УЧЕБНО-НАУЧНО-ПРОИЗВОДСТВЕННЫЙ КОМПЛЕКС
«МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ КЫРГЫЗСТАНА»

«УТВЕРЖДЕНО»
Ректор НОУ УНПК «МУК»
к.т.н., доцент Савченков Е.Ю.
_____
«16» 10 2018 г.

БАКАЛАВРИАТ

Кафедра «Компьютерные информационные системы и управление»

Учебно-методический комплекс дисциплины

Технологии удаленного доступа к базам данных

Направление: **710100 «Информатика и вычислительная техника»**

Профиль: **Компьютерные информационные системы в бизнесе**

Академическая степень **- бакалавр**

Форма обучения **(очная)**

График проведения модулей 7-семестр

неделя	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
лекц. зан.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
прак./лаб. зан.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

«РАССМОТРЕНО»
Протокол заседания кафедры
«КИСиУ»
№ 2 от 16.10.2018
Зав. кафедрой д.т.н., проф. Миркин Е.Л.
_____

«СОГЛАСОВАНО»
Проректор по академ. вопросам
проф. Мадалиев М.М.
_____

Составитель _____ Маджинов А.Р.

Директор Научной библиотеки _____ Асанова Ж.Ш.

БИШКЕК 2018

Оглавление

АННОТАЦИЯ

Одной из областей компьютерных технологий, бурно развивающихся в настоящее время, является автоматизация обработки информации на основе программно-вычислительных комплексов, называемых информационными системами. Информационные системы оперируют большими объемами информации, хранящимися во внешней памяти. В настоящее время, подавляющее большинство информационных систем строится на основе реляционных баз данных.

Одним из наиболее популярных инструментов для создания и управления базами данных является Microsoft Access.

Изучая данный продукт, студенты смогут не только ознакомиться с возможностями Access, но и выработать свой подход к решению различных задач. Кроме того, они научаться разрабатывать собственные приложения баз данных.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ (МОДУЛЕЙ)

1. Пояснительная записка

1.1. Миссия и стратегия

Миссия НОУ УНПК "МУК" – подготовка международно - признанных, свободно мыслящих специалистов, открытых для перемен и способных трансформировать знания в ценности на благо развития общества.

Видение НОУ УНПК «МУК»- создание динамичного и креативного университета с инновационными научно-образовательными программами и с современной инфраструктурой, способствующие достижению академических и профессиональных целей.

Стратегии развития - модернизация образовательной деятельности университета – совершенствование образовательного процесса в соответствии с требованиями Болонского процесса.

1.2. Цель и задачи дисциплины (модулей)

Целью курса является предоставление студентам технических навыков, необходимых для программирования баз данных, на примере Microsoft SQL Server. Данная дисциплина дает практику создания баз данных в среде клиент-сервер под управлением Microsoft SQL Server и опыт разработки приложений масштаба предприятия.

Изучение данной дисциплины позволит студентам ознакомится с основными, современными технологиями доступа к источникам данных, а также научится программировать такие задачи, как создание и обработка удаленных данных.

Задачи дисциплины:

- создание у студентов упорядоченной системы знаний по проектированию баз данных, управлению и администрированию базами данных, основам структурированного языка запросов SQL, о методах сжатия больших информационных массивов, о реальных возможностях СУБД;
- ознакомление студентов с практикой создания информационной модели данных для конкретной предметной области и применения СУБД для создания приложений баз данных.

Дисциплина «Базы данных» относится к циклу ОПД.Ф.02 Цикл общепрофессиональных дисциплин. Федеральный компонент

1.3. Формируемые компетенции, а также перечень планируемых результатов обучения по дисциплине (модулю) (знания, умения владения), сформулированные в компетентностном формате.

Дисциплина (модуль) направлена на формирование следующих компетенций:
профессиональными (ПК):

- способен разрабатывать модели компонентов информационных систем, включая
- модели баз данных (ПК-4)
- способность разрабатывать компоненты программных комплексов и баз данных, использовать современные инструментальные средства и технологии программирования (ПК-5);
- способность инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем (ПК-11).

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

1. Знать: ПК-4, ПК-5, ПК-11.
2. Уметь: ПК-4, ПК-5, ПК-11.
3. Владеть: ПК-4, ПК-5, ПК-11.

1.4. Место дисциплины (модулей) в структуре ООП ВПО

Дисциплина (модуль) «ТДкБД» является частью цикла (блока) дисциплин учебного плана по направлению подготовки 710100 «Информатика и вычислительная техника», специальности

Компьютерные информационные системы в бизнесе. Для освоения дисциплины (модулей) необходимы компетенции, сформированные в ходе изучения следующих дисциплин и прохождения практик: Основы программирования, Математическая логика и теория алгоритмов.

2. Структура дисциплины (модулей)

Структура дисциплины (модулей) для очной формы обучения

Общая трудоемкость дисциплины составляет 3 кредита, 90 ч., в том числе аудиторная работа обучающихся с преподавателем 48 ч., самостоятельная работа обучающихся 42 ч.

№ п/п	*Раздел, Темы Дисциплины*	*Семестр*	*Неделя семестра*	*Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)*				*Формы текущего контроля успеваемости (по неделям семестра) Форма промежуточной аттестации (по семестрам)*
				лекции	*Сем. Заня/лаб. заня*	*СРС*	*СРСиП*	
	Раздел 1.							
1	Технологии удаленного доступа к системам баз данных, тиражирование и синхронизация в распределенных системах баз данных. Реляционные базы данных и введение в язык SQL.	1	1	1	2	1	1	опрос, проверка задания, посещаемость
2	Логическая архитектура Microsoft SQL Server. Активное администрирование и объектный интерфейс SQL Server. Таблицы. Представления. Хранимые процедуры. Правила. Умолчания. Системные базы данных Microsoft SQL Server. Производительность сервера. Репликация. Служба преобразования данных. Принципы безопасности. Роли сервера и роли базы данных. Резервное копирование. Транзакции.	1	2	1	2	1	1	опрос, проверка задания, посещаемость
3	Модели данных (иерархическая, сетевая, реляционная). Обзор современных СУБД. Уровни представления баз данных. Декомпозиция отношений, транзитивные зависимости. Метод: сущность – связь.	1	3	1	2	1	1	опрос, проверка задания, посещаемость
4	Различные представления о данных в базах	1	4	1	2	1	1	опрос, проверка задания,

	данных. Основные этапы проектирования баз данных. 1. Различные представления о данных в базах данных							посещаемость
5	Основные этапы проектирования баз данных. 1. Концептуальное (инфологическое) проектирование. 2. Логическое (даталогическое) проектирование. 3. Физическое проектирование.	1	5	1	2	1	1	Модуль 1
	Раздел 2.							
6	Модели данных. 1. Классификация моделей данных. 2. Сетевая и иерархическая модели..	1	6	1	2	2	1	опрос, проверка задания, посещаемость
7	Реляционная модель данных. 1. Основные определения 2. Операции над отношениями. 3. Реляционная алгебра 4. Проектирование реляционной базы данных. Понятие нормализации.	1	7	1	2	2	1	опрос, проверка задания, посещаемость
8	Целостность баз данных. 1. Общие понятия и определения целостности 2. Операторы DDL в языке SQL с заданием ограничений целостности 3. Средства определения схемы базы данных 4. Средства изменения описания таблиц и средства удаления таблиц 5. Понятие представления. Операции создания представлений	1	8	1	2	2	1	опрос, проверка задания, посещаемость
9	Табличные языки запросов. 1. Понятие табличного языка запросов. 2. Общая характеристика языка QBE	1	9	1	2	2	1	
10	Программное обеспечение работы с современными базами данных. 1. Основные задачи программного обеспечения баз данных 2. Проблемы создания и ведения реляционных баз данных 3. Понятие языка SQL и его основные части	1	10	1	2	2	1	Модуль 2
	Раздел 3.							
11	Язык SQL. 1. Общее представление об основных операторах языка SQL 2. Интерактивный режим работы с SQL (интерактивный SQL) 3. Использование языка SQL для выбора информации из таблицы. Использование SQL для выбора информации из нескольких таблиц 4. Использование SQL для вставки, редактирования и удаления данных в таблицах 5. Язык SQL и операции реляционной алгебры	1	11	1	2	2	1	опрос, проверка задания, посещаемость
12	Разработка приложений. 1. Механизм доступа к данным.Сравнение BDE и ADO 2. Создание базы данных MS Access 3. Практика работы с	1	12	1	2	2	1	опрос, проверка задания, посещаемость

	БД MS Access из Delphi							
13	Распределенные БД. 1. Модели "клиент-сервер" в технологии баз данных 2. Двухуровневые модели 3. Модель сервера приложений 4. Модели серверов баз данных	1	13	1	2	2	1	опрос, проверка задания, посещаемость
14	Банки данных. 1. Понятие банка данных. 2. Преимущества и недостатки банков данных.	1	14	1	2	2	1	опрос, проверка задания, посещаемость
15	Хранилища данных 1. Понятия о хранилищах 2. Типы хранилищ данных	1	15	1	2	3	2	Модуль 3
16	Консультация	1	16	1	2	0	0	

3. Содержание дисциплины (модулей)

№	Наименование раздела, темы дисциплины	Краткое содержание
1	Основные понятия	Развитие основных понятий представления данных. . Основные определения. Основные свойства баз данных. Классификация баз данных Системы управления базами данных (СУБД). Понятие системы управления базами данных. Классификация СУБД. Основные функции систем управления базами данных Различные архитектурные решения, используемые при реализации многопользовательских СУБД. Краткий обзор СУБД. Централизованная архитектура. Технология с сетью и файловым сервером (архитектура "файл-сервер"). Технология "клиент – сервер". Трехзвенная (многозвенная) архитектура "клиент – сервер". Краткий обзор СУБД. Настольные СУБД. Серверные СУБД
2	Модели данных	Классификация моделей данных. Сетевая и иерархическая модели. Реляционная модель данных. Основные определения. Операции над отношениями. Реляционная алгебра. Проектирование реляционной базы данных. Понятие нормализации Целостность баз данных. Общие понятия и определения целостности. Операторы DDL в языке SQL с заданием ограничений целостности. Средства определения схемы базы данных. Средства изменения описания таблиц и средства удаления таблиц. Понятие представления. Операции создания представлений. Табличные языки запросов. Понятие табличного языка запросов. Общая характеристика языка QBE
3	Язык SQL	Общее представление об основных операторах языка SQL Интерактивный режим работы с SQL (интерактивный SQL). Использование языка SQL для выбора информации из таблицы. Использование SQL для выбора информации из нескольких таблиц. Использование SQL для вставки, редактирования и удаления данных в таблицах. Язык SQL и операции реляционной алгебры Разработка приложений. Механизм доступа к данным. Сравнение BDE и ADO.

4. Конспект лекций

Тема 1. Назначение и компоненты (объекты) базы данных.

1. Назначение и компоненты (объекты) базы данных. Проектирование реляционной базы данных. Язык манипулирования данными для реляционной модели.
2. Создание базы данных. Создание таблицы. Первичный и внешний ключ.
3. Функциональны зависимости. Связывание таблиц. Виды связей. Индексирование.
4. Каскадирование

Тема 2. Запросы. Виды запросов.

1. Запросы. Назначение запросов. Виды запросов. Запросы на выборку.
2. Группировка и сортировка записей в запросах. Статистические функции.
3. Параметрические и перекрестные запросы.
4. Запросы действия. Обновление записей. Удаление записей. Добавление записей.
5. Запросы SQL. Инструкции языка SQL.

Тема 3. Различные архитектурные решения, используемые при реализации многопользовательских СУБД. Краткий обзор СУБД.

1. Централизованная архитектура.
2. Технология с сетью и файловым сервером (архитектура "файл-сервер").
3. Технология "клиент – сервер".
4. Трехзвенная (многозвенная) архитектура "клиент – сервер".
5. Краткий обзор СУБД. Настольные СУБД. Серверные СУБД.

Тема 4. Различные представления о данных в базах данных. Основные этапы проектирования баз данных.

1. Различные представления о данных в базах данных.
2. Основные этапы проектирования базы данных.

Тема 5. Основные этапы проектирования баз данных.

1. Концептуальное (инфологическое) проектирование.
2. Логическое (даталогическое) проектирование.
3. Физическое проектирование.

Тема 6. Модели данных.

1. Классификация моделей данных.
2. Сетевая и иерархическая модели..

Тема 7. Реляционная модель данных.

1. Основные определения
2. Операции над отношениями.
3. Реляционная алгебра
4. Проектирование реляционной базы данных. Понятие нормализации.

Тема 8. Целостность баз данных.

1. Общие понятия и определения целостности
2. Операторы DDL в языке SQL с заданием ограничений целостности
3. Средства определения схемы базы данных
4. Средства изменения описания таблиц и средства удаления таблиц
5. Понятие представления. Операции создания представлений

Тема 9. Табличные языки запросов.

1. Понятие табличного языка запросов.
2. Общая характеристика языка QBE

Тема 10. Программное обеспечение работы с современными базами данных.

1. Основные задачи программного обеспечения баз данных
2. Проблемы создания и ведения реляционных баз данных
3. Понятие языка SQL и его основные части

Тема 11. Язык SQL.

1. Общее представление об основных операторах языка SQL
2. Интерактивный режим работы с SQL (интерактивный SQL)
3. Использование языка SQL для выбора информации из таблицы. Использование SQL для выбора информации из нескольких таблиц

4. Использование SQL для вставки, редактирования и удаления данных в таблицах
5. Язык SQL и операции реляционной алгебры
Тема 12. Разработка приложений.
1. Механизм доступа к данным.Сравнение BDE и ADO
2. Создание базы данных MS Access
3. Практика работы с БД MS Access из Delphi
Тема 13. Распределенные БД.
1. Модели "клиент-сервер" в технологии баз данных
2. Двухуровневые модели
3. Модель сервера приложений
4. Модели серверов баз данных
Тема 14. Банки данных.
1. Понятие банка данных.
2. Преимущества и недостатки банков данных.
Тема 15. Хранилища данных
1. Понятия о хранилищах
2. Типы хранилищ данных
Тема 16. Безопасность данных.
1. Защита информации в базах данных.
2. Основные концепции обеспечения безопасности баз данных. 3. Реализация системы защиты в MS SQL Server 4. Проверка полномочий.

5. Информационные и образовательные технологии

Информационные и образовательные технологии

№ п/п	Наименование раздела	Виды учебной работы	Формируемые компетенции (указывается код компетенции)	Информационные и образовательные технологии
1	Раздел №1.	Лекция	ПК-4, ПК-5, ПК-11	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций
		Лабораторная работа	ПК-4, ПК-5, ПК-11	Дискуссия, Консультирование с разбором абстрактных ситуаций
		Самостоятельная работа	ПК-4, ПК-5, ПК-11	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты
2	Раздел №2.	Лекция	ПК-4, ПК-5, ПК-11	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций

		Лабораторная работа	ПК-4, ПК-5, ПК-11	Дискуссия, Консультирование с разбором абстрактных ситуаций
		Самостоятельная работа	ПК-4, ПК-5, ПК-11	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты
3	Раздел №3.	Лекция	ПК-4, ПК-5, ПК-11	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций
		Лабораторная работа	ПК-4, ПК-5, ПК-11	Дискуссия, Консультирование с разбором абстрактных ситуаций
		Самостоятельная работа	ПК-4, ПК-5, ПК-11	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты

6. Фонд оценочных средств для текущего, рубежного и итогового контролей по итогам освоению дисциплины (модулей)

6.1. Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины

№ п/п	Контролируемые разделы дисциплины (модулей)	Код контролируемой компетенции (компетенций)	Наименование оценочного средства
1	Разделы №1, №2, №3	ПК-4, ПК-5, ПК-11	опрос, выполнение лабораторных работ
2	Разделы №1, №2, №3	ПК-4, ПК-5, ПК-11	опрос, выполнение лабораторных работ
3	Разделы №1, №2, №3	ПК-4, ПК-5, ПК-11	опрос, выполнение лабораторных работ
4	Разделы №1, №2, №3	ПК-4, ПК-5, ПК-11	опрос, выполнение лабораторных работ
5	Разделы №1, №2, №3	ПК-4, ПК-5, ПК-11	опрос, выполнение лабораторных работ
6	Разделы №1, №2, №3	ПК-4, ПК-5, ПК-11	опрос, выполнение лабораторных работ
7	Разделы №1, №2, №3	ПК-4, ПК-5, ПК-11	опрос, выполнение лабораторных

			работ
8	Разделы №1, №2, №3	ПК-4, ПК-5, ПК-11	опрос, выполнение лабораторных работ
9	Разделы №1, №2, №3	ПК-4, ПК-5, ПК-11	опрос, выполнение лабораторных работ

6.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос - выполнение лабораторных работ - посещаемость	 1, 2, 3, 4, 5 недели 1, 2, 3, 4, 5 недели 1, 2, 3, 4, 5 недели	 8 баллов 6 баллов 0,2	 До 40 баллов До 30 баллов 10 баллов
Рубежный контроль: (сдача модуля)	5 неделя	100%×0,2=20 баллов	
Итого за I модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос - выполнение лабораторных работ - посещаемость	 6, 7, 8, 9, 10 недели 6, 7, 8, 9, 10 недели 6, 7, 8, 9, 10 недели	 8 баллов 6 баллов 0,2	 До 40 баллов До 30 баллов 10 баллов
Рубежный контроль: (сдача модуля)	10 неделя	100%×0,2=20 баллов	
Итого за II модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос - выполнение лабораторных работ - посещаемость	 11, 12, 13, 14, 15 недели 11, 12, 13, 14, 15 недели 11, 12, 13, 14, 15 недели	 8 баллов 6 баллов 0,2	 До 40 баллов До 30 баллов 10 баллов
Рубежный контроль: (сдача модуля)	15 неделя	100%×0,2=20 баллов	
Итого за III модуль			До 100 баллов
Итоговый контроль (экзамен)	Сессия	ИК = Бср × 0,8 + Бэкз × 0,2	

Полученный совокупный результат (максимум 100 баллов) конвертируется в традиционную шкалу:

Рейтинговая оценка (баллов)	**Оценка экзамена**
От 0 до 54	неудовлетворительно
от 55 до 69 включительно	удовлетворительно
от 70 до 84 включительно	хорошо
от 85 до 100	отлично

6.3.Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Текущий контроль (0 - 80 баллов)

При оценивании посещаемости, опроса и выполнении лабораторных работ учитываются:

- посещаемость (10 баллов)
- степень раскрытия содержания материала (25 баллов);
- изложение материала (грамотность речи, точность использования терминологии и символики, логическая последовательность изложения материала (20 баллов);
- знание теории изученных вопросов, сформированность и устойчивость используемых при ответе умений и навыков (25 баллов).

