Учреждение образования
«Брестский государственный технический университет»
Экономический факультет
Кафедра информатики и прикладной математики

СОГЛАСОВАНО
Заведующий кафедрой
_____С.И. Парфомук
«27»____12____2022 г.

СОГЛАСОВАНО
Декан факультета
_____В.В. Зазерская
«27»____12____2022 г.

ЭЛЕКТРОННЫЙ УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО ДИСЦИПЛИНЕ

«БАЗЫ ДАННЫХ»

для специальностей

1– 28 01 02 «Электронный маркетинг»,
1 – 28 01 01 «Экономика электронного бизнеса»

СОСТАВИТЕЛЬ: И.М. Гучко, старший преподаватель кафедры информатики и прикладной математики

Рассмотрено и утверждено на заседании учебно-методического совета БрГТУ
«29» 12 2022 г. Протокол № 3

рег. N УМК 22/23-42

Пояснительная записка

Электронный учебно-методический комплекс (ЭУМК) – комплекс систематизированных учебных и методических материалов предназначен для реализации требований образовательных программ и образовательных стандартов высшего образования для экономических специальностей. Дисциплина «Базы данных» является базовым курсом, обеспечивающим подготовку студента к освоению современных информационных технологий.

ЭУМК соответствует образовательным стандартам ОСВО 1 – 28 01 02 – 2021 для специальности «Электронный маркетинг», ОСВО 1 – 28 01 01 – 2021 для специальности «Экономика электронного бизнеса» (утверждены Постановлением Министерства образования Республики Беларусь от 09.02.2022 г. № 24) и учебных планов специальностей. Составлен в соответствии с действующей учебной программой по дисциплине «Базы данных» для специальностей 1– 28 01 02 «Электронный маркетинг», 1 – 28 01 01 «Экономика электронного бизнеса» рег. № УД-22-2-033/уч.

ЭУМК разработан в соответствии со следующими нормативными документами:

– положение об учебно-методическом комплексе на уровне высшего образования, утвержденном постановлением Министерства образования Республики Беларусь №167 от 26.07.2011 г.;

– положение об учебно-методическом комплексе по учебной дисциплине учреждения образования «Брестский государственный технический университет» от 31.01.2019 г.;

Материал представлен на требуемом методическом уровне и адаптирован к современным образовательным технологиям.

Цели ЭУМК:

- ✓ обеспечение качественного методического сопровождения процесса обучения будущих экономистов знаниями в области современных технологий баз данных;
- ✓ организация эффективной самостоятельной работы студентов.

Структура ЭУМК дисциплины «Базы данных»:

Теоретический раздел представлен конспектом лекций в электронном виде.

В данном методическом комплексе рассматриваются следующие основные задачи:

- ✓ изучение основ представления экономической информации в автоматизированных информационных системах;
- ✓ изучение основных понятий баз данных, моделей данных;
- ✓ изучение основ организации и функционирования баз данных;
- ✓ изучение основных понятий о назначении, архитектуре, функциональных возможностях современных систем управления базами данных (СУБД) и направлениях их развития;
- ✓ изучение основ реляционной алгебры, реляционной модели данных, основных принципов и особенностей разработки логических и физических моделей данных;
- ✓ изучение процесса проектирования и создания реляционных бах данных;

- ✓ освоение приемов работы с технологическими и инструментальными средствами построения моделей данных, создания физических объектов БД;
- ✓ изучение основных конструкций языка создания и манипулирования данными SQL;
- ✓ изучение основ функционирования распределенных и многопользовательских баз данных;
- ✓ изучение способов управления, сопровождения и администрирования баз данных и обеспечения безопасности данных;
- ✓ овладение методами и спецификой работы клиент-серверных механизмов взаимодействия с базами данных;
- ✓ освоение механизмов взаимодействия с базами данных при решении прикладных задач.

Теоретический материал представлен в виде презентаций Power Point с наглядными примерами решения прикладных задач.

Практический раздел содержит задания, инструкции и образцы выполнения лабораторных работ, выполняемых в компьютерных залах. Лабораторный практикум представлен в электронном виде в локальной сети университета.

Раздел контроля знаний содержит перечень контрольных вопросов к экзамену и примерный вариант экзаменационного билета. Указанные материалы размещены в локальной сети университета в электронном виде.

Вспомогательный раздел включает учебную программу по дисциплине «Базу данных», утвержденной Советом УО «Брестский государственный технический университет» для студентов специальностей 1– 28 01 02 «Электронный маркетинг», 1 – 28 01 01 «Экономика электронного бизнеса» рег. № УД-22-2-033/уч.

Разработанный ЭУМК способствует повышению уровня компьютерной подготовки студентов; облегчает освоение прикладных задач обработки данных во всех сферах бизнеса и экономической деятельности.

Благодаря данному комплексу будущие специалисты с квалификацией «маркетолог-программист» и «экономист-программист» получают профессиональные технологические знания и умения по выбранной специальности, формируют умения применять на практике полученные теоретические знания в области проектирования, создания и ведения баз данных.

Технической базой комплекса является локальная вычислительная сеть университета, объединяющая все компьютерные залы с возможностью беспарольного доступа в Internet и оснащённая необходимым техническим обеспечением. Важнейшим инструментом обучения являются информационные ресурсы (ПО), а также постоянно модернизируемое и пополняемое электронное методическое обеспечение (ЭМО), основой которого являются, в частности, задания и образцы решения типовых задач в компьютерных системах.

Перечень рекомендуемой литературы

1. Базы данных: Учеб. для вузов / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев; Под ред. А. Д. Хомоненко. – СПб.: КОРОНА-Век, 2009. – 736 с.
2. Бекаревич, Ю. Б. Самоучитель Access 2010 / Ю. Б. Бекаревич, Н. В. Пушкина. – СПб. : БХВ-Петербург, 2011. – 432 с.

3. Быков, В. Л. Система автоматизированного проектирования баз данных Microsoft Access 2010: учебно-методическое пособие / В. Л. Быков, И. М. Гучко, А. М. Кулешова. – Брест : БрГТУ, 2014. – 75 с.
4. Грабер, М. Введение в SQL / М. Грабер. – Москва : Лори, 2019. – 378 с.
5. Грофф, Д. Р. SQL: полное руководство / Д. Р. Грофф, П. Н. Вайнберг, Э. Дж. Оппель. – Москва : Вильямс, 2018. – 960 с.
6. Гурвиц, Г. А. Microsoft Access 2010. Разработка приложений на реальном примере. / Г. А. Гурвиц. – СПб.: БХВ-СПб, 2010. – 469 с.
7. Гучко, И. М. Создание баз данных в среде СУБД Microsoft Access: пособие для самостоятельной работы студентов экономических специальностей / И. М. Гучко, Е. Н. Рубанова, И. Н. Аверина. – Брест : Изд-во БрГТУ, 2018. – 72 с.
8. Гучко, И. М., Рубанова, Е. Н. Методические указания по курсу "Компьютерные информационные технологии" 2-й раздел: "Технологии баз данных и знаний" для студентов экономических специальностей / Министерство образования Республики Беларусь, Брестский государственный технический университет, Кафедра информатики и прикладной математики ; сост.: И. М. Гучко, Е. Н. Рубанова. – Брест : БрГТУ, 2016. – 73 с.
9. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт. – 8-е изд. – Москва : Диалектика, 2019. – 1328 с.
10. Диго, С. М. Базы данных: проектирование и использование: учебник / С. М. Диго. – Москва : Финансы и статистика, 2005. – 592 с.
11. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для вузов / В. М. Илюшечкин. – Москва : Издательство Юрайт, 2021. – 213 с.– (Высшее образование). – ISBN 978-5-534-03617-6. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/468367 (дата обращения: 14.12.2022).
12. Кириллов, В. В. Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов. – Санкт-Петербург : БХВ-Петербург, 2012. – 464 с.
13. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. – Москва : Вильямс, 2017. – 1436 с.
14. Кренке, Д. Теория и практика построения баз данных. – СПб. : Питер, 2005. – 800 с.
15. Кузин, А. В. Базы данных: учеб. пособие / А. В. Кузин, С. В. Левонисова. – М. : Academia, 2012. – 320 с.
16. Кузин, А. В., Демин, В. М. Разработка баз данных в системе Microsoft Access: учебник. – М. : ИНФРА-М, 2014. –223 с.
17. Кузнецов, С. Д. Базы данных. Модели и языки: учебник / С. Д. Кузнецов. – Москва : Бином, 2008. – 720 с.
18. Левчук, Е. А. Технологии организации, хранения и обработки данных / Е. А. Левчук. – 3-е изд. – Минск : Выш. шк., 2007. – 239 с.
19. Малыхина, М. П. Базы данных: основы, проектирование, использование : учебное пособие / М. П. Малыхина. – 2-е изд. – Санкт-Петербург : БХВ–Петербург, 2006. – 528 с.
20. Марков, А. С. Базы данных: введение в теорию и методологию : учебник [рек. УМО РФ] / А. С. Марков, К. Ю. Лисовский. – Москва : Финансы и статистика, 2006. – 512 с.

21. Нестеров, С. А. Базы данных: учебник и практикум для СПО / С.А. Нестеров. – М. : Издательство Юрайт, 2019. – 230 с.
22. Одиночкина, С. В. Разработка баз данных в Microsoft Access 2010. СПб. : НИУ ИТМО, 2012. – 83 с.
23. Осипов, Д. Л. Технологии проектирования баз данных. – М. : ДМК Пресс, 2019. – 498 с.
24. Оскерко, В. С. Технологии баз данных: учеб. пособие/ В. С. Оскерко, З. В. Пунчик. Мн. : БГЭУ, 2015. – 215 с.
25. Редько, В. Н. Базы данных и информационные системы / В. Н. Редько, И. А. Бассараб. – Москва : Знание, 2011. – 602 с.
26. Сеннов, А. Access 2010. Учебный курс. – СПб.: Питер, 2010. – 288 с.
27. Смирнова, О. В. Access 2007 на практике. М.: Феникс, 2009. – 155 с.
28. Стружкин, Н. П. Базы данных: проектирование. Практикум: учеб. пособие для СПО / Н. П. Стружкин, В. В. Годин. – М. : Издательство Юрайт, 2019. – 291 с.
29. Стружкин, Н. П. Базы данных: проектирование. Учебник для академического бакалавриата / Н. П. Стружкин, В. В. Годин. – М. : Издательство Юрайт, 2019. – 477 с.
30. Сурядный, А. С. Microsoft Access 2010. Лучший самоучитель. – М. : Астрель, ВКТ, 2012. – 488 с.
31. Тарасов, С. В. СУБД для программиста. Базы данных изнутри / С. В. Тарасов. – Москва : СОЛОН-Пресс, 2015. – 320 с.
32. Тимошок, Т.В. Microsoft Office Access 2007: самоучитель / Т.В. Тимошок. – М. : Вильямс, 2008. – 464 с.
33. Туманов, В. Е. Основы проектирования реляционных баз данных / В. Е. Туманов. – Москва : Бином, 2012. – 420 с.
34. Федорова, Г. Разработка и администрирование баз данных / Г. Федорова. – Москва : Академия, 2015. – 313 с.
35. Харрингтон, Дж. Л. Проектирование реляционных баз данных. Пер. с англ. / Дж. Л. Харрингтон – М.: Лори, 2006. – 240 с.
36. Хомоненко, А. Д. Базы данных: Учеб. для вузов / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев; Под ред. А. Д. Хомоненко. – СПб.: КОРОНА-Век, 2009. – 736 с.
37. Наумов, А. Н. Системы управления базами данных и знаний / А. Н. Наумов, А. М. Вендров, В. К. Иванов. – Москва : Финансы и статистика, 2010. – 352 с.
38. Петров, В. Н. Информационные системы. СПб.: Питер, 2003. – 688 с.
39. Пирогов, В. Ю. Информационные системы и базы данных / В. Ю. Пирогов. – СПб. : БХВ-Петербург, 2009. – 528 с.
40. Роб, П., Коронел, К. Системы баз данных: проектирование, реализация и управление. – 5-е изд., перераб. и доп.: Пер. с англ. – СПб. : БХВ-Петербург, 2004. – 1040 с.
41. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – Москва : Юрайт, 2013. – 463 с.
42. Хендерсон, К. Профессиональное руководство по SQL Server: структура и реализация / К. Хендерсон. – пер. с англ. – Москва : Вильямс, 2006. – 184 с.

Рекомендации по организации работы с УМК

Использование разработанного ЭУМК предполагает работу студентов с конспектом лекций при подготовке к выполнению и защите лабораторных работ, подготовке к сдаче экзамена по дисциплине «Базы данных». Кроме того, теоретический материал полезен при выполнении научных исследований, при написании курсовых работ по специальным дисциплинам и выпускных дипломных проектов (работ).

ЭУМК направлен на повышение эффективности учебного процесса и организацию целостности системы учебно-предметной деятельности, что является одним из важнейших направлений стратегических инноваций в образовании.

Изучение дисциплины на основе ЭУМК, обеспечивает эффективную учебную деятельность, формирует профессиональные компетенции будущих специалистов, а именно:

- ✓ владеть основами исследовательской деятельности, осуществлять поиск, анализ и синтез информации;
- ✓ проектировать, создавать и администрировать информационные базы данных для информационного обеспечения программных комплексов и систем;
- ✓ работать в пакетах прикладных программ, автоматизирующих бизнес-процессы электронного бизнеса;
- ✓ использовать инструментальные средства информационных сред и автоматизированных систем.

ЭУМК способствует успешному усвоению студентами учебного материала, дает возможность планировать и осуществлять самостоятельную работу студентов, развивать способность к самоорганизации и самообразованию, обеспечивает рациональное распределение учебного времени по темам учебной дисциплины и совершенствование методики проведения занятий.

***Дисциплина*:**

«Базы данных»

Структура курса:

Лекции	16 часов
Лабораторные работы	34 часа
Самостоятельная работа	
без контроля преподавателя	58 часов
ИТОГО:	108 часов
Форма контроля	**Экзамен**

Глава 1.

Введение в теорию баз данных.

Понятия, лежащие в основе концепции баз данных и систем управления базами данных

§ 1.1. Основные понятия

Предметная область – часть реального мира, подлежащая изучению с целью организации управления и, в конечном счете, автоматизации.

Информация – сведения, обладающие структурой, смысловым наполнением в предметной области, по которым известны правила обработки.

Экономическая информация – совокупность данных (сведений) о четырех процессах экономики (производстве, распределении, перераспределении и потреблении), используемых при осуществлении функции управления экономическими объектами.

Виды экономической информации:

1) по функциям управления: учетная, плановая, директивная, статистическая;

2) по месту возникновения: внутренняя (полученная внутри экономического объекта), внешняя (поступающая из вышестоящих звеньев управления);

3) по стадиям образования: первичная (возникает из начальной стадии управления), вторичная (возникает в результате обработки первичной);

4) *по источнику поступления*: входная (поступает в фирму извне и используется как первичная информация), выходная (поступает из одной системы в другую);

5) по степени структурированности: неструктурированная, слабо структурированная, структурированная, формализовано структурированная, машинно-структурированная.

