1 ЛАБОРАТОРНАЯ РАБОТА № 1. АРХИТЕКТУРА РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Цель работы: изучить архитектуру баз данных СУБД, поддерживающих реляционную модель данных; научиться средствами СУБД создавать базу данных и работать с ее объектами.

1.1 Краткие теоретические сведения

Архитектуру баз данных можно рассмотреть на примере системы управления базами данных (СУБД) PostgreSQL.

СУБД PostgreSQL – это кросплатформенная свободно распространяемая объектно-реляционная система управления базами данных, наиболее развитая из открытых (open source) СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных.

Ниже приводятся основные возможности PostgreSQL.

Поддержка объектно-реляционной модели. Работа с данными в PostgreSQL основана на объектно-реляционной модели, что позволяет задействовать сложные процедуры и системы правил. Примерами нетривиальных возможностей этой категории являются декларативные запросы SQL, контроль параллельного доступа, поддержка многопользовательского доступа, транзакции, оптимизация запросов, поддержка наследования и массивов.

Простота расширения. В PostgreSQL поддерживаются пользовательские операторы, функции, методы доступа и типы данных.

Полноценная поддержка SQL. PostgreSQL соответствует базовой спецификации SQL99.

Гибкость API. Гибкость API PostgreSQL позволяет легко создавать интерфейсы к реляционной СУБД PostgreSQL. В настоящее время существуют программные интерфейсы для Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/ C+ и Pike.

Процедурные языки. В PostgreSQL предусмотрена поддержка внутренних процедурных языков, в том числе специализированного языка PL/pgSQL, являющегося аналогом PL/SQL, процедурного языка Oracle. Одним из преимуществ PostgreSQL является возможность использования Perl, Python и TCL в качестве внутренних процедурных языков.

Технология MVCC. MVCC (Multiversion Concurrency Control) используется в PostgreSQL для управления конкурентным доступом к данным на многовариантной основе. Эта технология позволяет предотвращать лишние блокировки (locking) операций чтения операциями, производящими обновление записей. PostgreSQL отслеживает все транзакции, выполняемые пользователями базы данных, что позволяет работать с записями без ожидания их освобождения. На практике это

означает, что при запросе к БД каждая транзакция видит как бы снимок данных (версию) на момент этого снимка, а не текущее состояние данных. Таким образом, транзакции защищаются от просмотра незафиксированных данных, которые в данный момент могут только формироваться конкурентными транзакциями в тех же самых строках таблицы. Основное преимущество MMVC состоит в том, что чтение данных никогда не блокирует запись, а запись никогда не блокирует чтение. MMVC позволяет избегать явного блокирования на уровне таблиц и отдельных записей, которое используется в традиционных СУБД, и, таким образом, минимизирует блокирование данных и увеличивает производительность в многопользовательских системах БД. Также реализовано отслеживание взаимных блокировок (deadlocks).

Клиент-серверная архитектура. В PostgreSQL используется архитектура «клиент-сервер» с распределением процессов между пользователями. В целом она напоминает методику работы с процессами в сервере Apache. Главный (master) процесс создает дополнительные подключения для каждого клиента, пытающегося установить соединение с PostgreSQL.

Полнотекстовый поиск. Начиная с версии 8.3 в ядро PostgreSQL включена функция полнотекстового поиска. Полнотекстовый поиск позволяет создавать такие запросы к текстовым документам, которые могут гибко настраиваться в зависимости от конкретных потребностей.

Сохраняющая регистрация WAL. WAL (Write-Ahead Logging) является стандартным методом для обеспечения целостности данных. Сохраняющая регистрация – метод регистрации (журналирования) транзакций, при котором запись в журнале делается до записи данных. Используется также в MS SQL Server. Суть WAL заключается в том, что изменения в файлах данных (таблицы и индексы) должны быть внесены только после записи в журнал (log), в котором фиксируются эти изменения. Эта процедура позволяет не переписывать страницы данных на диске при каждой транзакции, так как в случае аварии можно восстановить базу данных с помощью журнала. Механизм WAL обеспечивает следующие преимущества:

- повышение производительности работы СУБД за счет того, что записываются только внесенные изменения без переписывания всех данных в таблицах;
- повышение надежности хранения данных за счет предыдущего сохранения буферизованных данных в WAL;
- возможность отката состояния БД на любой момент времени, путем применения WAL к существующей резервной копии.

Репликация и технология Hot Standby. Начиная с версии 9.0, на основе WAL введена репликация по технологии Hot Standby. Технология позволяет получить на сервере вторую базу данных, которая является актуальной копией оригинальной базы данных, доступной лишь для чтения.

Технология может быть использована также и на отдаленном сервере, который подключается к primary- или master-серверу и загружает из него WAL-логи, предоставляя онлайновую репликацию базы данных и поддерживая копию базы данных на отдаленном сервере в актуальном состоянии, а также делая эту копию доступной для запросов на чтение.

Наследование. Таблицы могут наследовать характеристики и наборы полей от других таблиц (родительских). При этом данные, которые добавляются в порожденную таблицу, автоматически будут принимать участие в запросах к родительской таблице. Данная функция в настоящее время не является полностью реализованной, однако ее можно использовать;

Гибкая настройка сервера. Основной конфигурационный файл postgresql.conf включает более 150 настраиваемых параметров по разделам: файлы и пути к ним, авторизация и безопасность, выделение ресурсов и т.д. Дополнительный конфигурационный файл pg_hba.conf включает в себя настройку доступа к отдельным БД, такие как указание конкретных IP-адресов и (или) сетей, из которых разрешен доступ, а также метод авторизации для доступа в БД и возможность включения безопасных (зашифрованных) соединений.

1.1.1 Физическая архитектура баз данных PostgreSQL

1.1.1.1 Схема размещения файлов базы данных

Все данные, необходимые для кластера БД хранятся в каталоге, который, как правило, называют PGDATA (как и соответствующую переменную окружения). Несколько кластеров, управляемых разными экземплярами сервера, могут существовать на одной машине.

Каталог PGDATA содержит несколько подкаталогов и файлов управления, таких как: PG_VERSION – файл, содержащий номер версии PostgreSQL, base – подкаталог, содержащий подкаталоги баз данных, и др., а также необходимые файлы конфигурации кластера postgresql.conf , pg_hba.conf и pg_ident.conf .

