Инструкция по установке и эксплуатации
ПК «Циркон-СУБД»

СОДЕРЖАНИЕ

1. Аппаратные требования

1.1. Технические требования к конфигурации компьютера

Для функционирования ПК «Циркон-СУБД» необходима следующая минимальная конфигурация компьютера:

1) системный блок:

- процессор с архитектурой x86-64 (AMD, Intel) с тактовой частотой от 2,4 MHz;
- оперативная память – от 4 Гбайт;
- объем свободного дискового пространства – от 4 Гбайт;
- устройство чтения CD/DVD-дисков;

2) стандартный монитор SVGA;

3) клавиатура (лат./рус.).

Минимально требуемое свободное место для установки изделия на жестком диске – не менее 200 Мбайт (без учета создаваемых баз данных).

1.2. Требования к программным средствам

Сервер и клиентское программное обеспечение ПК «Циркон-СУБД» функционируют на технических средствах под управлением ОС «Циркон 37С» (из состава ПО АСТД 37С СДЕМ.00094-01).

2. Установка ПК «Циркон-СУБД»

ПК «Циркон-СУБД» устанавливается под ОС «Циркон 37С» (из состава ПО АСТД 37С СДЕМ.00094-01).

Дистрибутив ПК «Циркон-СУБД» поставляется в виде скомпилированных пакетов. Пакеты также доступны в репозитории на носителе данных ОС «Циркон 37С» (из состава ПО АСТД 37С СДЕМ.00094-01).

2.1. Установка пакетов

Установка пакетов производится при помощи команды:

```
dpkg -i <путь до .deb пакета postgresql»> <путь до .deb пакета postgresql-common»>
```

При помощи данной команды можно установить любые пакеты из состава дистрибутива, включая инструменты для разработчиков.

Для установки пакетов ПК «Циркон СУБД» из репозитория ОС «Циркон 37С» необходимо выполнить команду:

```
sudo apt install postgresql
```

При выполнении команды устанавливаются и настраиваются все компоненты и зависимости, необходимые для получения готовой к использованию конфигурации. Будет создана база данных по умолчанию, запущен сервер базы данных и настроен автозапуск сервера при загрузке системы.

Если необходимо добавить поддержку географических объектов в ПК «Циркон-СУБД» необходимо установить пакет Postgis:

```
sudo apt install --install-recommends postgis
```

Аналогично можно установить и другие пакеты из состава дистрибутива:

```
sudo apt install <имя пакета>
```

Более подробная информация по настройке сервера (в том числе: настройка параметров, расположение файлов, подключения и аутентификация, потребление ресурсов, журнал предзаписи, репликация, планирование запросов, регистрация ошибок и протоколирование работы сервера, статистика времени выполнения, автоматическая очистка, параметры клиентских сеансов по умолчанию, управление блокировками, обработка ошибок, параметры СЗИ, параметры ограничения потенциально опасного функционала, параметры аудита событий СУБД, предопределенные параметры, внесистемные параметры, краткие аргументы), аутентификации клиента, резервному копированию и восстановлению, дискреционному разграничению доступа, мандатному разграничению доступа и его настройке, аудите событий СУБД, тестированию средств защиты информации, очистке дисковой памяти приведена в Руководстве администратора на ПК «Циркон-СУБД».

3. Руководство пользователя

3.1. Начало работы

1. Включить компьютер.

2. Дожаться загрузки операционной системы.

3. Авторизоваться от имени пользователя ОС.

4. Вызвать эмулятор терминала из главного меню рабочего стола:

Приложение > Системные > Терминал MATE

5. Ввести команду для подключения к СУБД:

```
psql –h <IP-адрес сервера СУБД> -d <имя базы данных> -U <имя пользователя СУБД>
```

6. Начало работы с СУБД.

Например:

```
[имя базы данных]=>CREATE TABLE <название таблицы> [параметры]…
```

Команды вводятся в *psql* – интерактивном терминальном приложении СУБД.

