МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ РЕАЛИЗАЦИИ ПРОГРАММНОГО МОДУЛЯ ПМ. 11 РАЗРАБОТКА, АДМИНИСТРИРОВАНИЕ И ЗАЩИТА БАЗ ДАННЫХ

для специальности

09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ ПРОГРАММИРОВАНИЕ

Базовая подготовка
среднего профессионального образования

2024

Содержание

Пояснительная записка

Методические указания по ПМ.11. Разработка, администрирование и защита баз данных разработаны на основе рабочей программы ПМ.11. Разработка, администрирование и защита баз данных по специальности 09.02.07 Информационные системы и программирование.

Реализация профессионального модуля предусматривает проведение лабораторных и практических работ в форме практической подготовке обучающихся.

Практическая подготовка при реализации профессионального модуля организуется путем проведения практических занятий, практикумов, лабораторных работ и иных аналогичных видов учебной деятельности, предусматривающих участие обучающихся в выполнении отдель- ных элементов работ, связанных с будущей профессиональной деятельностью, а также демон- страцию практических навыков, выполнение, моделирование обучающимися определенных видов работ для решения практических задач, связанных с будущей профессиональной деятель- ностью в условиях, приближенных к реальным производственным.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся в ходе освоения профессионального модуля **должен**

иметь практический опыт:

– по работе с объектами базы данных в конкретной системе управления базами данных;
– в использовании стандартных методов защиты объектов базы данных;
– по работе с документами отраслевой направленности.

уметь:

– работать с современными case-средствами проектирования баз данных;
– проектировать логическую и физическую схемы базы данных;
– создавать хранимые процедуры и триггеры на базах данных; применять стандартные методы для защиты объектов базы данных; выполнять стандартные процедуры резервного копирования и мониторинга выполнения этой процедуры;
– выполнять процедуру восстановления базы данных и вести мониторинг выполнения этой процедуры;
– обеспечивать информационную безопасность на уровне базы данных

знать:

– основные положения теории баз данных, хранилищ данных, баз знаний;
– основные принципы структуризации и нормализации базы данных; основные принципы построения концептуальной, логической и физической модели данных;
– методы описания схем баз данных в современных системах управления базами данных;
– структуры данных систем управления базами данных, общий подход к организации представлений, таблиц, индексов и кластеров; методы организации целостности данных;
– способы контроля доступа к данным и управления привилегиями; основные методы и средства защиты данных в базах данных

Правила выполнения лабораторных и практических работ

Подготовка к лабораторным и практическим работам. Лабораторные и практические работы в группах проводятся в соответствии с расписанием учебных занятий в колледже в течение определенного времени. Поэтому для выполнения лабораторных работ студент должен руководствоваться следующими положениями:

– Предварительно ознакомиться с графиком выполнения лабораторных работ;
– Внимательно ознакомиться с описанием соответствующей лабораторной работы и установить, в чем состоит основная цель и задача этой работы;
– По лекционному курсу и соответствующим литературным источникам изучить теоретическую часть, относящуюся к данной лабораторной работе;
– Неподготовленные к работе студенты к выполнению лабораторной работы не допускаются.

После окончания работы в лаборатории рабочее место должно быть приведено в поря- док. В течение всего времени занятий в лаборатории студенты обязаны находиться на своих рабочих местах. Выходить из помещения лаборатории во время занятий можно только с разре- шения преподавателя.

Оформление отчета по лабораторным и практическим работам.

Составление отчета о проведенных исследованиях является важнейшим этапом выпол- нения лабораторной работы. По каждой выполненной работе в рабочей тетради составляют от- чет, руководствуясь следующими положениями:

– Указать название и порядковый номер лабораторной работы, а так же краткое сформулировать цель работы;

– Схемы и графики чертить с соблюдением принятых стандартных условий обозначений;

– Отчет по каждой лабораторной работе должен содержать основные выводы. В заголовке отчета указывают номер работы и ее полное наименование. При составлении отчета нужно кратко описать цель работы, ее содержание, указать использованные аппаратуру и обо- рудование.

– При выполнении лабораторных работ необходимо строго следовать правилам техники безопасности.

Критерии оценки работ

Оценка «отлично» ставится, если обучающийся выполнил работу в полном объеме с соблюдением необходимой последовательности действий; в «Отчете к практиче- ским работам» правильно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ ошибок.

Оценка «хорошо» ставится, если обучающийся выполнил требования к оценке "5", но допущены 2-3 недочета.

Оценка «удовлетворительно» ставится, если обучающийся выполнил работу не полностью, но объем выполненной части таков, что позволяет получить правильные результаты и выводы; в ходе проведения работы были допущены ошибки.

Оценка «неудовлетворительно» ставится, если обучающийся выполнил работу не полностью или объем выполненной части работы не позволяет сделать правильных выводов.

Перечень практических работ

Тема	Наименование	Количество во часов
Тема 1.1. Основы хранения и обработки данных. Проектирование БД.	Практическая работа №1 «Сбор и анализ информации»	2
	Практическая работа №2 «Проектирование реляционной схемы базы данных в среде СУБД»	3
	Практическая работа №3 «Приведение БД к нормальной форме 3НФ»	3
Тема 1.2. Разработка и администрирование БД.	Практическая работа №4 «Создание базы данных в среде разработки»	4
	Практическая работа №5 «Организация локальной сети. Настройка локальной сети»	5
	Практическая работа №6 «Установка и настройка SQL-сервера»	5
	Практическая работа №7 «Экспорт данных базы в документы пользователя»	4
	Практическая работа №8 «Импорт данных пользователя в базу данных»	4
	Практическая работа №9 «Выполнение настроек для автоматизации обслуживания базы данных»	4
	Практическая работа №10 «Мониторинг работы сервера»	4
Тема 1.3. Организация защиты данных в хранилищах	Практическая работа №11 «Выполнение резервного копирования»	4
	Практическая работа №12 «Восстановление базы данных из резервной копии»	4
	Практическая работа №13 «Реализация доступа пользователей к базе данных»	5
	Практическая работа №14 «Мониторинг безопасности работы с базами данных»	4
	Практическая работа №15 «Установка приоритетов»	5
	Практическая работа №16 «Развертывание контроллеров домена»	4
	Практическая работа №17 «Мониторинг сетевого трафика»	4
Итого		68

Практическая работа № 1. Проектирование реляционной схемы базы данных в среде СУБД.

Цель работы: приобретение практических навыков анализа предметной области, информационных задач и построения концептуальной модели базы данных.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

При проектировании системы обработки данных именно данные и интересуют нас в первую очередь. Причём больше всего нас интересует организация данных. Помочь понять ор- ганизацию данных, призвана информационная модель реального мира, на которой и основана система автоматизированной обработки данных.

Чтобы понять процесс построения информационной модели, необходимо знать ряд тер- минов, которые применяются при описании и представлении данных:

База данных – (БД, database) - поименованная совокупность структурированных данных, относящихся к определенной предметной области.

Предметная область - некоторая часть реально существующей системы, функционирующая как самостоятельная единица. Полная предметная область может представлять собой экономику страны или группы союзных государств, однако на практике для информационных систем наибольшее значение имеет предметная область масштаба отдельного предприятия или корпорации.

Система управления базами данных (СУБД) - комплекс программных и языковых средств, необходимых для создания и модификации *базы данных*, добавления, модификации, удаления, поиска и отбора информации, представления информации на экране и в печатном виде, разграничения прав доступа к информации, выполнения других операций с базой.

Реляционная БД - основной тип современных *баз данных*. Состоит из таблиц, между которыми могут существовать связи по ключевым значениям.

Таблица *базы данных* (table) - регулярная структура, которая состоит из однотипных строк (записей, records), разбитых на столбцы (поля, fields).

В теории реляционных *баз данных* синоним таблицы - отношение (relation), в котором строка называется кортежем, а столбец называется атрибутом.

В концептуальной модели реляционной БД аналогом таблицы является сущность (entity), с определенным набором свойств - атрибутов, способных принимать определенные значения (набор допустимых значений - домен).

Ключевой элемент таблицы (ключ, regular key) - такое ее поле (простой ключ) или стро- ковое выражение, образованное из значений нескольких полей (составной ключ), по которому можно определить значения других полей для одной или нескольких записей таблицы. На прак- тике для использования ключей создаются индексы - служебная информация, содержащая упо- рядоченные сведения о ключевых значениях. В реляционной теории и концептуальной модели понятие "ключ" применяется для атрибутов отношения или сущности.

Первичный ключ (primary key) - главный ключевой элемент, однозначно идентифицирующий строку в таблице. Могут также существовать альтернативный (candidate key) и уни- кальный (unique key) ключи, служащие также для идентификации строк в таблице.

В реляционной теории первичный ключ - минимальный набор атрибутов, однозначно идентифицирующий кортеж в отношении.

В концептуальной модели первичный ключ - минимальный набор атрибутов сущности, однозначно идентифицирующий экземпляр сущности.

Связь (relation) - функциональная зависимость между объектами. В реляционных *базах данных* между таблицами устанавливаются связи по ключам, один из которых в главной (parent, родительской) таблице - первичный, второй - внешний ключ - во внешней (child, дочер- ней) таблице, как правило, первичным не является и образует связь "один ко многим" (1:M). В случае первичного внешнего ключа связь между таблицами имеет тип "один к одному" (1:1). Информация о связях сохраняется в базе данных.

Внешний ключ (foreign key) - ключевой элемент подчиненной (внешней, дочерней) таблицы, значение которого совпадает со значением первичного ключа главной (родительской) таблицы.

Ссылочная целостность данных (referential integrity) - набор правил, обеспечивающих соответствие ключевых значений в связанных таблицах.

Хранимые процедуры (stored procedures) - программные модули, сохраняемые в базе данных для выполнения определенных операций с информацией базы.

Триггеры (triggers) - хранимые процедуры, обеспечивающие соблюдение условий ссы- лочной целостности данных в операциях изменения первичных ключей (возможно каскадное изменение данных), удалении записей в главной таблице (каскадное удаление в дочерних таб- лицах) и добавлении записей или изменении данных в дочерних таблицах.

Репликация базы данных - создание копий базы данных (реплик), которые могут обме- ниваться обновляемыми данными или реплицированными формами, отчетами или другими объектами в результате выполнения процесса синхронизации.

Транзакция - изменение информации в базе в результате выполнения одной операции или их последовательности, которое должно быть выполнено полностью или не выполнено во- обще. В СУБД существуют специальные механизмы обеспечения транзакций.

Язык SQL (Structured Query Language) - универсальный язык работы с базами данных, включающий возможности ее создания, модификации структуры, отбора данных по запросам, модификации информации в базе и прочие операции манипулирования базой данных.

Null - значение поля таблицы, показывающее, что информация в данном поле отсут- ствует. Разрешение на возможность существования значения Null может задаваться для отдель- ных полей таблицы.

Администратор базы данных *(АБД)* – лицо, или группа лиц, уполномоченных на ве- дение БД (модификация структуры и содержания БД, активизация доступа пользователей, вы- полнение других административных функций, которые затрагивают всех пользователей).

Банк данных *(БнД)* – система специально организованных данных, программных, языковых, организационных и технических средств. Предназначенных для централизованного накопления и коллективного многоцелевого использования данных.

Внешняя схема – представление данных с точки зрения пользователя или прикладной программы.

Внутренняя схема – физическая структура данных.

Логическая структура *БД* – определение БД на физически независимом уровне.

Модель концептуальная *(инфологическая)* – описание предметной области, выпол- ненное с использованием естественного языка, математических выражений, таблиц, графов и других средств, понятных всем людям, работающим над проектированием базы данных.

Модель физическая – определяющая размещение и способы поиска данных на внеш- них запоминающих устройствах СУБД.

Нормализация – представление сложных структур данных (документов) в виде дву- мерных таблиц (отношений).

Проектирование *БД* – упорядоченный формализованный процесс создания системы взаимосвязанных описаний - таких моделей предметной области, которые связывают (фикси- руют) хранимые в базе данные с объектами предметной области, описываемые этими данными. **Распределенная *БД*** – совокупность баз данных, которые обрабатываются и управля-
ются по отдельности, а также могут разделять информацию.

Объектом называется элемент информационной системы. В реляционной теории баз данных объект называется также *сущностью.* Объект может быть реальным, например - чело- век, какой - либо предмет или населённый пункт, и абстрактным - событие, счёт покупателя или курс изучаемый студентами.

Классом объектов называется совокупность объектов, обладающих одинаковым набо- ром свойств. Объекты и их свойства являются понятиями реального мира, В мире информации, существующем в представлении программиста свойства объектов называют атрибутами.

Атрибут - это информационное отображение свойств объекта. Например, клиент магазина, продающего автомобили, имеет такие атрибуты, как фамилию, имя, отчество, адрес, и возможно, идентификационный номер. Атрибут при реализации информационной модели на каком-либо носителе информации часто называют *элементом данных, полем данных* или просто *полем.*

Доменом называется набор записей данных одного типа, отвечающих поставленным условиям.

II. Классификация баз данных

По технологии обработки данных базы данных подразделяются на централизованные и распределенные.

Централизованная база данных хранится в памяти одной вычислительной системы. Эта вычислительная система может быть мэйнфреймом - тогда доступ к ней организуется с использованием терминалов - или файловым сервером локальной сети ПК.

Распределенная база данных состоит из нескольких, возможно, пересекающихся или даже дублирующих друг друга частей, которые хранятся в различных ЭВМ вычислительной сети. Работа с такой базой осуществляется с помощью системы управления распределенной базой данных (СУРБД).

По способу доступа к данным базы данных разделяются на **базы данных с локальным доступом** и **базы данных с сетевым доступом**.

Для всех современных баз данных можно организовать сетевой доступ с многопользо- вательским режимом работы.

Централизованные базы данных с сетевым доступом могут иметь следующую архитектуру:

- файл-сервер;
- клиент-сервер базы данных;
- "тонкий клиент" - сервер приложений - сервер базы данных (трехуровневая архитектура).

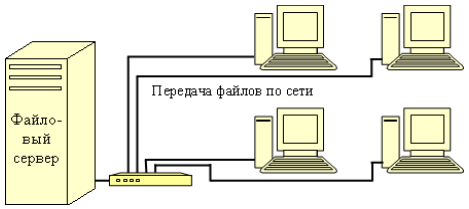

Рис.1. Схема работы с БД в локальной сети с выделенным файловым сервером

Файл-сервер. Архитектура систем БД с сетевым доступом предполагает выделение одной из машин сети в качестве центральной (файловый сервер). На этот компьютер устанавливается операционная система (ОС) для выделенного сервера (например, Microsoft Windows Server 2003). На нем же хранится совместно используемая централизованная БД в виде одного или группы файлов. Все другие компьютеры сети выполняют функции рабочих станций (могут работать в ОС Microsoft Windows 2000 Professional или Microsoft Windows 98). Файлы базы данных в соответствии с пользовательскими запросами передаются на рабочие станции, где и производится обработка информации. При большой интенсивности доступа к одним и тем же данным производительность информационной системы падает. Пользователи могут создавать также локальные БД на рабочих станциях.

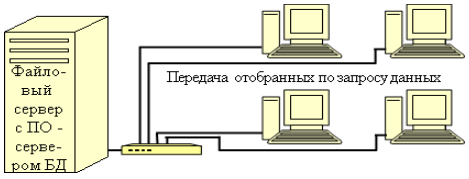

Рис. 2. Схема работы с БД в архитектуре "Клиент-сервер"

Клиент-сервер. В этой архитектуре на выделенном сервере, работающем под управле- нием серверной операционной системы, устанавливается специальное программное обеспече- ние (ПО) - сервер БД, например, Microsoft SQL Server или Oracle. СУБД подразделяется на две части: клиентскую и серверную. Основа работы сервера БД - использование языка запросов (SQL). Запрос на языке SQL, передаваемый клиентом (рабочей станцией) серверу БД, порож- дает поиск и извлечение данных на сервере. Извлеченные данные транспортируются по сети от сервера к клиенту. Тем самым, количество передаваемой по сети информации уменьшается во много раз.

Трехуровневая архитектура функционирует в Интранет- и Интернет-сетях. Клиентская часть ("тонкий клиент"), взаимодействующая с пользователем, представляет собой HTML- страницу в Web-браузере либо Windows-приложение, взаимодействующее с Web-сервисами. Вся программная логика вынесена на сервер приложений, который обеспечивает формирова- ние запросов к базе данных, передаваемых на выполнение серверу баз данных. Сервер прило- жений может быть Web-сервером или специализированной программой (например, Oracle Forms Server).

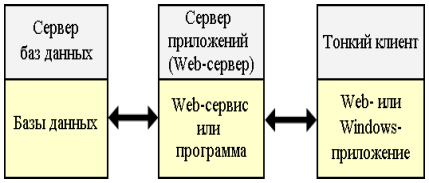

Рис. 3. Схема работы с БД в трехуровневой архитектуре

По типу используемой модели данных. БД может быть основана на одной модели или на совокупности нескольких моделей. Любую модель данных можно рассматривать как объект, который характеризуется своими свойствами (параметрами), и над ней, как над объектом, можно производить какие-либо действия.

Существуют три основных типа моделей данных – реляционная, иерархическая и сетевая.

Реляционная модель. Термин «реляционный» (от латинского relatio – отношение) указывает, прежде всего, на то, что такая модель хранения данных построена на взаимоотношении составляющих ее частей. В простейшем случае она представляет собой двухмерный массив или двухмерную таблицу, а при создании сложных информационных моделей составит совокуп- ность взаимосвязанных таблиц. Каждая строка такой таблицы называется записью, а столбец – полем.

Реляционная модель данных имеет следующие свойства:

– Каждый элемент таблицы – один элемент данных.
– Все поля в таблице являются однородными, т.е. имеют один тип.
– Каждое поле имеет уникальное имя.
– Одинаковые записи в таблице отсутствуют.
– Порядок записей в таблице может быть произвольным и может характеризоваться ко- личеством полей, типом данных.

Номер сотрудника	Фамилия	Зарплата	Номер отдела
1	Иванов	1000	1
2	Петров	2000	2
3	Сидоров	3000	1

Рис. 4 Реляционная модель данных

Иерархическая модель. Иерархическая модель БД представляет собой совокупность элементов, расположенных в порядке их подчинения от общего к частному и образующих перевернутое дерево (граф). Данная модель характеризуется такими параметрами, как уровни, узлы, связи. Принцип работы модели таков, что несколько узлов более низкого уровня соединяются при помощи связи с одним узлом более высокого уровня.

Узел – информационная модель элемента, находящегося на данном уровне иерархии.

Свойства иерархической модели данных:

– Несколько узлов низшего уровня связано только с одним узлом высшего уровня.

– Иерархическое дерево имеет только одну вершину (корень), не подчиненную никакой другой вершине.

– Каждый узел имеет свое имя (идентификатор).

– Существует только один путь от корневой записи к более частной записи данных.

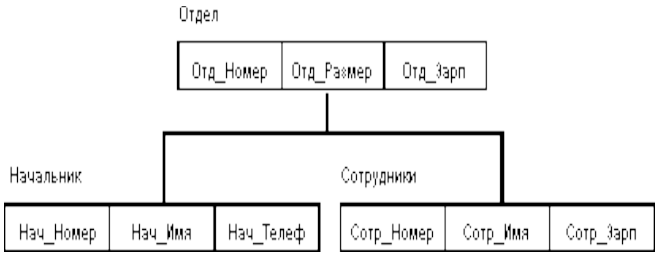

Рис.5 Иерархическая модель данных

Сетевая модель. Сетевая модель БД похожа на иерархическую. Она имеет те же основные составляющие (узел, уровень, связь), однако характер их отношений принципиально иной. В сетевой модели принята свободная связь между элементами разных уровней.

Рис. 6 Сетевая модель данных

III. Проектирование базы данных

Жизненный цикл БД. Как и любой программный продукт, база данных обладает соб- ственным жизненным циклом (ЖЦБД). Главной составляющей в жизненном цикле БД является создание единой базы данных и программ, необходимых для работы. Жизненный цикл системы базы данных определяет и жизненный цикл всей информационной системы организации, по- скольку база данных является фундаментальным компонентом информационной системы.

ЖЦБД включает в себя следующие основные этапы

– - планирование разработки базы данных;

– - определение требований к системе;

– - сбор и анализ требований пользователей;

– - проектирование базы данных:

концептуальное проектирование базы данных;

логическое проектирование базы данных;

физическое проектирование базы данных;

– разработка приложений:

– проектирование транзакций;

– проектирование пользовательского интерфейса;

– реализация;

– загрузка данных;

– тестирование;
– эксплуатация и сопровождение:
– анализ функционирования и поддержка исходного варианта БД;
– адаптация, модернизация и поддержка переработанных вариантов.

Здесь представлен перечень основных этапов ЖЦБД. Естественно, что конкретное наполнение каждого этапа в значительной степени зависит от сложности разрабатываемого продукта. Для малых же приложений с небольшим количеством пользователей и в некоторых других частных случаях, наоборот, жизненный цикл базы данных может быть значительно упрощен в результате корректировки содержания отдельных этапов.

Планирование разработки базы данных

Содержание данного этапа — разработка стратегического плана, в процессе которой осуществляется предварительное планирование конкретной системы управления базами данных. Общая информационная модель, созданная на этом шаге, должна быть вновь проанализи- рована и, если нужно, изменена на последующем этапе, этапе разработки проекта реализации. Планирование разработки базы данных состоит в определении трех основных компонентов: объема работ, ресурсов и стоимости проекта. Планирование разработки базы данных должно быть связано с общей стратегией построения информационной системы организации.

Важной частью разработки стратегического плана является проверка осуществимости проекта, состоящая из нескольких частей.

Первая часть — проверка технологической осуществимости. Она состоит в выяснении вопроса, существует ли оборудование и программное обеспечение, удовлетворяющее инфор- мационным потребностям фирмы.

Вторая часть — проверка операционной осуществимости — выяснение наличия экспертов и персонала, необходимых для работы БД.

Рис 7. Этапы проектирования базы данных

Третья часть — проверка экономической целесообразности осуществления проекта. При исследовании этой проблемы весьма важно дать оценку ряду факторов, в том числе и таким:

– целесообразность совместного использования данных разными отделами
– величина риска, связанного с реализацией системы базы данных;
– ожидаемая выгода от внедрения подлежащих созданию приложений;
– время окупаемости внедренной БД;
– влияние системы управления БД на реализацию долговременных планов организации.

Планирование разработки баз данных также должно включать разработку стандартов, которые определяют, как будет осуществляться сбор данных, каким будет их формат, какая потребуется документация, как будет выполняться проектирование и реализация приложений.

Для поддержки планирования разработки базы данных может быть создана корпоративная модель данных, имеющая вид упрощенной ER-диаграммы.

Если результат проверки осуществимости проекта оказался положительным, можно перейти к определению требований к проекту.