Для каждой БД в кластере есть подкаталог PGDATA / base , имя которого совпадает с OID (object identifier – идентификатор объекта) базы данных, хранящимся в системной таблице pg_database. Этот подкаталог используется для хранения файлов базы данных, в частности, его системных каталогов.

Каждая таблица и индекс сохраняется в отдельном файле с именем, совпадающим с дескриптором таблицы (filenode). Кроме того каждая таблица или индекс имеет **карту свободного пространства** (free space map), в которой хранится информация о доступном объеме памяти. Карта свободного пространства хранится в файле с именем, состоящим из дескриптора таблицы или индекса и суффикса _fsm.

Каждое отношение имеет также **карту видимости** (visibility map) ,

чтобы отслеживать, какие страницы содержат кортежи, видимые для всех активных транзакций Эта карта хранится в файле с именем, состоящим из дескриптора отношения и суффикса _vm. Например, если дескриптор отношения есть 12345, карта видимости хранится в файле с именем 12345_vm в том же каталоге, что и основной файл отношения. Обратите внимание, что индексы не имеют карт видимости.

В карте видимости на каждую страницу отводится 1 бит. Установленный бит означает, что все кортежи на странице видимы для всех транзакций, т.е. страница не содержит кортежей, которые необходимо очищать.

Когда таблица или индекс превышают 1 Гб, они делятся на гигабайтные сегменты (1 Гб – это размер по умолчанию, его можно настроить с помощью опции конфигурации -segsize при сборке PostgreSQL). Имя файла первого сегмента совпадает с filenode, последующие сегменты называются filenode.1, filenode.2 и т.д. Такая схема позволяет избежать проблем для платформ, которые имеют ограничения на размер файла.

PostgreSQL поддерживает **табличные пространства** (**tablespaces**), которые позволяют задать место хранения объектов БД в файловой системе. Сначала создается табличное пространство с определенным именем. Далее, это имя может быть использовано при создании таблиц, чтобы разместить эти таблицы именно в данном табличном пространстве. Каждое определяемое пользователем табличное пространство имеет ссылку внутри каталога PGDATA/pg_tblspc, которая указывает на физический каталог этого табличного пространства. Эта ссылка имеет такое же имя, как и OID табличного пространства. Внутри физического каталога табличного пространства есть подкаталог с именем, которое зависит от версии PostgreSQL сервера, например, PG_9.0_201008051. В этом подкаталоге содержатся подкаталоги для каждой БД (их имена совпадают с OID базы данных), которая имеет элементы в данном табличном пространстве. Для именования записи таблиц и индексов в этих подкаталогах используется схема, основанная на filenode.

Временные файлы (для таких операций, как сортировка больших объемов данных) создаются в PGDATA/base/pgsql_tmp, или в подкаталоге pgsql_tmp каталога табличного пространства, если для них указано табличное пространство, отличное от табличного пространства по умолчанию. Имя временного файла имеет вид pgsql_tmpPPP.NNN, где PPP – OID вычислительной машины базы данных (backend), NNN – метки различных временных файлов.

1.1.1.2 Хранение больших значений. Таблицы TOAST

Для хранения больших по размеру данных в PostgreSQL используется техника TOAST (The Oversized-Attribute Storage Technique). PostgreSQL использует фиксированный размер страницы (обычно 8 Кб), и не позволяет

кортежам занимать несколько страниц. Таким образом, невозможно хранить очень большие значения полей непосредственно в таблице. Чтобы преодолеть это ограничение, большие значения сжимаются и/или разбиваются на несколько физических строк. Этот процесс прозрачен для пользователя.

TOAST поддерживается только для определенных типов данных. Для поддержки TOAST тип данных должен иметь переменную длину (varlena), причем первое 32-разрядное слово содержит общую длину значения в байтах, включая само это слово.

TOAST резервирует два бита varlena-слова: старшие биты для машин с обратным порядок байтов (big-endian) и младшие биты для машин с прямым порядком байтов (little-endian). Тем самым размер любого значения ограничивается 1 Гб (2^{30} – 1 байт). Нулевые значения битов соответствуют обычным (не TOAST) значениям. В противном случае возможны две комбинации:

- самый старший/младший бит varlena-слова установлен, смежный бит сброшен. Это указывает на то, что данное значение имеет 1-байтовое, а не 4-байтовое , varlena-слова, а оставшиеся биты этого слова дают общий размер значения (включая байт длины) в байтах. Если остальные биты varlena-слова равны нулю, то данное значение является указателем на данные, хранящиеся в отдельной TOAST-таблице. При этом размер TOAST -указателя дает второй байт значения. Данные с однобайтовыми заголовками не выравниваются по какой-либо конкретной границе;
- самый старший/младший бит varlena-слова сброшен, смежный бит установлен. Это говорит о том, что значение было сжато и должно быть распаковано перед использованием. В этом случае оставшиеся биты длины слова дают общий размер сжатых данных, а не исходных данных Сжатие также возможно и для значений, хранящихся в TOAST-таблицах (информацию об этом содержит TOAST-указатель).

Механизм TOAST запускается только тогда, когда строка, которая будет храниться в таблице превышает TOAST_TUPLE_THRESHOLD байт (обычно 2 Кб). Значение столбца, для которого поддерживается TOAST, будет сжиматься и/или перемещаться в TOAST-таблицу пока длина строки не станет короче, чем TOAST_TUPLE_TARGET байт (также обычно 2 Кб).

Механизм TOAST использует четыре различные стратегии для хранения данных:

– PLAIN предусматривает либо сжатие, либо хранение вне основной таблицы, кроме того он исключает использование однобайтовых varlena-заголовков (единственная возможная стратегия для столбцов, для которых не поддерживается TOAST);
– EXTENDED позволяет как сжатие и хранение вне основной таблицы. Это – стратегия, используемая по умолчанию для

большинства TOAST-типов данных. Сначала будет предпринята попытка сжатия, а затем – запись в TOAST-таблицу, если строка по-прежнему слишком велика;

- EXTERNAL позволяет хранение вне основной таблицы, но не сжатие. Использование этой стратегии позволяет ускорить подстроковые операции в текстовых и байтовых полях быстрее (но за счет увеличения пространства для хранения), поскольку эти операции оптимизированы для извлечения необходимых частей данных, когда они не сжаты;
- MAIN позволяет сжатие, но не хранение вне основной таблицы. На самом деле, хранение вне основной таблицы будет по-прежнему доступно, но только в качестве крайней меры, когда нет возможности уменьшить строку так, чтобы она поместилась на странице.