Рубежный контроль (0 – 20 баллов)

При оценивании контрольной работы учитывается:

- полнота выполненной работы (задание выполнено не полностью и/или допущены две и более ошибки или три и более неточности) – 10 баллов;
- обоснованность содержания и выводов работы (задание выполнено полностью, но обоснование содержания и выводов недостаточны, но рассуждения верны) – 5 баллов;
- работа выполнена полностью, в рассуждениях и обосновании нет пробелов или ошибок, возможна одна неточность – 5 баллов.

Итоговый контроль (экзаменационная сессия) – ИК = Бср × 0,8 + Бэкз × 0,2

При проведении итогового контроля обучающийся должен ответить на 3 вопроса (два вопроса теоретического характера и один вопрос практического характера).

При оценивании ответа на вопрос теоретического характера учитывается:

- теоретическое содержание не освоено, знание материала носит фрагментарный характер, наличие грубых ошибок в ответе (0 баллов);
- теоретическое содержание освоено частично, допущено не более двух-трех недочетов (10 баллов);
- теоретическое содержание освоено почти полностью, допущено не более одного-двух недочетов, но обучающийся смог бы их исправить самостоятельно (20 баллов);
- теоретическое содержание освоено полностью, ответ построен по собственному плану (30 баллов).

При оценивании ответа на вопрос практического характера учитывается:

- ответ содержит менее 20% правильного решения (0-9 баллов);
- ответ содержит 21-89 % правильного решения (10-39 баллов);
- ответ содержит 90% и более правильного решения (40 баллов).

6.4. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности..

Перечень вопросов:

1. Эволюция концепций обработки данных и развитие технологий обработки данных.
2. Гипертекстовые базы данных.
3. Мультимедийные базы данных.
4. Распределенная обработка данных.
5. Доступ к данным с помощью ADO.
6. Доступ к данным с использованием ODBC.
7. Интерфейс к базам данных на платформе Java.
8. Корпоративные серверы приложений. Corba-технология.
9. Коммерческие БД.
10. Объектно-ориентированные БД.
11. XML-серверы.
12. Публикация БД с использованием XML.
13. Базы данных и Интернет.
14. Примеры организации данных фактографических БД.
15. Примеры организации данных документальных БД.
16. Персональные (настольные) СУБД.
17. Промышленные СУБД.

7. Учебно-методическое и информационное обеспечение дисциплины

7.1.Список источников и литературы

- Литература:
  - Основная:

- Балдин К. В., Уткин В. Б. Информационные системы в экономике. Учебник / К.В. Балдин. - М.: Дашков и Ко, 2012. – 395с.
- Илюшечкин, В.М. Основы использования и проектирования баз данных: Учебное пособие / В.М. Илюшечкин. - М.: Юрайт, 2011 - 213с.
- Советов Б. Я. , Цехановский В. В. , Чертовской В. Д. Базы данных: теория и практика: учебник для бакалавров /Б.Я. Советов. - М.: ЮРАЙТ, 2011. - 459 с.
- Маркин А. В. Построение запросов и программирование на SQL. Учебное пособие / А.В. Маркин.- М.: Диалог-МИФИ, 2008. – 318 с.
- Мишенин А. И. Теория экономических информационных систем: Учебник 4-е изд., доп. и перераб. / А.И. Мишенин. - М.: Финансы и статистика, 2008. - 240 с.

  - Дополнительная:

- Баженова, И.Ю. Основы проектирования приложений баз данных: Учебное пособие / И.Ю. Баженова. - М.: Интернет-Университет Информационных Технологий, 2009 - 325с.
- Бекаревич, Ю.Б. Microsoft Access за 21 занятие студента / Ю.Б. Бекаревич, Н.В. Пушкина. - СПб.: БХВ - Петербург, 2005 - 544с.
- Бумфрей Ф., Диренцо О., Даккетт Й. XML. Новые перспективы WWW. – Издательство «ДМКПресс», 2006 – 688 с.
- Гайдамакин, Н.А. Автоматизированные информационные системы, базы и банки данных: Вводный курс / Н.А. Гайдамакин. – М.: Гелиос АРВ, 2002 – 368с.
- Голицына, О.Л. Системы управления базами данных: Учебное пособие / О.Л. Голицына. - М.: ФОРУМ-ИНФРА-М, 2006 - 432с.
- Гринвальд, Р. ORACLE: Справочник / Р. Гринвальд, Д. Крейнс. - СПб: Символ-Плюс, 2005 - 976с.
- Дейт, К.Дж. Введение в системы баз данных. : Пер.с англ. / К.Дж. Дейт. - Киев; М.; СПб.: Вильямс, 1999 - 848с.
- Керман, М.К. Программирование и откладка в Delphi: Учебный курс / М.К. Керман. – Киев; М.; СПб.: Вильямс, 2002 – 672с.
- Кириллов, В.В. Введение в реляционные базы данных / В.В. Кириллов, Г.Ю. Громов. - СПб: БХВ - Петербург, 2009 - 464с.

- Мезенцев, К.Н. Автоматизированные информационные системы: учебник / К.Н. Мезенцев. – М.: Академия, 2010 – 176 с.
- Мусина, Т.В. Visual FoxPro 9.0:Учебный курс / Т.В. Мусина. - Киев: Век+;СПб:Корона-Век, 2011 - 736с.
- Смирнов, С.Н. Обработка документов средствами Oracle: Практикум по XML и JDBC /С.Н. Смирнов. – М.: Гелиос АРВ, 2004 – 192 с.
  -

7.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимый для освоения дисциплины (модулей)

1. Википедия - http://ru.wikipedia.org
2. Интернет-портал образовательных ресурсов по ИТ - http://www.intuit.ru
3. Портал математических интернет-ресурсов - http://www.math.ru/
4. Портал математических интернет-ресурсов - http://www.allmath.com/
5. Портал ресурсов по математике, алгоритмике и ИТ - http://algolist.manual.ru/
6. http://www.iprbookshop.ru/
7. http://kyrlibnet.kg/ru/
8. http://biblioteka.kg/

8. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся.

8.1. Планы практических (семинарских) и лабораторных занятий. Методические указания по организации и проведению

- Тема 1 (2 ч.) Основные понятия.
  - Цель занятия: Основные понятия и определения информационных систем, основанных на базах данных, предметная область, информационное обеспечение
  - Форма проведения – дискуссия
  - Содержание занятия:
- 1. Развитие основных понятий представления данных.
- 2. Основные определения.
- 3. Основные свойства баз данных.
- 4. Классификация баз данных.
  -
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 2 (2 ч.) Системы управления базами данных (СУБД).
  - Цель занятия: Понятие СУБД. Назначение и основные компоненты системы баз данных. Основные функции СУБД
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
- 1. Понятие системы управления базами данных.
- 2. Классификация СУБД.
- 3.Основные функции систем управления базами данных.
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ
  -

- Тема 3 (2 ч.) Различные архитектурные решения, используемые при реализации многопользовательских СУБД. Краткий обзор СУБД
  Цель занятия: Обзор современных систем управления базами данных (СУБД): dbase, Visual dBase, Clipper, FoxPro и Visual FoxPro, Paradox, Access, ORACLE, Microsoft SQL Server, Sybase, Informix, Линтер.
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
- Причины появления СУБД. Этапы развития СУБД. Типовая организация современной СУБД. Требования к СУБД при выборе. Достоинства и недостатки СУБД.
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 4 (2 ч.) Различные представления о данных в базах данных. Основные этапы проектирования баз данных.

Цель занятия: Трехуровневая архитектура БД. Уровни представления баз данных: внешний, концептуальный, внутренний. Понятия схемы и подсхемы.

  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
- 1. Различные представления о данных в базах данных.
- 2. Основные этапы проектирования базы данных.
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 5 (2 ч.) Основные этапы проектирования баз данных
  - Цель занятия: Подходы к упрощению концептуальной модели данных. Исключение связей. Исключение атрибутов. Методика преобразования концептуальных структур данных в реляционные структуры
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:

    Концептуальное (инфологическое) проектирование.

    Логическое (даталогическое) проектирование.
    - 3. Физическое проектирование
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 6 (2 ч.) Модели данных.
  - Цель занятия: Трехуровневая архитектура БД. Уровни представления баз данных: внешний, концептуальный, внутренний. Понятия схемы и подсхемы
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
- 1. Классификация моделей данных.
- 2. Сетевая и иерархическая модели.
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 7 (2 ч.) Реляционная модель данных
  Цель занятия: Жизненный цикл базы данных. Основные этапы.

Достоинства и недостатки моделей. Примеры моделей.
- Форма проведения – разработка программного обеспечения
- Содержание занятия:

- 1. Основные определения
- 2. Операции над отношениями.
- 3. Реляционная алгебра
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 8 (2 ч.) Целостность баз данных
  - Цель занятия: Подходы к проектированию. Основные принципы проектирования Форма проведения – разработка программного обеспечения
  - Содержание занятия:

- 1. Общие понятия и определения целостности
- 2. Операторы DDL в языке SQL с заданием ограничений целостности
- 3. Средства определения схемы базы данных
- 4. Средства изменения описания таблиц и средства удаления таблиц
- 5. Понятие представления. Операции создания представлений
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ
  -

Тема 9 (2 ч.) Табличные языки запросов.
Цель занятия: Структурированный язык запросов SQL. Язык определения данных (DDL). Язык манипулирования данными для реляционной модели (DML). Основные понятия и компоненты языка SQL
- Форма проведения – разработка программного обеспечения
- Содержание занятия:

- 1. Понятие табличного языка запросов.
- 2. Общая характеристика языка QBE
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 10 (2 ч.) Программное обеспечение работы с современными базами данных
  - Цель занятия: Администрирование баз данных. Управление учетными записями и правами доступа
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
    - 1. Основные задачи программного обеспечения баз данных
    - 2. Проблемы создания и ведения реляционных баз данных
    - 3. Понятие языка SQL и его основные части
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 11 (2 ч.) Язык SQL.

- Цель занятия: Структурированный язык запросов SQL. Язык определения данных (DDL). Язык манипулирования данными для реляционной модели (DML). Основные понятия и компоненты языка SQL
- Форма проведения – разработка программного обеспечения
- Содержание занятия:
  - 1. Общее представление об основных операторах языка SQL
  - 2. Интерактивный режим работы с SQL (интерактивный SQL)
  - 3. Использование языка SQL для выбора информации из таблицы. Использование SQL для выбора информации из нескольких таблиц
  - 4. Использование SQL для вставки, редактирования и удаления данных в таблицах
  - 5. Язык SQL и операции реляционной алгебры
- Литература: []
- Материально-техническое обеспечение занятия: ЭВМ

- Тема 12 (2 ч.) Разработка приложений
  - Цель занятия: изучение разработки приложений на основе накопленных знаний в области SQL
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
    - 1. Механизм доступа к данным.Сравнение BDE и ADO
    - 2. Создание базы данных MS Access
    - 3. Практика работы с БД MS Access из Delphi
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 13 (2 ч.) Распределенные БД
  - Цель занятия: изучение распределения в условия работающей БД
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
    - 1. Модели "клиент-сервер" в технологии баз данных
    - 2. Двухуровневые модели
    - 3. Модель сервера приложений
    - 4. Модели серверов баз данных
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 14 (2 ч.) Банки данных.
  - Цель занятия: изучение понятий работы БД
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
    - Понятие банка данных.
    - Преимущества и недостатки банков данных.
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

- Тема 15 (2 ч.) Хранилища данных
  - Цель занятия: изучение взаимодействий хранилищ с сервером БД
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
    - 1. Понятия о хранилищах
    - 2. Типы хранилищ данных
  - Литература: []
  - Материально-техническое обеспечение занятия: ЭВМ

8.2. Методические указания для обучающихся по освоению дисциплины (модулей)

Вид работы	Содержание (перечень вопросов)	Трудоемкость самостоятельной работы (в часах)	Рекомендации
Модуль №1.			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	Стратегии разработки программных средств и систем и реализующие их модели жизненного цикла	5	[1, 2]
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	Выбор модели жизненного цикла для конкретного проекта	5	[1, 2]
Итого		10	
Модуль №2.			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	Классические методологии разработки программных средств	8	[1, 2, 3]

Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	Case-технологии технологии структурного анализа и проектирования программных средств	8	[1, 2, 4]
Итого		16	
Модуль №3.			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	Методология объектно-технологии ориентированного анализа и проектирования сложных систем	8	[1, 2, 5]
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы	Инструментальные средства разработки программного обеспечения	8	[1, 2, 5]
Итого		16	
Итого по дисциплине		42	
Вид работы	Содержание (перечень вопросов)	Трудоемкость самостоятельной работы (в часах)	Рекомендации
Раздел №1.			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		2	

Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		2	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		2	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		2	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		2	
Итого		10	
Раздел №2.			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ.		3	

Работа с учебной литературой. Написание программы			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Итого		15	
Раздел №3.			
Конспектирование материала на лекционных занятиях Выполнение заданий		3	

лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы			
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Конспектирование материала на лекционных занятиях Выполнение заданий лабораторных работ Выполнение тестовых и контрольных работ. Работа с учебной литературой. Написание программы		3	
Закрепление пройденного курса		5	
Итого		17	
Итого по дисциплине		42	

8.3. Методические рекомендации по подготовке письменных работ

Разработанное программное обеспечение должно быть предоставлено в скомпилированом виде, а так же в виде текстового файла, содержащего исходный код программы.

8.4. Иные материалы

Не предусмотрено.

9. Материально-техническое обеспечение дисциплины

Для изучения дисциплины, необходимо следующее оборудование: ЭВМ, проектор.

Требования к аудитории: компьютерный класс, имеющий ЭВМ в количестве идентичном количеству обучающихся, ЭВМ для преподавателя с подключенным проектором, наличие доски и средств для отображения/удаления информации на доске (мел/ветошь, маркер/губка).

10. Глоссарий

11. Приложения.

Глоссарий.

Авторизация (клиента) - authorization - предоставление полномочий на выполнение определенных действий в системе обработки данных на удаленном сервере.

База данных - database - совокупность данных, организованных по определенным вилам, предусматривающим общие принципы описания, хранения манипулирования, независимая от прикладных программ. Является информационной моделью предметной области. Обращение к базам данных осуществляется с помощью системы управления базами данных (СУБД).

Идентификатор - уникальное сочетание имени и пароля пользователя для обеспечения процесса его идентификации.

Репликация **программного обеспечения - software replication** - тиражирование программного обеспечения типового информационного комплекса с целью дальнейшего клонирования на его основе нового отраслевого портала или комплекса.

Транзакция 1 - transaction -короткий по времени цикл взаимодействия объектов, включающий запрос - выполнение задания - ответ. Обычно осуществляется в режиме диалога.

Транзакция 2 - transaction - объединение нескольких действий в одно действие, которое выполняется или не выполняется как единое целое.

API - application programming interface — интерфейс прикладного программирования. Набор подпрограмм, используемых приложениями для запросов и выполнения служб нижнего уровня операционной системой компьютера. Эти подпрограммы обычно выполняют задачи обслуживания, такие как управление файлами и отображение данных.

DLL - Dynamic Link Library - библиотека динамической компоновки. Средство операционной системы, позволяющее хранить подпрограммы (обычно выполняющие конкретную функцию или набор функций) отдельно от основной программы в виде файлов с расширениями .dll. Такие подпрограммы загружаются только тогда, когда их вызывает основная программа.

Семантика - semantics - раздел языкознания, исследующий с семиотических позиций смыслы и значения единиц языка (слов, предложений и др.), его выражений и логических форм, участвующих в его порождении, построении и изменении. В компьютерном программировании - определяет сущность кодов, команд, сообщений и охватывает совокупность операций, служащих для определения либо кодирования смысла данных.