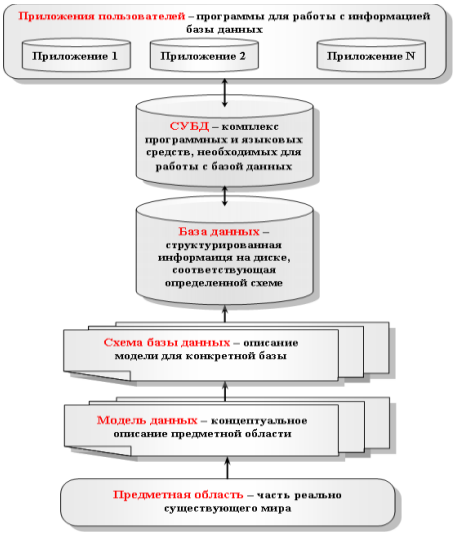

Рис.8. Области проектирования

Определение требований к системе

На данном этапе необходимо определить диапазон действия приложения базы данных, состав его пользователей и области применения.

Определение требований включает выбор целей БД, выяснение информационных потребностей различных отделов и руководителей фирмы и требований к оборудованию и программному обеспечению. При этом также требуется рассмотреть вопрос, следует ли создавать распределенную базу данных или же централизованную, и какие в рассматриваемой ситуации понадобятся коммуникационные средства. Написать краткий комментарий, описывающий цели системы.

Прежде чем приступать к проектированию приложения базы данных, важно установить границы исследуемой области и способы взаимодействия приложения с другими частями информационной системы организации. Эти границы должны охватывать не только текущих пользователей и области применения разрабатываемой системы, но и будущих пользователей и возможные области применения.

Сбор и анализ требований пользователей

Этот этап является предварительным этапом концептуального проектирования базы данных. Проектирование базы данных основано на информации о той части организации, которая будет обслуживаться базой данных.

Информационные потребности выясняются с помощью анкет, опросов менеджеров и работников фирмы, с помощью наблюдений за деятельностью предприятия, а также отчетов и форм, которыми фирма пользуется в текущий момент.

На данном этапе необходимо создать для себя модель движения важных материальных объектов и уяснить процесс документооборота. По каждому документу необходимо установить периодичность использования, определить данные, необходимые для выполнения выделенных функций (анализируя существующую и планируемую документацию, выясняют, как получа- ется каждый элемент данных, кем получается, где в дальнейшем используется, кем контроли- руется).

Самое пристальное внимание должно быть уделено дублированию информации, возможности появления ложной информации и причинам, которые ведут к их появлению. Также на этом этапе желательно представить общие параметры создаваемой базы.

В итоге собранная информация о каждой важной области применения приложения и пользовательской группе должна включать следующие компоненты: исходную и генерируемую документацию, подробные сведения о выполняемых транзакциях, а также список требований с указанием их приоритетов. На основании всей этой информации будут составлены спе- цификации требований пользователей в виде набора документов, описывающих деятельность предприятия с разных точек зрения.

Формализация собранной на этом этапе информации может быть повышена с помощью методов составления спецификаций требований, к числу которых относятся, например, техно- логия структурного анализа и проектирования, диаграммы потоков данных и графики "вход — процесс — выход".

Поскольку системы с неадекватной или неполной функциональностью будут лишь раз- дражать пользователей, а чрезмерно увеличенный набор функциональных возможностей вызо- вет существенное усложнение системы, важность этого этапа в процессе разработки БД сложно переоценить.

Проектирование базы данных

Полный цикл разработки базы данных включает концептуальное, логическое и физическое ее проектирование.

Основными целями проектирования базы данных являются:

– представление данных и связей между ними, необходимых для всех основных областей применения данного приложения и любых существующих групп его пользователей;

– создание модели данных, способной поддерживать выполнение любых требуемых транзакций обработки данных;

– разработка предварительного варианта проекта, структура которого позволяет удовлетворить требования, предъявляемые к производительности системы.

– В создании БД как модели ПрО выделяют:

– объектную (предметную) систему, представляющую фрагмент реального мира;

– информационную систему, описывающую некоторую объектную систему;

– датологическую систему, представляющую информационную систему с помощью данных.

Оптимальная модель данных должна удовлетворять таким критериям, как: структурная достоверность, простота, выразительность, отсутствие избыточности, расширяемость, целостность, способность к совместному использованию.

Концептуальное проектирование базы данных

Первая фаза процесса проектирования базы данных заключается в создании для анали- зируемой части предприятия *концептуальной модели данных.* Построение ее осуществляется в определенном порядке: в начале создаются подробные модели пользовательских представле- ний данных; затем они интегрируются в концептуальную модель данных. Концептуальное про- ектирование приводит к созданию *концептуальной схемы* базы данных.

Существует два основных подхода к проектированию систем баз данных: "нисходящий" и "восходящий".

При восходящем подходе, который применяется для проектирования простых баз данных с относительно небольшим количеством атрибутов, работа начинается с самого нижнего уровня — уровня определения атрибутов, которые на основе анализа существующих между ними связей группируются в отношения. Полученные отношения в дальнейшем подвергаются процессу нормализации, который приводит к созданию нормализованных взаимосвязанных таблиц, основанных на функциональных зависимостях между атрибутами.

Проектирование сложных баз данных с большим количеством атрибутов, поскольку установить среди атрибутов все существующие функциональные зависимости довольно затруднительно, осуществляется использованием нисходящего подхода. Начинается этот подход с разработки моделей данных, которые содержат несколько высокоуровневых сущностей и свя- зей, затем работа продолжается в виде серии нисходящих уточнений низкоуровневых сущно- стей, связей и относящихся к ним атрибутов.

Нисходящий подход демонстрируется в концепции модели "сущность -связь" (Entity-Relationship model — ER-модель) — самой популярной технологии высокоуровневого модели- рования данных, предложенной П. Ченом.

Модель "сущность — связь" относится к семантическим моделям. *Семантическое моделирование* данных, связанное со смысловым содержанием данных, независимо от их представления в ЭВМ, изначально возникло с целью повышения эффективности и точности проектирования баз данных. Методы семантического моделирования оказались применимы ко многим пользовательским проблемам и легко преобразуемы в сетевые, иерархические и реляцион- ные модели.

Помимо "нисходящего" и "восходящего" подходов, для проектирования баз данных могут применяться другие подходы, являющиеся некоторыми комбинациями указанных.

В построении *общей концептуальной модели данных* выделяют ряд этапов.

– Выделение локальных представлений, соответствующих обычно относительно независимым данным. Каждое такое представление проектируется как подзадача.
– Формулирование объектов, описывающих локальную предметную область проектируемой БД, и описание атрибутов, составляющих структуру каждого объекта.
– Выделение ключевых атрибутов.
– Спецификация связей между объектами. Удаление избыточных связей,
– Анализ и добавление не ключевых атрибутов.
– Объединение локальных представлений.

Построение концептуальной модели данных осуществляется на основе анализа описания предметной области на естественном языке, сделанного конечным пользователем. В процессе разработки концептуальная модель данных постоянно подвергается тестированию и про- верке на соответствие требованиям пользователей. Созданная концептуальная модель данных предприятия является источником информации для фазы логического проектирования базы данных.

Логическое проектирование базы данных

Цель второй фазы проектирования базы данных состоит в создании логической модели данных для исследуемой части предприятия.

Логическая модель, отражающая особенности представления о функционировании предприятия одновременно многих типов пользователей, называется *глобальной логической моделью данных.* Для создания глобальной логической модели данных предприятия можно вы- брать один из двух основных подходов — централизованный подход или подход на основе ин- теграции представлений.

Отправным моментом при *централизованном подходе,* который применим только для не слишком сложных баз данных, является образование единого списка требований путем объ- единения требований всех типов пользователей.

При использовании *метода интеграции представлений* осуществляется слияние отдельных локальных логических моделей данных, отражающих представления разных групп пользователей, в единую глобальную логическую модель данных всего предприятия.

В дальнейшем процесс проектирования БД должен опираться на определенную модель данных (реляционная, сетевая, иерархическая), которая определяется типом предполагаемой для реализации информационной системы СУБД. После чего сама концептуальная модель дан- ных уточняется и преобразуется в логическую модель данных.

В процессе разработки логическая модель данных должна постоянно подвергаться про- верке как на соответствие требованиям пользователей, так и на отсутствие избыточности дан- ных, способной вызвать в будущем аномалии обновления.

Построенная логическая модель данных в дальнейшем будет востребована на этапе фи- зического проектирования, а также на этапе эксплуатации и сопровождения уже готовой си- стемы, позволяя наглядно представить любые вносимые в базу данных изменения.

На этом шаге желательно создание следующих документов:

– набора подсхем;
– спецификаций для физического проектирования приложений;

– руководства по разработке программ (интерфейсы с пользователем и межпрограммные интерфейсы);

– руководства по сопровождению БД.

Концептуальное и логическое проектирование — это итеративные процессы, которые включают в себя ряд уточнений, продолжающиеся до тех пор, пока не будет получен наиболее соответствующий структуре предприятия продукт.

Физическое проектирование базы данных

Целью проектирования на данном этапе является создание описания СУБД ориентиро- ванной модели БД. Следует учитывать, что на этой стадии разработки возможны возвраты на более ранние этапы ЖЦБД. Например, решения, принимаемые на этапе физического проекти- рования с целью повышения производительности системы, могут привести к необходимости внести изменения в структуру логической модели данных.

Действия, выполняемые на этом этапе, слишком специфичны для различных моделей данных, поэтому их сложно обобщить. Остановимся на реляционной модели данных. В этом случае под физическим проектированием подразумевается:

– создание описания набора реляционных таблиц и ограничений для них на основе информации, представленной в глобальной логической модели данных;

– определение конкретных структур хранения данных и методов доступа к ним, обеспечивающих оптимальную производительность системы с базой данных;

– разработка средств защиты создаваемой системы.

Разработка приложений

Параллельно с проектированием системы базы данных выполняется разработка прило- жений. Главные составляющие данного процесса — это проектирование транзакций и пользо- вательского интерфейса.

Проектирование транзакций

Транзакции представляют некоторые события реального мира. Все транзакции должны обращаться к базе данных с той целью, чтобы хранимые в ней данные всегда гарантированно соответствовали текущей ситуации в реальном мире.

Транзакция может состоять из нескольких операций, однако с точки зрения пользователя эти операции представляют собой единое целое, переводящее базу данных из одного непротиворечивого состояния в другое. Реализация транзакций опирается на тот факт, что СУБД способна обеспечивать сохранность внесенных во время транзакции изменений в БД и непро- тиворечивость базы данных даже в случае возникновения сбоя. Для незавершенной транзакции СУБД гарантирует отмену всех внесенных ею изменений.

Проектирование транзакций заключается в определении:

– данных, которые используются транзакцией;

– функциональных характеристик транзакции;

– выходных данных, формируемых транзакцией;

– степени важности и интенсивности использования транзакции.

Существует три основных типа транзакций: извлечения, обновления и смешанные транзакции.

Транзакции извлечения используются для выборки некоторых данных с целью отображения их на экране или помещения в отчет.

Транзакции обновления используются для вставки новых, удаления старых или коррек- ции существующих записей базы данных.

Смешанные транзакции включают как операции извлечения, так и операции обновления данных.

Транзакции могут представлять собой сложные операции, которые раскладываются на несколько более простых операций, каждая из которых представляет собой отдельную транзак- цию.

Проектирование пользовательского интерфейса

Пользовательский интерфейс приложений базы данных является одним из важнейших компонентов системы. Интерфейс должен быть удобным и обеспечивать все функциональные возможности, предусмотренные в спецификациях требований пользователей.

Специалисты рекомендуют при проектировании пользовательского интерфейса использовать следующие основные элементы и их характеристики:

- легко узнаваемые названия полей;
- согласованную терминологию и сокращения;
- удобные средства перемещения курсора;
- средства исправления отдельных ошибочных символов и целых полей;
- средства вывода сообщений об ошибках при вводе недопустимых значений;
- особое выделение необязательных для ввода полей;
- средства вывода пояснительных сообщений с описанием полей;
- средства вывода сообщения об окончании заполнения формы.

Реализация

На данном этапе осуществляется физическая реализация базы данных и разработанных приложений, позволяющих пользователю формулировать требуемые запросы к БД и манипу- лировать данными в БД.

ХОД РАБОТЫ

Задание. Фирма выполняет ремонт компьютеров. Требуется разработать базу данных для хранения информации о выполнении ремонтных работ сотрудниками фирмы. При оформлении заказа фиксируется дата выполнения заказа, вид выполненной работы, исполнитель работы. Каждый исполнитель получает фиксированный процент вознаграждения от стоимости выполнения работы. Этот процент устанавливается персонально каждому исполнителю при за- ключении трудового договора между фирмой и работником. Исполнитель получает вознагражд- дение, которое вычисляется как Стоимость выполнения заказа * Фиксированный процент воз- награждения.

Анализ описания предметной области позволяет выделить набор данных, которые должны храниться в проектируемой базе данных:

- Фамилия исполнителя работы;
- Имя исполнителя работы;
- Отчество исполнителя работы;
- Процент вознаграждения (может различаться для разных исполнителей);
- Наименование работы;
- Стоимость работы (фиксированная для каждого наименования работы);
- Дата исполнения работы

Исходя из набора данных, которые должны храниться в БД, можно выделить два инфор- мационных объекта: Исполнитель (Фамилия, Имя, Отчество, Процент вознаграждения) и Ра- бота (Наименование, Стоимость работы). Определим соответствующие таблицы ИСПОЛНИ- ТЕЛИ и РАБОТЫ (рис.1). Ни одно из первоначально заданных полей таблицы ИСПОЛНИ- ТЕЛИ не определяет однозначно каждую запись таблицы, поэтому в таблицу введено поле Код исполнителя, значения в котором будут уникальными для каждого исполнителя. Это поле яв- ляется первичным ключом таблицы ИСПОЛНИТЕЛИ и будет определено как ключевое поле. С этой же целью в таблицу РАБОТЫ введен первичный ключ Код работы. Структурированная таблица (рис.2)

ИСПОЛНИТЕЛИ

Код исполнителя
Фамилия
Имя
Отчество
Процент вознаграждения

РАБОТЫ

Код работы
Наименование
Стоимость работы

Рисунок 1

В таблице ИСПОЛНИТЕЛИ будут храниться записи вида:

1	Иванов	Андрей	Петрович	20
2	Алексеев	Игорь	Андреевич	25

В таблице РАБОТЫ будут храниться записи вида:

1	Установка микропроцессора	100.00 р.
2	Замена вентилятора	50.00 р.

Рисунок 2

Практическая работа № 2. Проектирование реляционной схемы базы данных в среде СУБД.

Цель работы: приобретение практических навыков анализа предметной области, информационных задач и построения концептуальной модели базы данных.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Проектирование базы данных (БД) – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы (ИС). В результате её решения должны быть определены содержание БД, эффективный для всех её будущих пользователей способ органи- зации данных и инструментальные средства управления данными.

Основная цель процесса проектирования БД состоит в получении такого проекта, который удовлетворяет следующим требованиям:

– корректность схемы БД, т.е. база должна быть гомоморфным образом моделируемой предметной области (ПО), где каждому объекту предметной области соответствуют данные в памяти ЭВМ, а каждому процессу – адекватные процедуры обработки данных;

– обеспечение ограничений ;

– эффективность функционирования ;

– защита данных (от аппаратных и программных сбоев и несанкционированного доступа);

– простота и удобство эксплуатации;

– гибкость, т.е. возможность развития и адаптации к изменениям предметной области и/или требований пользователей.

Внимание! Базы данных всегда проектируются под конкретное назначение системы.

Техника проектирования баз данных может измениться в целом и в деталях в зависимости от назначения системы. Например, следует различать проектирование систем складирования данных и проектирование так называемых OLTP-систем, ориентируемых на оперативную обработку транзакций. В данном учебном курсе рассматривается проектирование баз данных в основном для OLTP-систем. Именно на таких системах исторически сложилась техника проек- тирования баз данных.

Этапы проектирования базы данных

Процесс проектирования включает в себя следующие этапы:

Концептуальное проектирование – это процедура конструирования информационной модели, не зависящей от каких-либо физических условий реализации.

Логическое проектирование – это процесс конструирования информационной модели на основе существующих моделей данных, не зависимо от используемой СУБД и других усло- вий физической реализации.

Физическое проектирование – это процедура создания описания конкретной реализации БД с описанием структуры хранения данных, методов доступа к данным.

Основными задачами концептуального проектирования являются определение предмет- ной области системы и формирование взгляда на ПО с позиций сообщества будущих пользова- телей БД, т.е. инфологической модели ПО.

Концептуальная модель ПО представляет собой описание структуры и динамики ПО, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД. Это описание выражается в терминах не отдельных объектов ПО и связей между ними, а их типов, связанных с ними ограничений целостности и тех процес- сов, которые приводят к переходу предметной области из одного состояния в другое.

Рассмотрим основные подходы к созданию концептуальной модели предметной области.

1. Функциональный подход к проектированию БД

Этот метод реализует принцип "от задач" и применяется тогда, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаётся рассматриваемая БД.

2. Предметный подход к проектированию БД

Предметный подход к проектированию БД применяется в тех случаях, когда у разработ- чиков есть чёткое представление о самой ПО и о том, какую именно информацию они хотели бы хранить в БД, а структура запросов не определена или определена не полностью. Тогда ос- новное внимание уделяется исследованию ПО и наиболее адекватному её отображению в БД с учётом самого широкого спектра информационных запросов к ней.

3. Проектирование с использованием метода "сущность-связь"

Метод "сущность–связь" (entity–relation, ER–method) является комбинацией двух преды- дущих и обладает достоинствами обоих. Этап инфологического проектирования начинается с моделирования ПО. Проектировщик разбивает её на ряд локальных областей, каждая из кото- рых (в идеале) включает в себя информацию, достаточную для обеспечения запросов отдельной группы будущих пользователей или решения отдельной задачи (подзадачи). Каждое локальное представление моделируется отдельно, затем они объединяются.

Выбор локального представления зависит от масштабов ПО. Обычно она разбивается на локальные области таким образом, чтобы каждая из них соответствовала отдельному внешнему приложению и содержала 6-7 сущностей.

Сущность – это объект, о котором в системе будет накапливаться информация. Сущности бывают как физически существующие (например, СОТРУДНИК или АВТОМОБИЛЬ), так и абстрактные (например, ЭКЗАМЕН или ДИАГНОЗ).

Для сущностей различают тип сущности и экземпляр. Тип характеризуется именем и списком свойств, а экземпляр – конкретными значениями свойств.

Типы сущностей можно классифицировать как сильные и слабые. Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных. Например, читатель библиотеки – сильная сущность, а абонемент этого читателя – слабая, которая зависит от наличия соответствующего читателя. Слабые сущности называют подчи- нёнными (дочерними), а сильные – базовыми (основными, родительскими).

Для каждой сущности выбираются свойства (атрибуты). Различают:

– Идентифицирующие и описательные атрибуты. Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа и являются потенциальными ключами. Они позволяют однозначно распознавать экземпляры сущности. Из потенциальных клю- чей выбирается один первичный ключ (ПК). В качестве ПК обычно выбирается потенциальный

ключ, по которому чаще происходит обращение к экземплярам записи. Кроме того, ПК должен включать в свой состав минимально необходимое для идентификации количество атрибутов. Остальные атрибуты называются описательными и заключают в себе интересующие свойства сущности.

– Составные и простые атрибуты. Простой атрибут состоит из одного компонента, его значение неделимо. Составной атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных (например, ФИО или адрес). Решение о том, ис- пользовать составной атрибут или разбивать его на компоненты, зависит от характера его об- работки и формата пользовательского представления этого атрибута.

– Однозначные и многозначные атрибуты (могут иметь соответственно одно или много значений для каждого экземпляра сущности).

– Основные и производные атрибуты. Значение основного атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов (например, возраст студента вычисляется на основе даты его рождения и текущей даты). Спецификация атрибута состоит из его названия, указания типа данных и описания огра- ничений целостности – множества значений (или домена), которые может принимать данный атрибут.

Далее осуществляется спецификация связей внутри локального представления. Связи могут иметь различный содержательный смысл (семантику). Различают связи типа "сущность- сущность", "сущность-атрибут" и "атрибут-атрибут" для отношений между атрибутами, кото- рые характеризуют одну и ту же сущность или одну и ту же связь типа "сущность-сущность".

Каждая связь характеризуется именем, обязательностью, типом и степенью. Различают факультативные и обязательные связи. Если вновь порождённый объект одного типа оказывается по необходимости связанным с объектом другого типа, то между этими типами объектов существует обязательная связь (обозначается двойной линией). Иначе связь является факультативной.

По типу различают множественные связи "один к одному" (1:1), "один ко многим" (1:N) и "многие ко многим" (M:N).

Степень связи определяется количеством сущностей, которые охвачены данной связью. Пример бинарной связи – связь между отделом и сотрудниками, которые в нём работают. При- мером тернарной связи является связь типа экзамен между сущностями ДИСЦИПЛИНА, СТУ- ДЕНТ, ПРЕПОДАВАТЕЛЬ. Из последнего примера видно, что связь также может иметь атри- буты (в данном случае это Дата проведения и Оценка). Пример ER–диаграммы с указанием сущностей, их атрибутов и связей приведен на рис. 3.

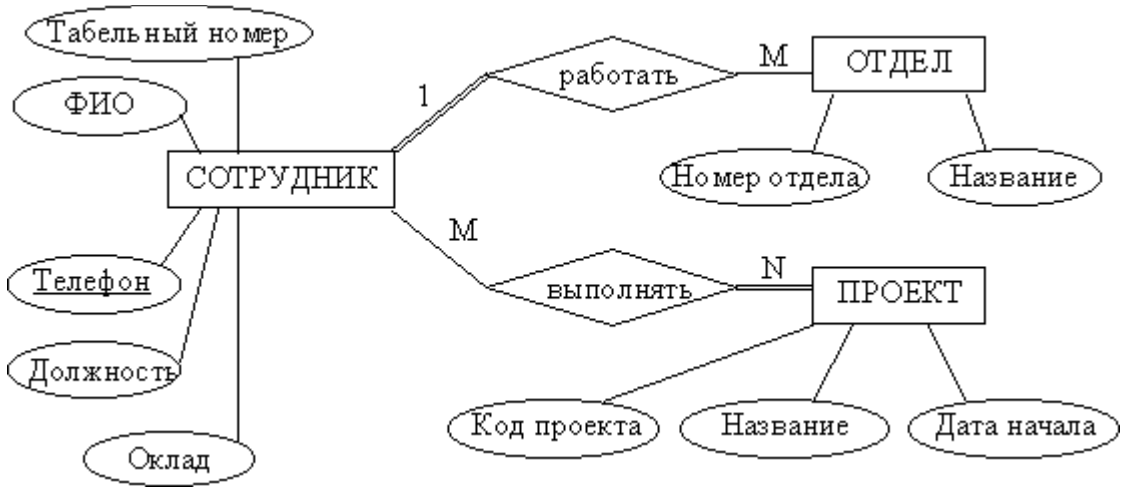

Рисунок 3

Анализ предметной области

В качестве примера возьмем базу данных компании, которая занимается издательской деятельностью.

База данных создаётся для информационного обслуживания редакторов, менеджеров и других сотрудников компании. БД должна содержать данные о сотрудниках компании, книгах, авторах, финансовом состоянии компании и предоставлять возможность получать разнообраз- ные отчёты.