1.1.1.3 Формат страницы базы данных

Каждая таблица и индекс в PostgreSQL хранится как массив страниц фиксированного размера (обычно 8 Кб). В таблице, все страницы логически эквивалентны, так что та или иная строка может быть сохранена на любой странице. В индексах, первая страница, как правило, защищены в качестве метастраницы, содержащей управляющую информацию, и в индексе могут быть различные типы страниц, в зависимости от метода доступа к индексу.

Каждая страница состоит из пяти частей:

- PageHeaderData (24 байт) – содержит общую информацию о странице, в том числе указатель свободного пространства;
- ItemIdData – массив пар (смещение, длина) по 4 байта каждая, указывающих на фактические элементы данных, хранящихся на странице;
- Free space – свободное место. Новые указатели на элементы данных локализуются в начале этой области, новые элементы – с конца;
- Items – фактические данные;
- Special space – специфические данные с индексным методом доступа, различным для различных данных (эта область – пустая в обычных таблицах).

Первые 24 байта на каждой странице отводятся под заголовок страницы (PageHeaderData), состоящий из нескольких полей, в которых содержится информация о частях страницы, описанных выше.

После заголовка страницы следуют 4-байтные идентификаторы элементов данных (ItemIdData). Идентификатор элемента содержит байт-смещение начала элемента, его длину в байтах и несколько бит атрибутов, которые влияют на его интерпретацию. Новые идентификаторы выделяются по мере необходимости с начала свободного пространства. Поскольку

идентификатор элемента никогда не смещается пока он не будет освобожден, его индекс может быть использован на долгосрочной основе для ссылки на элемент, даже если сам предмет перемещается по странице с целью сделать свободное пространство более компактным. Фактически, каждый указатель на элемент данных (ItemPointer, также известный как CTID), созданный PostgreSQL, состоит из номера страницы и индекса идентификатора элемента.

Сами элементы данных хранятся в пространстве, выделяемом в обратном порядке от конца свободного пространства. Точная структура хранения меняется в зависимости от того, что таблица будет содержать. Все строки таблиц и курсоров используют структуру под названием HeapTupleHeaderData, в которой выделяется заголовок фиксированного размера (занимает 23 байтов на большинстве машин), в котором содержатся физические параметры хранения строк, далее следуют необязательные битовый массив NULL-значений, ID объекта и пользовательские данные (т.е. собственно записи). Информация о наличии битового массива NULL-значений и ID объекта также указывается в заголовке.

Если битовый массив NULL-значений присутствует, он начинается сразу после фиксированного заголовка и занимает столько байт, сколько необходимо из расчета один бит на столбец данных. В этом массиве бит 1 указывает на не NULL-значение, бит 0 – на NULL-значение. Если битовый массив NULL-значений отсутствует, то предполагается, что ни один из столбцов не имеет NULL-значения.

Если ID объекта присутствует, то он размещается между битовым массивом NULL-значений и началом фактических данных с учетом того, что последние выравниваются по границе, определяемой параметром MAXALIGN для данной платформы, т.е. значение, соответствующее ID объекта может быть дополнено нулевыми битами (padding value).

Чтобы прочитать данные, соответствующие каждому атрибуту по очереди, сначала проверяется, имеет ли поле NULL-значение в соответствии с битовым массивом NULL-значений. Если это так, переходим к следующему полю. Если поле фиксированной длины, то все байты просто считываются с учетом выравнивания. Если поле переменной длины, то оно имеет заголовок varlena, который включает в себя общую длину сохраненного в этом поле значения, и некоторые флаговые биты. В зависимости от флагов, этот элемент данных может быть размещен в основной таблице либо в TOAST-таблице, он также может быть сжат.

1.1.2 Логическая архитектура баз данных PostgreSQL

К особенностям логической архитектуры PostgreSQL, кроме упоминавшихся выше табличных пространств, можно отнести следующие моменты: схемы, индексы, роли и привилегии, правила, хранимые процедуры и триггеры, функции, типы данных.

Схемы

PostgreQL поддерживает схемы. Схемы являются как бы дополнительными областями видимости внутри БД. Также схему можно сравнить и с дополнительным путем (название схемы должно указываться перед названием таблицы) и с каталогом, внутри которого можно разместить таблицы. В любой БД по умолчанию существует схема public, в которой создаются все таблицы и которую не нужно указывать специально. Администратор БД может создавать другие схемы (и разграничивать доступ к ним), что обеспечивает еще один уровень распределения прав доступа для пользователей, позволяет выделить каждому пользователю как бы персональный раздел внутри БД с теми же названиями таблиц, что и у других пользователей.

Индексы

В PostgreSQL существует 4 типа индексов: B-tree, Hash, GiST (Generalized Search Tree) и GIN (Generalized Inverted Index). Каждый тип индекса имеет свой алгоритм реализации, что позволяет существенно увеличить быстродействие, если для определенного вида данных выбирать определенный тип индекса.

PostgreSQL позволяет также создавать индексы с использованием выражений и частичные (partial) индексы (с использованием служебного слова WHERE).

Роли и привилегии

PostgreSQL управляет привилегиями в БД, используя концепцию ролей. Ролью может быть как отдельный пользователь БД, так и группа пользователей. Роли могут быть владельцами объектов в БД (например таблиц), а также могут назначать привилегии доступа к этим объектам для других ролей. Возможно предоставить одной роли членство в другой роли и соответственно передать этой роли права той роли, членом которой она будет. Концепция ролей заменила старую концепцию пользователей и групп, предоставив ту же функциональность. Начиная с версиии 9.0 в PostgreSQL поддерживаются права на схемы и права по умолчанию.

Правила

Механизм правил (rules) представляет собой механизм создания пользовательских обработчиков не только операций манипулирования, но и операции выборки данных. Основное отличие от механизма триггеров заключается в том, что правила срабатывают на этапе разбора запроса, до выбора оптимального плана выполнения и самого процесса выполнения. Правила позволяют переопределять поведение системы при выполнении SQL-операций к таблице. Например, при создании представления (view) создается правило, которое определяет, что вместо выполнения операции выборки к представлению система должна выполнять операцию выборки к базовой таблице/таблицам с учетом условий выборки, которые лежат в основе определения представления. Для создания представлений, которые поддерживают операции обновления, правила для операций вставки, изменения и удаления строк, должны быть определены пользователем.