3.2. Лексическая структура

Текст на языке SQL состоит из последовательности команд. Команда, в свою очередь, состоит из последовательности компонентов, завершаемой точкой с запятой «;». Конец текста так же завершает команду.

Компонентом программы может быть ключевое слово, идентификатор, идентификатор в кавычках, строка (константа) или спецсимвол. Обычно компоненты разделяются пробельными символами (пробелами, символами табуляции и новой строки), но иногда в этом нет необходимости (например, если компоненты разделяются спецсимволами).

Текст на языке SQL может дополнительно содержать комментарии. Комментарии не являются лексемами и по своему действию эквивалентны пробельным символам.

Ниже, для примера, приведен синтаксически корректный текст на языке SQL:

```
SELECT * FROM MY_TABLE;
UPDATE MY_TABLE SET A = 5;
INSERT INTO MY_TABLE VALUES (3, 'hi there');
```

Этот текст состоит из трех команд, по одной в каждой строке (это, однако, необязательно: несколько команд можно разместить и в одной строке, так же как и разбить для наглядности одну команду на несколько строк).

Синтаксис языка SQL не определяет строго, какие компоненты обозначают команды, а какие – идентификаторы или параметры. Как правило, первые несколько компонентов являются именем команды. О приведенном выше примере можно сказать, что он содержит команды SELECT, UPDATE и INSERT. Однако, например, команда UPDATE всегда требует компонент SET в определенной позиции, а данный вариант команды INSERT – компонент VALUES.

Точные лексические правила для каждой команды приведены в документе «Руководство пользователя. Часть 2. Описание команд».

Идентификаторы и ключевые слова

Такие компоненты, как: SELECT, UPDATE или VALUES являются примерами ключевых слов, т.е. слов, которые имеют фиксированное значение в языке SQL. Компоненты MY_TABLE и A – это примеры идентификаторов. Они идентифицируют имена таблиц, столбцов и других объектов базы данных (БД), в зависимости от команды, в которой они используются. Поэтому иногда их также называют именами. Ключевые слова и идентификаторы имеют одну и ту же лексическую структуру, т.е. без знания языка нельзя понять, является ли компонент идентификатором или ключевым словом.

Идентификаторы и ключевые слова языка SQL должны начинаться с буквы (a–z, A–Z, также допустимы и нелатинские буквы или диакритические знаки) или символа подчеркивания «_». Последующие символы идентификатора или ключевого слова могут быть буквами, цифрами (0–9) или символами подчеркивания. Стандарт SQL не определяет ни одного ключевого слова,

начинающегося или заканчивающегося подчеркиванием или содержащего цифры.

Система использует в идентификаторах не более 255 символов (это значение задается при компиляции программы с помощью специальной константы NAMEDATALEN и может быть изменено только разработчиками). В тексте на языке SQL могут быть использованы и более длинные идентификаторы, однако при его обработке они будут соответствующим образом усечены.

Идентификаторы и ключевые слова нечувствительны к регистру. Таким образом, команда:

```
UPDATE MY_TABLE SET A = 5;
```

может быть записана как:

```
uPDaTE my_TabLE SeT a = 5;
```

Часто используется неформальное соглашение для различения в тексте ключевых слов и имен. Например, ключевые слова записываются символами в верхнем регистре, а имена – в нижнем:

```
UPDATE my_table SET a = 5;
```

Второй тип идентификаторов, идентификатор в кавычках, формируется путем заключения произвольных последовательностей символов в двойные кавычки. Такой идентификатор никогда не бывает ключевым словом. Таким образом, идентификатор «select» можно использовать в качестве имени столбца или таблицы, учитывая при этом, что select без кавычек будет восприниматься как ключевое слово и вызовет синтаксическую ошибку при ее использовании там, где ожидается имя столбца или таблицы. Вот пример использования такого идентификатора:

```
UPDATE «Моя таблица» SET «Имя столбца» = 5;
```

Идентификатор в кавычках может содержать любые символы кроме двойной кавычки (чтобы включить в него двойную кавычку, ее надо записать как две двойные кавычки подряд). Такие идентификаторы позволяют задавать произвольные имена для таблиц и столбцов, что в противном случае было бы

невозможным (например, имена, содержащие пробелы). Длина таких идентификаторов ограничена той же величиной, что и обычных идентификаторов.

