МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению лабораторных работ по дисциплине
«МЕТОДЫ ПОСТРОЕНИЯ ЗАЩИЩЕННЫХ БАЗ ДАННЫХ»
для студентов направления подготовки
10.04.01 Информационная безопасность

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

Ставрополь

СОДЕРЖАНИЕ

Введение

Целью проведения лабораторных занятий по дисциплине «Основы построения защищенных баз данных» является обучение студентов принципам обеспечения безопасности информации в автоматизированных информационных системах (АИС), основу которых составляют базы данных (БД), навыкам работы со встроенными в системы управления базами данных (СУБД) средствами защиты.

Задачи изучения дисциплины:

1. Приобретение системного подхода к проблеме защиты информации в СУБД;
2. Изучение моделей и механизмов защиты в СУБД;
3. Приобретение практических навыков организации защиты БД.

Лабораторные занятия являются связующим звеном между лабораторными работами и самостоятельной работой студентов. В процессе выполнения лабораторных работ экспериментально проверяются ключевые вопросы данного курса, приобретаются практические навыки проектирования и разработки баз данных средствами современных систем управления базами данных, проверяется уровень усвоения основных положений предмета. Лабораторные работы выполняются на персональных компьютерах с использованием системы управления баз данных PostgreSQL, SQL Server, MySQL.

Студент обязан до лабораторного занятия прочитать методические указания к лабораторной работе и попробовать выполнить ее самостоятельно. Защита состоит в выполнении лабораторного задания, ответе на вопросы по теме лабораторной работы и внесении некоторых изменений в разрабатываемую базу данных в присутствии преподавателя. Отчеты оформляются с помощью текстового редактора Word на бумаге формата А4 в соответствии с требованиями стандартов на оформление технической документации.

1. ИССЛЕДОВАНИЕ РИСКОВ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ В БАЗАХ ДАННЫХ

Цель работы: приобретение студентами навыков по исследованию существующих или проектируемых баз данных на предмет наличия рисков информационной безопасности в реализации или проекте.

Теоретическая часть

При проектировании информационных систем различного назначения для хранения больших и сверхбольших объемов информации проектировщики обычно делают выбор в пользу реляционной СУБД. Такова сложившаяся практика. На последующих стадиях проектирования и разработки обеспечение безопасности базы данных (ядра всей системы) обычно сводится к выделению классов пользователей, их информационных потребностей и привилегий (эти и еще несколько этапов входят в формирование политики безопасности), проектированию системы разграничения доступа.

Для назначения/отмены привилегий используется язык SQL, включающий операторы GRANT, REVOKE и т. п. Большинство современных реляционных СУБД поддерживает дискреционную (DAC) и мандатную (MAC) модели разграничения доступа, а также дополнительные средства обеспечения безопасности.

Под угрозой обычно понимают потенциально возможное событие, действие (воздействие), процесс или явление, которое может привести к нанесению ущерба чьим-либо интересам.

Угрозой информационной безопасности автоматизированной информационной системе (АИС) назовем возможность воздействия на информацию, обрабатываемую в системе, приводящего к искажению, уничтожению, копированию, блокированию доступа к информации, а также возможность воздействия на компоненты информационной системы, приводящего к утрате, уничтожению или сбою функционирования носителя информации или средства управления программно-аппаратным комплексом системы.

Источники угроз информации баз данных

Разработка системы информационной безопасности должна базироваться на определенном перечне потенциальных угроз безопасности и установлении возможных источников их возникновения. Проектирование конкретной системы безопасности для любого объекта, в том числе и для систем баз данных, предполагает выявление и научную классификацию перечня источников угроз безопасности.

Сформулируем перечень внешних и внутренних угроз информационной безопасности баз данных.

Внешними дестабилизирующими факторами, создающими угрозы безопасности функционированию систем баз данных и СУБД, являются:

- умышленные, деструктивные действия лиц с целью искажения, уничтожения или хищения программ, данных и документов системы, причиной которых являются нарушения информационной безопасности защищаемого объекта;
- искажения в каналах передачи информации, поступающей от внешних источников, циркулирующих в системе и передаваемой потребителям, а также недопустимые значения и изменения характеристик потоков информации из внешней среды и внутри системы;
- сбои и отказы в аппаратуре вычислительных средств;
- вирусы и иные деструктивные программные элементы, рас пространяемые с использованием систем телекоммуникаций, обеспечивающих связь с внешней средой или внутренние коммуникации распределенной системы баз данных;
- изменения состава и конфигурации комплекса взаимодействующей аппаратуры системы за пределы, проверенные при тестировании или сертификации системы.

Внутренними источниками угроз безопасности баз данных и СУБД являются:

- системные ошибки при постановке целей и задач проектирования автоматизированных информационных систем и их компонент, допущенные при формулировке требований к функциям и характеристикам средств обеспечения безопасности системы;
- ошибки при определении условий и параметров функционирования внешней среды, в которой предстоит использовать информационную систему и, в частности, программно-аппаратные средства защиты данных;
- ошибки проектирования при разработке и реализации алгоритмов обеспечения безопасности аппаратуры, программных средств и баз данных;
- ошибки и несанкционированные действия пользователей, административного и обслуживающего персонала в процессе эксплуатации системы;
- недостаточная эффективность используемых методов и средств обеспечения информационной безопасности в штатных или особых условиях эксплуатации системы.

Полное устранение всех потенциальных угроз информационной безопасности баз данных принципиально невозможно. Реальная задача состоит в снижении вероятности реализации потенциальных угроз до приемлемого для конкретной системы уровня. Приемлемость

соответствующего уровня угроз может определяться областью применения, выделенным бюджетом или положениями действующего законодательства. Как правило, не удается построить дерево угроз со строгой иерархией. Поэтому совокупный риск является достаточно сложной функцией уязвимости компонентов системы. Различные негативные воздействия также достаточно сложным образом влияют на основные характеристики качества и безопасности систем баз данных.

Основным руководящим принципом создания систем защиты является принцип равнопрочности. Следует распределять доступные ресурсы, обеспечивающие информационную безопасность системы, таким образом, чтобы минимизировать некоторый обобщенный показатель риска при любых негативных внешних и внутренних воздействиях на систему. Наличие в системе угроз, для защиты от которых в системе не предусмотрено каких-либо мер противодействия, приводит к тому, что все усилия, затраченные на возведение эффективных барьеров для иных способов деструктивного воздействия на систему, к ожидаемому результату не приведут. Отсюда следует важный практический вывод: учет угроз должен быть всесторонний и для каждой из возможных угроз должен быть реализован соответствующий угрозе метод защиты.

Угрозы конфиденциальности информации

К угрозам такого типа можно отнести:

1. *Инъекция SQL.* Во многих приложениях используется динамический SQL - формирование SQL-предложений кодом программы путем конкатенации строк и значений параметров. Зная структуру базы данных, злоумышленник может либо выполнить хранимую программу в запросе, либо закомментировать «легальные» фрагменты SQL-кода, внедрив, например, конструкцию UNION, запрос которой возвращает конфиденциальные данные. II последнее время даже появились специальные программы, автоматизирующие процесс реализации подобных угроз.
2. *Логический вывод на основе функциональных зависимостей.* Пусть дана схема отношения: $R(A_р...,A_п)$. Пусть и=$\{A_р..., A_п\}$, X, У - подмножества из U. Говорят, что X функционально определяет У, если в любом отношении г со схемой $R(A_р..., A_п)$ не могут содержаться два кортежа с одинаковыми значениями атрибутов из X и с различными из У. В этом случае имеет место функциональная зависимость, обозначаемая X—>Y. Пример функциональной зависимости для схемы отношения (фамилия, имя, отчество, должность, зарплата): если должность = менеджер, то зарплата = 1200. В реальных базах данных при наличии сведений о функциональных зависимостях злоумышленник может вывести конфиденциальную информацию при наличии доступа только к части отношений, составляющих декомпозированное отношение.
3. *Логический вывод на основе ограничений целостности.* Для кортежей отношений в реляционной модели данных (РМД) можно задать ограничения целостности - логические условия, которым должны удовлетворять атрибуты кортежей.
4. *Использование оператора UPDATE для получения конфиденциальной информации.* В некоторых стандартах SQL пользователь, не обладая привилегией на выполнение оператора SELECT, мог выполнить оператор UPDATE со сколь угодно сложным логическим условием. Так как после выполнения оператора UPDATE сообщается, сколько строк он обработал, фактически пользователь мог узнать, существуют ли данные, удовлетворяющие этому условию.

Специфичными для систем управления базами данных *угрозами доступности* являются:

1. **Использование свойств первичных и внешних ключей.** В первую очередь сюда относится свойство уникальности первичных ключей и наличие ссылочной целостности. В том случае, если используются натуральные, а не генерируемые системой значения первичных ключей, можно создать такую ситуацию, когда в таблицу невозможно будет вставить новые записи, так как там уже будут записи с такими же значениями первичных ключей. Если в базе данных поддерживается ссылочная целостность, можно организовать невозможность удаления родительских записей, умышленно создав подчиненные записи. Важной особенностью реализации ссылочной целостности является вопрос об индексировании внешнего ключа.

2. **Блокировка записей при изменении.** Заблокировав записи или всю таблицу, злоумышленник может на значительное время сделать ее недоступной для обновления.

