

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ
ПО ДИСЦИПЛИНЕ
«Проектирование и администрирование баз данных»
для студентов направления подготовки
01.04.02 Прикладная математика и информатика

Ставрополь, 2026

Предисловие

Цель освоения дисциплины: формирование у студентов профессиональных компетенций, относящихся к производственно-технологической и организационно-управленческой деятельности, направленной на разработку программного и информационного обеспечения компьютерных сетей, автоматизированных систем вычислительных комплексов, сервисов, операционных систем и распределенных баз данных, разработку архитектуры, алгоритмических и программных решений системного и прикладного про-граммного обеспечения, а также организацию разработки системного программного обеспечения.

Задачи освоения дисциплины:

- формирование знаний, умений и навыков в области описания и манипулирования данными современных СУБД, методами доступа к базам данных средствами системы программирования;
- выработка умений применять инструментальные средства для выполнения функций администратора базы данных;
- знакомство с основными методами и технологиями программирования баз данных для информационных систем разной архитектуры;
- понимание принципов и методов обеспечения целостности баз данных;
 - понимание обязанностей и функций администратора (группы администрирования) базы данных;
- углубление знаний в области инфологического, семантического и даталогического проектирования, языков описания и манипулирования данными для соответствующих моделей.

Дисциплина «Проектирование и администрирование баз данных» относится к дисциплинам части УП, формируемой участниками образовательных отношений.

Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы.

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ПК-2. Способен разрабатывать, отлаживать, модифицировать и поддерживать системное программное обеспечение	ИД-1 _{ПК-2} Разрабатывает программное и информационное обеспечение компьютерных сетей и распределенных баз данных.	Применяет основные методы и технологии программирования баз данных для информационных систем разной архитектуры. Владеет методами доступа к базе данных из систем программирования.
	ИД-2 _{ПК2} Модифицирует и поддерживает системное и прикладное программное обеспечение.	Демонстрирует способность проектировать, реализовывать, эффективно использовать и сопровождать базу данных.
ПК-3. Способен осуществлять планирование организации разработки и интеграции системного программного	ИД-1 _{ПК-3} Использует современные языки программирования и пакеты прикладных программ для реализации конкретных алгоритмов и математических моделей.	Демонстрирует владение современными методами, инструментами и технологиями разработки БД. Применяет языки манипулирования данными в современных СУБД. Демонстрирует владение инструментальными средствами для

обеспечения		выполнения функций администратора БД.
	ИД-2 _{ПКЗ} Осуществляет управление разработкой, разработку и поддержание информационных систем и программных комплексов.	Демонстрирует способность управлять учётными записями пользователей БД и защищать БД от несанкционированного доступа. Владеет методами поддержки целостности БД.

Практические работы могут выполняться с использованием СУБД Microsoft SQL, а также любой СУБД, поддерживающей стандарт ANSI/ISO SQL2.

Язык SQL - Structure Query Language (Язык структурных запросов) является одним из доминирующих языков, используемых в реляционных базах данных. В качестве стандартного языка баз данных он принят такими организациями как American National Standards Institute – ANSI (Американский национальный институт стандартов) и International Standards Organization – ISO (Международная организация стандартов).

SQL является непроцедурным языком, что позволяет ему обрабатывать наборы записей и автоматизировать процесс управления данными. SQL не требует от пользователя спецификации метода доступа к данным, пользователь вообще не должен описывать, как должна выполняться та или иная операция над данными, он просто описывает, что должно быть сделано с данными, предоставляя право системе решать, как это сделать.

SQL включает в себя три основные составляющие:

Язык управления данными Data Definition Language (DDL)

Язык манипулирования данными Data Manipulation Language (DML)

Язык запросов к базе данных

Основная цель, которая ставится студенту при выполнении задания – практическое освоение всех этапов разработки надежной базы данных, как основы любой информационной системы. Практические работы выполняются индивидуально в соответствии с вариантом задания. Выполнив работу, студент оформляет отчет, который состоит из следующих разделов:

1 Тема и цель работы.

2 Условия задания.

3 Текст запросов (если требуется)

4 Результаты выполнения запроса.

5 Обоснование правильности разработанной программы.

6 Выводы.

При защите работы необходимо отвечать на вопросы и уметь пояснять работу.

ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА

Отчет должен быть выполнен печатным способом на одной стороне листа белой бумаги формата А4 через 1,15 интервала.

Цвет шрифта должен быть черным, высота букв, цифр и других знаков — не менее 1,8 мм (кегель не менее 12).

Шрифт Times New Roman.

Выравнивание – по ширине, абзац 1,25.

Текст отчета следует печатать, соблюдая следующие размеры полей: правое – 11,5 мм, верхнее и нижнее – 20 мм, левое – 30 мм.

Нумерация страниц – сквозная на весь отчет, в центре нижней части листа, арабской цифрой, без точки.

Титульный лист включают в общую нумерацию страниц отчета.

Номер страницы на титульном листе не проставляют

Разделы, подразделы, пункты и подпункты следует нумеровать арабскими цифрами и записывать с абзацного отступа, без точки.

Разделы, подразделы должны иметь заголовки. Пункты, как правило, заголовков не имеют. Заголовки должны четко и кратко отражать содержание разделов, подразделов.

Заголовки разделов, подразделов и пунктов следует печатать с абзацного отступа с прописной буквы без точки в конце, не подчеркивая.