Идентификаторы в кавычках чувствительны к регистру, тогда как прочие компоненты языка SQL всегда переводятся в нижний регистр перед обработкой. Например, FOO, «foo» и foo для ПК «Циркон-СУБД» одинаковы, а «Foo» и «FOO» – различны.

Примечание. Перевод имен в нижний регистр не соответствует требованиям стандарта SQL (стандарт требует переводить их в верхний регистр).

Таким образом, согласно стандарту одинаковыми именами в этом примере являются foo, «FOO» и «foo».

Еще один вариант идентификаторов в кавычках позволяет использовать символы Unicode по их кодам. Такой идентификатор начинается с U& (строчная или заглавная U и амперсанд), а затем сразу без пробелов идет двойная кавычка, например U&»foo» (при этом возникает неоднозначность с оператором &, чтобы ее избежать, необходимо окружать этот оператор пробелами.) Затем в кавычках можно записывать символы Unicode двумя способами: обратная косая черта, а за ней код символа из четырех шестнадцатеричных цифр, либо обратная косая черта, знак плюс, а затем код из шести шестнадцатеричных цифр. Например, идентификатор «data» можно записать так:

```
U&»d\0061t\+000061»
```

В следующем примере закодировано слово «слон», записанное кириллицей:

```
U&»\0441\043B\043E\043D»
```

Если использовать другой спецсимвол, а не обратную косую черту, его можно указать, добавив UESCAPE после строки, например:

```
U&"d!0061t!+000061" UESCAPE '!'
```

В качестве спецсимвола можно выбрать любой символ, кроме шестнадцатеричной цифры, знака плюс, апострофа, кавычки или пробельного

символа. Спецсимвол заключается не в двойные кавычки, а в апострофы, после UESCAPE.

Чтобы сделать спецсимволом знак апострофа, следует написать его дважды.

Записывать суррогатные пары UTF-16 и таким образом составлять символы с кодами больше чем U+FFFF можно либо в четырех-, либо в шестизначной форме, но наличие шестизначной формы технически делает это ненужным. Суррогатные пары не сохраняются непосредственно, а объединяются в один символ, который затем кодируется в UTF-8.

Когда кодировка сервера не UTF-8, символ с кодом, указанным этой спецпоследовательностью, преобразуется в фактическую кодировку сервера; если такое преобразование невозможно, выдается ошибка.

Константы

В ПК «Циркон-СУБД» существует три вида констант с неявным заданием типа: строки, последовательности битов и числа. Константы могут быть заданы и с явным определением типа, что может обеспечить их более эффективную обработку в системе.

Строковые константы

Строковая константа в языке SQL – это произвольная последовательность символов, заключенная в одинарные кавычки. Например:

```
'Это строковая константа'
```

Стандарт SQL позволяет использовать в строке символ одинарной кавычки, если он записан в виде последовательности из двух таких кавычек подряд:

```
'Одинарная кавычка («) в строковой константе'
```

Две строковые константы, разделенные только пробельными символами и, как минимум, одним символом новой строки, объединяются в одну строковую константу. Например:

```
SELECT 'foo'
'bar';
```

аналогично:

```
SELECT 'foobar';
```

при этом, однако, запись:

```
SELECT 'foo' 'bar';
```

не является допустимой с точки зрения синтаксиса (в этом ПК «Циркон-СУБД» полностью подчиняется правилам стандарта).

Строковые константы со спецпоследовательностями в стиле C

ПК «Циркон-СУБД» также принимает «спецпоследовательности», что является расширением стандарта SQL. Строка со спецпоследовательностями начинается с буквы E (заглавной или строчной), стоящей непосредственно перед апострофом, например: E'foo'.