В соответствии с предметной областью система строится с учётом следующих особенностей:

– каждая книга издаётся в рамках контракта;
– книга может быть написана несколькими авторами;
– контракт подписывается одним менеджером и всеми авторами книги;
– каждый автор может написать несколько книг (по разным контрактам);
– порядок, в котором авторы указаны на обложке, влияет на размер гонорара;
– если сотрудник является редактором, то он может работать одновременно над несколькими книгами;
– у каждой книги может быть несколько редакторов, один из них – ответственный редактор;
– каждый заказ оформляется на одного заказчика;
– в заказе на покупку может быть перечислено несколько книг.

Выделим базовые сущности этой предметной области:

1. Сотрудники компании. Атрибуты сотрудников – ФИО, табельный номер, пол, дата рождения, паспортные данные, ИНН, должность, оклад, домашний адрес и телефоны. Для редакторов необходимо хранить сведения о редактируемых книгах; для менеджеров – сведения о подписанных контрактах.

2. Авторы. Атрибуты авторов – ФИО, ИНН (индивидуальный номер налогоплательщика), паспортные данные, домашний адрес, телефоны. Для авторов необходимо хранить сведения о написанных книгах.

3. Книги. Атрибуты книги – авторы, название, тираж, дата выхода, цена одного экземпляра, общие затраты на издание, авторский гонорар.

4. Контракты будем рассматривать как связь между авторами, книгами и менеджерами. Атрибуты контракта – номер, дата подписания и участники.

5. Для отражения финансового положения компании в системе нужно учитывать заказы на книги. Для заказа необходимо хранить номер заказа, заказчика, адрес заказчика, дату по- ступления заказа, дату его выполнения, список заказанных книг с указанием количества экзем- пляров.

ER–диаграмма издательской компании приведена на рис. 4 (базовые сущности на рисун- ках выделены полужирным шрифтом).

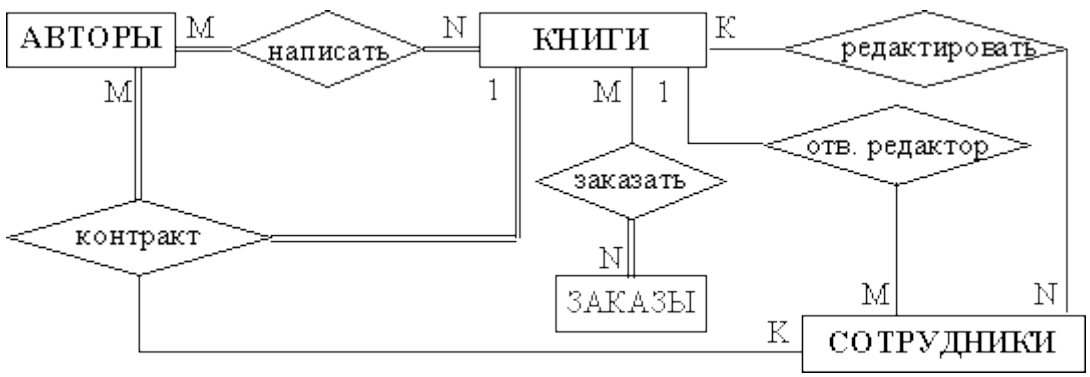

Рисунок 4

Анализ информационных задач и круга пользователей системы

Система создаётся для обслуживания следующих групп пользователей:

– администрация (дирекция);
– менеджеры;
– редакторы;
– сотрудники компании, обслуживающие заказы.

Определим границы информационной поддержки пользователей:

1) Функциональные возможности:

– ведение БД (запись, чтение, модификация, удаление в архив);
– обеспечение логической непротиворечивости БД;
– обеспечение защиты данных от несанкционированного или случайного доступа (определение прав доступа);
– реализация наиболее часто встречающихся запросов в готовом виде;
– предоставление возможности сформировать произвольный запрос на языке манипулирования данными.

2) Готовые запросы:

- получение списка всех текущих проектов (книг, находящихся в печати и в продаже);
– получение списка редакторов, работающих над книгами;
– получение полной информации о книге (проекте);
– получение сведений о конкретном авторе (с перечнем всех книг);
– получение информации о продажах (по одному или по всем проектам);
– определение общей прибыли от продаж по текущим проектам;
– определение размера гонорара автора по конкретному проекту.

Задание: по заданному описанию предметной области построить концептуальную модель базы данных :

– Выделите типы сущностей;
– Выделите типы связей и определите для них показатели кардинальности и степень участия сторон;
– Выделите атрибуты и свяжите их типами сущностей и связей;
– Определите потенциальные и первичные ключи сущностей;
– Нарисуйте ER-диаграмму и проанализируйте информационные задачи и группы пользователей.

Вариант1.

Задача – организация учебного процесса в вузе:

* Студенты: паспортные данные, адрес, дата зачисления, номер приказа, факультет, группа, является ли старостой, кафедра (специализация), изучаемые (изученные) предметы, оценки, задолженности, стипендия.
* Учебные курсы: название, факультет(ы), групп(ы), кафедра, семестр(ы), форма отчётности, число часов.
* Преподаватели: паспортные данные, адрес, телефон, фотография, кафедра, должность, учёная степень, начальник (зав. кафедрой), предмет(ы), число ставок, зарплата.

Вариант 2.

Учет и выдача книг в библиотеке вуза:

* Книги: авторы, название, раздел УДК, раздел (техническая, общественно-политическая и т.п.), место и год издания, издательство, количество страниц, иллюстрированность, цена, дата покупки, номер сопроводительного документа (чек, счёт/накладная), вид издания (книги, учебники, брошюры, периодические издания), инвентарный номер (есть только для

книг и некоторых учебников), длительность использования читателями (год, две недели, день), электронная версия книги или ее реферата (отсканированный текст).
* Читатели: номер читательского билета, ФИО, год рождения, адрес, дата записи, вид (студент, аспирант, преподаватель, сотрудник), курс, номер группы, названия взятых книг и даты их вы- дачи.

Вариант 3.

Отдел кадров некоторой компании.

* Сотрудники: ФИО, паспортные данные, фотография, дом. и моб. телефоны, отдел, комната, раб. телефоны (в т.ч. местный), подчинённые сотрудники, должность, тип(ы) работы, задание(я), проект(ы), размер зарплаты, форма зарплаты (почасовая, фиксированная).
* Отделы: название, комната, телефон(ы), начальник, размер финансирования, число сотрудников.
* Проекты: название, дата начала, дата окончания, размер финансирования, тип финансирова- ния (периодический, разовый), задачи и их исполнители, структура затрат и статьи расходов.

Вариант 4.

Отдел поставок некоторого предприятия:

* Поставщики: название компании, ФИО контактного лица, расчётный счёт в банке, телефон, факс, поставляемое оборудование (материалы), даты поставок (по договорам и реальные), метод и стоимость доставки.
* Сырьё: тип, марка, минимальный запас на складе, время задержки, цена, продукты, при про- изводстве которых используется, потребляемые объемы (необходимый, реальный, на единицу продукции).

Практическая работа № 3. Приведение БД к нормальной форме 3НФ

Цель работы: приведение базы данных к нормальной форме.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Теория нормализации отношений работает с 5 нормальными формами таблиц. Каждой нормальной форме соответствует некоторый определенный набор ограничений. Каждая после- дующая форма должна отвечать требованиям предыдущих плюс некоторые дополнительные требования.

Первая нормальная форма (1НФ), рис.5.

Таблица, находящаяся в первой нормальной форме должна отвечать следующим требованиям:

- таблица не должна иметь повторяющихся записей;
- в таблице должны отсутствовать повторяющиеся группы полей.

Код сотрудника
Имя
Фамилия
Отчество
Дата рождения
Адрес
Телефон
Должность
Разряд
Зарплата
Рейтинг
Дата приема
Дата увольнения

Рисунок 5

Вторая нормальная форма (2НФ), рис. 6.

Таблица, находящаяся во второй нормальной форме должна отвечать всем требованиям 1НФ, а также любое неключевое поле однозначно идентифицируется полным набором ключе- вых полей 2НФ применяется к таблицам, которые имеют составной ключ (см.рис.2).

Код физиче- ского лица
Имя
Фамилия
Отчество
Дата рождения
Адрес
Телефон

Код сотрудника
Код физиче- ского лица
Должность
Разряд
Зарплата
Дата приема
Дата увольне- ния

Рисунок 6

Третья нормальная форма (3НФ), рис 7

Таблица, находящаяся в третьей форме должна отвечать всем требованиям 2НФ, а также ни одно из неключевых полей не идентифицируется при помощи другого неключевого поля. Другими словами в таблице нет полей, которые не зависят от ключа. На практике третья нормальная форма схем отношений в большинстве случаев достаточна, и приведение к ней процесс проектирования реляционных баз данных обычно заканчивается.

Рисунок 7

Основные понятия реляционной модели данных являются:

- *Домен*. Наименьшая единица данных реляционной модели – это отдельное атомарное (неразложимое) для данной модели значение данных. Доменом называется множество атомарных значений одного и того же типа.
- *Тип данных* в реляционной модели данных полностью эквивалентно соответствующему понятию в алгоритмических языках. Все СУБД поддерживают следующие типы дан- ных: целочисленные, вещественные, строковые, специальные типы данных для времен- ных величин (дата и / или время).
- *Атрибут*. Столбцы отношения называют атрибутами, им присваиваются имена, по которым к ним затем производится обращение.
- *Ключ*. Простой ключ – ключ, содержащий только один атрибут. Составной ключ – это ключ, состоящий из несколько атрибутов.

Чтобы информация, хранящаяся в базе данных, была однозначной и непротиворечивой, в реляционной модели устанавливаются некоторые ограничительные условия. Ограничительные условия – это правила, определяющие возможные значения данных. Они обеспечивают логи- ческую основу для поддержания корректных значений данных в базе. Такие ограничения це- лостности позволяют свести к минимуму ошибки, возникающие при обновлении и обработки данных.

Реляционная база данных представляет собой совокупность отношений, содержащих всю информацию, которая должна храниться в базе данных. Однако пользователи могут восприни- мать такую базу данных как совокупность таблиц. Таким образом, реляционную базу данных можно рассматривать как хранилище данных, содержащих набор двухмерных таблиц. Набор

средств управления подобным хранилищем называется реляционной системой управления ба- зами данных. Она может содержать утилиты, приложения, службы, библиотеки и другие при- ложения.

Практическая работа № 4 Создание базы данных в среде разработки

Цель работы: изучить принципы создания базы данных при помощи SQL - операторов

ХОД РАБОТЫ:

Язык SQL стал фактически стандартным языком доступа к базам данных. Все СУБД, претендующие на название "реляционные", реализуют тот или иной диалект SQL. Многие нереляционные системы также имеют в настоящее время средства доступа к реляционным данным. Целью стандартизации является переносимость приложений между различными СУБД.

Нужно заметить, что в настоящее время, ни одна система не реализует стандарт SQL в полном объеме. Кроме того, во всех диалектах языка имеются возможности, не являющиеся стандартными. Таким образом, можно сказать, что каждый диалект - это надмножество некоторого подмножества стандарта SQL. Это затрудняет переносимость приложений, разработанных для одних СУБД в другие СУБД.

Язык SQL оперирует терминами, несколько отличающимися от терминов реляционной теории, например, вместо "отношений" используются "таблицы", вместо "кортежей" - "строки", вместо "атрибутов" - "колонки" или "столбцы".

Стандарт языка SQL, хотя и основан на реляционной теории, но во многих местах отходит он нее. Например, отношение в реляционной модели данных не допускает наличия одинаковых кортежей, а таблицы в терминологии SQL могут иметь одинаковые строки. Имеются и другие отличия.

Язык SQL является реляционно полным. Это означает, что любой оператор реляционной алгебры может быть выражен подходящим оператором SQL.

Операторы SQL

Основу языка SQL составляют операторы, условно разбитые не несколько групп по выполняемым функциям.

Можно выделить следующие группы операторов (перечислены не все операторы SQL):

Операторы DDL (Data Definition Language) - операторы определения объектов базы данных

CREATE SCHEMA - создать схему базы данных
DROP SHEMA - удалить схему базы данных
CREATE TABLE - создать таблицу
ALTER TABLE - изменить таблицу
DROP TABLE - удалить таблицу
CREATE DOMAIN - создать домен
ALTER DOMAIN - изменить домен
DROP DOMAIN - удалить домен
CREATE COLLATION - создать последовательность
DROP COLLATION - удалить последовательность
CREATE VIEW - создать представление
DROP VIEW - удалить представление
Операторы DML (Data Manipulation Language) - операторы манипулирования данными
SELECT - отобрать строки из таблиц
INSERT - добавить строки в таблицу
UPDATE - изменить строки в таблице
DELETE - удалить строки в таблице
COMMIT - зафиксировать внесенные изменения
ROLLBACK - откатить внесенные изменения
Операторы защиты и управления данными

CREATE ASSERTION - создать ограничение

DROP ASSERTION - удалить ограничение

GRANT - предоставить привилегии пользователю или приложению на манипулирование объектами

REVOKE - отменить привилегии пользователя или приложения

Кроме того, есть группы операторов установки параметров сеанса, получения информации о базе данных, операторы статического SQL, операторы динамического SQL.

Наиболее важными для пользователя являются операторы манипулирования данными (DML).

Примеры использования операторов манипулирования данными

INSERT - вставка строк в таблицу

Задание 1. Вставка одной строки в таблицу:

```
INSERT INTO
  P (PNUM, PNAME)
  VALUES (4, "Иванов");
```

Задание 2. Вставка в таблицу нескольких строк, выбранных из другой таблицы (в таблицу TMP_TABLE вставляются данные о поставщиках из таблицы P, имеющие номера, большие 2):

```
INSERT INTO
  TMP_TABLE (PNUM,
  PNAME) SELECT PNUM,
  PNAME FROM P
    WHERE P.PNUM>2;
```

UPDATE - обновление строк в таблице

Задание 3. Обновление нескольких строк в таблице:

```
UPDATE P
  SET PNAME = "Пушников"
  WHERE P.PNUM = 1;
```

DELETE - удаление строк в таблице

Пример 4. Удаление нескольких строк в таблице:

```
DELETE FROM P
  WHERE P.PNUM = 1;
```

Задание 5. Удаление всех строк в таблице:

```
DELETE FROM P;
```

Примеры использования оператора SELECT

Оператор SELECT является фактически самым важным для пользователя и самым сложным оператором SQL. Он предназначен для выборки данных из таблиц, т.е. он, собственно, и реализует одно их основных назначение базы данных - предоставлять информацию пользователю.

Оператор SELECT всегда выполняется над некоторыми таблицами, входящими в базу данных.

Замечание. На самом деле в базах данных могут быть не только постоянно хранимые таблицы, а также временные таблицы и так называемые представления. Представления - это просто хранящиеся в базе данные SELECT-выражения. С точки зрения пользователей представ- ления - это таблица, которая не хранится постоянно в базе данных, а "возникает" в момент об- ращения к ней. С точки зрения оператора SELECT и постоянно хранимые таблицы, и времен- ные таблицы и представления выглядят совершенно одинаково. Конечно, при реальном выпол- нении оператора SELECT системой учитываются различия между хранимыми таблицами и представлениями, но эти различия *скрыты* от пользователя.

Результатом выполнения оператора SELECT всегда является таблица. Таким образом, по результатам действий оператор SELECT похож на операторы реляционной алгебры. Любой оператор реляционной алгебры может быть выражен подходящим образом сформулированным оператором SELECT. Сложность оператора SELECT определяется тем, что он содержит в себе

все возможности реляционной алгебры, а также дополнительные возможности, которых в реляционной алгебре нет.

Отбор данных из одной таблицы

Задание 6. Выбрать все данные из таблицы поставщиков (ключевые слова ***SELECT***… ***FROM***…):

```
SELECT *
  FROM P;
```

Замечание. В результате получим новую таблицу, содержащую полную копию данных из исходной таблицы P.

Задание 7. Выбрать все строки из таблицы поставщиков, удовлетворяющих некоторому условию (ключевое слово ***WHERE***…):

```
SELECT *
  FROM P
  WHERE P.PNUM > 2;
```

Замечание. В качестве условия в разделе WHERE можно использовать сложные логические выражения, использующие поля таблиц, константы, сравнения (>, <, = и т.д.), скобки, союзы AND и OR, отрицание NOT.

Практическая работа № 5. Организация локальной сети. Настройка локальной сети.

Цель работы: изучить способы организации локальных сетей и их настройку для работы с базой данных.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Многие коммерческие предприятия имеют данные, доступ к которым предлагается пользователям вне компании - другим предприятиям, покупателям или производителям Напри- мер, предприятие может предлагать потенциальным покупателям доступ к дополнительной ин- формации о продуктах предприятия в надежде увеличить сбыт Должны быть учтены и нужды служащих предприятия Например, доступной может быть служебная информация о графиках работы и отпусков, обучении персонала, политике компании и т п Подобная база данных может быть создана и сделана легко доступной для пользователей средствами SQL и Internet

Прикладная часть клиент-серверного приложения (back-end application) состоит из сер- вера базы данных, источников данных и соответствующего промежуточного программного обеспечения, используемого для подключения приложения к Web или удаленной базе данных по локальной сети.

Еще раз напомним, что к наиболее распространенным серверам баз данных относят Oracle, Informix, Sybase, Microsoft SQL Server и Borland InterBase Именно с сервера начинается разворачивание приложения баз данных либо на все предприятие в рамках локальной сети (LAN) или сети intranet предприятия, либо в Internet Разворачивание (porting) представляет со- бой процесс внедрения приложения в среду, доступную пользователям Сервер базы данных должен быть установлен на рабочем месте администратора базы данных, который, в свою оче- редь, должен понимать и производственные потребности предприятия, и требования самого приложения

Промежуточное программное обеспечение приложения состоит из сервера Web и средств, позволяющих подключить сервер Web к серверу базы данных Главной целью в данном случае является наличие в Web приложения, предоставляющего доступ к корпоративным дан- ным

Интерфейсная часть

Интерфейсная часть приложения (front-end application) - это та часть приложения, с которой имеет дело пользователь Интерфейсная часть приложения может быть коммерческим продуктом какой-нибудь компании, производящей программное обеспечение на продажу, либо продуктом, разработанным внутри предприятия с применением различных программных средств.

До того, как на рынке сложилось имеющееся сегодня разнообразие приложений, предлагающих интерфейс пользователя базы данных, пользователю необходимо было уметь программировать на языках типа C HTML или любом другом из множества процедурных языков программирования, с помощью которых разрабатывались приложения для Web Языки типа ANSI C, COBOL, FORTRAN или Pascal использовались для разработки интерфейсной части внутри предприятия, и соответствующий интерфейс пользователя был, как правило, текстовым Сегодня большинство новых приложений интерфейсной части предлагают графический поль- зовательский интерфейс (GUI)

Интерфейсная часть приложения призвана обеспечить пользователю простоту доступа к базе данных и работы с ней Внутренние процессы программный код и происходящие в ней события должны быть незаметными для пользователя Интерфейсная часть приложения должна быть разработана для того чтобы по возможности избавить пользователя от необходимости те- ряться в догадках и интуитивно чувствовать систему в целом Новые технологии позволяют сделать приложения более понятными и простыми в применении что дает пользователю воз- можность сосредоточиться на решении своих конкретных задач повышая в конечном итоге эф- фективность своего труда.

Имеющиеся на сегодня средства разработки приложений достаточно просты в примене- нии и объектно-ориентированны, что достигается использованием в них пиктограмм, возмож- ностей перетаскивания объектов с помощью мыши, а также различных мастеров, автоматиче- ски генерирующих объекты с заданными свойствами. Среди наиболее популярных средств раз- работки Web-приложений следует упомянуть C++Builder, IntraBuilder фирмы Borland и Visual J++, C++ фирмы Microsoft. Для разработки программ, предназначенных для работы в рамках локальной сети предприятия, используют PowerBuilder фирмы Powersoft, Developer/2000 фирмы Oracle Corporation, Visual Basic фирмы Microsoft и Delphi фирмы Borland.

Прикладная часть располагается на сервере, там же размещается и сама база данных. Пользователями прикладной части являются разработчики базы данных, программисты, администраторы базы данных, системные администраторы и системные аналитики. Интерфейсная часть приложения размещается на машинах-клиентах, которыми обычно являются персональные компьютеры конечных пользователей. Интерфейсная часть приложения рассчитана на самую широкую аудиторию пользователей, включающую операторов ввода данных, бухгалтеров и т. д. Пользователь должен иметь возможность доступа к базе данных по сети-и такая сеть может быть как локальной (LAN), так и глобальной (WAN). Для предоставления пользователю такой возможности используется промежуточное программное обеспечение (например, драй- вер ODBC).

Удаленный доступ к базе данных

База данных может быть локальной, и тогда вы имеете возможность подключится к ней непосредственно. Но, как правило, пользователю требуется доступ к базе данных, которая нахо- дится на некотором удалении от его системы. Удаленная база данных - это некоторая база дан- ных, не являющаяся локальной, т. е. расположенной на том сервере, к которому вы подключены в данный момент, и предполагающая для доступа к ней использование сети и определенных сетевых протоколов.

Доступ к удаленной базе данных можно осуществить несколькими способами. Говоря в общем, доступ к удаленной базе данных осуществляется с помощью Подключения к сети или Internet посредством использования промежуточного программного обеспечения (например, стандартных средств ODBC).

Локальный сервер базы данных и главный локальный сервер часто оказываются одним и тем же объектом, поскольку база данных, как правило, размещается на главном локальном сервере. Но подключиться к удаленной базе данных с главного локального сервера можно и без подключения к локальной базе данных. Для конечного пользователя чаще всего предлагается подключение к удаленной базе данных средствами интерфейсного приложения. Во всех этих случаях используется передача запросов к базе данных по сети.

Технология ODBC (Open Database Connectivity - открытый интерфейс доступа к базам данных) обеспечивает возможность доступа к удаленным базам данных с помощью подходящего драйвера. Драйвер ODBC используется интерфейсным приложением для получения доступа к прикладной части базы данных для взаимодействия с данными нижнего уровня. Для доступа к удаленной базе данных, кроме того, может понадобиться и сетевой драйвер. Приложение вызывает функции ODBC, а соответствующий драйвер обеспечивает загрузку драйвера ODBC. Драйвер ODBC обрабатывает вызов функции, пересылает запрос к базе данных и возвращает результат этого запроса. На сегодня ODBC является стандартом, используемым целым рядом продуктов, в частности, PowerBuilder, FoxPro, Visual C++, Visual Basic, Delphi, Microsoft Access и многими другими.

Как часть ODBC, любой производитель систем управления базами данных предлагает со своими базами данных программный интерфейс приложения (API). Для примера из таких предлагаемых продуктов можно отметить Open Call Interface (OCI) фирмы Oracle и SQLGateway или SQLRouter фирмы Centura.

Другие интерфейсы доступа к данным

В дополнение к драйверу ODBC многие производители систем управления базами данных предлагают свое программное обеспечение, предназначенное для организации доступа к

удаленным базам данных. Каждый из таких продуктов оказывается специфическим для системы управления базами данных конкретного производителя и, вообще говоря, не предполагает переносимости на другие типы серверов баз данных.

Oracle Corporation для организации доступа к удаленным базам данных предлагает свой продукт под именем Net8. Net8 можно использовать практически с любыми сетевыми протоко- лами, в частности, TCP/IP, OSI, SPX/IPX и многими другими. Кроме того, Net8 может работать под управлением почти любой из наиболее распространенных операционных систем.