Нумерация иллюстраций, таблиц, формул – сквозная или по разделам, арабскими цифрами

•

Оглавление

1. Сбор и анализ информации
2. Проектирование реляционной схемы базы данных в среде СУБД
3. Приведение БД к нормальной форме 3НФ
4. Создание базы данных в среде разработки
5. Организация локальной сети. Настройка локальной сети
6. Установка и настройка SQL-сервера
7. Экспорт данных базы в документы пользователя
8. Импорт данных пользователя в базу данных
9. Выполнение настроек для автоматизации обслуживания базы данных
10. Мониторинг работы сервера
11. Выполнение резервного копирования
12. Восстановление базы данных из резервной копии
13. Реализация доступа пользователей к базе данных
14. Мониторинг безопасности работы с базами данных.
15. Установка приоритетов
16. Развертывание контроллеров домена. Мониторинг сетевого трафика

Практическая работа № 1. Сбор и анализ информации.

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 организацию данных, призвана информационная модель реального мира, на которой и основана система автоматизированной обработки данных.

Чтобы понять процесс построения информационной модели, необходимо знать ряд терминов, 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 строковое выражение, образованное из значений нескольких полей (составной ключ), по которому можно определить значения других полей для одной или нескольких записей таблицы. На практике для использования ключей создаются индексы - служебная информация, содержащая упорядоченные 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 уникальный (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, дочерней) 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 ссылочной целостности данных в операциях изменения первичных ключей (возможно каскадное изменение данных), удалении записей в главной таблице (каскадное удаление в дочерних таблицах) и добавлении записей или изменении данных в дочерних таблицах.

Репликация базы данных - создание копий базы данных (реплик), которые могут обмениваться 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 вообще.

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 отсутствует.

Разрешение на возможность существования значения Null может задаваться для отдельных полей таблицы.

Администратор базы данных (АБД) – лицо, или группа лиц, уполномоченных на ведение БД (модификация структуры и содержания БД, активизация доступа пользователей, выполнение 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, выполненное 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 внешних запоминающих устройствах СУБД.

Нормализация – представление сложных структур данных (документов) в виде двумерных таблиц (отношений).

Проектирование БД – упорядоченный формализованный процесс создания системы взаимосвязанных описаний - таких моделей предметной области, которые связывают (фиксируют) хранимые в базе данные с объектами предметной области, описываемые этими данными.

Распределенная БД – совокупность баз данных, которые обрабатываются и управляются 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 - человек, какой - либо предмет или населённый пункт, и абстрактным - событие, счёт покупателя или курс изучаемый студентами.

Классом объектов называется совокупность объектов, обладающих одинаковым набором 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 многопользовательским 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.

Клиент-сервер.

В этой архитектуре на выделенном сервере, работающем под управлением серверной операционной системы, устанавливается специальное программное обеспечение (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, порождает 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 формирование запросов к базе данных, передаваемых на выполнение серверу баз данных.

Сервер приложений может быть Web-сервером или специализированной программой (например, Oracle Forms Server).

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 двумерный массив или двумерную таблицу, а при создании сложных информационных моделей составит совокупность 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 количеством полей, типом данных.

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.

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.

Проектирование базы данных Жизненный цикл БД.

Как и любой программный продукт, база данных обладает собственным 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, поскольку 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 проанализирована 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 информационным потребностям фирмы.

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

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 долгосрочных 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.

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 получается каждый элемент данных, кем получается, где в дальнейшем используется, кем контролируется).

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 спецификации 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, технология структурного анализа и проектирования, диаграммы потоков данных и графики "вход — процесс — выход".

Поскольку системы с неадекватной или неполной функциональностью будут лишь раздражать пользователей, а чрезмерно увеличенный набор функциональных возможностей вызовет 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 анализируемой части предприятия концептуальной модели данных.

Построение ее осуществляется в определенном порядке: в начале создаются подробные модели пользовательских представлений данных; затем они интегрируются в концептуальную модель данных.

Концептуальное проектирование 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 связей, затем работа продолжается в виде серии нисходящих уточнений низкоуровневых сущностей, связей и относящихся к ним атрибутов.

Нисходящий подход демонстрируется в концепции модели "сущность -связь" (EntityRelationship model — ER-модель) — самой популярной технологии высокоуровневого моделирования 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 реляционные 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 проверке 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 выбрать один из двух основных подходов — централизованный подход или подход на основе интеграции представлений.

Отправным моментом при централизованном подходе, который применим только для не слишком сложных баз данных, является образование единого списка требований путем объединения 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 данных уточняется и преобразуется в логическую модель данных.

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

Построенная логическая модель данных в дальнейшем будет востребована на этапе физического проектирования, а также на этапе эксплуатации и сопровождения уже готовой системы, 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 ориентированной модели БД.

Следует учитывать, что на этой стадии разработки возможны возвраты на более ранние этапы ЖЦБД.

Например, решения, принимаемые на этапе физического проектирования 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 приложений.

Главные составляющие данного процесса — это проектирование транзакций и пользовательского 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 непротиворечивость 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 коррекции 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 транзакцию.

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 манипулировать 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 заключении трудового договора между фирмой и работником. Исполнитель получает вознаграждение, которое вычисляется как Стоимость выполнения заказа * Фиксированный процент вознаграждения.

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 информационных объекта: Исполнитель (Фамилия, Имя, Отчество, Процент вознаграждения) и Работа (Наименование, Стоимость работы).

Определим соответствующие таблицы ИСПОЛНИТЕЛИ и РАБОТЫ (рис.1).

Ни одно из первоначально заданных полей таблицы ИСПОЛНИТЕЛИ не определяет однозначно каждую запись таблицы, поэтому в таблицу введено поле Код исполнителя, значения в котором будут уникальными для каждого исполнителя.

Это поле является первичным ключом таблицы ИСПОЛНИТЕЛИ и будет определено как ключевое поле.