Система правил (более правильно говорить: система правил изменения запросов) позволяет изменять запрос согласно заданным правилам и потом передает измененный запрос планировщику запросов для планирования и выполнения. Система правил является очень мощным инструментом и может быть использована во многих случаях, таких как хранимые процедуры и представления.

Хранимые процедуры и триггеры

Хранимые процедуры в PostgreSQL могут быть написаны на любом из поддерживаемых встроенных языков. Хранимые процедуры могут быть использованы в триггерах и могут возвращать любой из поддерживаемых типов данных, а также массивы и списки. Начиная с версии 9.0, вызывать хранимые процедуры можно с указанием именуемых параметров.

Триггеры определяются как функции, которые инициируются операциями манипулирования. Триггеры могут быть назначены до или после операций INSERT, UPDATE или DELETE. Если произошло событие, на которое был назначен триггер, то вызывается закрепленная за этим триггером процедура. Например, операция INSERT может запускать триггер, проверяющий прибавленную запись на соответствии определенным условиям. При написании функций для триггеров могут использоваться разные языки программирования. Триггеры ассоциируются с таблицами и выполняются в алфавитном порядке.

В версии 9.0 введены триггеры на столбцы и, кроме того, при объявлении триггера можно использовать ключевое слово WHEN, которое добавляет дополнительное условие для срабатывания триггера.

Функции

Функции являются блоками кода, выполняемыми на сервере, а не на стороне клиента БД. Иногда функции отождествляются с хранимыми процедурами, однако между этими понятиями есть разница. Хотя они могут создаваться на чистом SQL, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки собственно SQL и требует использования некоторых языковых расширений. Функции могут писаться на одном из таких языков:

- встроенный процедурный язык PL/pgSQL, во многом аналогичный языку PL/SQL, что используется в СУБД Oracle;
- скриптовые языки – PL/Lua, PL/LOLCODE, PL/Perl, PL/PHP, PL/Python, PL/Ruby, PL/sh, PL/Tcl и PL/Scheme ;
- классические языки – C, C + +, Java (через модуль PL/Java);
- статистический язык R (через модуль PL/R).

PostgreSQL допускает использование функций, которые возвращают набор записей, который дальше можно использовать так же, как и результат выполнения обычного запроса (курсор). Функции могут выполняться как с правами их создателя, так и с правами текущего пользователя.

Начиная с 9.0, можно создавать функции без объявления имени (анонимные блоки) для выполнения блока операторов на любом встроенном

языке, который поддерживает PostgreSQL, прямо в командной строке.

Типы данных

PostgreSQL поддерживает большой набор встроенных типов данных:

- численные типы: целые, с фиксированной точкой, с плавающей точкой, денежный (отличается специальным форматом вывода, а в остальном аналогичен числам с фиксированной точкой и двумя знаками после запятой);
- символьные типы произвольной длины;
- двоичные типы (включая большой двоичный объект BLOB – Binary Large Object);
- типы «дата/время» (полностью поддерживают разные форматы, точность, форматы вывода, включая последние изменения в часовых поясах);
- логический тип;
- перечислимый тип;
- геометрические примитивы;
- сетевые типы: IP і IPv6 –адреса; CIDR –формат (Classless Inter-Domain Routing) – бесклассовая адресация – метод IP-адресации, позволяющий гибко управлять пространством IP-адресов, не используя жёстких рамок IP-адресации на основе классов сетей; MAC-адреса – уникальный идентификатор, присваиваемый каждой единице сетевого оборудован;
- UUID тип (Universally Unique Identifier) — это стандарт идентификации, используемый при создании программного обеспечения. Наиболее распространённым использованием данного стандарта является Globally Unique Identifier (GUID) фирмы Microsoft;
- XML тип (Xtensible Markup Language) – текстовый формат, предназначенный для хранения структурированных данных (взамен существующих файлов баз данных), для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки;
- массивы;
- OID –типы (Object identifiers) представляют идентификаторы различных объектов и используются обычно в PostgreSQL как первичные ключи для различных системных таблиц. Эти типы представляются как 4-байтовое целые числа без знака, т.е. имеют достаточно ограниченный диапазон значений, поэтому не могут использоваться в больших базах данных;
- композитные типы (composite type – составной тип) представляет структуру ряда или записи, т.е. по существу список имен полей и их типов данных;
- псевдотипы (pseudo-types) не могут использоваться в качестве типа данных столбца таблицы или представления, но могутт

использоваться для объявления аргументов функции или типа результата.

С любым объектом данных, представленным в PostgreSQL, связывается определенный тип, даже если на первый взгляд это и не очевидно. Тип данных одновременно определяет и ограничивает разновидности операций, которые могут выполняться с этими данными.

Хотя большинство типов данных PostgreSQL взято непосредственно из стандартов SQL, существуют и другие, нестандартные типы данных (например, геометрические и сетевые типы). В таблице 1.1 перечислены основные базовые типы данных PostgreSQL, а также их синонимы (альтернативные имена).

Таблица 1.1 – Типы данных PostgreSQL

Тип данных	Описание	Стандарт
Логические и двоичные типы данных		
boolean, bool	Отдельная логическая величина (true или false)	SQL99
bit(n)	Битовая последовательность фиксированной длины (ровно nбит)	SQL92
bit varying(n),varbit(n)	Битовая последовательность переменной длины (до n бит)	SQL92
Символьные типы		
character(n), char(n)	Символьная строка фиксированной длины (ровно n символов)	SQL89
character varying(n), varchar(n)	Символьная строка переменной длины (до n символов)	SQL92
text	Символьная строка переменной или неограниченной длины	PostgreSQL
Числовые типы		
small int, int2	2-байтовое целое со знаком	SQL89
integer, int, int4	4-байтовое целое со знаком	SQL92
bigint, int8	8-байтовое целое со знаком, до 18 цифр	PostgreSQL
real, float4	4-байтовое вещественное число	SQL89
double precision, floats, float	8-байтовое вещественное число	SQL89
numeric(p.s), decimal (p.s)	Число из p цифр, содержащее 5 цифр в дробной части	SQL99
money	Фиксированная точность, представление денежных величин	PostgreSQL, считается устаревшим
serial	4-байтовое целое с автоматическим приращением	PostgreSQL
Время и дата		
date	Календарная дата (день, месяц и год)	SQL92
time	Время суток	SQL92
time with time zone	Время суток с информацией о часовом поясе	SQL92
timestamp	Дата и время	SQL92
interval	Произвольный интервал времени	SQL92
Геометрические типы		
box	Прямоугольник на плоскости	PostgreSQL
line	Бесконечная линия на плоскости	PostgreSQL
Iseg	Отрезок на плоскости	PostgreSQL