3. **Загрузка системы бессмысленной работой.** Простейший пример - выполнение запроса, содержащего декартово произведение двух больших отношений. В реляционных СУБД возможны реализации и других классических угроз, например, атаки типа «троянский конь» - запуска пользователями программ, содержащих выполняющий определенные действия код, внедренный туда злоумышленником.

Практически все современные СУБД имеют встроенный процедурный язык программирования (PL/SQL, Transact SQL и т.п.) Программы на этом языке хранятся внутри базы данных и выполняются исполняющей подсистемой сервера баз данных, наличие и широкое использование производителями прикладного программного обеспечения механизма скрытия исходных текстов хранимых программ затрудняют его обнаружение. Так-же возможны многочисленные скрытые каналы, обусловленные семантикой данных и необходимостью обеспечения работы в условиях многопользовательского доступа.

На основании проведенного анализа можно сделать вывод, что в современных реализациях реляционных СУБД имеется значительное число уязвимостей, которые могут быть использованы для различных атак на информационные системы, построенные на их базе. Эта проблема отчасти может быть решена использованием специализированных программных средств анализа уязвимостей и защиты от угроз различных типов.

Но нельзя не заметить, что проблеме обеспечения информационной безопасности баз данных присуща определенная специфика. Источником дополнительных, специфических для баз данных, угроз являются:

- стандарт и реализация языка SQL - основного средства описания и обработки данных;
- обязательный интерфейс между СУБД и операционной системой, под управлением которой ведется обработка данных;
- протоколы сетевого взаимодействия на прикладном уров

не, которые обеспечиваются средствами СУБД.

Любая система обработки данных, допускающая много- поль-зовательский режим работы, предусматривает идентификацию и аутентификацию пользователей и процессов как обязательное предусловие предоставления доступа к ресурсам системы. Широко распространенной технологией аутентификации является использование паролей. Естественно ожидать, что технология подбора паролей также находится на достаточно высоком уровне развития. Можно выделить следующие *методы подбора паролей пользователей.*

1. **Тотальный перебор.** В этом случае злоумышленник последовательно опробует все

возможные варианты пароля. Для паролей длиннее шести символов во многих случаях данный метод может быть признан неэффективным.

2. **Тотальный перебор, оптимизированный по статистике встречаемости символов.** Разные символы встречаются в паролях пользователей с разной вероятностью. Например, вероятность того, что в пароле пользователя встретится буква «а», гораздо выше вероятности того, что в пароле присутствует символ «д». Согласно различным исследованиям, статистика встречаемости символов в алфавите паролей близка к статистике встречаемости символов в естественном языке.

При практическом применении данного метода злоумышленник вначале опробует пароли, состоящие из наиболее часто встречающихся символов, за счет чего время перебора существенно сокращается. Иногда при подборе паролей используется не только статистика встречаемости символов, но и статистика встречаемости диграмм и триграмм - комбинаций двух и трех последовательных символов соответственно.

Тотальный перебор, оптимизированный
с помощью словарей

В большинстве случаев пароли пользователей представляют собой слова английского или русского языка. Поскольку пользователю гораздо легче запомнить осмысленное слово, чем бессмысленную последовательность символов, пользователи предпочитают применять в качестве паролей осмысленные слова. При этом количество возможных вариантов пароля резко сокращается. Словарь С. И. Ожегова содержит около 60000 слов, объем словаря среднестатистического пользователя заметно меньше.

При использовании данного метода подбора паролей злоумышленник вначале опробует в качестве паролей все слова из словаря, содержащего наиболее вероятные пароли. В сети Интернет представлено множество подобных словарей, адаптированных для различных стран и групп пользователей. Если подбираемый пароль отсутствует в словаре, злоумышленник, как правило, может опробовать всевозможные комбинации слов из словаря, слова из словаря с добавленными к началу или к концу одной или несколькими буквами, цифрами и знаками препинания и т. д. «Хитрости» с написанием слов в обратном порядке в иной раскладке клавиатуры также не являются чем-то новым. Обычно данный метод используется в комбинации с предыдущим.

Человек склонен использовать пароли, которые легко запоминаются. Многие пользователи, чтобы не забыть пароль, выбирают в качестве пароля свое имя, фамилию, дату рождения, имена детей, любимых (в том числе принятые в узком кругу), номера телефонов и автомобилей и т. д. Хорошо известно, что некоторая категория мужчин склонна использовать в качестве (не отображаемого на дисплей) пароля ненормативную лексику, а определенная категория женщин часто использует уменьшительно-ласкательные имена домашних животных. В этом случае, если злоумышленник хорошо изучил пользователя, ему, как правило, достаточно провести меньше сотни опробований.

Базовые процедуры аутентификации на основе паролей в большинстве современных СУБД построены на сравнении вычисляемой значения хэш-функции от вводимого пароля с хранимым образом пароля, приписанного пользователю. Для вычисления образов паролей используются стандартные хэш-функции, поэтому их криптографические свойства не вызывают сомнения. Количество опробований (неуспешных попыток ввода) пароля устанавливается в профиле пользователя.

Наиболее слабым местом базовой процедуры аутентификации, видимо, является возможность доступа к образу пароля привилегированного пользователя. Если не используется процедура аутентификации средствами операционной системы, то образ пароля

хранится в справочнике данных. В этом случае привилегированный пользователь может не только считать образ для организации подбора пароля, но и незаметно для пользователя временно подменить пароль и выполнить некоторые действия от чужого имени.

Нецелевое расходование
вычислительных ресурсов сервера

Выполнение запросов и иные обращения к серверу баз данных связаны с расходованием его ресурсов, в первую очередь времени процессора, оперативной и внешней памяти. Это очевидное утверждение имеет одно неочевидное следствие. Если пользователю предоставлена возможность создавать процедуры и их выполнять, то может возникнуть проблема нарушения доступности в связи с нецелевым расходованием ресурсов.

Ясно, что запуск процедуры приведет к «замиранию всего живого» в системе. Эффект от подобной атаки может быть уменьшен назначением пользователю профиля, ограничивающего максимальное выделяемое время центрального процессора.

Напомним, что система контроля ограничений на ресурсы должна быть приведена в активное состояние.

Бороться с процедурами, расходующими ресурсы системы нецелевым образом, да и с иными деструктивными процедурами офицер безопасности может и принудительным завершением процессов-вредителей.

Использование SQL-инъекции для нештатного использования процедур и функций

Использование процедур и функций для параметрических запросов является широко распространенной практикой создания элементов информационных систем. Во многих случаях запрос к базе данных не зависит от параметров. Такой запрос называется статическим.

При использовании статического запроса программист создает PL/SQL программу, включая в код необходимые SQL-предложения. В зависимости от содержащихся в базе данных формируется некоторый результат, например, отчет о расходах или поступлениях товаров на склад на плановый период. В этом случае в тексте программы все SQL-предложения, которые необходимы в процессе ее работы, явно описываются, компилируются и могут храниться в вызываемых процедурах или функциях.

Но возможны и иные варианты. Например, значение параметра запроса (фамилия, адрес) зависит от ситуации. Для решения таких задач необходим механизм динамически формируемых SQL-предложений. Самое простое решение - предоставить пользователю оболочку для формирования запросов на языке SQL. Для реализации такого варианта пользователи должны владеть языком SQL. Это дорогое решение, к тому же создающее серьезные проблемы с безопасностью.

Чаще используется другой вариант: создаются программы на PL/SQL или ином языке с параметрами, которые отражают специфику запроса.

В комплект поставки сервера Oracle входит стандартный пакет DBMSSQL, обеспечивающий выполнение динамически формируемых SQL-предложений в программах на PL/SQL. Динамические SQL-предложения конструируются непосредственно во время выполнения программы в виде символьной строки, а затем передаются соответствующим программам пакета DBMS SQL для разбора и исполнения.

Ответственность за возможное нарушение объектных зависимостей и прав доступа, которые Oracle в данном случае во время компиляции не может проверить, теперь ложится на программиста.

Для динамически сформированных операторов SQL, которые возвращают данные (т. е. для операторов SELECT), используется следующая последовательность обработки: открытие

курсора, разбор запроса, связывание переменных, определение столбцов, выполнение запроса, выборка строк, получение значений столбцов, закрытие курсора.

Конечно, более изящно решение выполнения параметрического запроса с использованием динамического SQL. Но и в этом случае, несмотря на кажущуюся жесткость управления действиями пользователя, возможно деструктивное действие, известное как SQL-инъекция.

Оборудование и материалы

1. Компьютер под управлением операционной системы Windows 8/10 с параметрами по умолчанию.
2. Установленные на компьютер программы Microsoft Visio, PostgreSQL, MySQL, VMWare/VirtualBox.
3. Текстовый процессор Microsoft Word 2010/2013/2016.

Указания по технике безопасности

К выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж по технике безопасности.

Задания

1. Определить все возможные риски.
2. Определить вероятность возникновения риска и последствия.
3. Определить связанные риски.
4. Рассчитать влияние.
5. Выделить существенные риски.
6. Разработать план снижения рисков.

Для выполнения задач № 1 - 4 удобно использовать нижеследующую таблицу.

№	Идент и-	Источ ник	Услов ия	Вероят ность	Послед ствия	Влиян ие	Связан ные

Заполнение полей осуществляется по следующим рекомендациям:

Идентификатор - уникальное название, определяющее риск.