С этой же целью в таблицу РАБОТЫ введен первичный ключ Код работы.

ИСПОЛНИТЕЛИ					РАБОТЫ	
Код исполнителя					Код работы	
Фамилия					Наименование	
Имя					Стоимость работы	
Отчество						
Процент вознаграждения						

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 организации 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 проектирования 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 условий 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 предметной области системы и формирование взгляда на ПО с позиций сообщества будущих пользователей 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 зависящих 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 процессов, 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 разработчиков 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 основное 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 предыдущих и обладает достоинствами обоих.

Этап инфологического проектирования начинается с моделирования ПО.

Проектировщик разбивает её на ряд локальных областей, каждая из которых (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 подчинёнными (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 ключей 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 включают 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, использовать составной атрибут или разбивать его на компоненты, зависит от характера его обработки 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 ограничений 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, которые 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.

Примером тернарной связи является связь типа экзамен между сущностями ДИСЦИПЛИНА, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ.

Из последнего примера видно, что связь также может иметь атрибуты (в данном случае это Дата проведения и Оценка).

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 разнообразные 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 поступления заказа, дату его выполнения, список заказанных книг с указанием количества экземпляров.

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.

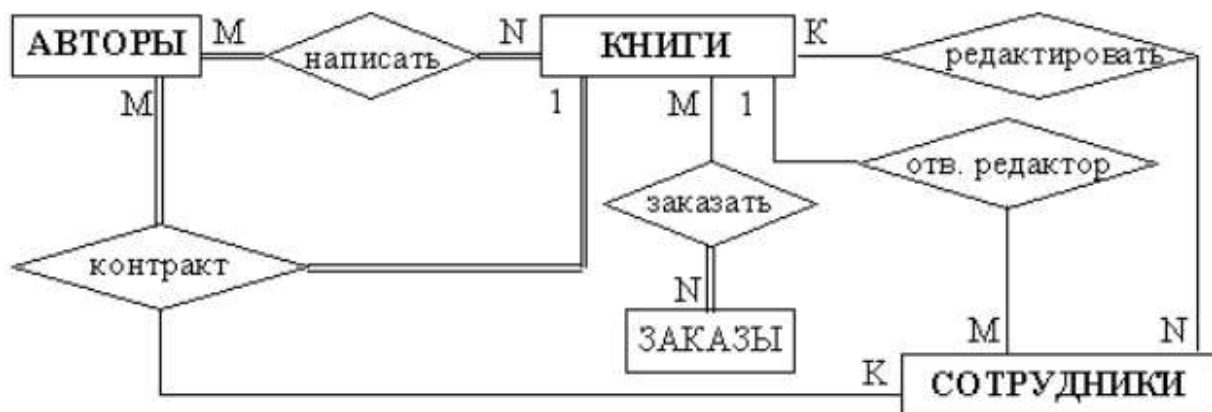


Рисунок 4

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 выдачи.

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 финансирования (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 производстве которых используется, потребляемые объёмы (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 последующая форма должна отвечать требованиям предыдущих плюс некоторые дополнительные требования.

Таблица, находящаяся в первой нормальной форме должна отвечать следующим требованиям: - таблица не должна иметь повторяющихся записей; - в таблице должны отсутствовать повторяющиеся группы полей.

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

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.

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 данных: целочисленные, вещественные, строковые, специальные типы данных для временных 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 нескольких 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 логическую основу для поддержания корректных значений данных в базе.

Такие ограничения целостности позволяют свести к минимуму ошибки, возникающие при обновлении и обработке данных.

Реляционная база данных представляет собой совокупность отношений, содержащих всю информацию, которая должна храниться в базе данных.

Однако пользователи могут воспринимать такую базу данных как совокупность таблиц.

Таким образом, реляционную базу данных можно рассматривать как хранилище данных, содержащих набор двумерных таблиц.

Набор средств управления подобным хранилищем называется реляционной системой управления базами данных.

Она может содержать утилиты, приложения, службы, библиотеки и другие приложения.

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 от 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 на 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 SCHEMA - 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 - отобразить 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 = "Пушкин" 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 из 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 представления - это таблица, которая не хранится постоянно в базе данных, а "возникает" в момент обращения к ней.

С точки зрения оператора SELECT и постоянно хранимые таблицы, и временные таблицы и представления выглядят совершенно одинаково.

Конечно, при реальном выполнении 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 (>, «ВЫПОЛНИТЬ» => в открытом окне набрать «Ехр» 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 измените параметр FILE.
FULL	N	Если FULL=Y, экспортироваться будет вся БД, включая сведения о табличных пространствах.
GRANTS	Y	Указывает, будут ли экспортироваться привилегии для экспортируемых 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 определения ограничений (первичный ключ, уникальный ключ) экспортируются, 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 используется 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 экспортированы.
OWNER		Список имен пользователей БД, объекты которых будут экспортированы.
USERID		Указывает имя и пароль пользователя, который осуществляет процесс экспорта. Формат параметра — «имя пользователя/пароль@сервер».

Формат параметра — «имя пользователя/пароль@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 существует 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 экспортировались первоначально.

Утилита Import может также преобразовывать данные, предоставленные 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 другую.

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 записи.

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.

Данные 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 завершающими символами, например, запятыми, пробелами, или заключены в ограничительные символы.

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 обрабатывать 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 пользователей (USER1) и (USER2).

Создайте новую роль.

Присвойте роли права подключаться, создавать 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 триггер 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 примените 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 городов.

Добавьте три записи в таблицу.

Удостоверьтесь, что триггер заполнил первичный ключ.

В поле «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 № 5 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 другие 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 определения 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 пользователей (USER1) и (USER2).

Создайте новую роль.

