СИСТЕМА
УПРАВЛЕНИЯ
БАЗАМИ
ДАННЫХ

ЛИНТЕР®

ЛИНТЕР БАСТИОН

ЛИНТЕР СТАНДАРТ

Геометрические типы данных

НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ

РЕЛЭКС®

Товарные знаки

РЕЛЭКС™, ЛИНТЕР® являются товарными знаками, принадлежащими ЗАО НПП «Реляционные экспертные системы» (далее по тексту – компания РЕЛЭКС). Прочие названия и обозначения продуктов в документе являются товарными знаками их производителей, продавцов или разработчиков.

Интеллектуальная собственность

О документе

Материал, содержащийся в данном документе, прошел доскональную проверку, но компания РЕЛЭКС не гарантирует, что документ не содержит ошибок и пропусков, поэтому оставляет за собой право в любое время вносить в документ исправления и изменения, пересматривать и обновлять содержащуюся в нем информацию.

Контактные данные

394006, Россия, г. Воронеж, ул. Бахметьева, 2Б.

Тел./факс: (473) 2-711-711, 2-778-333.

e-mail: market@relex.ru.

Техническая поддержка

С целью повышения качества программного продукта ЛИНТЕР и предоставляемых услуг в компании РЕЛЭКС действует автоматизированная система учёта и обработки пользовательских рекламаций. Обо всех обнаруженных недостатках и ошибках в программном продукте и/или документации на него просим сообщать нам в раздел Поддержка на сайте ЛИНТЕР.

Содержание

Предисловие 4
Назначение документа 4
Для кого предназначен документ 4
Необходимые предварительные знания 4
Дополнительные документы 4
Общие сведения 5
Модель геометрических данных OpenGIS 6
Иерархия геометрических классов 6
Класс Geometry 7
Класс Point 8
Класс Curve 8
Класс LineString 9
Класс Surface 9
Класс Polygon 9
Класс GeometryCollection 10
Класс MultiPoint 10
Класс MultiCurve 11
Класс MultiLineString 11
Класс MultiSurface 11
Класс MultiPolygon 11
Форматы геометрических типов данных 13
Текстовый формат 13
Двоичный формат 15
Создание/добавление столбцов геометрического типа 19
Геометрические BLOB-данные 21
Загрузка геометрических данных 22
Загрузка с помощью WKT-формата 23
Создание точки 23
Создание ломаной линии 24
Создание многоугольника 25
Создание группы точек 26
Создание группы ломаных линий 26
Создание группы многоугольников 27
Создание группы геометрических объектов 28
Создание прямоугольника 29
Создание прямой линии 29
Создание круга 29
Создание произвольного геометрического объекта 29
Загрузка с помощью WKB-представления 30
Создание точки 30
Создание ломаной линии 31
Создание многоугольника 32
Создание группы точек 32
Создание группы ломаных линий 33
Создание группы многоугольников 33
Создание группы геометрических объектов 33
Создание произвольного геометрического объекта 34
Выборка значений геометрических типов 35
Выборка в WKT-формате 36
Выборка в WKB-формате 37
Средства обработки геометрических типов данных 38
Функции для анализа свойств геометрических типов 38
Получить название геометрического типа 38

Получить размерность геометрического типа 38
Получить идентификатор системы координат 39
Преобразовать объект к заданной системе координат 39
Получить минимальный ограничивающий прямоугольник 40
Получить границу значения геометрического типа 41
Проверить существование значения геометрического типа 41
Проверить сложность геометрического объекта 42
Функции для создания новых значений геометрических типов 43
Функции для работы с точкой 43
Получить X-координату точки 43
Получить Y-координату точки 43
Функции для работы с ломаной линией 44
Получить координаты начала ломаной линии 44
Получить координаты конца ломаной линии 44
Получить координаты узла ломаной линии 45
Получить длину ломаной линии 45
Получить количество узлов ломаной линии 46
Проверка выпуклости ломаной линии 46
Проверка замкнутости ломаной линии 47
Функции для работы с группой ломаных линий 47
Получить длину группы ломаных линий 47
Проверка замкнутости группы ломаных линий 48
Функции для работы с многоугольником 49
Определение площади многоугольника 49
Определение количества пересекающихся областей многоугольника 49
Определение внешней границы многоугольника 50
Получение заданной пересекающейся области многоугольника 50
Определение геометрического центра многоугольника 51
Получение первой точки внешней границы многоугольника 51
Функции для работы с группой многоугольников 52
Определение площади группы многоугольников 52
Определение геометрического центра группы многоугольников 52
Получение первой точки внешней границы группы многоугольников 53
Функции для работы с группой геометрических объектов 53
Определение количества объектов в группе 53
Получить заданный объект группы 54
Функции отношения геометрических типов 55
Проверка пересечения геометрических объектов 55
Пересечение геометрических объектов 55
Объединение геометрических объектов 56
Разность геометрических объектов 57
Симметричная разность геометрических объектов 58
Расстояние между объектами 60
Проверка расстояния между объектами 60
Буферный геометрический объект 61
Выпуклая оболочка объекта 62
Проверка совпадения объектов 63
Проверка пересечения объектов 63
Проверка перекрытия объектов 64
Проверка разъединения объектов 65
Проверка вложенности объектов (вариант 1) 66
Проверка вложенности объектов (вариант 2) 67
Проверка касания объектов 68
Проверка скрещивания объектов 68
Оптимизация работы с геометрическими данными 70

Управление вводом геометрических данных 73
Проверка корректности вводимых данных 73
Проверка корректности геометрических данных для всей БД 73
Проверка корректности геометрических данных для текущей сессии 73
Управление хранением геометрических BLOB-данных 75
Управление размером геометрических BLOB-данных 76
Проверка корректности геометрических данных 77
Приложение 1. Примеры работы с геометрическим VARBYTE-объектом 79
Приложение 2. Примеры работы с геометрическим BLOB-объектом 82
Указатель функций 87

Предисловие

Назначение документа

Документ содержит описание работы с геометрическими типами данных СУБД ЛИНТЕР.

Документ предназначен для СУБД ЛИНТЕР СТАНДАРТ 6.0 сборка 17.96, далее по тексту СУБД ЛИНТЕР.

Для кого предназначен документ

Документ предназначен для программистов, разрабатывающих приложения с использованием геометрических (географических) данных.

Необходимые предварительные знания

Для работы с геометрическими типами данных необходимо:

- знать основы реляционных баз данных;
- владеть языком баз данных SQL для СУБД ЛИНТЕР.

Дополнительные документы

- СУБД ЛИНТЕР. Справочник по SQL
- СУБД ЛИНТЕР. Справочник кодов завершения
- СУБД ЛИНТЕР. Командный интерфейс

Общие сведения

В СУБД ЛИНТЕР для поддержки геометрических типов данных реализовано подмножество среды SQL с геометрическими типами (SQL with Geometry Types), спецификация которой предложена консорциумом OpenGIS. Столбец таблицы, в котором хранятся геометрические данные, имеет геометрический тип. Спецификация описывает набор геометрических типов данных SQL, а также функций, предназначенных для создания и анализа значений указанных геометрических типов данных.

В СУБД ЛИНТЕР поддерживается работа с геометрическими данными для BIG ENDIAN платформ и включает в себя следующее:

1) на MSBF-платформах бинарное представление значений геометрических типов хранится в нативном для этой платформы формате, для каждого значения геометрического типа в заголовке устанавливается байт
   wkbByteOrder = wkbXDR /* Big Endian */;
2) реализована функция автоматической конвертации, которая:
   - на Intel-платформах преобразовывает входящие значения геометрических типов с порядком байт Big Endian в значения с Little Endian;
   - на MSBF-платформах преобразовывает входящие значения геометрических типов с порядком байт Little Endian в значения с Big Endian.

Примечание
Функция автоматической конвертации не используется, если значения геометрических типов заносятся в БД не с помощью специализированных функций вида GeomFromWKB (PointFromWKB, PolyFromWKB, ...), а напрямую, с помощью функции HEX. В этом случае значения заносятся в БД "как есть" и их корректность можно проверить функцией GeoIsValid().

Геометрические типы данных позволяют генерировать, сохранять и анализировать географические данные, которые отражают элементы окружающего мира:

- географические объекты (например, гора, водоем, город);
- территории (например, область, задаваемая почтовым индексом);
- местоположения (например, перекресток, как специфическое место пересечения двух дорог).

Для включения режима совместимости преобразования значений гео-типов из текстового вида в бинарный (режим совместимости с PostgreSQL) необходимо ядро СУБД запускать с ключом /COMPATIBILITY=PGGEO или выполнить SQL-команду SET COMPATIBILITY 'PGGEO'. В данном режиме координаты могут быть разделены пробелами и запятыми, в стандартном режиме они могут быть разделены только пробелами. При этом для каждой точки допустимо указание до 4-х координат из многомерного пространства, реально используются из которых только первые две.

Примечание
Режим совместимости преобразования значений гео-типов из текстового вида в бинарный поддерживается со сборки 6.0.17.92.

Модель геометрических данных OpenGIS

Набор геометрических типов данных в спецификации для SQL основан на геометрической модели OpenGIS. В этой модели каждый геометрический объект обладает следующими свойствами:

- связан с системой координат, описывающей координатное пространство, в котором определен объект;
- принадлежит некоторому геометрическому классу.

Иерархия геометрических классов

Спецификация OpenGIS предусматривает следующую иерархию геометрических классов:

- Geometry
  - Point
  - Curve
    - LineString
      - LinearRing
  - Surface
    - Polygon
  - GeometryCollection
    - MultiPoint
    - MultiCurve
      - MultiLineString
    - MultiSurface
      - MultiPolygon

Класс Geometry – базовый класс (неинстанцируемый). Инстанцируемые подклассы Geometry ограничены размерностью 0, 1 и 2 и существуют в двумерном координатном пространстве. Все объекты инстанцируемых классов предполагаются топологически замкнутыми (то есть каждый объект включает собственную границу).

Подклассами Geometry являются классы:

1) размерности 0 – точка (Point);
2) размерности 1 – кривая (Curve) и ее подкласс LineString с подклассом LinearRing;
3) размерности 2 – поверхность (Surface) и ее подкласс Polygon;
4) составных объектов – набор объектов (GeometryCollection):
   - размерности 0 – MultiPoint (набор классов Point);
   - размерности 1 – MultiLineString (набор классов LineString);
   - размерности 2 – MultiPolygon (набор классов Polygon).

Примечание
Подклассы MultiCurve и MultiSurface введены как абстрактные суперклассы для обобщения интерфейсов подклассов Curve и Surface соответственно.

Geometry, Curve, Surface, MultiCurve и MultiSurface определены как неинстанцируемые классы. Они определяют общий набор методов для своих подклассов.

Point, LineString, Polygon, GeometryCollection, MultiPoint, MultiLineString, MultiPolygon являются инстанцируемыми классами (в иерархии объектов выделены жирным шрифтом).

Класс Geometry

Geometry – корневой класс иерархии. Каждый объект класса Geometry описывается множеством его свойств. Подклассы корневого класса Geometry наследуют свойства класса Geometry и имеют собственные свойства.

Свойства класса

Класс Geometry имеет следующие свойства:

1) тип, к которому принадлежит объект геометрического типа. Каждый объект геометрического типа принадлежит одному из инстанцируемых классов в иерархии;
2) идентификатор системы координат объекта (SRID) геометрического типа, описывающего координатное пространство, в котором определен объект геометрического типа;
3) координаты объекта геометрического типа в системе координат, представленные в виде вещественных чисел двойной точности (8 байтов, double). Все непустые объекты геометрического типа имеют не менее одной пары координат (X, Y). Пустые конфигурации не содержат координат;
4) внутренняя область, граница и внешняя область. Все объекты геометрического типа занимают некоторую позицию в пространстве. Внешняя область объекта геометрического типа – все пространство, не занятое им. Внутренняя область – пространство, занятое объектом геометрического типа. Граница – разделитель между внутренней областью геометрии и внешней;
5) минимальный ограничительный прямоугольник объекта геометрического типа. Это ограничительная граница вокруг объекта геометрического типа, сформированная минимумами и максимумами координат (X, Y):

```
((MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY))
```

6) простой (simple) или непростой (non-simple). Каждый геометрический класс определяет собственные критерии того, является ли принадлежащий ему объект простым или непростым;
7) замкнутый (closed) или не замкнутый (not closed). Значения геометрических объектов некоторых типов (LineString, MultiLineString) или замкнуты, или не замкнуты. Каждый геометрический класс определяет собственные критерии того, является ли принадлежащий ему объект замкнутым или не замкнутым;
8) пустой (empty) или не пустой (not empty). Объект геометрического типа пуст, если он не имеет точек. Внешняя область, внутренняя область и граница пустого объекта геометрического типа не определены, то есть они представлены NULL-значением. Пустой объект геометрического типа всегда простой. Пустой объект геометрического типа имеет площадь 0;

9) размерность объекта геометрического типа может иметь значение: -1, 0, 1 или 2:
   - размерность -1 установлена для пустых объектов геометрического типа;
   - размерность 0 установлена для объекта геометрического типа без длины и с нулевой площадью;
   - размерность 1 установлена для объекта геометрического типа с ненулевой длиной и нулевой площадью;
   - размерность 2 установлена для объекта геометрического типа с ненулевой площадью;
   - размерность подкласса Point равна 0;
   - размерность подкласса LineStrings равна 1;
   - размерность подкласса Polygon равна 2;
   - размерность объектов классов MultiPoint, MultiLineString и MultiPolygon та же самая, что и у элементов, из которых они состоят.

Класс Point

Класс представляет точку в пространстве координат.

Свойства класса

Класс Point имеет следующие свойства:

1) X-координата;
2) Y-координата;
3) граница Point – пустое множество;
4) размерность Point равна нулю.

Класс Curve

Класс определяет простой одномерный геометрический объект, представляемый последовательностью точек. Подкласс Curve определяет тип интерполяции линии, соединяющей точки. Curve – неинстанцируемый класс, единственный инстанцируемый подкласс – LineString (ломаная).

Свойства класса

Класс Curve имеет следующие свойства:

1) координаты точек кривой;
2) размерность кривой равна 1;
3) кривая проста, если не проходит через одну и ту же точку дважды (исключая случай замкнутой кривой);
4) кривая замкнута, если ее начальная точка равна ее конечной точке;
5) граница замкнутой кривой – пустое множество;
6) граница незамкнутой кривой состоит из ее начальной и конечной точки;
7) кривая, которая является простой и замкнутой – кольцо (Ring).

Примечание
В текущей версии не поддерживается.

Класс LineString

Класс LineString (ломаная) определяет кривую (Curve) с линейной интерполяцией линий, соединяющих точки.

Свойства класса

Класс LineString имеет следующие свойства:

1) координаты сегментов LineString определяются последовательностями пар точек;
2) LineString является линией (Line), если состоит из двух точек;
3) LineString является LinearRing, если кривая замкнута и проста.

Класс Surface

Класс Surface (поверхность) определяет геометрический объект с размерностью 2.

Свойства класса

Класс Surface имеет следующие свойства:

1) размерность Surface равна 2;
2) спецификация OpenGIS определяет класс Surface как плоскую поверхность, состоящую из единственной ломаной, образующей «внешнюю границу» и из нулевого или большего количества ломаных, образующих «внутреннюю границу»;
3) границы плоской поверхности – набор замкнутых кривых (Curve), соответствующих ее внешней и внутренней областям.

Единственным инстанцируемым подклассом класса Surface является класс Polygon.

Примечание
В текущей версии не поддерживается.

Класс Polygon

Класс Polygon (многоугольник) определяет плоскую поверхность (Surface) с одной внешней границей и нулем или большим количеством внутренних границ. Каждая внутренняя граница определяет отверстие в Polygon.

Свойства класса

Класс Polygon имеет следующие свойства:

1) граница Polygon состоит из набора LinearRings (простой и замкнутый вариант LineString), которые составляют его внешние и внутренние границы;
2) никакие два кольца в Polygon не могут пересекаться, кроме как по тангенсу (касаться);
3) Polygon не должен содержать вырезов в виде линии, выбросов или «проколов»;

4) внутренняя область каждого Polygon – связанный набор точек;

5) внешняя область Polygon с одним или более отверстиями представляет собой несвязанную область. Каждое отверстие определяет связанный компонент внешней области.