Источник - объект или субъект, порождающий или реализующий данный риск. Источником, может быть, как пользователь (проектировщик, программист, администратор и т. д.), так и программное обеспечение (ОС, СУБД, прикладное программное обеспечение и т. д.).

Условия возникновения - описание условий, при которых риск может быть реализован.

Вероятность - оценка вероятности возникновения конкретного риска при определенных условиях.

Последствия - оценка последствий реализации риска. Может быть выражена числом по 5-ти или 10-ти балльной шкале. Показывает относительную важность проблемы. Каждое значение шкалы должно быть четко определено и расписано.

Влияние - сводная количественная оценка риска для проекта, равная произведению его вероятности на количественную оценку его последствий.

Связанные риски — перечень рисков, связанных с данным.

Задача 5 - осуществляется при ранжировании рисков по влиянию. Для определения существенных рисков необходимо определить нижнюю границу, значение которое будет минимальным при определении существенности риска.

Задача 6 - Разработка плана снижения рисков, включает в себя определение мероприятий, которые должны снизить значение риска до нижней границы влияния, либо нивелировать риск совсем. Данный план должен включать в себя обоснования данных действий и мероприятий, а также расчеты, обосновывающие их применение.

Содержание 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 инфо- логическую модель БД на языке таблица-связь, описание разработанных ограничений целостности, рисунки с отчетами Data Dictionary Report для каждой таблицы БД.

Контрольные вопросы

1. Какие операторы использует язык SQL, для назначения отмены привилегий?
2. Что понимают под угрозой информационной безопасности автоматизированной информационной системе (АИС)?
3. Перечислите основные источники угроз информации баз данных.
4. Перечислите угрозы, специфичные для систем управления базами данных.
5. Перечислите методы подбора паролей пользователей.
6. Приведите пример использования SQL-инъекции для нештатного использования процедур и функций.

Рекомендуемая литература: 1-5.

2. РАЗГРАНИЧЕНИЕ ДОСТУПА К БАЗЕ ДАННЫХ

Цель работы: получение навыков по проектированию и реализации разграничения прав доступа к базе данных.

Теоретическая часть

В современных условиях любая деятельность сопряжена с оперированием большими объемами информации, которое производится широким кругом лиц. Защита данных от несанкционированного доступа является одной из приоритетных задач при проектировании любой информационной системы.

Для СУБД важны три основных аспекта информационной безопасности - конфиденциальность, целостность и доступность. Общая идея защиты базы данных состоит в следовании рекомендациям, сформулированным для класса безопасности С2 в «Критериях оценки надежных компьютерных систем».

Конфиденциальная информация (sensitive information) - информация, которая требует

защиты.

Доступ к информации (access to information) - ознакомление с информацией, ее обработка (в частности, копирование), модификация, уничтожение.

Субъект доступа (access subject) - лицо или процесс, действия которого регламентируются правилами разграничения доступа.

Объект доступа (access object) - единица информации автоматизированной системы, доступ к которой регламентируется правилами разграничения доступа. Объектами доступа (контроля) в СУБД является практически все, что содержит конечную информацию: таблицы (базовые или виртуальные), представления, а также более мелкие элементы данных: столбцы и строки таблиц и даже поля строк (значения). Таблицы базы данных и представления имеют владельца или создателя. Их объединяет еще и то, что все они для конечного пользователя представляются как таблицы, то есть как нечто именованное, содержащее информацию в виде множества строк (записей) одинаковой структуры. Строки таблиц разбиты на поля именованными столбцами.

Правила разграничения доступа (security policy) - совокупность правил, регламентирующих права субъектов доступа к объектам доступа.

Санкционированный доступ (authorizedaccess to information) - доступ к информации, который не нарушает правил разграничения доступа.

Несанкционированный доступ (unauthorized access to information) - доступ к информации, который нарушает правила разграничения доступа с использованием штатных средств, предоставляемых средствами вычислительной техники или автоматизированными системами.

Идентификатор доступа (access identifier) - уникальный признак объекта или субъекта доступа.

Идентификация (identification) - присвоение объектам и субъектам доступа идентификатора и (или) сравнение предъявляемого идентификатора с перечнем присвоенных идентификаторов.

Пароль (password) - идентификатор субъекта, который является его секретом.

Аутентификация (authentification) - проверка принадлежности субъекту доступа предъявленного им идентификатора, подтверждение подлинности.

В СУБД на этапе подключения к БД производится идентификация и проверка подлинности пользователей. В дальнейшем пользователь или процесс получает доступ к данным согласно его набору полномочий. В случае разрыва соединения пользователя с базой данных текущая транзакция откатывается, и при восстановлении соединения требуется повторная идентификация пользователя и проверка его полномочий.

Уровень полномочий субъекта доступа (subject privilege) - совокупность прав доступа субъекта доступа (для краткости в дальнейшем мы будем использовать термин «привилегия»).

Нарушитель правил разграничения доступа (security policy violator) - субъект доступа, который осуществляет несанкционированный доступ к информации.

Модель нарушителя правил разграничения доступа (security policy violator model) - абстрактное (формализованное или неформализованное) описание нарушителя правил разграничения доступа.

Целостность информации (information integrity) — способность средства вычислительной техники (в рассматриваемом случае - информационной системы в целом) обеспечить неизменность информации в условиях случайного и (или) преднамеренного искажения (разрушения).

Метка конфиденциальности (sensitivity label) — элемент информации, характеризующий конфиденциальность объекта.

Многоуровневая защита (multilevel secure) - защита, обеспечивающая разграничение доступа субъектов с различными правами доступа к объектам различных уровней конфиденциальности.

Дискреционная защита

В современных СУБД достаточно развиты средства дискреционной защиты.

Дискреционное управление доступам (discretionary access control) - разграничение доступа между поименованными субъектами и поименованными объектами. Субъект с определенным правом доступа может передать это право любому другому субъекту.

Однако дискреционная защита является довольно слабой, так как доступ ограничивается только к именованным объектам, а не собственно к хранящимся данным. В случае реализации информационной системы с использованием реляционной СУБД объектом будет, например, именованное отношение (то есть таблица), а субъектом - зарегистрированный пользователь. В этом случае нельзя в полном объеме ограничить доступ только к части информации, хранящейся в таблице. Частично проблему ограничения доступа к информации решают представления и использование хранимых процедур, которые реализуют тот или иной набор бизнес-действий.

Мандатная защита

Средства мандатной защиты предоставляются специальными (trusted) версиями СУБД.

Мандатное управление доступом (mandatory access control) - это разграничение доступа субъектов к объектам данных, основанное на характеризуемой меткой конфиденциальности информации, которая содержится в объектах, и на официальном разрешении (допуске) субъектов обращаться к информации такого уровня конфиденциальности.

Средства произвольного управления доступом характерны для уровня безопасности С. Как правило, их, в принципе, вполне достаточно для подавляющего большинства коммерческих приложений. Тем не менее они не решают одной весьма важной задачи - задачи слежения за передачей информации. Средства произвольного управления доступом не могут помешать авторизованному пользователю законным образом получить секретную информацию и затем сделать ее доступной для других, неавторизованных, пользователей. Нетрудно понять, почему это так. При произвольном управлении доступом привилегии существуют отдельно от данных (в случае реляционных СУБД - отдельно от строк реляционных таблиц), в результате чего данные оказываются «обезличенными» и ничто не мешает передать их кому угодно даже средствами самой СУБД; для этого нужно лишь получить доступ к таблице или представлению.

Использование СУБД с возможностями мандатной защиты позволяет разграничить доступ собственно к данным, хранящимся в информационной системе, от доступа к именованным объектам данных. Единицей защиты в этом случае будет являться, в частности, запись о договоре N, а не таблица или представление, содержащее информацию об этом договоре. Пользователь, который будет пытаться получить доступ к договору, уже никак не сможет обойти метку конфиденциальности. Существуют реализации, позволяющие разграничивать доступ вплоть до конкретного значения конкретного атрибута в конкретной строке конкретной таблицы. Дело не ограничивается одним значением метки конфиденциальности - обычно сама метка представляет собой набор значений, отражающих, например, уровень защищенности устройства, на котором хранится таблица, уровень защищенности самой таблицы, уровень защищенности атрибута и уровень защищенности конкретного кортежа.

Оборудование и материалы

1. Компьютер под управлением операционной системы Windows 8/10 с параметрами по умолчанию.
2. Установленные на компьютер программы Microsoft Visio, PostgreSQL, MySQL, VMWare/VirtualBox.
3. Текстовый процессор Microsoft Word 2010/2013/2016.

Указания по технике безопасности

К выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж по технике безопасности.

Задания

1. Реализовать пример разграничения доступа на практическом макете.
2. Для своего индивидуального задания по курсовой работе спроектировать и реализовать разграничение доступа в виде ролевой или дискреционной моделей.

Описание предметной области

Проектируемая база данных должна содержать в себе *информацию'.*

- контактная информация заказчика;
- контактная информация поставщика;
- дата и время доставки товара после получения заказа.

Основные пользователи базы данных:

- администратор базы данных;
- администраторы по заполнению информации;
- заказчики.