Шлюз - Устройство, соединяемое с несколькими физическими сетями TCP/IP и обеспечивающее маршрутизацию и доставку пакетов IP между этими сетями. Шлюзы выполняют трансляцию между различными транспортными протоколами и форматами данных (например, IPX и IP) и обычно добавляются в сеть в основном для поддержания возможности трансляции.

Раздел 4.

Краткий конспект лекций.

Модуль №1.

Эволюция технологий доступа к данным.

Лекций - 5 ч.

Темы лекционных занятий:

- Эволюция технологий доступа к данным. Технология доступа к базам данных ODBC. Архитектура ODBC.
- Технологии доступа к базам данных DAO и RDO.
- Технология доступа к базам данных OLE DB. Компоненты OLE DB.
- Технология доступа к базам данных ADO. Объектная модель ADO.
- Развитие технологии доступа к базам данных ADO.NET. Архитектура ADO.NET.

Эволюция технологий доступа к данным.

Базы данных и их технологии развивались довольно быстро – от реляционных бах данных к нереляционным информационным хранилищам, таким, как электронная почта и файловые системы. Развитие баз данных идет в ногу со стремительными изменениями в технике. А с появлением клиент-серверных и многоуровневых архитектур разработчикам уже приходиться разбираться во всем многообразии технолгий доступа к базам данных.

Проследив развитие технологий, проще выбрать подходящую технологию и оптимизировать ее для своих целей.

Одной из основных задач, необходимых решить проектировщику или программисту ИС – это выбор технологии доступа к БД. Выбор технологии доступа к данным является одной из стратегических задач, от решения которой зависит, как производительность будущей системы и способность реализовать дополнительные функции, так и совместимость ее с другими программными платформами и технологиями, переносимость с одной платформы на другую.

Эволюция способов доступа к данным развивалась по пути от создания средств низкоуровневого доступа к данным, требующего от программиста глубокого понимания структуры хранимой информации и возможностей ее извлечения и записи, а также необходимости написания достаточно объемного программного кода, до создания высокоуровневых способов доступа, призванных упростить процедуру программирования операций извлечения данных из структур и записи изменений в эти структуры.

В настоящее время при работе с базами данных, существует возможность организации доступа к данным, как на низком уровне, так и на высоком уровне. У каждого из этих способов есть свои преимущества и недостатки.

Низкоуровневый доступ к данным (физическая реализация доступа).

Достоинство: позволяет обеспечить максимальное быстродействие при обмене информацией между клиентским приложением и самой БД.

Недостатки:

- сложность организации доступа, вследствие привязки к особенностям реализации СУБД и принципам организации доступа к данным именно этой СУБД;
- необходимость изучения большого объема технической документации, низкоуровневых команд и примеров их использования;
- нет переносимости программы с одной СУБД на другую, т.к. каждый производитель СУБД использует свои подходы, принципы и оригинальные решения в области доступа к данным.

Высокоуровневый доступ к данным (логика обработки данных). *(устраняет приведенные выше недостатки за счет использования «надстроек» над низким уровнем)*

Достоинство:

- позволяет программисту не задумываться об отличиях физической реализации доступа к данным, сосредотачиваясь, таким образом, на логике их обработке;
- простота организации;
- возможность простого переноса данных с одой СУБД на другую.

Недостаток: небольшое снижение скорости доступа к данным.

Технология ODBC.
Open Database Connectivity.

ODBC – это открытый интерфейс доступа к базам данных. (Microsoft). Он представляет собой API довольно низкого уровня и предназначен, в основном, для прямого использования в программах, написанных на большинстве современных языков программирования.

Стандартный ODBC интерфейс обеспечивает высокую степень универсальности приложения – один и тот же код применяется для взаимодействия с различными типами СУБД.

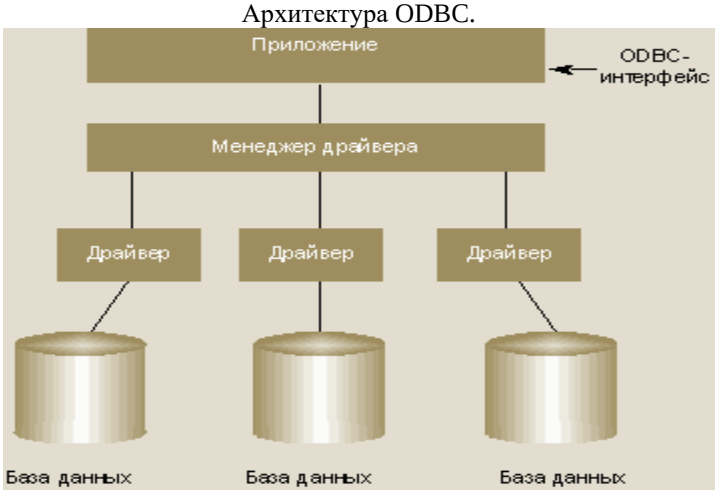
Это позволяет разработчикам создавать и продавать клиент-серверные приложения, не ориентируюсь на какую-то определенную СУБД, а значит, не тратя силы и время на учет особенностей конкретных платформ серверов БД, с которыми работает приложение. ODBC драйверы – все, что нужно такому приложению, для взаимодействия с внешним источником данных. Эти драйверы создают либо сами поставщики СУБД, либо сторонние разработчики.

Одним из преимуществ ODBC является его широкое распространение. Такие положения, как Access, Excel, Word имеют встроенный программный код, необходимый для использования ODBC. Кроме того, такие средства разработки, как VC, VBasic, Delphi имеют встроенный программный код, содержащий вызовы ODBC.

Таким образом, ODBC – механизм воздействия с реляционными базами данных. Для обеспечения доступа к БД необходимы клиентская часть СУБД, ODBC-драйвер для доступа к этой СУБД и соответствующая настройка ODBC на компьютере (имя драйвера, имя пользователя, имя базы, пароль, указание некоторых параметров драйвера). Драйвер при выполнении приложения, загружается в адресное пространство приложения и используется для доступа к БД. Для каждой СУБД используется собственный ODBC-драйвер. ODBC API стандартизирован. В визуальных средах разработки достаточно помещения на форму соответствующего компонента, указания источника данных ODBC, имени таблицы и связь с БД устанавливается или через клиентскую часть СУБД, или через соответствующий драйвер.

Технология ODBC обеспечивает общий интерфейс для доступа к разнородным базам данных стандарта SQL. ODBC использует язык SQL как стандарт для доступа к данным. Этот интерфейс очень удобен: одно приложение может обращаться к различным базам данных SQL через общий набор команд. Таким образом, разработчик может создавать и распространять приложения, не привязываясь к конкретной базе данных.

Архитектура ODBC.



1. Приложение. В роли приложения ODBC может выступать, как готовый продукт, та и пользовательское приложение.
2. Менеджер драйверов. Работа с менеджером драйверов заключается в вызове необходимых функций с определенными параметрами и в определенной последовательности. В функции менеджера входит:
   - установка и завершение связи с источником данных;
   - подготовка и выполнение SQL-операторов;
   - получение результатов и навигация по полученным наборам записей;
   - управление транзакциями;
   - идентификация ошибок;
   - получение различной вспомогательной информации.

   Для выполнения этих функций менеджер вначале должен подготовить некоторые системные ресурсы:
   - идентификатор окружения HENV. Он указывает на область памяти для общей информации (сведения обо всех соединениях с БД, какое соединения является текущим…);
   - идентификатор соединения HDBC. Этот идентификатор указывает на область памяти для информации о конкретном соединении;
   - идентификатор оператора HSTMT. Он указывает на область памяти для информации о SQL-операторе.

   В конце менеджер должен освободить эти ресурсы и вернуть их системе.
3. Драйверы. Непосредственно взаимодействуют с источником данных. Что это за драйверы и как осуществляется такое взаимодействие, имеет значение только в том случае, если Вы собираетесь

написать собственный драйвер ODBC для доступа к источнику данных. Разработчики СУБД поставляют драйвер ODBC для доступа к своему источнику данных.

Технология DAO.
Data Access Objects.

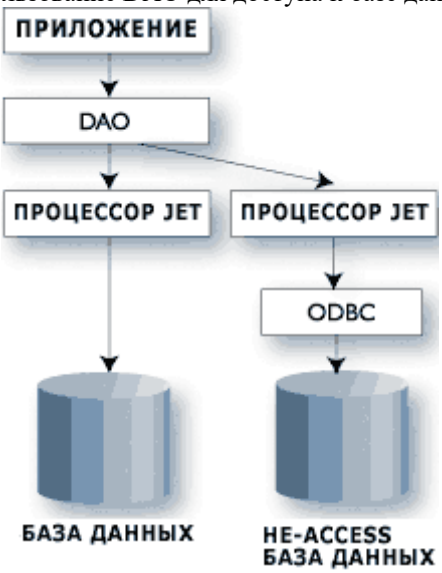
DAO – интерфейс программирования процессора базы данных Microsoft Jet, первоначально создавался для инструментальных средств разработки приложений Visual Basic и Visual Basic for Applications (VBA). DAO применяет Microsoft Jet для предоставления набора объектов доступа к данным, скрывающих стандартные объекты базы данных: таблицы, запросы и наборы записей.

Обычно DAO применяли для доступа к локальным источникам данных, хотя сама технология вполне пригодна для доступа к удаленным источникам. Характерной особенностью является наличие отдельного компонента, который доступен во всех приложениях Microsoft, поддерживающих VBA. На самом низком уровне объекты доступны через COM –интерфейсы, но чаще всего к ним применяются соответствующие классы, описанные в библиотеках классов среды разработки приложения (например, MFC в Visual Studio).

Microsoft Jet был первым объектно-ориентированным интерфейсом для связи с Access. Приложения, использующие Access, могут задействовать DAO для прямого доступа к данным. Поскольку DAO создавалась сразу же вслед за Access, применение этой технологии — самый быстрый и наиболее эффективный способ доступа к базам данных Access.

DAO может работать и с отличными от Access базами данных, такими, как SQL Server и Oracle. DAO использует ODBC, но, поскольку метод DAO спроектирован специально для взаимодействия с JET, JET транслирует запросы между DAO и ODBC. Этот дополнительный шаг трансляции и является причиной замедления работы с базами данных, отличными от Access.

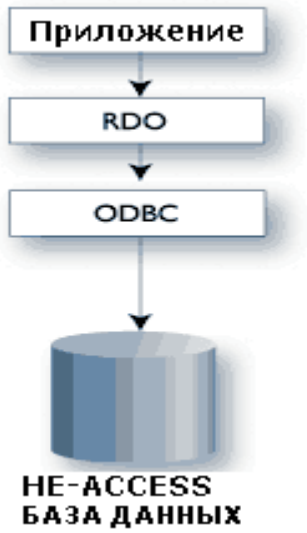
Использование DAO для доступа к базе данных.



Технология RDO.
Remote Data Objects.

RDO – является объектно-ориентированной оболочкой ODBC API, а непосредственный доступ к данным выполняет ODBC драйвер. Таким образом, RDO сочетает простой стиль программирования DAO и высокую производительность низкоуровневых функций ODBC. Модель RDO предназначена преимущественно для работы с удаленными реляционными источниками данных.

Использование RDO для доступа к базе данных.



Технологии RDO и DAO имеют один общий недостаток. Это иерархическая структура объектов модели. Такая структура приводит к тому, что программисту для обращения к результирующему набору данных приходится обращаться с начала к объекту находящемуся на самом верхнем уровне иерархии, затем к нижестоящим объектам и только потом к результирующему набору записей. В результате программист получает более сложный и объемный программный код.

Технология OLE DB.

Object Linking and Embedding Database.

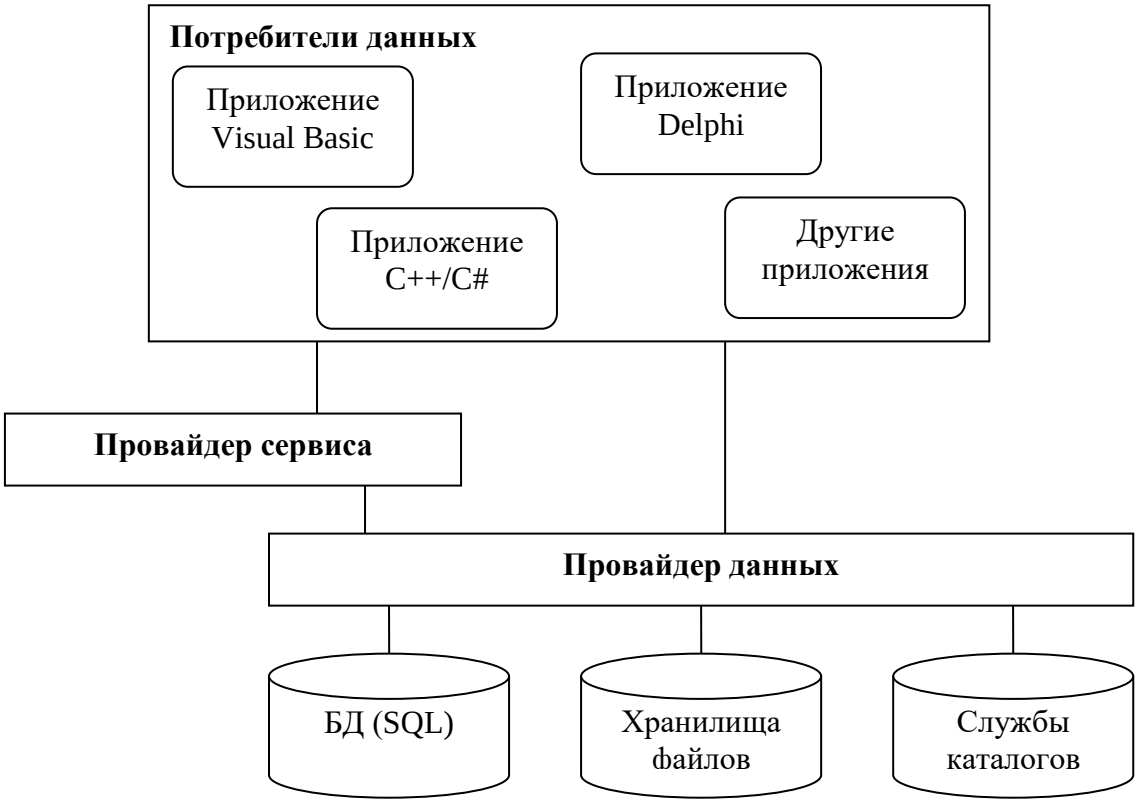
(Связь и внедрение объектов БД)

OLE DB представляет собой набор COM интерфейсов, предоставляющих приложению единообразный доступ к данным самых различных источников, независимо от их местонахождения и типа. Открытая спецификация OLE DB основана на технологии ODBC, она предоставляет открытый стандарт доступа к данным любого типа. ODBC создавалась для взаимодействия с реляционными БД, а OLE DB разрабатывалась как для реляционных, так и для нереляционных источников, включая БД на серверах, ПК, а также хранилища файлов и сообщений электронной почты, электронные таблицы, инструментальные средства управления проектами и пользовательские объекты.

Т.о. технология OLE DB построена на ODBC и расширяет ее до компонентной архитектуры, которая обеспечивает высокоуровневый интерфейс доступа к данным. Эта архитектура предоставляет постоянный доступ к SQL-данным, не SQL-данным и неструктурированным источникам данных по локальным сетям и Internet.

Компоненты OLE DB.

OLE DB состоит из 3-х компонентов.



1. Потребители данных (data consumers). Это компонент программного обеспечения, который использует интерфейсы OLE DB. Это может быть бизнес-приложение, инструментальное средство разработки ПО (Borland Delphi, Visual Studio .NET), сложные приложения.

Провайдер – это часть программного обеспечения, реализующая и предоставляющая набор базовых интерфейсов OLE DB.

2. Провайдер данных (data provider).Представляет собой компонент программного обеспечения. Он находится между потребителем и непосредственно массивом данных. В OLE DB все провайдеры предоставляют данные в табличном формате, в виде виртуальных таблиц. Провайдер данных выполняет следующие задачи:
   - принимает запросы, поступающие от потребителя, на доступ к данным;
   - выполняет выборку или обновление данных из массива данных;
   - возвращает эти данные потребителю.
3. Провайдер сервисов (service provider) или компонент сервисов (service component).Реализует расширенные функциональные возможности, которые не поддерживаются обычными провайдерами данных. То есть провайдер обеспечивает:
   - сортировку;
   - фильтрацию;
   - управление транзакциями;
   - обработку SQL-запросов;
   - функции указателя (курсора).

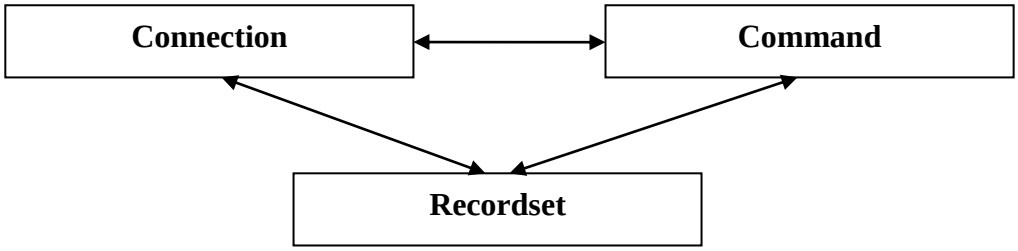
Технология ADO.
Active X Data Objects.