Все объекты класса Polygon, удовлетворяющие перечисленным выше свойствам, являются простыми объектами.

Примеры многоугольников с 1, 2 и 3 кольцами соответственно (рисунок 1).

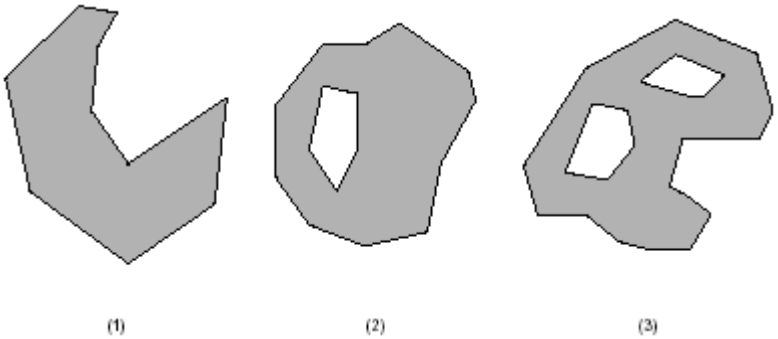


Рисунок 1. Примеры многоугольников

Класс GeometryCollection

Класс GeometryCollection определяет геометрический тип, который является набором из нуля и более объектов других геометрических типов.

Все элементы в GeometryCollection должны быть в одной и той же системе координат. Класс GeometryCollection не имеет других ограничений на свои элементы.

Свойства класса

Подклассы GeometryCollection могут иметь ограничения, основанные на:

- типе элементов;
- размерности;
- степени пространственного перекрытия между элементами.

Класс MultiPoint

Класс MultiPoint (набор точек) определяет набор графических объектов типа точка (Point). Точки в наборе не связаны между собой и не упорядочены.

Свойства класса

Класс MultiPoint имеет следующие свойства:

1) нулевая размерность;

2) класс является простым, если никакие две точки в нем не равны (то есть не имеют идентичных координат);

3) граница класса – пустое множество.

Класс MultiCurve

Класс MultiCurve (набор кривых) определяет набор графических объектов типа кривая (Curve). MultiCurve – неинстанцируемый класс.

Свойства класса

Класс MultiCurve имеет следующие свойства:

1) класс является простым, только если все его элементы просты, а одиночные пересечения между любыми двумя элементами происходят в точках, находящихся на границах обоих элементов;
2) граница класса определяется с помощью применения правила объединения «mod 2»: точка находится в границе класса, если она расположена на границе нечетного числа элементов класса;
3) размерность класса равна 1;
4) класс замкнут, если все его элементы замкнуты;
5) граница закрытого класса – всегда пустое множество.

Примечание
В текущей версии не поддерживается.

Класс MultiLineString

Класс MultiLineString (набор ломаных) определяет набор графических объектов типа LineString.

Класс MultiSurface

Класс MultiSurface (набор поверхностей) определяет набор графических объектов типа поверхность (Surface). MultiSurface – неинстанцируемый класс.

Свойства класса

Класс MultiSurface имеет следующие свойства:

1) внутренние области любых двух поверхностей не могут пересекаться;
2) границы любых двух элементов могут пересечься не более, чем в конечном числе точек.

Единственный инстанцируемый подкласс класса MultiSurface – это MultiPolygon.

Примечание
В текущей версии не поддерживается.

Класс MultiPolygon

Класс MultiPolygon (набор многоугольников) определяет набор графических объектов типа многоугольник (Polygon).

Свойства класса

Класс MultiPolygon имеет следующие свойства:

1) внутренние поверхности двух многоугольников, которые являются элементами MultiPolygon, не могут пересечься;
2) границы любых двух многоугольников, которые являются элементами MultiPolygon, не могут пересекаться и могут касаться только в конечном числе точек;
3) MultiPolygon не может содержать вырезов линий, выбросов или проколов: MultiPolygon – правильный, замкнутый набор точек;
4) внутренняя область MultiPolygon, содержащая более, чем один многоугольник, не связана, число связанных частей внутренней области MultiPolygon равно числу многоугольников в MultiPolygon;
5) размерность класса равна 2;
6) граница MultiPolygon – набор закрытых кривых (LineStrings), образованных границами его элементов – многоугольников;
7) каждая кривая (Curve) на границе MultiPolygon находится на границе точно одного многоугольника, и каждая кривая (Curve) на границе многоугольника находится также и на границе MultiPolygon.

Форматы геометрических типов данных

Представление (описание) геометрических типов данных возможно двумя способами:

1) в текстовом виде – WKT-формат (Well-Known Text);
2) в двоичном виде – WKB-формат (Well-Known Binary).

Текстовый формат

Текстовое представление данных (WKT-формат) определяет формат в виде текстовой строки, содержащей:

1) имя типа объекта (Point, Linestring, Polygon, Multipoint, Multilinestring, Multipolygon, Geometrycollection, Box, Circle);
2) пары чисел как координаты точек;
3) скобки для группировки элементов;
4) символы табуляции и перевода строки.

Синтаксис WKT-представления

[1] <WKT-представление геометрического объекта>::=
<WKT-представление точки>
|<WKT-представление прямой линии>
|<WKT-представление ломаной линии>
|<WKT-представление многоугольника>
|<WKT-представление набора точек>
|<WKT-представление набора ломаных линий>
|<WKT-представление набора многоугольников>
|<WKT-представление GeometryCollection Tagged Text>
|<WKT-представление прямоугольника>
|<WKT-представление круга>

[2] <WKT-представление точки>::=
POINT <описание точки>| <описание точки>:: POINT

[3] <WKT-представление прямой линии>::=
LINE <описание прямой линии>| <описание прямой линии>:: LINE

[4] <WKT-представление ломаной линии>::=
LINESTRING <описание ломаной линии>
|<описание ломаной линии>:: LINESTRING

[5] <WKT-представление многоугольника>::=
POLYGON <описание многоугольника>

[6] <WKT-представление набора точек>::=
MULTIPOINT <описание набора точек>

[7] <WKT-представление набора ломаных линий>::=
MULTILINESTRING <описание набора ломаных линий>

[8] <WKT-представление набора многоугольников>::=
MULTIPOLYGON <описание набора многоугольников>

[9] <WKT-представление набора геометрических объектов>::=
GEOMETRYCOLLECTION <описание набора геометрических объектов>

[10] <WKT-представление прямоугольника>::=
BOX <описание прямоугольника>

[11] <WKT-представление круга>::=
CIRCLE <описание круга>

[12] <описание точки>::=
{EMPTY |([<координаты точки>])}

[13] <описание прямой линии>::=
{EMPTY | (<пара точек>)}
[14] <описание ломаной линии>::=
{EMPTY | (<описание точки> {, <описание точки>...})}
[15] <описание многоугольника>::=
{EMPTY | (<описание ломаной линии> {, <описание ломаной линии>...})}
[16] <описание набора точек>::=
{EMPTY | (<описание точки> {, <описание точки>...})}
[17] <описание набора ломаных линий>::=
{EMPTY | (<описание ломаной линии>
{, <описание ломаной линии>...})}
[18] <описание набора многоугольников>::=
{EMPTY | (<описание многоугольника> {, <описание многоугольника>...})}
[19] <описание набора геометрических объектов>::=
{EMPTY | (<WKT-представление геометрического объекта>
{,<WKT-представление геометрического объекта>...})}
[20] <описание прямоугольника>::=
{EMPTY | (<пара точек>)}
[21] <описание круга>::=
{EMPTY | (<описание точки> <радиус>)}
[22] <координаты точки>::=
<x> <y> | (<x> <y>) | (<x>, <y>)
[23] <пара точек>::=
<описание точки> <описание точки> | <описание точки>, <описание точки>
[24] <радиус>::= вещественный литерал
[25] <y>::= вещественный литерал
[26] <x>::= вещественный литерал

Примеры WKT-представления

1) Point:

```
POINT(10 10)
```

2) LineString с тремя точками:

```
LINESTRING(10 10, 20 20, 30 40)
LINESTRING((10,10), (20,20), (30,40))
```

3) Polygon с одной внешней и без внутренних границ:

```
POLYGON((10 10, 10 20, 20 20, 20 15, 10 10))
POLYGON((10 10), (10 20), (20 20), (20 15), (10 10))
POLYGON((10 10), 10 20, (20 20), (20 15), 10 10))
```

4) MultiPoint из двух точек:

```
MULTIPOINT(10 10, 20 20)
MULTIPOINT((10 10), (20 20))
MULTIPOINT(10 10, (20 20))
```

5) MultiLineString из двух ломаных линий:

```
MULTILINESTRING((10 10, 20 20), (15 15, 30 15))
MULTILINESTRING(((10 10), (20 20)), ((15 15), 30 15))
```

6) MultiPolygon из двух многоугольников:

```
MULTIPOLYGON(((10 10, 10 20, 20 20, 20 15, 10 10)), ((60 60, 70 7,
 80 60, 60 60 )))
```


7) GeometryCollection из двух Point и одной LineString:

```
GEOMETRYCOLLECTION(POINT(10 10), POINT(30 30), LINESTRING(15 15,
 20 20))
```

Двоичный формат

Двоичное представление данных (WKB-формат) определяет формат в виде структур данных, описание которых содержится, как правило, в заголовочном файле приложения (см. пример). Каждый элемент содержит спецификатор порядка байт (для работы на различных аппаратных платформах). Все коды элементов и их компонентов выражаются 4-байтовыми беззнаковыми целыми числами, координаты – вещественными числами двойной точности.

Пример заголовочного файла для описания данных в WKB-формате

Основные типы WKB

```
// byte   : 8-bit unsigned integer (1 byte)
// uint32 : 32-bit unsigned integer (4 bytes)
// double : double precision number (8 bytes)
enum wkbGeometryType
{
wkbPoint              = 1,
wkbLineString         = 2,
wkbPolygon            = 3,
wkbMultiPoint         = 4,
wkbMultiLineString    = 5,
wkbMultiPolygon       = 6,
wkbGeometryCollection = 7
}
enum wkbByteOrder
{
wkbXDR = 0,    // Прямой порядок байтов
wkbNDR = 1     // Обратный порядок байтов
}
```

Примечание
В двоичном формате СУБД ЛИНТЕР поддерживает только прямой порядок байтов.

Вспомогательные типы WKB

```
// Building Blocks : Point, LinearRing
Point
{
        double  x;
        double  y;
}
LinearRing
{
```

```
        uint32  numPoints;
        Point   points[numPoints];
}
```

WKB-представление геометрических типов

```
WKBPoint
{
        byte    byteOrder;
        uint32  wkbType;     // 1
        Point   point;
}

WKBLineString
{
        byte    byteOrder;
        uint32  wkbType;     // 2
        uint32  numPoints;
        Point   points[numPoints];
}

WKBPolygon
{
        byte       byteOrder;
        uint32     wkbType;  // 3
        uint32     numRings;
        LinearRing rings[numRings];
}

WKBMultiPoint
{
        byte       byteOrder;
        uint32     wkbType;  // 4
        uint32     num_wkbPoints;
        WKBPoint   WKBPoints[num_wkbPoints];
}

WKBMultiLineString
{
        byte          byteOrder;
        uint32        wkbType;    // 5
        uint32        num_wkbLineStrings;
        WKBLineString WKBLineStrings[num_wkbLineStrings];
}

WKBMultiPolygon
{
```



```
        byte       byteOrder;
        uint32     wkbType;    // 6
        uint32     num_wkbPolygons;
        WKBPolygon wkbPolygons[num_wkbPolygons];
}

WKBCircle
{
        byte    byteOrder;
        uint32  wkbType;
        Point   point;
        double  r;           // радиус
}

WKBGeometry
{
        union
        {
                WKBPoint              point;
                WKBLineString         linestring;
                WKBPolygon            polygon;
                WKBGeometryCollection collection;
                WKBMultiPoint         mpoint;
                WKBMultiLineString    mlinestring;
                WKBMultiPolygon       mpolygon;
        }
}

WKBGeometryCollection
{
        byte        byte_order;
        uint32      wkbType;    // 7
        uint32      num_wkbGeometries;
        WKBGeometry wkbGeometries[num_wkbGeometries];
}
```

Примечание

Тип данных BOX не имеет собственного идентификатора типа (wkbType), так как топологически не отличается от POLYGON, а типы CIRCLE и LINE – имеют: `wkbCircle = 0x81, wkbLine = 0x82.`

Пример WKB-формата

WKB-представление точки POINT (1,1) – последовательность из 21 байта:

```
01010000000000000000F03F000000000000F03F
```

где:

```
Byte order: 01
```

```
WKB type  : 01000000
X         : 000000000000F03F
Y         : 000000000000F03F
```


Создание/добавление столбцов геометрического типа

Добавление столбца с геометрическим типом данных при создании таблицы выполняется с помощью стандартного SQL-оператора `CREATE OR REPLACE TABLE`.

В спецификации типа данных столбца задается один из допустимых геометрических типов:

`POINT`	точка
`LINESTRING[ (n) ]`	ломаная линия
`POLYGON[ (n) ]`	многоугольник
`MULTIPOINT[ (n) ]`	набор точек
`MULTILINESTRING[ (n) ]`	набор ломаных линий
`MULTIPOLYGON[ (n) ]`	набор многоугольников
`BOX[ (n) ]`	прямоугольник
`LINE`	прямая линия
`CIRCLE`	круг
`GEOMETRYCOLLECTION[ (n) ]`	набор геометрических объектов
`GEOMETRY`	обобщенный геометрический тип (может содержать любой геометрический объект)

Синтаксические правила

1) Формат хранения значений типа LINE совпадает с форматом LINESTRING, имеющим две точки.
2) Все геометрические типы данных унаследованы от типа varbyte и имеют максимальную длину 4000 байт. В связи с этим для типов с переменной длиной значений (Linestring, Polygon, Multipoint, Multilinestring, Geometrycollection) установлено ограничение: по умолчанию размер значений перечисленных типов составляет 1024 байт. Для того чтобы создавать значения с большим размером, надо при создании таблицы явно указать этот размер, например:

```
create or replace table LINESTRING_TEST(LS LINESTRING(4000));
```

3) Тип данных BOX определяет прямоугольник. Две точки, которые являются вершинами прямоугольника, задают координаты диагонали. Данный формат автоматически преобразуется в формат POLYGON с пятью точками:

   конструкция `BOX(1 1, 2 2)` эквивалентна `POLYGON (1 1,1 2,2 2,2 1,1 1)`.

4) Тип данных LINE определяет прямую линию. Формат хранения этого типа эквивалентен формату хранения ломаной линии с двумя точками:

   конструкция `LINE(1 1, 2 2)` имеет то же WKB-представление (за исключением числа, задающего тип), что и `LINESTRING (1 1, 2 2)`.

5) Тип данных CIRCLE определяет круг: `CIRCLE(1 1, 2)` – круг с координатой центральной точки (1,1) и с радиусом 2.
6) В столбец с типом данных GEOMETRYCOLLECTION можно загружать все типы геометрических данных, в столбец MULTIPOINT – данные типа POINT, в столбец

MULTILINESTRING – данные типа LINESTRING, в столбец MULTIPOLYGON – данные типа POLYGON.

7) Добавление столбца с геометрическим типом данных в уже существующую таблицу выполняется с помощью стандартного SQL-оператора `ALTER TABLE ADD COLUMN`:

```
ALTER TABLE GEO_TEST ADD COLUMN PL POLYGON;
```

8) Для столбца с обобщенным типом данных GEOMETRY может применяться любая геометрическая функция, однако, если хранящийся в таком столбце геометрический объект не является корректным для вызываемой функции, будет выдаваться соответствующий код завершения СУБД ЛИНТЕР.

9) CHECK-условия запрещены для BLOB-столбцов с геометрическими типами данных.

Геометрические BLOB-данные

Создаваемые геометрические данные имеют тип данных VARBYTE/GEOMETRY с максимальным размером 4000 байт, что во многих случаях не позволяет создавать и хранить в БД сложные геометрические объекты больше указанной размерности. В этом случае геометрические данные большой размерности можно хранить в виде BLOB-значений.