Примечание. Когда константа со спецпоследовательностью разбивается на несколько строк, букву E нужно поставить только перед первым открывающим апострофом.

Внутри таких строк символ обратной косой черты (\) начинает C-подобные спецпоследовательности, в которых сочетание обратной косой черты со следующим символом(ами) дает определенное байтовое значение, как показано в таблице 1.

Таблица 1 – Спецпоследовательности

Спецпоследовательность	Интерпретация
\b	Символ «забой»
\f	Подача формы
\n	Новая строка
\r	Возврат каретки
\t	Табуляция
o, *oo*, *ooo* (*o* = 0–7)	Восьмеричное значение байта
\x*h*, \x*hh* (*h* = 0–9, A–F)	Шестнадцатеричное значение байта
\u*xxxx*, \U*xxxxxxxx* (*x* = 0–9, A–F)	16- или 32-битный шестнадцатеричный код символа Unicode

Любой другой символ, идущий после обратной косой черты, воспринимается буквально. Таким образом, чтобы включить в строку обратную косую черту, нужно написать две косых черты (\\). Так же можно включить в строку апостроф, написав \', в дополнение к обычному способу ''.

ВНИМАНИЕ! Если параметр конфигурации standard_conforming_strings имеет значение off, СУБД распознает обратную косую черту как спецсимвол и в обычных строках, и в строках со спецпоследовательностями. Если необходимо, чтобы обратная косая черта представляла специальный символ, следует задать строковую константу с E.

В дополнение к standard_conforming_strings поведением обратной косой черты в строковых константах управляют параметры escape_string_warning и backslash_quote.

Строковая константа не может включать символ с кодом 0.

Строковые константы со спецпоследовательностями Unicode

ПК «Циркон-СУБД» также поддерживает еще один вариант спецпоследовательностей, позволяющий включать в строки символы Unicode по их кодам. Строковая константа со спецпоследовательностями Unicode начинается с U& (строчная или заглавная U и амперсанд), а затем сразу без пробелов идет апостроф, например U&'foo'.

Затем в апострофах можно записывать символы Unicode двумя способами: обратная косая черта, а за ней код символа из четырех шестнадцатеричных цифр, либо обратная косая черта, знак плюс, а затем код из шести шестнадцатеричных цифр. Например, строку 'data' можно записать так:

```
U&'d\0061t\+000061'
```

В следующем примере закодировано слово «слон», записанное кириллицей:

```
U&'\0441\043B\043E\043D'
```

Если использовать другой спецсимвол, а не обратную косую черту, его можно указать, добавив UESCAPE после строки, например:

```
U&'d!0061t!+000061' UESCAPE '!'
```

В качестве спецсимвола можно выбрать любой символ, кроме шестнадцатеричной цифры, знака плюс, апострофа, кавычки или пробельного символа.

Чтобы включить спецсимвол в строку буквально, напишите его дважды.

Записывать суррогатные пары UTF-16 и таким образом составлять символы с кодами больше чем U+FFFF можно либо в четырех-, либо в шестизначной форме, но наличие шестизначной формы технически делает это ненужным (суррогатные пары не сохраняются непосредственно, а объединяются в один символ, который затем кодируется в UTF-8).

Когда кодировка сервера не UTF-8, символ с кодом, указанным этой спецпоследовательностью, преобразуется в фактическую кодировку сервера; если такое преобразование невозможно, выдается ошибка.

Спецпоследовательности Unicode в строковых константах работают, только когда параметр конфигурации standard_conforming_strings равен on. Это объясняется тем, что иначе клиентские программы, проверяющие SQL-операторы, можно будет ввести в заблуждение и эксплуатировать это как уязвимость, например, для SQL-инъекций. Если этот параметр имеет значение off, эти спецпоследовательности будут вызывать ошибку.