Требования к экономической информации:

1. **Корректность** – обеспечивает ее однозначное восприятие всеми потребителями.
2. **Полезность** – проявляется в том случае, если она способствует достижению стоящей перед потребителем цели.
3. **Оперативность** – отражает актуальность информации для необходимых расчетов и принятия решений в изменившихся условиях.
4. **Точность** – определяет допустимый уровень искажения как исходной, так и результативной информации, при котором сохраняется эффективность функционирования системы.
5. **Достоверность** – определяется свойством информации отражать реально существующие объекты с необходимой точностью.

6. **Устойчивость** – отражает способность реагировать на изменения исходных данных без нарушения необходимой точности. Устойчивость определяется выбранной методикой ее отбора и формирования.

7. **Достаточность** – она содержит минимально необходимый объем сведений для принятия правильного решения. Не полная информация снижает эффективность принятия решений. Избыточность обычно снижает оперативность и затрудняет принятие решения, но зато делает информацию более устойчивой.

8. **Своевременность подачи потребителя**м.

9. **Простота кодирования**.

10. **Минимизация расходов на формирование и обработку**.

Экономические показатели описывают разные сущности (*предметы, процессы, явления, объекты*) как простые, так и сложные.

Каждая сущность имеет определенные свойства (*вес, габариты, цена и пр.*).

Совокупность сведений, отражающих какую-либо сущность, называют **составной единицей информации**.

Например, единицей информации данных о поставщике являются данные:

- ✓ «ФИО»,
- ✓ «АДРЕС»,
- ✓ «НОМЕНКЛАТУРА ТОВАРОВ»,
- ✓ «УСЛОВИЯ ПОСТАВКИ»
- ✓ и т.д.

Структурные единицы экономической информации: *реквизиты, показатели, документы, массивы.*

Неделимая на смысловые единицы единица информации получила название **реквизит** (элемент данных, атрибут).

Реквизиты – это совокупность формальных элементов, выражающих определенные свойства объекта.

Реквизиты принято делить на ***реквизиты-признаки*** и ***реквизиты-основания***.

Реквизит-признак характеризует качественное свойство объекта (ФИО, время или место действия и т.д.).

Реквизит-основание – это количественная характеристика, выраженная в определенных единицах (объем продукции, цена и т.д.).

Каждый реквизит характеризуется своим *наименованием и значением*, которое, в свою очередь, имеет свой формат. ***Формат*** указывает тип и максимальную длину значения реквизита. Тип может быть числовым, символьным или логическим.

Сочетание одного реквизита-основания с несколькими реквизитами-признаками образуют **показатель**.

На основе показателей строятся документы.

Документом называется совокупность логически связанных реквизитов, имеющих юридическую силу.

Совокупность документов, объединенных по определенному признаку, называется **массивом**.

Система – это совокупность взаимосвязанных элементов, подчинённых единой цели.

Признаки системы:

1) составляющие элементы взаимосвязаны и взаимодействуют в рамках системы;

2) каждый элемент может рассматриваться как самостоятельная система, но он выполняет только часть функции системы;

3) совокупность элементов выполняет определённую функцию, которая не может быть сведена к функциям отдельно взятого элемента;

4) подсистемы могут взаимодействовать как между собой, так и с внешней средой и изменять при этом свое содержание или внутреннее строение.

Система управления – реализует функции управления и состоит из таких подсистем, как *прогнозирование, планирование, учёт, анализ, контроль и регулирование*.

Любой системе управления экономическим объектом соответствует ***экономическая информационная система (ЭИС),*** или совокупность внутренних и внешних потоков прямой и обратной информационной связи экономического объекта, методов, средств, специалистов, участвующих в процессе обработки информации и выработке управляющих решений.

Общая схема системы управления экономическим объектом:

Процессы, обеспечивающие работу ИС любого назначения, условно можно представить в виде схемы, состоящей из следующих блоков:

1. Ввод информации (сбор информации о состоянии внешней среды и объекта управления, т. е. создание первичной, или входной, информации и представление ее в нужном формате);
2. База данных (хранилище данных);
3. Обработка информации (поиск, фильтрация, сортировка, агрегирование, анализ, вывод информации для представления потребителям или передачи в другую систему);
4. Обратная связь (передача информации, переработанной потребителем для коррекции входной информации, т.е. выработка управляющих воздействий).

Важнейшими функциями ЭИС являются:

- ✓ учет,
- ✓ анализ,
- ✓ прогнозирование и планирование экономических процессов.

В экономике с учётом сферы применения выделяются:

- ✓ банковские информационные системы;
- ✓ информационные системы фондового рынка;
- ✓ страховые информационные системы;
- ✓ налоговые информационные системы;
- ✓ статистические информационные системы;
- ✓ бухгалтерские информационные системы;
- ✓ и т.д.

Информационное обеспечение (ИО) – совокупность данных объекта управления, языковых средств описания данных, программных средств обработки информационных массивов, методов организации, накопления, хранения и доступа к информационным массивам.

Важнейшими элементами ИО являются:

- ✓ системы показателей, описывающих деятельность экономического объекта;
- ✓ системы классификации и кодирования информации;
- ✓ информационная база;
- ✓ документация, содержащая показатели деятельности экономического объекта.

Информационная база (ИБ) включает потоки внутренней (функционирующей внутри экономического объекта) и внешней (возникающей вне данного экономического объекта) информации.

Программный компонент информационного обеспечения предназначен для ведения информационной базы системы, т.е. организации контроля взаимодействия данных, сохранение, накопление, внесение изменений и контроль за данными.

§ 1.2. Определение базы данных (БД) и системы управления базами данных (СУБД)

Данные – информация, представленная в виде, пригодном для обработки автоматическими средствами при возможном участии человека.

Обработка данных – специальный класс решаемых на ЭВМ задач, связанных с ***вводом, хранением, сортировкой, отбором и группировкой*** записей данных однородной структуры.

База данных (database) – это поименованная совокупность взаимосвязанных данных, организованных на машинном носителе средствами ***системы управления базами данных*** (СУБД), отображающая отношения и свойства объектов в некоторой предметной области.

Объектом может быть предмет, вещество, событие, лицо, явление, абстрактное понятие, т.е. все то, что характеризуется набором значений некоторой совокупности атрибутов – информационного отображения свойств объекта.

Например, объект «книга» характеризуется атрибутами: наименование, авторы, количество страниц, тираж, цена и т.п.

Преимущество использования базы данных:

- ✓ Независимость данных – сокращение размеров программной поддержки (внутри отдельных программ).
- ✓ Увеличение эффективности разработки приложений.
- ✓ Возможность создания и использования стандартов.
- ✓ Минимальная избыточность хранения данных.
- ✓ Увеличение плотности данных и совместного доступа к данным.
- ✓ Улучшенный доступ к данным и их соответствие конкретным решаемым задачам.
- ✓ Увеличение качества данных.
- ✓ Безопасность, сохранение и восстановление.

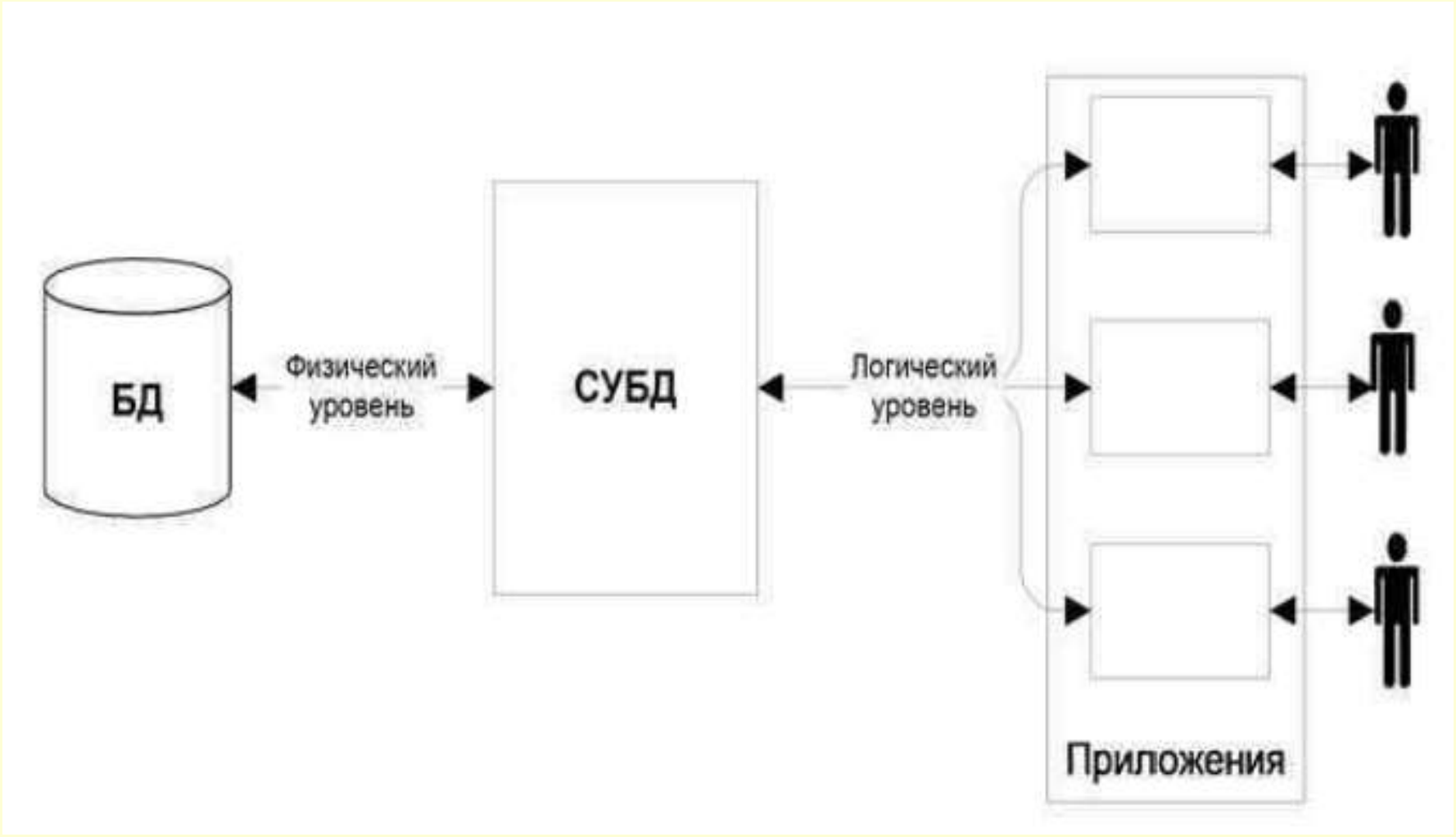
Области применения баз данных:

- ✓ Data Mining
- ✓ MRP – планирование потребностей в материалах;
- ✓ MRP II – планирование ресурсов производства;
- ✓ ERP – планирование ресурсов производства + управление финансами и кадрами предприятия;
- ✓ Системы принятия решений;
- ✓ Анализ данных;
- ✓ Организация хранилищ данных;
- ✓ Географические БД;
- ✓ Мультимедиа базы данных;
- ✓ Мобильные и персональные БД;
- ✓ Информационные системы;
- ✓ Распределенные информационные системы;
- ✓ Система World Wide Web.

Системы управления базами данных* (СУБД) – это управляющие программы, которые обеспечивают все манипуляции с базами данных: создание базы, ее ведение, ее использование многими пользователями и др., т. е. реализуют сложный комплекс функций по централизованному управлению базой данных и обслуживают интересы пользователей.

* database management system

Схема взаимодействия пользователей с базой данных:



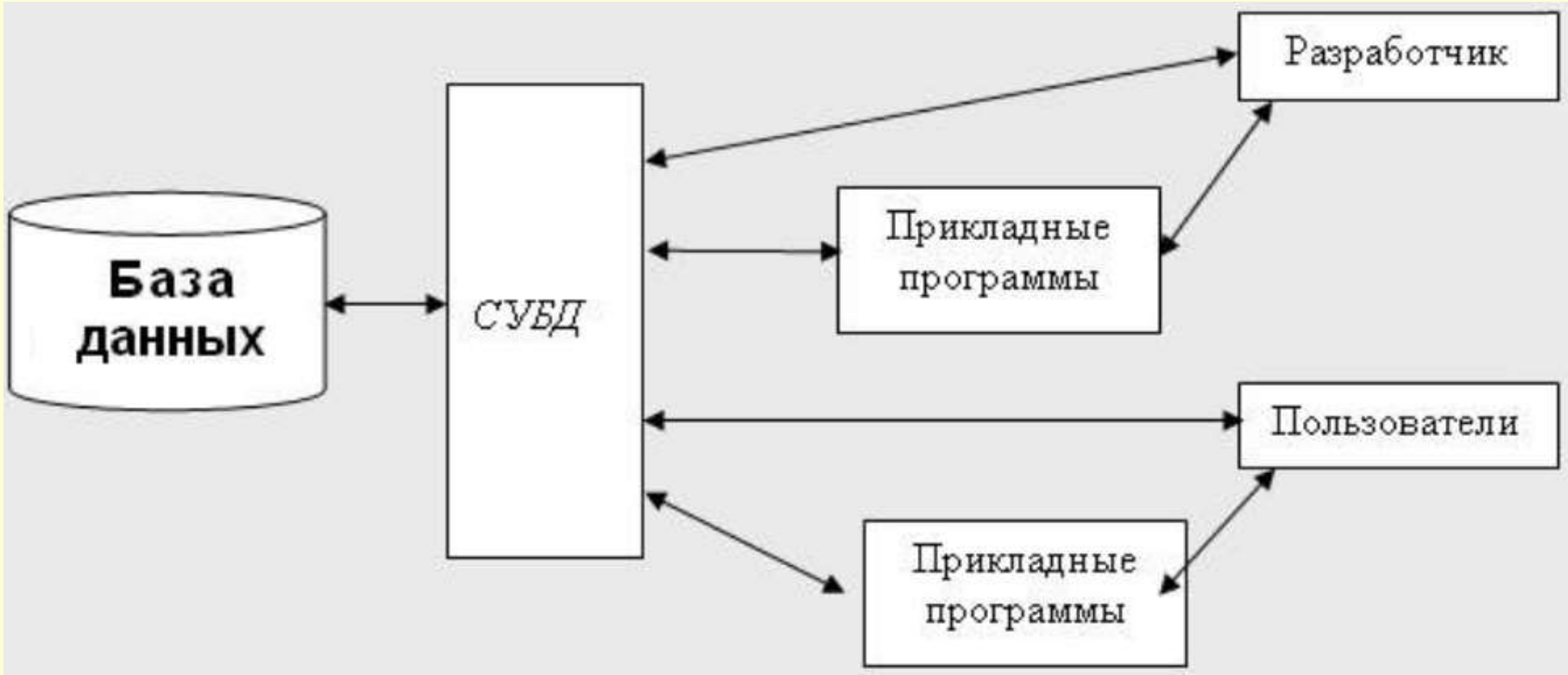
Понятия интегрированности и разделимости данных:

В общем случае данные в базе данных (по крайней мере, в больших системах) являются интегрированными и разделяемыми.

Интегрированность данных – это возможность представить базу данных как объединение нескольких отдельных файлов данных, полностью или частично исключающее избыточность хранения информации.

Разделяемость данных – возможность использования отдельных элементов, хранимых в БД несколькими различными пользователями. Имеется в виду, что каждый из пользователей сможет получить доступ к одному и тому же элементу данных, возможно, для достижения различных целей.

Основные компоненты системы базы данных:



Глава 2.