Поддерживается создание одностолбцовых индексов для BLOB-столбцов геометрических типов и использовать предикаты сравнения (>, <, =, <>, >= и <=). Индекс модифицируется при каждой модификации BLOB-значения.

Чтобы создаваемые геометрические данные хранились в БД в виде BLOB-значений необходимо перед созданием таблиц с геометрическими данными выполнить команду

```
SET SESSION blob geometry storage On;
```

(см. раздел «Управление вводом геометрических данных»)

В результате в текущей сессии во всех каналах геометрические объекты большой размерности (LINESTRING, POLYGON, MULTIPOINT, MULTILINESRING, MULTIPOLYGON, GEOMETRYCOLLECTION, GEOMETRY) будут храниться в БД в виде BLOB-значений (геометрические данных малой размерности (типа POINT) по прежнему будут представлены в виде VARBYTE-значений).

Например, в результате выполнения запроса

```
CREATE TABLE GTEST (GEOM GEOMETRY);
```

столбец GEOM будет иметь тип (BLOB/GEOMETRY), а не (VARBYTE(1028)/ GEOMETRY).

Для управления максимально допустимым размером геометрических BLOB-данных используется команда

```
alter database set blob size limit <максимальный_размер>;
```

(см. раздел «Управление вводом геометрических данных»).

Загрузка геометрических данных

Загрузка геометрических данных в таблицу БД выполняется с помощью стандартного SQL-оператора INSERT и набора встроенных в СУБД функций, преобразующих WKT/WKB-формат представления данных в соответствующий столбцу геометрический тип данных, либо с помощью прямой типизации данных (когда тип можно определить из контекста), например:

```
insert into GEOTABLE(COLUMN_WITH_TYPE_POINT) values ('POINT (1
 1)');
```

Примечание

В связи с добавлением большого числа новых функций их названия могут случайно совпасть с уже существующими в БД именами столбцов и таблиц. Чтобы исключить эту неоднозначность, в СУБД ЛИНТЕР предусмотрен режим совместимости с предыдущими версиями. Для включения этого режима следует запустить ядро СУБД с ключом /COMPATIBILITY=GEOPREFIX. В этом случае к именам геометрических функций добавляется префикс "LIN_". Например, стандартные имена геометрических SQL-функций AsText() и PointFromText() будут заменены на имена Lin_AsText и Lin_PointFromText:

```
select lin_AsText(lin_PointFromText('POINT (1 1)'));
|POINT (1 1) |
```

Загрузка геометрических BLOB-данных также выполняется обычным способом, например, с помощью стандартного SQL-запроса INSERT.

```
INSERT INTO GTEST(GEOM)
 VALUES(GEOMFROMWKB(hex('01030000000100000005000000000000000000244000000
0000000024400000000000002440000000000000494000000000000049400000000000004940000000000000494000000000000024400000000000002440000000000000244000000000000024400000000000002440'),
 1234));
```

либо с помощью команды интерфейса нижнего уровня (call-интерфейса) ABLB. Например, вышеприведенная INSERT-команда в утилите INL (использующей ABLB) выглядит следующим образом:

```
INSERT INTO GTEST(GEOM) VALUES(NULL);
blob insert column=1
0103000000010000000500000000000000000024400000000000002440000000000000244000000000000024400000000000004940000000000000494000000000000049400000000000004940000000000000244000000000000024400000000000002440D2040000;
```

Ограничения и особенности при добавлении геометрического BLOB-значения с помощью команды ABLB:

1) в конец буфера со значением надо добавлять значение SRID (4 байта);
2) проверка на корректность введённого значения не проверяется (т.к. значение заносится в БД порциями).

   В этом случае для проверки корректности введённого значения надо воспользоваться функцией GEOISVALID, например, выполнить запрос вида:

```
select GEOISVALID(GEOM) from GTEST;
```

(запрос вернёт 1 для корректных значений и 0 – для некорректных);

3) геометрические BLOB-значения нельзя передавать в качестве параметров в SELECT-запросах.

   Т.е. если, например, требуется выполнить запрос вида

```
select UNION(GEOM, ?) from GTEST;
```

   то можно передать ему в качестве параметра байтовую последовательность длиной не более 4000 байт, например, такую:

```
0103000000010000000500000000000000000024400000000000002440000000000000244000000000000024
4000000000000049400000000000004940000000000000494000000000000049400000000000004940000000
0000000244000000000000024400000000000002440D2040000
```

Загрузка с помощью WKT-формата

Создание значений специфических геометрических типов (за исключением BOX, LINE, CIRCLE) обеспечивается с помощью соответствующих каждому из типов функций.

Создание точки

Функция

Преобразование WKT-представления точки в соответствующий геометрический тип данных.

Спецификация

`PointFromText` (`<wkt>`[,`<srid>`])

`<wkt>` – <WKT-представление точки> | <SQL-параметр>;
`<srid>` – идентификатор системы координат данной точки.

Синтаксические правила

Если аргумент `<srid>` не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа POINT, соответствующее внутреннему представлению в БД типа данных POINT (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1)

```
create or replace table  geo_test (p point);
insert into geo_test(p)values (pointfromtext('point(1.45 3.06)'));
insert into geo_test values (pointfromtext('point 1.45 3.06', 1));
insert into geo_test(p) values (pointfromtext('point(25 67)',
 45));
```

2) sql-скрипт:

```
create or replace table geo_test (geom GEOMETRY null);
insert into geo_test (geom) values (PointFromText(?)),
 (PointFromText(:param));
POINT(10 5)
POINT(12 4)
```

Создание ломаной линии

Функция

Преобразование WKT-представления ломаной линии в соответствующий геометрический тип данных.

Спецификация

`LineFromText|LineStringFromText` (<wkt>[,<srid>])

<wkt> – <WKT-представление ломаной линии> | <SQL-параметр>;
<srid> – идентификатор системы координат данной ломаной линии.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа LINESTRING, соответствующее внутреннему представлению в БД типа данных LINESTRING (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1)

```
create or replace table  geo_test (l linestring);
insert into geo_test(l)
values (linefromtext('linestring (1 1,2 2,1 2,5 7)'));
insert into geo_test
values (linestringfromtext('linestring ((2 2), (4 5), (8
 15))',3));
```

2)

```
CREATE OR REPLACE TABLE LS_TEST(L LINESTRING);
INSERT INTO LS_TEST
VALUES(LineFromText('LINESTRING (2 2,1 0,10.1 2)'));
INSERT INTO LS_TEST
VALUES('((0 1),3 3,(0 0),(10 10) 11 21)'::LINESTRING);
INSERT INTO LS_TEST VALUES('LINESTRING (1 2,4 1,67 85)');
```

3)

```
create or replace table  geo_test (l linestring(2048));
insert into geo_test(l) values (linefromtext(?));
linestring (1 1, 2 2, 1 2, 5 7)
insert into geo_test(l) values (LineStringFromText(:param,2));
linestring ((2 2), (4 5), (8 15))
```

Создание многоугольника

Функция

Преобразование WKT-представления многоугольника в соответствующий геометрический тип данных.

Спецификация

`PolyFromText` | `PolygonFromText` (<wkt>[,<srid>])

<wkt> – <WKT-представление многоугольника> | <SQL-параметр>;
<srid> – идентификатор системы координат данного многоугольника.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа POLYGON, соответствующее внутреннему представлению в БД типа данных POLYGON (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1)

```
create or replace table  geo_test (p polygon);
insert into geo_test(p) values (PolyFromText('polygon((2 2,1 1,0
 0,10 0,2 2))'));
insert into geo_test(p) values (PolyFromText('polygon(((2 2),(1
 1),(0 0),(10 0),(2 2)))'));
insert into geo_test(p) values ('((0 0,0 3,3 3,3 0,0 0), (0 0,1
 1,1 0.55,0 0))'::polygon);
insert into geo_test(p) values ('POLYGON (1 2,4 1,67 85,1 2)');
insert into geo_test(p) values (PolygonFromText('polygon(((2 2),1
 1,(0 0),(10 0),2 2))'));
```

2)

```
create or replace table  geo_test (p polygon(4000));
insert into geo_test(p) values (PolyFromText(?));
polygon((2 2,1 1,0 0,10 0,2 2))
insert into geo_test(p) values (PolygonFromText(:param));
polygon(((2 2),1 1,(0 0),(10 0),2 2))
```

Создание группы точек

Функция

Преобразование WKT-представления группы точек в соответствующий геометрический тип данных.

Спецификация

`MPointFromText | MultiPointFromText (<wkt>[,<srid>])`

`<wkt>` – <WKT-представление группы точек> | <SQL-параметр>;
`<srid>` – идентификатор системы координат данного набора точек.

Синтаксические правила

Если аргумент `<srid>` не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа MULTIPOINT, соответствующее внутреннему представлению в БД типа данных MULTIPOINT (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1)

```
create or replace table  geo_test (mpo multipoint);
insert into geo_test(mpo) values (MPointFromText('multipoint (2
 2,1 1,2 2,1 0,0 0)'));
insert into geo_test(mpo) values ('multipoint (2 2,1 1,2 2,1 0,0
 0)');
insert into geo_test(mpo) values (MPointFromText('multipoint ((2
 2),(1 1),2 2,(1 0),0 0)'));
insert into geo_test(mpo) values (MultiPointFromText('multipoint
 ((2 2),(1 1),2 2,(1 0),0 0)'));
```

2)

```
create or replace table  geo_test (mpo multipoint(2048));
insert into geo_test(mpo) values (MPointFromText(?));
multipoint (2 2,1 1,2 2,1 0,0 0)
insert into geo_test(mpo) values (MultiPointFromText(:param));
multipoint ((2 2),(1 1),2 2,(1 0),0 0)
```

Создание группы ломаных линий

Функция

Преобразование WKT-представления группы ломаных линий в соответствующий геометрический тип данных.

Спецификация

`MLineFromText | MultiLineStringFromText (<wkt>[,<srid>])`

<wkt> – <WKT-представление группы ломаных линий> | <SQL-параметр>;
<srid> – идентификатор системы координат данной группы ломаных линий.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа MULTILINESTRING, соответствующее внутреннему представлению в БД типа данных MULTILINESTRING (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1)

```
create or replace table  geo_test (mls multilinestring);
insert into geo_test(mls) values
 (MLineFromText('multilinestring((2 2,1 1),(0 0,10 0))'));
insert into geo_test(mls) values
 (MultiLineStringFromText('multilinestring((2 2,(1 1)),((0 0),(10
 0)))'));
insert into geo_test(mls) values ('((0 0,0 3,3 0),(0 0, 0 100, 1
 1, 1 0.55))'::multilinestring);
insert into geo_test(mls) values ('multilinestring((1 2,4 1),(67
 85,1 2))');
```

2)

```
create or replace table  geo_test (mls multilinestring(2048));
insert into geo_test(mls) values (MLineFromText(?));
multilinestring((2 2,1 1),(0 0,10 0))
insert into geo_test(mls) values
 (MultiLineStringFromText(:param));
multilinestring((2 2,(1 1)),((0 0),(10 0)))
```

Создание группы многоугольников

Функция

Преобразование WKT-представления группы многоугольников в соответствующий геометрический тип данных.

Спецификация

MPolyFromText | MultiPolygonFromText (<wkt>[,<srid>])

<wkt> – <WKT-представление группы многоугольников> | <SQL-параметр>;
<srid> – идентификатор системы координат данной группы многоугольников.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа MULTIPOLYGON, соответствующее внутреннему представлению в БД типа данных MULTIPOLYGON (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1)

```
create or replace table  geo_test (mp multipolygon);
insert into geo_test(mp) values (MPolyFromText('multipolygon (((2
 2,1 1,0 0,10 0,2 2)))'));
insert into geo_test(mp) values
 (MultiPolygonFromText('multipolygon((2 2,(1 1),3 0,(2 2)),(0 0,
 (0 3),3 3,(3 0))'));
insert into geo_test(mp) values ('((0 0,0 3,3 3,3 0,0 0),(0 0,1
 1,1 0.55,0 0))'::multipolygon);
insert into geo_test(mp) values ('multipolygon ((((1 2), (4 1), 67
 85, (1 2))))');
```

2)

```
create or replace table  geo_test (mp multipolygon(2048));
insert into geo_test(mp) values (MPolyFromText(?));
multipolygon (((2 2,1 1,0 0,10 0,2 2)))
insert into geo_test(mp) values (MultiPolygonFromText(:param));
multipolygon((2 2,(1 1),3 0,(2 2)),(0 0, (0 3),3 3,(3 0))
```

Создание группы геометрических объектов

Функция

Преобразование WKT-представления группы геометрических объектов в соответствующий геометрический тип данных.

Спецификация

`GeomCollFromText` (<wkt>[,<srid>])

<wkt> – <WKT-представление группы геометрических объектов> | <SQL-параметр>;
<srid> – идентификатор системы координат данной группы геометрических объектов.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа GEOMETRYCOLLECTION, соответствующее внутреннему представлению в БД типа данных GEOMETRYCOLLECTION (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Пример

```
create or replace table  geo_test (gcl geometrycollection);
```



```
insert into geo_test(gcl) values
 (GeomCollFromText('geometrycollection(linestring((2 2),(1 0)),
polygon( ( 0 0,0 3,3 0,0 0), (0 0, 1 1, 1 0.55, 0 0) ) )'));
insert into geo_test(gcl) values ('(point(1 1), linestring(1 4,2
 5))'::geometrycollection);
insert into geo_test(gcl) values ('geometrycollection (point(1
 1),point(10 10),point(100 100))');
```

Создание прямоугольника

Создание прямоугольника (многоугольника со сторонами, параллельными осям координат), выполняется с помощью простой типизации объекта, например:

```
create or replace table box_test(b box);
insert into box_test values('box (1 2, 4 1)');
insert into box_test values('((0 1,3 3))'::box);
```

Пример

```
create or replace table box_test(b box(2048));
insert into box_test values(polyfromtext('box (2 2,1 0)'));
insert into box_test values(polyfromtext(?));
box (2 2,1 0)
insert into box_test values(polygonfromtext(:param));
box (1 2, 4 1)
```

Создание прямой линии

Создание прямой линии выполняется с помощью простой типизации объекта, например:

```
CREATE OR REPLACE TABLE LINE_TEST(L LINE);
INSERT INTO LINE_TEST VALUES('((0 1),3 3)'::LINE);
INSERT INTO LINE_TEST VALUES('LINE (1 2,4 1)');
```

Создание круга

Создание круга выполняется с помощью простой типизации объекта, например:

```
CREATE OR REPLACE TABLE CIRCLE_TEST( CR CIRCLE );
INSERT INTO CIRCLE_TEST VALUES( 'CIRCLE (1 1,1)' );
INSERT INTO CIRCLE_TEST VALUES( '(1 2) 3'::CIRCLE );
```

Создание произвольного геометрического объекта

Функция

Преобразование WKT-представления любого геометрического объекта в соответствующий геометрический тип данных.

Спецификация

`GeomFromText | GeometryFromText (<wkt>[,<srid>])`

<wkt> – <WKT-представление геометрического <объекта> | <SQL-параметр>;
<srid> – идентификатор системы координат данного геометрического объекта.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение инстанцируемого геометрического типа, содержащееся в строковом аргументе этой функции, соответствующее внутреннему представлению в БД полученного при преобразовании геометрического объекта.
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1)

```
create or replace table geom_test (g geometry);
create index g on geom_test;
insert into geom_test(g) values (geomfromtext('point (1
 1)',1001));
insert into geom_test(g) values (geomfromtext('linestring (1 1,2
 2,3 3)',1001));
```

2)

```
create or replace table geom_test (g geometry(2048));
insert into geom_test(g) values (geomfromtext(?));
point (1 1)
insert into geom_test(g) values (geomfromtext(:param,5));
linestring (1 1,2 2,3 3)
```

Загрузка с помощью WKB-представления

Создание значений специфических геометрических типов (за исключением BOX, LINE, CIRCLE) обеспечивается с помощью соответствующих каждому из типов функций.

Создание точки

Функция

Преобразование WKB-представления точки в соответствующий геометрический тип данных.