Функции пользователей базы данных:

- администратор базы данных - поддержание работоспособности базы данных;
- администраторы по заполнению информации - добавление, изменение, удаления данных, просмотр всей информации;
- заказчик - только просмотр данных. Перечень вводимой информации:
- имя;
- место проживания;
- номер телефона.

База данных должна поддерживать накопление и хранение информации об основных компонентах бизнеса и автоматизированное выполнение бизнес-процессов.

Бизнес-правила:

- Вносить изменения может только Администратор по заполнению информации.
- Иные зарегистрированные пользователи базы данных могут просматривать данные о заказах.

Для реализации базы данных необходимо спроектировать концептуальную модель и реализовать ее в выбранной СУБД. Процесс реализации уже готовой логической модели представлен на рисунках 2.1 - 2.4.

Рис. 2.1. Рабочая среда

	id	Name	Adress	Phone
▶	1	Jane	New-York, USA	7777777
	2	Michal	Washington, USA	4444444
	3	Roma	Tomsk, Russia	777
	4	Vasya	Moskov, Russia	134124
	5	Elena	Surgut city, Russia	89222222222
*	NULL	NULL	NULL	NULL

Рис. 2.2. Таблица

	id	Name	Adress	Phone
▶	1	Microsoft	USA	999
	2	Apple	USA	222
	3	Pelmeni	Tomsk	124123
*	NULL	NULL	NULL	NULL

Рис. 2.3. Таблица

	id	id_pok	id_pos	date	srok_days
▶	1	1	1	2001-02-22	128
	2	2	1	2011-12-02	33
	3	3	2	2003-02-22	342
*	NULL	NULL	NULL	NULL	NULL

Рис. 2.4. Таблица «Заказ»

Реализуем физическую модель данных (рисунок 2.5).

Рисю 2.5.
Диаграмма БД

Для создания разграничение доступа нам надо создать имена входа. На вкладке БД мы выбираем пункт «Безопасность» и в ней выбираем пункт «Имена входа» и создаем его (рисунок 2.6).

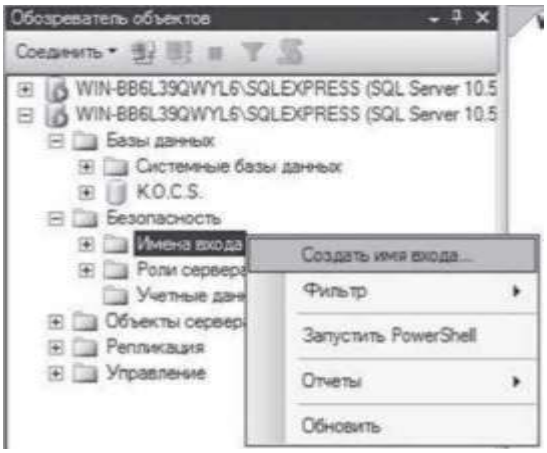


Рис. 2.6. Создание «имени входа»

При создании «имен входа» открывается интерфей представленный на рисунке 2.7.

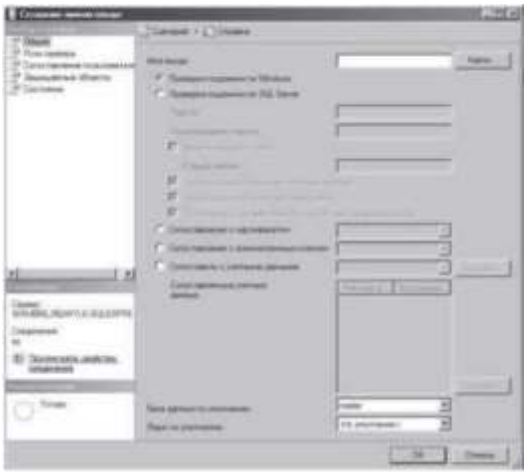
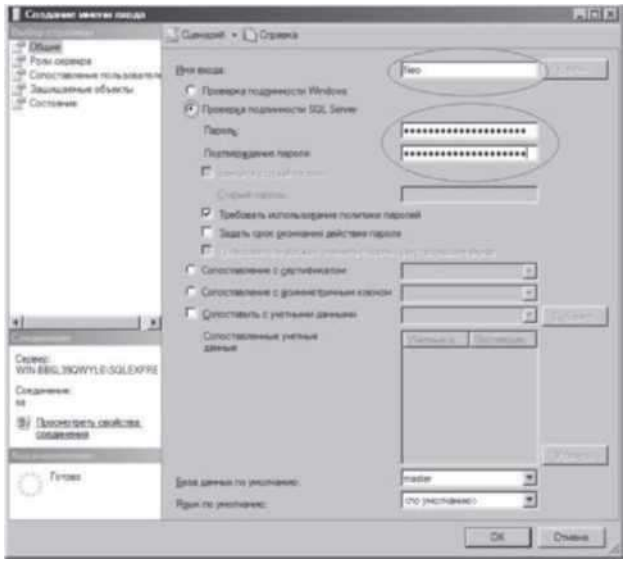


Рис. 2.7. Меню создания «имени входа»

Заменяем проверку подлинности на «Проверка подлинности SQL Server» и создаем имя



входа (рисунок 2.8).

Соединить ’ i ■ T 3

E , £ WIN-BBGL39QWYL6\SQLEXPRESS (SQL Server 10.5
B WIN-BB6L39QWYL6\SQLEXPRESS (SQL Server 10.5
B i_J Базы данных
(2 J Системные базы данньк
B (J K.O.C.S.
E J Диаграммы баз данных
B J Представления
E , 4 Синонимы
Б ГЗ Программирование
E 4 Компонент Service Broker fi j Хранилище
B Г-З Безопасность

Создать пользователя...

Фильтр
Политики
Запустить PowerShel
Обновить

Рис. 2.8. Создание имени

После создания имени входа, присваиваем его конкретному пользователю (рисунки 2.9 - 2.13).

Рис. 2.9. Создание Пользователя

В поле «Имя пользователя» записываем то имя, которое будет распознаваться в БД, и присваиваем Имя Входа (рисунок 2.10).

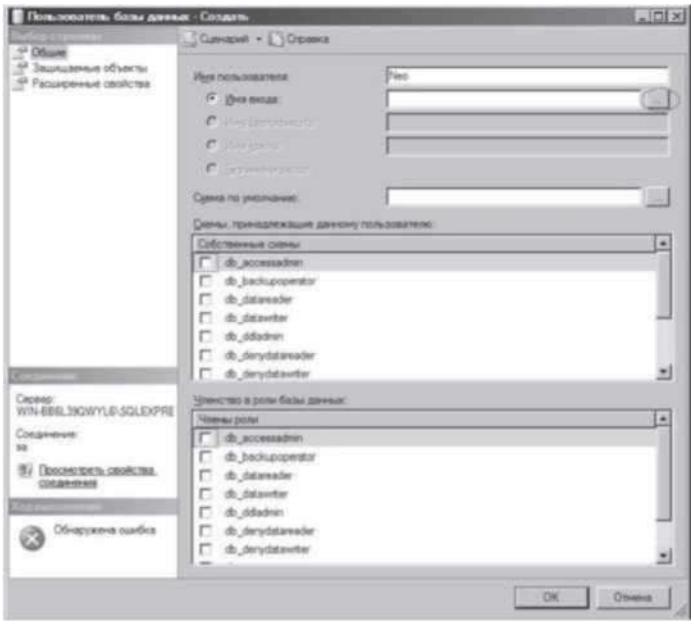


Рис. 2.10. Создание «Имени Пользователя»

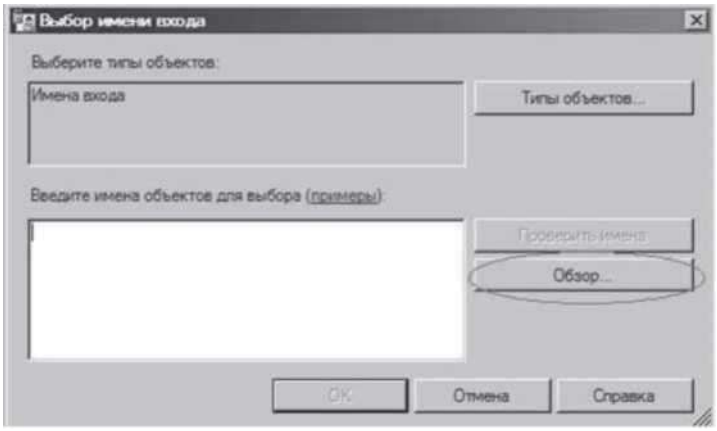


Рис. 2.11. Поис к «имени входа 1»

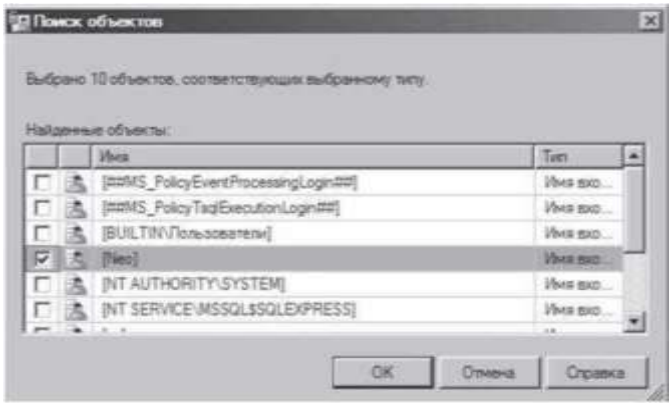


Рис. 2.12. Поиск «имени входа 2»

Задаем Схему базы данных и Роль для данного пользователя (рисунок 2.13).