Строковые константы, заключенные в знаки долларов

Стандартный синтаксис для строковых констант обычно достаточно удобен, но он может плохо читаться, когда строка содержит много апострофов или обратных косых черт, так как каждый такой символ приходится дублировать. Чтобы и в таких случаях запросы оставались читаемыми, в ПК «Циркон-СУБД» есть способ записи строковых констант – «заключение строк в доллары». Строковая константа, заключенная в доллары, начинается со знака доллара «$», необязательного «тега» из нескольких символов и еще одного знака доллара, затем содержит обычную последовательность символов, составляющую строку, и оканчивается знаком доллара, тем же тегом и замыкающим знаком доллара. Например, строку «Кот д'Ивуар» можно записать в долларах двумя способами:

```
$$Кот д'Ивуар$$
$SomeTag$Кот д'Ивуар$SomeTag$
```

Внутри такой строки апострофы не нужно записывать особым образом. На самом деле, в строке, заключенной в доллары, все символы можно записывать в чистом виде: содержимое строки всегда записывается буквально. Ни обратная косая черта, ни даже знак доллара не являются спецсимволами, если только они не образуют последовательность, соответствующую открывающему тегу.

Строковые константы в долларах можно вкладывать друг в друга, выбирая на разных уровнях вложенности разные теги. Чаще всего это используется при написании определений функций. Например:

```
$function$
BEGIN
    RETURN ($1 ~ $q$[\t\r\n\v\\]$q$);
END;
$function$
```

Здесь последовательность q[\t\r\n\v\\]q представляет в долларах текстовую строку [\t\r\n\v\\], которая будет обработана, когда ПК «Циркон-СУБД» будет выполнять эту функцию. Но так как эта последовательность не соответствует внешнему тегу в долларах ($function$), с точки зрения внешней строки это обычные символы внутри константы.

Тег строки в долларах, если он присутствует, должен соответствовать правилам, определенным для идентификаторов без кавычек, и к тому же не должен содержать знак доллара. Теги регистрозависимы, так что tagString contenttag – правильная строка, а TAGString contenttag – нет.

Строка в долларах, следующая за ключевым словом или идентификатором, должна отделяться от него пробельными символами, иначе доллар будет считаться продолжением предыдущего идентификатора.

Заключение строк в доллары полезно, когда нужно представить строковую константу внутри другой строки, что часто требуется в определениях процедурных функций. Ограничившись только апострофами, каждую обратную косую черту в приведенном примере пришлось бы записывать четырьмя такими

символами, которые бы затем уменьшились до двух при разборе внешней строки, и наконец, до одного при обработке внутренней строки во время выполнения функции.

Битовые строковые константы

Битовые строковые константы выглядят как строковые константы с символом B (в нижнем или верхнем регистре), стоящим сразу перед открывающей кавычкой (без пробела), например – B'1001'. В битовых константах в такой форме могут использоваться только символы «0» или «1».

Битовые константы также могут быть определены с помощью шестнадцатеричной формы записи чисел, с использованием в качестве стартового символа X (в верхнем или нижнем регистре), например – X'1FF'. Каждый символ такой строки представляет собой четыре бита результирующей битовой константы.

Обе формы записи допускают перенос строк так же, как и обычные строковые константы. Однако заключать в доллары битовые строки нельзя.

Числовые константы

Числовые константы могут быть записаны в любой из следующих основных форм:

```
цифры
цифры.[цифры][E[+-]цифры]
цифры].цифры[E[+-]цифры]
цифрыE[+-]цифры
```

где «цифры» – это одна или более десятичных цифр (от 0 до 9).

При этом, по крайней мере, одна цифра должна находиться до или после десятичной точки, если она используется и, по крайней мере, одна цифра должна находиться после маркера экспоненты (E), если он используется. Числовая константа не может содержать пробелы или символы, не входящие в ее определение.

Символы «+» или «–» перед числовой константой являются не частью ее определения, а операторами, к ней примененными.