Спецификация

PointFromWKB (<wkb>[,<srid>])

<wkb> – <WKB-представление точки> | <SQL-параметр>;
<srid> – идентификатор системы координат данной точки.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа POINT, соответствующее внутреннему представлению в БД типа данных POINT (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1) Программа клиентского приложения должна сама правильно сформировать WKB-представление, например:

```
!point (1 1)
create or replace table point_test(p point);
insert into point_test
values
(pointfromwkb(hex('0101000000000000000000f03f000000000000f03f')));
```

2)

```
create or replace table point_test(p point);
insert into point_test values (pointfromwkb(hextoraw(?)));
0101000000000000000000f03f000000000000f03f
```

Создание ломаной линии

Функция

Преобразование WKB-представления ломаной линии в соответствующий геометрический тип данных.

Спецификация

`LineFromWKB | LineStringFromWKB (<wkb>[,<srid>])`

<wkb> – <WKB-представление ломаной линии> | <SQL-параметр>;
<srid> – идентификатор системы координат данной ломаной линии.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа LINESTRING, соответствующее внутреннему представлению в БД типа данных LINESTRING (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1)

```
! line (1 1, 2 2)
create or replace table line_test(p linestring);
insert into line_test
values (linefromwkb
(hex('0102000000020000000000000000F03F000000000000F03F0000000000000
040000000000000004000000000')));
```

2)

```
create or replace table line_test(p linestring);
insert into line_test values (linefromwkb(:param));

Значение параметра:
0102000000020000000000000000F03F000000000000F03F0000000000000040000
000000000004000000000
```

Создание многоугольника

Функция

Преобразование WKB-представления многоугольника в соответствующий геометрический тип данных.

Спецификация

`PolyFromWKB` | `PolygonFromWKB` (<wkb>[,<srid>])

<wkb> – <WKB-представление многоугольника> | <SQL-параметр>;
<srid> – идентификатор системы координат данного многоугольника.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа POLYGON, соответствующее внутреннему представлению в БД типа данных POLYGON (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Создание группы точек

Функция

Преобразование WKB-представления группы точек в соответствующий геометрический тип данных.

Спецификация

`MPointFromWKB` | `MultiPointFromWKB` (<wkb>[,<srid>])

<wkb> – <WKB-представление группы точек> | <SQL-параметр>;
<srid> – идентификатор системы координат данного набора точек.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа MULTIPOINT, соответствующее внутреннему представлению в БД типа данных MULTIPOINT (в случае нормального завершения).

2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Создание группы ломаных линий

Функция

Преобразование WKB-представления группы ломаных линий в соответствующий геометрический тип данных.

Спецификация

`MLineFromWKB` | `MultiLineStringFromWKB` (<wkb>[,<srid>])

<wkb> – <WKB-представление группы ломаных линий> | <SQL-параметр>;
<srid> – идентификатор системы координат данной группы ломаных линий.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа MULTILINESTRING, соответствующее внутреннему представлению в БД типа данных MULTILINESTRING (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Создание группы многоугольников

Функция

Преобразование WKB-представления группы многоугольников в соответствующий геометрический тип данных.

Спецификация

`MPolyFromWKB` | `MultiPolygonFromWKB` (<wkb>[,<srid>])

<wkb> – <WKB-представление группы многоугольников> | <SQL-параметр>;
<srid> – идентификатор системы координат данной группы многоугольников.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа MULTIPOLYGON, соответствующее внутреннему представлению в БД типа данных MULTIPOLYGON (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Создание группы геометрических объектов

Функция

Преобразование WKB-представления группы геометрических объектов в соответствующий геометрический тип данных.

Спецификация

`GeomCollFromWKB` (<wkb>[,<srid>])

<wkb> – <WKB-представление группы геометрических объектов> | <SQL-параметр>;
<srid> – идентификатор системы координат данной группы геометрических объектов.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение типа GEOMETRYCOLLECTION, соответствующее внутреннему представлению в БД типа данных GEOMETRYCOLLECTION (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Создание произвольного геометрического объекта

Функция

Преобразование WKB-представления любого геометрического объекта в соответствующий геометрический тип данных.

Спецификация

`GeomFromWKB` | `GeometryFromWKB` (<wkb>[,<srid>])

<wkb> – <WKB-представление геометрического <объекта> | <SQL-параметр>;
<srid> – идентификатор системы координат данного геометрического объекта.

Синтаксические правила

Если аргумент <srid> не задан, по умолчанию используется значение 0.

Возвращаемое значение

1) Значение инстанцируемого геометрического типа, содержащееся в аргументе этой функции и соответствующее внутреннему представлению в БД полученного при преобразовании геометрического объекта.
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Выборка значений геометрических типов

Значения геометрических типов, предварительно сохраненные в таблице, могут быть выбраны в WKT/WKB-формате с помощью соответствующих функций (см. приложения 1, 2).

В предикате равенства в поисковых запросах с геометрическими типами данных необходимо задавать одновременно значение геометрического типа и идентификатор его координат (с помощью функции SRID), например,

```
create or replace table  geo_test (p point);
insert into geo_test(p) values (PointFromText('point (1 1)'));
insert into geo_test(p) values (PointFromText('point (1 2)', 1));
insert into geo_test(p) values (PointFromText('point (1 3)', 1));
insert into geo_test(p) values (PointFromText('point (1 2)', 2));
insert into geo_test(p) values (PointFromText('point (1 2)', 3));
select count(*)  from geo_test where p='point(1 2)';
select count(*)  from geo_test where p='point(1 2)'and srid(p)=2;

create or replace table geom (g geometry);
select x(g) from geom where GeometryType(g)='POINT';
```

Тип графических данных конкретного столбца хранится в элементе Prec поля $$$S24 системной таблицы $$$ATRRI.

Кроме того, информацию о столбцах с геометрическими типами данных можно получить из представления GEOMETRY_COLUMNS, создать которое можно с помощью скрипта `geo_cat.sql`, размещаемого в подкаталоге `/dict` установочного каталога СУБД ЛИНТЕР:

```
create view GEOMETRY_COLUMNS AS
select
cast ('' as varchar(256)) as f_table_catalog,
cast ($$$s34 as varchar(256)) as f_table_schema,
cast ($$$s13 as varchar(256)) as f_table_name,
cast ($$$s23 as varchar(256)) as f_geometry_column,
cast ( 2 as integer ) as coord_dimension,
cast (-1 as integer) as srid
from
LINTER_SYSTEM_USER.$$$sysrl,
LINTER_SYSTEM_USER.$$$attri,
LINTER_SYSTEM_USER.$$$usr
where $$$s11 = $$$s21 and
$$$s12 = $$$s31 and
$$$s32 = 0 and
GetByte($$$S24,1) = 9 and
GetByte($$$S24,2) > 0;
```

Выборка в WKT-формате

Функция

Преобразование внутреннего представления геометрического объекта в WKT-представление.

Спецификация

AsText | to_char (<имя столбца>[,<формат>])

<имя столбца> – имя столбца с геометрическим типом данных;
<формат> – символьный литерал формата преобразования строки.

Синтаксические правила

Аргумент <формат> задает формат преобразования строки (см. описание функции to_char в документе «СУБД ЛИНТЕР. Справочник по SQL»).

Возвращаемое значение

1) В случае нормального завершения:
   - значение типа char, соответствующее WKB-представлению геометрического объекта типа VARBYTE. Длина строки вычисляется динамически и не может превышать 4000 символов;
   - при выводе значений координат без указания формата будет выведено 10 знаков после запятой;
   - при выводе больших значений координат они будут преобразованы в экспоненциальный вид;
   - значение типа blob, соответствующее WKB-представлению геометрического объекта типа BLOB. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1) геометрический VARBYTE-объект

```
CREATE OR REPLACE TABLE POINT_TEST(P POINT);
INSERT INTO POINT_TEST VALUES( PointFromText('POINT (1 1)'));
INSERT INTO POINT_TEST VALUES( PointFromText('POINT (0 1)'));
INSERT INTO POINT_TEST VALUES( PointFromText('POINT (1 2)'));
INSERT INTO POINT_TEST VALUES( PointFromText('POINT (1 1)'));

select AsText(P) from POINT_TEST;
 INL : начальное время : 21:14:42 конечное время :  21:14:42

|POINT (1 1)                                |
|POINT (0 1)                                |
|POINT (1 2)                                |
|POINT (1 1)                                |
```


2) геометрический BLOB-объект

```
select getblobstr(astext(GEOM), 1, 50), lenblob(astext(GEOM)),
 lenblob(GEOM) from GTEST;
|POLYGON ((10 10,10 50,50 50,50 10,10 10))| 41| 97|
```

Выборка в WKB-формате

Функция

Преобразование внутреннего представления геометрического объекта в WKB-представление.

Спецификация

`AsBinary(<имя столбца>)`

`<имя столбца>` – имя столбца с геометрическим типом данных.

Возвращаемое значение

1) В случае нормального преобразования:
   - значение типа varbyte, соответствующее WKB-представлению геометрического объекта VARBYTE-типа. Длина значения зависит от геометрического типа объекта;
   - значение типа blob, соответствующее WKB-представлению геометрического объекта типа BLOB. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примеры

1) геометрический VARBYTE-объект

```
select AsBinary(P) from POINT_TEST;

| 01 01 00 00 00 00 00 00 00 00 00 F0 3F 00 00 00 00 00 00 F0 3F|
| 01 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F0 3F|
| 01 01 00 00 00 00 00 00 00 00 00 F0 3F 00 00 00 00 00 00 00 40|
| 01 01 00 00 00 00 00 00 00 00 00 F0 3F 00 00 00 00 00 00 F0 3F|
```

2) геометрический BLOB-объект

```
select cast getblobstr(asbinary(GEOM), 1, 93) as byte(93) from
 GTEST;
0103000000010000000500000000000000000024400000000000002440000000000000
0244000000000000049400000000000004940000000000000494000000000000049400
0000000000002440000000000000244000000000000024400
```

Средства обработки геометрических типов данных

СУБД ЛИНТЕР предоставляет набор функций для выполнения различных операций над геометрическими данными. Эти функции условно могут быть разделены на следующие группы:

- функции для анализа свойств геометрических данных;
- функции для создания новых значений геометрических типов из уже существующих;
- функции, которые описывают отношения между двумя значениями геометрических типов.

Примечание
Вычисление значений ряда функций (например Area, Length) поддерживается только для плоской системы координат.

Функции для анализа свойств геометрических типов

Получить название геометрического типа

Функция

Предоставление имени геометрического типа данных для геометрического объекта любого типа.

Спецификация

`GeometryType (<объект>)`

`<объект>` – любой геометрический объект.

Возвращаемое значение

1) Значение типа char(18), соответствующее названию геометрического типа переданного значения (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Пример

```
SELECT GeometryType(GeomFromText('POINT(1 1)'));
|POINT              |
```

Получить размерность геометрического типа

Функция

Предоставление размерности значения любого геометрического типа данных.

Спецификация

`Dimension (<объект>)`

<объект> – любой геометрический объект.

Возвращаемое значение

1) Целочисленное значение (integer), соответствующее размерности переданного значения геометрического типа (в случае нормального завершения):
   - -1 – объект пуст (empty);
   - 0 – нулевая размерность (точка);
   - 1 – линия;
   - 2 – поверхность.
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Пример

```
SELECT Dimension(GeomFromText('LineString(1 1,2 2)'));
|          1|
```

Получить идентификатор системы координат

Функция

Предоставление идентификатора системы координат значения любого геометрического типа данных.

Спецификация

SRID(<объект>)

<объект> – любой геометрический объект.

Возвращаемое значение

1) Целочисленное значение (integer), соответствующее идентификатору системы координат переданного значения геометрического типа (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Пример

```
SELECT SRID(GeomFromText('LineString(1 1,2 2)',101));
|        101|
```

Преобразовать объект к заданной системе координат

Функция

Преобразование объекта к новой системе координат.

Спецификация

Transform(<объект>,<srid>)

<объект> – любой геометрический объект.
<srid> – идентификатор системы координат.

Возвращаемое значение

1) В случае нормального завершения – геометрический объект в заданной системе координат.

   Тип возвращаемого значения:

   - char для геометрического объекта типа VARBYTE. Длина строки вычисляется динамически и не может превышать 4000 символов;
   - blob для геометрического объекта типа BLOB. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»);
   - NULL, если один или оба аргумента NULL.

2) Код завершения СУБД ЛИНТЕР (при ошибке преобразования).

Примечание
В текущей версии возвращается исходный (не преобразованный) объект.

Примеры

1) геометрический VARBYTE-объект

```
SELECT AsText(transform(GeomFromText('LineString(1 1,2 2)'),101));
|LINESTRING (1 1,2 2)|
```

2) геометрический BLOB-объект

```
SELECT getblobstr(astext(TRANSFORM(GEOM, 1)), 1, 80) from GTEST;
|POLYGON ((10 10,10 50,50 50,50 10,10 10))|
```

Получить минимальный ограничивающий прямоугольник

Функция

Предоставление минимального ограничивающего прямоугольника любого геометрического объекта.

Спецификация

```
Envelope(<объект>)
```

<объект> – любой геометрический объект.

Возвращаемое значение

1) Значение типа POLYGON, представляющее минимальный ограничивающий прямоугольник с координатами (MINX MINY, MAXX MINY, MAXX MAXY, MINX MAXY, MINX MINY) (в случае нормального завершения).
2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT AsText(Envelope(GeomFromText('LineString(1 1,2 2)',101)));
|POLYGON ((1 1,2 1,2 2,1 2,1 1))                                  |
```


Получить границу значения геометрического типа

Функция

Предоставление границы (возможно, комбинированной) значения геометрического типа данных.

Спецификация

`Boundary(<объект>)`

`<объект>` – любой геометрический объект.

Возвращаемое значение

1) В случае нормального завершения – геометрический объект, являющийся границей заданного геометрического объекта.

   Тип возвращаемого значения:
   - char для геометрического объекта типа VARBYTE. Длина строки вычисляется динамически и не может превышать 4000 символов;
   - blob для геометрического объекта типа BLOB. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»).

Примечание
Для геометрического типа GeometryCollection возвращается полная граница (без применения метода «mod 2»).

2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примеры

1) Геометрический VARBYTE-объект

```
SELECT AsText(Boundary(GeomFromText('LineString(1 1,2 2,3 3)')));

|MULTIPOINT (1 1,3 3)
```

2) Геометрический BLOB-объект

```
select getblobstr(astext(BOUNDARY(GEOM)), 1, 50),
 GLENGTH(BOUNDARY(GEOM)) from GTEST;
LINESTRING (10 10,10 50,50 50,50 10,10 10)| 160|
```

Проверить существование значения геометрического типа

Функция

Проверка существования значения геометрического типа данных.

Спецификация

`IsEmpty(<объект>)`

`<объект>` – любой геометрический объект.

Возвращаемое значение

1) Значение типа integer:
   - 1 – значение геометрического типа пусто;
   - 0 – значение геометрического типа определено;
   - -1 – в случае, если аргумент функции равен NULL.
2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примеры

1)

```
SELECT IsEmpty(GeomFromText('Point(1 1)'));
|          0|
```

2)

```
SELECT IsEmpty(GeomFromText('Point EMPTY'));
|          1|
```

Проверить сложность геометрического объекта

Функция

Проверка сложности геометрического объекта. Описание каждого инстанцируемого геометрического класса включает условия, при которых элемент, принадлежащий этому классу, классифицируется как простой или не простой (сложный). Как правило, простота или сложность объекта зависит от наличия точек пересечения или линий соприкосновения элементов объекта, например, рисунок 2.

Рисунок 2. Примеры геометрического объекта

Спецификация

`IsSimple(<объект>)`

`<объект>` – любой геометрический объект.

Возвращаемое значение

1) Значение типа integer:
   - 1 – геометрический объект является простым;
   - 0 – геометрический объект является сложным;

- -1 – в случае, если аргумент функции равен NULL.

2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примеры

1)

```
SELECT IsSimple(GeomFromText('LineString(0 0,0 1,1 0,1 1,0 0)'));

|          0|
```

2)

```
SELECT IsSimple(GeomFromText('LineString(0 0,0 1,1 1,1 0,0 0)'));

|          1|
```

Функции для создания новых значений геометрических типов

Функции для работы с точкой

Получить X-координату точки

Функция

Получение X-координаты точки.

Спецификация

`X (<точка>)`

`<точка>` – геометрический объект типа POINT.

Возвращаемое значение

1) Значение типа double, соответствующее X-координате точки.
2) NULL, если точка пуста.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT X(GeomFromText('Point(56.7 53.34)',101));

|                  56.7|
```

Получить Y-координату точки

Функция

Получение Y-координаты точки.