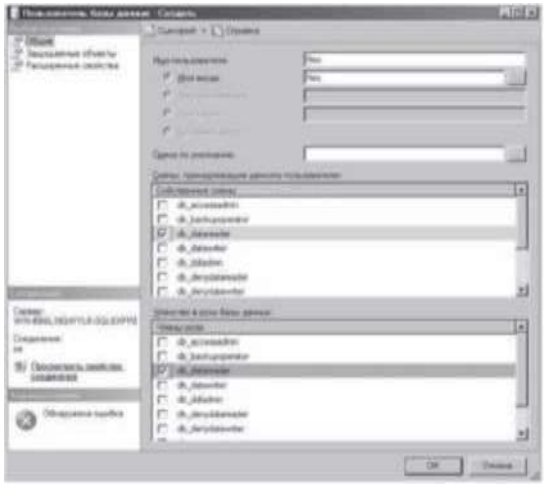


Рис. 2.13. Присвоения членства

Данные действия позволили пользователю присвоить роль «db_ datareader», позволяющей пользователю читать все данные представленные в БД без права внесения каких-либо изменений в нее.

Для подключения к БД под соответствующим Пользователем необходимо выбрать в меню «Файл» вкладку «Подключится к обозревателю объектов» (рисунок 2.14).

Среда Microsoft SQL Server Management Studio I

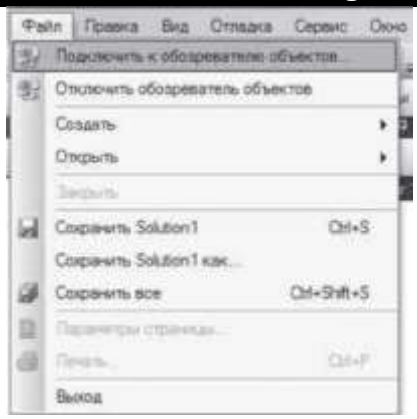


Рис. 2.14. Подключение к БД

И выбирать в пункте «Проверка подлинности» «Проверка подлинности SQL Server’а»

(рисунок 2.15).

Рис. 2.15. Подключение к БД

Для проверки прав доступа, которые были присвоены Новому пользователю БД можно проделать, следующие действия, например, добавить новую таблицу в нашу БД (рисунок 2.16).

Обозреватель объектов

Соединить - ■ Т 3 ез!

B j WIN-BBSL39QWYL6XSQLEXPRESS (SQL Server 1 C.5 B Q Базы данных

S □ Системные базы данньа B J K.O.C.S.

E 1-4 Хранилище
S 1 J Безопасность

Q Диаграммы баз данньа

S П Безопасность

S UJ Объекты сервера

S Г~3 Репликация

S □ Управление

Рис. 2.16. Добавление новой таблицы

В результате выполнения данного действия - ошибка (рисунок 2.17).

Текущий пользователь не является владельцем базы данных J или системным администратором. Пользователь не может сохранять изменения в таблицах, владельцем которых он не является

Некоторые изменения требуют разрешения CREATE TABLE

Рис. 2.17. Ошибка создания таблицы

При попытке изменения данных в таблице, результат - ошибка (рисунок 2.18).

I WIRBB6L33QWYL S - BЬo Заказ id id_pok id_pos date srok.days

И 1 __ 11___ 11 2001-02-22 128

2	2	1	2011-12-02	33
3	3	2	20034)2-22	342

J | 7 ***NULL NULL NULL NULL***

*NULL NULL NULL NULL NULL

Рис. 2.18. Ошибка изменения данных таблицы

Данные результаты демонстрируют правильность работы приложения по разграничению прав доступа.

Содержание 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 инфо- логическую модель БД на языке таблица-связь, описание разработанных ограничений целостности, рисунки с отчетами Data Dictionary Report для каждой таблицы БД.

Контрольные вопросы

1. Что такое относительная защитой информации в БД?
2. Дайте определение групп пользователей СУБД;
3. Что такое дискреционная защита?
4. Что такое данные о разграничениях доступа при дискреционной защите?
5. Что является объектом и субъектом защиты при настройке привилегий дискреционной защиты?
6. Что такое представление и какие ограничения по настройке безопасности имеются при работе средств SQL?
7. Что такое мандатная защита?
8. Почему при произвольном управлении доступом данные оказываются «обезличенными»?
9. Что такое логическая защита?
10. Что такое группы принадлежности?
11. Что включает в себя механизм меток безопасности;
12. Дайте определения каждого уровни доступа при реализации мандатной защиты?
13. От кого наследуют уровни пользователя при внесении данных?

Рекомендуемая литература: 1-5.

3. ИСПОЛЬЗОВАНИЕ И ЗАЩИТА ТРАНЗАКЦИЙ В БАЗАХ ДАННЫХ

Цель работы: изучить транзакцию и осуществить ее в MySQL.

Теоретическая часть

Транзакция - действие или ряд действий, выполняемых одним пользователем или прикладной программой, которые осуществляют чтение или изменение содержимого базы данных. Транзакция является логической единицей работы, выполняемой в базе данных. Она может быть представлена отдельной программой, частью программы или даже отдельной

командой (например, командой INSERT или UPDATE языка SQL) и включать произвольное количество операций, выполняемых в базе данных.

Фиксированная транзакция - транзакция выполнение, в которой выполнение всех действий успешно завершилось.

Любая транзакция завершается одним из двух возможных способов. В случае успешного завершения результаты транзакции фиксируются (commit) в базе данных, и последняя переходит в новое состояние. Если окажется, что зафиксированная транзакция была ошибочной, потребуется выполнить другую транзакцию, отменяющую действия, выполненные первой транзакцией.

Ни в одной СУБД не может быть предусмотрен априорный способ определения того, какие именно операции обновления могут быть сгруппированы для формирования единой логической транзакции. Поэтому должен применяться метод, позво
ляющий указывать границы каждой из транзакций извне, со стороны пользователя. В большинстве языков манипулирования данными для указания границ отдельных транзакций используются операторы BEGIN TRANSACTION, COMMIT и ROLLBACK.В случае аварийного завершения программы в базе данных автоматически будет выполнена команда ROLLBACK.

Источники отказов:

1) Системные ошибки. Система вошла в нежелательное состояние. Такое как взаимоблокировка, не позволяющая нормально производить выполнение программы. Приводит к повреждению данных.
2) Отказ оборудования. Ошибка диска и потеряспособно- сти передачи данных через соединительные линии.
3) Логические ошибки. Плохие данные или их отсутствие.

Существуют определенные свойства, которыми должна обладать любая из транзакций. Ниже представлены четыре основных свойства транзакций, которые принято обозначать аббревиатурой ACID (Atomicity, Consistency, Isolation, Durability - неразрывность, согласованность, изолированность, устойчивость), состоящей из первых букв названий этих свойств.

Неразрывность. Это свойство, для описания которого применимо выражение «все или ничего». Любая транзакция представляет собой неделимую единицу работы, которая может быть либо выполнена вся целиком, либо не выполнена вообще. За обеспечение неразрывности отвечает подсистема восстановления СУБД.

Согласованность. Каждая транзакция должна переводить базу данных из одного согласованного состояния в другое. Ответственность за обеспечение согласованности возлагается и на СУБД, и на разработчиков приложений. В СУБД согласованность может обеспечиваться путем выполнения всех ограничений, заданных в схеме базы данных, таких как ограничения целостности и ограничения предметной области. Но подобное условие является недостаточным для обеспечения согласованности. Например, предположим, что некоторая транзакция предназначена для перевода денежных средств с одного банковского счета на другой, но программист допустил ошибку в логике транзакции и предусмотрел снятие денег с правильного счета, а зачисление - на неправильный счет. В этом случае база данных переходит в несогласованное состояние, несмотря на наличие правильно заданных ограничений. Тем не менее ответственность за устранение такой несогласованности не может быть возложено на СУБД, поскольку в ней отсутствуют какие-либо средства обнаружения

подобных логических ошибок.

Изолированность. Все транзакции выполняются независимо друг от друга. Иными словами, промежуточные результаты незавершенной транзакции не должны быть доступны для других транзакций. За обеспечение изолированности отвечает подсистема управления параллельным выполнением.

Устойчивость. Результаты успешно завершенной (зафиксированной) транзакции должны храниться в базе данных постоянно и не должны быть утеряны в результате последующих сбоев. За обеспечение устойчивости отвечает подсистема восстановления.

Модели транзакций. Плоские транзакции - это основные строительные блоки для реализации принципа атомарности; иначе говоря, выделение некоторой последовательности действий в виде плоской транзакции обеспечивает принцип «все или ничего». Во многих прикладных окружениях, в особенности с централизованными обработкой и управлением ресурсами (например, базами данных и файлами), механизм плоских транзакций на протяжении многих лет предоставлял вполне удовлетворительные возможности как для создания, так и для выполнения приложений; простые преобразования состояний системы вполне укладывались в рамки атомарных единиц работы.