Продолжение таблицы 1.1

circle	Круг с заданным центром и радиусом	PostgreSQL
path	Замкнутая или разомкнутая геометрическая фигура на плоскости	PostgreSQL
point	Точка на плоскости	PostgreSQL
polygon	Замкнутый многоугольник на плоскости	PostgreSQL
Сетевые типы		
cidr	Спецификация сети IP	PostgreSQL
inet	Сетевой IP-адрес с необязательными битами подсети	PostgreSQL
macaddr	MAC-адрес (например, аппаратный адрес адаптера Ethernet)	PostgreSQL
Системные типы		
oid	Идентификатор объекта (записи)	PostgreSQL
xid	Идентификатор транзакции	PostgreSQL

Помимо встроенных типов данных, пользователь может самостоятельно создавать новые необходимы ему типы и программировать для них механизмы индексирования с помощью методов GiST.

Пользовательские объекты

PostgreSQL может быть расширен пользователем для собственных потребностей практически в любом аспекте. Есть возможность добавлять:

- типы данных и их преобразования;
- домены;
- функции (включая агрегатные);
- индексы;
- операторы (включая переопределение уже существующих);
- процедурные языки.

В базе данных PostgreSQL организованы в нескольких разных объектах (рисунок 1.1):

- домены (domains);
- конфигурация полнотекстового поиска (FTS configuration);
- словари полнотекстового поиска (FTS dictionaries);
- синтаксические анализаоры полнотекстового поиска (FTS parsers);
- шаблоны полнотекстового поиска (FTS templates);
- функции (functions);
- последовательности (sequences);
- таблицы (tables);
- триггеры (trigger functions);
- представления (views).

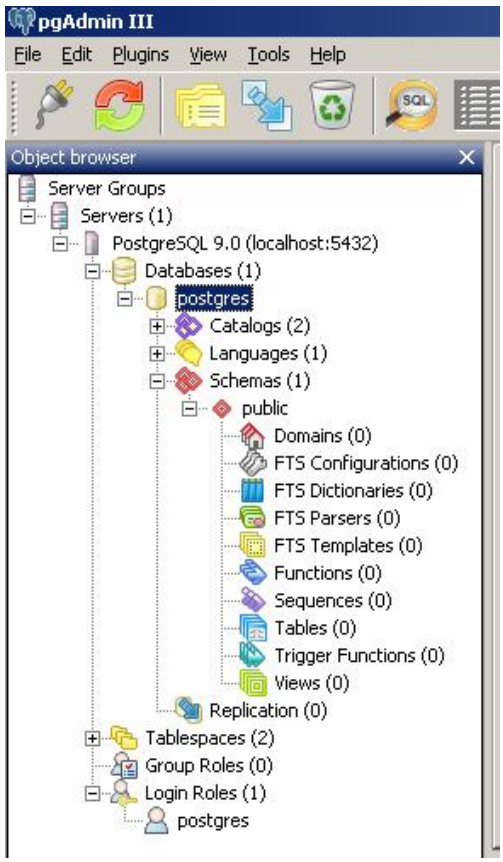

Рисунок 1.1 – Основные компоненти базы данных PostgreSQL 9.0

1.1.3 Разработка логической модели базы данных

1.1.3.1 Средства для разработки и администрирования баз данных

Для работы с базами данных существует несколько возможностей:

– запуск интерактивной терминальной программы, которая позволяет вводить, редактировать и выполнять команды SQL:

СУБД	Интерфейс командной строки
MS SQL Server	isqlw
PostgreSQL	psql

– использование пакета с графическим интерфейсом(GUI):

СУБД	GUI
MS SQL Server	SQL Server Enterprise Manager
PostgreSQL	pgAdmin

– написание специального приложения, используя один из нескольких доступных языков программирования, которые поддерживаются СУБД.

Далее будет рассмотрена работа в среде PostgreSQL с использованием pgAdmin.

1.1.3.2 Пример создания базы данных

Шаг 1. Создание базы данных BookShop

Параметры новой БД показаны на рисунке 1.2.