Спецификация

`Y (<точка>)`

<точка> – геометрический объект типа POINT.

Возвращаемое значение

1) Значение типа double, соответствующее Y-координате точки.
2) NULL, если точка пуста.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT Y(GeomFromText('Point(56.7 53.34)',101));

|                 53.34|
```

Функции для работы с ломаной линией

Получить координаты начала ломаной линии

Функция

Получение координат начальной точки ломаной линии.

Спецификация

StartPoint (<ломаная линия>)

<ломаная линия> – геометрический объект типа LINESTRING.

Возвращаемое значение

1) Значение типа POINT, соответствующее начальной координате ломаной линии.
2) NULL, если ломаная линия пуста.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT AsText(StartPoint(GeomFromText('LineString(1 1,2 2,3
 3)')));

|POINT (1 1)                                |
```

Получить координаты конца ломаной линии

Функция

Получение координат конечной точки ломаной линии.

Спецификация

EndPoint (<ломаная линия>)

<ломаная линия> – геометрический объект типа LINESTRING.

Возвращаемое значение

1) Значение типа POINT, соответствующее конечной координате ломаной линии.
2) NULL, если ломаная линия пуста.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT AsText(EndPoint(GeomFromText('LineString(1 1,2 2,3 3)')));

|POINT (3 3)                                |
```

Получить координаты узла ломаной линии

Функция

Получение координат заданного узла ломаной линии.

Спецификация

PointN (<ломаная линия>,[<узел>])

<ломаная линия> – геометрический объект типа LINESTRING;
<узел> – целочисленное положительное значение.

Синтаксические правила

1) Аргумент <узел> задает порядковый номер точки (узла) ломаной линии. Отсчет узлов начинается с 1.
2) Если <узел> не задан, по умолчанию принимается значение 1.

Возвращаемое значение

1) Значение типа POINT, соответствующее координатам заданного узла (точки) ломаной линии.
2) NULL, если ломаная линия пуста.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT AsText(PointN(GeomFromText('LineString(1 1,2 2,3 3)'),2));

|POINT (2 2)                                |
```

Получить длину ломаной линии

Функция

Получение длины ломаной линии.

Спецификация

Length | GLength (<ломаная линия>)

<ломаная линия> – геометрический объект типа LINESTRING.

Возвращаемое значение

1) Значение типа double, соответствующее длине ломаной линии.
2) NULL, если ломаная линия пуста.
3) Для функции Length – значение типа double, соответствующее длине в байтах внутреннего представления графического объекта (если в качестве аргумента передан геометрический объект, отличный от ломаной линии).

Примечание
Длину геометрического типа функция `Length` будет возвращать только в том случае, если транслятор сможет определить геометрический тип значения. В противном случае будет возвращена длина значения в байтах.

Примеры

1)

```
SELECT GLength(GeomFromText('LineString(1 1,2 2,3 3)'));

|      2.82842712474619|
```

2)

```
SELECT Length(GeomFromText('LineString(1 1,2 2,3 3)'));

|      2.82842712474619|
```

Получить количество узлов ломаной линии

Функция

Получение количества узлов ломаной линии.

Спецификация

`NumPoints(<ломаная линия>)`

`<ломаная линия>` – геометрический объект типа LINESTRING.

Возвращаемое значение

1) Значение типа integer, соответствующее количеству узлов ломаной линии.
2) NULL, если ломаная линия пуста.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT NumPoints(GeomFromText('LineString(1 1,2 2,3 3)'));
|          3|
```

Проверка выпуклости ломаной линии

Функция

Проверка выпуклости ломаной линии. Ломаная линия считается выпуклой (кольцом), если координаты начальной и конечной точки совпадают, и линия является простой (не проходит через некоторую точку дважды).

Спецификация

`IsRing(<ломаная линия>)`

`<ломаная линия>` – геометрический объект типа LINESTRING.

Возвращаемое значение

1) Значение типа integer, соответствующее типу ломаной линии:
   - 1 – ломаная линия выпуклая;
   - 0 – ломаная линия не выпуклая;
   - -1 – в случае, если аргумент функции равен NULL.
2) NULL, если ломаная линия пуста.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT IsRing(GeomFromText('LineString(1 1,2 2,3 2,1 1)'));

|          1|
```

Проверка замкнутости ломаной линии

Функция

Проверка замкнутости ломаной линии. Ломаная линия считается замкнутой, если координаты начальной и конечной точки совпадают, и линия является простой (не проходит через одну и ту же точку дважды, за исключением начальной и конечной точки).

Спецификация

`IsClosed(<ломаная линия>)`

`<ломаная линия>` – геометрический объект типа LINESTRING.

Возвращаемое значение

1) Значение типа integer, соответствующее типу ломаной линии:
   - 1 – ломаная линия замкнута;
   - 0 – ломаная линия разомкнута;
   - -1 – в случае, если аргумент функции равен NULL.
2) NULL, если ломаная линия пуста.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT IsClosed(GeomFromText('LineString(1 1,2 2,3 3)'));

|          0|
```

Функции для работы с группой ломаных линий

Получить длину группы ломаных линий

Функция

Получение длины группы ломаных линий.

Спецификация

Length|GLength(<группа ломаных линий>)

<группа ломаных линий> – геометрический объект типа MULTILINESTRING.

Возвращаемое значение

1) Значение типа double, соответствующее длине ломаной линии.
2) NULL, если ломаная линия пуста.
3) Для функции Length – значение типа double, соответствующее длине в байтах внутреннего представления графического объекта (если в качестве аргумента передан геометрический объект, отличный от группы ломаных линий).

Примеры

1)

```
SELECT Length(MLineFromText('MultiLineString((1 1,2 2,3 3),(4 4,5
 5))'));
|      4.24264068711929|
```

2)

```
SELECT Length(GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5
 5))'));
|        4.24264068711929|
```

Проверка замкнутости группы ломаных линий

Функция

Проверка замкнутости группы ломаных линий. Группа считается замкнутой, если замкнутыми являются одновременно все ломаные линии группы.

Спецификация

IsClosed(<группа ломаных линий>)

<группа ломаных линий> – геометрический объект типа MULTILINESTRING.

Возвращаемое значение

1) Значение типа double, соответствующее длине заданной группы ломаных линий:
   - 1 – группа ломаных линий замкнута;
   - 0 – группа ломаных линий разомкнута;
   - -1 – в случае, если аргумент функции равен NULL.
2) NULL, если ломаная линия пуста.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT IsClosed(GeomFromText('MultiLineString((1 1,2 2,3 3),(4 4,5
 5))'));
```



```
|          0|
```

Функции для работы с многоугольником

Определение площади многоугольника

Функция

Получение площади многоугольника, вычисленной в системе координат этого многоугольника.

Спецификация

```
Area (<многоугольник>)
```

<многоугольник> – геометрический объект типа POLYGON.

Возвращаемое значение

1) Значение типа double, соответствующее площади многоугольника.
2) NULL, если многоугольник пуст.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT Area(GeomFromText('Polygon((0 0,0 3,3 3,3 0,0 0),(1 1,1 2,2
 2,2 1,1 1))'));
|                    8|
```

Определение количества пересекающихся областей многоугольника

Функция

Получение количества пересекающихся областей многоугольника.

Спецификация

```
NumInteriorRing|NumInteriorRings (<многоугольник>)
```

<многоугольник> – геометрический объект типа POLYGON.

Возвращаемое значение

1) Значение типа integer, соответствующее количеству пересекающихся областей многоугольника.
2) NULL, если многоугольник пуст.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT NumInteriorRings(GeomFromText('Polygon((0 0,0 3,3 3,3 0,0
 0),(1 1,1 2,2 2,2 1,1 1))'));
|          1|
```

Определение внешней границы многоугольника

Функция

Получение внешней границы многоугольника.

Спецификация

`ExteriorRing` (<многоугольник>)

<многоугольник> – геометрический объект типа POLYGON.

Возвращаемое значение

1) В случае нормального завершения – геометрический объект типа LINESTRING, являющийся внешней границей многоугольника.

   Тип возвращаемого значения:
   - char для геометрического объекта типа VARBYTE. Длина строки вычисляется динамически и не может превышать 4000 символов;
   - blob для геометрического объекта типа BLOB. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»).
   - NULL, если <многоугольник> пуст.
2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

Геометрический VARBYTE-объект

```
SELECT AsText(ExteriorRing(GeomFromText('Polygon((0 0,0 3,3 3,3
 0,0 0),(1 1,1 2,2 2,2 1,1 1))')));
|LINESTRING (0 0,0 3,3 3,3 0,0 0)                              |
```

Получение заданной пересекающейся области многоугольника

Функция

Получение внутренней границы многоугольника.

Спецификация

`InteriorRingN` (<многоугольник>[,<номер>])

<многоугольник> – геометрический объект типа POLYGON;
<номер> – порядковый номер требуемой области.

Синтаксические правила

1) Аргумент <номер> задает порядковый номер самопересекающейся внутренней области многоугольника. Отсчет областей узлов начинается с 1.
2) Если <номер> не задан, по умолчанию принимается значение 1.

Возвращаемое значение

1) В случае нормального завершения – геометрический объект типа LINESTRING, соответствующий границе заданной области.

Тип возвращаемого значения:

- char для геометрического объекта типа VARBYTE. Длина строки вычисляется динамически и не может превышать 4000 символов;
- blob для геометрического объекта типа BLOB. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»).
- NULL, если <многоугольник> пуст.

2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции, в частности, 1036 – требуемая внутренняя граница не существует).

Пример

Геометрический VARBYTE-объект

```
SELECT AsText(InteriorRingN(GeomFromText('Polygon((0 0,0 3,3 3,3
 0,0 0),(1 1,1 2,2 2,2 1,1 1))'),1));
|LINESTRING (1 1,1 2,2 2,2 1,1 1)                                |
```

Определение геометрического центра многоугольника

Функция

Получение геометрического центра многоугольника (может находиться за пределами многоугольника). Геометрический центр вычисляется на основе «центра масс» многоугольника при условии, что вся его «масса» равномерно распределена между вершинами внешней границы.

Спецификация

Centroid(<многоугольник>)

<многоугольник> – геометрический объект типа POLYGON.

Возвращаемое значение

1) Значение типа POINT, соответствующее координатам геометрического центра многоугольника.
2) NULL, если многоугольник пуст.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT AsText(Centroid(GeomFromText('Polygon((0 0,0 3,3 3,3 0,0
 0),(1 1,1 2,2 2,2 1,1 1))')));
|POINT (1.5 1.5)                          |
```

Получение первой точки внешней границы многоугольника

Функция

Получение первой точки внешней границы многоугольника.

Спецификация

PointOnSurface (<многоугольник>)

<многоугольник> – геометрический объект типа POLYGON.

Возвращаемое значение

1) Значение типа POINT, соответствующее координатам первой внешней границы многоугольника.
2) NULL, если <многоугольник> пуст.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT AsText(PointOnSurface(GeomFromText('Polygon((0 0,0 3,3 3,3
 0,0 0),(1 1,1 2,2 2,2 1,1 1))')));
|POINT (0.75 0.75)                              |
```

Функции для работы с группой многоугольников

Определение площади группы многоугольников

Функция

Получение площади группы многоугольников, вычисленной в системе координат этой группы.

Спецификация

Area (<группа многоугольников>)

<группа многоугольников> – геометрический объект типа MULTIPOLYGON.

Возвращаемое значение

1) Значение типа double, соответствующее площади группы многоугольников.
2) NULL, если многоугольник пуст.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT Area(GeomFromText('MultiPolygon(((0 0,0 3,3 3,3 0,0 0),(1
 1,1 2,2 2,2 1,1 1)))'));
|                    8|
```

Определение геометрического центра группы многоугольников

Функция

Получение геометрического центра группы многоугольников (может находиться за пределами группы). Геометрический центр вычисляется на основе «центра масс» входящих в группу многоугольников.

Спецификация

Centroid(<группа многоугольников>)

<группа многоугольников> – геометрический объект типа MULTIPOLYGON.

Возвращаемое значение

1) Значение типа POINT, соответствующее координатам геометрического центра группы многоугольников.
2) NULL, если многоугольник пуст.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT AsText(Centroid(GeomFromText('MultiPolygon(((0 0,0 3,3 3,3
 0,0 0)),((11 11,11 12,12 12,12 11,11 11)))')));
|POINT (6.5 6.5)                          |
```

Получение первой точки внешней границы группы многоугольников

Функция

Получение первой точки внешней границы группы многоугольников.

Спецификация

PointOnSurface(<группа многоугольников>)

<группа многоугольников> – геометрический объект типа MULTIPOLYGON.

Возвращаемое значение

1) Значение типа POINT, соответствующее координатам первой внешней границы группы многоугольников.
2) NULL, если многоугольник пуст.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT AsText(PointOnSurface(GeomFromText('MultiPolygon(((10 10,10
 13,13 13,13 10,10 10)))')));
|POINT (11.5 11.5)                        |
```

Функции для работы с группой геометрических объектов

Определение количества объектов в группе

Функция

Получение количества объектов, входящих в состав группы геометрических объектов.

Спецификация

NumGeometries(<группа объектов>)

<группа объектов> – геометрический объект типа GEOMETRYCOLLECTION, MULTIPOINT, MULTILINESTRING или MULTIPOLYGON.

Возвращаемое значение

1) Количество объектов в группе (значение типа integer).
2) NULL, если многоугольник пуст.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT NumGeometries(GeomFromText('GeometryCollection(Point(1
 1),LineString(2 2, 3 3))'));
|          2|
```

Получить заданный объект группы

Функция

Получение заданного объекта из группы геометрических объектов.

Спецификация

`GeometryN (<группа объектов>[,<номер>])`

`<группа объектов>` – геометрический объект типа GEOMETRYCOLLECTION, MULTIPOINT, MULTILINESTRING или MULTIPOLYGON;
`<номер>` – порядковый номер требуемого объекта.

Синтаксические правила

1) Аргумент `<номер>` задает порядковый номер объекта в группе геометрических объектов. Отсчет начинается с 1.
2) Если `<номер>` не задан, по умолчанию принимается значение 1.

Возвращаемое значение

1) В случае нормального завершения – геометрический объект с заданным порядковым номером из состава группы объектов.

   Тип возвращаемого объекта:
   - char для геометрического объекта <группа объектов> типа VARBYTE. Длина строки вычисляется динамически и не может превышать 4000 символов;
   - blob для геометрического объекта <группа объектов> типа BLOB. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»);
   - NULL, если <группа объектов> пуста.
2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примеры

1) геометрический VARBYTE-объект

```
SELECT AsText(GeometryN(GeomFromText('GeometryCollection(Point(1
 1),LineString(2 2, 3 3))'),1));

|POINT (1 1)                                |
```


2) геометрический BLOB-объект

```
select getblobstr(astext(GEOMETRYN(SYMDIFFERENCE(GEOM,
GeomFromText('POLYGON (5 5,5 25,25 25,25 5,5 5)', 4284)), 2)), 1,
 80)
from GTEST;

|POLYGON ((25 10,25 5,5 5,5 25,10 25,10 10,25 10)) |
```

Функции отношения геометрических типов

Проверка пересечения геометрических объектов

Функция

Проверка пересечения двух геометрических MBR-объектов.