Присвойте роли права подключаться, создавать 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 триггер 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 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 примените курсорный цикл.

Вызовите хранимую процедуру.

6. Импортируйте схему пользователя student из файла в схему пользователя USER2. Подключитесь к учебной БД под учетной записью USER2. Напишите запрос, который бы возвращал 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 управляющий файл и импортируйте в таблицу городов данные.

Напишите запрос, который бы возвращал все записи таблицы городов.

Убедитесь, что после импорта появились новые записи.

Практическая работа № 6 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: Управление базой данных.

Нажмите кнопку Ок.

В окне БД появится небольшая миниатюра панели.

Перетащите 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.

Перетащите 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 названием форм – Группа и Простая форма по запросу.

10. Добавьте в категорию Отчеты меню Управление базой данных пункты с названиями 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 встроенных панелей инструментов.

Закройте окно Параметры запуска.

Закройте базу данных, затем повторно откройте.

Откроется окно базы данных, содержащее только одну строку пользовательского меню Управление 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 созданную таблицу, чтобы предотвратить ее удаление.

Макрос должен сам создавать таблицы с именами, 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 выбирать 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 записей в окне свойств формы на Ведомость 00. Для этого откройте форму в режиме конструктора 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 выполните команду ВИД/Условия или щелкните по кнопке инструментальной панели с соответствующим 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, СохранениеПодсказка.

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 выполнения этой макрокоманды.

В следующей строке выберите макрокоманду ОткрытьФорму.

Определите аргументы макрокоманды: Имя формы - Форма для макроса, Режим - Форма, Режим окна - Обычное.

Введите комментарий: Открыть форму на основе таблицы Ведомость 00. Следующая макрокоманда должна осуществить переход на первую строку таблицыформы 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 ведомости 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 Условие 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 переименования ведомости для группы 851 и дисциплины с кодом 3, а также для других групп и дисциплин.

При 3-х дисциплинах должно быть 3 макрокоманды на одну группу студентов.

Введите последнюю макрокоманду Закрывать 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 макрос.

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 № 7 Установка и настройка SQL-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 текущей производительности для обнаружения процессов, вызывающих неполадки, и постоянный 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 использование базы данных и добиться компромисса между такими конфликтующими нагрузками, 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 административные 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 статистики 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 Задание.

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, интегрируемыми 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 данных может открыть такие панели: 1. Overview (обзор).

На этой панели демонстрируется время обработки запросов процессором, 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 оперативной памяти и сервера, сети, компиляция и др.

В этой же панели администратор базы данных 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 сворачивает 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 № 8 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 технологию 6. Наберите и исполните следующую команду, чтобы создать полную резервную копию базы данных: `BACKUP DATABASE MMM TO DISK = 'C:\.....TEST\AW.BAK'` Ознакомьтесь с результатами запроса – какая информация обработана, сколько страниц, сколько файлов.

4. Внесите изменение в таблицу «Модель» базы данных MMM.

Добавьте одну запись (придумайте 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 страниц, 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 страниц, 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 страниц, 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, созданной вашей программой 11. Убедитесь, что после восстановления добавленных строк в таблице «Модель» нет.

Практическая работа №9. Восстановление базы данных из резервной копии

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

ХОД РАБОТЫ

1. Для восстановления базы данных из резервной копии следует щелкнуть правой клавишей мыши по 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:

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? При помощи SQL Server Configuration Manager определите, на основе каких сетевых библиотек клиент может подключаться к MS SQL Server

Установите соединение с SQL сервером.

На рабочей станции запустите SQL Server Management Studio и выберите из списка логическое имя сервера, запущенного на вашем компьютере.

Если нужного сервера нет в списке, то можно выбрать 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 расположенную 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) соответственно.

3. С помощью команды SELECT @@version определите и запишите 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 соединение 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.

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 подключиться 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 системе, тип аутентификации пользователей и максимальное количество пользователей, поддерживаемых сервером.

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

1. Создать базу данных с именем Stud__1 средствами СУБД MS SQL Server с журналом средствами SQL Server Management Studio и с именем Stud__2 средствами Query Editor и запишите в отчет результаты выполнения процедуры sp_helpdb

Для созданных вами БД .

2. Переименуйте созданную Вами базу данных Stud__1 в Stud_ 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 текущий момент.

4. Удалите созданную вами базу данных с именем Stud__2 и отобразите в отчете результат выполнения оператора удаления.

5. Отключить/подключить созданную вами БД Stud_ от сервера.

Если БД создавалась на жестком диске, то переместить ее на резервный носитель и отобразите в отчете результат выполнения оператора.

Практическая работа № 10. 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 может получить доступ отдельный пользователь.

После успешного подключения сервер определяет, 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 выполнить 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 процедур.

Чтобы выполнить по отношению к некоторому объекту некоторое действие, пользователь должен иметь соответствующее право доступа.

Например, если пользователь хочет выполнить 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 одинаковые 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 пользователей.

Опытный пользователь может выполнять в базе данных любые операции; обычный пользователь может модифицировать некоторые типы данных и обновлять данные; неквалифицированному пользователю обычно запрещается модифицировать любые типы данных.

Роль уровня приложения позволяет пользователю выполнять права некоторой роли.

Когда 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 данных.

Роль уровня приложения имеет смысл применять в среде, где пользователи делают запросы 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 использование 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 пароль хранятся самой СУБД, и ее же средствами выполняется проверка подлинности.

При использовании смешанной аутентификации егановится доступной административная учетная запись sa, которую рекомендуется переименовать и назначить ей надежный пароль.

Учетная запись авторизуется на выполнение одной из серверных ролей. Разрешено выполнять любые действия на сервере.

dbcreator 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.