Sybase Incorporated предлагает свой продукт под именем Open Client/C Developers Kit, поддерживающий продукты других производителей, в частности Net8 фирмы Oracle.

Доступ к удаленным базам данных с помощью интерфейса Web

Доступ к удаленным базам данных посредством интерфейса Web подобен доступу к ба- зам данных по локальной сети Главное отличие состоит в том, что в Web все запросы к базе данных направляются через Web-сервер.

Конечный пользователь инициирует доступ к удаленной базе данных с помощью броузера Web. Броузер Web используется для связывания с заданным URL или адресом IP в Internet, соответствующим нужному серверу Web. Сервер Web, проверив имя и пароль пользователя, пересылает пользовательский запрос базе данных, которая, в свою очередь, тоже может потребовать проверки имени и пароля. Затем сервер базы данных вернет результаты запроса серверу Web, а последний отобразит эти результаты в окне пользовательского броузера Web. Несанкционированный доступ к конкретному серверу может пресекаться с помощью бранд- мауэра (firewall).

Брандмауэр (firewall) - это аппаратно-программная система межсетевой защиты от несанкционированного доступа к серверу. Для зашиты от несанкционированного доступа к серверу может использоваться как одна, так и несколько таких систем. Точно также одна или несколько систем защиты могут использоваться для управления доступом к серверу базы данных и к самой базе данных.

При пересылке информации по Web следует принять все возможные меры безопасности на всех уровнях. К таким уровням можно отнести сервер Web, главный локальный сервер и удаленную базу данных Частные данные, такие как идентификационные номера служащих, всегда должны быть защищены от доступа к ним случайных лиц и не должны распространяться в Web.

SQL и Internet

SQL можно использовать в рамках приложений, создаваемых средствами C или COBOL. Точно так же SQL можно использовать и в Internet-приложениях, создаваемых средствами та- ких языков программирования, как Java. Текст HTML тоже можно транслировать в запрос SQL, чтобы потом с помощью интерфейсного приложения Web переслать этот запрос удаленной базе данных. Возвращенный базой данных результат затем транслируется обратно в текст HTML и отображается броузером Web на экране пользователя, пославшего запрос. В следующих разде- лах использование SQL в Internet обсуждается подробнее.

С изобретением и распространением Internet по всему миру данные стали доступными покупателям и производителям в любой стране. Такие данные обычно бывают доступными только для чтения с помощью соответствующего интерфейсного приложения.

Предназначенные для покупателей данные могут состоять из общей информации для покупателей, информации о конкретных продуктах, бланков заказов, информации о текущих и выполненных ранее заказах и т.д. Частная информация, например, информация о корпоратив- ной стратегии и о служащих компании доступной быть не должна.

Наличие информационной странички в Web стало почти обязательным для компаний, стремящихся успешно конкурировать с другими в своем бизнесе. Страничка Web оказывается весьма эффективным средством информирования большого числа потенциальных покупателей об услугах, продуктах и других аспектах деятельности компании, не требуя при этом чрезмер- ных затрат.

Предоставление доступа к данным служащим и привилегированным клиентам

С помощью Internet или сети intranet компании базу данных можно сделать доступной для служащих этой компании или ее клиентов. Использование технологий Internet оказывается весьма удобным для информирования служащих о политике компании, преимуществах работы в ней, обучающих программах и т. п.

Интерфейсные приложения Web, использующие SQL

Имеется целый ряд приложений, обеспечивающих доступ к базам данных. Многие из таких приложений предлагают графический интерфейс пользователя, так что пользователю даже нет необходимости понимать SQL, чтобы составить запрос к базе данных. В таких прило- жениях пользователю предлагается указывать и щелкать мышью на объектах, представляющих таблицы, манипулировать данными этих объектов, задавать критерии отбора возвращаемых данных и т. д. Такие приложения часто разрабатываются и настраиваются в полном соответ- ствии с конкретными требованиями конкретной компании.

SQL и intranet

IBM изначально создавала SQL для доступа к базам данных, размещенным на мэйнфрей- мах, с клиентских машин пользователей. Пользователи при этом связывались с мэйнфреймами по локальной сети. Позже SQL стал стандартным языком коммуникации пользователей с ба- зами данных. Intranet, по сути, является миниатюрным аналогом Internet. Основным различием между ними является то, что intranet предна-шачлегся для использования внутри некоторой ор- ганизации, а Internet открыта для доступа всем. Пользовательский (клиентский) интерфейс в intranet остается тем же, что и в модели клиент/сервер. Запросы SQL направляются базе данных сервером Web с использованием соответствующего языка (например, HTML).

Безопасность в рамках базы данных значительно выше, чем в Internet. Поэтому всегда используйте средства безопасности, предлагаемые вашим сервером базы данных.

ХОД РАБОТЫ

Задание 1. Войдите в Internet и ознакомьтесь с информационными страницами несколь- ких из представленных там компаний. Если ваша компания тоже имеет информационную стра- ницу в Web, сравните ее с информационными страницами конкурентов. Ответьте для себя на следующие вопросы в отношении просмотренных страниц.

а. Открывается ли страница быстро или ее открытие тормозится наличием слишком большого числа графических изображений?

б. Интересно ли читать представленную на странице информацию?

в. Получили ли вы в результате чтения имеющейся на странице информации представ-ление о предлагаемых компанией услугах и продуктах и о компании в целом?

г. Если на странице предлагается доступ к некоторой базе данных, то достаточно ли быстро осуществляется такой доступ?

д. Можно ли сделать вывод об использовании на данной странице Web каких-либо средств безопасности?

Задание 2. Если в вашей компании используется intranet, войдите в сеть и посмотрите, какая информация о компании там представлена. Доступна ли там какая-нибудь база данных? Если да, то кто является производителем соответствующей системы управления базами дан-ных? Какого типа интерфейсные приложения предлагаются при этом конечному пользова-телю?

Практическая работа №6. Установка и настройка SQl-сервера.

Цель работы: изучить этапы установки и настройка SQl-сервера

ХОД РАБОТЫ

Для установки нам потребуется дистрибутив SQL Server Express with Tools с сайта Microsoft. Данная сборка уже включает в себя графическое средство управления — среду SQL Server Management Studio Express.

1. Запустить установщик с правами администратора на данном компьютере. В разделе «Планирование» нажать пункт «Средство проверки конфигурации»: Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно пройдены. Если будут обнаружены ка- кие-то проблемы, то необходимо их устранить и запустить повторную проверку кнопкой
«Включить заново». Затем закрыть данное окно кнопкой «ОК».
2. Нажать на раздел «Установка» и затем пункт «Новая установка изолированного SQL Server или добавление компонентов …».
3. Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно пройдены. Если будут обнаружены какие-то проблемы, то необходимо их устранить и запустить повторную проверку кнопкой «Включить заново». Затем нажать кнопку «ОК».
4. Ввести приобретенный ключ продукта (для бесплатной версии не требуется) и нажать кнопку «Далее». Прочитать лицензию, установить галочку и нажать кнопку «Далее». Нажать кнопку «Установить».
5. Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно пройдены. Если будут обнаружены какие-то проблемы, то необходимо их устранить и запустить повторную проверку кнопкой «Включить заново». Затем нажать кнопку «Далее»

Примечание. Если появится предупреждение в строке «Брандмауэр Windows», то его можно проигнорировать – оно просто акцентирует Ваше внимание на том, что потребуется до- полнительная настройка «Брандмауэра Windows» для доступа к SQL Server с других компью- теров (см. ниже).

6. Выбрать компоненты для установки (можно воспользоваться кнопкой «Выделить все»), и нажать кнопку «Далее»:

Выбрать опцию «Экземпляр по умолчанию» и нажать кнопку
«Далее» Нажать кнопку «Далее»

Выбрать опции, как показано на рисунке, и перейти на закладку «Параметры сортировки»:

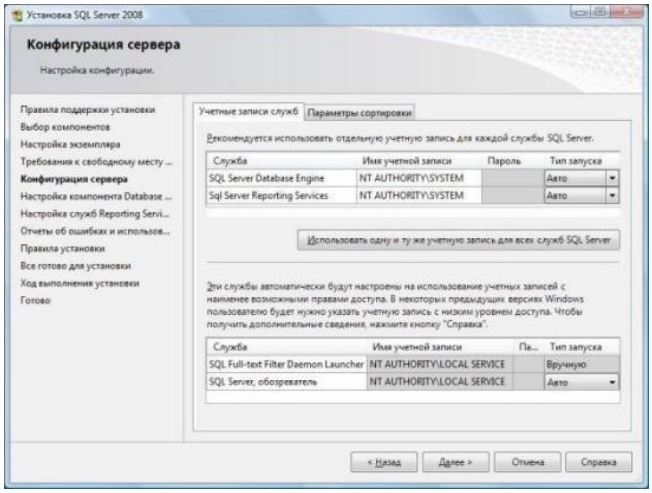

Рисунок 8

Выбрать опции, как показано на рис. 8, и нажать кнопку «Далее»:

Примечание: в данном пункте мы указываем кодовую страницу для не-Unicode типов данных (char, varchar, text) и порядок сортировки текстовых данных.

Выбрать опцию «Смешанный режим» и задать пароль для встроенной учетной записи администратора «sa» (эта учетная запись обладает максимальными правами доступа ко всем функциям и объектам на SQL-сервере). Дополнительно можно указать учетные записи пользо- вателей Windows или целые группы пользователей Windows, которые должны обладать макси- мальными правами доступа к SQL Server (например, встроенную группу «Администраторы»). Затем перейти на закладку «Каталоги данных»

В поле «Корневой каталог данных» ввести путь к папке, где будут размещаться файлы баз данных (рекомендуется использовать отдельный от ОС физический диск), и нажать кнопку
«Далее»

Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно прой- дены. Если будут обнаружены какие-то проблемы, то необходимо их устранить и запустить повторную проверку кнопкой «Включить заново». Затем нажать кнопку «Далее»

Нажать кнопку «Установить»

После завершения установки нажать кнопку «Далее»

Нажать кнопку «Закрыть».

Установка Microsoft SQL Server 2008 Express завершена!

Практическая работа №7. Экспорт данных базы в документы пользователя.

Цель работы: изучить технологию экспорта данных базы а документы пользователя

ХОД РАБОТЫ

СУБД имеет в своем составе инструментарий для копирования информации, как в БД, так и во внешние источники, например, БД других производителей (MS SQL Server, Interbase). **Экспорт** - Копирует данные во внешние файлы, которые предназначены только для последую- щего импорта в другую БД СУБД. Данные сохраняются в двоичном формате

Импорт	Копирует данные в БД СУБД ORACLE из внеш- них файлов, которые были созданы утилитой экспорта ORACLE.	IMP
Загрузка	Копирует данные в БД из внешних, текстовых файлов, которые имеют либо стандартных формат разде- лителей, либо любой другой формат, поддерживаемый утилитой.	sqlldr

Export - дополнительнпя утилиты в основном, эти утилиты применяются для резервного копирования и миграции БД (между серверами). Ниже приведены другие возможности утилит Export:

1. хранение данных в файлах ОС для архивирования;
2. выборочное резервное копирование частей БД;
3. перемещение данных из одной пользовательской схемы в другую;
4. перемещение данных с одной аппаратной платформы или ОС в другую.
5. экономия пространства и повышение производительности за счет уменьшения фраг- ментации.

Утилита Export записывает информацию о таблицах и других объектах БД (операторы создания индексов, привилегии на экспортируемые объекты и т.д.), а также данные самих таб- лиц. Затем утилита Export сохраняет эту информацию в именованных файлах ОС. Файлы ОС, создаваемые утилитой Export, известны как файлы дампа. Файлы дампа, которые представлены в двоичном формате ORACLE, могут применяются только в утилите Import. Можно назвать файл дампа любым именем, допустимым в ОС. Если вы не укажите имя выходного файла для утилиты Export, то по умолчанию файл примет название “EXPDAT.DMP”.

После экспорта созданные утилитой файлы дампа можно записать на съемный носитель для дальнейшего хранения, перемещения или восстановления. Чтобы запустить утилиту экс- порта необходимо выполнить:

Кнопка «ПУСК» => «ВЫПОЛНИТЬ» => в открывшемся окне набрать «Exp» и нажать «ОК».

Основные параметры утилиты экспорта перечислены в таблице 1.

Таблица 1

Параметр	Значение по умолчанию	Описание
CONSTRAINTS	N	Указывает, экспортируются ли табличные ограничения.
FILE	expdat.dmp	Имя файла, в который будут импортироваться данные. По умолчанию именем файла будет expdat.dmp (сокращенно от EXPort DATa.DuMP). Если требуется другое имя файла, то из- мените параметр FILE.
FULL	N	Если FULL=Y, экспортироваться будет вся БД, включая сведе- ния о табличных пространствах.
GRANTS	Y	Указывает, будут ли экспортироваться привилегии для экспор- тируемых объектов.
HELP	N	Если задано HELP=Y, то другие параметры не требуются. На компьютер выводится справочная информация.
INDEXES	Y	Указывает, экспортируются ли определенные пользователем индексы. Системные индексы, созданные посредством опреде- ления ограничений (первичный ключ, уникальный ключ) экс- портируются, независимо от значения параметра INDEXES
LOG		Имя файла, в который будет записан журнал экспорта. Если не указано иное, Oracle дает файлу расширение .LOG
ROWS	Y	Указывает, будут ли экспортироваться данные таблиц. Если ROWS=N, то экспортируются только определения объектов
TABLES		Указывает список таблиц (с запятой в качестве разделителя), которые должны быть экспортированы. Этот параметр исполь- зуется совместно с параметром FROMUSER. В не-UNIX среде, например, в Windows, список таблиц необходимо заключать в круглые скобки
TABLESPACES		Список табличных пространств, которые должны быть экспор- тированы.
OWNER		Список имен пользователей БД, объекты которых будут экс- портированы.
USERID		Указывает имя и пароль пользователя, который осуществляет процесс экспорта. Формат параметра — «имя пользователя/па- роль@сервер».

Рассмотрим несколько сценариев экспорта:

1. **Экспорт таблиц**. Режим экспорта таблиц используется для экспорта одной таблицы или нескольких таблиц БД. Пользователи, имеющие доступ к таблицам других пользователей, могут экспортировать эти таблицы, указав перед таблицей имя схемы. Пример команды:

exp userid=reldb/ret@LOCALHOST file=c:\kbk.dmp log=c:\kbk.log tables=ver11.kbk.

2. **Экспорт схемы пользователя**. Режим экспорта схемы пользователя используется для экспорта всех объектов, принадлежащих схеме. Этот режим работает хорошо при создании пользователя, который является владельцем всех объектов приложения. Например, если суще- ствует пользователь с именем SALES, который является владельцем всех объектов в схеме SALES, экспорт схемы может выглядеть следующим образом:

exp userid=reldb/ret@proddb file=c:\USER_exp.dmp log=c:\USER_EXP.log owner= SALES.

3. **Экспорт БД**. Режим экспорта БД используется для экспорта всех объектов БД, за исключением объектов, которые обычно создаются и поддерживаются учетной записью SYS. Экспортировать БД могут только пользователи, которым назначена роль EXP_FULL_DATABASE.

exp userid=reldb/ret@proddb file=c:\DB_exp.dmp log=c:\DB_exp.log full=y.

Утилита Import противоположна по действию утилите Export. Она отвечает за чтение файлов дампа в целях воссоздания объектов БД, а также любого состояния, в котором они экс- портировались первоначально. Утилита Import может также преобразовывать данные, предо- ставленные с разных платформ, например, с UNIX машины в ASCII кодах, на мейнфрейм с кодировкой EBCDIC и наоборот, что позволяет перемещать данные с одной платформы на дру- гую. Утилита Import может работать в интерактивном режиме или в режиме командной строки.

Примеры использования утилиты экспорта:

Экспорт таблицы с данными.

IMP USERID=HR/1@LOCALHOST FILE=C:\IMP.DAT LOG=C:\IMP.LOG TABLES=ABC IGNORE=Y.

Экспорт схемы пользователя.

IMP USERID=HR/1@LOCALHOST FILE=C:\IMP.DAT LOG=C:\IMP.LOG FROMUSER=STUDENT TOUSER=UTEST IGNORE=Y.

Экспорт БД.

IMP USERID=SYS/1@LOCALHOST FILE=C:\DB.DAT LOG=C:\DBIMP.LOG FULL=Y.

SQL*Loader. Одной из типичных проблем, с которой часто сталкиваются администраторы БД, является перемещение данных из внешних источников в БД ORACLE. Сложность этой задачи возрастает с появлением хранилищ данных, приходится перемещать уже не мегабайты данных, а гигабайты, а в некоторых случаях и терабайты. ORACLE предусмотрел для решения этой задачи специальную утилиту. **SQL*Loader** - универсальное инструментальное средство, которое загружает внешние данные в таблицы БД ORACLE. Утилита SQL*Loader является гибкой и настраиваемой до такой степени, что обычно удается обойтись без процедур на языках третьего поколения с внедренными операторами SQL.

Для работы утилиты SQL*Loader необходимы входные параметры двух типов:

1. **внешние данные**, которые находятся на диске;

2. **управляющая информация**, которая содержится в управляющем файле и описывает характеристики внешних данных.

В результате выполнения утилиты формируются выходные данные, часть из которых является необязательной, таблицы ORACLE, журналы, файлы некорректных записей и файлы отвергнутых записей.

Основные параметры утилиты SQL*Loader

Параметр	Описание
USERID	Указывает имя и пароль пользователя, который осуществляет процесс загрузки. Формат параметра — «имя пользователя/пароль@сервер».
CONTROL	Управляющий файл.
BAD	Параметр указывает путь к файлу, в который будут записаны незагруженные за-

	писи.
DATA	Параметр указывает на файл с внешними данными для загрузки в БД.
ERRORS	Максимальное количество допустимых ошибок при выполнении загрузки.
LOG	Файл протокола, который после выполнения загрузки содержит подробное описание процесса загрузки.

Внешние данные. Утилита SQL*Loader может обрабатывать файлы данных практически любого типа и поддерживает собственные типы данных почти для любой платформы. Дан- ные обычно считываются из одного или нескольких файлов данных. Допускается располагать данные для загрузки непосредственно в управляющем файле.

SQL*Loader позволяет загружать данные либо в двоичном формате, либо в текстовом. Данные могут находиться в файлах как фиксированного, так и переменного форматов. При фиксированном формате поля данных всегда имеют одинаковую длину, независимо от содержащихся значений. В файлах с переменным форматом данные находятся в записях, которые изменяются по длине в зависимости от размера значений. Поля имеют длину, необходимую для размещения данных. Поля в файлах с переменным форматом могут быть разделены завершаю- щими символами, например, запятыми, пробелами, или заключены в ограничительные сим- волы.

Управляющий файл. Прежде, чем утилита SQL*Loader сможет обработать внешние данные, необходимо задать определения этих данных. Управляющий файл - это файл произвольного формата, который содержит информацию, указывающую SQL*Loader, как обрабаты- вать внешние данные.

Основные параметры управляющего файла

Параметр	Описание
LOAD DATA CHARACTERSET	Кодировка символов, в которой производится загрузка данных. Для кириллицы значение этого параметра следует задать CL8MSWIN1251.
INFILE	INFILE "Путь\Имя файла" указывает файл с данными INFILE * данные для загрузки находятся в управляющей файле.
REPLACE (APPEND)	Заменить существующие записи таблицы данными из файла (Добавить данные к существующим записям таблицы).
INTO TABLE ИМЯ ТАБЛИЦЫ	Указывают таблицу для загрузки данных.
FIELDS TERMINATED BY 'разделитель'	Указывает на разделитель между полями в файле данных
(СПИСОК_КОЛОНОК) (A,B,C,D)	Список колонок, в которые будут загружаться данные.
BEGINDATA	Начало данных в управляющем файле

Пример 1.

Файл данных:

```
1,3,5
2,4,2
3,21,02
```

Управляющий файл:

```
load data
infile "c:\load.txt"
append
into table abc
fields terminated by ',' (a,b,c)
```

*Вызов SQL*Loader:*

```
SQLLdr userid=student/istas@localhost control=c:\upr.txt log=c:\loader.log bad=c:\bad.txt.
```

Пример 2.

Управляющий файл:

```
load data
CHARACTERSET CL8MSWIN1251
infile *
append
into table Sotr
fields terminated by ','
(FirstName, LastName, Age)
begindata
Семен,Иванов,24
Иван,Петров,42
Олег,Сидоров,23 (a,b,c)
```

*Вызов SQL*Loader:*

```
SQLLdr userid=student/istas@localhost control=c:\upr.txt log=c:\loader.log bad=c:\bad.txt.
```

ХОД РАБОТЫ:

1. Подключитесь к учебной БД под учетной записью student. Создайте двух новых поль- зователей (USER1) и (USER2). Создайте новую роль. Присвойте роли права подключаться, со- здавать таблицы, создавать последовательности, создавать триггеры и роль DBA.

2. Создайте таблицу-справочник стран: ID (первичный ключ), название страны (сим- вольное, уникальное). Добавьте в таблицу две-три записи.

3. Создайте таблицу-справочник городов: ID (первичный ключ), страна (внешний ключ к таблице стран), название города (символьный). Создайте последовательность. Создайте триг- гер к таблице, который перед вставкой записи заполняет первичный ключ. Добавьте в таблицу пять записей.

4. Экспортируйте таблицу стран вместе с данными. Подключитесь к учебной БД под учетной записью USER1. Напишите запрос, который бы возвращал все записи таблицы стран. Добавьте три записи в таблицу. Удалите таблицу стран.

5. Подключитесь под учетной записью student. Создайте хранимую процедуру, которая бы возвращала список городов с привязанными странами, т.е. город и страна. Для этого приме- ните курсорный цикл. Вызовите хранимую процедуру.

6. Экспортируйте всю схему пользователя student в файл.. Подключитесь к учебной БД под учетной записью USER2. Напишите запрос, который бы возвращал все записи таблицы го- родов. Добавьте три записи в таблицу. Удостоверьтесь, что триггер заполнил первичный ключ.

В после «Имя сервера» указываем локальную машину и название сервера, с которым будем работать. Вводим установленный в п.15 пароль для учётной записи «sa».

Далее видим типичный для Visual Studio интерфейс:

Практическая работа № 8 Импорт данных пользователя в базу данных

Цель работы: изучить технологию импорта данных пользователя в базу данных

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Импорт - копирует данные в БД СУБД из внешних файлов, которые были созданы утилитой экспорта .

Import - дополнительная утилита в основном применяется для резервного копирования и миграции БД (между серверами либо из более старой версии в более новую). Ниже приведены дру- гие возможности утилиты Import :

1. хранение данных в файлах ОС для архивирования;
2. выборочное резервное копирование частей БД;
3. перемещение данных из одной пользовательской схемы в другую;
4. перемещение данных с одной аппаратной платформы или ОС в другую.

5. экономия пространства и повышение производительности за счет уменьшения фрагментации.

FILE expdat.dmp Имя файла, из которого будут импортироваться данные. По умолчанию именем файла, из которого осуществляется импорт, будет expdat.dmp.