Спецификация

`Filter(<объект1>,<объект2>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) 1 – объекты пересекаются.
2) 0 – объекты не пересекаются.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
select filter(GeomFromText('POLYGON ((0 0,0 5,5 5,5 0,0 0),(2 2,2
 4,3 4,3 2,2 2))'),
GeomFromText('POLYGON ((1 1,1 6,6 6,6 1,1 1),(2 3,2 4,4 4,4 3,2
 3))'));
|          1|
```

Пересечение геометрических объектов

Функция

Получение пересечения двух геометрических объектов.

Спецификация

`Intersection(<объект1>,<объект2>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) В случае нормального завершения – геометрический объект, представляющий пересечение заданных геометрических объектов.
2) Для включения в результирующий объект составных типов (MULTIPOINT, MULTILINESTRING, MULTIPOLYGON) они разбиваются на объекты простых типов.

Примечание
Пересечение значений типа Circle и значения, в состав которого входит тип Polygon (Polygon, Multipolygon, Geometrycollecion с Polygon), осуществляется особым образом, что связанно с отсутствием типа «кривая». Круг аппроксимируется вписанным правильным многоугольником с 32 вершинами, и полученный многоугольник пересекается со вторым значением геометрического типа.

3) GEOMETRYCOLLECTION (EMPTY), если объекты не пересекаются.

Тип возвращаемого значения:

- char, если оба геометрических объекта являются объектами VARBYTE-типа. Длина строки вычисляется динамически и не может превышать 4000 символов;
- blob, если один или оба геометрических объектов являются объектами BLOB-типа. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»);
- NULL, если один или оба аргумента NULL;

4) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примеры

1) геометрический VARBYTE-объект

```
select AsText(Intersection(GeomFromText('POLYGON ((0 0,0 5,5 5,5
 0,0 0),(2 2,2 4,3 4,3 2,2 2))'),
GeomFromText('POLYGON ((1 1,1 6,6 6,6 1,1 1),(2 3,2 4,4 4,4 3,2
 3))')));
Результат:
|POLYGON ((5 1,1 1,1 5,5 5,5 1),(2 3,2 4,3 4,4 4,4 3,3 3,3 2,2 2,2
 3))                                          |
```

2) геометрический BLOB-объект

```
select getblobstr(astext(INTERSECTION(GEOM, GeomFromText('POLYGON
 (5 5,5 25,25 25,25 5,5 5)', 4284))), 1, 200) from GTEST;
|POLYGON ((25 10,10 10,10 25,25 25,25 10)) |
```

Объединение геометрических объектов

Функция

Получение объединения двух геометрических объектов.

Спецификация

`Union(<объект1>,<объект2>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) В случае нормального завершения – геометрический объект, представляющий объединение двух заданных геометрических объектов.

Тип возвращаемого значения:

- char, если оба геометрических объекта являются объектами VARBYTE-типа. Длина строки вычисляется динамически и не может превышать 4000 символов;
- blob, если один или оба геометрических объектов являются объектами BLOB-типа. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»);
- NULL, если один или оба аргумента NULL.

2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примечание
Так же, как и для функции INTERSECTION, при объединении с некоторыми геометрическими типами круг может быть аппроксимирован многоугольником с 32 вершинами.

Примеры

1) геометрический VARBYTE-объект

```
select AsText(Union(GeomFromText('POLYGON ((0 0,0 3,3 3,3 0,0
 0))'),GeomFromText('POLYGON ((1 1,1 4,4 4,4 1,1 1))')));
Результат:
|POLYGON ((3 1,3 0,0 0,0 3,1 3,1 4,4 4,4 1,3 1)) |
```

2) геометрический BLOB-объект

```
select getblobstr(astext(UNION(GEOM, GeomFromText('POLYGON (5 5,5
 25,25 25,25 5,5 5)', 4284))), 1, 100) from GTEST;
|POLYGON ((10 25,10 50,50 50,50 10,25 10,25 5,5 5,5 25,10 25)) |
```

Разность геометрических объектов

Функция

Получение геометрической разности двух объектов.

Спецификация

`Difference` (`<объект1>`,`<объект2>`)

`<объект1>`,`<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) В случае нормального завершения – геометрический объект, представляющий разность двух заданных геометрических объектов.

Тип возвращаемого значения:

- char, если оба геометрических объекта являются объектами VARBYTE-типа. Длина строки вычисляется динамически и не может превышать 4000 символов;
- blob, если один или оба геометрических объектов являются объектами BLOB-типа. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»);

- NULL, если один или оба аргумента NULL.

2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примеры

1) геометрический VARBYTE-объект

Получить разность двух отрезков линии:

```
Объект 1    (0,0)_____________________________(5,0)
Объект 2              (3,0)_____________(4,0)
Разность    (0,0)_________(3,0)     (4,0)_____(5,0)

CREATE OR REPLACE TABLE TEST (Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES(LineFromText('LINESTRING (0 0,0 5)'),
LineFromText('LINESTRING (0 3,0 4)'));
select astext(DIFFERENCE(LINE1, LINE2)) FROM TEST;
Результат:
|MULTILINESTRING ((0 0,0 3),(0 4,0 5))   |
```

2)

```
Объект 1    (0,0)_____________________________(5,0)
Объект 2    (0,0)_________________(3,0)
Разность                  (3,0)_____________(5,0)

CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES(LineFromText('LINESTRING (0 0,0 5)'),
LineFromText('LINESTRING (0 0,0 3)'));
select astext(DIFFERENCE(LINE1, LINE2)) FROM TEST;
Результат:
|LINESTRING (0 3,0 5)                |
```

3) геометрический BLOB-объект

```
select
getblobstr(astext(DIFFERENCE(GEOM, GeomFromText('POLYGON (5 5,5
25,25 25,25 5,5 5)', 4284))), 1, 110)from GTEST;
|POLYGON ((10 25,10 50,50 50,50 10,25 10,25 25,10 25)) |
```

Симметричная разность геометрических объектов

Функция

Получение симметричной геометрической разности двух объектов. Симметричная разность вычисляется путем исключения общей (пересекающейся) части геометрических объектов.

Спецификация

`SymDifference (<объект1>,<объект2>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) В случае нормального завершения – геометрический объект, представляющий симметричную разность двух заданных геометрических объектов.

   Тип возвращаемого значения:

   - char, если оба геометрических объекта являются объектами VARBYTE-типа. Длина строки вычисляется динамически и не может превышать 4000 символов;
   - blob, если один или оба геометрических объектов являются объектами BLOB-типа. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»);
   - NULL, если один или оба аргумента NULL.

2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примеры

1) геометрический VARBYTE-объект

Получить симметричную разность двух отрезков линии:

```
Объект 1                (0,0)_____________________(5,0)
Объект 2                      (3,0)_____________________(7,0)
Симметричная разность (0,0)_______(3,0)   (5,0)_________(7,0)

CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES(LineFromText('LINESTRING (0 0,5 0)'),
LineFromText('LINESTRING (3 0,7 0)'));
select astext(SYMDIFFERENCE(LINE1, LINE2)) FROM TEST;
Результат:
|MULTILINESTRING((0 0,3 0),(5 0,7 0))|
```

2)

```
CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES(LineFromText('LINESTRING (0 0, 5 0)'),
LineFromText('LINESTRING (6 0,7 0)'));

INSERT INTO TEST VALUES(LineFromText('LINESTRING EMPTY'),
LineFromText('LINESTRING (6 0,7 0)'));

select decode(astext(SYMDIFFERENCE(LINE1, LINE2)), null,'null',
astext(SYMDIFFERENCE(LINE1, LINE2))) FROM TEST;
Результат:
|MULTILINESTRING((0 0,5 0),(6 0,7 0))|
|null                                |
```

3) геометрический BLOB-объект

```
select
  getblobstr(astext(SYMDIFFERENCE(GEOM, GeomFromText('POLYGON (5
 5,5 25,25 25,25 5,5 5)', 4284))), 1, 110) from GTEST;
```

```
Результат:
|MULTIPOLYGON (((10 25,10 50,50 50,50 10,25 10,25 25,10 25)),((25
 10,25 5,5 5,5 25,10 25,10 10,25 10)))|
```

Расстояние между объектами

Функция

Вычисление расстояния между заданными объектами.

Спецификация

`Distance (<объект1>,<объект2>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) Кратчайшее расстояние между заданными объектами (значение типа double).
2) NULL, если один из аргументов равен NULL.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT Distance(GeomFromText('POLYGON ((0 0,0 3,3 3,3 0,0 0),(1
 1,1 2,2 2,2 1,1 1))'),GeomFromText('MULTIPOINT (1.3 1.3, 1.5 1.5,
 1.8 1.8)'));
Результат:
|                  0.2|
```

Проверка расстояния между объектами

Функция

Проверка расстояния между заданными объектами.

Спецификация

`Dwithin (<объект1>,<объект2>,<расстояние>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.
`<расстояние>` – проверяемое расстояние (тип данных double). Единица измерения расстояния соответствует единицам измерения <объекта1> и <объекта2>.

Возвращаемое значение

1) 1 – если расстояние между объектами не больше указанного.
2) 0 – расстояние между объектами больше указанного.
3) NULL, если один из аргументов равен NULL.
4) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
SELECT Dwithin(GeomFromText('POLYGON ((0 0,0 3,3 3,3 0,0 0),(1 1,1
 2,2 2,2 1,1 1))'),
GeomFromText('MULTIPOINT (1.3 1.3, 1.5 1.5, 1.8 1.8)'), 0.7);
|          1|
```


Буферный геометрический объект

Функция

Получение буферного геометрического объекта (т. е. объекта, все точки которого равноудалены от заданного объекта).

Спецификация

`Buffer (<объект>,<расстояние>)`

`<объект>` – геометрические объекты произвольного типа (кроме типа LINE – бесконечной прямой);
`<расстояние>` – положительное вещественное значение.

Возвращаемое значение

1) В случае нормального завершения – геометрический объект, все точки которого расположены от `<объекта>` на заданное `<расстояние>`.

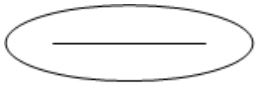
   Тип возвращаемого значения:

   - char для геометрического объекта типа VARBYTE. Длина строки вычисляется динамически и не может превышать 4000 символов;
   - blob для геометрического объекта типа BLOB. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»);
   - NULL, если один или оба аргумента NULL.

2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

Получить буферный геометрический объект от отрезка линии VARBYTE-типа, например:



```
create or replace table test( line1 linestring);
insert into test(line1) values( linefromtext('linestring (1 0,5
 0)'));
select astext(buffer(line1, 1)) from test;
Результат:
|POLYGON ((5 1,
1 1,
0.80491 0.980785,
0.617317 0.92388,
0.44443 0.83147,
0.292893 0.707107,
0.16853 0.55557,
0.07612 0.382683,
0.019215 0.19509,
```

```
0 0,
0.019215 -0.19509,
0.07612 -0.382683,
0.16853 -0.55557,
0.292893 -0.707107,
0.44443 -0.83147,
0.617317 -0.92388,
0.80491 -0.980785,
1 -1,
5 -1,
5.19509 -0.980785,
5.382683 -0.92388,
5.55557 -0.83147,
5.707107 -0.707107,
5.83147 -0.55557,
5.92388 -0.382683,
5.980785 -0.19509,
6 0,
5.980785 0.19509,
5.92388 0.382683,
5.83147 0.55557,
5.707107 0.707107,
5.55557 0.83147,
5.382683 0.92388,
5.19509 0.980785,
5 1))|
```

Выпуклая оболочка объекта

Функция

Получение выпуклой оболочки заданного объекта.

Спецификация

`ConvexHull` (`<объект>`)

`<объект>` – геометрические объекты произвольного типа;

Возвращаемое значение

1) В случае нормального завершения – геометрический объект, являющийся выпуклой оболочкой <объекта>.

   Тип возвращаемого значения:

   - char для геометрического объекта типа VARBYTE. Длина строки вычисляется динамически и не может превышать 4000 символов;
   - blob для геометрического объекта типа BLOB. В этом случае для получения значения типа char необходимо использовать SQL-функцию getblobstr (см. документ «СУБД ЛИНТЕР. Справочник по SQL»);

- NULL, если аргумент равен NULL.

2) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

Геометрический BLOB-объект

```
SELECT getblobstr(astext(ConvexHull(GEOM)), 1, 80) from GTEST;
|POLYGON ((10 10,50 10,50 50,10 50,10 10)) |
```

Проверка совпадения объектов

Функция

Проверка совпадения двух заданных объектов.

Две ломаных линии LineString считаются совпадающими, даже если обход точек у них разный (например, LINESTRING(0 0, 1 1) и LINESTRING(1 1, 0 0) совпадают).

Два объекта типа Polygon считаются совпадающими, если у них одинаковое количество границ и совпадают сами границы (обход вершин в границах и порядок следования внутренних границ может быть произвольным).

Для сложных геометрических типов (MultiPoint, MultiPolygon, MultiLineString, GeometryCollection) при сравнении также учитывается, что порядок следования составляющих их частей может отличаться.

Спецификация

`Equals (<объект1>,<объект2>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) 1, если объекты совпадают.
2) 0, если объекты не совпадают.
3) NULL, если один из аргументов равен NULL.
4) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
select Equals(GeomFromText('POLYGON ((-1 -1,-1 4,4 4,5 2,4 -1,-1
 -1),(0 0,0 1,1 1,1 0,0 0),(2 2,2 3,3 3,3 2,2 2))'),
GeomFromText('POLYGON ((4 4,-1 4,-1 -1,4 -1,5 2,4 4),(2 2,3 2,3
 3,2 3,2 2),(0 1,1 1,1 0,0 0,0 1))'));
Результат:
|          1|
```

Проверка пересечения объектов

Функция

Проверка пересечения двух заданных объектов (см. также функции Overlaps и Crosses).

Спецификация

`Intersects (<объект1>,<объект2>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) 1, если объекты пересекаются (имеют хотя бы одну общую точку).
2) 0, если объекты не пересекаются.
3) NULL, если один из аргументов равен NULL.
4) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примеры

1)

```
Объект 1 (0,0)_____________________ (5,0)
Объект 2               (3,0)_________________(7,0)

Пересекаются? Да

CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES( LineFromText('LINESTRING (0 0, 5 0)'),
LineFromText('LINESTRING (3 0,7 0)'));
select astext(intersects(LINE1, LINE2)) FROM TEST;
Результат:
|1 |
```

2)

```
select Intersects (GeomFromText('MULTIPOLYGON (((0 1,1 2,2 1,1 0,0
 1)),((2 1,3 2,4 1,3 0,2 1)))'),
GeomFromText('POLYGON (-1 1,-1 3,1 3,1 1,-1 1)'));
Результат:
|          1|
```

Проверка перекрытия объектов

Функция

Проверка перекрытия двух объектов.

Спецификация

`Overlaps (<объект1>,<объект2>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) 1, если для `<объекта1>` и `<объекта2>` выполняется одно из условий:
   - перекрывают друг друга;

- область перекрытия имеет ту же размерность, что и каждый из указанных объектов;
- область перекрытия не совпадает с <объектом1> или <объектом2>.

2) 0, если <объект1> и <объект2> не перекрываются или имеют разную размерность.

3) NULL, если один из аргументов равен NULL.

4) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примеры

1)

```
Объект 1 (0,0)_____________________(5,0)
Объект 2             (3,0)_____________________(7,0)
Перекрываются?  Да

CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES( LineFromText('LINESTRING (0 0, 5 0)'),
LineFromText('LINESTRING (3 0,7 0)'));
select astext(overlaps(LINE1, LINE2)) FROM TEST;

Результат:
|        1|
```

2)

```
Объект 1 (0,0)________(3,0)
Объект 2         (3,0)_____________________(7,0)
Перекрываются? Нет

Пересекаются?  Да

CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES( LineFromText('LINESTRING (0 0, 3 0)'),
 LineFromText('LINESTRING (3 0,7 0)'));

select  astext(overlaps(LINE1, LINE2)), astext(intersects(LINE1,
 LINE2)) FROM TEST;

Результат:
|      0|     1|
```

Проверка разъединения объектов

Функция

Проверка разъединения двух заданных объектов.

Спецификация

Disjoint(<объект1>,<объект2>)

<объект1>,<объект2> – геометрические объекты произвольного типа.