Последний способ является более предпочтительным и позволяет организовать управление доступом 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.

Более подробную информацию об организации системы безопасности СУБД SQL Server можно получить из справочной системы TechNet: <http://technet.microsoft.com/ruru/library/bb510589.aspx>.

Роли уровня сервера

Таблица 2

Роль	Описание возможностей
sysadmin	Разрешено выполнять любые действия на сервере.
dbcreator	Разрешено создавать базы данных.
bulkadmin	Могут выполнять инструкцию BULK INSERT.
diskadmin	Позволяет управлять файлами на диске.
processadmin	Позволяет управлять подключениями, запускать и приостанавливать экземпляры 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.

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 пользователю dbo, авторизованному на роль db owner.

5. Используя приведенный ниже скрипт, создайте в базе данных таблицы.

Перед тем как запустить скрипт, уберите символы комментария («--») из первой строки и после ключевого слова use укажите имя вашей базы данных.

use MyTest1
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 varchar(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 объекты базы данных.

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.

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 information_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 выполнении работы создайте учётную запись Windows и учётную запись SQL Server для нее) создайте 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.

Например (если схема не указана, подразумевается dbo): USE MyTest1 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 данные 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 разрешения.

Оператор DENY позволяет запретить выполнение каких-то действий, а оператор REVOKE отменяет установленные оператором GRANT или DENY настройки разрешений.

Таким 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 не 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 получил указанные разрешения.

9. Самостоятельно по справке ознакомьтесь с форматом оператора CREATE VIEW, особое 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 должен.

Практическая работа №11. 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.

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 пользователь 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 каждой 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 записывается информация об идентификаторе учетной записи или группы Windows (но не пароль).

Общая схема системы безопасности SQL Server рис.15.

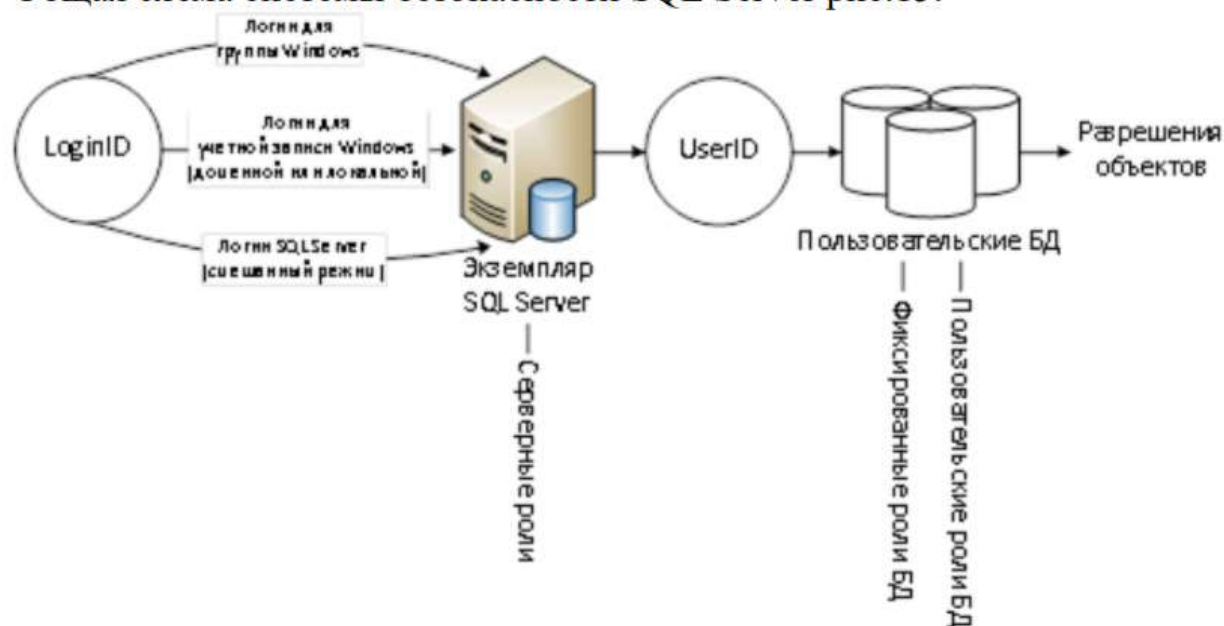


Рисунок 15

Аутентификация (т.е. проверка имени пользователя и пароля) производится обычными средствами Windows при входе пользователя на свой компьютер.

При использовании логина SQL Server пароль для этого логина (точнее, его хэшированное значение) хранится вместе с идентификатором логина в базе данных master.

При подключении 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 дополнительные 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 разработчиками: 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 удобство и защищенность, логины SQL Server - это большая гибкость и независимость от администратора 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) аутентификацию операционной системе.

В смешанном режиме аутентификация Windows и самого сервера сосуществуют независимо 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 данных NorthwindCopy из файла C:\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 «Заккрыть».

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:\МОС\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	% загрузки 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 счетчиков 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 №12. Установка приоритетов

Цель работы: изучить способы установки приоритетов

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 использования этого компонента в новых разработках и запланируйте изменение существующих приложений, 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 выполнении 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 предопределенным 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 «Стандартная» 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.