Жизненный цикл базы данных. Модели данных.

§ 2.1. Жизненный цикл БД

Жизненный цикл БД (ЖЦБД) – это процесс проектирования, реализации и поддержки БД.

ЖЦБД состоит из следующих этапов:

1. предварительное планирование
2. определение требований
3. проектирование БД (концептуальное, логическое, физическое)
4. разработка приложений
5. реализация
6. загрузка данных
7. тестирование
8. эксплуатация и сопровождение

Этапы проектирования БД

1. ***Концептуальное*** проектирование - анализ описания предметной области на естественном языке - построение концептуальной модели.

2. ***Логическое*** проектирование - выбор модели БД (сетевая, реляционная) - преобразование концептуальной модели в логическую.

3. ***Физическое*** проектирование - выбор конкретной СУБД (Oracle, MySQL) - определение физических структур хранения - разработка средств защиты БД (роли).

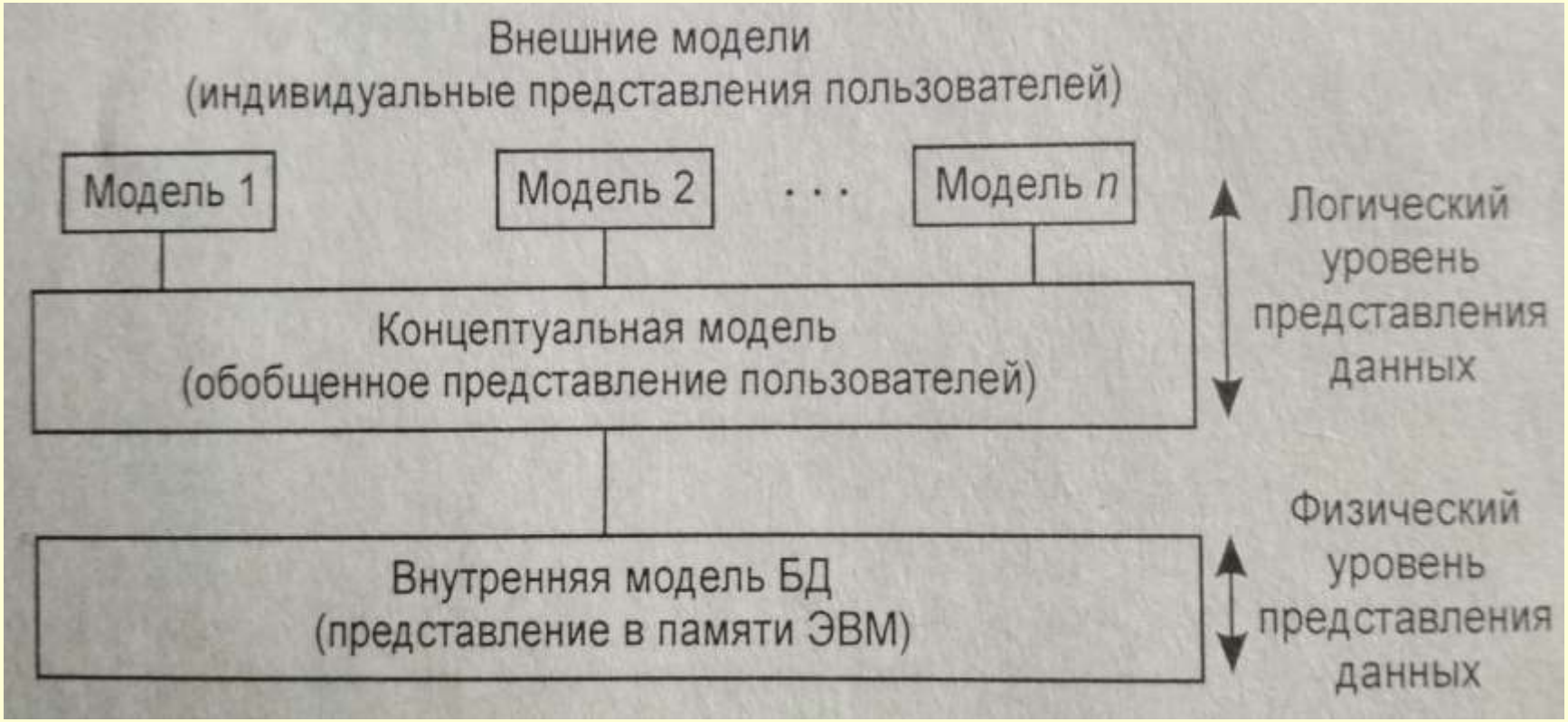
§ 2.2. Модели данных

БД организуются на основе различных ***моделей данных.***

Модель данных – это совокупность допустимых структур данных и операций над ними, поддерживаемая компьютерной средой (в т. ч. СУБД) для определения логической структуры базы данных и динамического моделирования состояний предметной области.

Информационно-логическая модель (ИЛМ) – совокупность информационных объектов (сущностей) предметной области и связей между ними.

Многоуровневое представление данных в БД:



Внешняя модель пользователя – это отображение концептуальных требований этого пользователя в логической модели (реализуется через организацию ролей БД, где определяются возможности выполнения операций, доступности взаимодействия с таблицами и пр).

Концептуальная модель (КМ) – представляет интегрированные концептуальные требования всех пользователей к базе данных данной предметной области. (т.е. отображает предметную область в виде взаимосвязанных объектов без указания способов их физического хранения).

Логическая модель (ЛМ) – это версия КМ, которая может быть реализована конкретной СУБД.

ЛМ отображает логические связи между информационными данными в данной концептуальной модели и может быть ***реляционной; иерархической; сетевой***.

Физическая (внутренняя) модель (ФМ) – определяет размещение данных, методы доступа и технику индексирования в данной логической модели.

Проектирование БД

Процесс проектирования базы данных состоит из следующих этапов:

1) *Сбор информации;*
2) *Идентификация объектов;*
3) *Моделирование объектов;*
4) *Идентификация типов информации для каждого объекта;*
5) *Идентификация отношений;*
6) *Нормализация;*
7) *Преобразование к физической модели;*
8) *Создание базы данных.*

Пп. 1-6 – это логическое моделирование,

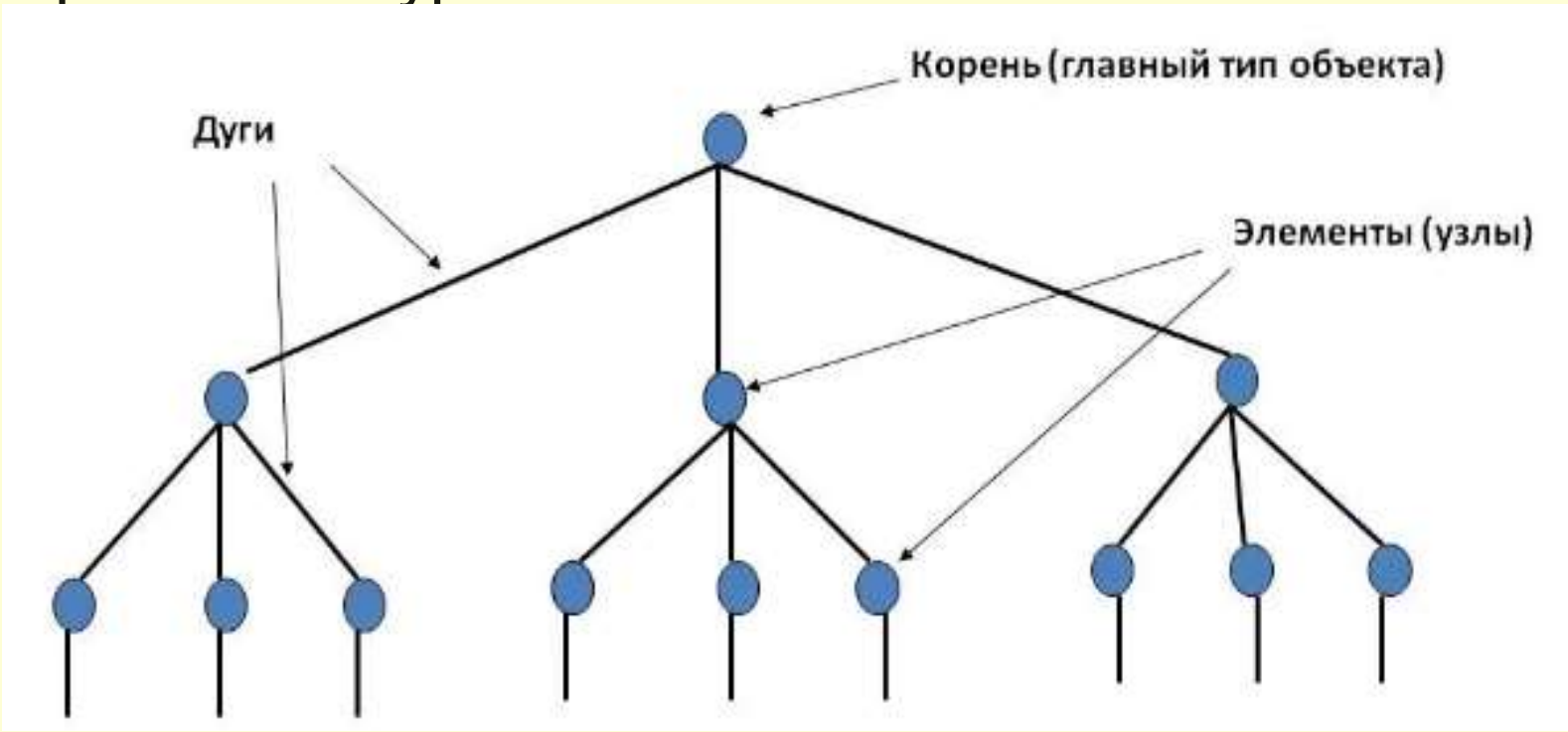
Пп. 6-7 – физическое моделирование БД.

Эволюция моделей БД

Электронные картотеки (до 1965 г.)	✓ Системы, основанные на файлах
Первые модели БД (1965 – 1975 гг.)	✓ Сетевая модель ✓ Иерархическая модель
Реляционная модель (с 1970 г. по наши дни)	✓ Реляционная модель ✓ Концептуальная модель «сущность – связь»
Слабоструктурированные данные (с 1998 г. по наши дни)	✓ Язык XML
Документ-ориентированная модель (с 2008 г. по наши дни)	✓ Системы NoSQL

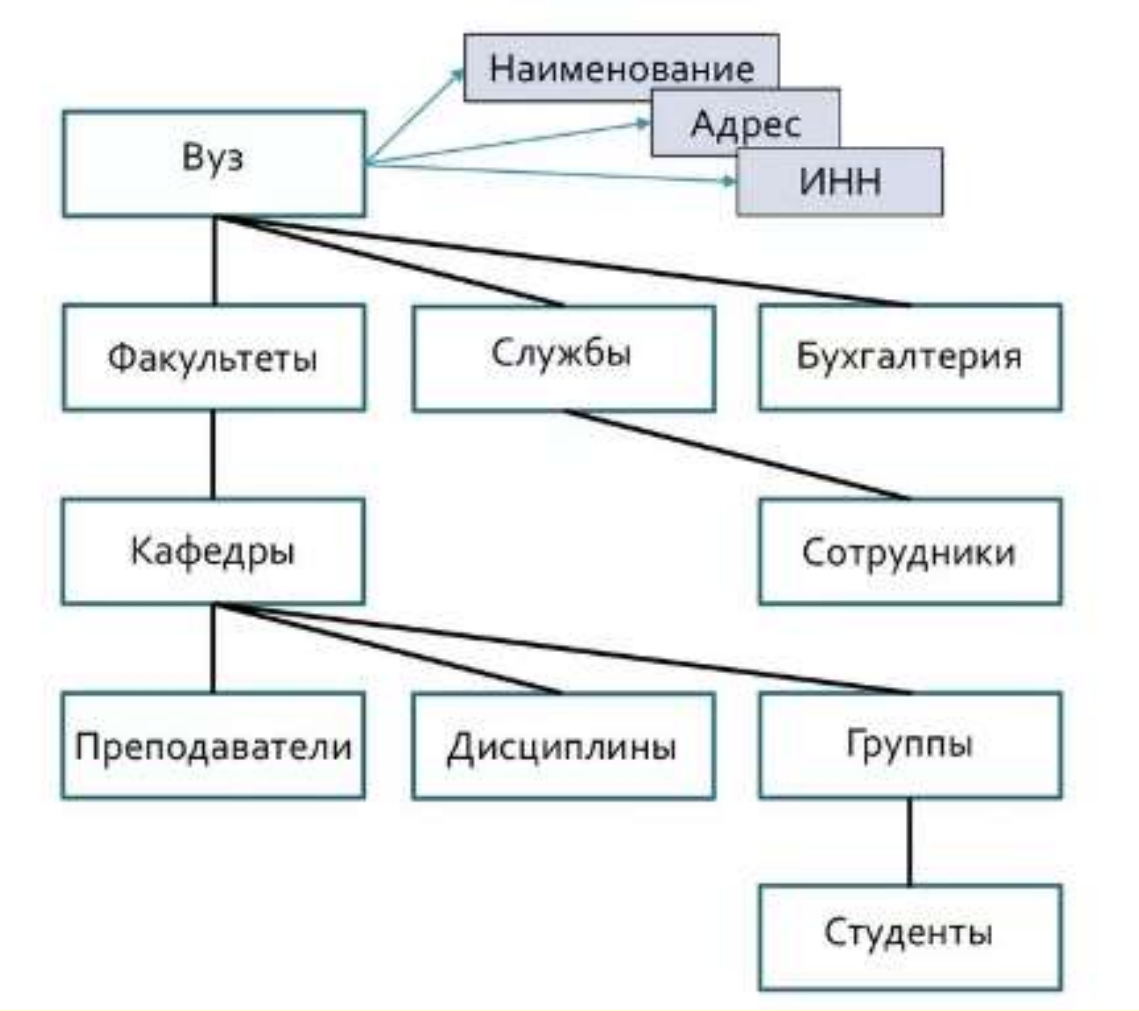
Типы моделей данных

I. Иерархическая модель (ИМ) – модель, в которой взаимосвязи между объектами отражаются по принципу иерархии типов объекта в виде связанного графа, вершины которого размещены на разных иерархических уровнях.