Модель вложенных транзакций. Транзакция рассматривается как коллекция взаимосвязанных подзадач или субтранзакций, каждая из которых также может состоять из любого количества субтранзакций. Модель вложенных транзакций была предложена Моссом (Moss) в 1981 году. В этой модели вся транзакция рассматривается как набор связанных подзадач, называемых субтранзакциями, каждая из которых также может состоять из произвольного количества субтранзакций. В данном определении полная транзакция представляет собой древовидную структуру или некоторую иерархию субтранзакций. В модели вложенных транзакций присутствует транзакция верхнего уровня, содержащая некоторое количество дочерних транзакций, каждая из которых, в свою очередь, может включать вложенные транзакции, и т.д. В исходном варианте, предложенном Моссом, выполнять операции в базе данных разрешается только транзакциям самого нижнего уровня (субтранзакциям самого нижнего уровня вложенности).

Хроника. Последовательность (плоских) транзакций, которая может передаваться с другими транзакциями СУБД гарантирует, что-либо все входящие в хронику транзакции будут успешно завершены, либо будут запущены компенсирующие транзакции, необходимые для устранения достигнутых частичных результатов. В отличие от метода вложенных транзакций, допускающего произвольный уровень вложения, метод хроник разрешает наличие единственного уровня вложения. Более того, для каждой выделенной субтранзакции должна существовать соответствующая компенсирующая транзакция, которая будет семантически правильно аннулировать результаты, достигаемые с помощью данной субтранзакции.

Модель вложенных транзакций требует, чтобы процесс фиксации субтранзакций выполнялся снизу-вверх, в направлении транзакции верхнего уровня. Поэтому данную модель принято называть моделью закрытых вложенных транзакций. Это позволяет подчеркнуть, что свойство неразрывности в транзакциях сохраняется до самого верхнего уровня. В противоположность этому, существует и модель открытых вложенных транзакций, в которой неразрывность нарушается и частичные результаты выполнения субтранзакций могут быть доступны вне транзакции.

Модели рабочих потоков. Все обсуждавшиеся в этом разделе модели были разработаны с целью преодоления ограничений, накладываемых моделью плоских транзакций, поэтому не

приемлемых для транзакций большой продолжительности. Однако было отмечено, что все эти модели по-прежнему не обладают мощностью, необходимой для удовлетворения потребностей деловых процессов определенных видов. Поэтому были предложены более сложные модели, представляющие собой комбинации моделей открытых и вложенных транзакций. Однако, поскольку эти модели не в полной мере отвечают требованиям ACID плоских транзакций, для них используется более подходящее название - модели рабочих потоков.

Рабочий поток представляет собой некоторый вид деятельности, предусматривающий координированное выполнение множества заданий, осуществляемых различными обрабатывающими субъектами, которые могут представлять собой людей или некоторые программные комплексы (например, СУБД, прикладные программы или службы электронной почты). Рассмотрим пример с подготовкой соглашений о сдаче объектов недвижимости в аренду, который можно найти в учебном проекте DreamHome. Клиент, желающий арендовать некоторый объект недвижимости, устанавливает контакт с соответствующим сотрудником компании, отвечающим за этот объект недвижимости. Этот сотрудник обращается к контролеру компании, в обязанности которого входит проверка кредитоспособности клиента и определение того, может ли компания ему доверять. С этой целью контролер использует соответствующие источники информации, например, бюро проверки кредитоспособности. Собрав интересующие его сведения, контролер принимает решение о возможности дальнейшей работы с данным клиентом и сообщает о своем решении сотруднику, который сообщает клиенту о принятом решении.

Классификация систем обработки транзакций

С появлением множества стандартизированных систем обработки транзакций нового поколения полезным представляется проведение их классификации с точки зрения спектра предоставляемых ими функций. Подобную классификацию, включающую пять измерений, предложили Абрахам Лефф и Калтон Пу из Колумбийского университета:

M - множество машин; P - множество процессов; H - степень неоднородности машин и программного обеспечения; D - множество логических данных; S - множество узлов.

Эта классификация характеризует любую систему обработки транзакций, от простейших (Pl, Ml, Hl, DI, S1) до более сложных многоузловых неоднородных окружений с поддержкой разнотипных наборов данных (Pn, Mп, Hn, Dn, Sn). В статье, написанной в 1991 г., эти авторы представили трехмерную классификацию, которую они применили для оценки различных исследовательских и коммерческих систем.

Оборудование и материалы

1. Компьютер под управлением операционной системы Windows 8/10 с параметрами по умолчанию.
2. Установленные на компьютер программы Microsoft Visio, PostgreSQL, MySQL, VMWare/VirtualBox.
3. Текстовый процессор Microsoft Word 2010/2013/2016.

Указания по технике безопасности

К выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж по технике безопасности.

Задания

1. Перечисление денежных средств с одного счета на другой.
2. Удаление внесенных данных. Из нескольких таблиц.
3. Перечисление зарплаты работникам из фирмы.
4. Заказ билетов на самолет.
5. Резервирование мест в гостинице.
6. Аренда автомобиля.
7. Автоматизировать систему прописки.
8. Написать транзакцию для получения наличных денег из автоматического кассового автомата.
9. Запись товара в магазин на склады, используя штрих-код продукта.
10. Подготовка соглашений о сдачи объектов в аренду.
11. Запись в телефонную книгу. Используя максимум данных.
12. Покупки через интернет.
13. Оплата штрафов (пример с карт. Счета, через мультикассу или иными способами).
14. Регистрация по месту прибивания.

Содержание 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 инфо- логическую модель БД на языке таблица-связь, описание разработанных ограничений целостности, рисунки с отчетами Data Dictionary Report для каждой таблицы БД.

Контрольные вопросы

1. Что такое транзакция?
2. Перечислите виды транзакции.
3. Краткое описание видов транзакции.
4. Основные команды для работы с транзакциями?
5. Что такое хранимая процедура?
6. Что такое откатка? И в каких случаях происходит?
7. Свойства транзакции.
8. Источники отказов.
9. Что такое фиксированная транзакция?
10. Классификация систем обработки транзакций

Рекомендуемая литература: 1-5.

4. 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.

Теоретическая часть

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.

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.

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 МУСС. 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.

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 -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.

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;
- ItemldData - 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 (ItemldData). 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 СТШ), 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 байта 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 О - 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-значе- ние 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.

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 b права по умолчанию.

Правила

Механизм правил (rules) представляет собой механизм создания пользовательских обработчиков не только операцш 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 (views) 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

  CIDR (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

  UUID (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

  XML (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

  OID (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

  Композитный (составной) тип (composite type) представляет структуру ряда или записи, т.е. по существу список имен полей и их типов данных.
- Псевдотипы

  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 табл. 1 перечислены основные базовые типы данных PostgreSQL, а также их синонимы (альтернативные имена).

Таблица 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 п6ит)	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
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

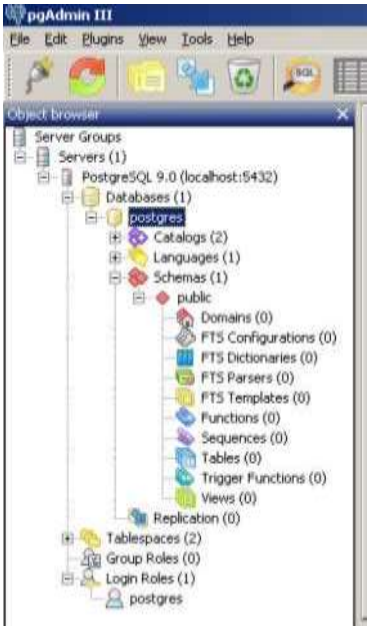


Рис. 1. Основные компоненти базы данных PostgreSQL 9.0.

Как и в SQL Server, данные в базе данных PostgreSQL организованы в нескольких разных объектах:(рис. 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).

1. Разработка логической модели базы данных

1.1. 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

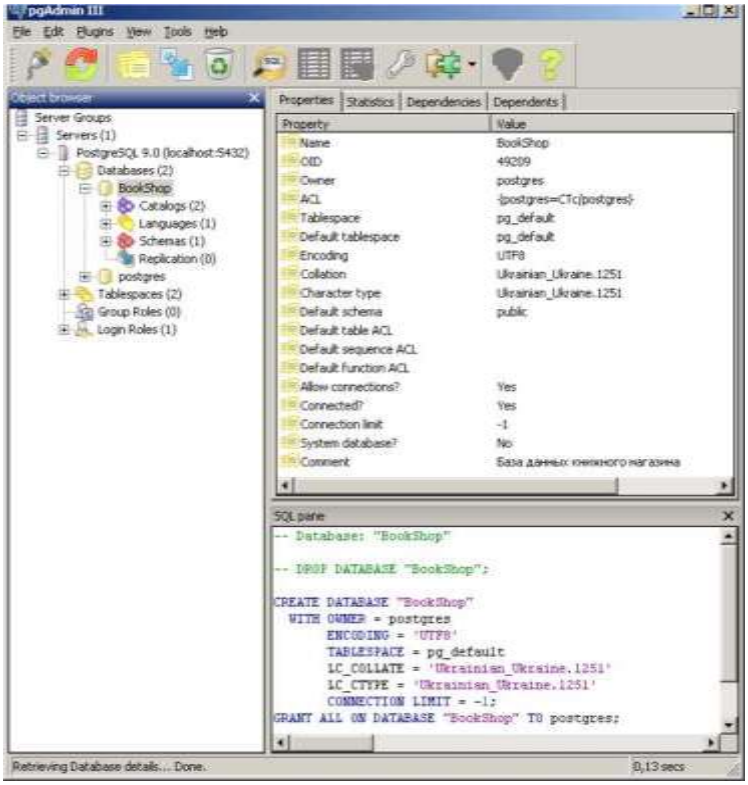
- Использование пакета с графическим интерфейсом^Ш):

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.