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

Методические указания
по организации и проведению
самостоятельной работы по дисциплине
«Проектирование и администрирование баз данных»
для студентов направления подготовки
01.04.02 «Прикладная математика и информатика»
Направленность (профиль):
«Математическое моделирование»

Методические рекомендации по организации самостоятельной работы

Учебный процесс в высшем учебном заведении в значительной степени строится на самостоятельной работе студентов, без которой трудно в полной мере овладеть сложным программным материалом и научиться в дальнейшем постоянно совершенствовать приобретенные знания и умения. Самостоятельная работа является внеаудиторной и предназначена для самостоятельного ознакомления студента с определенными разделами курса по рекомендованным педагогом материалам и подготовки к выполнению индивидуальных заданий по курсу. Целью самостоятельной работы студентов является:

- научить студента осмысленно и самостоятельно работать сначала с учебным материалом, затем с научной информацией, заложить основы самоорганизации и самовоспитания с тем, чтобы привить умение в дальнейшем непрерывно повышать свою квалификацию.

- закрепление, расширение и углубление знаний, умений и навыков, полученных студентами на аудиторных занятиях под руководством преподавателей;

- изучение студентами дополнительных материалов по изучаемым дисциплинам и умение выбирать необходимый материал из различных источников; воспитание у студентов самостоятельности, организованности, самодисциплины, творческой активности, потребности развития познавательных способностей и упорства в достижении поставленных целей.

Предлагаемый подход к освоению материала усиливает мотивацию к аудиторной и внеаудиторной активности, что обеспечивает необходимый уровень знаний по изучаемым дисциплинам и позволяет повысить готовность студентов к сдаче экзаменов.

Приступая к изучению новой учебной дисциплины, студенты должны ознакомиться с учебной программой, учебной, научной и методической литературой, имеющейся в библиотеке, получить в библиотеке рекомендованные учебники и учебно-методические пособия, завести новую тетрадь для конспектирования лекций и работы с первоисточниками. В ходе лекционных занятий вести конспектирование учебного материала. Обращать внимание на категории, формулировки, раскрывающие содержание тех или иных явлений и процессов, научные выводы и практические рекомендации. Желательно оставить в рабочих конспектах поля, на которых делать пометки из рекомендованной литературы, дополняющие материал прослушанной лекции, а также подчеркивающие особую важность тех или иных теоретических положений. Задавать преподавателю уточняющие вопросы с целью уяснения теоретических положений, разрешения спорных ситуаций. В ходе подготовки к семинарам изучить основную литературу, ознакомиться с дополнительной литературой, новыми публикациями в периодических изданиях: журналах, газетах и т.д. При этом учесть рекомендации преподавателя и требования учебной программы. Дорабатывать свой конспект лекции, делая в нем соответствующие записи из литературы, рекомендованной преподавателем и предусмотренной учебной программой. Подготовить тезисы для выступлений по всем учебным вопросам, выносимым на семинар. Готовясь к докладу или реферативному сообщению, обращаться за методической помощью к преподавателю. Составить план-конспект своего выступления. Принимать активное участие в обсуждении учебных вопросов: выступать с докладами, рефератами, обзорами научных статей, отдельных публикаций периодической печати, касающихся содержания темы занятия. В ходе своего выступления использовать технические средства обучения, доску и мел. С целью более глубокого усвоения изучаемого материала задавать вопросы преподавателю. После подведения итогов семинара устранить недостатки, отмеченные преподавателем. При подготовке к экзамену (в конце семестра) повторять пройденный материал в строгом соответствии с учебной программой, примерным перечнем учебных вопросов, выносящихся на зачет и содержащихся в данной программе. Использовать конспект лекций и литературу, рекомендованную преподавателем. Обратить особое внимание на темы учебных занятий, пропущенных студентом по разным причинам. При необходимости обратиться за консультацией и методической помощью к преподавателю.

1. Методические рекомендации для студентов по отдельным формам самостоятельной работы

Система вузовского обучения подразумевает значительно большую самостоятельность студентов в планировании и организации своей деятельности. Вчерашнему школьнику сделать это бывает весьма непросто: если в школе ежедневный контроль со стороны учителя заставлял постоянно и систематически готовиться к занятиям, то в вузе вопрос об уровне знаний вплотную встает перед студентом только в период сессии. Такая ситуация оборачивается для некоторых соблазном весь семестр посвятить свободному времяпрепровождению («когда будет нужно –

выучу!»), а когда приходит пора экзаменов, материала, подлежащего усвоению, оказывается так много, что никакая память не способна с ним справиться в оставшийся промежуток времени.

2. Правила и приемы конспектирования лекций

- Записывать информацию необходимо в соответствии с пунктами плана лекции, разбивая весь текст на логические блоки. Стоит использовать нумерацию, чтобы четко понимать последовательность изложения. Не будет лишним предусматривать в конспекте красную строку;
 - Сокращайте ключевые термины до заглавной буквы, предварительно выделив их еще на этапе написания заголовка. К примеру, фраза «нематериальные ресурсы в структуре активов» может быть записана как «НР в структуре А». Если какие-то обозначения совпадают, имеет смысл пометить их кружком или любым другим опознавательным знаком;
 - Стержневые понятия, которые используются при изучении какой-то дисциплины достаточно часто, можно заменять специальными сокращениями, которые будут актуальны в пределах конкретного конспекта. Если думаете, что забудете обозначения, то не стоит лениться записать их в конце тетради;
 - Сокращение длинных распространенных слов позволяет ускорить процесс конспектирования. К примеру, слово «государство» можно заменить сокращенным «гос-во» и т.п.
 - **Быстрое конспектирование** не обходится без использования ваших собственных значков важности. Придумайте их сами или возьмите на вооружение следующие:
 - «!» — важно;
 - «?» — под вопросом;
 - «У» — смотреть учебник;
 - «ЗВ» — например;
 - «R» — перепроверить.
 - Обозначения основных понятий имеет смысл записывать отдельным абзацем, при этом выделяя его любым способом. Подчеркивание цветным маркером облегчает процесс запоминания;
 - Скоростное конспектирование требует использования математических обозначений. Очевидно, что писать «+» вместо «плюс» и « Σ » вместо «сумма» в разы продуктивнее;
 - Не стоит пренебрегать общепринятыми аббревиатурами. Заменять термин типа «система управления базами данных» на «СУБД» является обязательным;
 - Периодизации и классификации лучше записывать в виде диаграмм и схем;
- Эти навыки помогут сделать ваши записи более понятными, а время на фиксацию информации сократить в разы.