Ниже приведены несколько примеров допустимых числовых констант:

```
42
3.5
4.
001
5e2
1.925e-3
```

Числовые константы, которые не содержат ни десятичной точки, ни маркера экспоненты, интерпретируются как числа типа «integer», если их значение может быть представлено с помощью 32-битного числа, как числа типа «bigint», если они могут быть представлены с помощью 64-битного числа или как числа типа «numeric» во всех остальных случаях. Константы, содержащие десятичную точку и/или маркер экспоненты, всегда представляются типом «numeric».

Начальная интерпретация типа числовой константы – это первый шаг алгоритма определения ее действительного типа. В большинстве случаев константа будет представлена наиболее подходящим типом по контексту ее применения. При необходимости можно явно задать тип числовой константы, используя явное приведение типа. Например, можно задать интерпретацию константы типа «numeric» как константы типа «real», записав ее как:

```
REAL '1.23'
```

или как:

```
1.23::REAL
```

Константы других типов

Константа произвольного типа может быть задана с помощью одного из следующих обозначений:

```
<имя_типа> 'строка'
'строка'::<имя_типа>
CAST('строка' AS <имя_типа>)
```

Текст строки передается в подпрограмму преобразования входных данных для типа «имя_типа». Результатом является константа указанного типа. Явное

приведение типа может быть опущено, если существует только один вариант интерпретации типа константы (например, когда она передается как аргумент в неперегруженную функцию). В этом случае необходимое преобразование типа будет выбрано автоматически.

Принудительное задание интерпретации типа константы может быть так же задано с помощью функционального стиля:

```
<имя_типа>('строка')
```

Но не все имена типов допускают использование такого синтаксиса.

«::», CAST() и функциональную запись преобразования типов можно использовать так же для преобразования типов в различных выражениях. Более подробно 3.2.9.

Форма «имя_типа> 'строка'» может использоваться только для определения типа строковой константы. Другим ограничением этой формы задания типа является то, что она не может быть использована для задания типа константных массивов.

Операторы

Оператор – это последовательность длиной до 255 символов из следующего списка:

```
+ - * / < > = ~ ! @ # % ^ & | ' ?
```

Существуют определенные ограничения, помимо длины, накладываемые на имена операторов:

– последовательности символов -- и /* – не могут быть использованы как часть имени оператора, т.к. будут восприняты как начало комментария;

– многосимвольное имя оператора не может заканчиваться на символы плюс «+» или минус «-», если оно не содержит также одного из следующих символов:

```
~ ! @ # % ^ & | ' ?
```

Например, «@-» – это допустимое имя оператора, «a *-» – нет. Это ограничение позволяет ПК «Циркон-СУБД» правильно разбирать тексты, совместимые со стандартом SQL, не требуя наличия пробелов между

компонентами.

При работе с именами операторов, не определенными в стандарте SQL, обычно необходимо разделять соседние операторы пробелами для устранения неоднозначности. Например, если левый одноместный оператор назван именем «@», то нельзя записать последовательность операций как «X*@Y». Вместо этого необходимо написать «X* @Y», чтобы ПК «Циркон-СУБД» интерпретировал эту последовательность именно как два оператора с именами «*» и «@», а не один с именем «*@».

Специальные символы

Ряд символов, не являющихся алфавитно-цифровыми, имеют специальное значение. Подробнее об их использовании можно узнать там, где описывается элемент с соответствующим синтаксисом. В этом пункте приведены только некоторые общие сведения:

– знак доллара «$», за которым идут цифры, используется для ссылки на параметры функции в ее описании. В остальных контекстах он может быть частью имени оператора;

– круглые скобки «()» имеют обычное значение и применяются для группировки выражений и повышения приоритета операций. В ряде случаев они так же являются частью синтаксиса некоторых SQL-команд;

– квадратные скобки «[]» используются для выборки элементов массива. Подробнее о массивах говорится в 7.15;

– запятые «,» используются в некоторых синтаксических конструкциях для разделения элементов списка;

– точка с запятой «;» завершает команду SQL. Она не может появиться внутри команды, за исключением ее использования в строковой константе или идентификаторе в кавычках;

– двоеточие «:» используется для выделения «срезов» массивов. В некоторых диалектах языка SQL (например, во встроенном SQL) оно предваряет имена переменных;

– звездочка «*» имеет специальное значение в команде SELECT или при

использовании в агрегатной функции «count»;

– точка «.» используется в константах с плавающей точкой (в качестве разделителя) и разделяет имена схем, таблиц и столбцов.