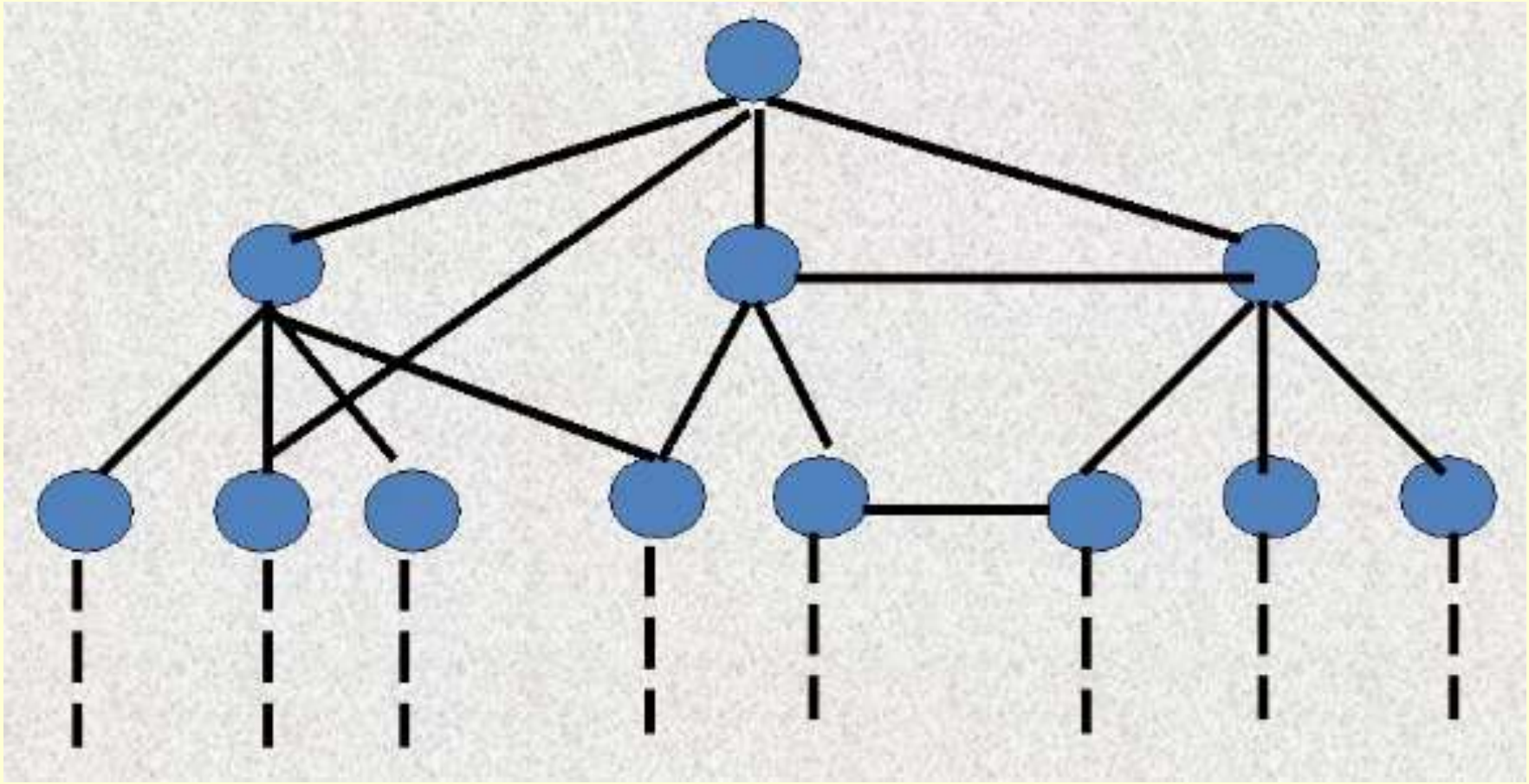


В данной модели любой объект может подчиняться только одному объекту вышестоящего уровня.

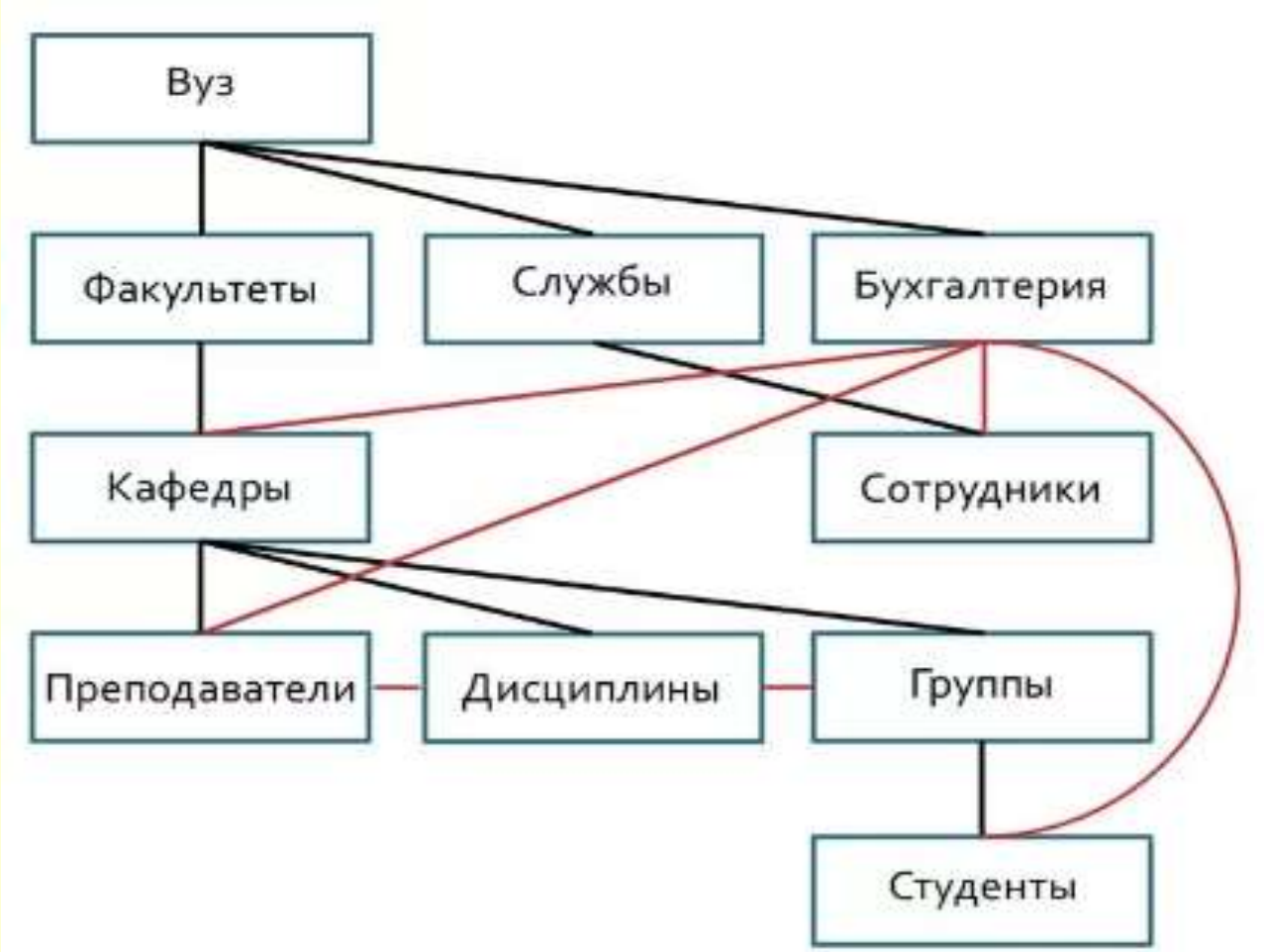
Т.о. один объект главный, остальные – подчиненные.



II. Сетевая модель – модель, в которой любой объект может быть подчинен нескольким объектам, т.е. может быть и главным и подчиненным, и может участвовать в любом количестве взаимосвязей.



Сетевая БД состоит из наборов записей, которые связаны между собой так, что записи могут содержать явные ссылки на другие наборы записей. Тем самым наборы записей образуют сеть.



III. Реляционная модель – модель, в которой информация представляется в виде прямоугольных таблиц.

Каждая таблица отражает объект реального мира (**сущность**), состоит из **строк** и **столбцов** и имеет имя, уникальное внутри БД.

Каждая строка (**запись**) такой таблицы содержит информацию, относящуюся только к одному конкретному объекту (экземпляр сущности), а каждый столбец (**поле**) таблицы имеет уникальное для своей таблицы имя и содержит одну характеристику (признак, параметр) этого объекта.

В реляционной модели таблицы могут быть связаны между собой посредством значений одного или нескольких совпадающих полей (ключей).

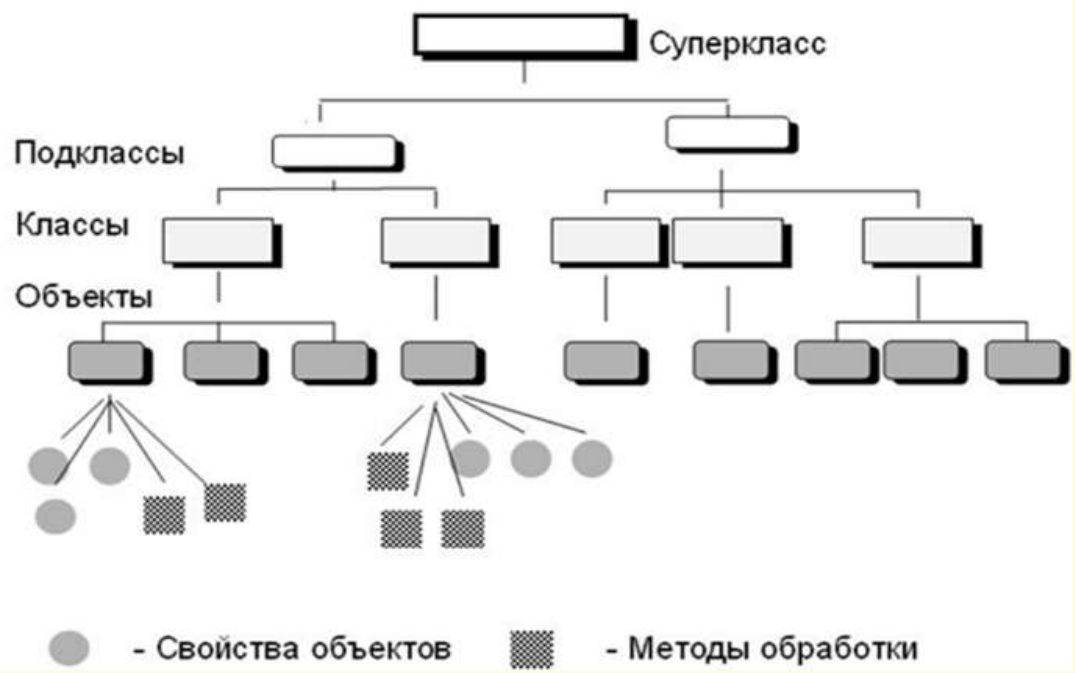
Реляционная БД (РБД) – это совокупность простейших двумерных логически взаимосвязанных таблиц-отношений, состоящих из множества полей и записей, отражающих некоторую предметную область.

№ зач. кн.	ФИО	Группа
125501	Иванов Иван Иванович	ЭАС-503
125704	Максимов Максим Максимович	ЭАС-502
125745	Петров Петр Петрович	ЭАС-502
125451	Сергеев Сергей Сергеевич	ЭАС-503

Реляционная система управления базами данных (РСУБД) – набор средств для управления РБД, содержащий утилиты, приложения, службы, библиотеки, средства создания приложений и другие компоненты.

Будучи связанной посредством общих ключевых полей, информация в РБД может объединяться из множества таблиц в единый результирующий набор.

IV. Объектно-ориентированная модель представляет структуру, которую можно изобразить графически в виде дерева, узлами которого являются **объекты**. Между записями БД и функциями их обработки устанавливаются связи с помощью механизмов, подобных тем, которые имеются в объектно-ориентированных языках программирования.



Глава 3.
Концептуальное проектирование базы данных. Операции реляционной алгебры. Типы связей между сущностями.

§ 3.1. Концептуальное проектирование БД

Определение сущности

Построение реляционной модели данных (РМД) основывается на том понимании, что любой набор данных может быть представлен в виде **отношения**, оформляемого по форме таблицы, где данные представляются атрибутами и значениями на пересечении соответствующего атрибута с записью (кортежем).

Отношение – это множество данных, объединённых в совокупность записей (кортежей) и описанных заголовком, содержащим множество атрибутов.

В моделировании данных отношение наполняется содержательным смыслом и применяется для определения объектов реальной действительности и связей между ними.

Наглядным образом отношения является реляционная таблица. Отношение, подобно таблице, содержит заголовок и тело. Заголовок отношения представлен конечным множеством атрибутов.

Атрибуты определены на соответствующих доменах и в заголовке представлены своими содержательными именами.

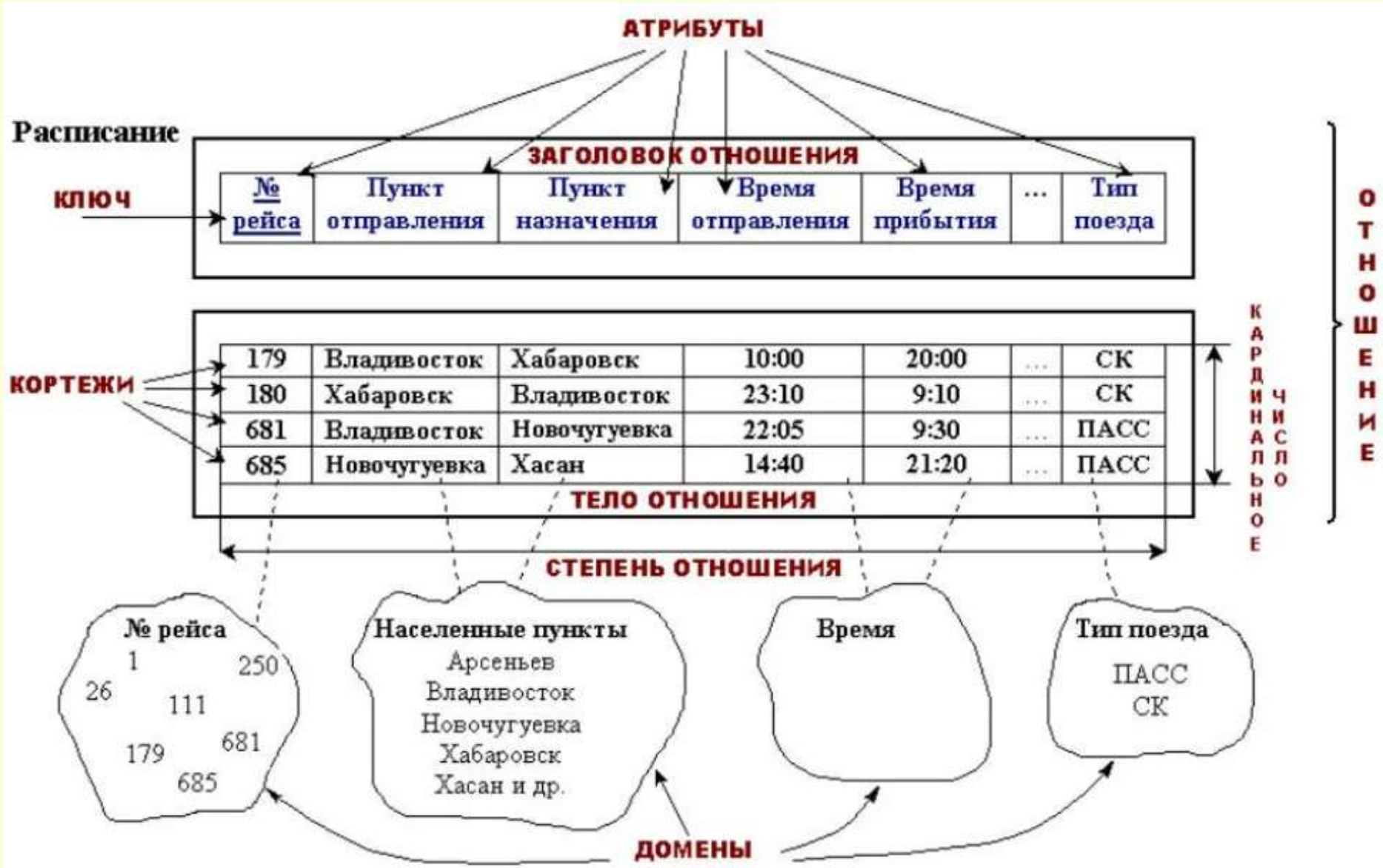
Домен – это допустимое множество поименованных значений одного типа, имеющих определённый смысл.