Возвращаемое значение

1) 1, если объекты разъединены (не имеют ни одной общей точки).
2) 0, если объекты пересекаются (имеют хотя бы одну общую точку).
3) NULL, если один из аргументов равен NULL.
4) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Примеры

1)

```
Объект 1   (0,0)_____________(5,0)
Объект 2                        (6,0)___(7,0)
Разъединены? Да

CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES( LineFromText('LINESTRING (0 0, 5 0)'),
LineFromText('LINESTRING (6 0,7 0)'));
select  astext(disjoint(LINE1, LINE2)) FROM TEST;
Результат:
|         1|
```

2)

```
select astext(disjoint(GeomFromText('MULTIPOLYGON (((0 1,1 2,2 1,1
 0,0 1)),
((2 1,3 2,4 1,3 0,2 1)))'),GeomFromText('POLYGON (-1 1,-1 3,1 3,1
 1,-1 1)'));
Результат:
|          0|
```

Проверка вложенности объектов (вариант 1)

Функция

Проверка вложенности первого объекта во второй.

Спецификация

Within(<объект1>,<объект2>)

<объект1>,<объект2> – геометрические объекты произвольного типа.

Возвращаемое значение

1) 1, если <объект2> целиком содержит <объект1>, причем пересечение их внутренних областей не пусто.
2) 0, если <объект2> не является вложенным в <объект1>.

3) NULL, если один из аргументов равен NULL.
4) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
Объект 1   (0,0)_____________(3,0)
Объект 2   (0,0)_____________________________(5,0)
Вложен?  Да

CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES( LineFromText('LINESTRING (0 0, 3 0)'),
 LineFromText('LINESTRING (0 0,5 0)'));

select astext(within(LINE1, LINE2)) FROM TEST;
Результат:
|        1|
```

Проверка вложенности объектов (вариант 2)

Функция

Проверка вложенности второго объекта в первый.

Спецификация

`Contains (<объект1>,<объект2>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) 1, если `<объект1>` целиком содержит `<объект2>`, причем пересечение их внутренних областей не пусто.
2) 0, если `<объект1>` не является вложенным в `<объект2>`.
3) NULL, если один из аргументов равен NULL.
4) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
Объект 1   (0,0)_____________(3,0)
Объект 2   (0,0)_____________________(5,0)

Вложен? Нет

CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES(LineFromText('LINESTRING (0 0, 3 0)'),
 LineFromText('LINESTRING (0 0,5 0)'));

select astext(contains(LINE1, LINE2)) FROM TEST;
```

```
Результат:
|         0|
```

Проверка касания объектов

Функция

Проверка касания двух объектов.

Спецификация

`Touches (<объект1>,<объект2>)`

`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) 1, если `<объект1>` и `<объект2>` касаются друг друга (пересекаются объекты, но не пересекаются их внутренние области).
2) 0, если `<объект1>` и `<объект2>` не пересекаются либо касаются своими внутренними областями.
3) NULL, если один из аргументов равен NULL или EMPTY.
4) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

```
Объект 1   (0,0)_________(3,0)
Объект 2             (3,0)________(5,0)
Касаются?   Да

CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES(LineFromText('LINESTRING (0 0, 3 0)'),
 LineFromText('LINESTRING (3 0,5 0)'));
select astext(touches(LINE1, LINE2)) FROM TEST;

Результат:

|         1|
```

Проверка скрещивания объектов

Функция

Проверка скрещивания двух объектов.

Спецификация

`Crosses (<объект1>,<объект2>)`

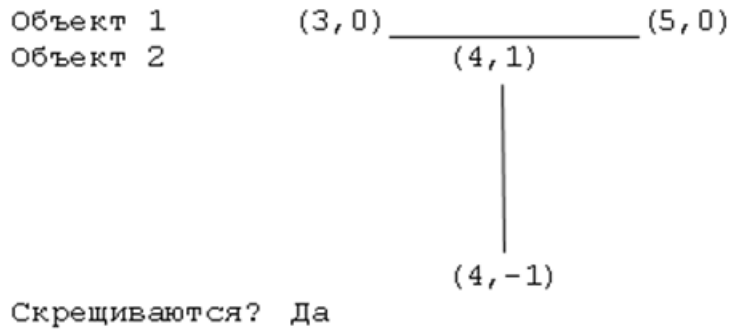
`<объект1>,<объект2>` – геометрические объекты произвольного типа.

Возвращаемое значение

1) 1, если для <объекта1> и <объекта2> выполняется одно из условий:
   - пересекаются их внутренние области;
   - размерность пересечения меньше максимальной размерности указанных объектов;
   - пересечение не совпадает с <объектом1> или <объектом2>.
2) 0, если <объект1> и <объект2> не пересекаются.
3) NULL, если один из аргументов равен NULL или EMPTY.
4) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

Проверить скрещивание двух объектов, например:




```
CREATE OR REPLACE TABLE TEST(Line1 LINESTRING, Line2 LINESTRING);
INSERT INTO TEST VALUES(LineFromText('LINESTRING (3 0, 5 0)'),
 LineFromText('LINESTRING (4 1,4 -1)'));
select astext(crosses(LINE1, LINE2)) FROM TEST;

Результат:

|         1|
```

Оптимизация работы с геометрическими данными

Для оптимизации поисковых операций с участием геометрических данных необходимо использовать индексы. Индексирование геометрических данных в СУБД ЛИНТЕР выполняется с помощью B-дерева.

Значения, хранящиеся в индексе, представляют собой значения геометрического типа MBR (Minimum Bounding Rectangle, минимальный ограничительный прямоугольник). В БД MBR-значение хранится в виде двух точек ((MINX,MINY),(MAXX,MAXY)). Все координаты имеют тип double.

Нижняя и верхняя граница оператора BETWEEN для геометрических данных задают левый нижний и правый верхний угол прямоугольника, при полном или частичном попадании в который оператор BETWEEN возвращает значение True (Истина).

Другие операторы сравнения (>,<.=,<>,>=,<=) также используют MBR.

Для сравнения двух объектов геометрических типов используется их MBR. Сначала сравнивается левый нижний угол (x, потом y), потом правый верхний угол (x, потом y). Если у какого-то MBR соответствующая координата при сравнении больше, то этот объект считается больше, если равны – сравнение продолжается. Если все соответствующие координаты равны, считается, что объекты равны между собой.

Команды создания индекса для столбцов геометрических типов данных имеют тот же синтаксис, что и команды создания простого индекса (см. документ «СУБД ЛИНТЕР. Справочник по SQL»). Поддерживается создание составного индекса для геометрических типов данных.

Примечание
При расчете длины индекса необходимо учесть, что на каждый столбец геометрического типа данных дополнительно добавляется 16 байт (4 значения DOUBLE – ограничивающий прямоугольник).

Пример создания простого индекса:

```
CREATE OR REPLACE TABLE POINT_TEST(P POINT);
CREATE INDEX P ON POINT_TEST;
```

Пример создания составного индекса:

```
CREATE OR REPLACE TABLE LSPOINT_TEST(P POINT, LS LINESTRING);
CREATE INDEX "TEST" ON LSPOINT_TEST(P,LS);
```

Примечание
Есть возможность реализовать конструкции PRIMARY KEY, FORAIGN KEY (включая ON UPDATE CASCADE, ON DELETE CASCADE), UNIQUE с участием геометрического индекса, но практическое их использование вызывает сомнение с точки зрения логики.

Пример

1) Создаем тестовую таблицу из 65543 записей:

```
CREATE OR REPLACE TABLE POINT_TEST(P POINT);
```

```
INSERT INTO POINT_TEST VALUES('POINT (100,100)');
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;
INSERT INTO POINT_TEST SELECT * FROM POINT_TEST;

INSERT INTO POINT_TEST VALUES('POINT (1 1)');
INSERT INTO POINT_TEST VALUES('POINT (1 1)');
INSERT INTO POINT_TEST VALUES('POINT (0 1)');
INSERT INTO POINT_TEST VALUES('POINT (1 2)');
INSERT INTO POINT_TEST VALUES('POINT (0 0)');
INSERT INTO POINT_TEST VALUES('POINT (0 0)');
INSERT INTO POINT_TEST VALUES('POINT (2 0)');
```

2) Выполняем поисковый запрос без использования индекса:

```
SELECT AsText(P) FROM POINT_TEST WHERE P BETWEEN
 PointFromText('POINT (1 1)')
AND PointFromText('POINT (1 2)');
 INL : начальное время : 21:41:37.00 конечное время :  21:41:37.31

|POINT (1 1)                              |
|POINT (1 1)                              |
|POINT (1 2)                              |
 INL : выдано строк     : 3
```

3) Создаем индекс и повторно выполняем поисковый запрос:

```
CREATE INDEX "TEST" ON POINT_TEST(P);
 INL : начальное время : 21:44:45 конечное время :  21:44:47
SELECT AsText(P) FROM POINT_TEST WHERE P BETWEEN
 PointFromText('POINT (1 1)')
AND PointFromText('POINT (1 2)');
 INL : начальное время : 21:45:05 конечное время :  21:45:05

|POINT (1 1)                              |
```

```
|POINT (1 1)                                |
|POINT (1 2)                                |
 INL : выдано строк     : 3
```

4) Анализируем результаты: без использования индекса время выполнения запроса повышается с 0.00 секунд до 0.31 секунды.

Управление вводом геометрических данных

Проверка корректности вводимых данных

Проверка корректности вводимых значений некоторых геометрических типов (Polygon, MultiPolygon) с помощью пользовательской функции требует значительного времени (проверка простоты и замкнутости внешних и внутренних границ, непересекаемость этих границ). Для сокращения этого времени и не допущения записи в БД некорректных данных можно воспользоваться командами СУБД ЛИНТЕР.

Проверка корректности геометрических данных для всей БД

Функция

Управление проверкой корректности геометрических данных для всей БД.

Спецификация

<проверкой корректности геометрических данных для всей БД>::=
```
SET CONNECTION GEODATA VALIDITY CHECKING {ON | OFF};
```

Общие правила

1) Опция ON устанавливает, OFF отменяет режим проверки корректности вводимых геометрических данных для всей БД.
2) По умолчанию действует опция ON.
3) Команда не выполняет проверку корректности геометрических BLOB-объектов.
4) Команда действует только на вновь открываемые соединения в текущей сессии и на все соединения при последующих запусках СУБД ЛИНТЕР. Команда может выполняться по любому соединению с СУБД.

Проверка корректности геометрических данных для текущей сессии

Функция

Управление проверкой корректности геометрических данных для текущей сессии.

Спецификация

<проверка корректности геометрических данных для текущей сессии>::=
```
SET DATABASE GEODATA VALIDITY CHECKING {ON | OFF};
```

Общие правила

1) Опция ON устанавливает, OFF отменяет режим проверки корректности вводимых геометрических данных в текущей сессии СУБД ЛИНТЕР.
2) По умолчанию действует опция ON.
3) Команда не выполняет проверку корректности геометрических BLOB-объектов.

4) Команда должна выполняться отдельно для каждого соединения с СУБД ЛИНТЕР, в котором необходимо выполнять проверку корректности вводимых геометрических объектов.

Управление хранением геометрических BLOB-данных

Функция

Управление хранением геометрических данных в пределах текущей сессии.

Спецификация

<управление созданием геометрических BLOB-данных>::=
```
SET SESSION BLOB GEOMETRY STORAGE {ON | OFF};
```

Общие правила

1) По умолчанию создаваемые геометрические данные имеют тип VARBYTE(1028)/ GEOMETRY, что во многих случаях не позволяет создавать и хранить в БД сложные геометрические объекты размерностью более 1028 байт. Данная команда действует на все каналы сессии и заставляет после её подачи с опцией ON автоматически создавать геометрические данные в виде BLOB-значений с длиной по умолчанию 4000 байт, т. е., например, после выполнения запроса

```
CREATE TABLE GTEST (GEOM GEOMETRY);
```

   столбец GEOM будет иметь тип (BLOB/GEOMETRY), а не (VARBYTE(1028)/ GEOMETRY).

2) Команда с опцией OFF отключает режим создания столбцов с геометрическими BLOB-данными в текущей сессии.

3) В геометрических BLOB-данных могут храниться значения геометрических типов потенциально большой размерности (LINESTRING, POLYGON, MULTIPOINT, MULTILINESRING, MULTIPOLYGON, GEOMETRYCOLLECTION, GEOMETRY). Значения остальных геометрических типов небольшой размерности (типа POINT) продолжают храниться в виде VARBYTE даже после выполнения данной команды.

4) По умолчанию используется опция OFF.

Управление размером геометрических BLOB-данных

Функция

Ограничение максимального размера геометрических BLOB-данных в пределах текущей сессии.

Спецификация

<управление размером геометрических BLOB-данных>::=
`ALTER DATABASE SET BLOB SIZE LIMIT` <максимальный размер>`;`

<максимальный размер>::= целочисленный литерал.

Общие правила

1) Команда должна использоваться в том случае, если геометрические BLOB-данные могут превышать размер 4000 байт.
2) <максимальный размер> задает максимально возможный в текущей сессии размер (в байтах) буфера оперативной памяти, используемого все соединениями с СУБД для работы с геометрическим BLOB-объектами. Допустимый диапазон значений – от 4000 до 65536 (задаваемое пользователем значение выравнивается на 8).
3) Информация о максимальном размере буфера запоминается в БД ЛИНТЕР и при каждом запуске ядро СУБД ЛИНТЕР резервирует буфер указанного размера в своей оперативной памяти.
4) Чтобы указанный размер стал использоваться ядром СУБД, необходимо в текущей сессии СУБД выполнить данную команду и затем перезапустить ядро СУБД.
5) Задаваемый размер буфера должен позволять загружать в него любые геометрические BLOB-объекты. В противном случае, если геометрический BLOB-объект не может быть полностью размещен в буфере, выдается код завершения 183, а часть размещенного в буфере геометрического BLOB-объекта при ее проверке с помощью функции будет распознана как некорректная.

Проверка корректности геометрических данных

Функция

Проверка корректности значения геометрического типа данных.

Спецификация

```
GeoIsValid(<объект>)
```

<объект> – произвольный геометрический объект.

Возвращаемое значение

1) Значение типа integer:
   - 1 – корректный объект;
   - 0 – некорректный объект.
2) NULL, если геометрический объект пуст.
3) Код завершения СУБД ЛИНТЕР (при неправильном аргументе функции).