FROMUSER Если указан этот параметр, то импортируются только те объекты, владельцем которых является пользователь с иден-тификационным кодом FROMUSER

FULL N Если FULL=Y, то импортироваться будет вся БД.

GRANTS Y Указывает, будут ли заданы все полномочия для экспортированных объектов.

HELP N Если задано HELP=Y, то другие параметры не требуются. На экране будет выведена справочная информация.

IGNORE NO Если задано IGNORE=Y, то ошибки при создании объектов игнорируются и строки вставляются в таблицу. Будьте внимательны, поскольку, если для таблицы не определено ограничение уникальности значений, то могут появиться дублирующие записи. Отметим, что о других ошибках, не связанных с созданием объектов (например, проблемах с ОС), пользователь будет информирован в обычном режиме.

INDEXES Y Указывает, экспортируются ли определяемые пользователем индексы. Системные индексы, созданные посредством опре-деления ограничений (первичный ключ, уникальный ключ) импортируются независимо от значения параметра ISDEXES.

LOG Имя файла, в который будет записан журнал импорта. Если не указано иное, Oracle даст файлу расширение LOG.

TABLES Указывает список таблиц (с запятой в качестве разделителя), которые должны быть импортированы. Этот параметр используется совместно с параметром FROMUSER. В не-UNIX среде, как, например, Windows, необходимо заключать список таблиц в круглые скобки.

TOUSER Параметр TOUSER указывает имя пользователя, который будет владельцем импортируемых объектов. Данный параметр необходимо использовать совместно с параметром FROMUSER.

USERID Указывает имя и пароль пользователя, который осуществляет процесс импорта. Формат параметра — «имя пользователя/пароль@сервер».

ХОД РАБОТЫ

1. Подключитесь к учебной БД под учетной записью student. Создайте двух новых поль- зователей (USER1) и (USER2). Создайте новую роль. Присвойте роли права подключаться, со- здавать таблицы, создавать последовательности, создавать триггеры и роль DBA.

2. Создайте таблицу-справочник стран: ID (первичный ключ), название страны (символьное, уникальное). Добавьте в таблицу две-три записи.

3. Создайте таблицу-справочник городов: ID (первичный ключ), страна (внешний ключ к таблице стран), название города (символьный). Создайте последовательность. Создайте триг- гер к таблице, который перед вставкой записи заполняет первичный ключ. Добавьте в таблицу пять записей.

4. Импортируйте таблицу из файла в схему пользователя USER1. Подключитесь к учеб- ной БД под учетной записью USER1. Напишите запрос, который бы возвращал все записи таб- лицы стран. Добавьте три записи в таблицу. Удалите таблицу стран.

5. Подключитесь под учетной записью student. Создайте хранимую процедуру, которая бы возвращала список городов с привязанными странами, т.е. город и страна. Для этого приме- ните курсорный цикл. Вызовите хранимую процедуру.

6. Импортируйте схему пользователя student из файла в схему пользователя USER2. Под- ключитесь к учебной БД под учетной записью USER2. Напишите запрос, который бы возвра- щал все записи таблицы городов. Добавьте три записи в таблицу. Удостоверьтесь, что триггер заполнил первичный ключ.

7. Создайте текстовый файл или новый документ в EXCEL. Заполните десять строк для переноса в таблицу городов с помощью SQL*Loader. Создайте в текстовом редакторе управля- ющий файл и импортируйте в таблицу городов данные. Напишите запрос, который бы возвра- щал все записи таблицы городов. Убедитесь, что после импорта появились новые записи.

Практическая работа № 9 Выполнение настроек для автоматизации обслуживания базы данных

Цель работы: Освоение некоторых возможностей автоматизации управления базой данных

ХОД РАБОТЫ

При работе с базой данных часто приходится многократно выполнять одинаковые порой рутинные операции. Вполне естественно было бы автоматизировать их выполнение. Для этого СУБД располагает достаточными средствами, позволяющими во многом автоматизировать и упорядочить работу с базой данных. К числу таких средств относятся:

- пользовательские меню и инструментальные панели;
- кнопочные формы управления базой данных;
- средства настройки параметров запуска базы данных;
- макросы и модули.

Задание 1. Создайте пользовательское меню для управления базой данных, содержащее категории Формы и Отчеты с пунктами (командами) для открытия ранее составленных форм и отчетов.

1. Для создания новой строки меню откройте окно Настройка. Для этого выполните команду ВИД/Панели инструментов/Настройка или, щелкнув правой клавишей по любой панели инструментов, выберите в контекстном меню пункт Настройка.
2. В окне Настройка на вкладке Панели инструментов щелкните по кнопке Создать.
3. В окне Создание панели инструментов введите имя панели инструментов: Управле- ние базой данных. Нажмите кнопку Ok. В окне БД появится небольшая миниатюра панели. Пе- ретащите созданную панель инструментов в верхнюю часть окна установив над строкой меню.
4. В окне Настройка нажмите кнопку Свойства и определите тип созданной панели - Строка меню. Закройте окно установки свойств.
5. Добавьте в строку созданного меню категорию Формы. Для этого в окне Настройка откройте вкладку Команды и в списке категорий щелкните по категории Новое меню. Перетащите команду Новое меню из списка команд в правом подокне на строку меню Управление базой данных. Не закрывая окно Настройка, щелкните правой клавишей в строке меню по категории Новое меню и в контекстном меню замените имя категории на Формы.
6. Добавьте в меню категорию Отчеты аналогично пункту 5.
7. Аналогично добавьте в меню Формы новое подменю, назвав его Простые.
8. В окне Настройка на вкладке Команды выделите категорию Все формы. Перета- щите строку с названием одной из созданных ранее форм - Студент простая в область команд (пунктов) категории Формы строки меню Управление базой данных. Включив контекстное меню новой команды, установите стиль отображения - Только текст.
9. Аналогично добавьте в область команд категории Формы/Простые пункты с назва- нием форм – Группа и Простая форма по запросу.
10. Добавьте в категорию Отчеты меню Управление базой данных пункты с названи- ями отчетов. Закройте окно Настройка. Проверьте работу меню.
11. Выполните команду СЕРВИС/Параметры запуска и установите в окне Параметры запуска следующие параметры запуска при открытии базы данных:

- введите в качестве заголовка приложения название Академия;
- выберите в качестве строки меню строку Управление базой данных.
- отмените вывод на экран окна базы данных, строки состояния, полного набора меню встро- енных панелей инструментов.

Закройте окно Параметры запуска. Закройте базу данных, затем повторно откройте. От- кроется окно базы данных, содержащее только одну строку пользовательского меню

Управле- ние базой данных с категориями Формы и Отчеты. Убедитесь в правильной работе команд меню.

12. Восстановите для базы данных Академия отображение окна базы данных, полного набора меню, встроенных панелей инструментов. Для этого перезагрузите базу данных и при повторном открытии держите нажатой клавишу SHIFT. Выполните команду СЕРВИС/Параметры запуска и восстановите исходное состояние флажков.

Задание 2. Создайте макрос для автоматического формирования экзаменационных ведомостей, рассмотренных ранее. Отдельные таблицы должны быть созданы для каждой группы студентов, имеющейся в базе данных, и для выбранной дисциплины.

Целью разработки макроса является исключение необходимости каждый раз перед созданием новой таблицы (экзаменационной ведомости) вручную переименовывать ранее создан- ную таблицу, чтобы предотвратить ее удаление. Макрос должен сам создавать таблицы с име- нами, соответствующими номерам групп и кодам дисциплин, по схеме: Ведомость NNN-K,

где NNN - номер группы, введенный в диалоговом окне, K - код дисциплины.

Для того, чтобы передать параметры создаваемых таблиц (номер группы и код дисциплины) можно, например, использовать форму, созданную на основе временной таблицы и вы- бирать эти параметры из первой строки этой формы.

Кроме того, для того чтобы избежать лишних остановок при выполнении макроса, поручите макросу не выводить вспомогательные служебные сообщения и сообщения-предупреждения. При выполнении макроса пользователь должен будет вводить только номер группы и код дисциплины.

При конструировании макроса можно использовать ранее созданный запрос с именем Запрос на создание экзаменационной ведомости.

Поскольку условия выполнения макрокоманд могут определяться только значениями полей или элементов управления форм и отчетов предварительно следует создать вспомогательную табличную форму на основании таблицы Ведомость 1.

Автоматически создайте новую табличную форму на основании таблицы Ведомость Для этого в окне База данных выберите объект Формы и щелкните по кнопке Создать. В диалоговом окне Новая форма выберите способ создания - Автоформа: табличная, а в качестве источника данных - таблицу Ведомость Щелкните по кнопке Ok. После появления на экране формы закройте ее и сохраните под именем Форма для макроса.

Для того, чтобы форма не зависела от таблицы Ведомость 1 замените имя источника за- писей в окне свойств формы на Ведомость 00. Для этого откройте форму в режиме конструк- тора и щелкните по кнопке Свойства, расположенной инструментальной панели Конструктор форм. Отредактируйте значение свойства Источник записей на вкладке Данные.

В окне База данных выберите объект Макросы и щелкните по кнопке Создать. Появится окно конструктора макросов. Добавьте в окне еще один столбец - Условия. Для этого выпол- ните команду ВИД/Условия или щелкните по кнопке инструментальной панели с соответству- ющим названием.

Щелкните внутри ячейки первой строки и столбца Макрокоманда. Появится поле со списком макрокоманд. Выберите макрокоманду: УстановитьСообщения. В нижней части окна появится аргумент этой команды: Нет. Оставьте его без изменения. Введите в графу Примечание краткий комментарий: Отключение системных и предупреждающих сообщений.

Перейдите на следующую строку и выберите для нее макрокоманду Открыть Запрос. Определите аргументы макрокоманды в нижней части окна. Раскройте список в поле Имя запроса и выберите в нем имя Запрос на создание экзаменационной ведомости. Сохраните значение аргументов Режим - таблица и Режим данных - изменение. Введите комментарий к этой макрокоманде: на создание экзаменационной ведомости. При выполнении данной макрокоманды будет создана таблица Ведомость 1.

После того, как запрос на создание ведомости отработал, его можно закрыть, поэтому в третьей строке выберите макрокоманду Закрыть. Определите аргументы макрокоманды: Тип объекта - Запрос, Имя объекта - Запрос на создание экзаменационной ведомости, Сохранение-Подсказка. Введите комментарий: Закрытие запроса.

Перед открытием созданной ранее Формы для макроса необходимо произвести копирование таблицы Ведомость1 в другую таблицу, которая и будет использована для открытия формы - Ведомость 00. Для этого выберите макрокоманду КопироватьОбъект. Определите аргументы макрокоманды: Новое имя - Ведомость 00, Тип объекта - таблица, Имя объекта - Ведомость 1. Введите комментарий: копирование таблицы Ведомость 1 в Ведомость 00.

Примечание. При определении для макрокоманды в качестве аргументов имен объектов совершенно не обязательно, чтобы соответствующие объекты существовали в базе данных в момент конструирования макроса. Важно, чтобы они были в базе данных к моменту выполне- ния этой макрокоманды.

В следующей строке выберите макрокоманду ОткрытьФорму. Определите аргументы макрокоманды: Имя формы - Форма для макроса, Режим - Форма, Режим окна - Обычное. Вве- дите комментарий: Открыть форму на основе таблицы Ведомость 00.

Следующая макрокоманда должна осуществить переход на первую строку таблицы-формы для проверки значений полей N группы и Код дисциплины, введенных при выполнении запроса в диалоговые окна. Поэтому в очередной строке выберите макрокоманду НаЗапись. Выберите из списка для аргумента Запись значение Первая. Введите комментарий: Перейти на 1-ую запись табличной формы.

Выберите для следующей строки макрокоманду Переименовать для переименования ве- домости для группы 851 и дисциплины с кодом 1. Определите аргументы макрокоманды: Новое имя - Ведомость 851_1, Тип объекта - Таблица, Старое имя - Ведомость 1. В графу Усло- вие введите с помощью построителя для этой строки выражение :

[Формы]![Форма для макроса]![N группы]=851 And [Формы]![Форма для макроса]![Код дисциплины]=1

Введите комментарий: Переименовать Ведомость 1 в Ведомость 851_1

Введите следующую макрокоманду для переименования ведомости для группы 851 и дисциплины с кодом 2. Определите аргументы макрокоманды: Новое имя - Ведомость 851_2, Тип объекта - Таблица, Старое имя - Ведомость 1. Введите Условие:

[Формы]![Форма для макроса]![N группы]=851 And [Формы]![Форма для макроса]![Код дисциплины]=2

Примечание. Для ввода условий и комментариев к макрокомандам целесообразно использовать приемы копирования в буфер, а затем вставки с последующей корректировкой в окне области ввода, открываемом при нажатии сочетания клавиш SHIFT и F2.

Введите следующие строки макроса по аналогии с двумя предыдущими для переимено- вания ведомости для группы 851 и дисциплины с кодом 3, а также для других групп и дисци- плин. При 3-х дисциплинах должно быть 3 макрокоманды на одну группу студентов.

Введите последнюю макрокоманду Закрыть с аргументами: Тип объекта - Форма, Имя объекта - Форма для макроса, Сохранение - Подсказка. Окончательный вид макроса представлен на рис. 7.1

Сохраните макрос, щелкнув по кнопке Сохранить инструментальной панели, под именем Макрос для создания ведомостей. Запустите макрос на выполнение в пошаговом режиме. Для этого, находясь в режиме конструктора щелкните по кнопке инструментальной панели По шагам, а затем по кнопке Запуск. Проследите по шагам правильность исполнения макрокоманд. В случае неверного выполнения или аварийного завершения найдите ошибку и исправьте мак- рос.

После отладки макроса отключите пошаговый режим запуска, повторно щелкнув по кнопке По шагам, закройте макрос. Произведите запуск макроса из окна базы данных. Для этого в окне База данных выберите объект Макросы, выделите Макрос для создания ведомостей и щелкните по кнопке Запуск на инструментальной панели окна. Запустите макрос для формирования экзаменационных ведомостей по разным группам и дисциплинам.

Практическая работа № 10 Мониторинг работы сервера

Цель работы: изучение мониторинга работы сервера

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Наблюдение за базами данных выполняется с целью оценки производительности сервера. Эффективное наблюдение подразумевает регулярное создание моментальных снимков те- кущей производительности для обнаружения процессов, вызывающих неполадки, и постоян- ный сбор данных для отслеживания тенденций роста или изменения производительности.

Постоянная оценка производительности базы данных помогает добиться оптимальной производительности путем минимизации времени ответа и максимального увеличения пропускной способности. Приблизительный сетевой трафик, дисковый ввод-вывод и загрузка ЦП — ключевые факторы, влияющие на производительность. Следует тщательно проанализировать требования приложения, понять логическую и физическую структуру данных, оценить ис- пользование базы данных и добиться компромисса между такими конфликтующими нагруз- ками, как оперативная обработка транзакций (OLTP) и поддержка решений.

Мониторинг и настройка производительности баз данных

В состав Microsoft SQL Server и операционной системы Microsoft Windows входят служебные программы, позволяющие следить за текущим состоянием базы данных и измерять производительность, если это состояние меняется. Для наблюдения за Microsoft SQL Server можно использовать целый ряд средств и методик. Наблюдение за SQL Server позволяет решать следующие задачи:

Определять возможности увеличения производительности. Например, выполняя мониторинг времени ответа для часто используемых запросов, можно определить, требуется ли изменить текст запроса или индексы таблицы.

Оценивать активность пользователей. Например, выполняя мониторинг пользователей, которые подключаются к экземпляру SQL Server, можно определить, правильно ли настроены параметры безопасности, и проверить работу приложений и систем разработки. Контролируя выполнение SQL-запросов, можно определить, правильно ли они написаны, и проверить результаты, которые они возвращают.

Устранять проблемы или отлаживать компоненты приложений, например хранимые процедуры.

Мониторинг в динамической среде

Изменение этих условий приведет к изменению производительности. По результатам оценки можно заметить изменения производительности при увеличении числа пользователей, изменении методов доступа пользователей и методов соединения, при увеличении объема содержимого базы данных, изменении клиентского приложения и данных в приложении, а также при усложнении запросов и увеличении объема сетевого трафика. С помощью средств контроля производительности можно связывать изменения отдельных показателей производительности с изменениями условий и сложных запросов. Примеры:

Отслеживая время отклика на часто используемые запросы, можно определить, нужно ли изменять запросы или индексы опрашиваемых таблиц.

Отслеживая выполнение запросов Transact-SQL можно определить правильность их написания, а также соответствие ожидаемым результатам.

Отслеживая пользователей, пытающихся подключиться к экземпляру SQL Server, можно проверить надежность защиты и протестировать приложения или системы разработки.

Время отклика — это время ожидания возврата пользователю первой строки результирующего набора в форме визуального подтверждения обработки запроса. Пропускная способность — это общее количество запросов, которые сервер может обработать за единицу времени.

С увеличением числа пользователей растет соперничество за ресурсы сервера, что в свою очередь увеличивает время ответа и уменьшает общую пропускную способность.

Задачи наблюдения и настройки производительности

Мониторинг компонентов SQL Server Необходимые действия для мониторинга компонентов SQL Server, такие как монитор активности, расширенные события, динамические ад- министративные представления и функции и т. д.

Средства контроля и настройки производительности Список средств наблюдения и настройки, доступных в SQL Server, например статистики динамических запросов и помощник по настройке ядра СУБД.

Обновление баз данных с помощью помощника по настройке запросов Поддержание стабильной производительности рабочей нагрузки во время обновления до нового уровня совместимости базы данных.

Мониторинг производительности с использованием хранилища запросов Использование хранилища запросов для автоматической регистрации журнала запросов, планов и стати- стики выполнения и сохранение этих данных для просмотра.

Формирование базовых показателей производительности Инструкции по формированию базовых показателей производительности.

Локализация проблем производительности Локализация проблем производительности базы данных.

Выявление узких мест Наблюдение за производительностью сервера и отслеживание его работы для выявления узких мест.

Использование динамических административных представлений для определения статистики использования и производительности представлений Рассматриваются методы и скрипты, используемые для получения информации о производительности запросов.

Мониторинг производительности и действий сервера Использование средств наблюдения за производительностью и активностью SQL Server и Windows.

Отслеживание использования ресурсов Использование системного монитора (также известного как perfmon) для измерения производительности SQL Server с помощью счетчиков производительности.

ХОД РАБОТЫ

<u>Задание.</u> Изучить инструменты диагностики SQL Server

Администраторы баз данных проводят диагностику SQL Server с помощью встроенных инструментов и скриптов, облегчающих использование этих инструментов. Скрипты в данном случае могут быть как отдельными программами, так и программными файлами, интегрируе- мыми в структуру софта.

В SQL Server встроено два инструмента: Activity Monitor (монитор активности) и Data Collector (сборщик данных). С их помощью выполняются практически все задачи диагностики.

Activity Monitor: функции, задачи, преимущества и недостатки использования

Activity Monitor – это утилита, позволяющая оценивать активность пользователей приложения или сети. Она показывает текущее состояние SQL Server, осуществляемые на момент проверки процессы и то, как они отражаются на производительности СУБД.

Activity Monitor выглядит как окно с несколькими вкладками. Администратор базы дан- ных может открыть такие панели:

1. Overview (обзор). На этой панели демонстрируется время обработки запросов процес- сором, количество ожидающих запросов, количество запросов в секунду, ввод и вывод данных.

2. Processes (процессы). На этой панели отражаются все активные процессы и подробная информация по ним. В Processes также можно запустить скрипт, который автоматически анализирует выбранный процесс.

3. Resource Waits (ожидающие ресурсы). На этой панели отображается, какие ресурсы необходимы СУБД для выполнения заданных функций. В перечень ресурсов входит объем опе- ративной памяти и сервера, сети, компиляция и др. В этой же панели администратор базы дан- ных может просмотреть общий и средний промежуток времени ожидания ресурсов.

4. Data File I/O (ввод-вывод данных). На этой панели отражаются все операции, связанные с внесением изменений в файлы БД, а также полная информация об этих файлах.

5. Recent Expensive Queries (последние ресурсоемкие запросы). На этой панели отражаются те запросы, которые были выполнены в течение ближайших 30 секунд, и обработка которых затребовала наибольшего числа ресурсов. В некоторых версиях SQL Server эта панель называется Activity Expensive Queries (активные ресурсоемкие запросы).

Сбор и обработка данных с помощью Activity Monitor рис.9 ведется в режиме реального времени и только при условии развертывания панели. Если администратор базы данных свора- чивает панель, обработка информации прекращается. При этом на экране можно развернуть все пять панелей, чтобы оценить активность пользователей по разным

параметрам.

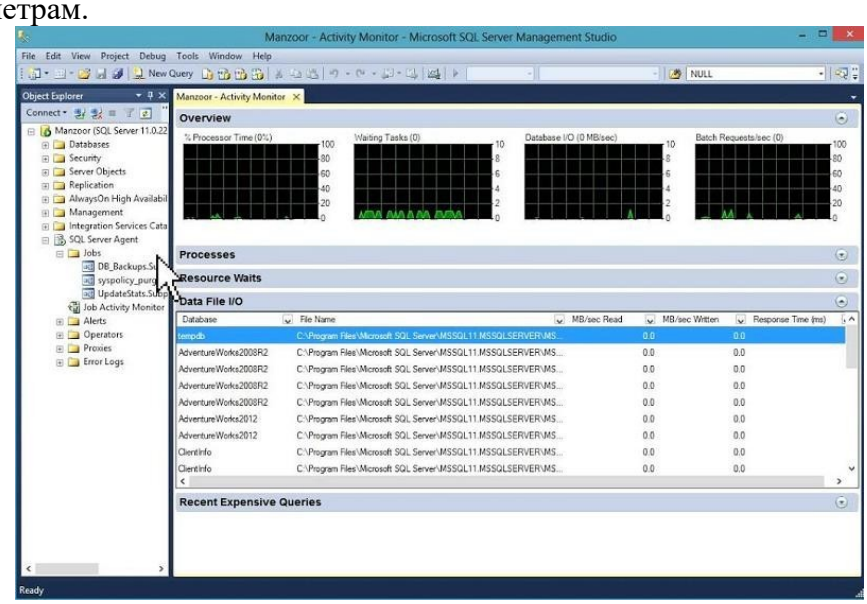

Рисунок 9

Для облегчения процесса использования Activity Monitor можно фильтровать, сортировать и менять местами столбцы с данными диагностики. Это делается с помощью компьютерной мышки прямо на развернутой панели.

Преимуществом использования Activity Monitor является то, что использование этого инструмента практически не отражается на производительности СУБД.

Что касается недостатков этого инструмента, в их число можно включить:

- ограниченное количество параметров, по которым ведется диагностика;
- отсутствие эталона, то есть шаблона с указанием допустимых значений параметров;
- отсутствие возможности формирования отчета о результатах проверки для их последующего анализа.

Из этого можно сделать вывод, что Activity Monitor – это прекрасный инструмент для проведения быстрой диагностики и поиска запросов, затратных с точки зрения потребления ресурсов.

Практическая работа № 11 Выполнение резервного копирования

Цель работы: ознакомиться с основными конструкциями для резервного копирования БД

ХОД РАБОТЫ

Задание 1. необходимо создать резервные копии базы данных «МММ» с использованием полного резервного копирования, разностного резервного копирования и резервного копирования журнала транзакций.