Тело отношения содержит множество кортежей-строк.

Кортеж – это подмножество взаимосвязанных данных, определяемых доменами.

Максимальное число строк-кортежей называется кардинальным числом (**мощностью**) отношения.

Число столбцов-атрибутов называется **степенью** отношения.


АТРИБУТЫ
Расписание
ЗАГОЛОВОК ОТНОШЕНИЯ
КЛЮЧ
№ рейса
Пункт отправления
Пункт назначения
Время отправления
Время прибытия
...
Тип поезда
ОТНОШЕНИЕ
КОРТЕЖИ
179 Владивосток Хабаровск 10:00 20:00 ... СК
180 Хабаровск Владивосток 23:10 9:10 ... СК
681 Владивосток Новочугуевка 22:05 9:30 ... ПАСС
685 Новочугуевка Хасан 14:40 21:20 ... ПАСС
КАРДИНАЛЬНОЕ ЧИСЛО
ТЕЛО ОТНОШЕНИЯ
СТЕПЕНЬ ОТНОШЕНИЯ
№ рейса
1
250
26
111
179
681
685
Населенные пункты
Арсеньев
Владивосток
Новочугуевка
Хабаровск
Хасан и др.
Время
Тип поезда
ПАСС
СК
ДОМЕНЫ

Свойства сущности:

1. В отношении нет одинаковых кортежей.
2. Кортежи не являются упорядоченными по какому-либо правилу (сверху вниз).
3. Атрибуты не являются упорядоченными по какому-либо правилу (слева направо).
4. Все значения атрибутов атомарны (каждый кортеж содержит только одно значение, соответствующего типа по каждому атрибуту).

Определение ключевого поля таблицы

Ключом называется набор атрибутов, однозначно определяющий запись.

Другими словами, ***ключевое поле (или поля)*** – это такое поле (или набор полей), значения которого (или комбинация значений которых) для каждой записи таблицы не повторяется, т.е. делает запись уникальной.

Ключи делятся на два класса: простые и составные.

Простой ключ состоит из одного атрибута (*например: Табельный номер в таблице (списке) сотрудников; Инвентарный номер в таблице (списке) основных средств и т.д.*).

Составной ключ состоит из нескольких атрибутов (*например: НомерНакладной+ДатаНакладной+НомерТовара в таблице учета реализации товаров*).

Применение составных ключей усложняет объединение таблиц.

Существуют следующие **виды** ключей:

Первичный ключ – это одно или несколько полей (столбцов), значения которых однозначно определяют каждую запись в таблице.

Значение первичного ключа в таблице БД должно быть уникальным, то есть в таблице не должно существовать двух или более записей с одинаковым значением первичного ключа. Первичный ключ не допускает значение Null и всегда должен иметь уникальный индекс. Первичный ключ используется для связывания таблицы с внешними ключами в других таблицах.

Первичный ключ может быть:

естественным, который состоит из информационных полей таблицы, (т. е. полей, содержащих полезную информацию об описываемых объектах) .

искусственным – это дополнительное служебное поле, назначение которого – служить первичным ключом. Значения этого поля генерируются искусственно в виде числовой последовательности (*например, создаются поля с типом данных* ***Счетчик***).

§ 3.2. Операции реляционной алгебры

Под **реляционной алгеброй** принято понимать комплекс операций, которые в качестве основных операндов и возвращаемого результата используют отношения.

Традиционно, определяют ***восемь*** реляционных операторов, объединенных в две группы.

Теоретико-множественные операторы:

объединение, пересечение, вычитание, декартово произведение.

Специальные реляционные операторы:

выборка, проекция, соединение, деление.

Объединение

Объединением двух совместимых по типу отношений и называется отношение с тем же заголовком, что и у отношений **A** и **B**, и телом, состоящим из кортежей, принадлежащих или **A**, или **B**, или обоим отношениям.

Синтаксис операции объединения: **A** UNION **B**

Замечание. Объединение, как и любое отношение, не может содержать одинаковых кортежей.

Поэтому, если некоторый кортеж входит и в отношение **A**, и отношение **B**, то в объединение он входит один раз.

Пример1. Пусть даны два отношения **A** и **B** с информацией о сотрудниках:

Табельный номер	Фамилия	Зарплата
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Таблица 1 Отношение A

Табельный номер	Фамилия	Зарплата
1	Иванов	1000
2	Пушников	2500
4	Сидоров	3000

Таблица 2 Отношение B

Табельный номер	Фамилия	Зарплата
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000
2	Пушников	2500
4	Сидоров	3000

Таблица 3 Отношение A UNION B

Пересечение

Пересечением двух совместимых по типу отношений **A** и **B** называется отношение с тем же заголовком, что и у отношений **A** и **B**, и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям **A** и **B**.

Синтаксис операции объединения: **A** INTERSECT **B**

Замечание. Никакие реляционные операторы не передают результатирующему отношению никаких данных о потенциальных ключах.

Пример2. Пусть даны два отношения **A** и **B** с информацией о сотрудниках:

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Таблица 1 Отношение A

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000
2	Пушников	2500
4	Сидоров	3000

Таблица 2 Отношение B

Табельный номер	**Фамилия**	**Зарплата**
1	Иванов	1000

Таблица 4 Отношение A INTERSECT B

Вычитание

Вычитанием двух совместимых по типу отношений **A** и **B** называется отношение с тем же заголовком, что и у отношений **A** и **B** и телом, состоящим из кортежей, принадлежащих отношению **A** и не принадлежащих отношению **B**.

Синтаксис операции объединения: **A** MINUS **B**

Пример3. Пусть даны два отношения **A** и **B** с информацией о сотрудниках:

Табельный номер	Фамилия	Зарплата
1	Иванов	1000
2	Петров	2000
3	Сидоров	3000

Таблица 1 Отношение A

Табельный номер	Фамилия	Зарплата
1	Иванов	1000
2	Пушников	2500
4	Сидоров	3000

Таблица 2 Отношение B

Табельный номер	Фамилия	Зарплата
2	Петров	2000
3	Сидоров	3000

Таблица 5 Отношение A MINUS B

Декартово произведение

Декартовым произведением двух отношений **A(A_1,A_2,…,A_n)** и **B(B_1,B_2,…,B_m)** называется отношение, заголовок которого является сцеплением заголовков отношений **A** и **B**:

(A_1,A_2,…,A_n, B_1,B_2,…,B_m),

а тело состоит из кортежей, являющихся сцеплением кортежей отношений **A** и **B**:

(a_1,a_2,…,a_n, b_1,b_2,…,b_m),

таких, что **(a_1,a_2,…,a_n)** ∈ **A**, **(b_1,b_2,…,b_m)** ∈ **B**

Синтаксис операции объединения: **A** TIMES **B**

Замечания.

1. Мощность произведения **A** TIMES **B** равна произведению мощностей отношений **A** и **B**, т.к. каждый кортеж отношения **A** соединяется с каждым кортежем отношения **B**.

2. Если в отношения **A** и **B** имеются атрибуты с одинаковыми наименованиями, то перед выполнением операции декартового произведения такие атрибуты необходимо переименовать.

3. Перемножать можно любые два отношения, совместимость по типу при этом не требуется.

Пример4. Пусть даны два отношения **A** и **B** с информацией о поставщиках и деталях:

Номер поставщика	Наименование поставщика
1	Иванов
2	Петров
3	Сидоров

Таблица 6 Отношение A (Поставщики)

Номер детали	Наименование детали
1	Болт
2	Гайка
3	Винт

Таблица 7 Отношение B (Детали)

Номер поставщика	Наименование поставщика	Номер детали	Наименование детали
1	Иванов	1	Болт
1	Иванов	2	Гайка
1	Иванов	3	Винт
2	Петров	1	Болт
2	Петров	2	Гайка
2	Петров	3	Винт
3	Сидоров	1	Болт
3	Сидоров	2	Гайка
3	Сидоров	3	Винт

Таблица 8 Отношение A TIMES B

§ 3.3. Типы связей между сущностями

Связи между таблицами устанавливаются по равенству значений ***первичного*** и ***внешнего*** ключей.

Связанные отношениями таблицы взаимодействуют по принципу «главная-подчиненная».

Таблица, в которой используется для связи ***первичный ключ***, называется **главной**, а таблица ***с внешним ключом*** – **подчиненной**.

Одна и та же таблица может быть главной по отношению к одной таблице БД и подчиненной (дочерней) по отношению к другой.

Внешний ключ – представляет собой одно или несколько полей (столбцов), содержащих ссылку на поле или поля первичного ключа в другой таблице.
!!! Внешний ключ определяет способ связи таблиц.

Типы отношений между таблицами

Существует три типа отношений между таблицами:

Один-к-одному (1:1). Значению ключа в каждой записи в главной таблице могут соответствовать значения в связанном поле только в одной записи подчиненной таблицы.

!!! В этом случае связь между таблицами может быть установлена только через ключевые поля обеих таблиц.

Один-ко-многим (1:М). Значению ключа в каждой записи в главной таблице могут соответствовать значения в связанном поле (полях) в нескольких записях подчиненной таблицы.

Много-ко-многим (М:М). Возникает между двумя таблицами, когда одна запись с первой таблицы А (выходная связь) может быть связана больше чем с одной записью другой таблицы В (принимающая), в свою очередь, одна запись с другой таблицы может быть связана больше чем с одной записью первой таблицы. Эта схема реализуется только при помощи третьей соединительной таблицы, ключ связи которой состоит, как минимум, из двух полей. Эти поля являются полями внешнего ключа в таблицах А и В. Первичный ключ для соединительной таблицы – это обычно комбинация из внешних ключей.

Если между таблицами имеются связи типа M:M, создается дополнительная таблица пересечений, с помощью которой связь M:M будет сведена к двум связям типа 1:M.

Глава 4.

Логическое проектирование базы данных.

Нормализация данных.

Целостность данных

§ 4.1. Логическое проектирование БД

Последовательность построения логической модели данных:

Шаг 1. Создать отношения для логической модели данных.

Шаг 2. Проверить отношения с помощью правил нормализации.

Шаг 3. Проверить соответствие отношений требованиям пользовательских транзакций.

Шаг 4. Проверить ограничения целостности данных.

Шаг 5. Обсудить разработанную логическую модель данных с пользователями.

Шаг 6. Проверить возможность расширения модели в будущем.

§ 4.2. Нормализация данных

Нормализация представляет собой процесс реорганизации данных путем ликвидации повторяющихся групп и иных противоречий с целью приведения таблиц к виду, позволяющему осуществлять непротиворечивое и корректное редактирование данных.

Окончательная цель нормализации сводится к получению такого проекта БД, в котором каждый факт появляется только в одном месте, т.е. исключена избыточность информации.

1. Каждое поле любой таблицы должно быть уникальным.

2. Каждая таблица должна иметь уникальный идентификатор (первичный ключ), который может состоять из одного или нескольких полей таблицы.

3. Для каждого значения первичного ключа должно быть одно и только одно значение любого из столбцов данных, и это значение должно относиться к объекту таблицы (т.е. в таблице не должно быть данных, которые не относятся к объекту, определяемому первичным ключом, а также информация в таблице должна полностью описывать объект).

4. Должна иметься возможность изменять значения любого поля (не входящего в первичный ключ), и это не должно повлечь за собой изменения другого поля (т.е. не должно быть вычисляемых полей).

1NF. Отношение находится в первой нормальной форме (1NF), если на пересечении каждого столбца и строки находятся только элементарные (атомарные, неделимые) значения атрибутов.

Степень неделимости (атомарности), т. е. решение о том, следует разбивать неатомарный атрибут на атомарные или оставить его псевдоатомарным, определяется проектировщиком БД исходя из конкретных условий.

Если при обработке таблиц нет необходимости различать атомарные составляющие псевдоатомарного атрибута, то его можно не делить (например, атрибуты «Фамилия, имя, отчество», «Адрес» и т. д.).

2NF. Отношение находится во второй нормальной форме (2NF), если оно находится в 1NF, и каждый неключевой атрибут характеризуется полной функциональной зависимостью от первичного ключа.

Полная функциональная зависимость определяется следующим образом. В некотором отношении атрибут **B** полностью зависит от атрибута **A**, если атрибут **B** функционально зависит от полного значения атрибута **A** и не зависит от какого-либо подмножества полного значения атрибута **A**.

Например, таблица «Оценки по экзаменам» характеризуется следующим набором атрибутов {Номер зачетной книжки, Дисциплина, Дата сдачи, ФИО студента, № группы, Оценка}.

Очевидно, что первичным ключом является набор {Номер зачетной книжки, Дисциплина, Дата сдачи}. Полной функциональной зависимостью обладает только один неключевой атрибут «Оценка».

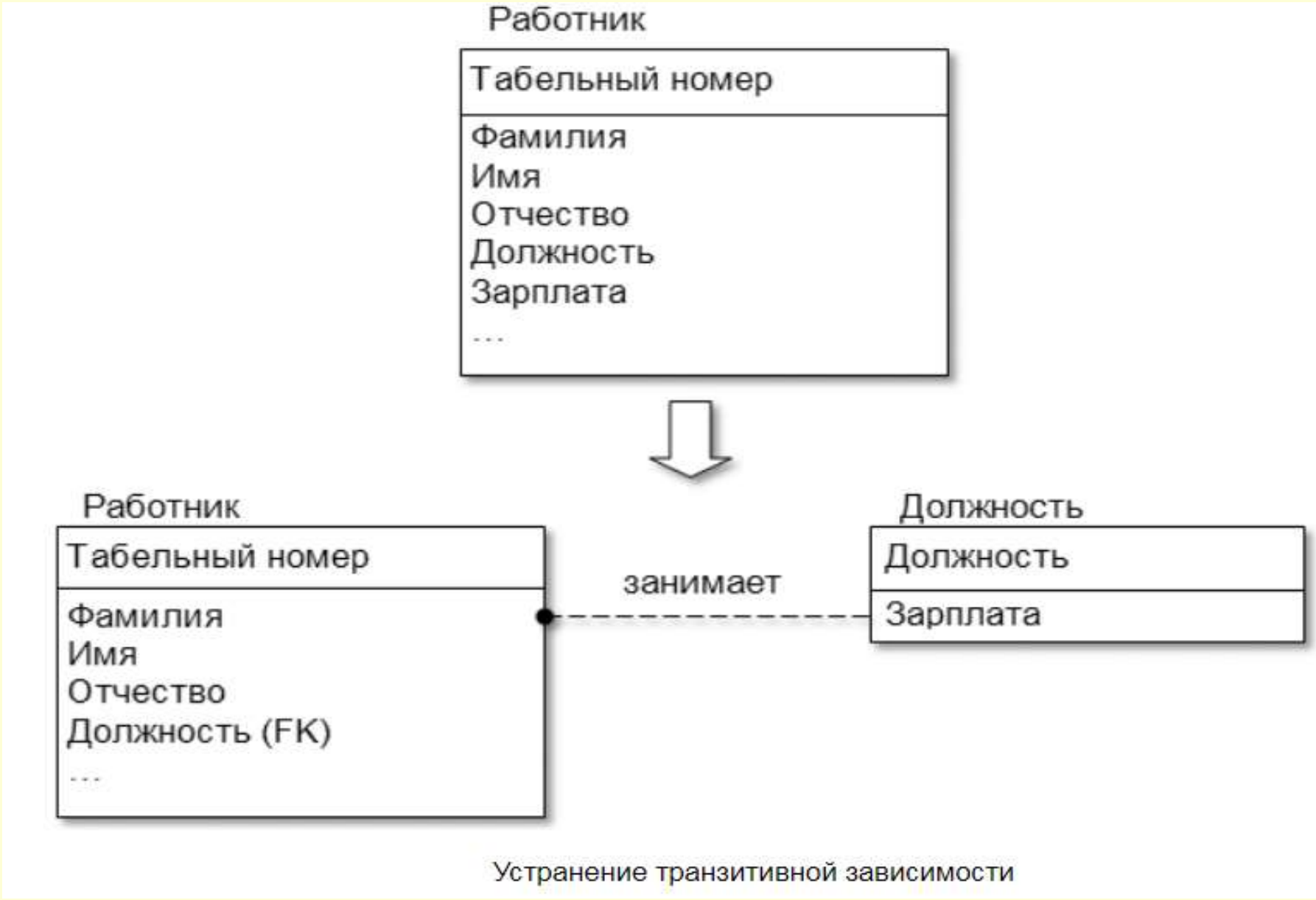
Атрибуты «ФИО студента» и «№ группы» могут быть однозначно определены по части первичного ключа – «Номер зачетной книжки».