Комментарии

Комментарий – это произвольная последовательность символов, которая начинается с двойного дефиса и продолжается до конца строки. Например:

```
-- Это комментарий в стиле SQL
```

Дополнительно можно использовать блочные комментарии в стиле C:

```
/*  многострочный комментарий
* с вложенностью: /* вложенный многострочный комментарий */
*/
```

где «комментарий» начинается с пары символов «/*» и продолжается до соответствующей ему по уровню вложенности пары символов «*/». Как определено в SQL, но не в C/C++, блочные комментарии могут вкладываться и, таким образом, можно комментировать блоки кода, которые уже содержат блочные комментарии.

Комментарии исключаются из входного потока перед этапом синтаксического анализа и заменяются пробелами.

Приоритеты операторов

Таблица 2 показывает приоритеты и очередность операторов в ПК «Циркон-СУБД».

Таблица 2 – Приоритет операторов (от большего к меньшему)

Оператор/элемент	Очередность	Описание
.	слева-направо	Разделитель имен схем/таблиц/столбцов
::	слева-направо	Преобразование типа
[]	слева-направо	Выбор элемента массива
+ -	справа-налево	Унарный плюс, унарный минус
^	слева-направо	Возведение в степень
* / %	слева-направо	Умножение, деление, остаток от деления

Оператор/элемент	Очередность	Описание
+ -	слева-направо	Сложение, вычитание
(любой другой оператор)	слева-направо	Все другие встроенные и пользовательские операторы
BETWEEN IN LIKE ILIKE SIMILAR		Проверка диапазона, проверка членства, сравнение строк
< > = <= >= <>		Операторы сравнения
IS ISNULL NOTNULL		IS TRUE, IS FALSE, IS NULL, IS DISTINCT FROM и т. д.
NOT	справа-налево	Логическое НЕТ
AND	слева-направо	Логическое И
OR	слева-направо	Логическое ИЛИ

Правила приоритета операторов также применяются к операторам, определенным пользователем с теми же именами, что и вышеперечисленные встроенные операторы. Например, если определить оператор «+» для некоторого нестандартного типа данных, он будет иметь тот же приоритет, что и встроенный оператор «+», независимо от того, что он делает.

При использовании полного имени оператора с ключевым словом OPERATOR, например:

```
SELECT 3 OPERATOR(pg_catalog.+) 4;
```

старшинство оператора будет соответствовать строке «(любой другой оператор)» (см. таблицу 1) вне зависимости от реально используемого оператора.

Более подробная информация о синтаксисе языка SQL, об определении данных (в том числе: основы таблиц, значения по умолчанию, генерируемые столбцы, ограничения, системные столбцы, редактирование структуры таблиц, права пользователей, политики защиты строк, схемы (пространства имен), наследование, секционирование таблиц, сторонние данные, другие объекты БД, отслеживание зависимостей), модификации данных, запросах, типах данных, функциях и операторах, преобразовании типов, индексах, полнотекстовом поиске, управлении конкурентным доступом, оптимизации производительности, параллельном запросе, а также описание команд SQL и клиентских приложений приведена в Руководстве пользователя на ПК «Циркон-СУБД».

АКЦИОНЕРНОЕ ОБЩЕСТВО
"МНОГОПРОФИЛЬНОЕ
ВНЕДРЕНЧЕСКОЕ ПРЕДПРИЯТИЕ
"СВЕМЕЛ"

127254, г. Москва, Огородный пр., д. 5, стр.5
Тел/Факс: +7(495) 926-7187, +7(499) 750-7065
E-mail: post@swemel.ru