ADO реализует высокопроизводительный и удобный прикладной интерфейс для OLE DB. Технология ADO нетребовательна к системным ресурсам, создает минимальную нагрузку на сеть и отличается минимальным числом уровней между приложением и источником данных. ADO предоставляет COM интерфейс Automation, поэтому она поддерживается любой ведущей инструментальной средой быстрой разработки приложений, а также другими инструментальными средствами разработки БД, приложений и сценариев.

ADO объединяет все лучшее, что было в RDO и DAO. В ней применяются похожие соглашения, но с упрощенной семантикой.

Часть ADO, носящая название службы удаленных данных RDS (Remote Data Service), отвечает за передачу клиентам отсоединенных наборов записей по протоколу HTTP, что позволяет разрабатывать полнофункциональные, ориентированные на работу с данными, Web-приложения.

Объектная модель ADO.



Объектная модель ADO состоит из трех основных частей:

1. Connection (подключение). Устанавливает соединение между приложением и внешним источником данных. Кроме того, отвечает за инициализацию и создание подключения, выполнение запросов и механизм транзакций. Хранит информацию о подключении к источнику данных (имя, местоположение данных, имя пользователя и пароль, имя провайдера).
2. Command (команда). Формирует запросы на выборку записей из источников данных, учитывая заданные пользователем параметры. Объект command создается на базе таблицы БД или результатов SQL-запросов. Кроме того, можно задать отношения между несколькими объектами command для представления взаимосвязанных данных в виде иерархической структуры.
3. Recordset (набор данных). Обеспечивает доступ к записям, выбранным SQL запросом, его применяют для редактирования, добавления или удаления записи в источнике данных. Объект хранит результаты выполнения запросов, т.е. реализацию методов, позволяющих пользователю обновлять данные в БД и перемещать указатель текущей записи по результирующему набору.

Отличие объектной модели ADO от RDO и DAO состоит в том, что многие объекты независимы друг от друга. Иерархия объектов ADO допускает создание только непосредственно нужных объектов.

Технология ADO.NET.

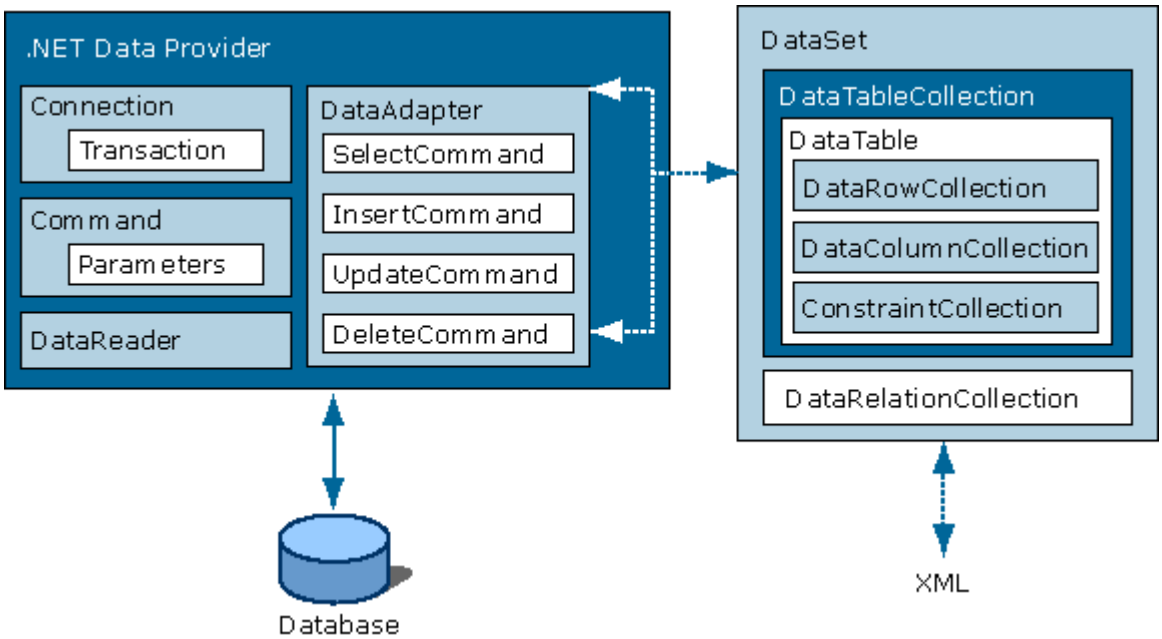
В настоящее время с появлением и активным развитием платформы Microsoft .NET дальнейшее развитие получила технология ADO, которая стала более универсальной и ориентированной на создание распределенных приложений.

Фирма Microsoft разработала технологию ADO.NET на базе уже зарекомендовавшей себя объектной технологии ADO. Но ADO.NET ориентируется на три важные возможности, которые не поддерживаются ADO:

- поддержка модели доступа к несвязанным данным, что является ключевым элементом для работы в Web;
- поддержка тесной интеграции с XML;
- интеграция с .NET Framework (например, совместимость с базовой библиотекой классов типичной системы).

XML - язык наращиваемой разметки (extensible markup language), предоставляет формат для описания структурированных данных. Это позволяет более точно объявлять содержимое и получать более значимые результаты поиска на нескольких платформах.

Архитектура ADO.NET.



Как видно из рисунка, основу ADO.NET составляют два основных: DataSet и Провайдер Данных.

Провайдер данных.

Провайдер Данных, как это ясно следует из его названия, отвечает за связь приложения с источником данных и манипуляции данными. Основные объекты, входящие в его состав, - это Connection, Command, DataAdapter и DataReader.

1. Connection используется для установления соединения с источником данных, а также для управления транзакциями.
2. Command позволяет манипулировать данными источника, а также выполнять хранимые процедуры. При этом могут использоваться параметры для передачи данных в обоих направлениях.
3. DataAdapter служит связующим звеном между DataSet и источником данных. Он использует Command для выполнения команд SQL как для заполнения DataSet данными, так и для обратной передачи измененных клиентом данных к источнику. Для выполнения этих функций объект имеет 4 метода: SelectCommand, InsertCommand, UpdateCommand и DeleteCommand. DataReader представляет однонаправленный поток данных от источника только на чтение. Если приложение клиента не модифицирует данные и не требуется произвольная выборка данных, а достаточно их однократного просмотра, то использование DataReader вместо DataSet позволит сохранить ресурсы RAM и CPU, а также поднять быстродействие приложения.

DataSet

DataSet можно представить себе как небольшую реляционную базу данных, резидентную в оперативной памяти клиента. В DataSet загружаются данные, после чего соединение с их источником (или источниками, которых в принципе может быть несколько, поскольку в одном экземпляре DataSet могут объединяться данные из нескольких источников) может быть разорвано. Далее клиент в автономном режиме производит обработку данных, при необходимости модифицирует их, после чего снова устанавливается соединение с источником и модифицированные данные передаются обратно. Такая схема работы может оказаться предпочтительной для корпоративных приложений, работающих в глобальных сетях, где поддержание постоянного соединения может повлечь дополнительные накладные расходы.

DataSet включает две основные коллекции: DataTableCollection и DataRelationCollection.

1. DataTableCollection – это коллекция объектов DataTable, каждый из которых представляет одну таблицу отношения и в свою очередь состоит из коллекций:
    - DataColumnCollection – представляет все столбцы таблицы. Помимо данных, столбец может включать формулу для динамического расчета своего содержимого.
    - DataRowCollection – представляет набор строк таблицы (возможно, пустой). Следует отметить, что хранятся не только текущие значения данных, но и первоначальные, что позволяет выделить измененные данные.
    - ConstraintCollection – набор ограничений целостности данных (например, можно задать, допустимо ли для данного столбца значение NULL или должны ли данные быть уникальными).
2. DataRelationCollection - служит для навигации по связанному набору таблиц и содержит ограничения ссылочной целостности DataSet. В ней, например, можно задать ограничения, которое не позволит удалить запись главной таблицы, если на нее есть ссылки в подчиненной таблице посредством внешнего ключа, тем самым не допуская появления «осиротевших» (orphaned) записей.

Как видно из диаграммы объектной модели ADO.NET, помимо работы с Провайдерами Данных, DataSet может работать и с данными формата XML, причем данные эти могут, как храниться в файлах, так и транспортироваться по сетям передачи данных посредством транспорта HTTP.

Модуль №2.

Технологии удаленного доступа к системам баз данных, тиражирование и синхронизация в распределенных системах баз данных.

Лекций - 2 ч.

Темы лекционных занятий:

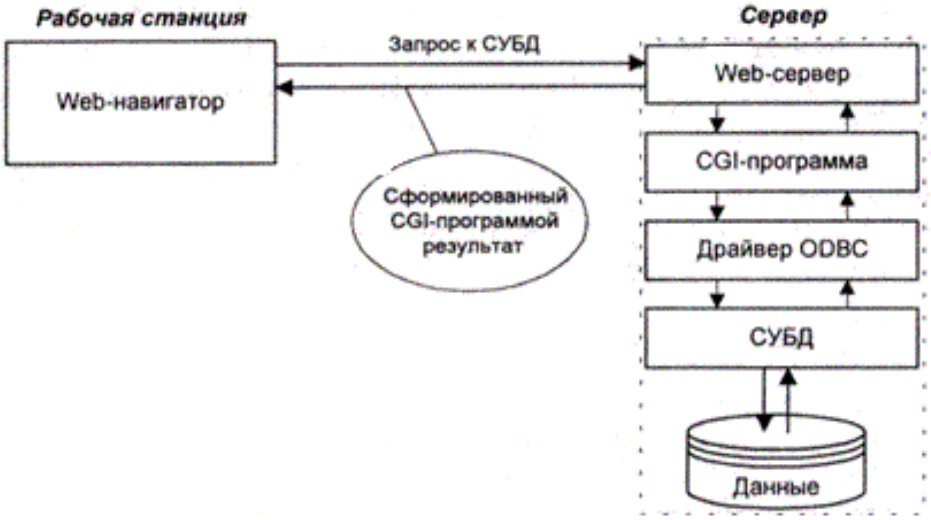
- Технологии удаленного доступа к системам баз данных, тиражирование и синхронизация в распределенных системах баз данных.
- Реляционные базы данных и введение в язык SQL.

Технологии удаленного доступа к системам баз данных, тиражирование и синхронизация в распределенных системах баз данных

В условиях удаленного доступа к информации хорошим способом являются средства Internet, рассмотрим доступ к серверу СУБД через Web-сервер. Для доступа Web-навигатора к серверу СУБД через Web-сервер используется система программных шлюзов.

Программный шлюз, получив запрос от Web-сервера, выступает в качестве посредника между сервером Web и сервером СУБД. Программные шлюзы разрабатываются в соответствии с определенными стандартами, определяющими способы вызова Web-сервером прикладных программ или функций динамических библиотек, а также способы обмена информацией с этими программными объектами. Одними из наиболее распространенных стандартов данного типа являются интерфейс CGI (Common Gateway Interface — общий интерфейс шлюзов), а также его усовершенствованная спецификация, названная как FastCGI (ускоренный CGI).

Схема доступа к СУБД через CGI-программу



Для доступа Web-навигатора к серверу СУБД через Web-сервер по стандарту CGI необходима соответствующая CGI-программа, выполняющая роль программного шлюза между Web-сервером и сервером СУБД.

CGI-приложения работают независимо от Web-сервера, а их запуск осуществляется по вызову с Web-документа при его обработке Web-навигатором. CGI-программа взаимодействует с Web-сервером посредством двустороннего обмена переменными среды через стандартные каналы ввода/вывода данного приложения.

Поскольку CGI-программы работают независимо от Web-сервера и имеют простой общий интерфейс, разработчики Web-документов имеют возможность создавать свои CGI-программы на любом языке, поддерживающем стандартные файловые операции ввода/вывода. Кроме того, при независимой разработке можно создавать такие приложения, которые легко переносятся с одного на другой Web-сервер. Существуют и стандартные CGI-программы, специально разработанные для взаимодействия Web-серверов с различными СУБД, например, программа WebDBC.

В качестве интерфейса между Web-навигатором и сервером СУБД в составе Web-документов применяют HTML-формы, которые позволяют формулировать запросы к базе данных. CGI-программа получает информацию от Web-сервера либо через переменные окружения, либо через стандартный ввод. Все зависит от метода доступа, который используется при обмене данными между Web-навигатором и Web-сервером. Далее CGI-программа через драйвер ODBC (Open DataBase Connectivity) обращается к серверу СУБД и возвращает Web-серверу ответ на запрос через стандартный вывод.

Драйвер ODBC обеспечивает унифицированный способ доступа к различным СУБД посредством стандартного языка запросов SQL. Благодаря стандарту ODBC прикладные программы могут использовать единственный диалект SQL и взаимодействовать с разными СУБД. Можно обойтись и без драйвера ODBC, но в этом случае CGI-программа должна быть написана с ориентацией на конкретную СУБД, функционирующую на сервере.

Таким образом, разработчику CGI-приложения не надо ничего знать о том, как устроен Web-сервер. Более того, ему вовсе не обязательно использовать сложные языки типа C++. CGI-программа может быть написана и на командном языке, например Perl. Главное выдержать все соглашения, накладываемые стандартом CGI. Такой подход существенно облегчает разработку прикладного программного обеспечения для Web вообще и для сопряжения баз данных с Web-сервером в частности.

НО! Стандарт CGI обладает и недостатком — снижение скорости обработки запросов при увеличении интенсивности их поступления. При каждом вызове CGI-программы ее приходится загружать с диска (т. е. запускать новый процесс). По завершении работы программы требуется освободить использовавшиеся ею ресурсы. Такие операции создают заметную дополнительную нагрузку на сервер, что сказывается на его производительности. К тому же запуск нового процесса при каждом запросе снижает

эффективность постоянных процессов и доступность данных. Информацию, которая сгенерирована в ходе обработки одного запроса, невозможно использовать при обработке другого.

Для того чтобы обойти проблемы, связанные с быстродействием CGI, многие поставщики Web-серверов, включая Microsoft и Netscape, разработали соответствующие интерфейсы прикладного программирования (API). Корпорацией Microsoft был разработан интерфейс ISAPI (Internet Server API), а корпорацией Netscape — интерфейс NSAPI (Netscape Server API).

Эти интерфейсы тесно интегрированы с Web-сервером, позволяя сохранять доступность постоянно используемых процессов и данных. Программы с интерфейсом ISAPI компилируются в файлы динамически подключаемых библиотек DLL. Они загружаются в память во время первого обращения к ним и поэтому для повторного вызова этих программ не нужно порождать новый процесс. Функции интерфейса NSAPI загружаются в серверное пространство процессов. Соответственно при вызове этих функций также не порождаются дополнительные процессы. Благодаря API-интерфейсу использующая его программа может оставлять соединение с СУБД открытым, так что следующему запросу к базе данных не придется тратить время на открытие и закрытие соединения.

Однако API-интерфейсы Web-серверов — хоть и неплохое, но нестандартное решение. Большинство приложений нельзя переносить с одного API на другой, и очень редко удается переносить приложения на другие платформы. Кроме того, большинство приложений для Web-серверов все еще создаются для интерфейса CGI, поэтому переход к приложениям на базе API не представляется экономически оправданным.

Поэтому стали появляться способы построения некоторого промежуточного варианта, который, с одной стороны, удовлетворял бы требованиям мобильности, независимости и простоты программирования, а с другой стороны - был бы достаточно эффективным. Одним из таких решений является спецификация FastCGI. Идея этой спецификации в том, что прикладная программа использует способ передачи параметров и данных, который применяется в CGI, но при этом не удаляется из памяти, а остается резидентной, обрабатывая поступающие запросы.

Таким образом, приложения на базе FastCGI, подобно CGI-программам, работают независимо от Web-сервера и запускаются через стандартные ссылки в Web-документах. Но, как и программы на базе API, программы для FastCGI являются постоянно действующими. Когда программа заканчивает обработку очередного запроса, ее процесс остается открытым в ожидании нового запроса.

При доступе Web-навигатора к реляционной базе данных через интерфейс FastCGI получается схема, в которой фактически используются три сервера: Web-сервер, FastCGI-программа и сервер базы данных. Web-сервер принимает запрос Web-навигатора и передает его FastCGI-программе, которая в свою очередь обращается к серверу баз данных. Результат возвращается по обратной цепочке.

Модуль №3.

Архитектура Microsoft SQL Server

Лекций - 8 ч.

Темы лекционных занятий:

- Логическая архитектура Microsoft SQL Server. Активное администрирование и объектный интерфейс SQL Server.
- Таблицы. Представления. Хранимые процедуры. Правила. Умолчания.
- Системные базы данных Microsoft SQL Server.
- Производительность сервера.
- Репликация. Служба преобразования данных.
- Принципы безопасности. Роли сервера и роли базы данных.
- Резервное копирование.
- Транзакции.