Таким образом, требуется разбиение исходной таблицы на две.

<u>3NF.</u> Отношение находится в третьей нормальной форме (3NF), если оно находится во 2NF и никакой неключевой атрибут функционально не зависит от другого неключевого атрибута, т.е. нет транзитивных зависимостей.

Транзитивная зависимость. Если для атрибутов **A, B** и **C** некоторого отношения существуют зависимости вида **A** → **B** и **B** → **C**, то атрибут **C** транзитивно зависит от атрибута **A** через атрибут **B**.

Например, таблица «Работник» характеризуется набором атрибутов {Табельный номер, Фамилия, Имя, Отчество, Должность, Зарплата, …}, первичный ключ – {Табельный номер}. В этой таблице от первичного ключа («Табельный номер») зависит неключевой атрибут «Должность», а от «Должности» другой неключевой атрибут «Зарплата». Для приведения к 3NF необходимо добавить новую таблицу.



Устранение транзитивной зависимости

§ 4.3. Целостность данных

В БД, построенной на реляционной модели, задается ряд правил целостности, которые, по сути, являются ограничениями для всех допустимых состояний БД и гарантируют корректность данных.

Целостность данных – соответствие значений всех данных БД определенному непротиворечивому набору данных.

Можно выделить следующие типы ограничений целостности данных: ***обязательные данные; ограничения для доменов полей; целостность сущностей; ссылочная целостность***.

1. Обязательные данные. Некоторые поля всегда должны содержать одно из допустимых значений, другими словами, эти поля не могут иметь пустого значения.

2. Ограничения для доменов полей. Каждое поле имеет свой домен, представляющий собой набор его допустимых значений.

3. Целостность сущностей. Это ограничение целостности касается ***первичных ключей*** базовых таблиц. По определению, первичный ключ – минимальный идентификатор (одно или несколько полей), который используется для уникальной идентификации записей в таблице. Таким образом, никакое подмножество первичного ключа не может быть достаточным для уникальной идентификации записей.

Целостность сущностей определяет, что в базовой таблице ни одно поле первичного ключа не может содержать отсутствующих значений, обозначенных NULL.

Если допустить присутствие определителя NULL в любой части первичного ключа, это равносильно утверждению, что не все его поля необходимы для уникальной идентификации записей, и противоречит определению первичного ключа.

4. Ссылочная целостность. Указанное ограничение целостности касается внешних ключей. ***Внешний ключ*** – это поле (или множество полей) одной таблицы, являющееся ключом другой (или той же самой) таблицы. Внешние ключи используются для установления логических связей между таблицами. Связь устанавливается путем присвоения значений внешнего ключа одной таблицы значениям ключа другой.

Ссылочная целостность определяет: если в таблице существует внешний ключ, то его значение должно либо соответствовать значению первичного ключа некоторой записи в базовой таблице, либо задаваться определителем NULL.

Существует несколько важных моментов, связанных с внешними ключами.

Во-первых, следует проанализировать, допустимо ли использование во внешних ключах пустых значений.

В общем случае, если участие дочерней таблицы в связи является обязательным, то рекомендуется запрещать применение пустых значений в соответствующем внешнем ключе.

В то же время, если имеет место частичное участие дочерней таблицы в связи, то помещение пустых значений в поле внешнего ключа должно быть разрешено.

Ссылочная целостность может нарушиться в результате операций с кортежами отношений. Таких операций три: ***вставка, обновление и удаление кортежей в отношениях.***

Рассмотрим эти операции для родительского отношения (РО) и дочернего отношения (ДО).

При операциях с кортежами **родительского** отношения возможны следующие ситуации:

1. ***Вставка кортежа в родительское отношение***. Так как допустимо существование кортежей РО, на которые нет ссылок из ДО, то вставка кортежа в РО не нарушает ссылочной целостности.

2. ***Обновление кортежа родительского отношения***. При обновлении кортежа РО может измениться значение ключа. Если есть экземпляры ДО, ссылающиеся на обновляемый кортеж РО, то значения их внешних ключей станут некорректными. Обновление кортежа в РО может привести к нарушению ссылочной целостности, если это обновление затрагивает значение ключа.

3. ***Удаление кортежа родительского отношения***. При удалении кортежа РО удаляется значение ключа. Если есть кортежи ДО, ссылающиеся на удаляемый кортеж РО, то значения их внешних ключей станут некорректными. Удаление кортежа РО может привести к нарушению ссылочной целостности.

При операциях с кортежами **дочернего** отношения возможны следующие ситуации:

1. ***Вставка кортежа дочернего отношения*** может привести к нарушению ссылочной целостности, если вставляемое значение внешнего ключа некорректно.

2. ***Обновление кортежа дочернего отношения*** может привести к нарушению ссылочной целостности при некорректном изменении значения внешнего ключа.

3. При ***удалении кортежа дочернего отношения*** ссылочная целостность не нарушается.

Таким образом, ссылочная целостность в принципе может быть нарушена при выполнении одной из четырех операций:

1) обновление кортежа в родительском отношении;

2) удаление кортежа в родительском отношении;

3) вставка кортежа в дочернее отношение;

4) обновление кортежа в дочернем отношении.

Существуют две основные стратегии поддержания ссылочной целостности:

1) **RESTRICT** (ОГРАНИЧИТЬ) – не разрешать выполнение операции, приводящей к нарушению ссылочной целостности. Это самая простая стратегия, требующая только проверки, имеются ли кортежи дочернего отношения, связанные с некоторыми кортежами родительского отношения.

2) **CASCADE** (КАСКАД) – разрешить выполнение требуемой операции, но внести при этом необходимые поправки в других кортежах отношений так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи. Изменение начинается в родительском отношении и каскадно выполняется в дочернем отношении. Так как дочернее отношение может быть родительским для некоторого третьего отношения, то может потребоваться выполнение каскадной стратегии и для этой связи и т.д. Это самая сложная стратегия, но она хороша тем, что при этом не нарушается связь между кортежами родительского и дочернего отношений.

РЕЗЮМЕ:

Целостность данных – это набор правил, которые поддерживают корректность связей между записями в связанных таблицах и обеспечивают защиту данных от случайных изменений или удалений:

1. В подчиненной таблице нельзя вводить записи, которые не связаны с записью главной таблицы.
2. В главной таблице нельзя изменять значение ключевого поля, если в подчиненной таблице существуют записи, которые с ней связаны.
3. В главной таблице нельзя удалять записи, если в подчиненной таблице существуют связанные с ней записи.

Глава 5.

Назначение и функции СУБД. Физическая организация баз данных

Физическое моделирование данных – это заключительная стадия разработки при проектировании БД.

Критерии выбора физической организации данных:

1. Физические модели баз данных определяют способы размещения этих данных в среде хранения и способы доступа к этим данным, поддерживаемым на физическом уровне.

2. Физическая модель отражает все свойства (атрибуты) информационных объектов базы и связи между ними с учетом способа их хранения используемой СУБД.

При физической организации БД мы имеем дело не с представлением данных в прикладных программах, а с их размещением на запоминающих устройствах.

3. Критерии, определяющие выбор физической организации, отличаются от тех, которые определяют выбор логической организации данных. При выборе физической организации решающим фактором является эффективность, причем на первом месте стоит обеспечение эффективности поиска, далее идут эффективность операций занесения и удаления и затем обеспечение компактности данных.

4. От физического размещения данных в памяти ЭВМ существенно зависит время решения прикладных задач.

Наиболее распространенным критерием способов физической организации данных в различных СУБД служит время доступа к данным, однако в качестве критерия может выбираться, например, трудоемкость реализации соответствующих методов.

Физические модели данных служат для отображения моделей данных.

Основными понятиями модели данных являются поле, логическая запись, логический файл. Слово "логический" введено, чтобы отличать понятия, относящиеся к логической модели данных, от понятий, относящихся к физической модели данных.

§ 5.1. Понятие СУБД

Системы управления базами данных* (СУБД) – это управляющие программы, которые обеспечивают все манипуляции с базами данных: создание базы, ее ведение, ее использование многими пользователями и др., т. е. реализуют сложный комплекс функций по централизованному управлению базой данных и обслуживают интересы пользователей.

* database management system

Основные функции СУБД:

- ✓ управление данными во внешней памяти (на дисках);
- ✓ управление данными в оперативной памяти с использованием дискового кэша;
- ✓ журнализация изменений, резервное копирование и восстановление базы данных после сбоев;
- ✓ поддержка языков БД (язык определения данных, язык манипулирования данными).

Компоненты СУБД:

- ✓ ядро, которое отвечает за управление данными во внешней и оперативной памяти, и журнализацию,
- ✓ процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода,
- ✓ подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД,
- ✓ сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

§ 5.2. Классификация СУБД

1. По модели данных:
   - ✓ *Иерархические*
   - ✓ *Сетевые*
   - ✓ *Реляционные*
   - ✓ *Объектно-ориентированные*
2. По степени распределённости:
   - ✓ *Локальные СУБД (все части локальной СУБД размещаются на одном компьютере)*
   - ✓ *Распределённые СУБД (части СУБД могут размещаться на двух и более компьютерах).*

3. По способу доступа к БД:

3.1. ***Файл-серверные***. В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. СУБД располагается на каждом клиентском компьютере (рабочей станции). Доступ СУБД к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок. Преимуществом этой архитектуры является низкая нагрузка на ЦП сервера. Недостатки: потенциально высокая загрузка локальной сети; затруднённость централизованного управления; затруднённость обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность. Применяются чаще всего в локальных приложениях, которые используют функции управления БД. На данный момент файл-серверные СУБД считаются устаревшими. Примеры: Microsoft Access, Paradox, dBase, FoxPro, Visual FoxPro.

3. По способу доступа к БД:

3.2. ***Клиент-серверные.***

Клиент-серверная СУБД располагается на сервере вместе с БД и осуществляет доступ к БД непосредственно, в монопольном режиме. Все клиентские запросы на обработку данных обрабатываются клиент-серверной СУБД централизованно. Недостаток клиент-серверных СУБД состоит в повышенных требованиях к серверу. Достоинства: потенциально более низкая загрузка локальной сети; удобство централизованного управления; удобство обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность.

Примеры: Oracle, Firebird, Interbase, IBM DB2, MS SQL Server, Sybase, PostgreSQL, MySQL, ЛИНТЕР, Cache', MDBS.

3. По способу доступа к БД:

3.3. ***Встраиваемые***

Встраиваемая СУБД — библиотека, которая позволяет унифицированным образом хранить большие объёмы данных на локальной машине. Доступ к данным может происходить через SQL либо через особые функции СУБД. Встраиваемые СУБД быстрее обычных клиент-серверных и не требуют установки сервера, поэтому востребованы в локальном ПО, которое имеет дело с большими объёмами данных (например, геоинформационные системы).

Примеры: OpenEdge, SQLite, BerkeleyDB, один из вариантов Firebird, MySQL, Sav Zigzag, Microsoft SQL Server Compact, ЛИНТЕР.

.

§ 5.3. *Производительность СУБД*

Производительность – это величина, обратно пропорциональная времени, которое СУБД затрачивает на определенную операцию по обработке данных.

Показатели производительности:

– время выполнения запросов;

– время создания индексов и выполнения операций обновления, вставки и удаления данных;

– время поиска информации в неиндексированных полях;

– время выполнения операций импорта данных из файлов других форматов;

– максимальное число параллельных обращений к данным в многопользовательском режиме;

– время генерации отчетов.

§ 5.4. Режимы работы пользователя с СУБД

Все современные СУБД имеют графический пользовательский интерфейс, через который возможна работа пользователя с СУБД в трех режимах:

1. Режим работы **через меню системы** обеспечивает взаимодействие пользователя с БД в интерактивном режиме. Он реализуется чаще всего в виде различных меню и диалоговых окон, с помощью которых пользователь постепенно уточняет, какие действия он хочет выполнить и какую информацию получить из БД. Для этого не надо знать языка СУБД.

2. **Командный режим** обеспечивает диалог с БД на уровне синтаксических конструкций языка СУБД. Этот режим требует определенной подготовки пользователя, но обеспечивает более быстрый доступ к ресурсам БД.

3. **Программный режим** обеспечивает организацию доступа к данным и управление ими из прикладных программ.

Глава 6.