1. Запустите SQL Server Management Studio (SSMS), подключитесь к своему экземпляру SQL Server, используя технологию 1.
2. Создайте папку с именем c:\Student\ВашаПапка\test.
3. Откройте окно нового запроса. Измените контекст на базу данных master, используя тех- нологию 6. Наберите и исполните следующую команду, чтобы создать полную резерв- ную копию базы данных:
   BACKUP DATABASE MMM TO DISK = 'C:\.....TEST\AW.BAK’
   Ознакомьтесь с результатами запроса – какая информация обработана, сколько стра- ниц, сколько файлов.
4. Внесите изменение в таблицу «Модель» базы данных МММ. Добавьте одну запись (при- думайте сами)/
5. Откройте окно нового запроса наберите и исполните следующую команду, чтобы создать резервную копию журнала транзакций и сохранить только что внесенное изменение:

BACKUP LOG MMM TO DISK = ‘C:\.....TEST\AW1.TRN'

Ознакомьтесь с результатами запроса – какая информация обработана, сколько стра- ниц, сколько файлов.

6. Внесите еще одно изменение в таблицу«Модель».
7. Откройте окно нового запроса наберите и исполните следующую команду, чтобы создать разностную резервную копию базы данных:

BACKUP DATABASE MMM TO DISK = ‘C:\.....\TEST\AWDIFF1.BAK' WITH DIFFERENTIAL

Ознакомьтесь с результатами запроса – какая информация обработана, сколько стра- ниц, сколько файлов.

8. Внесите еще одно изменение в таблицу«Модель».
9. Откройте окно нового запроса наберите и исполните следующую команду, чтобы создать полную резервную копию базы данных в указанном месте на диске:

BACKUP LOG MMM TO DISK = ‘C:\....TEST\AW2.TRN'

Ознакомьтесь с результатами запроса – какая информация обработана, сколько стра- ниц, сколько файлов.

Задание 2. Необходимо организовывать со стороны клиентского приложения, созданного в Visual Studio удаленное администрирование БД (резервное копирование).

Ход работы:

1. Создайте новый проект Windows Application и сохраните его в своей папке под именем Лабы_MMM_2 семестр.
2. В главную форму добавьте меню - *Файл (Открыть, Закрыть, Выход)*
   Справочники (Модель, Магазин, Дерево моделей)
   Заказы (Работа с заказами)
   Отчеты (Прайс-лист, Бланк заказов)
   Администрирование БД (Резервное копирование, Безопасность)
   Сервис (Калькулятор)
   Помощь (Справка, О программе)
3. Добавьте новую форму в проект
4. Добавьте на только что созданную форму компоненты
5. Обеспечьте функциональную работу формы (напишите обработчик кнопки «Резервное копирование» с использованием объектов SMO. Описание объектов SMO, их свойств и методов см. в лекционном материале.)
5. Добавьте возможность открытия данной формы при выборе в главной форме пункта меню Администрирование БД ◊ Резервное копирование
6. Запустите проект, проверьте работу формы.
7. Закройте проект
8. Убедитесь в появлении файла резервной копии на диске (файл, который указан в тексте программы).
9. Откройте SSMS. Добавьте в таблицу «Модель» новую строку данных (самостоятельно).
10. Средствами оболочки SSMS, выполните восстановление БД из резервной копии, создан- ной вашей программой
11. Убедитесь, что после восстановления добавленных строк в таблице «Модель» нет.

Практическая работа №12 Восстановление базы данных из резервной копии

Цель работы: ознакомиться с основными конструкциями для восстановления БД из резервного копирования

ХОД РАБОТЫ

1. Для восстановление базы данных из резервной копии следует щелкнуть правой клавишей мыши оп элементу, представляющему выбранную базу данных в правой части

фрейма SQL Server Management Studio, и во всплывающем меню выбрать пункт ***Tasks*** > ***Restore*** > ***Database*** При этом открывается диалоговое окно рис.10 для управления восстановлением выбранной базы данных:

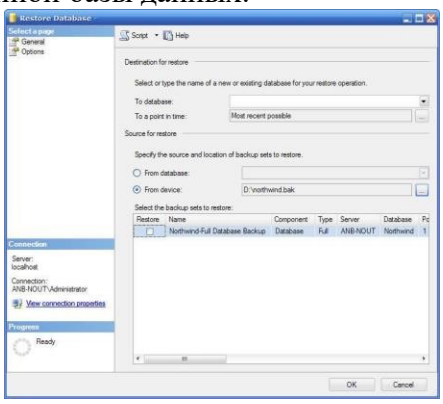

Рисунок 10

Задание 1.

1. Изучите утилиту SQL Server Configuration.

Запустите утилиту SQL Server Configuration Manager и с ее помощью определите список запущенных на сервере служб. Запишите этот список в отчет.

На сервере с установленным MS SQL Server с помощью утилиты Services определите параметры запуска служб MS SQL Server и запишите их в отчет. (Если нет доступа к утилите Services, то при помощи SQL Server Configuration Manager рис.11).

Определите, с помощью каких сетевых библиотек может быть установлено соединение с MS SQL Server (см. пример рис). Какие библиотеки являются активными в момент запуска?

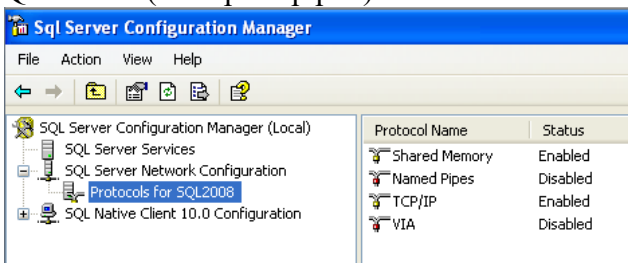

Рисунок 11

При помощи SQL Server Configuration Manager определите, на основе каких сетевых биб- лиотек клиент может подключаться к MS SQL Server (см. пример рис.12).

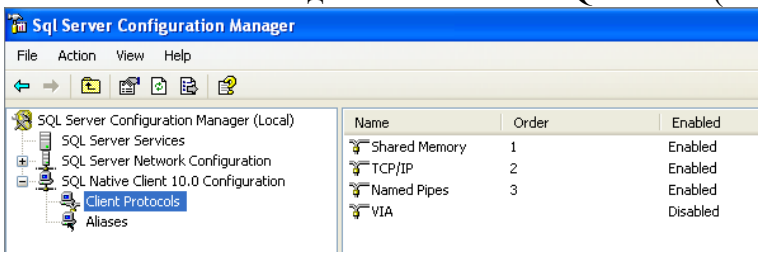

Рисунок 12

2. Установите соединение с SQL сервером.

На рабочей станции запустите SQL Server Management Studio и выберите из списка ло- гическое имя сервера, запущенного на вашем компьютере. Если нужного сервера нет в списке, то можно выбрать <Browse for more…> и найти требуемый сервер в списке серверов, к которым может быть выполнено подключение.

Подключитесь к серверу с использованием средств аутентификации MS SQL Server.

Для того чтобы написать новый запрос необходимо выполнить команду New Query рас-

положенную на панели инструментов *SQL Server Management Studio*. В результате откроется новая вкладка, которая предоставляет следующие возможности:

- заголовок, в котором указывается логическое имя сервера, текущая база данных и имя пользователя, установившего соединение;
- область запроса, используемая для ввода запросов, передаваемых MS SQL Server;

- область результатов, в которой отображаются результаты выполнения запроса, а способ отображения задается кнопками Messages (в виде текста) и Results (в виде таблицы) со- ответственно.

3. С помощью команды SELECT @@version определите и запищите в отчет информацию об используемой версии MS SQL Server и операционной системы (результат запроса должен быть отображен в текстовом виде) (пример рис.13).

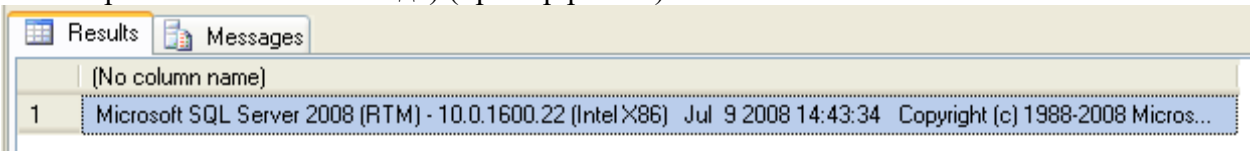

Рисунок 13

Примечание: Для выполнения запроса необходимо выполнить команду Query – Execute (F5), а для анализа правильности его синтаксической записи можно воспользоваться командой Query – Parse (Ctrl+F5).

SQL Server Management Studio позволяет открывать несколько окон запросов и работать с несколькими базами данных одновременно. В каждом окне устанавливается собственное со- единение с MS SQL Server на основе различных учетных записей пользователей и их паролей. Для создания нового подключения используется команда File – New – Database Engine Query.

Содержимое области запроса текущего подключения может быть сохранено в файле на внешнем носителе командой File – Save.

При помощи панели *Object Explorer* определите имена поддерживаемых баз данных и какие базы данных сервера являются системными (для этого нужно развернуть узел Databases в панели Object Explorer). Запишите эту информацию в отчет.

4. Изучите параметры конфигурации MS SQL Server.

Конфигурирование службы MSSQLServer может быть выполнено либо специальной хранимой процедурой, выполняемой в утилите SQL Server Management Studio, либо графическим способом средствами этой же утилиты. Выбор способа не имеет значения, т.к. графический способ осуществляет доступ к системным данным с помощью этой же хранимой процедуры, только в более наглядной форме.

Для изменения параметров службы с помощью SQL Server Management Studio необходимо выбрать нужный сервер в Object Explorer и в контекстном меню выбрать команду ***Properties.*** В появившемся диалоговом окне можно выполнить настройку всех необходимых параметров.

5. Отобразите список параметров сервера (пример рис.14).

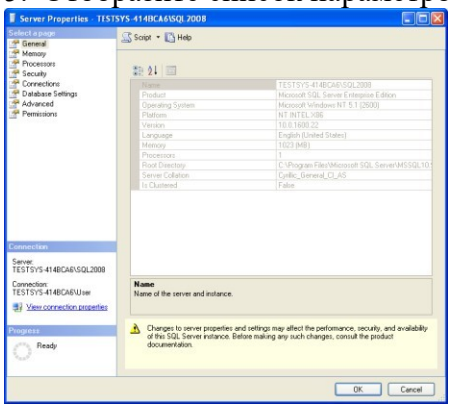

Рисунок 14

Рис. Свойства MS SQL Server 2008

На вкладке *General* отображаются основные сведения о системе: версия операционной системы, объем памяти, количество процессоров и др., а также параметры запуска служб сервера.

Вкладка *Memory* позволяет управлять выделением памяти для выполнения действий MS SQL Server: либо динамическое управление памятью, либо установить фиксированный размер. С помощью вкладки *Security* определяется тип аутентификации пользователей, также определяются параметры аудита доступа к серверу. Можно настроить сервер на использование

определенной учетной записи, под которой будет запускаться служба *MSSQLServer***.**

Вкладка *Connections* позволяет конфигурировать подключения клиентские подключения к серверу. Максимальное количество пользователей, которые могут одновременно подклю- читься к серверу. Если указано нулевое значение, то их количество составляет 32767.

Вкладка *Advanced* содержит некоторые общие установки сервера. Например, определяется язык по умолчанию для сообщений сервера или регулируется поддержка 2000 года, которая определяет, как будут интерпретироваться две последние цифры года.

С помощью вкладки *Database Settings* указываются настройки вновь создаваемых баз данных: параметры индексов и работы с устройствами резервного копирования, время восстановления базы данных.

Определите и запишите в отчет корневой каталог сервера, количество процессоров в си- стеме, тип аутентификации пользователей и максимальное количество пользователей, поддер- живаемых сервером.

Задание 2. Изучите остальные свойства MS SQL Server, доступные в этом диалоге.

1. Создать базу данных с именем Stud_<*фио_студента*>_1 средствами СУБД MS SQL Server с журналом средствами SQL Server Management Studio и с именем Stud_<*фио_студента*>_2 средствами Query Editor и запишите в отчет результаты выполнения процедуры sp_helpdb …. Для созданных вами БД .
2. Переименуйте созданную Вами базу данных Stud_<*фио_студента*>_1 в Stud_<*фио_студента*> ***и*** отобразите в отчете результат выполнения оператора переименования.
3. Определите сведения о дисковом пространстве, занимаемом созданной вами БД. Сожмите базу данных так, чтобы она содержала только 25% пространства, доступного ей на теку- щий момент.
4. Удалите созданную вами базу данных с именем Stud_<*фио_студента*>_2 и отобразите в отчете результат выполнения оператора удаления.
5. Отключить/подключить созданную вами БД Stud_<*фио_студента*> от сервера. Если БД создавалась на жестком диске, то переместить ее на резервный носитель и отобразите в отчете результат выполнения оператора.

Практическая работа №13 Реализация доступа пользователей к базе данных

Цель работы: изучение способов доступа пользователей к базе данных

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Понятие пользователь базы данных относится к базе (или базам) данных, к которым мо- жет получить доступ отдельный пользователь. После успешного подключения сервер опреде- ляет, имеет ли этот пользователь разрешение на работу с базой данных, к которой обращается. Единственным исключением из этого правила является пользователь guest (гость). Особое имя пользователя guest разрешает любому подключившемуся к SQL Server пользователю получить доступ к этой базе данных. Пользователю с именем guest назначена роль public.

Права доступа

Для управления правами доступа в SQL Server используются следующие команды:

1. GRANT. Позволяет выполнять действия с объектом или, для команды — выполнять ее;
2. REVOKE. Аннулирует права доступа для объекта или, для команды — не позволяет вы- полнить ее;
3. DENY. Не разрешает выполнять действия с объектом (в то время, как команда REVOKE просто удаляет эти права доступа).

Объектные права доступа позволяют контролировать доступ к объектам в SQL Server, предоставляя и аннулируя права доступа для таблиц, столбцов, представлений и хранимых про- цедур.

Чтобы выполнить по отношению к некоторому объекту некоторое действие, пользова-

тель должен иметь соответствующее право доступа. Например, если пользователь хочет выпол- нить оператор SELECT * FROM table, то он должен иметь права выполнения оператора SELECT для таблицы table.

Командные права доступа определяют тех пользователей, которые могут выполнять административные действия, например, создавать или копировать базу данных.

Ниже приведены командные права доступа:

CREATE DATABASE — право создания базы данных;

CREATE DEFAULT — право создания стандартного значения для столбца таблицы;

CREATE PROCEDURE — право создания хранимой процедуры.

CREATE ROLE — право создания правила для столбца таблицы;

CREATE TABLE — право создания таблицы;

CREATE VIEW — право создания представления;

BACKUP DATABASE — право создания резервной копии;

BACKUP TRANSACTION — право создания резервной копии журнала транзакций.

Роли

Назначение пользователю некоторой роли позволяет ему выполнять все функции, разрешенные этой ролью. По сути роли логически группируют пользователей, имеющих одинако- вые права доступа.

Роли, определяемые пользователем, позволяют группировать пользователей и назначать каждой группе конкретную функцию безопасности. Существуют два типа ролей уровня базы данных, определяемых пользователем:

1. стандартная роль;
2. роль уровня приложения.

Стандартная роль предоставляет зависящий от базы данных метод создания определяемых пользователем ролей. Самое распространенное назначение стандартной роли — логически сгруппировать пользователей в соответствии с их правами доступа. Например, в приложениях выделяют несколько типов уровней безопасности, ассоциируемых с тремя категориями поль- зователей. Опытный пользователь может выполнять в базе данных любые операции; обычный пользователь может модифицировать некоторые типы данных и обновлять данные; неквалифи- цированному пользователю обычно запрещается модифицировать любые типы данных.

Роль уровня приложения позволяет пользователю выполнять права некоторой роли. Ко- гда пользователь принимает роль уровня приложения, он берет на себя выполнение новой роли и временно отказывается от всех других назначенных ему прав доступа к конкретной базе дан- ных. Роль уровня приложения имеет смысл применять в среде, где пользователи делают за- просы и модифицируют данные с помощью клиентского приложения.

Управление разрешениями на объекты реляционной базы данных несколько отличается от аналогичных операций в отношении объектов файловой системы. В данной лабораторной работе на примере СУБД SQL Server эти отличия будут показаны.

При работе с разрешениями в SQL Server используется понятие участников (principals), которые могут запрашивать ресурсы SQL Server, и которым могут предоставляться разрешения на использование таких ресурсов. Выделяются следующие группы участников:

участники уровня Windows, к которым относятся локальные и доменные учетные записи пользователей и группы;

участники уровня SQL Server, к которым относятся учетные записи SQL Server и роли уровня сервера;

участники уровня базы данных — пользователи базы данных и роли уровня базы данных и приложения.

Необходимо отметить, что SQL Server разделяет понятие учетной записи (login) и пользователя (user). Сервер может быть сконфигурирован на использование только аутентификации Windows *(англ.* Windows Authentication Mode, используется по умолчанию) или на использова- ние смешанного режима аутентификации *(англ.* SQL Server and Windows Authentication mode). В первом случае login можно создать только для пользователя или группы Windows. Во втором случае также возможно использовать собственные учетные записи SQL Server — логин и па- роль хранятся самой СУБД, и ее же средствами выполняется проверка подлинности. При ис- пользовании смешанной аутентификации егановится

доступной административная учетная за- пись sa, которую рекомендуется переименовать и назначить ей надежный пароль. Учетная за- пись авторизуется на выполнение одной из серверных ролей таю.2

Роли уровня сервера

Таблица 2

Роль	Описание возможностей
sysadmin	Разрешено выполнять любые действия на сервере.
dbereator	Разрешено создавать базы данных.
bulkadmin	Могут выполнять инс трукцию BULK INSERT.
diskadmin	Позволяет управлять файлами на диске.
processadmin	Позволяет управлять подключениями, запускать и приостанавливать экземпляр SQL Server.
securityadmin	Создание и управление учетными записями, право «сбросить» пароль учетной записи. Управление разрешениями на уровне сервера и на уровне базы данных (при наличии доступа к базе данных).
serveradmin	Включает возможности ролей diskadmin и processadmin, позволяет изменять параметры конфигурации на уровне сервера и выключать сервер.
setupadmin	Добавление и удаление связанных серверов.
public	Каждая учетная запись принадлежит этой роли, членство в роли public изменить нельзя.

На уровне базы данных учетной записи сопоставляется пользователь (user). Для одной и той же учетной записи в различных базах данных сервера могут создаваться пользователи с разными именами.

Пользователи получают разрешения на работу с объектами базы данных или напрямую, или путем авторизации пользователя на выполнение одной из ролей уровня базы данных. По- следний способ является более предпочтительным и позволяет организовать управление досту- пом в соответствии с ролевой моделью, описанной в первой главе пособия.

Список ролей уровня сервера предопределен, и новые создавать нельзя (табл. 2). Также есть предопределенный набор ролей уровня базы данных (табл. 3), но в этом случае имеется возможность создавать новые роли.

Роли уровня базы данных

Таблица 3

Роль	Описание возможностей
db_owner	Владелец базы данных, можно выполнять все действия по настройке и обслуживанию базы данных, а также удалять базу данных.
db_securityadmin	Управление составом ролей (кроме роли db_owner) и связанными с ними разрешениями.
db_accessadmin	Добавление и удаление пользователей базы данных.
db_backupoperator	Возможность создавать резервные копии базы данных.
db_ddladmin	Выполнение DDL-инструкций (создание, изменение, удаление объектов базы данных, таких как таблицы, представления и т. д.).
db_datareader	Чтение данных (SELECT) из всех пользовательских таблиц, представлений и функций.
db_denydatareader	Запрет на чтение данных (SELECT) из всех пользовательских таблиц, представлений и функций.
db_datawriter	Право добавлять, удалять или изменять данные во всех пользовательских таблицах.
db_denydatawriter	Запрет добавлять, изменять или удалять данные в пользовательских таблицах.
public	Роль по умолчанию, имеющаяся в каждой базе данных. Каждый пользователь БД авторизован на эту роль.

Болес подробную информацию об организации системы безопасности СУБД SQL Server можно получить из справочной системы TechNet: http://technet.microsoft.com/ru-ru/library/bb510589.aspx.

ХОД РАБОТЫ

1. Используя указанную преподавателем доменную или локальную учетную запись Windows, с помощью SQL Server Management Studio подключитесь к используемому экземпляру SQL Server. Проверьте установленный на сервере режим аутентификации.

2. В окне Object Explorer (по умолчанию — левая часть окна Management Studio) откройте список учетных записей (logins). На выполнение каких серверных ролей авторизована используемая вами учетная запись?

3. В каких базах данных сервера вашей учетной записи сопоставлены пользователи? На выполнение каких ролей они авторизованы?

4. В среде Management Studio создайте новую базу данных. Откройте список пользователей и ролей. Убедитесь, что учетная запись, под которой вы работаете, сопоставлена пользо- вателю dbo, авторизованному на роль db owner.

5. Используя приведенный ниже скрипт, создайте в базе данных таблицы. Перед гем как запустить скрипт, уберите символы комментария («--») из первой строки и после ключевого слова use укажите имя вашей базы данных.

```
use MyTestl GO
CREATE TABLE dbo.Book (
book_id int IDENTITY (1, 1) primary key,
Title varchar(50) NOT NULL, —название книги Author varchar(50), — автор Publisher var- char(50), — издательство [Year] smallint) — год издания GO
CREATE TABLE dbo.Status (
Status_id int IDENTITY (1, 1) primary key, Status_name varchar(50) NOT NULL ) — статус: выдана, в библиотеке и т.д.
GO
CREATE SCHEMA libr GO
CREATE TABLE libr.Book_in_lib (
lib_id int primary key , —номер экземпляра book_id int references dbo.Book , status_id int references dbo.[Status])
```

Обратите внимание, что приведенный скрипт создает не только три таблицы, но и схему libr. В SQL Server схема является контейнером логического уровня, к которому относятся объ- екты базы данных. Во вновь созданной БД уже будет несколько схем: dbo, sys, information_schema и т. д. Схема dbo — это схема по умолчанию для новых пользовательских объектов, sys и infor- mation schema используются системными объектами. Оператором CREATE SCHEMA в БД можно создавать новые схемы.

Защищаемым объектом, на действия с которым пользователю предоставляются разрешения, может быть база данных, схема или объект базы данных. Определенное для схемы разрешение неявным образом распространяется на все объекты схемы, разрешение для базы данных — на все схемы и объекты этих схем.

6. Для указанной преподавателем учетной записи SQL Server (при самостоятельном вы- полнении работы создайте учётную запись Windows и учётную запись SQL Server для нее) со- здайте пользователя в вашей базе данных, в качестве схемы по умолчанию выберите dbo. В Management Studio это можно сделать из графического интерфейса (контекстное меню узла Security для выбранной БД, там New...-> User) или выполнив оператор CREATE USER. Напри- мер (если схема не указана, подразумевается dbo):

```
USE MyTestl до
CREATE USER ns FOR LOGIN [HOME s]
```

Добавьте этого пользователя в роль db_datareader. Это можно сделать или через графический интерфейс или с помощью системной хранимой процедуры sp_addrolemember, первым параметром которой будет имя роли, а вторым — имя пользователя.

```
EXEC sp_addrolemember 'db_datareader', 'ns [1] Введите в таблицы тестовый набор данных.
```

Подключитесь к серверу с учетной записью другого пользователя. Убедитесь, что можно получить доступ к базе данных и читать записи из всех таблиц, а добавлять или изменять дан- ные нельзя.