Пример

Создаем таблицу и заносим в нее данные:

```
CREATE OR REPLACE TABLE GEOMCOLL_TEST(G GEOMETRYCOLLECTION);
SET CONNECTION GEODATA VALIDITY CHECKING OFF;

/* CORRECT POINT */
INSERT INTO GEOMCOLL_TEST VALUES('GEOMETRYCOLLECTION (POINT (1
 1))');

/* RADIX < 0 */
INSERT INTO GEOMCOLL_TEST VALUES('GEOMETRYCOLLECTION (CIRCLE (1 1,
 -1))');

/* EXTERIOR RING IS NOT CORRECT */
INSERT INTO GEOMCOLL_TEST
VALUES('GEOMETRYCOLLECTION
(POLYGON ((0 0,0 3,3 3, 3 -1, 3 0, 0 0),(1 1,1 2,2 2,2 1,1 1)))');

/* POLYGONS INTERSECTS */
INSERT INTO GEOMCOLL_TEST
VALUES('GEOMETRYCOLLECTION
(MULTIPOLYGON(((0 1,1 2,2 1,1 0,0 1)),((2 2,3 1,2 0,1 1,2 2))))');
```

Проверяем корректность введенных данных:

```
SELECT GeoIsValid(G) FROM GEOMCOLL_TEST;
 INL : начальное время : 17:22:18 конечное время :  17:22:18
```

```
|             1|
|             0|
|             0|
|             0|
```

Приложение 1
Примеры работы с геометрическим VARBYTE-объектом

Примечание
Примеры выполняются в командном интерфейсе INL.

```
!Выполняем запрос в текущей сессии СУБД ЛИНТЕР и перезапускаем её
alter database set blob size limit 65536;

!Перезапускаем СУБД ЛИНТЕР
linternt …

!РАБОТА С ГЕОМЕТРИЧЕСКИМ VARBYTE-ДАННЫМИ

!Создание таблицы с геометрическими VARBYTE-данными

CREATE OR REPLACE TABLE GTEST (OBJECTID INT PRIMARY KEY, NAME
 NVARCHAR (250), GEOM GEOMETRY);

!Загрузка геометрических данных из шестнадцатеричного
 представления:

INSERT INTO GTEST (OBJECTID,NAME,GEOM) VALUES (1 , n'TEST',
 GEOMFROMWKB(hex('0103000000010000000500000000000000000024400000000000
0024400000000000002440000000000000494000000000000049400000000000004940
0000000000004940000000000000244000000000000024400000000000002440'),
 4284));

!Просмотр геометрического значения в виде символьной строки с
 помощью функции !ASTEXT
select cast astext(GEOM) as char(80) from GTEST;
|POLYGON ((10 10,10 50,50 50,50 10,10 10))|

!Просмотр геометрического значения в виде шестнадцатеричной строки
 с помощью функции ASBINARY: (длина должна быть равна 93)
select length(cast asbinary(GEOM) as byte), cast asbinary(GEOM) as
 byte(93) from GTEST;
|93|
0103000000010000000500000000000000000024400000000000002440000000000000
2440000000000000494000000000000049400000000000004940000000000000494000
0000000000244000000000000024400000000000002440

!Вычисление площади геометрической фигуры

select area(GEOM) from GTEST;
```

```
|1600|

!Загрузка геометрических данных их текстового представления
INSERT INTO GTEST (OBJECTID,NAME,GEOM) VALUES (2 , n'TEST2',
 GeomFromText('POLYGON (0 0,0 20,20 20,20 0,0 0)', 4284));

!Просмотр геометрического значения в виде символьной строки
select cast astext(GEOM) as char(80) from GTEST where OBJECTID=2;
|POLYGON ((0 0,0 20,20 20,20 0,0 0)) |

!Найти  геометрические объекты, которые пересекается с заданными
select cast astext(GEOM) as char(80) from GTEST where
 DISJOINT(GEOM, GeomFromText('POLYGON (0 0,0 1,1 1,1 0,0 0)',
 4284)) = 0;
|POLYGON ((0 0,0 20,20 20,20 0,0 0)) |

select cast astext(GEOM) as char(80) from GTEST where
 DISJOINT(GEOM, GeomFromText('POLYGON (49 49,49 50,50 50,50 49,49
 49)', 4284)) = 0;
|POLYGON ((10 10,10 50,50 50,50 10,10 10)) |


!Объединить заданный геометрический объект с объектами из таблицы:
select cast astext(UNION(GEOM, GeomFromText('POLYGON (5 5,5 25,25
 25,25 5,5 5)', 4284))) as char(80) from GTEST;
|POLYGON ((10 25,10 50,50 50,50 10,25 10,25 5,5 5,5 25,10 25)) |
|POLYGON ((20 5,20 0,0 0,0 20,5 20,5 25,25 25,25 5,20 5))      |


!Получить пересечение заданного геометрического объекта с
 объектами из таблицы
select cast astext(INTERSECTION(GEOM, GeomFromText('POLYGON (5 5,5
 25,25 25,25 5,5 5)', 4284))) as char(80) from GTEST;
|POLYGON ((25 10,10 10,10 25,25 25,25 10)) |
|POLYGON ((5 20,20 20,20 5,5 5,5 20))      |


!Получить следующие параметры:
! - симметричную разность геометрических объектов
! - количество объектов в группе
! - объект с заданным порядковым  номером в группе

select
  cast astext(SYMDIFFERENCE(GEOM, GeomFromText('POLYGON (5 5,5
 25,25 25,25 5,5 5)', 4284))) as char(110),
```



```
  NUMGEOMETRIES(SYMDIFFERENCE(GEOM, GeomFromText('POLYGON (5 5,5
 25,25 25,25 5,5 5)', 4284))),
  cast astext(GEOMETRYN(SYMDIFFERENCE(GEOM, GeomFromText('POLYGON
 (5 5,5 25,25 25,25 5,5 5)', 4284)), 2)) as char(80)
from GTEST;

|MULTIPOLYGON (((10 25,10 50,50 50,50 10,25 10,25 25,10 25)),((25
 10,25 5,5 5,5 25,10 25,10 10,25 10)))| 2| POLYGON ((25 10,25 5,5
 5,5 25,10 25,10 10,25 10)) |

|MULTIPOLYGON (((20 5,20 0,0 0,0 20,5 20,5 5,20 5)),((5 20,5 25,25
 25,25 5,20 5,20 20,5 20))) | 2|POLYGON ((5 20,5 25,25 25,25 5,20
 5,20 20,5 20)) |


!Получить разность двух геометрических объектов
select cast astext(DIFFERENCE(GEOM, GeomFromText('POLYGON (5 5,5
 25,25 25,25 5,5 5)', 4284))) as char(200) from GTEST;
|POLYGON ((10 25,10 50,50 50,50 10,25 10,25 25,10 25)) |
|POLYGON ((20 5,20 0,0 0,0 20,5 20,5 5,20 5))          |


!Представить геометрический объект в заданной системе координат
SELECT cast astext(TRANSFORM(GEOM, 1)) as char(80) from GTEST;
|POLYGON ((10 10,10 50,50 50,50 10,10 10)) |
|POLYGON ((0 0,0 20,20 20,20 0,0 0))       |
```

Приложение 2

Примеры работы с геометрическим BLOB-объектом

```
!РАБОТА С ГЕОМЕТРИЧЕСКИМИ BLOB-ДАННЫМИ
!Установка режима хранения в БД геометрических данных
!в виде BLOB-значений
set session blob geometry storage on;

!Создание таблицы с геометрическими BLOB-данными
CREATE OR REPLACE TABLE GTEST (OBJECTID INT PRIMARY KEY, NAME
 NVARCHAR (250), GEOM GEOMETRY);

!Загрузка геометрических данных из шестнадцатеричного
 представления

INSERT INTO GTEST (OBJECTID,NAME,GEOM) VALUES (1 , n'TEST', NULL);
blob insert column=3
0103000000010000000500000000000000000024400000000000002440000000000000
0244000000000000494000000000000049400000000000004940000000000004940
000000000000244000000000000024400000000000002440FFFF0000;
INSERT INTO GTEST (OBJECTID,NAME,GEOM) VALUES (2 , n'TEST', NULL);
blob insert column=3
0103000000010000000600000000000000000024400000000000002440000000000000
0244000000000000494000000000000049400000000000004940000000000004940
000000000000244000000000000024400000000000002440000000000000244000000
00000002440FFFF0000;

!Проверка корректности загруженных геометрических объектов
! первый объект – корректный, второй – нет.

select GEOISVALID(GEOM) from GTEST;
1
0

!Создание таблицы с геометрическими BLOB-данными
CREATE OR REPLACE TABLE GTEST (OBJECTID INT PRIMARY KEY, NAME
 NVARCHAR (250), GEOM GEOMETRY);

!Загрузка геометрических данных из шестнадцатеричного
 представления
INSERT INTO GTEST (OBJECTID,NAME,GEOM) VALUES (1 , n'TEST',
GEOMFROMWKB(hex('0103000000010000000500000000000000000024400000000000002
0244000000000000024400000000000049400000000000004940000000000004940
000000000004940000000000002440000000000000244000000000000002440'),
 4284));
```

```
!Выборка геометрических BLOB-данных

!Информация о BLOB-значении: тип и длина BLOB-значения
select astext(GEOM) from GTEST;
|000000041 000000 000000 0 001|

!Шестнадцатеричное представление геометрического BLOB-значения
blob get column=1;
504F4C59474F4E2028283130203130 2C3130203530 2C3530203530 2C3530203130 2C313
02031302929

!Символьное представление геометрического BLOB-значения
select getblobstr(astext(GEOM), 1, 50), lenblob(astext(GEOM)),
 lenblob(GEOM) from GTEST;
|POLYGON ((10 10,10 50,50 50,50 10,10 10))| 41| 97|

!Получение длины и  атрибутов геометрического BLOB-значения
select lenblob(asbinary(GEOM)), asbinary(GEOM) from GTEST;
|         93|000000093 000002 000002 1 001|

blob get column=2;
0103000000010000000500000000000000000024400000000000002440000000000000
2440000000000000494000000000000049400000000000004940000000000000494000
0000000002440000000000002440000000000002440

select cast getblobstr(asbinary(GEOM), 1, 93) as byte(93) from
 GTEST;
0103000000010000000500000000000000000024400000000000002440000000000000
2440000000000000494000000000000049400000000000004940000000000000494000
0000000002440000000000002440000000000002440

!Вычисление размера геометрического BLOB-значения
select area(GEOM) from GTEST;
1600

!Загрузка текстового представления геометрического BLOB-объекта

INSERT INTO GTEST (OBJECTID,NAME,GEOM) VALUES (2 , n'TEST2',
 GeomFromText('POLYGON (0 0,0 20,20 20,20 0,0 0)', 4284));

Получение текстового представления геометрического BLOB-объекта
select getblobstr(astext(GEOM), 1, 50) from GTEST where
 OBJECTID=2;
|POLYGON ((0 0,0 20,20 20,20 0,0 0)) |
```

```
!Проверка пересечения геометрических BLOB-значений
select getblobstr(astext(GEOM), 1, 50) from GTEST where
 DISJOINT(GEOM, GeomFromText('POLYGON (0 0,0 1,1 1,1 0,0 0)',
 4284)) = 0;
|POLYGON ((0 0,0 20,20 20,20 0,0 0)) |

select getblobstr(astext(GEOM), 1, 50) from GTEST where
 DISJOINT(GEOM, GeomFromText('POLYGON (49 49,49 50,50 50,50 49,49
 49)', 4284)) = 0;
|POLYGON ((10 10,10 50,50 50,50 10,10 10)) |

!Объединение геометрических BLOB-значений
select getblobstr(astext(UNION(GEOM, GeomFromText('POLYGON (5 5,5
 25,25 25,25 5,5 5)', 4284))), 1, 100) from GTEST;
|POLYGON ((10 25,10 50,50 50,50 10,25 10,25 5,5 5,5 25,10 25)) |
|POLYGON ((20 5,20 0,0 0,0 20,5 20,5 25,25 25,25 5,20 5))      |


!Пересечение геометрических BLOB-значений
select getblobstr(astext(INTERSECTION(GEOM, GeomFromText('POLYGON
 (5 5,5 25,25 25,25 5,5 5)', 4284))), 1, 200) from GTEST;
|POLYGON ((25 10,10 10,10 25,25 25,25 10)) |
|POLYGON ((5 20,20 20,20 5,5 5,5 20))      |

!Получить следующие параметры геометрическим BLOB-значений:
! - симметричную разность геометрических объектов
! - количество объектов в группе
! - объект с заданным порядковым  номером в группе

select
  getblobstr(astext(SYMDIFFERENCE(GEOM, GeomFromText('POLYGON (5
 5,5 25,25 25,25 5,5 5)', 4284))), 1, 110),
  NUMGEOMETRIES(SYMDIFFERENCE(GEOM, GeomFromText('POLYGON (5 5,5
 25,25 25,25 5,5 5)', 4284))),
  getblobstr(astext(GEOMETRYN(SYMDIFFERENCE(GEOM,
 GeomFromText('POLYGON (5 5,5 25,25 25,25 5,5 5)', 4284)), 2)), 1,
 80)
from GTEST;
|MULTIPOLYGON (((10 25,10 50,50 50,50 10,25 10,25 25,10 25)),((25
 10,25 5,5 5,5 25,10 25,10 10,25 10))) |
| 2|
|POLYGON ((25 10,25 5,5 5,5 25,10 25,10 10,25 10)) |

|MULTIPOLYGON (((20 5,20 0,0 0,0 20,5 20,5 5,20 5)),((5 20,5 25,25
 25,25 5,20 5,20 20,5 20))) |
| 2|
```

```
|POLYGON ((5 20,5 25,25 25,25 5,20 5,20 20,5 20)) |

!Найти разность двух геометрических BLOB-объектов
select getblobstr(astext(DIFFERENCE(GEOM, GeomFromText('POLYGON (5
 5,5 25,25 25,25 5,5 5)', 4284))), 1, 200) from GTEST;
|POLYGON ((10 25,10 50,50 50,50 10,25 10,25 25,10 25)) |
|POLYGON ((20 5,20 0,0 0,0 20,5 20,5 5,20 5))          |

!Получить название геометрического BLOB-объекта
select GEOMETRYTYPE(GEOM) from GTEST;
|POLYGON|
|POLYGON|

!Получить SRID геометрического BLOB-объекта
select SRID(GEOM) from GTEST;
|4284|
|4284|

!Найти минимальный прямоугольник, ограничивающий заданный
!геометрический BLOB-объект
select cast astext(ENVELOPE(GEOM)) as char(80) from GTEST;
|POLYGON ((10 10,50 10,50 50,10 50,10 10)) |
|POLYGON ((0 0,20 0,20 20,0 20,0 0))       |

Определить размерность геометрического BLOB-объекта
select DIMENSION(GEOM) from GTEST;
|2|
|2|

!function BOUNDARY (LINESTRINGs), LENGTH:
!Найти границу геометрического BLOB-объекта и её длину
select getblobstr(astext(BOUNDARY(GEOM)), 1, 50),
 GLENGTH(BOUNDARY(GEOM)) from GTEST;
|LINESTRING (10 10,10 50,50 50,50 10,10 10)| 160|
|LINESTRING (0 0,0 20,20 20,20 0,0 0)| 80|

!Найти геометрический центр геометрического BLOB-объекта
select cast astext(CENTROID(GEOM)) as char(50), X(CENTROID(GEOM)),
 Y(CENTROID(GEOM)) from GTEST;
|POINT (30 30)| 30| 30|
|POINT (10 10)| 10| 10|

!Определить внешнюю границу геометрического BLOB-объекта
SELECT getblobstr(astext(ExteriorRing(GEOM)), 1, 80) from GTEST;
|LINESTRING (10 10,10 50,50 50,50 10,10 10) |
|LINESTRING (0 0,0 20,20 20,20 0,0 0)       |
```

```
!Преобразовать геометрический BLOB-объект в новую систему
 координат
SELECT getblobstr(astext(TRANSFORM(GEOM, 1)), 1, 80) from GTEST;
|POLYGON ((10 10,10 50,50 50,50 10,10 10)) |
|POLYGON ((0 0,0 20,20 20,20 0,0 0))       |
```

Указатель функций

A
Area, 49, 52
AsBinary, 37
AsText, 36

B
Boundary, 41
Buffer, 61

C
Centroid, 51, 52
Contains, 67
ConvexHull, 62
Crosses, 68

D
Difference, 57
Dimension, 38
Disjoint, 66
Distance, 60
Dwithin, 60

E
EndPoint, 44
Envelope, 40
Equals, 63
ExteriorRing, 50

F
Filter, 55

G
GeoIsValid, 77
GeomCollFromText, 28
GeomCollFromWKB, 34
GeometryFromText, 29
GeometryFromWKB, 34
GeometryN, 54
GeometryType, 38
GeomFromText, 29
GeomFromWKB, 34
GLength, 45, 48

I
InteriorRing, 50
Intersection, 55
Intersects, 64
IsClosed, 47, 48
IsEmpty, 41
IsRing, 46
IsSimple, 42

L
Length, 45, 48
LineFromText, 24
LineFromWKB, 31
LineStringFromText, 24
LineStringFromWKB, 31

M
MLineFromText, 26
MLineFromWKB, 33
MPointFromText, 26
MPointFromWKB, 32
MPolyFromText, 27
MPolyFromWKB, 33
MultiLineStringFromText, 26
MultiLineStringFromWKB, 33
MultiPointFromText, 26
MultiPointFromWKB, 32
MultiPolygonFromText, 27
MultiPolygonFromWKB, 33

N
NumGeometries, 53
NumInteriorRing, 49
NumInteriorRings, 49
NumPoints, 46

O
Overlaps, 64

P
PointFromText, 23
PointFromWKB, 30
PointN, 45
PointOnSurface, 52, 53
PolyFromText, 25
PolyFromWKB, 32
PolygonFromText, 25
PolygonFromWKB, 32

S
SRID, 39
StartPoint, 44
SymDifference, 58

T
to_char, 36
Touches, 68
Transform, 39

U
Union, 56

W
Within, 66

X
X, 43

Y
Y, 43