Архитектура баз данных SQL Server.
Файлы.

Для каждой БД существует минимум два файла: для хранения данных и для хранения транзакций. Количество файлов может быть довольно большим. Они могут объединяться в группы, что упрощает процесс администрирования. В состав БД могут входить следующие файлы:

1) Основной файл - содержит системную информацию о БД, ее объектах и размещении всех файлов. Также если нет вторичных файлов, основной файл может содержать и все данные, хранящиеся в БД. По умолчанию основной файл имеет расширение mdf.
2) Вторичный файл - используется только для хранения данных, и системной информации не содержит. БД может иметь несколько вторичных файлов или не иметь их вовсе. По умолчанию вторичный файл имеет расширения ndf.
3) Файл журнала транзакций - содержит информацию о транзакциях данной БД. Любая БД имеет хотя бы один файл для журнала транзакций, по молчанию расширение - ldf.

Возможность хранения БД в нескольких файлах, а также сохранение журнала транзакций в отдельном файле может оптимизировать работу сервера. Кроме того, хранение журнала транзакций и основного файла на разных дисках повышает отказоустойчивость систем, т.к. вероятность поломки обоих дисков одновременно мала.

Объекты базы данных.

1) Таблицы. (tables). Типы полей:
   a) Символьные (char, varchar, nchar, nvarchar)
   b) Двоичные (binary, varbinary)
   c) Целочисленные (tinyint, smallint, int, bigint)
   d) Приближенные числовые (float, real)
   e) Точные числовые (decimal, numeric)
   f) Финансовые (money)
   g) Битовые (bit)
   h) Дата и время (datetime, smalldatetime)
2) Представления (views). Это виртуальные таблицы с данными из одной или нескольких реально существующих таблиц. С помощью представлений можно облегчить доступ пользователя к таблицам, показывая только поля необходимые ему для работы. Представления могут работать не только на чтение, но и на запись.
3) Умолчания (defaults). Нам, известно, что одним из свойств поля таблицы является значение по умолчанию. Но умолчание, может существовать как самостоятельный объект БД. Каждое умолчание имеет свое имя и значение, которое может быть представлено как константой, так и выражаться некоторой формулой. Созданные пользователем умолчания, отображаются в дизайнере таблицы.
4) Правила (rules). Используются для ограничения значений, хранимых в столбцах или в пользовательских типах данных.
5) Типы данных определяемые пользователем. (user defined data type). Пользовательский тип данных создается на основе базовых типов дынных. Также сразу можно определить правило и умолчание для создаваемого типа данных. Кроме того, если пользовательский тип данных является производным от строкового типа, то можно определить его длину.
6) Диаграммы (diagrams). Диаграмма - это визуальное представление таблиц и связей между ними. Диаграммы позволяют создавать новые таблицы, модифицировать структуру уже существующих таблиц и изменять их свойства, устанавливать новые связи и редактировать их характеристики.
7) Хранимые процедуры (stored procedures). Хранимая процедура - это именованный объект базы данных, состоящий из последовательности команд Transact-SQL. Необходимость их существования на сервере SQL диктуется тем, что не всякий результат можно получить одним запросом.. Последовательность нескольких запросов хорошо бы хранить именно на сервере, а не воспроизводить ее каждый раз на клиентской стороне. Иногда возникает необходимость реализации логики, плохо укладывающейся в рамки языка запросов; для этой цели лучше могут подойти структуры обычного алгоритмического языка программирования. Именно такими возможностями и дополняется язык Transact-SQL . Хранимые процедуры могут содержать огромное количество команд, а вызываться из клиентской программы всего одной строкой. Это, естественно, снижает сетевой трафик. Использование хранимых процедур позволяет построить приложение, при котором на клиентской стороне будет реализован только интерфейс пользователя.
8) Расширенные хранимые процедуры (extended stored procedures). По своей структуре, расширенные хранимые процедуры представляют собой динамические библиотеки, написанные на каком-либо языке программирования (обычно на C++). Особенность расширенных процедур заключается в том, что они являются глобальными, т.е. не относятся к какой-либо базе данных. Регистрируется она в системной базе данных master.
9) Функции пользователя (user defined functions). Пользовательские функции появились только в 2000 версии SQL Server. Они очень похожи на хранимые процедуры, но для них имеются дополнительные ограничения. Функции не могут выполнять обновление таблиц, объявлять глобальные курсоры, создавать объекты базы данных, открывать и закрывать транзакции, использовать некоторые системные функции. При вызове функции необходимо обязательное указание владельца, чего не требуется для других объектов базы данных.
10) Триггеры (triggers). Триггер представляет собой хранимую процедуру особого вида, которая выполняется при операциях трех видов: вставке, обновлении и удалении записи. Если триггер обращается к таблице, которая защищена другим триггером, то возникает ситуация вложенности. Всего по умолчанию возможно до 32 вложений. Если уровень вложенности превосходит допустимый, то возникает ошибка. Возможен и рекурсивный вызов триггера при его обращении к той же таблице. По умолчанию рекурсивный вызов триггера отключен. Но так же, как и вложенными, так и рекурсивными триггерами лучше не злоупотреблять.

Системные базы данных и таблицы.

SQL Server 2000хранит в системных базах данных и таблицах практически всю информацию о настройках, защите и расположению объектов. Он поддерживает четыре системные базы данных, которые необходимы для правильного функционирования сервера.

1) master. Это главная база данных. В ней хранится информация системного уровня для данного сервера. В частности master содержит информацию о других пользовательских базах данных, учетных записях, параметрах настройки системы и многое другое. Информация в главной базе данных постоянно изменяется; например: при создании какого-либо пользовательского объекта или при выполнении некоторых системных процедур. В базе данных master сосредоточено большое количество системных хранимых процедур, функций и расширенных хранимых процедур.
2) msdb. Основное назначение этой базы данных - это сохранение информации, касающейся планирования работы заданий, операторов, предупреждений…, которая используется службой SQL Server Agent. Здесь же хранится резервная копия информации, необходимая при восстановлении БД.
3) model. Это база данных, где хранятся шаблоны. При создании новой базы данных содержимое БД model копируется в новую БД. Можно поместить некоторые необходимые пользователю объекты в model и тогда они будут автоматически формироваться при создании пользовательской БД.
4) tempdb. Данная БД используется для хранения временных объектов: временных таблиц, временных хранимых процедур, переменных типа курсор и т.п. Временные таблицы могут создаваться как пользователем, так и самим сервером для своих внутренних служб.

Средства управления MS SQL Server и доступа к данным.
Управление базой данных.

Для формирования новой базы данных можно воспользоваться любым из трех способов:

1) Создавать новые объекты с помощью программы Enterprise Manager.
2) Прибегнуть к помощи Мастера создания баз данных. SQL Server имеет большой набор мастеров, упрощающих выполнение тех или иных операций. Для вывода окна со списком мастеров (Wizards) необходимо на все той же консоли выбрать пункт Tools/Wizards. Все мастера разбиты на 4 группы. В группе DataBase имеется мастер Create Database Wizard.
3) Использовать средства Transact_SQL.
   Кроме описанных выше возможностей управления базами данных существует еще две: использование системных команд DBCC и применение огромного количества системных хранимых процедур.

Производительность сервера.
Программа SQL Server Profiler.

Достижение высокой производительности приложений, работающих на основе SQL Server, зависит от многих факторов: структуры базы данных, ее расположения, выбора и оптимизации индексов, формулировки запросов, оптимизации работы компьютера, сетевых настроек и т.д.

Для средних и больших баз данных, когда количество хранимых записей достигает десятков, сотен тысяч и даже миллионов, фактор оптимизации выдвигается на первое место и часто заставляет проектировщиков отступать от канонических принципов проектирования РБД.

Программа SQL Server Profiler предназначена для регистрации событий, происходящих в SQL Server. Особенно полезно использовать эту программу при анализе работы сложной хранимой процедуры, содержащей большое количество запросов и конструкций языка SQL. С ее помощью можно легко определить участки кода с наибольшей длительностью выполнения.

При интенсивной многопользовательской работе Profiler позволяет проанализировать распределение блокировок в системе, интенсивность обращения к большим таблицам… Программа Profiler позволяет выводить полученные результаты в окно, файл или таблицу.

Применения связанных серверов.

В большой сети могут существовать разные источники данных. Особенно это характерно для сетей эксплуатируемых достаточно длинный промежуток времени, в течение которого приобретались и разрабатывались различные системы, основанные на разных технологиях. В этих системах, возможно, сформированы большие объемы данных, которые по-прежнему востребованы. После установки SQL Server перевод данных на новую платформу может потребовать много времени, а, следовательно, и средств. Однако в таком глобальном преобразовании нет необходимости, т.к. SQL Server обладает рядом механизмов, позволяющих использовать самые различные источники данных.

Одним из таких механизмов является - механизм связанных серверов - linked servers.

Обращение к различным источникам данных осуществляется посредством технологии OLE DB, которая позволяет организовать доступ, как к реляционным, так и к не реляционным источникам данных.

Преобразование команд SQL Server в формат интересующего источника данных осуществляется соответствующим провайдером, т.е. набором процедур OLE DB. В состав SQL Server входит набор провайдеров, обеспечивающих доступ к большинству современных источников данных (oracle, paradox…).

Для создания связанного сервера можно воспользоваться программой Enterprise Manager или сделать это программным путем, используя системные хранимые процедуры.

Вся информация о связанных серверах формируется при их создании и заносится в таблицу sysservers системной базы данных master.

Если в SQL-запросе указывается имя, отличное от имени текущего сервера, то SQL Server обращается к данной таблице, и если нужная строка найдена, то посредством провайдера OLE DB осуществляется соединение с указанным источником данных и выполняется соответствующее действие.

Репликация.

Репликация (дублирование) - это одно из средств перемещения информации между серверами в реальном времени. В большой организации может быть несколько серверов MS SQL. Часто базе данных расположенной на одном сервере, требуется информация, размещенная на другом сервере.

Репликация представляет собой копирование данных с одного сервера на один или несколько других серверов.

Основные термины репликации:

1) Издатель (publisher) - сервер SQL, предоставляющий свои данные для копирования на другие серверы.
2) Подписчик (subscriber) - сервер, копирующий данные, представляемые издателем.
3) Дистрибьютор (distributor) - сервер, выступающий в роли посредника между создателем и подписчиком.
4) Статья (article) - таблица или ее часть, выбранная для копирования.
5) Публикация (publication) - набор статей, копируемых с сервера издателя, как одно целое.

Модели репликации:

1) Репликация мгновенных снимков (snapshot replication). Моментальные снимки статей издателя передаются подписчикам. Если размер статей велик, то такой вид репликации может сильно загрузить сеть. Положительным моментом репликации данного вида является малая загрузка процессора, т.к. в этом случае не требуется проводить анализа и отбора передаваемых данных. Такая модель репликации очень удобна для редко обновляемых баз данных.
2) Репликация транзакций (transaction replication). Может быть использована для копирования таблиц и хранимых процедур. В этой модели происходит непрерывный отбор транзакций и копирование их в базу данных дистрибьютора. Далее эти транзакции рассылаются подписчикам и там же выполняются. При наличии быстрого надежного канала связи все изменения, происходящие с объектами репликации на издателе, отражаются у подписчиков практически в реальном времени.
3) Репликация сведением. (merge replication). Это наиболее сложный тип репликации, предполагающий двухстороннюю связь между издателем и подписчиком. Изменения, вносимые в статьи на стороне подписчика, переносятся на соответствующие данные издателя, а затем оказываются у других подписчиков.

Методы обновления подписчиков:

1) Репликация по запросу (pull subscription). Подписчик периодически подключается к дистрибьютору и требует у него все изменения, произошедшие со времени последнего подключения.
2) Принудительная репликация (push subscription). Дистрибьютор сам устанавливает соединение с подписчиком и передает ему все требуемые данные.

Для работы с репликацией удобнее всего воспользоваться одним из ее мастеров. Для этого на консоли программы Enterprise Manager следует выбрать пункт меню Tools/Wizards. В разделе Replication перечислены все мастера для работы с репликацией.

Служба преобразования данных.
DTS.

Служба DTS предназначена для автоматизации процессов импорта, экспорта и преобразования данных. Она работает с самыми разными СУБД и их типами данных и основана на механизме OLE DB.

Служба DTS разрешает сформировать график ее запуска. Это позволяет полностью положится на ее автоматизированное функционирование. В каждом конкретном задании определяются все соединения и процессы. Все это в целом называется пакетом, который может быть сохранен в одной из следующих форм:

- в системной базе данных msdb (sysdtspackages);
- в COM-структурированном файле;
- в виде программы на VB.

В состав каждого пакета могут входить одно или несколько заданий и одно соединение. Любое задание состоит из шагов. Шаг может включать в себя одну или несколько задач и набор условий. Шаги призваны определять последовательность выполнения заданий. Во многих случаях эта последовательность очень важна. Например, сначала необходимо создать таблицу, а только затем происходит перекачивание в нее информации.

Для создания пакета можно воспользоваться мастером экспорта/импорта. На консоли программы Enterprise Manager следует выбрать пункт меню Tools/Wizards команду Data Transformation Services.

Система безопасности в MS SQL Server.

Принципы безопасности.

MS SQL Server 2000 - прежде всего сетевая система. Использование SQL Server в качестве настольной системы доступа к базам данных имеет смысл разве только для отладочных целей. Следовательно, ее безопасность должна, как-то соотносится, с сетевой системой безопасности, т.к. все пользователи SQL Server являются вместе с тем и сетевыми пользователями.

Обычно разграничение доступа к сетевым ресурсам строится на основе принадлежности сотрудника к тому или иному подразделению. Поскольку данные, которыми пользуются сотрудники, чаще всего, хранятся в виде таблиц SQL Server, то явно выстраивается линия безопасности Windows 2000 Server и MS SQL Server 2000.

Поэтому разработчики SQL Server тесно интегрируют свой продукт с операционной системой и системой безопасности Active Directory.

Система безопасности Active Directory позволяет гибко управлять доступом пользователей к ресурсам сети. Объединяя системы безопасности SQL Sever и Windows 2000, можно значительно упростить процесс администрирования. Т.е. если система входа в сеть отлажена хорошо, то администратор SQL Server может положиться на нее. В таком случае невозможна ситуация, когда разрешение на работу с базами данных получит человек, которому отказано в доступе к локальной сети.

Процесс допуска пользователя к ресурсам SQL Server распадается на две составляющие: аутентификацию и авторизацию.

Аутентификация - это подтверждение личности пользователя, что обусловлено именем и паролем. Существует два вида аутентификации: SQL Server and Windows и Windows only.

Первый вид аутентификации (смешанный), предполагает, что после того, как пользователь прошел аутентификацию в сети Windows, он затем обязан пройти ее и на SQL Server. Это означает, что SQL Server должен иметь свою базу учетных записей, элементы которой могут и не совпадать с учетными записями базы Windows.

Во втором случае на SQL Server хранятся идентификаторы допущенных к работе с сервером учетных записей Windows, права которых контролируются при попытке входа на сервер.

Авторизация - это процесс проверки прав доступа или запретов для данного пользователя. Процесс авторизации предполагает связывание учетной записи с определенным пользователем конкретной базы данных.

При установке SQL Server автоматически создаются две учетные записи:

- BULITIN/Администраторы - учетная запись Windows 2000, которая обеспечивает автоматический доступ к SQL Server всем членам группы Администраторы. Т.о. члены группы Администраторы могут выполнять административные функции и в SQL Server. Впрочем, эту учетную запись можно удалить, т.к. в конкретных условиях администратор сети и администратор базы данных могут оказаться разными людьми.
- sa - специальная учетная запись для администратора SQL Server, играющая запасную роль. Для нее обязательно следует установить пароль, чтобы обезопасить базу данных от доступа к ней лиц, не являющихся администраторами SQL Server.

Роли.

Понятие «роль» и «группа пользователей» очень близки по смыслу. Роли позволяют упростить администрирования сервера путем объединения нескольких пользователей, выполняющих одинаковые виды операций.

Роли в SQL Server бывают четырех видов:

- фиксированные роли сервера;
- фиксированные роли базы данных;
- роли приложений;
- пользовательские роли на уровне БД.

Шифрование данных.

Для сокрытия конфиденциальных сведений в SQL Server используется также шифрование данных . Эта операция применяется :

1. при обмене информацией между сервером и клиентом; при этом для передачи данных необходимо использовать multiprotocol;
2. для скрытия паролей учетных записей или ролей приложения; все пароли хранятся в системной базе данных master в зашифрованном виде;
3. для защиты любого кода, используемого в хранимых процедурах, пользовательских функциях, триггерах, представлениях.

Резервное копирование.