7. Создадим новую роль уровня базы данных и добавим ей разрешение на удаление (DELETE), изменение (UPDATE) и добавление данных (INSERT) в объектах схемы libr. Добавим нашего пользователя к этой роли. Указанные действия надо выполнять с правами администратора или владельца базы данных. Как и в предыдущем случае, все это можно сделать в графическом интерфейсе или запуском скрипта.

```
CREATE ROLE libr_writer GO
GRANT INSERT, UPDATE, DELETE ON SCHEMA :: libr TO
libr_writer
Go
EXEC sp_addrolemember 'libr_writer', 'ns'
```

Используемый в приведенном скрипте оператор GRANT позволяет предоставить разре- шения. Оператор DENY позволяет запретить выполнение каких-то действий, а оператор REVOKE отменяет установленные оператором GRANT или DENY настройки разрешений. Та- ким образом, у разрешения может быть три состояния: «разрешено», «запрещено», «не задано». Действие можно выполнить, только если оно разрешено непосредственно пользователю или одной из ролей, на которые он авторизован. Запрещение более приоритетно, чем разрешение: если пользователь авторизован на выполнение двух ролей, одной из них действие разрешено, а другой — запрещено, то пользователь это действие выполнить нс сможет. В SQL Server Management Studio можно просмотреть эффективные разрешения для пользователя (рис. 5.5).

Конкретный набор возможных разрешений зависит от типа объекта.

Выполните описанные действия. Убедитесь, что пользователь с ограниченными правами может изменять данные в таблице Book_in_lib, относящейся к схеме libr.

8. Иногда нужно предоставить пользователю права на изменение отдельных столбцов. Как отмечается в документации SQL Server, на столбец могут быть предоставлены только разрешения SELECT, REFERENCES и UPDATE.

   Например:

   ```
   GRANT UPDATE ON dbo.Book(Title) TO libr_writer
   ```

Выполните аналогичные действия в своей базе данных, проверьте, что пользователь по- лучил указанные разрешения.

9. Самостоятельно по справке ознакомьтесь с форматом оператора CREATE VIEW, осо- бое внимание обратите на задаваемые дополнительные параметры. Создайте представление, выбирающее из таблицы Book книги, изданные не ранее 2000 года. Предоставьте пользователю с ограниченными правами возможность изменять и добавлять подобные книги. Возможности изменять прочие записи таблицы и добавлять книги, изданные до 2000 года, он иметь не дол- жен.

Практическая работа №14 Мониторинг безопасности работы с базами данных

Цель работы: изучение мониторинга безопасности работы с базами данных

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Система безопасности SQL Server основана на концепции защищаемых объектов (securables), т.е. объектов, на которые можно назначать разрешения, и принципалов (principles), т.е. объектов, которым можно назначать разрешения. Принципалами могут быть логины на уровне сервера, пользователи и роли на уровне базы данных. Роли назначаются пользователям. Разрешения на доступ к объектам могут предоставляться как непосредственно пользователям,

так и через роли. Каждый объект имеет своего владельца, и права собственности также влияют на разрешения.

Общая схема системы безопасности SQL Server рис.15.

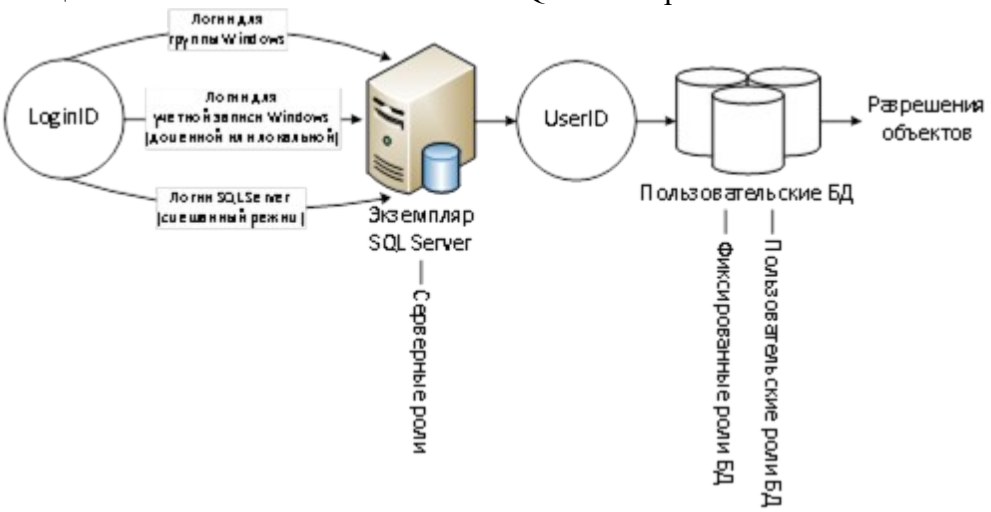

Рисунок 15

SQL Server использует двухэтапную схему аутентификации. На уровне сервера пользователь распознается по своему идентификатору (LoginID), который может быть либо именем входа SQL Server, либо группой или учетной записью Windows. После входа на сервер пользо- ватель получает те права, которые были назначены ему администратором на уровне сервера, в частности с помощью фиксированных серверных ролей. Если пользователь принадлежит роли sysadmin, то он имеет полный доступ ко всем функциям сервера, а также ко всем базам данных и объектам на нем.

Для получения доступа к базе данных логин пользователя должен быть сопоставлен с соответствующим ему идентификатором пользователя (UserID), который специфичен для каж- дой базы данных. Вполне возможна ситуация, когда пользователь был распознан в SQL Server, но у него нет доступа ни к одной из баз данных. Также возможно и обратное: пользователю открыт доступ к базам данных, но он не был распознан сервером. Перемещение базы данных и ее разрешений на другой сервер без параллельного перемещения имен входа сервера может привести к возникновению таких "осиротевших" пользователей.

На уровне базы данных пользователю может быть предоставлен определенный набор разрешений с помощью назначения ему фиксированных ролей базы данных. Все пользователи автоматически становятся членами стандартной роли public, у которой по умолчанию нет никаких разрешений. Пользовательские роли - это дополнительные роли, служащие в качестве групп. Роли может быть разрешен доступ к объектам базы данных, а пользователю могут быть назначены роли.

Разрешения к объектам назначаются с помощью инструкций GRANT (предоставить), REVOKE (отозвать) и DENY (запретить). Запрет привилегии замещает собой ее предоставление, а предоставление привилегии замещает собой ее отзыв. Пользователю может быть предоставлено множество разрешений к объекту (индивидуальных, наследованных от роли, обеспеченных принадлежностью к роли public). Если какая-либо из этих привилегий запрещена, для пользователя блокируется доступ к объекту. В противном случае, если какая-либо из привилегий предоставляет разрешение, пользователь получает доступ к объекту.

Разрешения объекта достаточно детализированы. Существуют отдельные разрешения для каждого из возможных действий (SELECT, INSERT, UPDATE, RUN и т.д.) над объектом.

Выбор типа логина и настройка режима аутентификации

SQL Server поддерживает два типа логинов (имен входа):

логин Windows (логин для локальной учетной записи Windows, логин для доменной учетной записи Windows, логин для группы Windows);

логин SQL Server.

При использовании логинов Windows в системные таблицы базы данных master записы- вается информация об идентификаторе учетной записи или группы Windows (но не пароль). Аутентификация (т. е. проверка имени пользователя и пароля) производится обычными сред- ствами Windows при входе пользователя на свой компьютер.

При использовании логина SQL Server пароль для этого логина (точнее, его хэширован- ное значение) хранится вместе с идентификатором логина в базе данных master. При подклю- чении пользователя к серверу ему придется указать имя логина и пароль.

Предпочтительный вариант логина для пользователя - это логин Windows, при этом не для учетной записи, а для группы (лучше всего для локальной доменной группы). Преимуществ у такого решения множество:

пользователю достаточно помнить один пароль - для входа на свой компьютер;

повышается уровень защищенности SQL Server. Это происходит, по крайней мере, за счет того, что пароль не будет передаваться по сети открытым текстом, как это происходит по умолчанию при использовании команд CREATE LOGIN и ALTER LOGIN. Кроме того, хэши Windows более защищены, чем хэши логинов SQL Server;

проверка при входе пользователя производится быстрее.

Эти преимущества справедливы для любых логинов Windows: как для учетных записей, так и для групп. Но при использовании логинов для групп Windows появляются дополнитель- ные преимущества:

снижается размер системных таблиц базы данных master, в результате чего аутентификация производится быстрее. На одном сервере SQL Server вполне может быть несколько тысяч логинов для пользователей Windows или, что значительно удобнее, всего пара десятков логинов для групп;

значительно упрощается предоставление разрешений для новых учетных записей.

Использование логинов SQL Server может быть обусловлено следующей причиной: очень часто на предприятиях администрированием SQL Server и домена Windows занимаются разные люди, которым сложно согласовывать свои действия. Логины SQL Server позволяют администратору базы данных быть независимым от администратора домена. Кроме того, у логинов SQL Server есть и другие преимущества, которые принимаются во внимание разработчи- ками:

на предприятии вполне может не оказаться домена Windows (если, например, сеть построена на основе NetWare или UNIX);

пользователи SQL Server могут не входить в домен (например, если они подключаются к SQL Server из филиалов или через Web-интерфейс с домашнего компьютера).

Таким образом, выбор используемых типов логинов зависит от многих факторов и в каждом конкретном случае решение принимается индивидуально. Логины Windows - это удоб- ство и защищенность, логины SQL Server- это большая гибкость и независимость от админи- стратора сети.

При установке SQL Server одним из решений, которые следует принять, является выбор используемого режима аутентификации.

В режиме аутентификации Windows SQL Server полностью доверяет (делегирует) аутен- тификацию операционной системе.

В смешанном режиме аутентификация Windows и самого сервера сосуществуют незави- симо друг от друга.

Третьего варианта, в котором использование логинов Windows было бы запрещено, не предусмотрено: логины этого типа доступны всегда.

Установленный при инсталляции режим аутентификации можно изменить в утилите Management Studio выбрав нужный переключатель в группе «Серверная проверка подлинности» на странице «Безопасность» диалогового окна «Свойства сервера».

ХОД РАБОТЫ

Задание 1. Использование программы Windows SystemMonitor

1. ***Создайте базу данных NorthwindCopy.***Для этого восстановите резервную копию базы данныхNorthwindCopyиз файлаC:\MOC\2072a\Labfiles\L08\NorthwindCopy.bak.

Сконфигурируйте Windows System Monitor. Для этого выполните следующие действия:

запустите программу «Системный монитор» (Мой компьютер /Администрирование /Системный монитор);

на панели инструментов щелкните по кнопке «Добавить» (+);

в окне диалога добавьте счетчики, используя информацию табл.4. В конце закройте окно кнопкой «Закрыть».

Таблица 4

Объект	Счетчик	Выбрать вхождение из списка
SQL Server:Access Methods (Методы доступа)	FullScans/sec(Полных сканирований в сек)	
SQL Server:Access Methods	IndexSearch/sec(Индексных поисков в сек)	
SQL Server:Buffer Manager (Диспетчер буферов)	Buffer Cashe Hit Ratio (коэффициент кэшированных операций)	
SQL Server:Databases	Active Transactions (Активные транзакции)	NorthwindCopy
SQL Server:Databases	PercentLogUsed(% использования журнала транзакций)	NorthwindCopy
SQL Server:Databases	Transactions/sec (Транзакций/сек)	NorthwindCopy
SQL Server:Memory Manager (Диспетчер памяти)	LockBlocks(Препятствующие блокировки)	
SQL Server:SQL Statistics (Статистика SQL Server)	BatchRequests/sec(Пакетных запросов в секунду)	

2. Проведите имитацию деятельности сервера

Деятельность сервера будет имитировать программа C:\MOC\2072a\Labfiles\L08\Monitor.bat, которую следует вызвать из командной строки (Пуск/Выполнить).

3. Наблюдайте окно «Просмотр диаграммы» во время выполнения командных файлов. Запишите значения счетчиков. Опишите в отчете, какие тенденции Вы отметили.

4. Отслеживание использования памяти и процессора.

В окне системного монитора щелкните по кнопке «Новый набор счетчиков» и добавьте набор счетчиков в соответствии с табл.4.

Таблица 5

Объект	Счетчик	Выбрать вхождение из списка
Память	Обмен страниц/сек	

Память	Ошибок страниц/сек	
Процесс	% загруженности процессора	Sqlserver
Процесс	Ошибок страниц/сек	Sqlserver
SQL Server:Cache Manager (Диспетчер КЕШ-памяти)	CacheHitRatio(Коэффициент успешного обращения к КЕШ памяти	Adhoc SQL Plans
SQL Server:Memory Manager	Connection Memory(KB)	
SQL Server:Memory Manager	Total Server Memory (KB)	

5. Наблюдайте за окном «Просмотр диаграммы» во время выполнения программы Monitor.bat. Какие тенденции вы наблюдаете? Кнопкой на панели инструментов перейдите в режим отображения значений счетчиков. Скопируйте это окно в отчет, сравните значения счет- чиков с допустимыми, сделайте выводы.

6. Закройте все окна командной строки на панели задач. Счетчики программы Системный монитор должны отразить снижение активности на сервере.

7. Создайте журнал для записи показаний системного монитора через каждые 20 сек в течение 5-10 мин. Остановите запись и просмотрите журнал.

Практическая работа №15 Установка приоритетов

Цель работы: изучить способы установки при оритетов

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Рассмотрим настройки параметра конфигурации сервера priority boost в SQL Server с помощью среды SQL Server Management Studio или Transact-SQL. С помощью параметра priority boost задается, должен ли Microsoft SQL Server выполняться с большим приоритетом в Microsoft Windows по сравнению с остальными процессами на том же компьютере. Если установить этот параметр в значение 1, SQL Server выполняется в планировщике Windows или Windows Server R2 с базовым приоритетом 13. Значением по умолчанию является 0, что соответствует базовому значению приоритета 7.

В будущей версии Microsoft SQL Server этот компонент будет удален. Избегайте исполь- зования этого компонента в новых разработках и запланируйте изменение существующих при- ложений, в которых он применяется.

Настройка параметра повышения приоритета с помощью:

Ограничения

Задание слишком высокого приоритета может лишить ресурсов операционную систему и сетевые функции, что может вызвать проблемы при завершении работы SQL Server и выпол- нении на сервере других административных задач.

Безопасность

Разрешения на выполнение хранимой процедуры sp_configure без параметров или только с первым параметром по умолчанию предоставляются всем пользователям. Для выполнения процедуры sp_configure с обоими параметрами для изменения параметра конфигурации или запуска инструкции RECONFIGURE необходимо иметь разрешение ALTER SETTINGS на уровне сервера. Разрешение ALTER SETTINGS неявным образом предоставлено предопреде- ленным ролям сервера sysadmin и serveradmin .

ХОД РАБОТЫ

Задание 1. Настроить параметр повышения приоритета

1. В обозревателе объектов щелкните правой кнопкой мыши сервер и выберите пункт Свойства. Щелкните узел Процессоры.

2. В разделе Потоки установите флажок Повысить приоритет SQL Server. Остановите и снова запустите SQL Server.

Использование Transact-SQL

Задание 2. Настройка параметра повышения приоритета

1. Установите соединение с компонентом Компонент Database Engine. На панели «Стан- дартная» нажмите Создать запрос.

2. Скопируйте следующий пример в окно запроса и нажмите кнопку Выполнить. В этом примере описывается использование процедуры sp_configure для задания значения параметра priority boost равным 1.

```
SQL
Копировать
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'priority boost', 1 ;
GO
RECONFIGURE;
GO
```

После настройки параметра priority boost, чтобы изменения вступили в силу, необходимо перезапустить сервер.

Практическая работа №16 Развертывание контроллеров домена

Цель работы: Освоить навыки развертывания контроллера домена и проверки его работоспособности.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Существование контроллера домена невозможно без службы каталогов Active Directory. Повышение роли рядового сервера до контроллера домена осуществляется установкой на него службы AD. Инструмент, который используется для установки (или удаления) Active Directory на сервер, называется Active Directory Installation Wizard (мастер установки Active Directory) – dcpromo.exe.

Сервер, на который осуществляется установка AD, должен удовлетворять целому ряду требований, перечисленных ниже:

Перед установкой на сервере должен быть установлен стек протоколов TCP/IP и для каждого из интерфейсов сервера выделен статический IP-адрес. Впоследствии администратор может изменить этот адрес и заново зарегистрировать в базе данных DNS доменное имя с уже новым адресом.

Для сервера должен быть установлен DNS-суффикс, соответствующий имени домена, для которого будет устанавливаться контроллер домена. Последнее требование является необя- зательным, если установлен флажок «Сменить основной DNS-суффикс при смене членства в домене» изменения имени компьютера в свойствах системы (Мой компьютер).

В этом случае система автоматически определит DNS-суффикс при включении сервера в состав некоторого домена.

Служба каталога может быть установлена на раздел диска с файловой системой NTFS. Это требование обусловлено соблюдением должного уровня безопасности, требующего разграничения доступа к файлам, непосредственно на уровне файловой системы (скажем, фай- ловая система FAT не предоставляет такой возможности). Кроме того, раздел, предназначен- ный для установки службы каталога, должен иметь как минимум 250 Мбайт свободного диско- вого пространства. С целью повышения производительности службы каталога администратор может разместить файлы хранилища каталога и журнала транзакций на отдельные физические диски. Это позволит избежать конкуренции операции ввода/вывода. Разумеется, в этом случае каждый из задействованных разделов должен быть отформатирован под NTFS.

Операция установки контроллера домена требует наличия у выполняющего ее пользователя определенных полномочий. Установка первого контроллера домена в лесе осуществляется на одиночном сервере, не являющимся частью какого-либо домена. В этой ситуации пользователь должен обладать полномочиями локального администратора на том сервере, на котором происходит установка. Если происходит установка первого контроллера в домене (в рамках уже существующего леса доменов), пользователь должен являться членом группы Enterprise Admins (Администраторы корпорации). В случае установки дополнительного контроллера в домене пользователь должен быть либо членом уже упомянутой группы, либо членом группы Domain Admins (Администраторы домена).

Прежде чем запустить на сервере мастер установки Active Directory, администратор дол- жен проверить настройки стека протоколов TCP/IP для данного компьютера, обратив, в первую очередь, внимание на параметры службы DNS-клиента. Одним из важнейших параметров в этой ситуации является адрес предпочитаемого DNS-сервера (preferred DNS server, рис. 16).

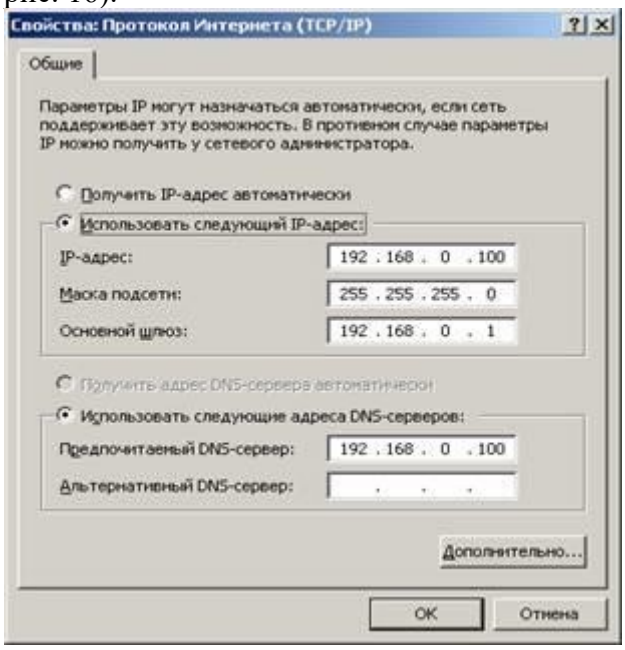

Рисунок 16

Именно указанный в этом параметре сервер будет использоваться мастером установки для диагностики пространства имен DNS, предшествующего созданию нового домена Active Directory, и поиска существующих носителей копий каталога. Этот же DNS-сервер впоследствии будет использоваться для регистрации доменного имени сервера. Ошибки, допущенные на этом этапе, могут привести к тому, что по окончании процедуры установки контроллер домена окажется неработоспособным. Типичны следующие ошибки:

настройки сервера, выбранного на роль контроллера домена, не содержат сведений о предпочитаемом DNS-сервере;

DNS-сервер, указанный в настройках будущего контроллера домена в качестве пред- почтительного, не является носителем требуемой зоны. Кроме того, возможна и другая ситуа- ция, когда указан сервер, являющийся дополнительным носителем зоны. Как следствие, этот DNS-сервер не может быть использован для динамической регистрации доменных имен.

ХОД РАБОТЫ

Если вы осуществляете установку первого контроллера домена в лесу доменов (фактически это означает первый этап развертывания в сети службы каталога), то вы должны предоставить возможность мастеру установки Active Directory установить на сервере службу DNS и произвести ее последующее конфигурирование. При этом в настройках стека протоколов TCP/IP данного сервера параметр Preferred DNS Server (Предпочитаемый DNS-сервер) должен указывать непосредственно на сам сервер. То есть после установки контроллера домена все ассоциированные с ним ресурсные записи будут зарегистрированы службой DNS, функционирующей на этом же сервере. Все последующие контроллеры домена должны указывать уже на существующие DNS-серверы (например, на первый установленный DNS-сервер).

Кроме того, необходимо проверить доступность DNS-сервера с компьютера, который выбран на роль контроллера домена. Различные проблемы с сетевыми компонентами могут привести к тому, что хотя DNS-сервер функционирует и успешно используется другими хостами, вновь устанавливаемый контроллер домена не имеет с ним сетевого соединения.

Процедура установки контроллера домена выполняется с помощью мастера установки Active Directory. Для запуска мастера можно воспользоваться командой dcpromo, которая запускается из меню Пуск|Выполнить (Start|Run).

Альтернативный вариант - выбрать команду Пуск|Программы|Администрирование|Мастер настройки сервера или Управление данным сервером (Пуск|Все программы|Администрирование |Управление данным сервером), в открывшемся окне последовательно нажать на ссылку «Добавить или удалить роль», затем – установка Active Directory.

Поскольку мы устанавливаем дополнительные контроллеры домена в существующий домен, в появившемся окне мастера установки для данного варианта установки следует установить переключатель **Добавочный контроллер домена в существующем домене (Additional domain controller for an existing domain)**и нажать кнопку Далее.

Введите имя, пароль и полное DNS-имя домена для пользовательской записи с административными правами в домене (это может быть член группы Администраторы или пользователь, имеющий права на подключение компьютеров к домену).

Введите полное DNS-имя существующего домена; при этом можно выбрать домен из списка существующих, нажав кнопку Обзор (Browse).