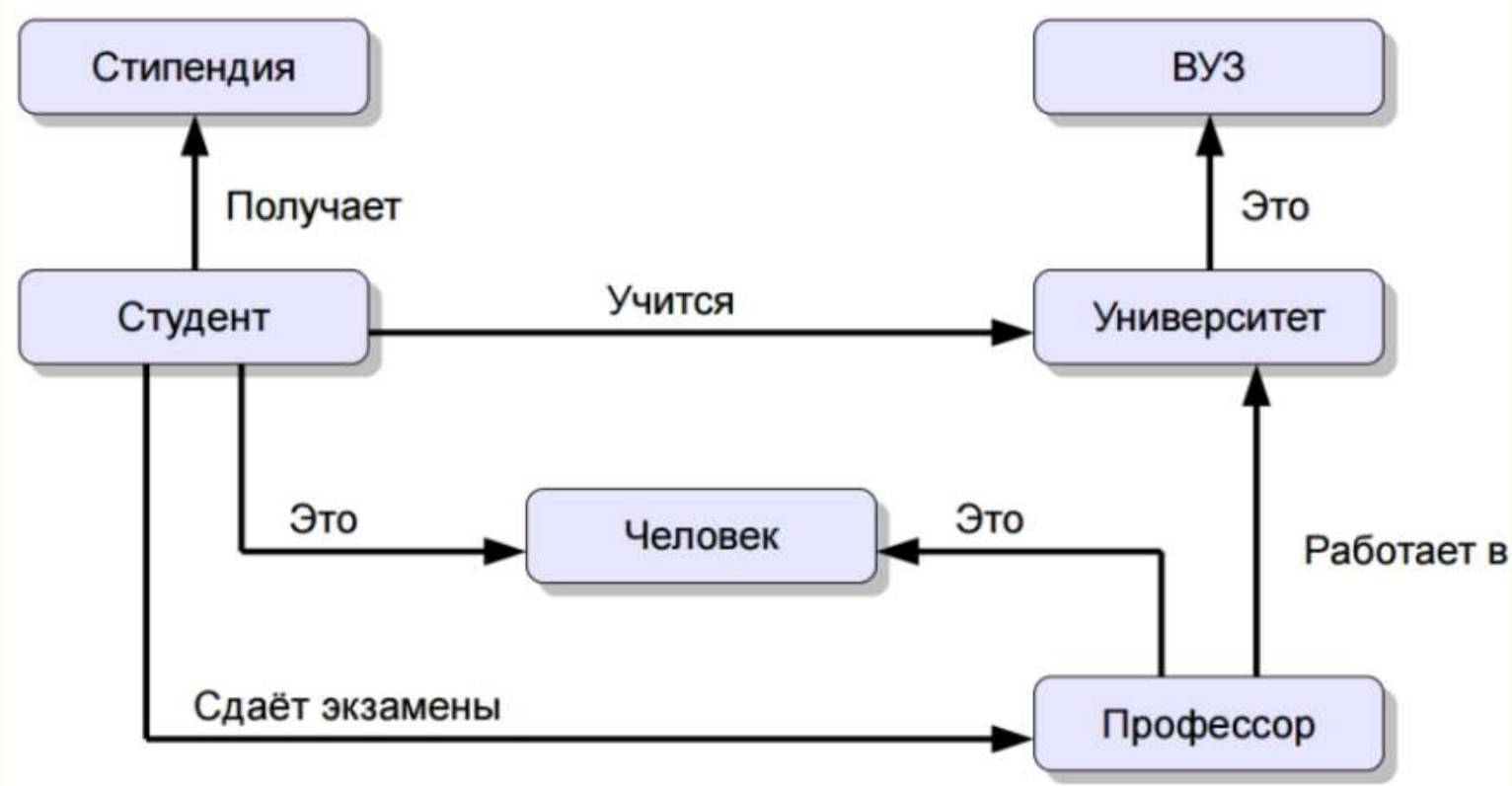
Проектирование реляционных баз данных

6.1. Модель "сущность" – "связь" англ. entity-relation (сокр. ER)

Этап **инфологического** проектирования БД – это построение модели предметной области, не привязанной к конкретной СУБД, понятной не только разработчикам ИС, но и экономистам, менеджерам и другим специалистам.

В то же время модель предметной области должна максимально точно отражать семантику предметной области, выявлять бизнес-правила и позволять легко перейти к модели данных конкретной СУБД.

Семантическая сеть – это информационная модель предметной области, имеющая вид ориентированного графа, вершины которого соответствуют объектам предметной области, а дуги (ребра) – задают отношения между ними.



Такими моделями являются модели **"сущность-связь"**. Зачастую эти модели называют моделями данных. Но этот термин неудачен, т. к. перекликается с термином "модель данных". Известно несколько методологий построения моделей "сущность-связь".

Наибольшее распространение получила методология IDEF1X.

Модель "сущность-связь" строится в виде диаграммы "сущность-связь", основными компонентами которой являются **сущности** (Entity) и **связи** (Relationship). Отсюда происходят часто используемые названия модели (ER-модель) и диаграммы (ER-диаграмма).

Сущность – это абстракция множества предметов или явлений реального мира, информацию о которых надо сохранить (т.е. описывает класс однотипных объектов).

Отдельный элемент этого множества называется **экземпляром** сущности. Реально существующий объект или предмет может быть представлен в нескольких сущностях модели данных.

Все экземпляры сущности имеют одинаковые характеристики и подчиняются одним и тем же правилам поведения.

Например, можно выделить сущность СТУДЕНТ. Экземплярами сущности СТУДЕНТ будут данные о конкретных студентах. Сущность должна иметь имя – существительное в единственном числе.

Сущности обладают определенными свойствами – атрибутами.

Атрибут – это абстракция одной характеристики объекта или явления реального мира. Каждый атрибут должен иметь имя – существительное в единственном числе, и получать значение из некоторого множества допустимых значений – домена.

У каждой сущности должен быть выделен идентификатор, или первичный ключ. **Первичный ключ** – это один или несколько атрибутов, однозначно определяющих каждый экземпляр сущности. Если первичный ключ состоит из нескольких атрибутов, то он называется составным. Первичный ключ не должен изменяться и принимать неопределенное значение (NULL). Ключ должен быть компактным, т. е. не содержать слишком много атрибутов.

Сущность может иметь несколько **потенциальных** ключей, из которых должен быть выбран первичный ключ.

Иногда приходится использовать **искусственный** первичный ключ (некоторый номер или код), когда ключ содержит слишком много атрибутов (в пределе каждый экземпляр сущности может определяться всем множеством атрибутов). Используется также понятие внешнего ключа. **Внешний** ключ – это первичный ключ другой сущности, который мигрирует (копируется) в сущность и служит для связи сущностей.

Графически сущность изображается в виде прямоугольника, в верхней части которого записано имя сущности, а в середине размещены свойства или атрибуты сущности.

Сущность может иметь несколько **потенциальных** ключей, из которых должен быть выбран первичный ключ.

Иногда приходится использовать **искусственный** первичный ключ (некоторый номер или код), когда ключ содержит слишком много атрибутов (в пределе каждый экземпляр сущности может определяться всем множеством атрибутов). Используется также понятие внешнего ключа. **Внешний** ключ – это первичный ключ другой сущности, который мигрирует (копируется) в сущность и служит для связи сущностей.

Графически сущность изображается в виде прямоугольника, в верхней части которого записано имя сущности, а в середине размещены свойства или атрибуты сущности.

Пример сущности:

Каждая сущность должна сопровождаться описанием.

Описание сущности должно объяснять ее смысл, а не то, как будет использоваться информация данной сущности.

Описание должно быть ***ясным, полным и непротиворечивым***, понятным специалистам предметной области.

Сущности и атрибуты выделяются в результате анализа требований к системе. При выборе атрибутов целесообразно придерживаться правил нормализации.

Связи между объектами реального мира отражаются в виде ***отношений***, (ассоциаций) между сущностями. Отношение представляется в модели линией, соединяющей две сущности, и именем отношения – глагольной конструкцией, которая описывает, как две сущности зависят друг от друга.

Имена сущностей, соединенные именем отношения, должны образовывать осмысленную фразу, описывающую бизнес-правило отношения.

Например,

СТУДЕНТ <Обучается в> УЧЕБНАЯ ГРУППА.

В примере имя отношения показано в угловых скобках. Отношения двунаправлены, поэтому должны иметь имена в каждом направлении.

Отношение обладает следующими **свойствами**: ***степень, направленность, тип, мощность, обязательность.***

Степень отношения представляет собой число сущностей, ассоциированных с отношением. Чаще всего используются бинарные отношения, связывающие две сущности.

Унарные, или рекурсивные отношения представляют случаи, когда экземпляр сущности связан с другим экземпляром той же самой сущности.

Часто унарные или рекурсивные отношения рассматриваются как бинарные рекурсивные отношения, связывающие экземпляр сущности с другим ее экземпляром.

Направленность отношения указывает на исходную сущность в отношении.

Сущность, из которой отношение исходит, называется ***родительской*** сущностью.

Сущность, в которой отношение заканчивается, называется подчиненной (***дочерней***) сущностью.

Направленность отношения определяется взаимосвязью между сущностями и зависит от типа и мощности отношения.

В отношении между независимой и зависимой сущностями отношение исходит из независимой сущности и заканчивается в зависимой сущности. Если обе сущности независимые, отношение симметрично.

В отношении один-ко-многим родительской является сущность, входящая в отношение однократно.

Отношения многие-ко-многим симметричны. Ключ родительской сущности мигрирует (повторяется) в дочерней сущности. Такой мигрировавший ключ в дочерней сущности называется ***внешним*** ключом.

Внешний ключ в зависимости от типа связи может стать частью составного ключа дочерней сущности или неключевым атрибутом дочерней сущности.

С помощью внешнего ключа экземпляр дочерней сущности ссылается на соответствующий экземпляр родительской сущности.

6.2. Инструментальные средства моделирования БД

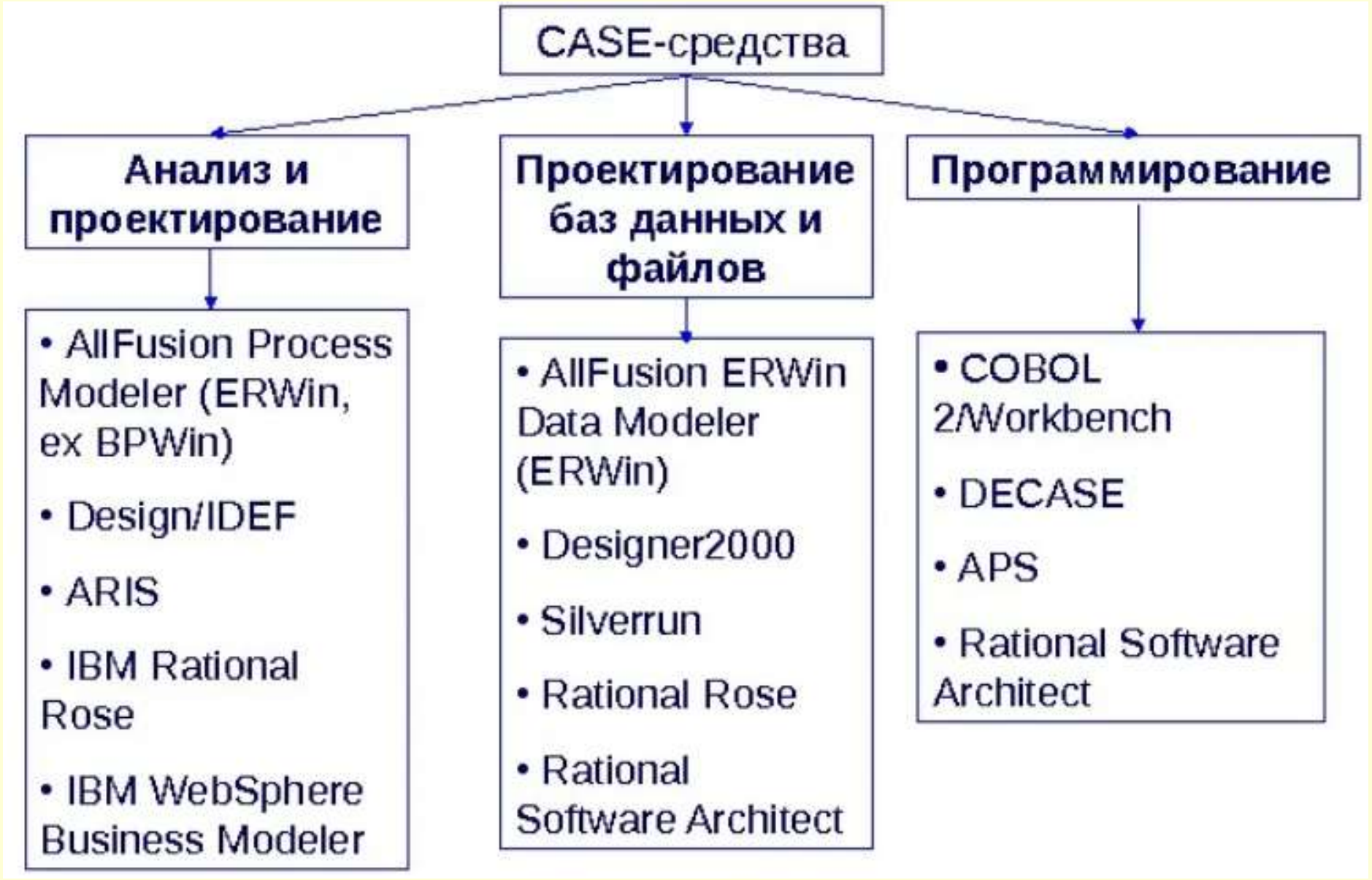
Под ***инструментальным средством*** для моделирования баз данных понимается компьютерная программная реализация (программное приложение), реализующая одну определенную или множество нотаций представления структур данных и связей между ними в рамках некоторой методологии проектирования.

В настоящее время для проектирования БД активно используются CASE-средства, в основном ориентированные на использование ERD (Entity – Relationship Diagrams, диаграммы «сущность-связь»).

С их помощью определяются важные для предметной области объекты (сущности), отношения друг с другом (связи) и их свойства (атрибуты).

Следует отметить, что средства проектирования ERD в основном ориентированы на реляционные базы данных (РБД), и если существует необходимость проектирования другой системы, скажем объектно-ориентированной, то лучше избрать другие методы проектирования.

CASE-средства (Computer - Aided Software Engineering) – это методы и технологии, которые позволяют проектировать различные информационные системы (в частности, базы данных) и автоматизировать их создание.


CASE-средства
Анализ и проектирование
Проектирование баз данных и файлов
Программирование
• AllFusion Process Modeler (ERWin, ex BPWin)
• Design/IDEF
• ARIS
• IBM Rational Rose
• IBM WebSphere Business Modeler
• AllFusion ERWin Data Modeler (ERWin)
• Designer2000
• Silverrun
• Rational Rose
• Rational Software Architect
• COBOL 2/Workbench
• DECASE
• APS
• Rational Software Architect

Функции CASE-средств:

1. Построение логической и физической схемы БД.
2. Прямой и обратный инжиниринг БД.
3. Проектирование других объектов БД (обзоры, триггеры, хранимые процедуры).
4. Генерация DDL-скрипта для создания БД.
5. Разбиение большой схемы БД на отдельные подсхемы.
6. Генерация отчетов.

ПРЯМОЕ ПРОЕКТИРОВАНИЕ

Процесс генерации физической схемы базы данных на основе физической модели данных называется **прямым** проектированием (Forward Engineering).

Методоло́гия — учение о методах, способах и стратегиях исследования предмета.

Концептуальное проектирование с использованием методологии IDEF1X

Методология IDEF1X применяется для построения информационной модели, которая представляет структуру информации, необходимой для поддержки функций производственной системы.

Построение IDEF1X-модели тесно связано с понятием управления данными, того, чтобы управлять данными, необходимо понимать их основные характеристики и организацию с целью дальнейшей интерпретации этих данных.

Методология IDEF1X была разработана в США и теперь используется в правительственных, аэрокосмических и финансовых учреждениях, а также в большом числе частных компаний.

Методология IDEF1X определяет стандарты терминологии, используемой при информационном моделировании, и графического изображения типовых элементов на диаграммах.

IDEF1X использует подход сущностей - отношений к семантическому моделированию данных. Разработка IDEF1X базируется на понятиях сущности-отношения по методу П.Чена, объединенных с понятиями реляционной теории Т.Кодда.

Компонентами IDEF1 X-модели являются сущности, отношения и атрибуты.

Последовательность шагов при концептуальном проектировании:

1. Выделение сущностей.