Существует четыре типа резервного копирования:

1. Полное резервное копирование базы данных (database backup). В этом случае все содержимое базы данных будет помещено в один или несколько файлов. Полная резервная копия будет содержать всю информацию на момент окончания копирования. Другими словами, если в процессе резервного копирования в базе происходили изменения, то они отразятся в окончательном варианте архива.
2. Дифференциальное резервное копирование (differential database backup). При таком методе сервером отслеживаются только изменения, произошедшие со времени полного резервного копирования. Такая копия компактна и требует меньше времени для своего создания.
3. Резервное копирование файлов и групп файлов (file and filegroup backup). При таком резервном копировании архивируются только отдельные файлы или группы файлов. Другими словами таблицу или даже отдельные ее столбцы можно поместить в отдельный файл. И если изменениям подвергается только данная таблица, то есть смысл осуществлять регулярное резервное копирование именно этого файла, имея, конечно же, полную резервную копию всей базы, сделанную некоторое время назад.
4. Резервное копирование журнала транзакций (transaction log backup). При данном типе резервного копирования создается копия информации о транзакциях, зафиксированных в соответствующем журнале. При таком подходе сервер сканирует журнал транзакций и помещает в архив информацию только о тех транзакциях, которые произошли с момента последнего резервного копирования.

Транзакции.

Транзакция - это группа инструкций SQL, исполняемых, как единое целое.

Свойства транзакций:

1. Атомарность. Т.е. транзакция представляет собой неделимую единицу работы, которая может быть исполнена полностью или не выполнена вовсе.
2. Согласованность или непротиворечивость. Каждая транзакция должна переводить систему из одного согласованного состояния в другое. Т.е. по окончании транзакции все структуры будут находиться в корректном состоянии.
3. Изолированность. Все транзакции должны выполнятся независимо друг от друга. Т.е. любые промежуточные результаты незавершенной транзакции не должны быть доступны из другой транзакции.
4. Долговечность. Результат успешно завершенной транзакции должен быть записан в базе данных. Если сразу после окончания транзакций в системе произойдет сбой, то при следующем запуске сервера он по журналу транзакций приведет данные в стандартное состояние.

С точки зрения правил выполнения, транзакции можно разделить на следующие типы:

1. Явные транзакции. Для них в программном коде требуется явное указание начала и конца транзакции.
2. Автоматические транзакции. Это когда каждая команда Transact-SQL рассматривается как самостоятельная транзакция.
3. Неявные транзакции. При установке в соединении такого режима сервер автоматически заканчивает текущую транзакцию и начинает новую.
4. Распределенные транзакции. Они используются при обращении к другим базам данных.
5. Вложенные транзакции. Внутри явных транзакций можно инициировать новые явные транзакции. Все вложенные транзакции будут завершены только после окончания самой верхней транзакции.

Для управления транзакциями в SQL Server 2000 используется специальный журнал. В него сервер заносит сведения о каждой транзакции. Все изменения, произошедшие от начала транзакций, сначала кэшируются. Если во время выполнения транзакций произошла ошибка, то сервер производит откат транзакции. Только после получения команды, подтверждающей успешное выполнение последнего оператора транзакции, данные КЭШа, записываются на диск, а в журнал транзакций заносится информация об ее фиксации.

При полной потере базы данных достаточно ее резервной копии и журнала транзакций, чтобы восстановить базу данных в исходное состояние.

Литература, рекомендуемая для самостоятельного изучения.

1. Александр Волоха. Microsoft SQL Server 2005. Новые возможности. Спб., БХВ-Петербург, 2005.
2. М. Хотек. Microsoft SQL Server 2008. Реализация и обслуживание. Издательство: Русская Редакция, 2011 г.
3. Майк Гандерлой, Джозеф Джорден, Дейвид Чанц. Освоение Microsoft SQL Server 2005. Издательство: Вильямс, 2007 г.
4. Роберт Виейра. Программирование баз данных Microsoft SQL Server 2008. Базовый курс. Издательства: Диалектика, Вильямс, 2010 г.
5. Пол Нильсен. SQL Server 2005. Библия пользователя. Издательство: Вильямс, 2008 г.

Раздел 5.
Методические указания для лабораторных работ.

Модуль № 1

Лабораторных занятий - 10 ч.

Темы практических занятий:

- Создание базы данных. Создание таблиц.
- Создание связей (первичные и внешние ключи).
- Создание представлений (запросов). Запросы на выборку.
- Запросы в Enterprise Manager. Запросы в Query Analyzer.
- Запросы с группировкой. Агрегированные функции.

1. Создание базы данных

Пример 1.1

```
CREATE DATABASE Student    имя базы данных
ON
( NAME = Student_dat,   имя файла данных
  FILENAME = 'c:\program files\microsoft sql server\mssql\data\Studentdat.mdf', место на
диске
  SIZE = 5,
  MAXSIZE = 20,
  FILEGROWTH = 5 )
LOG ON
( NAME = 'Student_log',
  FILENAME = 'c:\program files\microsoft sql server\mssql\data\Studentlog.ldf',
  SIZE = 5MB,
  MAXSIZE = 15MB,
  FILEGROWTH = 5MB )
GO
```

2. Создание таблиц

Пример 2.1

Создание простой таблицы

```
CREATE TABLE groups
(
  gr_id  smallint
    IDENTITY(1,1)
    PRIMARY KEY,
  name_group      char(50)    NOT NULL
    DEFAULT 'KIS',
  head char(20)  NULL
)
```

Пример 2.2

Пример создания таблицы с ограничениями

```
CREATE TABLE jobs
(
  job_id  smallint
    IDENTITY(1,1)
    PRIMARY KEY ,
```

```
  job_desc       varchar(50)     NOT NULL
    DEFAULT 'New Position - title not formalized yet',
  min_lvl tinyint NOT NULL
    CHECK (min_lvl >= 10),
  max_lvl tinyint NOT NULL
    CHECK (max_lvl <= 250)
)
```

Пример 2.3
Таблица employee содержит поле job_id, которое является внешним ключом.

```
CREATE TABLE employee
(
  emp_id  smallint
    IDENTITY(1,1)
    PRIMARY KEY ,
  emp_name       varchar(50)     NOT NULL,

  job_id smallint NOT NULL REFERENCES jobs (job_id)
)
```

Таблицы jobs и employee связаны по полям job_id отношением один ко многим

Пример 2.4
Пример использования в таблице вычисляемого поля

```
CREATE TABLE TestTable
  (ColA INT PRIMARY KEY,
   ColB INT NOT NULL,
   ColC AS (ColA + ColB) * 2)
```

3 Ввод данных в таблицу

Добавление строк в таблицу (представление) выполняет следующий оператор

INSERT [INTO] *имя таблицы (представления)* [(список полей)] VALUES *(значения)*

Пример 3.1

```
INSERT INTO MyTable (PriKey, Description)
       VALUES (123, 'A description of part 123.')
GO
```

В первых скобках через запятую перечисляются имена всех полей, во вторых скобках конкретные значения этих полей. В случае если в таблице имеется поле с автоматически изменяющимся значением, имя этого поля и его значение не указываются

Пример 3.2

Оператор INSERT INTO позволяет добавлять в таблицу только по одной строке. Пример добавления нескольких строк в таблицу приведен ниже

```
INSERT INTO MyTable  (PriKey, Description)
       SELECT ForeignKey, Description
       FROM SomeView
```

```
В этом примере оператором SELECT выбираются строки из таблицы
SomeView и добавляются в таблицу  MyTable. Значения берутся
только из двух полей (типы этих полей должны совпадать с
типами полей в таблице MyTable).
```

4 Изменение таблицы

Таблицу можно модифицировать после ее создания, с помощью команды ALTER TABLE
С помощью этой команды можно добавлять. Удалять столбцы, изменять их тип данных, добавлять и удалять ограничения.
Синтаксис команды ALTER TABLE

ALTER TABLE имя_таблицы [COLUMN имя_столбца] [тип данных |NULL NOT NULL]
[DROP] [CONSTRAINT ограничение]
[ADD] [COLUMN] имя_столбца

Пример 4.1
Добавления в таблицу employee поля с ограничением «уникальный»
ALTER TABLE employee ADD emp_card int NULL
CONSTRAINT emp_card UNIQUE

Пример 4.2
Удаление поля из таблицы
ALTER TABLE employee DROP COLUMN emp_card

Пример 4.3
Изменение типа данного поля таблицы
ALTER TABLE employee ALTER COLUMN emp_name char(30) NOT NULL

Для удаления объектов базы данных применяется команда

drop объект имя_объекта

Например:

drop table jobs – удаление таблицы jobs
drop database my – удаление базы данных my

Запросы в Enterprise Manager.

Для задания критерия отбора в поле типа char используется выражение Like со следующими ключами:

% - любой набор символов и цифр
_ - одиночный символ
[] - любой одиночный символ в указанном диапазоне
^ - любой одиночный символ, не входящий в указанный диапазон

Примеры:

Условие	Значение
LIKE '5[%]'	5%

LIKE '[_]n'	_n
LIKE '[a-cdf]'	a, b, c, d, or f
LIKE '[-acdf]'	-, a, c, d, or f
LIKE '[[ ]'	[
LIKE ']'	]
LIKE 'abc[_]d%'	abc_d and abc_de
LIKE 'abc[def]'	abcd, abce, and abcf
LIKE '5[^%]'	Число 5 с любым символом кроме %
LIKE ‘Ab%’	Любые слова, первые 2 буквы в которых Ab
LIKE ‘London’	Значение в поле должно совпадать с условием London
LIKE ‘%o%’	Выйдут записи в которых есть хотя бы одна буква o

Для числовых полей и полей с датами используются операции <, <=, >, >=, =, <>, а для более сложных условий логические операторы OR и AND.

Примеры:
=100
>=100 AND <=1000
>=100 OR <= 1000
<> 100

Запросы в Query Analyzer.

Все команды выполняются в контексте одной базы данных, которая называется текущей. Для того чтобы сделать базу данных текущей используется команда USE
Пример
use pubs

Инструкция select.
Простейшая структура раздела Select имеет следующий вид:

Select [all | distinct] [top n [percent]] список полей
FROM - указывает источник столбцов
WHERE – условия отбора

All – ключевое слово указывает, что в результате выборки должны быть включены все строки возвращаемые запросом. Результат выборки может содержать повторяющиеся строки. Параметр All используется по умолчанию.
Distinct – исключает повторяющиеся строки из запроса.
Top n [percent] - Ограничивает количество строк выборки. Если указан только параметр n, то будет возвращено n строк.
Если после n указан параметр percent, то будет возвращено n процентов от существующего количества строк в таблице.

Список полей – в этом списке указываются имена полей таблицы или представления, если необходимо включить все поля достаточно указать параметр *. Также в списке может быть указано выражение, на основе которого будет формироваться содержимое столбца, при этом столбец который формируется по выражению не имеет никакого имени.

Для указания псевдонима имени любого столбца, в том числе, и выражения, используется параметр as.

Пример 1
```
select top  4  students.surname
from students
```

будет возвращено 4 строки поля surname из таблицы students

Пример 2
```
select top  50 percent  students.surname
from students
```

будет возвращено 50 процентов строк, т.е. если в таблице students 10 строк, будет возвращено 5

Пример 3
```
select *
from students
```

Этот простейший запрос выбирает абсолютно все поля (символ *) из таблицы students

Пример 4
```
select (summa*10/100) as [отчисления]
from payment
```

В этом запросе будет вычислено 10 процентов от поля summa и результат выведен в поле и с именем отчисления. Этот пример показывает использование вычисляемого поля в запросе

Агрегированные функции.

SQL Server поддерживает агрегированные функции аналогичные групповым (которые использовались в представлениях), но в отличие от групповых функций они применяются ко всему набору строк. Т.е. используются после слова SELECT

Агрегированные функции:
sum- сумма значений
avg- среднее значение
min- минимальное значение
max- максимальное значение
count- количество значений

Пример 5
```
select sum((summa*10/100)) as [Всего]
from payment
```
В результате этого запроса будет вычислена общая сумма всех отчислений в размере 10% по полю summa, а результат выведен под псевдонимом Всего.

Запросы по нескольким таблицам.

В случае, если запрос формируется по нескольким таблицам, должен использоваться оператор JOIN.
Оператор JOIN используется со следующими параметрами связи:
INNER – при задании этого параметра каждая из двух участвующих в связывании таблиц будет включать только те строки, для которых есть соответствие во 2-й таблице. Данный тип связи используется по умолчанию и его можно не указывать.
LEFT – в этом случае в левой таблице будут оставлены все строки , независимо от того есть ли для них соответствие в правой таблице.
RIGHT – действие обратное предыдущему.
FULL – разрешает использовать все строки связанных таблиц.

Пример 6

```
select students.surname,  payment.summa
from students join  payment on students.ID_st=payment.ID_st
```

В примере выводятся по одному полю из 2-х таблиц

Пример 7

```
select students.surname,  payment.summa, groups.group_name
from groups
join students on groups.ID_gr=students.ID_gr
join payment on students.ID_st=payment.ID_st
```

Информация в этом примере берется из трех связанных таблиц

Пример 8

Пример сложного запроса, в котором добавлены условия отбора

```
select surname as [имя],  summa as [оплата], group_name as [группа]
from groups
 join students on groups.ID_gr=students.ID_gr
 join  payment on students.ID_st=payment.ID_st
where (surname like 'A%') and (group_name like 'mks%')
```

Запросы с группировкой. Группировка данных.

Полный синтаксис инструкции SELECT имеет следующий вид:

SELECT - выбирает столбцы в запрос
FROM - указывает источник столбцов
WHERE – условия отбора
GROUP BY – указывает столбец для группировки
HAVING - указывает условия группировки
UNION - служит для объединения двух запросов
ORDER BY - упорядочение результатов запроса
COMPUTE - позволяет выводить отдельной строкой результаты выполнения агрегирующих функций

Инструкция HAVING по своим функциям похожа на WHERE, но используется только вместе с группировкой, так как позволяет ограничивать набор строк подвергаемых группировке.

В примере вычисление суммы будет выполнено только применительно к группе записей, которые соответствуют KIS-06:

```
use example
go
select  sum(payment.summa), groups.group_name
from students join payment on students.ID_st= payment.ID_st join groups on
students.ID_gr=groups.ID_gr
group by group_name
having group_name='KIS-06
```

В случае если требуется слить несколько таблиц или запросов в один массив используется инструкция UNION. Причем необходимо указывать одинаковое количество столбцов с совместимыми типами. Например, в базе данных имеется таблица с данными об абитуриентах. Требуется получить запрос, содержащий полный список студентов и абитуриентов:

```
use example
select surname, name
from students
union
select surname, name
from abiturienty
```

Инструкция ORDER BY определяет сортировку (ASC – возрастание, DESC – убывание).

```
select ID_st, surname, name
from students
order by surname asc
```

Агрегирование данных с помощью инструкции COMPUTE отличается от агрегирования с помощью инструкции GROUP BY тем, что последняя выдает только результат агрегирования, а COMPUTE выводит результат агрегирования как дополнительный параметр.

Синтаксис:

```
SELECT
COMPUTE функция агрегирования (имя столбца)
BY  имя столбца
```

После ключевого слова COMPUTE перечисляются имена столбцов, к которым применяются функции агрегирования, причем имена этих столбцов обязательно должны быть указаны и после инструкции SELECT.

В приведенном примере запрос выводит 2 столбца: фамилии и оплату и подводит итого по полю оплата (суммарное и среднее значение):

```
select  students.surname,payment.summa
from students join payment on students.ID_st=payment.ID_st
compute sum(summa), avg(summa)
```

Инструкцию COMPUTE можно использовать вместе с группировкой при этом столбцы, по которым будет выполнена группировка, указываются после инструкции ORDER BY. Инструкция BY указывает порядок проведения группировки по столбцам.

В приведенном ниже примере, кроме того, что вычисляется среднее и суммарное значение по полю оплата, проводится группировка по фамилии:

```
select  students.surname,payment.summa
from students join payment on students.ID_st=payment.ID_st
order by surname
compute sum(summa), avg(summa)
by surname
```

Пример группировки по двум полям курс и группа:

```
select  students.course,payment.summa, groups.group_name
from students join payment on students.ID_st=payment.ID_st join groups on
students.ID_gr=groups.ID_gr
order by course, group_name
compute sum(summa), avg(summa)
by course, group_name
```

Модуль №2

Лабораторных занятий - 4 ч.

Темы практических занятий:

- Правила в SQL. Ограничения в SQL. Умолчания в SQL.
- Использование Enterprise Manager для создания правил, умолчаний, ограничений.

Правила

Для создания правила необходимо выбрать в базе данных группу правила (Rules), а затем команду в Action→New rule. В появившемся окне ввести имя правила и текст правила например:
@var1>=0 and @var1<=20000
где @var1- переменная. Имена переменных начинаются с символа @.
Данное правило ограничивает данные интервалом от 0 до 20000.

Далее это правило необходимо назначить конкретному объекту (полю базы данных), используя хранимую процедуру:

sp_bindrule 'имя правила', 'имя объекта'

например, правило rule1 будет назначено столбцу summa таблицы payment:

sp_bindrule 'rule1', 'payment.summa'

После чего SQL Server не позволит в это поле ввести отрицательное значение или значение, превышающее 20000.

Пример ограничения на символьное или текстовое поле:

@var1 like 'KIS%' or @var1 like 'MKS%'
это правило позволит вводить в поле только группы KIS и MKS

Для работы с правилами используется еще 2 хранимые процедуры

sp_unbindrule 'имя объекта' – удаляет связь между правилом и полем
sp_helptext 'имя правила' - выводит код правила.

Для создания правила используется команда:
create rule ‘имя’ as текст правила

Для удаления правила:
drop rule ‘имя’

Умолчания.