В следующих окнах мастера нужно указать дополнительные параметры (местоположение базы данных Active Directory, журналов регистрации событий, реплицируемого системного тома, а также пароль администратора для восстановления службы каталогов).

В появляющемся окне сводки проверьте правильность параметров и нажмите кнопку **Далее** и начнется процесс повышения роли сервера.

После перезагрузки компьютер будет работать как один из контроллеров указанного домена

Примечание

Если на сервере до начала процесса повышения роли была установлена служба DNS, то она полностью конфигурируется с использованием записей основного DNS-сервера.

Таким образом, легко получить резервный DNS-сервер, повысив отказоустойчивость сети. При этом предпочтительнее, если зоны DNS хранятся в Active Directory.

В процессе установки контроллера домена происходит наполнение каталога. В случае установки первого контроллера домена в лесе все содержимое каталога создается непосредственно мастером установки. Если в лесе создается новый домен, то мастер установки создает исключительно доменный раздел. Раздел схемы и каталога копируется с уже существующих контроллеров домена. В ситуации, когда администратор создает дополнительный контроллер в уже существующем домене, имеется два варианта наполнения каталога:

все содержимое каталога копируется с уже существующего контроллера домена;

содержимое каталога воссоздается из резервной копии каталога. Этот вариант целесообразно использовать в удаленных филиалах, соединенных с первичным контроллером домена низкоскоростными каналами связи.

Довольно часто возникает необходимость убедиться в том, процесс перехода сервера в новое качество успешно завершен. Если, например, служба репликации файлов (FRS – File Replication Service) не может успешно стартовать, она не инициализирует системный том, в результате чего служба Netlogon не может сделать общей (shared) системную папку SYSVOL. Как следствие папка NETLOGON также становится недоступной для общего доступа. Это при- водит к возникновению проблем с выполнением групповых политик, а также многих других проблем, в частности, с репликацией и аутентификацией. При возникновении неисправностей в Active Directory, перед тем, как выявлять проблемы, касающиеся соединений между контрол- лерами домена, аутентификации и т. д., вы в обоих случаях должны убедиться в том, что сер- веры Windows Server действительно являются контроллерами домена.

Существует несколько способов, посредством которых администратор может убедиться в том, что некоторый сервер на базе Windows Server по окончании операции повышения роли сервера (promotion) выполняет функции контроллера домена.

Раздел реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services должен содержать подраздел NTDS.

Введите в командной строке net accounts. Поле Computer role (Роль компьютера) должна содержать значение PRIMARY или BACKUP для контроллера домена. Для обычных серверов это поле имеет значение SERVERS.

Введите в командной строке net start. В списке запущенных сервисов должна присутствовать служба Kerberos Key Distribution Center (Центр распределения ключей Kerberos). Если эта служба не запущена на контроллере домена, механизм аутентификации может не работать.

Введите в командной строке nbtstat -n. Имя домена, имеющее тип <ic>, должно быть зарегистрировано (в поле Status указано значение REGISTERED).

Введите в командной строке net share. На сервере должны присутствовать общие папки SYSVOL (%SystemRoot%\SYSVOL\sysvol) и NETLOGON (%SystemRoot%\SYSVOL\sysvol\<DomainDNSName>\SCRIPTS).

С помощью утилиты Ldp.exe проверьте значение атрибута isSynchronized объекта RootosE. По окончании процесса повышения роли сервера система должна полностью синхронизировать все разделы каталога. Когда синхронизация закончена, атрибут isSynchronized при- нимает значение TRUE.

Используйте утилиту командной строки NLtest.exe. Эта утилита поставляется в составе пакета вспомогательных утилит Windows Server 2003 Support Tools.

С помощью утилиты Ntdsutil.exe можно подключиться к только что установленному контроллеру домена и проверить его способность отвечать на запросы LDAP. Утилита позволяет также проверить, знает ли контроллер о расположении ролей FSMO в своем домене.

Задание является общим для трех подгрупп и предполагает развертывание трех дополнительных контроллеров домена. Для выполнения работы необходимо знание основных принципов организации Active Directory.

Задание выполняется в несколько этапов:

1. С помощью утилиты Ipconfig определить имя домена, в котором будет работать контроллер домена.
2. Проверить, удовлетворяет ли компьютер, повышаемый до роли контроллера домена, описанным выше требованиям.
3. Проверить доступность первичного контроллера домена по сети (имя первичного кон- троллера домена можно определить с помощью утилиты Ipconfig).
4. Запустить мастер установки Active Directory и следуя инструкциям установить AD, повысив тем самым роль сервера до контроллера домена.
5. Убедиться, что контроллер домена функционирует корректно описанными выше спо- собами.
6. Сделать резервную копию AD.

Практическая работа 16. Мониторинг сетевого трафика

Цель работы: изучить инструменты по работе с анализаторами сетевого трафика

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

На начальном уровне перехват и анализ сетевого трафика осуществляется на отдельном хосте. Для этого используются программы «Анализаторы трафика», или «снифферы». Эти про- граммы позволяют осуществить перехват всего трафика по выбранному сетевому интерфейсу и его деинкапсуляцию до прикладного уровня. Как правило, они обладают средствами филь- трации и поиска в перехваченном наборе кадров. Наиболее известным кроссплатформенным решением является Wireshark.

Кроме них существуют стандартные консольные утилиты arp, netstat (Windows, Linux), ss, lsof и tcpdump (Linux). Как правило, подобные утилиты работают на сетевом уровне и выше.

К назначению средств анализа начального уровня относятся анализ текущих соединений на хосте и поиск неисправностей при сетевом взаимодействии.

ХОД РАБОТЫ

1. Установите на виртуальном хосте программу Wireshark.
2. Настройте виртуализацию сети в VirtualBox, так чтобы получать трафик приходящий на реальный сетевой адаптер (пропустите этот пункт если Wirwshark работает на реальном хо- сте).
4. Настройте перехват трафика, так чтобы он завершился после сбора 15 Мб (для увели- чения интенсивности генерации кадров открыть любой сайт в браузере).
5. Используя инструментарий статистики определите:
   a. Узел с максимальной активностью (по объему переданных данных),
   b. Узел осуществивший наибольшее количество широковещательных рассылок,
   c. Самый активный TCP-порт на хосте (по количеству переданных пакетов)
   d. Постройте на одной координатной сетке постройте графики интенсивности TCP и UDP трафика (пункт Io Graphs).
   e. Постройте граф связей только для пакетов, содержащих сообщения протокола HTTP (пункт Flow Graph)
6. Напишите фильтры которые выделяют из общего числа пакеты:
   a. Относящиеся к работе протоколов HTTP и FTP при работе в качестве клиента операционной системы на которой запущена среда виртуализации (или самого хоста если среда виртуализации не используется).
   b. Все кадры Ethernet, отправленные с сетевого интерфейса хоста, на котором запу- щена среда виртуализации (или самого хоста, если среда виртуализации не используется).
   c. Напишите фильтр, отбирающий только широковещательные сообщения. Опреде- лите назначение как минимум 3-х широковещательных рассылок разных протоколов.
   d. Определить адреса, на которые поступают данные кадры и пакеты для канального и сетевого уровня
   e. Напишите фильтры для каждой из трех широковещательных рассылок, выбран- ных в пункте 6-c.
   f. На основании собранной статистики определить, к какому типу коммутацион- ного оборудования подключен используемый компьютер (концентратор, коммутатор или маршрутизатор).
7. Запустите одновременно виртуальную машины Linux и Windows. Убедитесь, что на Windows есть ssh клиент putty, а на Linux telnet клиент. Если их нет, то установите клиенты.

Программа putty доступна на http://www.putty.org/. Telnet клиент на Linux доступен в ре- позитариях (для CentOS команда yum install telnet).

8. Настройте между ними внутреннюю сеть и установите на сетевых интерфейсах IP ад- реса из сети 192.168.0.0/24 (маска 255.255.255.0).
9. Запустите на Windows Telnet-сервер (консоль Службы / Services)

10. С Windows с помощью терминального клиента Putty подключитесь к SSH серверу на Linux.

11. С Linux с помощью telnet клиента подключитесь к Windows машине.

12. Используя утилиту netstat или lsof (для Linux) вывести все активнее (прослушиваемые) порты на обоих платформах. Используя утилиту netstat или ss (для Linux) все открытые соединения на обоих платформах.

13. С помощью команды tcpdump на Linux настроить вывод на экран содержимого пакетов от Windows-хоста по протоколу telnet.

14. Завершите ssh и telnet соединения. На одном из хостов запустите перехват трафика Wireshark и начните ssh и telnet сессии заново.

15. С помощью фильтров отберите трафик telnet и ssh. Сравните содержимое сообщений прикладного уровня в обоих случаях.

Литература

Основные источники:

1. Кумскова, И.А. Базы данных : учебник / Кумскова И.А. — Москва : КноРус, 2020. — 400 с. — (СПО). — ISBN 978-5-406-07467-1. — URL: https://book.ru/book/932493 (дата обращения: 21.10.2020). — Текст : электронный.
2. Кондрашов, Ю.Н. Язык SQL. Сборник ситуационных задач по дисциплине «Базы данных : учебно-практическое пособие / Кондрашов Ю.Н. — Москва : Русайнс, 2020. — 125 с. — ISBN 978-5-4365-4598-1. — URL: https://book.ru/book/935744 (дата обращения: 21.10.2020). — Текст : электронный.
3. Астахова, И.Ф. Объектные базы данных : учебное пособие / Астахова И.Ф., Борисенков Д.В., Киселева Е.И., Самойлов Н.К. — Москва : Русайнс, 2020. — 93 с. — ISBN 978-5-4365-5404-4. — URL: https://book.ru/book/936907 (дата обращения: 21.10.2020). — Текст : электронный.
4. Чулюков, В.А. Проектирование баз данных. Практический курс : учебное пособие / Чулюков В.А., Астахова И.Ф., Башарина С.О., Сидорова О.А. — Москва : Русайнс, 2020. — 163 с. — ISBN 978-5-4365-5748-9. — URL: https://book.ru/book/938011 (дата обращения: 21.10.2020). — Текст : электронный.
5. Сидорова, Н. П. Базы данных: практикум по проектированию реляционных баз данных : учебное пособие / Н. П. Сидорова. — Королёв : МГОТУ, 2020. — 92 с. — ISBN 978-5-4499-0799-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/149436 (дата обращения: 22.10.2020). — Режим доступа: для авториз. пользователей.
6. Каминский, В. Н. Базы данных : учебное пособие / В. Н. Каминский. — Санкт-Петербург : БГТУ "Военмех" им. Д.Ф. Устинова, 2017. — 106 с. — ISBN 978-5-906920-36-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/121826 (дата обращения: 22.10.2020). — Режим доступа: для авториз. пользователей.

Интернет ресурсы:

1. Электронный ресурс. URL: http://www.intuit.ru
2. Электронный ресурс. URL: http://www.edu.bpwin.ru

ПРИЛОЖЕНИЯ

Приложение 1. Основные математические функции MySQL

Обозначение	**Описание**
ABS(X)	Возвращает абсолютное значение аргумента *X*
ACOS(X)	Возвращает арккосинус аргумента *X* или *NULL*, если значение *X* не находится в диапазоне от -1 до 1
ASIN(X)	Возвращает арксинус аргумента *X* или *NULL*, если значение *X* не находится в диапазоне от -1 до 1
ATAN(X)	Возвращает арктангенс аргумента *X*
CEIL(X)	Принимает дробный аргумент *X* и возвращает первое целое число, находящееся справа от аргумента
COS(X)	Вычисляет косинус угла *X,* заданного в радианах
COT(X)	Вычисляет котангенс угла *X,* заданного в радианах
DEGREES(X)	Возвращает значение угла *X,* преобразованное из радиан в градусы
EXP(X)	Вычисляет значение e^x
FLOOR(X)	Принимает дробный аргумент *X* и возвращает первое целое число, находящееся слева от аргумента
LN(X)	Вычисляет натуральный логарифм числа *X*
LOG2(X)	Вычисляет логарифм числа *X* по основанию 2
LOG10(X)	Вычисляет десятичный логарифм числа *X*
MOD(M,N)	Возвращает остаток от деления целого числа *M* на целое число *N*
PI( )	Используется без аргументов. Возвращает число π
POW(X,Y)	Возвращает значение числа *X*, возведенного в степень *Y*
RADIANS(X)	Возвращает значение угла *X,* преобразованное из градусов в радианы
RAND(X)	Возвращает случайное значение с плавающей точкой в диапазоне от 0.0 до 1.0
ROUND(X)	Возвращает округленное до ближайшего целого значение числа *X*
SIGN(X)	Позволяет определить знак числа *X*. Возвращает -1, 0 или 1, если *X* отрицательно, равно нулю или положительно
SIN(X)	Вычисляет синус угла *X,* заданного в радианах
SQRT(X)	Вычисляет квадратный корень числа *X*
TAN(X)	Вычисляет тангенс угла *X,* заданного в радианах
TRUNCATE (X,D)	Возвращает число *X* с дробной частью, имеющей *D* знаков после запятой. Если количество знаков в *X* больше *D*, лишние разряды усекаются. Если меньше, то в конец числа добавляются нули

Приложение 2. Основные строковые функции MySQL

Обозначение	**Описание**
ASCII(str)	Возвращает значение ASCII-кода первого символа строки *str*. Для пустой строки возвращается значение 0

BIN(N)	Принимает десятичное число *N* и возвращает его двоичное представление
BIT_LENGTH(str)	Принимает строку *str* и возвращает ее длину в битах
CHAR(N1, N2, ...)	Принимает последовательность из ASCII-кодов и возвращает строку, построенную путем объединения соответствующих им символов
CHAR_LENGTH(str)	Принимает строку *str* и возвращает число символов в строке
CHARSET(str)	Возвращает имя кодировки, в которой представлена строка
COLLATION(str)	Возвращает порядок сортировки, установленный для кодировки аргумента *str*
CONCAT(str1, str2, ...)	Возвращает строку, созданную путем объединения всех аргументов, количество которых не ограничено. Если хотя бы один аргумент равен *NULL*, то возвращается значение *NULL*
CONCAT_WS(separator, str1, str2, ...)	Также объединяет аргументы *str1, str2* и т.д., помещая между ними разделитель *separator*
CONV(N, from_base, to_base)	Преобразует число *N* из одной системы счисления *from_base* в другую *to_base*. Параметры *from_base* и *to_base* могут принимать значения от 2 до 36
ELT(N, str1, str2, ...)	Возвращает *N*-ю строку из списка аргументов *str1, str2, ...* (для *N*=1 возвращается *str1*, для *N*=2 – *str2* и т.д.)
FIELD(str, str1, str2, ...)	Находит строку *str* в списке *str1, str2, ...*и возвращает номер строки в этом списке (нумерация начинается с 1)
FIND_IN_SET(str, str_list)	Ищет вхождение строки *str* в список *str_list* и возвращает номер строки в этом списке (нумерация начинается с 1). Параметр *str_list* – набор строк, разделенных запятыми
HEX(N_or_S)	Возвращает значение аргумента в виде шестнадцатеричного числа. Аргумент может быть как числом, так и строкой. Во втором случае функция переводит в шестнадцатеричное представление каждый символ строки и объединяет результат
INSERT(str, pos, len, new_str)	Возвращает строку *str*, в которой подстрока, начинающаяся с позиции *pos* и имеющая длину *len* символов, заменена подстрокой *new_str*
INSTR(str, substr)	Возвращает позицию первого вхождения подстроки *substr* в строку *str*
LEFT(str, len)	Возвращает *len* крайних левых символов строки *str*
LENGTH(str)	Возвращает длину строки *str*
LOCATE(substr, str [, pos])	Возвращает позицию первого вхождения подстроки *substr* в строку *str*. При наличии необязательного аргумента *pos* поиск начинается с позиции, указанной в этом аргументе
LOWER(str)	Возвращает строку *str* , записанную строчными символами
LPAD(str, len, padstr)	Возвращает строку *str*, дополненную слева строкой *padstr* до длины *len* символов. Если строка содержит более *len* символов, то она усекается до *len*
LTRIM(str)	Возвращает строку *str*, в которой удалены все начальные пробелы
MID(str, pos [, len])	Возвращает подстроку строки *str*, которая начинается с позиции *pos* и имеет длину *len* символов. Если параметр *len* не указывается, то подстрока возвращается, начиная с позиции *pos* и до конца строки *str*
OCT(N)	Принимает десятичное число *N* и возвращает его в восьмеричной системе счисления
ORD(str)	Возвращает значение ASCII-кода первого символа строки *str*. В отличие от функции *ASCII( )* корректно работает с многобайтными кодировками
REPEAT(str, count)	Возвращает строку, полученную из *count* повторений строки *str*
REPLACE(str, from_str, to_str)	Возвращает строку *str*, в которой все подстроки *from_str* заменены *to_str*
REVERSE(str)	Возвращает строку *str*, записанную в обратном порядке
RIGHT(str, len)	Возвращает *len* крайних правых символов строки *str*, или всю строку,

	если аргумент *len* равен *NULL* или меньше 1
RPAD(str, len, padstr)	Возвращает строку *str*, дополненную справа строкой *padstr* до длины *len* символов
RTRIM(str)	Возвращает строку *str*, в которой удалены все конечные пробелы
SPACE(N)	Возвращает строку, состоящую из *N* пробелов, или пустую строку, если *N* имеет отрицательное значение
SUBSTRING_INDEX(str, delim, N)	Возвращает подстроку строки *str*. Если параметр *N* имеет положительное значение, то функция находит *N*-е вхождение (отсчет слева) подстроки *delim* в строку *str* и возвращает всю часть строки, расположенную слева от подстроки *delim*. Если *N* имеет отрицательное значение, то функция находит *N*-е вхождение (отсчет справа) подстроки *delim* в строку *str* и возвращает часть строки, расположенную справа от подстроки *delim*
TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)	Удаляет из строки *str* расположенные в начале (в конце) символы, указанные в строке *remstr*. Если указано ключевое слово *LEADING*, удаляются расположенные в начале символы, если *TRAILING* – в конце, если *BOTH* – и в начале и в конце. Если ключевые слова не заданы, по умолчанию принимается *BOTH*. Если строка *remstr* не задана, то в качестве удаляемых символов выступают пробелы
UNHEX(str)	Является обратной функции *HEX( )* и интерпретирует каждую пару символов строки *str* как шестнадцатеричный код, который необходимо преобразовать в символ
UPPER(str)	Переводит все символы строки *str* в верхний регистр

Приложение 3. Основные функции даты и времени MySQL

Обозначение	**Описание**
ADDDATE(date, INTERVAL expr type)	Возвращает дату *date*, к которой прибавлен временной интервал, определяемый вторым параметром. Например, *ADDDATE('2009-03-20', INTERVAL 10 DAY)*
ADDTIME(expr1, expr2)	Возвращает результат сложения двух временных значений
CURDATE( )	Возвращает текущую дату в формате *'YYYY-MM-DD'*
CURTIME( )	Возвращает текущее время суток в формате *'hh:mm:ss'*
DATE(datetime)	Извлекает из значения *datetime* дату, отсекая часы, минуты и секунды
DATEDIFF(begin, end)	Вычисляет разницу в днях между датами *begin* и *end*
DATE_FORMAT(date, format)	Форматирует время *date* в соответствии со строкой *format*
DAY(date)	Возвращает порядковый номер дня в месяце (от 1 до 31)
DAYNAME(date)	Возвращает день недели в виде полного английского названия
DAYOFWEEK(date)	Возвращает порядковый номер дня недели. В западных странах неделя начинается с воскресенья, номер которого1.
DAYOFYEAR(date)	Возвращает порядковый номер дня в году (от 1 до 366)
EXTRACT(type FROM datetime)	Принимает дату и время суток и возвращает часть, определяемую параметром *type*. Например, *EXTRACT(YEAR FROM '2009-12-31 14:30:15')*
FROM_DAYS(N)	Принимает число дней N, прошедших с нулевого года, и возвращает дату в формате *'YYYY-MM-DD'*. Обычно используется совместно с функцией *TO_DAYS(date)*
HOUR(datetime)	Извлекает из значения *datetime* часы (от 0 до 23)
LAST_DAY(datetime)	Принимает дату и время суток и возвращает дату – последний день текущего месяца
MAKEDATE(year, dayofyear)	Принимает год *year* и номер дня в году *dayofyear* и возвращает дату в формате *'YYYY-MM-DD'*
MAKETIME(hour, minute, second)	Принимает час *hour,* минуты *minute* и секунды *second* и возвращает время суток в формате *'hh:mm:ss'*
MINUTE(datetime)	Извлекает из значения *datetime* минуты (от 0 до 59)
MONTH(datetime)	Возвращает числовое значение месяца года (от 1 до 12)
MONTHNAME(datetime)	Возвращает название месяца в виде полного английского названия
NOW( )	Возвращает текущую дату и время в формате *'YYYY-MM-DD hh:mm:ss'*
PERIOD_ADD(period, N)	Добавляет *N* месяцев к значению даты *period*. Аргумент *period* должен быть представлен в числовом формате *YYYYMMDD* или *YYMM*

Основные функции даты и времени MySQL (продолжение)

Обозначение	**Описание**
PERIOD_DIFF(period1, period2)	Вычисляет разницу в месяцах между двумя датами, представленными в числовом формате *YYYYMMDD* или *YYMM*

QUARTER(datetime)	Возвращает значение квартала года (от 1 до 4)
SECOND(datetime)	Извлекает из значения *datetime* секунды (от 0 до 59)
SUBDATE(date, INTERVAL expr type)	Возвращает дату *date*, из которой вычитается временной интервал, определяемый вторым параметром. Например, *SUBDATE('2009-05-15', INTERVAL 5 DAY)*
SUBTIME(datetime, time)	Вычитает из величины *datetime* время *time*
TIME(datetime)	Извлекает из значения *datetime* время суток
TIMEDIFF(expr1, expr2)	Возвращает разницу между временными значениями *expr1* и *expr2*
TIMESTAMP(date, time)	Принимает в качестве аргумента дату *date* и время *time* и возвращает полный вариант в формате *'YYYY-MM-DD hh:mm:ss'*
TIMESTAMPADD(interval, int_expr, datetime_expr)	Прибавляет к дате и времени суток *datetime_expr* временной интервал *int_expr*, единицы измерения которого задаются параметром *interval*. Например, *TIMESTAMPADD(WEEK, 1, '2009-09-02')*
TIMESTAMPDIFF(interval, datetime_expr1, datetime_expr2)	Возвращает разницу между двумя датами *datetime_expr1* и *datetime_expr1*. Единицы измерения интервала задаются параметром *interval*. Например, *TIMESTAMPDIFF(MONTH, '2005-02-01', '2005-05-01')*
TO_DAYS(date)	Принимает дату *date* и возвращает число дней *N*, прошедших с нулевого года
WEEK(date)	Возвращает номер недели в году (от 0 до 53) для даты *date*. Предполагается, что неделя начинается с воскресенья
WEEKDAY(date)	Возвращает номер дня недели (0 – для понедельника, 1 – для вторника, 6 – для воскресенья) для даты *date*
YEAR(datetime)	Возвращает год из значения *datetime*
YEARWEEK(date)	Возвращает число в формате *YYYYWW*, представляющее год и номер недели (от 0 до 53) в году и соответствующее дате *date*