления ошибок и чистки.

A.1.21.1. Обновление
Для обновления с предыдущих релизов вам необходим дамп/восстановление. Более подробную информацию смотрите в главе Обновление.

A.1.21.2. Изменения библиотеки
Исправление ошибки в multi()
Ранний возврат [когда noop] с multi()

A.1.21.3. Изменения скриптов
Удалены функции {x,y}{min,max}(box2d).

A.1.21.4. Прочие изменения
Исправление ошибки в скрипте postgis_restore.pl.
Исправление ошибки в дампере на платформе 64-бит.

A.1.22. Релиз 1.0.0RC5
Дата релиза: 2005/03/25
Пятый релиз-кандидат для 1.0.0. Содержит новые исправления ошибок и улучшения.

A.1.22.1. Обновление
Если вы обновляетесь с релиза 1.0.0RC4, вы *НЕ* нуждаетесь в дампе/перезаливке.
Обновление с любых других релизов требует дампа/перезаливки. Более подробную информацию ищите в главе Обновление.

A.1.22.2. Изменения библиотеки
Исправление ошибки (ошибка сегментации) в вычислении box3d (да, еще раз!).
Исправление ошибки (ошибка сегментации) в estimated_extent().

A.1.22.3. Прочие изменения
Уменьшение скрипта сборки и чистка утилит.
Дополнительные советы по производительности в документации.

A.1.23. Релиз 1.0.0RC4
Дата релиза: 2005/03/18
Четвертый релиз-кандидат для 1.0.0. Содержит исправления ошибок и новые улучшения.

A.1.23.1. Обновление
Вам следует сделать дамп/восстановление предшествующего релиза. Смотрите подробности в главе "Обновление".

A.1.23.2. Изменения библиотеки
Исправление ошибки (ошибка сегментации) в geom_accum().
Исправление ошибки поддержки на 64-битной архитектуре.
Исправление ошибки вычеслений функции box3d для коллекций.
НОВШЕСТВО. Поддержка вложенных запросов в оценщике выборки.
Ранний возврат для force_collection.
Множество исправлений в SnapToGrid().
Для вывода Box2d снова изменена точность: 15 значащих цифр.

A.1.23.3. Изменения скриптов
НОВАЯ функция distance_sphere().
Изменена реализация get_proj4_from_srid: используется PL/PGSQL вместо SQL.

A.1.23.4. Прочие изменения
Исправление ошибки загрузчика и дампера в обработке многолинейных шейпов.
Исправление ошибки загрузчика, пропускавшего все, кроме первой точки полигона.
jdbc2: почищено код, улучшен Makefile.
Переменные FLEX и YACC устанавливаемые *after* pgsql Makefile.global включаются только если *stripped*-версия pgsql вычисляется как пустая строка.
В релиз добавлен уже сгенерированный парсер.
Окультурена конструкция скриптов.
Усовершенствован контроль версий, который был централизован в Version.config.
Усовершенствований в postgis_restore.pl

A.1.24. Релиз 1.0.0RC3
Дата релиза: 2005/02/24
Третий релиз-кандидат для 1.0.0. Содержит много исправлений ошибок и улучшений.

A.1.24.1. Обновление
Вам необходим дамп/восстановление с предшествующих релизов. Смотрите подробности в главе "Обновление".

A.1.24.2. Изменения библиотеки
Исправление ошибки в transform(): улучшенная обработка ошибки с отсутствующим SRID.
Исправление ошибки в управлении распределением памяти.
Исправление ошибки в force_collection(), вызывающем ошибку коннекта с mapserver на простых (единичных) геометрических типах.
Исправление ошибки в GeometryFromText() с добавлением в кэш bbox.
Понижена точность вывода box2d.
Макросу DEBUG дан префикс PGIS_ во избежание столкновения с одноименным в pgsql.
Закрыта утечка в конвертере GEOS2POSTGIS.
Засчет ускорения освобождения контекста запроса снижены требования к памяти.

A.1.24.3. Изменения скриптов
Исправление ошибки в переплетах 72 индексов.
Исправление ошибки в probe_geometry_columns() с работой с PG72 и поддержкой нескольких геометрических столбцов в одной таблице.
НОВШЕСТВО bool::text cast
Для улучшения характеристик некоторые функции переделаны из STABLE в IMMUTABLE.

A.1.24.4. Изменения JDBC
jdbc2: Небольшие патчи, тесты box2d/3d, ревизия документации и лицензии.
jdbc2: Исправлена ошибка в авторегистрации типов pgjdbc 8.0.
jdbc2: Прекращено использование только возможностей jdk1.4 для сборки со старыми релизами jdk.
jdbc2: Добавлена поддержка сборки без pg72jdbc2.jar
jdbc2: Обновлен и почищен makefile
jdbc2: Добавленна BETA-поддержка геометрических классов jts.
jdbc2: Скрыты тесты "известная-неудача" в применении к старым серверам PostGIS.
jdbc2: Исправлен обработчик метрических геометрий в EWKT.

A.1.24.5. Прочие изменения
Новая глава в руководстве, посвященная советам по производительности.
обновление документации: требуется pgsql72, lwpostgis.sql.

Несколько изменений в autoconf.
Извлечение BUILDDATE сделано более портируемым.
Исправлен spatial_ref_sys.sql. теперь избегает vacuum всю базу данных.
spatial_ref_sys: изменено вхождение Paris в более чем один дистрибутив с 0.x.

A.1.25. Релиз 1.0.0RC2

Дата релиза: 2005/01/26
Второй релиз-кандидат для 1.0.0, содержащий исправления ошибо и новые усовершенствования.

A.1.25.1. Обновление

Для обновления вам необходим дамп/восстановление с предшествующих релизов. Более подробную информацию вы можете получить в главе Обновление.

A.1.25.2. Изменения библиотеки

Исправление ошибки в вычислении точечного массива box3d.
Исправление ошибки в определении distance_spheroid
Исправление ошибки в transform() с отсутствием обновления кэша bbox.
НОВЫЙ драйвер jdbc (jdbc2)
Поддержка синтаксиса GEOMETRYCOLLECTION(EMPTY) для обратной совместимости.
Ускорение бинарного вывода.
Точные конструкторы OGC WKB/WKT.

A.1.25.3. Изменения в скриптах

Более корректное использование STABLE, IMMUTABLE, STRICT в lwpostgis.sql
Точные конструкторы OGC WKB/WKT.

A.1.25.4. Прочие изменения

Быстрый и более надежный загрузчик (пока без i18n).
Предварительный скрипт автоконфигурации.

A.1.26. Релиз 1.0.0RC1

Дата релиза: 2005/01/13
Это первый кандидат в основные релизы PostGIS, со встроенным хранением типов PostGIS, переработанный, чтобы стать меньше и быстрее на запросах с индексами.

A.1.26.1. Обновление

Для обновления вам необходим дамп/восстановление с предшествующих релизов. Более подробную информацию вы можете получить в главе Обновление.

A.1.26.2. Изменения

Ускорение парсинга канонического ввода.
Потери канонического вывода.
Канонические бинарные IO EWKB с PG>73.
Поддержка координат до 4d, снижение потерь при преобразовании shapefile->postgis->shapefile.
Новые функции: UpdateGeometrySRID(), AsGML(), SnapToGrid(), ForceRHR(), estimated_extent(), accum().
Вертикально позиционированные операторы индексирования.
Функция выбора JOIN.
Больше геометрических конструкторов / редакторов.
Расширенное API PostGIS.
Поддержка UTF8 загрузчиком.