Для работы с умолчаниями используют следующие хранимые процедуры:
sp_bindefault имя , 'имя объекта'
sp_unbindefault 'имя объекта' – удаляет связь между умолчанием и полем
sp_helptext 'имя правила' - выводит код правила.

Для создания умолчания используется команда:
create default ‘имя’ as текст умолчания
Для удаления умолчания:
drop default ‘имя’

Модуль №3

Лабораторных занятий - 16 ч.

Темы практических занятий:

- Создание хранимых процедур. Объявление переменных.
- Вызов процедур. Назначение специальных хранимых процедур.
- Триггеры. Синтаксис триггера. Временные таблицы.
- Вызов хранимых процедур в триггере.
- Программирование курсоров. Открытие курсора. Закрытие курсора. Основные функции, применяемые для работы с курсором.
- Использование библиотеки MFC для доступа к базам данных MS SQL Server.
- Наращивание возможностей приложения. Создание диалоговых окон.

Создание хранимых процедур.

Хранимая процедура – это подпрограмма, которая выполняется на сервере. Процедура может использоваться несколькими клиентскими приложениями. Хранимые процедуры могут активизироваться клиентскими приложениями и триггерами. Они пишутся на языке SQL и могут иметь входные, выходные параметры или не иметь их.

Процедуры создаются через Enterprise Manager или Query Analyzer.
В Enterprise Manager нужно выбрать объект Stored Procedures, правой кнопкой мыши вызвать команду New Stored Procedure.В появившемся окне можно ввести текст хранимой процедуры.

CREATE PROCEDURE <имя процедуры> as

Тело процедуры

В процедуре могут использоваться:

1. Комментарии (-- или /* */).

2. Объявления переменных.

Для объявления переменных используется следующая инструкция:
DECLARE @имя_переменной тип данных
Например:
DECLARE @var int

3. Операторы присваивания.
SET @ переменная=выражение

4. Блоки инструкций, ограничиваемые ключевыми словами:
BEGIN
…
END
Блоки инструкций используются для организации ветвлений и циклов.

5. Инструкция PRINT для вывода сообщений в вызывающую программу (например Query Analyzer):

PRINT 'сообщение'
Сообщений может содержать до 8000 символов.
6. Ветвление.
IF логическое выражение
….
ELSE
…..

Ограничиваются блоки IF и ELSE оператором BEGIN … END.

6. Циклы.
WHILE (логическое выражение)

BEGIN

…
END

7. Инструкция **GOTO** для выполнения безусловного перехода.

GOTO метка
....
метка:

8. Инструкция **WAITFOR**
Для приостановления выполнения процедуры, имеет 2 формы:

WAITFOR DELAY ’00:10:30’

Слово DELAY определяет задержку на 10 минут 30 секунд (максимальная задержка 24 часа).

WAITFOR TIME ’14:30’

Слово TIME определяет задержку до половины третьего.

Пример 1

Пример создания процедуры в Query Analyzer подсчитывающей количество городов в таблице cities:

```
create procedure first_p
as
declare @count int          --объявляется целая переменная @count
select  @count=count(city_name)from  cities   -- функцией count вычисляется количество значений в поле city
return @count    -- функция возвращаете переменную @count
```

Для того чтобы вызвать эту процедуру необходимо записать следующее:

```
declare @d int  --объявляется переменная целого типа
 exec @d=first_p   -- вызывается процедура и ее результат присваивается переменной @d
print @d –печатается результат сохраненный в переменной @d
```

последнее действие также может быть выведено следующим образом:

```
select @d
```

Приведенная процедура имеет только возвращаемое значение, следующая процедура имеет входной параметр, объявленный в скобках @c, тип этого параметра совпадает с типом поля city таблицы cities.
Процедура принимает в качестве параметра название города и подсчитывает сколько раз этот город встречается в базе данных.

Пример 2

```
CREATE procedure second_p (@c char(20))
as
declare @count int
select @count=count(city_name) from cities where city_name=@c
return  @count
```

Вызов процедуры следующий:

```
declare @d int
 exec @d=second_p 'Bishkek'
select @d
```

В этом коде после имени процедуры указан входной параметр.

Пример 3

```
CREATE procedure third_p (@d int)
as
if exists (select id_st from payment where summa=@d)
print  'yes'
else
print 'no'
```

В этом примере процедура принимает входной параметр (например 0) и проверяет существуют ли (инструкция exists) значение 0 в поле summa и выводит соответствующее сообщение.

Вызов процедуры следующий:
exec third_p 0

Для работы с процедурами применяются специальные хранимые процедуры:
sp_help ‘имя_процедуры’ - возвращает имя владельца, дату и время создания
sp_helptext ‘имя_процедуры’ – возвращает исходный код процедуры
sp_depends ‘имя_процедуры’ - возвращает список объектов на которые ссылается процедура
sp_rename ‘старое имя’, ‘новое имя’ – переименовывает процедуру.

Триггеры.

Триггер – это специальный тип хранимой процедуры. Триггер автоматически активизируется при выполнении операции, с которой он связан (обновление, удаление, добавление).

В триггере запрещается выполнять следующие операции:

1. создание и удаление правил и умолчаний;
2. создание и удаление триггеров;
3. создание и удаление индексов;
4. создание, изменение и удаление хранимых процедур;
5. создание, изменение и удаление таблиц;
6. создание, изменение и удаление представлений.

Триггер невозможно вызвать командой **exec**, триггер не может иметь параметров и не может возвращать значение.
Для создания триггера в **Enterprise Manager** выбрать правой кнопкой мыши таблицу, далее команду **All tasks →Manage triggers**. Или в дизайнере таблицы с помощью кнопки Triggers на панели инструментов.

Синтаксис триггера:

```
CREATE  TRIGGER  имя_триггера ON имя_таблицы

FOR  [DELETE], [INSERT], [UPDATE]
AS

IF UPDATE (column)
```

SQL STATEMENT

FOR – указывает тип создаваемого триггера
DELETE –создаваемый триггер будет вызываться при попытке удаления данных из таблицы.
INSERT- будет вызван при попытке добавления новых строк в таблицу
UPDATE – будет вызван при попытке изменения данных в указанной таблице.

AS - на этом слове заканчивается определение триггера и описываются действия, которые будет выполнять триггер.

IF UPDATE (column) – разрешает выполнение триггера только при выполнении изменений в определенном столбце таблицы, имя которого указывается в скобках. Эта опция указывается только для триггеров типа INSERT и UPDATE. Чтобы разрешить вызов триггера при изменении нескольких столбцов имена этих столбцов указываются с помощью параметра **UPDATE (column),** объединяется множество таких параметров с помощью логических операторов OR или AND.

Например:

IF UPDATE (column_1) OR UPDATE(column_2) OR UPDATE(column_3)

SQL STATEMENT – указываются команды на языке SQL, выполняемы при вызове триггера, допускаются команды выборки данных, цикла, изменения, вставки и удаления строк.

При запуске триггеров создаются 2 временные таблицы INSERTED и DELETED, которые хранятся в оперативной памяти. Временные таблицы имеют туже структуру, что и базовые таблицы.
DELETED содержит удаленные строки таблицы, с которой связан данный триггер.
INSERTED содержит все новые обновленные строки таблицы, с которой связан триггер.

Пример1.

```
CREATE TRIGGER check1 ON [dbo].[cities]
FOR  DELETE
AS
ROLLBACK TRANSACTION -- откат транзакции (отмена удаления)
```

Таблица cities имеет триггер который запускается при попытке удаления из нее записи

Пример2.

```
CREATE TRIGGER check2 ON [dbo].[cities]
FOR  UPDATE
AS
IF UPDATE(city_name)
ROLLBACK TRANSACTION
```

Второй пример запрещает внесение изменений в поля city и каждый раз производит откат транзакции (отмену действия)

Пример3.

```
CREATE TRIGGER check1 ON [dbo].[cities]
FOR   DELETE
AS
DECLARE @del INT
BEGIN
SELECT @del=id_ct FROM deleted –присваивает переменной @del значение кода города,
                                                находящегося в таблице deleted
IF EXISTS (SELECT* FROM students WHERE id_ct=@del)
ROLLBACK TRANSACTION
END
```

Этот триггер при удалении города из таблицы cities проверяет, есть ли студенты, проживающие в этом городе в таблице students. Если есть производится откат транзакции.

Пример3а.

```
CREATE TRIGGER check1 ON [dbo].[cities]
FOR   DELETE
AS
DECLARE @del int, @n int
BEGIN
SELECT @del=id_ct FROM deleted
EXEC  @n=check_city @del
IF @n<>0
ROLLBACK TRANSACTION
END
```

В триггере допускается вызов хранимых процедур. В данном примере процедура **check_city** принимает параметр @del, в котором хранится код удаленного города и подсчитывает сколько записей в таблице students соответствует этому городу. Количество записей сохраняется в переменной @n. Если значение этой переменной не равно нулю, то производится отмена удаления.

Текст процедуры check_city:

```
CREATE PROCEDURE check_city (@c int) AS
declare @a int
select @a= count(students.id_ct) from students where id_ct=@c
return @a
```

Для вывода сообщений в триггерах и хранимых процедурах о возникновении ошибок используется инструкция:

RAISERROR (сообщение, серьезность, состояние)

Сообщение может содержать до 8000 символов и должно быть заключено в одинарные кавычки.

Серьезность – это параметр в диапазоне от 0 до 25. Значение от 20 и выше вызывают прекращение выполнимой процедуры или триггера. Кроме того, значения выше 18 могут использоваться при выдаче сообщений только администраторами.

Состояние – целочисленный параметр в диапазоне от 1 до 127.

Пример3б.

```
CREATE TRIGGER check1 ON [dbo].[cities]
FOR  DELETE
AS
DECLARE @del int, @n int
BEGIN
SELECT @del=id_ct FROM deleted
PRINT  @del
EXEC  @n=check_city @del
IF @n<>0
BEGIN
RAISERROR ('Нельзя удалить данную запись', 16,1)
ROLLBACK TRANSACTION
END
END
```

Для работы с триггерами применяются специальные хранимые процедуры
sp_help ‘имя_триггера’ - возвращает имя владельца, дату и время создания
sp_helptext ‘имя_триггера’ – возвращает исходный код триггера
sp_depends ‘имя_триггера’ - возвращает список объектов на которые ссылается триггер
sp_helptrigger ‘имя таблицы’ –возвращает список триггеров определенных для данной таблицы.

Программирование курсоров.

Курсор представляет собой некий объект, обеспечивающий доступ к любой из строк (путем перемещения от строки к строке) набора данных и, соответственно, возможность выполнения с ней операции.

Курсоры (указатели набора записей) используются для создания многостраничных запросов. Курсоры предоставляют пользователю возможность доступа к любой строке выборки по ее номеру.

Курсор – это особый временный объект, предназначенный для использования в программах и хранимых процедурах. С его помощью можно в цикле пройти по результирующему набору строк запроса, по отдельности считывая и обрабатывая каждую его строку. В хранимых процедурах с помощью курсоров можно выполнить сложные вычисления. Кроме того, курсоры позволяют выполнить выборочное обновление и удаление строк результирующей таблицы запроса.

Существуют следующие виды курсоров:

1. Статические – при создании статического курсора сервер сохраняет полученные данные в системной базе данных tempdb, предназначенной для временного хранения данных. При работе с такими курсорами сервер обращается не к исходным таблицам, а к данным сохраненным в базе данных tempdb. В статических курсорах не поддерживается отображение изменений, сделанных в исходных

данных. При помощи статических курсоров пользователь может работать с данными, которые уже не существуют в исходных таблицах или были изменены таким образом, что при повторном создании статического курсора они бы уже не вошли в результирующий набор. Создаются с параметром STATIC.

2. Ключевые – курсоры, базирующиеся на наборе ключей, представляют собой набор ссылок на строки, которые удовлетворяют условиям отбора, указанным в инструкции where. Ключевые курсоры в отличие от статических выбирают не всю строку, а лишь ключевые поля.
3. Последовательные – предназначены для одиночного, последовательного сканирования всех строк курсора. Этот тип курсоров поддерживает только обращение к следующей строке курсора и не позволяет вернуться к любой из ранее выбранных строк, не позволяет перескакивать через одну или более строк. Пользователь может перебирать одну строку за другой, начиная с первой и заканчивая последней. Создаются с параметром FORWARD_ONLY.
4. Динамические - позволяют обращаться к любой строке результирующего набора. При каждом обращении к строкам сервер заново выполняет ассоциированный с курсором запрос SELECT, обновляя тем самым результирующий набор. Если в исходный набор пользователем были внесены изменения, влияющие на список строк результирующего набора, то такие изменения будут автоматически отражены в курсоре. Создаются с параметром DYNAMIC .

В Transact-SQL предусмотрены следующие пять основных операций с курсорами:

1. Создание курсора;
2. Открытие курсора;
3. Манипуляция со строками курсора:
   - считывание данных из курсора;
   - изменение данных посредством курсора;
   - удаление данных посредством курсора.
4. Закрытие курсора;
5. Освобождение курсора.

Для работы с курсорами используются следующие операторы SQL

1. Оператор DECLARE CURSOR – определяет выполняемый запрос, задает имя курсора и связывает результаты запроса с заданным курсором. Этот оператор определяет структуру будущего множества записей и связывает ее с уникальным именем курсора.
2. Оператор OPEN дает команду СУБД выполнить описанный запрос, создать виртуальный набор строк, который соответствует заданному запросу. Оператор OPEN устанавливает курсор перед первой строкой виртуального набора строк результата запроса.
3. Оператор FETCH продвигает указатель записей на следующую позицию в виртуальном наборе записей.
4. Оператор CLOSE закрывает курсор и прекращает доступ к виртуальному набору записей. Он ликвидирует связь между курсором и результатом выполнения базового запроса.

DECLARE имя_курсора параметры CURSOR FOR SELECT_запрос

Курсор может объявляться со следующими параметрами:

LOCAL – ключевое слово, указывающее на создание локального курсора, видимого в пределах создавшего его триггера, процедуры и т. д. По завершении процедуры такой курсор автоматически уничтожается.
GLOBAL – создается глобальный курсор, который будет существовать до завершения соединения в котором он был создан.
FORWARD_ONLY – опция, устанавливающая, что создается последовательный курсор. Перемещение по нему возможно только последовательно от первой строки к последней.
SCROLL - ключевое слово, определяющее механизм перемещения по строкам. При его наличии курсор можно прокручивать в обе стороны. Если это слово отсутствует, то курсор можно просматривать только от начала к концу.
KEYSET - создается ключевой курсор. Ключевой курсор представляет собой набор ключей, идентифицирующих строки полного результирующего набора курсора.
DYNAMIC - создается динамический курсор.
INSENSITIVE – опция, устанавливающая, что (при ее наличии) создается статический курсор, в противном случае динамический.
SCROLL_LOCKS – сервер блокирует все строки, входящие в результирующий набор. Это повышает скорость операций чтения и модификации данных с помощью курсора, т.к. не надо ожидать разблокирования нужных ресурсов.

После объявления курсора его необходимо открыть:

OPEN имя_курсора

После открытия курсора пользователь может приступить к выборке и изменению данных
Для считывания строки данных из курсора предназначена команда FETCH, которая имеет следующий синтаксис:

FETCH
Операции
FROM
Имя курсора
INTO
@var1, @var2,@var3,...,@varN

Инструкция FETCH может включать следующие операции:
FIRST – на первую строку
LAST – на последнюю строку
PRIOR – на строку, предшествующую текущей
NEXT - на строку, следующую за текущей
RELATIVE - n>1 на n-ю строку вперед от текущей
n=1 эквивалентно ключевому слову NEXT
n=0 повторное чтение текущей строки
n<-1 на n-ю строку назад от текущей
n=-1 эквивалентно ключевому слову PRIOR

ABSOLUTE n – n>0 на n-ю строку
n=0 никакая строка не возвращается
n<0 на n-ю строку с конца

@var1, @var2,@var3,...,@varN - имена переменных, в которых будут сохранены значения всех столбцов текущей строки курсора

После выполнены операции с применением курсора его необходимо закрыть. Закрытие курсора подразумевает освобождение всех областей памяти (оперативной и базы данных tempdb), используемых для хранения данных курсора. Но при этом не происходит удаление курсора как объекта. И при повторном открытии курсор будет содержать более свежие данные.

Синтаксис закрытия курсора:
CLOSE имя_курсора

Для полного удаления всей информации, связанной с курсором, включая его имя, и код запроса. Необходимо выполнить освобождение курсора:

Синтаксис освобождения курсора
DEALLOCATE имя_курсора

Для работы с курсорами применяется функции:

@@FETCH_STATUS- проверяет не выполняется ли обращение к строке, находящейся за пределами курсора. Если функция возвращает значение 0 то операция чтения строки не вышла за пределы курсора, если же функция возвращает –1, значит была попытка чтения строки, находящейся за пределами результирующего набора. Если функция возвращает значение –2, значит, пользователь попытался выбрать поврежденную строку.

@@CURSOR_ROWS - возвращает количество строк, полученных в последнем открытом соединении курсора. Если эта функция возвращает значение 0, это означает что, либо курсор не содержит ни одной строки, либо в соединении не было открыто ни одного курсора.