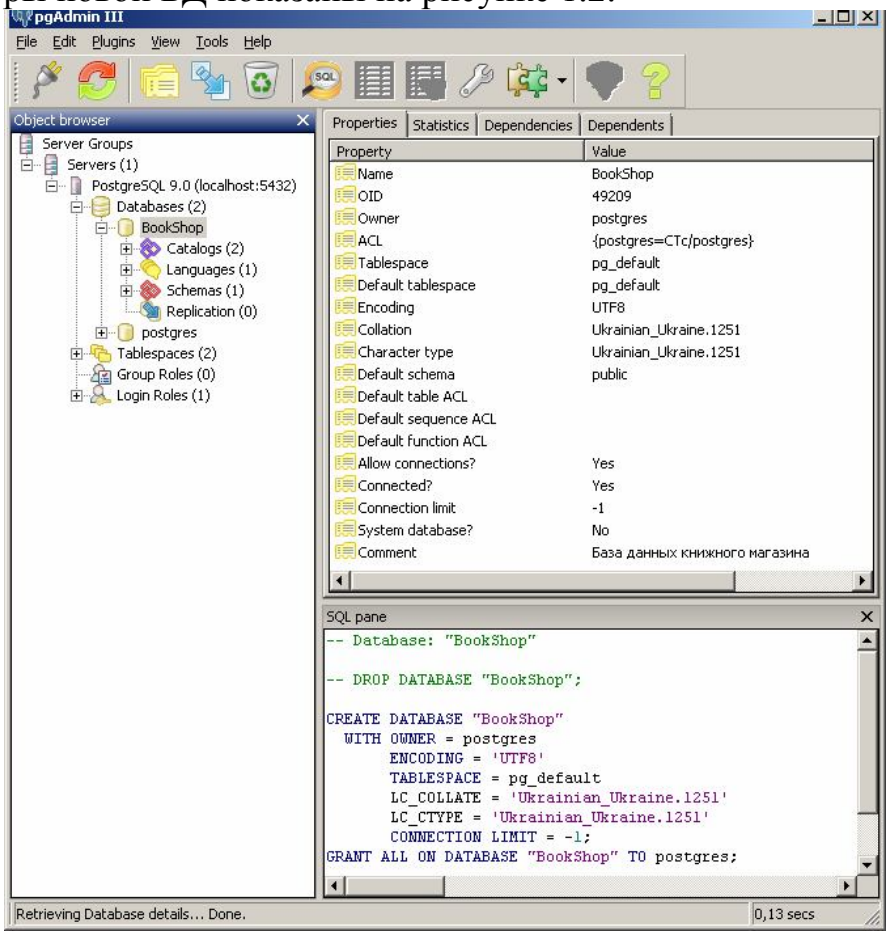

Рисунок 1.2 – Параметры БД BookShop

При создании новой базы данных в системную таблицу pg_database (см. pg_catalog/Tables) добавляется строка, содержащая параметры этой базы данных.

Шаг 2. Создание таблиц базы данных

После того, как БД создана, можно приступать к созданию ее основных объектов – таблиц. Прежде всего, для каждого столбца любой таблицы необходимо указать определенный тип данных. В SQL тип данных задается после имени столбца с помощью соответствующего ключевого слова, потом вводятся параметры представления значений столбцов, такие как длина, значение по умолчанию и др. После определения тип данных столбца таблицы сохраняется в виде постоянной характеристики столбца и не может быть изменен.

В отношении типов данных следует помнить, что недопустимо называть объекты именами команд или использовать для этой цели другие зарезервированные слова. Типы данных – это полноценные объекты БД, которые хранятся в системной таблице pg_type вместе с их OID.

Создаем таблицы с полями, указанными в таблице 1.2.

Таблица 1.2 – Описание таблиц БД BookShop

Таблица	**Поле**	**Тип**
Книги	Код_книги	serial
	Автор	character varying(80)
	Название	character varying(160)

	Издательство	character varying(80)
	Цена	money
	Остаток	smallint
Поставщики	Код_поставщика	serial
	Название	character varying(40)
	Город	character varying(40)
	Адрес	character varying(80)
	Телефон	character(13)
Заказы	Код_заказа	serial
	Код_книги	integer
	Код_заказчика	integer
	Оплачен	character varying(3)
	Дата	date
Заказчики	Код_заказчика	serial
	Имя	character varying(40)
	Адрес	character varying(80)
	Телефон	character(13)
Поставки	Номер	serial
	Код_книги	integer
	Код_поставщика	Integer
	Количество	integer
	Дата	date

Шаг 3. Введение ограничений целостности

Следующий момент в процессе создания таблиц, которому необходимо уделить особенное внимание, связан с обеспечением целостности данных. Для обеспечения целостности данных в таблицах определяются ограничения на значения столбцов (constraints). Эти ограничения могут быть введены при создании таблицы для каждого столбца в отдельности или добавлены в таблицу позже с помощью специальной команды SQL ALTER TABLE. В PostgreSQL поддерживаются следующие основные ограничения целостности:

- PRIMARY KEY – первичный ключ;
- FOREIGN KEY/REFERENCES – внешний ключ (ссылка);
- UNIQUE – уникальность;
- CHECK – проверка условия на значение.

Ограничение первичного ключа на значение столбца используется для обеспечения уникальности данных в столбцах и в целом для обеспечения ссылочной целостности (при связывании таблиц посредством внешних ключей). Определение условия primary key для таблицы имеет несколько эффектов. Во-первых, оно устанавливает определенные условия на значение первичного ключа – запрещается введения одинаковых значений и значений NULL в те столбцы, для которых оно определено. Во-вторых, primary key создает уникальный индекс для этих столбцов, что позволяет ускорить поиск строк в таблице.

Определение условия primary key в одной таблице само по себе не обеспечивает целостность по ссылкам. Необходимо также определить соответствующие внешние ключи тех таблиц, строки которых будут комбинироваться со строками той таблицы, где определено ограничение на

значение столбца PRIMARY KEY.

Ограничение внешнего ключа на значение столбца обычно применяется вместе с предварительно определенным ограничением primary key (на самом деле достаточно ограничения UNIQUE) в ассоциируемой таблице. Условие на значение foreign key ставит в соответствие один или несколько столбцов таблицы идентичному набору столбцов другой таблицы, для которых определенно ограничение primary key (или UNIQUE). Когда обновляются или удаляются значения тех столбцов таблицы, на которые ссылаются внешние ключи других таблиц, возникает вопрос: что делать с соответствующими значениями вешних ключей? Существует несколько вариантов решения этой проблемы:

- ничего не делать (no action) – это событие должно обрабатываться неким отличным от стандартного способом, иначе будет выдаваться сообщение об ошибке;
- запретить любые изменения (restrict);
- автоматически обновить/удалить значения соответствующих внешних ключей (cascade);
- установить для внешних ключей NULL-значения (set null) (для этого соответствующие столбцы не должны иметь ограничения NOT NULL);
- установить для внешних ключей значения по умолчанию (set default).

Автоматическое обновление соответствующих столбцов в разных таблицах после того, как для них определены ограничения на значение столбцов primary key и foreign key, называется декларативной ссылочной целостностью (declarative referential integrity).

Ограничение на значение столбцов primary key и foreign key обеспечивают соответствие строк связанных таблиц, потому столбцы с такими ограничениями используются для реализации операции соединения таблиц.

Наглядное представление о структуре связей между таблицами в базе данных можно получить с помощью диаграмм «таблица-связь», на которых указываются ограничения primary key и foreign key и такая характеристика связей, как степень связи. На рисунке 1.3 показана диаграмма базы данных BookShop.