1.2. Пример создания базы данных.



Шаг 1: Создание базы данных BookShop

Шаг 2: Создаем таблицы Книги, Поставщики, Заказчики, Заказы, Поставки.

Параметры базовой базы данных показаны на рис. 2

После того, как база данных создана, можно приступать к созданию ее основных объектов - таблиц. Прежде всего, для каждого столбца любой таблицы необходимо указать определенный тип данных. В SQL пип 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 базы данных, которые хранятся в системной таблице pg_type вместе с их OID.

Создаем таблицы со следующими полями:

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 сам 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(гс^1пс1);
- 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 т-а^{3}к$^{8-}$ая характеристика связей, как степень связи. На рис. 3 показана диаграмма базы данных BookShop.

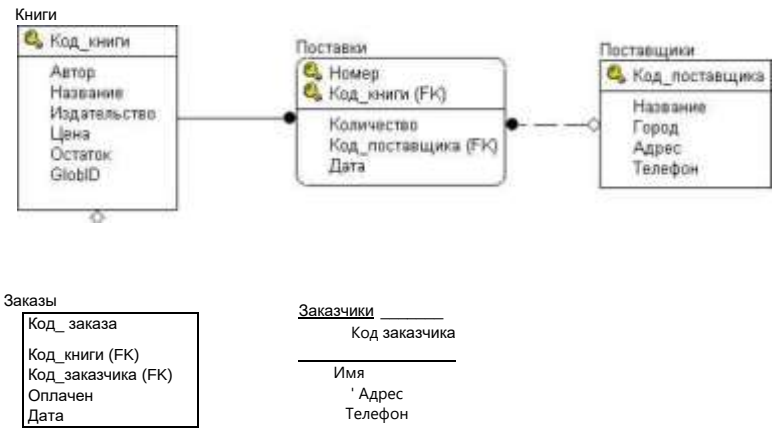


Рис. 3. Диаграмма «таблица-связь» базы данных BookShop.

Сплошная линия - идентифицирующие связи, пунктирные - 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. Далее вводим для столбцов созданных таблиц следующие ограничения:

Таблица	**Поле**	**Ограничение**
Поставщики	Код поставщика	Primery key
	Название	Unique
	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”, “Ні”)
	Дата	Defaults (timenow())
Заказчики	Код заказчика	Primery key
	Телефон	Defaults (“000 111-11-11”)
Книги	Код книги	Primery key
Поставки	Номер	Primery key
	Код_книги	Primery key Foreign key (Код_книги из Книги, NULL-значения не допускаются, 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 генерируем отчет **Data Dictionary Report**. 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
рк_поставки	Primary key	("Номер")	
fk_nocmaeKu_1	Foreign key	("Код_книги") REFERENCES "Книги" ("Код_книги") MATCH SIMPLE ON UPDATE RESTRICT ON DELETE RESTRICT	
fk_noCTaeku_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 базы данных. Не следует забывать, что: 1) внешние ключи могут принимать только те значения, которые уже введены (!) в поля тех таблиц, на которые эти внешние ключи ссылаются; 2) в столбцы с типом serial (счетчик) нельзя самим вводить значения; 3) 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 базе данных, для чего нужно выполнить команду Refresh, нажав одноименную кнопку на панели инструментов окна Edit Data.

Чтобы удалить какую-либо строку таблицы, достаточно выделить ее, щелнув 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 базе данных.

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

2. Контрольное задание.

3.1. Используя метод нормализации универсального отношения, разработать инфологическую модель базы данных, определить ограничения целостности, создать БД и

ввести тестовые данные в каждую из созданных таблиц.

№	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	Комунальные платежи.
	Код услуги, наименование, единица измерения, тарифная зона, стоимость единицы, лицевой счет клиента, ФИО кшента, 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, Ф1О 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, Ф1О 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 дебитора, банк дебитора, юридическое наименование дебитора, номер счета кредитора, банк кредитора, юридическое наименование кредитора, номер договора о кредитовании, дата выдачи кредита, срок пользования кредитом, сумма кредита, процентная ставка.
11	Экспорт/импорт.
	Код предприятия, название предприятия, юридический адрес, телефон/факс, WWW, балансовый год, валовой доход, чистая прибыль, код товарной номенклатуры внешнеэкономической деятельности (ТНЗД), наименование ТНЗД, код группы производства (ГВ), наименования ГВ, год операции экспорта/импорта, объем экспорта/импорта, рейтинг предприятия.
12	Проект.
	Код работы, наименование работы, дата начала, самая ранняя дата начала, дата окончания, самая поздняя дата окончания, общий объем в человеко-часах, общая потребность в материалах в условных единицах, номер вехи, показатель объема выполненной работы в процентах к общему объему, показатель использованных материалов в процентах к общей потребности, код исполнителя, наименования исполнителя, юридический адрес исполнителя, номер недели с начала года, объем выполненной работы, количество использованных материалов.

2.2. Оформить отчет по результатам выполнения контрольного задания:

Отчет должен содержать:

1. Инфологическую модель БД (на языке таблица-связь).
2. Разработанные ограничения целостности.
3. Отчеты Data Dictionary Report, сгенерированные для каждой созданной при выполнении контрольного задания таблицы.
4. Отчеты Data Dictionary Report, сгенерированные для каждой таблицы БД BookShop.

Контрольные вопросы:

1. Сформулируйте основные свойства реляционной таблицы.
2. Что такое первичный ключ?
3. Что такое норма-л42и-зация отношений? Дайте определение 1NF, 2NF, 3NF, NFBK.
4. Сформулируйте правила нормализации отношений.
5. Из каких файлов состоит база данных SQL Server?
6. Сформулируйте правила обеспечения целостности данных в реляционных СУБД.
7. Что такое инфологическая модель базы данных?
8. Что такое логическая модель базы данных?
9. Из каких стандартных объектов состоит модель базы данных SQL Server?
10. Что такое транзакция?
11. Что такое глобальный уникальный идентификатор?
12. Что такое декларативная ссылочная целостность?
13. Что такое экстент? Какие типы экстентов используются в SQL Server?
14. Почему в SQL Server используются несколько журналов транзакций, расположенных на разных физических дисках?

15 Какова структура файла базы данных SQL Server?
16 Для чего и как определяются ограничения на значение столбцов?
17 Что такое ограничение первичного ключа?
18 Что такое ограничение внешнего ключа?
19 Что такое связывание?
20 Для чего предназначены правила? В чем разница между правилами и ограничениями на значение столбцов?
21 В чем разница между значением по умолчанию и установкой по умолчанию?
Что такое ниладические функции? Для чего они используются?
22 Привидите пример.
Каков размер страницы в SQL Server?
23 Какие типы страниц поддерживает SQL Server ?
24 Для чего предназначены страницы GAM в файлах баз данных SQL
25 Server? Какова их структура?
Что такое представление?
26 Что такое триггер?
27 Что такое ограничения целостности? Какие ограничения
28 целостности поддерживает PostgreSQL?
Для чего предназначена и в чем суть технологии MVCC?
29 Что такое WAL?
30 Что представляет собой технология TOAST? В чем особенности ее
31 реализации
Какова структура страниц файла данных PostgreSQL?
32 В чем заключается концепция табличных пространств,
33 используемая в PostgreSQL?
34. Что такое схемы в PostgreSQL? Для чего они предназначены?

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по организации самостоятельной работы по дисциплине
«Методы построения защищённых баз данных»
ля студентов направления подготовки 10.04.01 «Информационная безопасность»
(направленность (профиль) "Математическое и программное обеспечение защиты информации")

Ставрополь 2025

СОДЕРЖАНИЕ

Введение

Целью изучения дисциплины «Методы построения защищенных баз данных» является обучение студентов принципам обеспечения безопасности информации в автоматизированных информационных системах (АИС), основу которых составляют базы данных (БД), навыкам работы со встроенными в системы управления базами данных (СУБД) средствами защиты.

Задачи курса:

– приобретение системного подхода к проблеме защиты информации в СУБД;
– изучение моделей и механизмов защиты в СУБД;
– приобретение практических навыков организации защиты БД.

Компетенции обучающегося, формируемые в результате освоения дисциплины:

1. Общая характеристика самостоятельной работы студента

Основными видами учебной работы по достижению результатов освоения дисциплины являются лекции, лабораторные работы, практические занятия и самостоятельная работа студентов (СРС).

На лекциях раскрываются основные положения и понятия курса, формируются знания в области теории сложности комбинаторных алгоритмов. На лабораторных работах и практических занятиях формируются умения и навыки, необходимые для решения задач профессиональной деятельности.

Приступая к изучению учебной дисциплины, необходимо ознакомиться с учебной программой, получить в библиотеке рекомендованные учебные пособия, а также получить у ведущего преподавателя в электронном виде конспекты лекций, методические рекомендации к лабораторным занятиям и тестовые задания, завести новую тетрадь для конспектирования лекций и выполнения практических заданий.