Пример 1.

```
DECLARE curs3 CURSOR SCROLL
FOR
SELECT surname, course
FROM students
```

OPEN curs3

```
DECLARE @s char(20), @c int
SELECT @@CURSOR_ROWS
FETCH FIRST FROM curs3 INTO @s, @c
SELECT [ФАМИЛИЯ]=@s,[КУРС]=@c
FETCH NEXT FROM curs3 INTO @s, @c
SELECT [ФАМИЛИЯ]=@s, [КУРС]=@c
FETCH NEXT FROM curs3 INTO @s, @c
SELECT [ФАМИЛИЯ]=@s,[КУРС]=@c
FETCH NEXT FROM CURS3 INTO @s, @c
SELECT [ФАМИЛИЯ]=@s,[КУРС]=@c
CLOSE curs3
DEALLOCATE curs3
```

Пример 1а.

```
DECLARE curs3 CURSOR SCROLL FOR
SELECT surname, course FROM students
OPEN curs3
```

```
DECLARE @s char(20), @c int, @n int
SELECT @@CURSOR_ROWS  - - определяется количество строк, в курсоре
FETCH ABSOLUTE 9 FROM curs3 INTO @s, @c - -из курсора извлекается 9-я строка
SELECT [ФАМИЛИЯ]=@s, [КУРС]=@c
CLOSE curs3
DEALLOCATE curs3
```

Курсоры могут использоваться для удаления и изменения данных, при этом используются позиционные операции.
Предложение WHERE CURRENT OF имя_курсора, используемое в инструкциях UPDATE и DELETE указывает, какая строка таблицы должна быть обновлена или удалена. Эти две операции называют позиционным обновлением и позиционным удалением.

Пример 2.

```
DECLARE curs4 CURSOR  FOR
SELECT ID_st, summa FROM payment
OPEN curs4
DECLARE @s int, @c int

WHILE(0=0)
BEGIN
        FETCH NEXT FROM curs4 INTO @s, @c
        IF(@c=0 or @c is NULL)
                BEGIN
                  DELETE FROM payment
                  WHERE CURRENT OF curs4
                END
        ELSE IF(@c<8500)
                BEGIN
                  UPDATE payment
                  SET summa=100
                  WHERE CURRENT OF curs4
                  SELECT @s, @c

                END
if(@@fetch_status<>0) break
END
CLOSE curs4
DEALLOCATE curs4
```

Этот пример создает набор записей, в который входит два поля student и summa, затем это набор проверяется и в случае если в поле summa встречается сумма оплаты равная нулю или значению NULL, то данная строка удаляется из таблицы (т.е. происходит отчисление студентов, не оплативших учебу). Если же сумма оплаты ниже значения 8500, в поле записывается сумма 100.

Пример 3.

```
DECLARE curs4 SCROLL CURSOR  FOR
SELECT surname, summa FROM students join payment on ID_st=ID_st
OPEN curs4
DECLARE @s char(20), @c int
```

```
while(0=0)
begin
        fetch next from curs4 into @s, @c
        if(@c<8500)
                select @s,@c

        if(@@fetch_status<>0) break
end

close curs4
deallocate curs4
```

Этот пример показывает возможность, включения в курсор данных из нескольких таблиц.

Курсоры делятся на курсоры клиента и курсоры сервера. Курсор сервера создается и выполняется на сервере. Курсоры сервера определяются в хранимых процедурах или в триггерах.

Курсоры клиента создаются на клиентской машине. Набор строк, связанных с этим курсором пересылаются на клиент и там обрабатывается. Если с курсором связан большой набор данных, то операция пересылки данных может занять значительное время и значительные ресурсы клиентского компьютера.

Пример создания курсора в хранимой процедуре.

Пример 4.

```
CREATE PROCEDURE CURS_CHECK
AS
DECLARE curs4 CURSOR  FOR
SELECT ID_st, summa FROM payment
OPEN curs4
DECLARE @s int, @c int
declare @k int
set @k=0
while(0=0)
begin

        fetch next from curs4 into @s, @c
        if(@c=0)
                begin
                set  @k=@k+1
                end
if(@@fetch_status<>0) break

end
select @k
close curs4
deallocate curs4
```

Вызов хранимой процедуры:

```
EXEC CURS_CHECK
```

Раздел 6.
Методические рекомендации по СРС.

Модуль № 1

Самостоятельная работа студента – 14 ч.

Самостоятельная работа студентов

- Понятие нормальная форма. Пять нормальных норм.
- Установка сервера MS SQL Server.
- Настройка сервера MS SQL Server.
- Служба SQL Server Agent. Настройка SQL Server Agent.
- Программа SQL Server Profiler.

Рекомендуемая литература:

1. Р. Шнейдер. SQL Server. Создание эффективных баз данных. Москва 2000.
2. Роберт Виейра. Программирование баз данных Microsoft SQL Server 2008. Базовый курс. Издательства: Диалектика, Вильямс, 2010 г.

Модуль №2

Самостоятельная работа студента – 6 ч.

Самостоятельная работа студентов.

- Встроенные функции языка Transact-SQL. (Математические, строковые, функции даты-времени, системные и т.д.)
- Типы переменных в Transact-SQL.

Рекомендуемая литература:

1. Мартин Грабер. SQL: справочное руководство. Москва. Издательство «Лори» 1998.
2. Мартин Грабер. Введение в SQL. Москва. Издательство «Лори» 1998.

Модуль №3

Самостоятельная работа студента – 22 ч.

Самостоятельная работа студентов

- Системные процедуры.
- Использование ODBC для доступа к SQL Server .
- Использование библиотеки MFC для доступа к базам данных SQL Server.
- Использование Web-технологий в MS SQL Server .
- Импорт и экспорт данных в MS SQL Server .
- Язык XML.

Рекомендуемая литература:

1. М. Хотек. Microsoft SQL Server 2008. Реализация и обслуживание. Издательство: Русская Редакция, 2011 г.
2. Майк Гандерлой, Джозеф Джорден, Дейвид Чанц. Освоение Microsoft SQL Server 2005. Издательство: Вильямс, 2007 г.

Раздел 7.
Самостоятельная работа студента под руководством преподавателя.

Модуль № 1

Самостоятельная работа с преподавателем – 7 ч.

Самостоятельная работа с преподавателем.

- Различие архитектур DAO и RDO и способов доступа к базам данных.
- Различие маршрутов приложений в ADO.
- Различие маршрутов приложений в ADO.NET.
- Изменение таблицы, ввод данных в таблицу.
- Запросы по нескольким таблицам.

Рекомендуемая литература:

1. Джефри Д.Шенк. Технология клиент-сервер и ее приложения. Издательство «Лори» 1999. Москва.
2. Р. Шнейдер. SQL Server. Создание эффективных баз данных. Москва 2000.

Модуль №2

Самостоятельная работа с преподавателем – 3 ч.

Самостоятельная работа с преподавателем.

- История SQL.
- Объекты базы данных SQL Server.

Рекомендуемая литература:

1. Пирогов В.Ю. «MS SQL Server 2000: управление и программирование». Спб., БХВ-Петербург, 2005.
2. Пол Нильсен. SQL Server 2005. Библия пользователя. Издательство: Вильямс, 2008 г.

Модуль №3 (8 недель).

Самостоятельная работа с преподавателем – 11 ч.

Самостоятельная работа с преподавателем.

- Применение связанных серверов.
- Ограничения. Индексы. Ключи.
- Шифрование данных.
- Операторы присваивания, блоки инструкций, ветвление, циклы.
- Вывод сообщений об ошибках в триггерах и хранимых процедурах.
- Синтаксис команд для работы с курсором.

Рекомендуемая литература:

1. Александр Волоха. Microsoft SQL Server 2005. Новые возможности. Спб., БХВ-Петербург, 2005.
2. М. Хотек. Microsoft SQL Server 2008. Реализация и обслуживание. Издательство: Русская Редакция, 2011 г.

Раздел 8.
Контрольно-измерительные средства.

ВОПРОСЫ ДЛЯ МОДУЛЬНО-РЕЙТИНГОВОЙ АТТЕСТАЦИИ

Модуль 1. ***Контрольные вопросы:***

1. Технологии обработки данных.
2. Архитектура клиент-сервер.
3. Архитектура файл-сервер.
4. Трехслойная архитектура.
5. Эволюция технологий доступа к данным.
6. Архитектура технологии ODBC.
7. Технологии DAO и RDO. Различие архитектур.
8. Технология OLE DB. Компоненты OLE DB.
9. Технология доступа к базам данных ADO.
10. Объектная модель ADO. Различие маршрутов приложений в ADO.
11. Развитие технологии доступа к базам данных ADO.NET.
12. Архитектура ADO.NET.
13. Различие маршрутов приложений в ADO.NET.
14. Актуальность технологии ADO.NET.
15. Создание таблиц. Создание простой таблицы.
16. Создание таблиц. Создание таблицы с ограничениями.
17. Использование внешнего ключа.
18. Создание вычисляемых полей в таблице.
19. Ввод данных в таблицу. Изменение таблицы.
20. Запросы. Запросы на выборку.
21. Запросы в Enterprise Manager.
22. Запросы в Query Analyzer.
23. Запросы из нескольких таблиц.
24. Агрегированные функции.
25. Группировка данных.
26. Инструкции в SQL.

Модуль 2. ***Контрольные вопросы:***

1. Технологии удаленного доступа к системам баз данных, тиражирование и синхронизация в распределенных системах баз данных.
2. Интерфейс CGI. Доступ к СУБД через CGI-программу.
3. Интерфейсы прикладного программирования.
4. Интерфейсы API и FastCGI.
5. Правила в SQL.
6. Ограничения в SQL.
7. Умолчания в SQL.
8. Индексы.
9. Ключи.

Модуль 3. ***Контрольные вопросы:***

1. Реляционные базы данных.
2. Язык SQL. История SQL.
3. Логическая архитектура Microsoft SQL Server.
4. Активное администрирование и объектный интерфейс SQL Server.
5. Создание хранимых процедур.
6. Объявление переменных.
7. Операторы присваивания, блоки инструкций, ветвление, циклы.

8. Вызов процедур.
9. Назначение специальных хранимых процедур.
10. Триггеры. Синтаксис триггера.
11. Временные таблицы.
12. Вызов хранимых процедур в триггере.
13. Вывод сообщений об ошибках в триггерах и хранимых процедурах.
14. Системные базы данных Microsoft SQL Server.
15. Роли Сервера.
16. Роли базы данных.
17. Производительность сервера.
18. Служба преобразование данных.
19. Шифрование данных.
20. Резервное копирование.
21. Определение данных и оптимизация запросов.
22. Задачи и средства администратора безопасности баз данных.
23. Виды курсоров (статические, ключевые, последовательные динамические).
24. Операторы SQL для работы с курсорами.
25. Определение курсора.
26. Синтаксис команд для работы с курсором.
27. Основные функции, применяемые для работы с курсором.
28. Включение в курсор данных из нескольких таблиц.
29. Подключение источника данных. Компиляция.
30. Функции для работы с базами данных
31. Создание приложения.
32. Создание элементов управления, с помощью редактора панелей инструментов.
33. Создание диалоговых окон.

Контрольные вопросы по предмету
«Технологии удаленного доступа к базам данных».

I

1. Технологии доступа к данным.
2. Технология ODBC.
3. Архитектура технологии ODBC.
4. Технология DAO.
5. Использование технологии DAO для доступа к базам данных.
6. Технология RDO.
7. Использование технологии RDO для доступа к базам данных.
8. Технологии RDO и DAO. Их недостаток.
9. Технология OLE DB.
10. Компоненты технологии OLE DB.
11. Технология ADO.
12. Объектная модель ADO.
13. Различие маршрутов приложений в ADO.
14. Технология ADO.NET.
15. Архитектура ADO.NET.
16. Архитектура ADO.NET. Провайдер данных.
17. Архитектура ADO.NET. DataSet.
18. Различие маршрутов в ADO.NET. Провайдеры данных .Net.
19. Технологии обработки данных.
20. Технология файл-сервер. Достоинства, недостатки.
21. Технология клиент-сервер. Достоинства, недостатки.
22. Доступ к СУБД через CGI-программу.
23. Интерфейс CGI.
24. Интерфейс FastCGI.
25. Интерфейсы прикладного программирования ISAPI и NSAPI.
26. Язык XML.

II

1. Возможности сервера MS SQL 2000.

Архитектура баз данных SQL Server.

2. Файлы.
3. Объекты базы данных SQL Server.
4. Типы данных.
5. Представления. Агрегированные функции.
6. Правила и умолчания.
7. Хранимые процедуры.
8. Расширенные хранимые процедуры.
9. Функции пользователя. Типы данных, определяемые пользователем.
10. Триггеры.
11. Системные базы данных.

Средства управления MS SQL Server и доступа к данным.

12. Управление базой данных. (Способы создания баз данных.)
13. Производительность сервера. Программа SQL Server Profiler.

14. Применение связанных серверов.
15. Репликация. Основные термины репликации.
16. Репликация. Модели репликации.
17. Репликация. Методы обновления подписчиков.
18. Служба преобразования данных. (DTS).

Система безопасности в MS SQL Server.

19. Принципы безопасности.
20. Роли.
21. Шифрование данных.
22. Резервное копирование.

Транзакции.

23. Транзакции. Свойства транзакций.
24. Транзакции. Типы транзакций.
25. Курсоры. Виды курсоров.

III

1. Инструкция create table. Пример использования.
2. Понятие псевдонима. Пример использования.
3. Инструкция alter table. Пример использования.
4. Инструкции declare и set. Пример использования.
5. Инструкция exec. Пример использования.
6. Инструкция create database. Пример использования.
7. Функции для работы с курсорами (@@FETCH_STATUS, @@CURSOR_ROWS).
8. Инструкция RAISERROR. Пример использования.
9. Инструкция create rule. Пример использования.
10. Инструкция create default. Пример использования.
11. Инструкции order by и compute. Пример использования.
12. Инструкция create procedure. Пример использования.
13. Операторы языка SQL для работы с курсорами (declare cursor, open, fetch, close…).
14. Инструкции default и check. Пример использования.
15. Функции агрегирования.
16. Инструкция drop. Пример использования.
17. Инструкция create trigger. Пример использования.
18. Инструкции select, from и where. Пример использования.
19. Инструкция create table. Пример использования.
20. Инструкции group by и having. Пример использования.
21. Инструкция insert. Пример использования.
22. Первичный и внешний ключ. Пример использования.

Тест по предмету «Технологии удаленного доступа к базам данных».

1. Какая из технологий доступа к данным базируется на технологии Microsoft Jet?
    a) ODBC
    b) DAO
    c) ADO.NET

2. Какая технология предлагает единообразный метод доступа к данным, хранящимся в разных источниках информации, в том числе и в не реляционных БД?
    a) OLE DB
    b) RDO
    c) ADO
3. Драйвера ODBC бывают:
    a) только одноуровневые
    b) только многоуровневые
    c) одноуровневые и многоуровневые
4. Какая из технологий доступа к данным является наследницей технологий DAO и RDO?
    a) ADO
    b) ODBC
    c) JDBC
5. Основные компоненты архитектуры технологии ADO.NET:
    a) Приложение и менеджер драйверов
    b) Провайдер данных и DataSet
    c) DataRelation и DataConnection
6. При технологии «клиент-сервер» -
    a) база данных хранится на сервере, а обработка данных выполняется на одной из рабочих станций
    b) база данных хранится на рабочей станции, и обработка также происходит на рабочих станциях
    c) сервер помимо хранения данных, также выполняет и функции их обработки
7. Что является недостатком интерфейса CGI?
    a) снижение скорости обработки запросов при увеличении интенсивности их поступления
    b) плохая переносимость на другие платформы
    c) CGI приложения не могут работать независимо от Web-сервера
8. Интерфейс прикладного программирования ISAPI разработан:
    a) Корпорацией Netscape
    b) Корпорацией Microsoft
    c) Корпорацией Sun Microsystems
9. Репликация - это
    a) способ обмена информацией между базами данных, принадлежащим различным серверам
    b) процесс создания резервной копии
    c) служба преобразования данных в различные форматы
10. Основной файл - это
    a) файл, где хранятся только данные, хранящиеся в базе данных
    b) файл, где хранится системная информация и данные, хранящиеся в базе данных
    c) файл, где хранится журнал транзакций
11. Сколько системных баз данных поддерживает SQL Server 2000?
    a) 4
    b) 1
    c) 3
12. Системная база данных tempdb, используется:
    a) для хранения информации системного уровня
    b) для хранения временных объектов
    c) такой базы данных не существует
13. Инструкция GROUP BY языка SQL:
    a) указывает столбец для группировки

b) служит для объединения двух запросов
c) указывает источник столбцов

14. Триггер - это
    a) тип данных, определяемый пользователем
    b) визуальное представление таблиц и связей между ними
    c) хранимая процедура особого вида, которая выполняется при операциях трех видов: вставке, обновлении и удалении записи
15. Инструкция drop языка SQL:
    a) Используется для удаления объекта базы данных MS SQL Server
    b) Используется для создания объекта базы данных MS SQL Server
    c) Используется для изменения объекта базы данных MS SQL Server