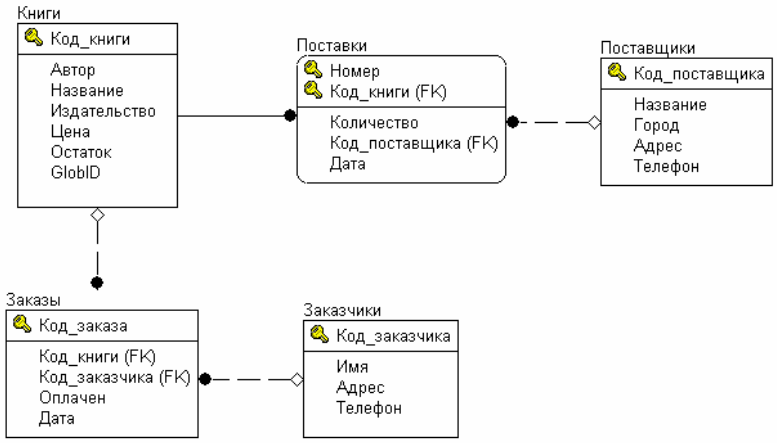

Рисунок 1.3 – Диаграмма «таблица-связь» базы данных BookShop

На приведенном рисунке сплошная линия обозначает идентифицирующие связи, пунктирные линии – не идентифицирующие связи. Все связи имеют степень 1:N (N – со стороны зависимой таблицы).

Ограничение уникальности на значение столбца можно назначить для того, чтобы запретить повторение значений в любом столбце таблицы. Для столбца, входящего в состав первичного ключа, подобное ограничение не может быть определено, т.к. ограничение уникальности реализуется с помощью автоматического создания уникального индекса для столбца таблицы, а для первичного ключа также автоматически создается уникальный индекс. Однако, столбец (или столбцы, если ограничение UNIQUE определено сразу для нескольких столбцов) с ограничением UNIQUE может быть использован для связи с другими таблицами, т.е. на него могут ссылаться внешние ключи этих таблиц.

Проверочное ограничение на значение столбца устанавливает диапазон значений, которые могут быть введены в один или несколько столбцов таблицы базы данных. Ограничение check может использоваться, например, для того, чтобы установить диапазон значений, которые допускается хранить в столбце, определенном для данных числовых типов.

Процесс установки проверки значения для столбца таблицы называется связыванием (binding). Можно определить и ввести несколько проверок в один столбец. Проверка может быть определена для столбца даже в том случае, если для него уже существует какое-либо правило.

Хотя проверочные ограничения на значение работают быстрее и устанавливаются проще, но правила гибче. После определения правила оно может быть связано со столбцами нескольких таблиц, но для одного столбца можно задать лишь одно правило и несколько проверочных условий.

В таблице 1.3 приведены ограничения целостности для БД BookShop.

Таблица 1.3 – Ограничения целостности БД BookShop

Таблица	**Поле**	**Ограничение**
Поставщики	Код_поставщика	Primery key
	Название	Unique, NOT NULL
	Телефон	Defaults (“000 111-11-11”)

Заказы	Код_заказа	Primery key
	Код_книги	Foreign key (Код_книги из Книги, NULL-значения не допускаются, автомат. обновление при изменении Код_книги из Книги, удаления из Книги не допускаются).
	Код_заказчика	Foreign key (Код_заказчика из Заказчики, NULL-значения не допускаются, автомат. обновление при изменении Код_заказчика из Заказчики, удаления из Заказчики не допускаются)
	Оплачен	Check (“Так”, “Hi”), NOT NULL
	Дата	Defaults (timenow()), NOT NULL
Заказчики	Код_заказчика	Primery key
	Телефон	Defaults (“000 111-11-11”)
Книги	Код_книги	Primery key
Поставки	Номер	Primery key
	Код_книги	Foreign key (Код_книги из Книги, NULL-значения не допускаються, обновления при изменении Код_книги из Книги не допускаются, удаления из Книги не допускаются).
	Код_поставщика	Foreign key (Код_поставщика из Поставщики, NULL-значения допускаются, автомат. обновления при изменении Код_поставщика из Поставщики, при удалении из Поставщики установить в NULL)

В заключение для каждой таблицы генерируются отчеты Data Dictionary Report (отчет по словарю данных или DDL-отчет). Для таблицы Поставки, например, он должен выглядеть так:

Table Data dictionary report - Поставки
Generated: 25.09.2011 19:49:30
Server: PostgreSQL 9.0 (localhost:5432)
Database: BookShop
Schema: public
Columns

Name	Data type	Not Null?	Primary key?	Default	Comment
Номер	integer	Yes	Yes	nextval('"Поставки_Номер_seq"'::regclass)	
Код_книги	integer	Yes	No		
Код_поставщика	integer	No	No		
Количество	integer	No	No		
Дата	date	No	No		

Constraints

Name	Type	Definition	Comment
pk_поставки	Primary key	("Номер")	
fk_поставки_1	Foreign key	("Код_книги") REFERENCES "Книги" ("Код_книги") MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT	
fk_поставки_2	Foreign key	("Код_поставщика") REFERENCES "Поставщики" ("Код_поставщика") MATCH SIMPLE ON UPDATE CASCADE ON DELETE SET NULL	

Шаг 4. Ввод данных в таблицы

Для просмотра и манипулирования данными в таблицах в pgAdmin предназначены соответствующие команды из меню Tools или кнопки на панели инструментов. При вводе данных в таблицы следует помнить об ограничениях, которые были определены для столбцов и таблиц БД. Не следует забывать, что:

- внешние ключи могут принимать только те значения, которые уже введены в поля тех таблиц, на которые эти внешние ключи ссылаются;
- в столбцы с типом serial (счетчик) нельзя самим вводить значения;
- если в столбец, для которого определено значение по умолчанию, не водится никакого значения, то автоматически будет введено значение по умолчанию.

Для того, чтобы ввести в таблицу новое значение, необходимо просто перейти к строке, отмеченной звездочкой (*) и начать вводить необходимые значения в соответствующие столбцы.

После того, как введены все новые строки или после изменения данных, необходимо зафиксировать этот факт в БД, для чего нужно выполнить команду Refresh, нажав одноименную кнопку на панели инструментов окна Edit Data.

Чтобы удалить какую-либо строку таблицы, достаточно выделить ее, щелкнув левой кнопкой мыши по ее номеру, и выполнить команду Delete контекстного меню. Но при этом следует помнить о целостности данных по ссылкам, если она применяется в БД.

Вводим несколько произвольных строк в таблицы БД BookShop с учетом определенных ограничений на значения столбцов.