Для изучения дисциплины предлагается список основной и дополнительной литературы. Основная литература предназначена для обязательного изучения, дополнительная – поможет более глубоко освоить отдельные вопросы, подготовить исследовательские задания и выполнить задания для самостоятельной работы.

В ходе лекционных занятий студент обязан осуществлять конспектирование учебного материала, особое внимание, обращая на категории, формулировки, раскрывающие содержание тех или иных понятий, физических явлений, процессов, эффектов. В рабочих конспектах желательно оставлять поля, на которых следует делать пометки из рекомендованной литературы, дополнять материал прослушанной лекции, формулировать научные выводы и практические рекомендации. Студент имеет право задавать преподавателю уточняющие вопросы с целью уяснения теоретических положений, разрешения спорных ситуаций.

Самостоятельная работа студентов над материалом учебной дисциплины является неотъемлемой частью учебного процесса и должна предполагать углубление знания учебного материала, излагаемого на аудиторных занятиях, и приобретение дополнительных знаний по отдельным вопросам самостоятельно.

Основными видами самостоятельной работы студентов по учебной дисциплине являются:

– самостоятельное изучение учебного материала по заданным темам;
– подготовка к лабораторным занятиям, оформление отчета по выполненному лабораторной работе и подготовка к собеседованию по результатам работы.

Основными методами самостоятельной работы студентов являются:

1) для овладения знаниями:

– чтение текста (конспекта лекций, учебника, дополнительной литературы) и

конспектирование текста;

– работа с нормативными документами;
– поиск информации в Интернет;

2) для закрепления и систематизации знаний:

– работа с конспектом лекции (обработка текста);
– повторная работа над учебным материалом (учебника, дополнительной литературы, слайдами);
– составление плана и тезисов ответа или графическое изображение структуры ответа;
– составление таблиц для систематизации учебного материала;
– изучение нормативных документов;
– ответы на контрольные вопросы;
– тестирование в программе компьютерного тестирования «Iren»;

3) для формирования умений:

– подготовка к лабораторным занятиям;
– решение задач и практических заданий;
– подготовка отчетов по лабораторным занятиям;
– рефлексивный анализ профессиональных умений.

В случае пропуска учебного занятия студент может воспользоваться содержанием различных блоков учебно-методического комплекса (лекции, практические занятия, контрольные вопросы и тесты) для самоподготовки и освоения темы. Для самоконтроля необходимо использовать вопросы и задания, предлагаемые к лабораторным занятиям.

2. Технологическая карта самостоятельной работы студента

Коды реализуемых компетенций, индикатора(ов)	Вид деятельности студентов	Средства и технологии оценки	Объем часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
3 семестр					
ИД-1 УК-2 ИД-2 УК-2 ИД-1 ПК-2 ИД-2 ПК-2 ИД-3 ПК-2	Подготовка к лабораторной работе	Собеседование, проверка конспектов	40,5		40,5
Итого за 3 семестр			40,5		40,5
Итого			40,5		40,5

3. Самостоятельное изучение тем лекций

№ п/п	Темы для самостоятельного	Рекомендуемые источники информации (№ источника)

	изучения	Основная	Дополни-тельная	Методи-ческая	Интернет-ресурсы
1.	Тема: Механизмы обеспечения конфиденциальности в СУБД	2	1,2	1, 2	1, 2,3
2.	Тема: СУБД MariaDB (MySQL).	1	1, 2, 4	1, 2	1, 2,3

4. Паспорт фонда оценочных средств для проверки самостоятельной работы

Код оцениваемой компетенции (или её части)	Этап формирования компетенции (№ темы)	Тип контроля	Вид контроля	Количество элементов, шт.	
				Базовый	Повышенный
ИД-1 УК-2 ИД-2 УК-2 ИД-1 ПК-2	1-2	текущий	Вопросы для собеседования по самостоятельному изучению тем дисциплины	4	2

5. Вопросы для собеседования по самостоятельному изучению тем дисциплины

Тема 1. Механизмы обеспечения конфиденциальности в СУБД
Базовый уровень:
1. Что такое СУБД?
2. Как обеспечить конфиденциальность информации в СУБД?

Повышенный уровень:
3. Основные принципы защиты информации в СУБД.

Тема 2. СУБД MariaDB (MySQL)
Базовый уровень:
1. Что такое СУБД MariaDB?
2. Чем СУБД MariaDB отличается от обычной СУБД?

Повышенный уровень:
3. Как обеспечить конфиденциальность информации в СУБД MariaDB?

Критерии оценки для проверки уровня обученности:

Оценка «отлично» выставляется студенту, если студент показал глубокое, прочное и аргументированное освоение программного учебного материала, при этом поставленный вопрос раскрыт последовательно, четко и логически стройно, в полном исчерпывающем объеме, основные категории, понятия и термины учебного курса формулировались правильно, не допущено при ответе ошибок.

Оценка «хорошо» выставляется студенту, если студент показал твердое знание программного учебного материала, при этом поставленный вопрос раскрыт грамотно и по существу, в достаточно полном объеме, основные категории, понятия и термины учебного курса формулировались правильно, допущены при ответе отдельные неточности или одна ошибка.

Оценка «удовлетворительно» выставляется студенту, если студент показал знание только основной части учебного материала без его частных деталей, при этом поставленный

вопрос раскрыт с нарушением логической последовательности, не в полном объеме; были допущены неточные формулировки основных категорий, понятий и терминов учебного курса, а также ошибки (не более двух) или ряд незначительных неточностей, не исказивших существенно суть ответа.

Оценка «неудовлетворительно» выставляется студенту, который не знает значительной части программного материала, допускает существенные ошибки (более двух), существенно исказившие его суть. Оценка неудовлетворительно выставляется также, если отсутствует ответ на вопрос, либо студент отказался его сдавать.

6. Список рекомендуемой литературы

Основная литература:

1. Колисниченко Д. PHP и MySQL. Разработка Web-приложений.- БХВ-Петербург, 2014.
2. Бекаревич Ю. Access 2010 . .- БХВ-Петербург, 2013.
3. Иванова И, М.М. Корниенко. Базы данных. Системы управления базами данных .- БХВ-Петербург, 2011.
4. Колисниченко Д.Н. PHP 5/6 и MySQL 6. Разработка Web-приложений.- БХВ-Петербург, 2013.
5. Новиков Б.А. Настройка приложений баз данных. Гриф УМО МО РФ. -БХВ-Петербург, 2012.
6. Дунаев В.В. Базы данных. Язык SQL для студента.- БХВ-Петербург, 2012.
7. Колисниченко Д. Н. PHP и MySQL. Разработка веб-приложений.- BHV, 2015.
8. Благодаров А. В. Алгоритмы категорирования персональных данных для систем автоматизированного проектирования баз данных информационных систем Издатель - BHV, 2015.

Дополнительная литература:

1. Кузнецов С.Д. Основы современных баз данных. Учебник. - http://www.citforum.ru/database/osbd/contents.shtml
2. Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие. – http://www.citforum.ru/database/dbguide/index.shtml
3. Носков Ю.М. Система программирования Delphi. - раздел «Основы работы с базами данных». – http://www.mgopu.ru/PVU/2.1/Delphi/index.html
4. Пушников А.Ю. Введение в системы управления базами данных. Учебное пособие/Изд-е Башкирского ун-та. - Уфа, 1999. - 246 с. - http://www.citforum.ru/database/dblearn/index.shtml
5. Козленко Л. Информационная безопасность в современных системах управления базами данных. - КомпьютерПресс 3'2002. -http://www.citforum.ru/security/articles/safe_db/
6. Гайдамакин Н.А. Автоматизированные информационные системы, базы и банки данных. Вводный курс: Учебное пособие. – М.: Гелиос АРВ, 2002.
7. Гайдамакин Н.А. Разграничение доступа к информации в компьютерных системах. – Екатеринбург: Изд-во Урал. Ун-та, 2003. – 328 с.
8. Базы данных. Учебник для вузов под ред. А. Хомоненко. – М.: Корона Принт.
9. Вьюкова Н.И., Галатенко В.А. Информационная безопасность систем управления базами данных. СУБД, 1996, № 1.
10. Белкин П.Ю., Михальский О.О., Першаков А.С. и др. Защита программ и данных: Учебное пособие для ВУЗов. – М.: Радио и связь, 1999.
11. Кузнецов С.Д. СУБД и файловые системы. – М.: Майор, 2001.
12. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. -http://www.citforum.ru/database/case/index.shtml.

Методическая литература:

1. Электронный курс лекций.
2. Электронные методические рекомендации для выполнения лабораторных заданий.

Интернет-ресурсы:

1. Курсы информационных технологий Яндекса (КИТ)https://academy.yandex.ru/events/kit/5/
2. Базы данных ("обзорно-понятийный курс", создатели: Яндекс, CS клуб при ПОМИРАН и т.д.):(2012г.) - http://www.lektorium.tv/course/?id=22894

Программное обеспечение:

1. Microsoft Visio 2012
2. Операционная система Linux
3. (PostgreSQL)
4. MariaDB (MySQL).

Материально-техническое обеспечение дисциплины

1. Мультимедийная лекционная аудитория № 431.
2. Аудитории, оснащённые компьютерами с установленным требуемымпрограммным обеспечением, для проведения лабораторных работ.