Эти умения могут использоваться не только в вузе, но и при посещении курсов дополнительного профессионального образования, семинаров, тренингов и разного рода конференций. Зная, как правильно конспектировать, вы сможете легко освоить большой объем информации, сделав ее сжатой, простой и понятной.

Вы можете находить все более продуктивные методы фиксирования информации сами. Запомните, конспект является собой сжатую запись письменного или устного текста, которая создается с целью последующего личного использования. По этой причине одинаковых конспектов у разных людей быть не может. Не стоит полностью перенимать чьи-то методы, надо создавать свой уникальный.

3. Работа с книгой

При работе с книгой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги. Важно помнить, что рациональные навыки работы с книгой - это всегда большая экономия времени и сил.

Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу. Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода). При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить

аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь. Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались. Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и понятия. Такой лист помогает запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.

4. Основные виды систематизированной записи прочитанного:

1. Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;
2. Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;
3. Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;
4. Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;
5. Конспектирование – краткое и последовательное изложение содержания прочитанного.

Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.

5. Методические рекомендации по составлению конспекта:

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова. При записи не забудьте вынести справочные данные на поля конспекта;
2. Выделите главное, составьте план;
3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора;
4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.
5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.

Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

6. Приступая к работе, каждый студент должен принимать во внимание следующие положения.

Дисциплина (модуль) построена по тематическому принципу, каждая тема представляет собой логически заверченный раздел.

Лекционный материал посвящен рассмотрению ключевых, базовых положений курсов и разъяснению учебных заданий, выносимых на самостоятельную работу студентов.

Лабораторные работы проводятся с целью закрепления усвоенной информации, приобретения навыков ее применения при решении практических задач в соответствующей предметной области.

Самостоятельная работа студентов направлена на самостоятельное изучение дополнительного материала, подготовку к практическим занятиям, а также выполнения всех видов самостоятельной работы.

Для успешного освоения дисциплины, необходимо выполнить все виды самостоятельной работы, используя рекомендуемые источники информации.

8. Учебно-методическое и информационное обеспечение дисциплины

8.1. Перечень основной и дополнительной литературы, необходимой для освоения дисциплины (модуля)

8.1.1. Перечень основной литературы:

1. Кузнецов, С. Д. Введение в реляционные базы данных : учебное пособие / С. Д. Кузнецов. - Введение в реляционные базы данных, 2022-12-24. - Электрон. дан. (1 файл). - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. - 247 с. - электронный. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4497-0902-8, экземпляров неограничено
2. Карпова, Т.С. Базы данных. Модели, разработка, реализация Электронный ресурс : учебное пособие / Т.С. Карпова. - Базы данных. Модели, разработка, реализация, 2021-01-23. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 403 с. - Книга находится в базовой версии ЭБС IPRbooks.

8.1.2. Перечень дополнительной литературы:

1. Базы данных и экспертные системы: учебное пособие / сост.: А.Н. Макоха, И.А. Журавлёва. – Ставрополь: Изд-во СКФУ, 2020. – 248 с.
2. Волк, В. К. Базы данных. Проектирование, программирование, управление и администрирование Электронный ресурс / Волк В. К. : учебник. - Санкт-Петербург : Лань, 2020. - 244 с. - ISBN 978-5-8114-4189-1, экземпляров неограничено
3. Костюк, А. И. Администрирование баз данных и компьютерных сетей : учебное пособие / А. И. Костюк, Д. А. Беспалов. - Администрирование баз данных и компьютерных сетей, Весь срок охраны авторского права. - Электрон. дан. (1 файл). - Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2020. - 127 с. - электронный. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-9275-3577-4, экземпляров неограничено
4. Сидорова, Н. П. Базы данных: практикум по проектированию реляционных баз данных Электронный ресурс / Сидорова Н. П. : учебное пособие. - Королёв : МГОТУ, 2020. - 92 с. - ISBN 978-5-4499-0799-8, экземпляров неограничено
5. Тарасов, С. В. СУБД для программиста. Базы данных изнутри / С. В. Тарасов. - СУБД для программиста. Базы данных изнутри, 2022-05-25. - Электрон. дан. (1 файл). - Москва : СОЛОН-Пресс, 2018. - 320 с. - электронный. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-2-7466-7383-0, экземпляров неограничено
6. Швецов, В. И. Базы данных Электронный ресурс : Учебное пособие для СПО / В. И. Швецов. - Базы данных, 2022-07-04. - Саратов : Профобразование, 2019. - 219 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4488-0357-4, экземпляров неограничено

8.2. Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)

1. Методические рекомендации по организации самостоятельной работы студентов по дисциплине "Проектирование и администрирование баз данных" : Направление подготовки 01.04.02 - Прикладная математика и информатика. Профиль подготовки

«Математическое моделирование». Квалификация выпускника - магистр. Очная форма обучения. Изучается в 4 семестре. Учебный план 2016 г. - Ставрополь : СКФУ, 2016. - 34 с., экземпляров неограничено

2. Екимова, М.А. Методическое руководство по разработке электронного учебно-методического обеспечения в системе дистанционного обучения Moodle Электронный ресурс : учебно-методическое пособие / М.А. Екимова. - Омск : Омская юридическая академия, 2015. - 22 с. - Книга находится в базовой версии ЭБС IPRbooks., экземпляров неограничено

8.3. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля)

1. Поспелов Д.А. Становление информатики в России - <http://inf.1september.ru/1999/art/ocherk1.htm>

2. Электронно-библиотечная система «Лань» – URL: <https://e.lanbook.com/>

3. Электронно-библиотечная система «IPRbooks» – URL: <https://iprbookshop.ru/>.