1.2 Порядок выполнения работы

1. Ознакомиться с теоретическими сведениями к лабораторной работе.
2. Установить СУБД PostgreSQL.
3. Используя метод нормализации универсального отношения, разработать инфологическую модель базы данных по вариантам, приведенным в таблице 1.4, определить ограничения целостности, создать БД и ввести тестовые данные в каждую из созданных таблиц.

Таблица 1.4 – Варианты заданий

№	Название разрабатываемой БД и атрибуты универсального отношения
1	Торговля
	Код изделия, наименования, марка, производитель, номер приходной накладной, дата поступления на состав, количество единиц, закупочная цена, розничная цена, номер счета-фактуры, дата продажи, количество проданных единиц, сумма, тип платежа нал/безнал, банковские реквизиты покупателя для безналичного расчета.
2	Коммунальные платежи.
	Код услуги, наименование, единица измерения, тарифная зона, стоимость единицы, лицевой счет клиента, ФИО клиента, адрес, телефон, месяц/год, объем потребления

	услуги, сумма к оплате, сумма задолженности.
3	Услуги.
	Личный номер клиента, ФИО клиента, дата рождения, домашний адрес, телефон, дата и время приема заказа, тип выполняемой работы, тариф, личный номер мастера, ФІО мастера, номер ордера на выполнение заказа, дата и время начала работы, дата и время закрытия ордера, объем выполненной работы, стоимость.
4	Начисление зарплаты.
	Номер личного дела, ФИО сотрудника, домашний адрес, домашний телефон, рабочий телефон, дата приема, на работу, стаж работы по специальности, общий стаж работы, образование, квалификация, должность, ставка заработной платы, месяц/год, количество рабочих дней за месяц, фактически отработанных дней, премиальные, отпускные, удержания в процентах от начисления, аванс, сумма, к выдаче.
5	Поставки.
	Код продукции, наименование продукции, единица измерения, код производителя, месяц и год выпуска, код поставщика, юридическое наименование поставщика, юридический адрес поставщика, банковские реквизиты поставщика, телефон поставщика, номер договора на поставку, дата подписания договора, срок поставки, количество единиц, показатель качества, оптовая цена, условия поставки.
6	Билетная касса.
	Номер рейса, пункт отправления, пункт назначения, время отправления, время прибытия, категория билета, стоимость билета, тип транспортного средства, общее количество мест, количество мест данной категории, уникальный порядковый номер пассажира, ФІО пассажира, дата и время отправления по билету, дата и время продажи билета, дата и время бронирования билета.
7	Отдел кадров.
	Личный номер сотрудника, ФИО сотрудника, номер паспорта, дата рождения, домашний адрес, домашний телефон, образование, номер диплома, код специальности, наименования специальности, квалификация, код должности, наименования должности, должностной оклад, дата прохождения повышения квалификации, присвоена квалификация, свидетельство о повышении квалификации, дата поощрения, вид поощрения, дата наложенного взыскания, вид взыскания.
8	Отель.
	Номер паспорта клиента, ФИО, дата рождения, гражданство, номер комнаты, категория, стоимость проживания за сутки, дата и время поселения, дата и время выселения, дата бронирования, сумма к оплате, личный номер администратора, ФИО администратора.
9	Производство.
	Код продукции, наименование, код сырья, наименования сырья, норма затрат сырья для производства единицы продукции, процент потерь при изготовлении продукции, себестоимость продукции, оптовая цена, год выпуска, план выпуска, фактически произведено, процент бракованной продукции, валовой доход, чистая прибыль, номер экспортной партии, дата отправления на экспорт, наименование импортера, страна импортера, количество продукции в партии.
10	Банк.
	Код клиента, юридическое наименование клиента, юридический адрес клиента, ФИО директора, телефон директора, ФИО главного бухгалтера, телефон бухгалтера, номер счета клиента, остаток на счете, номер счета кредитора, банк кредитора, юридическое наименование кредитора, номер договора о кредитовании, дата выдачи кредита, срок пользования кредитом, сумма кредита, процентная ставка.

11	Музыкальный магазин.
	ФИО музыканта (псевдоним), название группы, музыкальный стиль, название композиции, год выхода композиции, продолжительность композиции, код композиции, название диска, дата выпуска диска, код диска, количество выпущенных экземпляров, цена диска, ФИО покупателя, код покупателя, дата покупки, количество купленных дисков, стоимость покупки.

4. Реализовать на уровне структуры БД средства обеспечения целостности данных: уникальность и обязательность ввода первичных ключей; поддержка целостности для внешних ключей (каскадное удаление, обновление и т.д.); значения атрибутов по умолчанию (Default Values) и обязательность ввода значений атрибутов (NULL\NOT NULL); ограничения на значения данных атрибутов вида: «интервал», «перечислимое значение» и «сравнение значений двух атрибутов одной таблицы».
5. Сгенерировать отчеты Data Dictionary Report для каждой созданной при выполнении контрольного задания таблицы.
6. Оформить отчет по результатам выполнения лабораторной работы.

1.3 Содержимое отчета

Отчет о выполнении лабораторной работы должен содержать:

- название и тему лабораторной работы;
- цель лабораторной работы;
- краткие теоретические сведения;
- ход выполнения работы;
- выводы.

Раздел «Ход выполнения работы» должен содержать инфологическую модель БД на языке таблица-связь, описание разработанных ограничений целостности, рисунки с отчетами Data Dictionary Report для каждой таблицы БД.

1.4 Контрольные вопросы

В данном разделе приведено лишь несколько примеров контрольных вопросов, остальные будут приблизительно такого же содержания.

1. Сформулируйте основные свойства реляционной таблицы.
2. Что такое первичный ключ?
3. Сформулируйте правила обеспечения целостности данных в реляционных СУБД.
4. Что такое инфологическая модель базы данных?
5. Что такое логическая модель базы данных?
6. Что такое глобальный уникальный идентификатор?
7. Что такое декларативная ссылочная целостность?

8. Что такое ограничения целостности? Какие ограничения целостности поддерживает PostgreSQL?
9. Для чего и как определяются ограничения на значение столбцов?
10. Что такое ограничение первичного ключа?
11. Что такое ограничение внешнего ключа?
12. Что представляет собой технология TOAST? В чем особенности ее реализации
13. Какова структура страниц файла данных PostgreSQL?
14. В чем заключается концепция табличных пространств, используемая в PostgreSQL?
15. Что такое схемы в PostgreSQL? Для чего они предназначены?