4. AIPORTAL [Электронный ресурс] : портал искусственного интеллекта. – Режим доступа: <http://www.aiportal.ru>, свободный (дата обращения: 15.01.2023).

5. eLIBRARY.RU [Электронный ресурс] : научная электронная библиотека. – Режим доступа: <http://elibrary.ru/defaultx.asp>, свободный (дата обращения: 15.01.2023).

6. Единое окно доступа к образовательным ресурсам [Электронный ресурс] : федеральный портал. – Режим доступа: <http://window.edu.ru/>, свободный (дата обращения: 15.01.2023).

7. Интернет Университет Информационных технологий. [Электронный ресурс] : сайт. – Режим доступа: <http://www.intuit.ru/>, свободный (дата обращения: 15.01.2023).

8. Российский общеобразовательный портал [Электронный ресурс] : образовательный портал. – Режим доступа: <http://www.school.edu.ru/>, свободный (дата обращения: 15.01.2023).

9. Сервер Информационных Технологий [Электронный ресурс] : сайт. – Режим доступа: <http://citforum.ru/>, свободный (дата обращения: 15.01.2023).

10. Федеральный центр информационно-образовательных ресурсов [Электронный ресурс] : Единое окно доступа к образовательным ресурсам. – Режим доступа: <http://fcior.edu.ru>, свободный (дата обращения: 15.01.2023).

9. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень программного обеспечения и информационных справочных систем

При чтении лекций используется компьютерная техника, демонстрации презентационных мультимедийных материалов. На семинарских и практических занятиях студенты представляют презентации, подготовленные ими в часы самостоятельной работы.

Информационные справочные системы:

Информационно-справочные и информационно-правовые системы, используемые при изучении дисциплины:

Программное обеспечение:

1	PostgreSQL — свободная объектно-реляционная СУБД
2	Microsoft Office Standard 2013 (договор №01-за/13 от 25.02.2013, лицензия №61541869)
3	Microsoft Windows 10 Prof (договор № 27-за/16 от 02.08.2016, лицензия № 69366049)

10. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Материально-техническая база обеспечивает проведение всех видов дисциплинарной и междисциплинарной подготовки, лабораторной, научно-исследовательской работы обучающихся (переносной ноутбук, переносной проектор, компьютеры с необходимым программным обеспечением и выходом в интернет).

Лекционные занятия	1	Учебная аудитория для проведения занятий лекционного типа укомплектована специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории: ноутбук, проектор. Учебно-наглядные пособия в виде тематических презентаций, соответствующих рабочим программам дисциплин.
Практические занятия	1	Помещение, оснащено компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа к электронной информационно-образовательной среде.
Самостоятельная работа	1	Помещение для самостоятельной работы обучающихся, оснащено компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа к электронной информационно-образовательной среде.

11. Особенности освоения дисциплины (модуля) лицами с ограниченными возможностями здоровья

Обучающимся с ограниченными возможностями здоровья предоставляются специальные учебники, учебные пособия и дидактические материалы, специальные технические средства обучения коллективного и индивидуального пользования, услуги ассистента (помощника), оказывающего обучающимся необходимую техническую помощь, а также услуги сурдопереводчиков и тифлосурдопереводчиков.

Освоение дисциплины (модуля) обучающимися с ограниченными возможностями здоровья может быть организовано совместно с другими обучающимися, а также в отдельных группах.

Освоение дисциплины (модуля) обучающимися с ограниченными возможностями здоровья осуществляется с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья.

В целях доступности получения высшего образования по образовательной программе лицами с ограниченными возможностями здоровья при освоении дисциплины (модуля) обеспечивается:

1) для лиц с ограниченными возможностями здоровья по зрению:

- присутствие ассистента, оказывающий студенту необходимую техническую помощь с учетом индивидуальных особенностей (помогает занять рабочее место, передвигаться, прочесть и оформить задание, в том числе, записывая под диктовку),
- письменные задания, а также инструкции о порядке их выполнения оформляются увеличенным шрифтом,
- специальные учебники, учебные пособия и дидактические материалы (имеющие крупный шрифт или аудиофайлы),
- индивидуальное равномерное освещение не менее 300 люкс,
- при необходимости студенту для выполнения задания предоставляется увеличивающее устройство;

2) для лиц с ограниченными возможностями здоровья по слуху:

- присутствие ассистента, оказывающий студенту необходимую техническую помощь с учетом индивидуальных особенностей (помогает занять рабочее место, передвигаться, прочитать и оформить задание, в том числе, записывая под диктовку),

- обеспечивается наличие звукоусиливающей аппаратуры коллективного пользования, при необходимости обучающемуся предоставляется звукоусиливающая аппаратура индивидуального пользования;

- обеспечивается надлежащими звуковыми средствами воспроизведения информации;

3) для лиц с ограниченными возможностями здоровья, имеющих нарушения опорно-двигательного аппарата (в том числе с тяжелыми нарушениями двигательных функций верхних конечностей или отсутствием верхних конечностей):

- письменные задания выполняются на компьютере со специализированным программным обеспечением или надиктовываются ассистенту;

- по желанию студента задания могут выполняться в устной форме.