В графической нотации IDEF1X для отображения сущности используются обозначения, изображенные на следующем рисунке:

а) независимая б) зависимая

2. Определение атрибутов.

Как правило, атрибуты указываются только для сущностей. Если у связи имеются атрибуты, то это указывает на тот факт, что связь является сущностью.

Самый простой способ определения атрибутов – после идентификации сущности или связи, задать себе вопрос «Какую информацию требуется хранить о …?».

Существенно помочь в определении атрибутов могут различные бумажные и электронные формы и документы, используемые в организации при решении задачи. Это могут быть формы, содержащие как исходную информацию (например, «Ведомость …»), так и результаты обработки данных (например, «Форма № 1»).

3. Определение связей.

Связь характеризуется следующим набором параметров:

- ***именем*** – указывается в виде глагола и определяет семантику (смысловую подоплеку) связи;

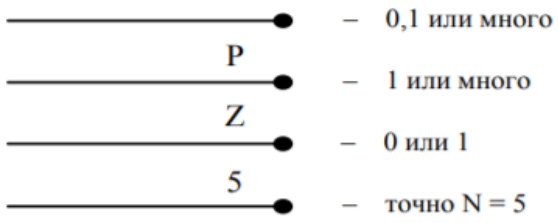
- ***кратностью*** (кардинальность, мощность): один-к-одному (1:1), один-ко-многим (1:N) и многие-ко-многим (N:M, N = M или N <> M). Кратность показывает, какое количество экземпляров одной сущности определяется экземпляром другой;

> ***Мощность*** *связи (Cardinality) служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней сущности. Различают 4 типа мощности связи:*
>
> *– общий случай – **не помечается каким-либо символом** и соответствует ситуации, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности;*

*– символом **P** помечается случай, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности, т.е. исключено нулевое значение;*

*– символом **Z** помечается случай, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляр дочерней сущности, т.е. исключены множественные значения;*

*– **цифрой помечается случай точного соответствия**, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.*



- ***типом***: идентифицирующая (атрибуты одной сущности, называемые внешним ключом, входят в состав дочерней и служат для идентификации ее экземпляров, т.е. входят в ее первичный ключ) и неидентифицирующая (внешний ключ имеется в дочерней сущности, но не входит в состав первичного ключа);

- ***обязательностью***: обязательная (при вводе нового экземпляра в дочернюю сущность заполнение атрибутов внешнего ключа обязательно и введенные значения должны совпадать со значениями атрибутов первичного ключа какого-либо экземпляра родительской сущности) и необязательная (заполнение атрибутов внешнего ключа в экземпляре дочерней сущности необязательно или введенные значения не совпадают со значениями атрибутов первичного ключа ни одного экземпляра родительской сущности);

- ***степенью участия*** – количеством сущностей, участвующих в связи. В основном между сущностями существуют бинарные связи, т.е. ассоциации, связывающие две сущности (степень участия равна 2).

4. Правила ссылочной целостности

Целостность данных является понятием базы данных. ERwin позволяет в модели "сущность-связь" задать правила ссылочной целостности.

При генерации схемы базы данных ERwin генерирует правила декларативной ссылочной целостности и триггеры. То есть, правила ссылочной целостности задаются в модели "сущность-связь", а реализуются в построенной по этой модели реляционной базе данных.

Соответствие между моделью и базой данных легко устанавливается, учитывая, что сущностям модели соответствуют отношения реляционной БД, а экземплярам сущностей – кортежи отношений.

Так как внешние ключи кортежей дочернего отношения служат ссылками на соответствующие кортежи родительского отношения, то эти ссылки не должны указывать на несуществующие кортежи. Это определяет следующее правило целостности внешних ключей (правило ссылочной целостности): для каждого значения внешнего ключа должно существовать соответствующее значение первичного ключа в родительском отношении.

Ссылочная целостность может нарушиться в результате операций с кортежами отношений. Таких операций три: ***вставка, обновление и удаление кортежей в отношениях***.

Рассмотрим эти операции для родительского и дочернего отношений.

При операциях с кортежами **родительского отношения** возможны следующие ситуации:

1. ***Вставка кортежа в родительское отношение***. Так как допустимо существование кортежей родительского отношения, на которые нет ссылок из дочерних отношений, то вставка кортежа в родительское отношение не нарушает ссылочной целостности.
2. ***Обновление кортежа родительского отношения.*** При обновлении кортежа родительского отношения может измениться значение ключа. Если есть экземпляры дочернего отношения, ссылающиеся на обновляемый кортеж родительского отношения, то значения их внешних ключей станут некорректными. Обновление кортежа в родительском отношении может привести к нарушению ссылочной целостности, если это обновление затрагивает значение ключа.
3. ***Удаление кортежа родительского отношения***. При удалении кортежа родительского отношения удаляется значение ключа. Если есть кортежи дочернего отношения, ссылающиеся на удаляемый кортеж родительского отношения, то значения их внешних ключей станут некорректными. Удаление кортежа родительского отношения может привести к нарушению ссылочной целостности.

При операциях с кортежами **дочернего отношения** возможны следующие ситуации:

1. Вставка кортежа дочернего отношения может привести к нарушению ссылочной целостности, если вставляемое значение внешнего ключа некорректно.

2. Обновление кортежа дочернего отношения может привести к нарушению ссылочной целостности при некорректном изменении значения внешнего ключа.

3. При удалении кортежа дочернего отношения ссылочная целостность не нарушается.

Таким образом, ссылочная целостность в принципе может быть нарушена при выполнении одной из четырех операций:

1) обновление кортежа в родительском отношении;
2) удаление кортежа в родительском отношении;
3) вставка кортежа в дочернее отношение;
4) обновление кортежа в дочернем отношении.

Существуют ***две*** основные стратегии поддержания ссылочной целостности:

1) **RESTRICT (ОГРАНИЧИТЬ)** – не разрешать выполнение операции, приводящей к нарушению ссылочной целостности. Это самая простая стратегия, требующая только проверки, имеются ли кортежи дочернего отношения, связанные с некоторыми кортежами родительского отношения.

2) **CASCADE (КАСКАД)** – разрешить выполнение требуемой операции, но внести при этом необходимые поправки в других кортежах отношений так, чтобы не допустить нарушения ссылочной целостности и сохранить все имеющиеся связи. Изменение начинается в родительском отношении и каскадно выполняется в дочернем отношении. Так как дочернее отношение может быть родительским для некоторого третьего отношения, то может потребоваться выполнение каскадной стратегии и для этой связи и т.д. Это самая сложная стратегия, но она хороша тем, что при этом не нарушается связь между кортежами родительского и дочернего отношений.

Эти стратегии являются стандартными и присутствуют во всех СУБД, в которых имеется поддержка ссылочной целостности.

ERwin реализует также дополнительные стратегии поддержания ссылочной целостности (если они реализованы в целевой СУБД):

1) **NONE (НИКАКОЙ)** – никаких операций по поддержке ссылочной целостности не выполняется. В этом случае в дочернем отношении могут появляться некорректные значения внешних ключей, и вся ответственность за целостность базы данных ложится на приложение.

2) **SET NULL (УСТАНОВИТЬ В NULL)** – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей заменять на неопределенные значения (null-значения). При этом кортежи

дочернего отношения теряют всякую связь с кортежами родительского отношения.

3) **SET DEFAULT (УСТАНОВИТЬ ПО УМОЛЧАНИЮ)** – разрешить выполнение требуемой операции, но все возникающие некорректные значения внешних ключей изменять на некоторое значение, принятое по умолчанию. При этом должен существовать кортеж родительского отношения, первичный ключ которого принят как значение по умолчанию для внешних ключей. Этот кортеж нельзя удалять из родительского отношения, и в этом кортеже нельзя изменять значение ключа. Кроме того, как и в предыдущем случае, кортежи дочернего отношения теряют всякую связь с кортежами родительского отношения.

Рассмотрим применение стратегии поддержания ссылочной целостности.

При **обновлении кортежа в родительском отношении** допустимы следующие стратегии:

1) ***RESTRICT*** – не разрешать обновление, если имеется хотя бы один кортеж дочернего отношения, ссылающийся на обновляемый кортеж родительского отношения.

2) ***CASCADE*** – выполнить обновление и каскадно изменить значения внешних ключей во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж.

3) ***SET NULL*** – выполнить обновление и во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж, изменить значения внешних ключей на null-значение.

4) ***SET DEFAULT*** – выполнить обновление и во всех кортежах дочернего отношения, ссылающихся на обновляемый кортеж, изменить значения внешних ключей на некоторое значение, принятое по умолчанию.

5) ***NONE*** – выполнить обновление, не обращая внимания на нарушения ссылочной целостности.

При **удалении кортежа в родительском отношении** допустимы стратегии:

1) ***RESTRICT*** – не разрешать удаление, если имеется хотя бы один кортеж в дочернем отношении, ссылающийся на удаляемый кортеж.

2) ***CASCADE*** – выполнить удаление и каскадно удалить кортежи в дочернем отношении, ссылающиеся на удаляемый кортеж.

3) ***SET NULL*** – выполнить удаление и во всех кортежах дочернего отношения, ссылающихся на удаляемый кортеж, изменить значения внешних ключей на null-значение.

4) ***SET DEFAULT*** – выполнить удаление и во всех кортежах дочернего отношения, ссылающихся на удаляемый кортеж, изменить значения внешних ключей на некоторое значение, принятое по умолчанию.

5) ***NONE*** – выполнить удаление, не обращая внимания на нарушения ссылочной целостности.

При ***вставке кортежа*** **в дочернее отношение** возможны стратегии:

1) ***RESTRICT*** – не разрешать вставку, если внешний ключ во вставляемом кортеже не соответствует ни одному значению потенциального ключа родительского отношения.

2) ***SET NULL*** – вставить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а null-значение.

3) ***SET DEFAUL***T – вставить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а некоторое значение, принятое по умолчанию.

4) ***NONE*** – вставить кортеж, не обращая внимания на нарушения ссылочной целостности

При **обновлении кортежа в дочернем отношении** возможны стратегии:

1) ***RESTRICT*** – не разрешать обновление, если внешний ключ в обновляемом кортеже становится не соответствующим ни одному значению потенциального ключа родительского отношения.

2) ***SET NULL*** – обновить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а null-значение.

3) ***SET DEFAULT*** – обновить кортеж, но в качестве значения внешнего ключа занести не предлагаемое пользователем некорректное значение, а некоторое значение, принятое по умолчанию.

4) ***NONE*** – обновить кортеж, не обращая внимания на нарушения ссылочной целостности.

Глава 7.

Общая характеристика СУБД Microsoft ACCESS

MS Access (MsA) – это функционально полная реляционная СУБД, в которой предусмотрены все необходимые средства для определения и обработки данных, а также для управления ими при работе с большими объемами информации. Различные ее версии входят в состав программного пакета MS Office и работают в среде Windows.

§ 7.1. Создание файла БД в ACCESS

Способы создания файла БД:

1. С помощью шаблонов (в окне ***Создание***, вкладка ***Базы данных***),

2. Создание нетиповой БД:

в окне ***БД*** выбрать *п.м. Файл → Создать БД*,

в окне ***Создание БД*** установить переключатель в положение *Новая БД* и нажать **ОК**,

в появившемся окне ***Файл новой БД*** указать путь, где создать файл и присвоить имя файлу БД,

нажать кнопку **Создать**.

§ 7.2. Основные объекты СУБД Access

Объектами в Access являются:

Таблица – объект, используемый для хранения пользовательских данных. Таблица содержит поля и записи.

Запрос – объект, который позволяет пользователю получить нужные данные из одной или нескольких таблиц.

Форма – объект, предназначенный для ввода данных, отображения их на экране или управления работой приложения.

Отчет – объект, предназначенный для формирования выходного документа, который может быть распечатан.

Страницы доступа к данным позволяют редактировать, просматривать и обрабатывать данные, используя интернет-браузер.

Макрос – набор команд, который позволяет автоматизировать часто выполняемые операции.

Модуль – программы, написанные на языке программирования Visual Basic, которые могут разрабатываться пользователем для реализации нестандартных процедур при создании приложения.

§ 7.3. Режимы работы с объектами в MsA:

Открыть – позволяет перейти в режим редактирования таблицы, выполнения запроса, загрузки формы, построения отчета, запуска макроса.

Конструктор – обеспечивает переход к режиму настройки выбранного объекта.

Создать – позволяет приступить к созданию нового объекта выбранного типа.

§ 7.4. Типы данных, обрабатываемых MsA:

Текстовый – для хранения обычного текста с максимальным количеством символов 255.

Поле МЕМО – для хранения больших объемов текста до 65 535 символов.

Числовой – для хранения действительных чисел.

Дата/время – для хранения календарных дат и текущего времени.

Денежный – эти поля содержат денежные суммы.

Счетчик – для определения уникального системного ключа таблицы. Обычно используется для порядковой нумерации записей. При добавлении в таблицу новой записи значение этого поля увеличивается на 1 (единицу). Значения в таких полях не обновляются.

Логический – для хранения данных, принимающих значения: Да или Нет.

Поле объекта OLE – для хранения объектов, созданных в других приложениях.

Гиперссылка – для хранения ресурсов сети Internet или Intranet.

Глава 8.

Создание и работа с БД посредством графического интерфейса СУБД.

Технологии работы с базой данных в MS Access.

§ 8.1. Проектирование таблиц в MsA

Таблица как объект БД

Таблица – объект, используемый для хранения пользовательских данных.

Способы создания таблиц в MS Access:

- построить новую таблицу «с нуля» непосредственно в режиме таблицы;
- спроектировать таблицу, воспользовавшись Конструктором;
- запустить Мастер таблиц – специальную программу, предлагающую создать таблицу в пошаговом режиме на базе типовых решений, имеющихся в Access;
- импортировать таблицу БД из файла какой-либо программы, например, текстового редактора или табличного процессора Excel.

§ 8.2. Связи между таблицами в MsA

Схема данных

Структура РБД в **MsA** задается **схемой данных**, графически отображаемой в отдельном окне, где таблицы представлены списками полей, а связи – линиями между полями разных таблиц.

Окно «Схема данных» вызывается с помощью кнопки Схема данных 

в ACCESS 2003: на панели инструментов или командой в п.м. Сервис

в ACCESS 2007/2010: на вкладке ленты **Работа с базой данных** в группе **Отношения** (или **Показать или Скрыть**).

При построении схемы данных **MsA** автоматически определяет по выбранному полю тип связи между таблицами:

1. если поле, по которому устанавливается связь, является уникальным ключом как в главной таблице, так и в подчиненной, **MsA** устанавливает связь **1:1**;
2. если поле связи является уникальным (первичным) ключом в главной таблице, а в подчиненной таблице является не ключевым, или входит в составной ключ, **MsA** устанавливает связь **1:M** от главной таблицы к подчиненной.

Условия создания связей между таблицами

Установление между таблицами БД связей и задание параметров целостности данных возможно только при следующих условиях:

1. связанное поле главной таблицы является первичным ключом или альтернативным ключом (имеет уникальный индекс);

2. обе таблицы принадлежат одной базе данных MsA.

3. связанные поля (первичный ключ главной таблицы и внешний ключ подчиненной таблицы) имеют один и тот же тип данных. Существует исключение: поле с типом данных **Счетчик** может быть связано с числовым полем, свойство которого ***Размер поля*** имеет значение *Длинное целое;*

4. свойства ***Размер поля*** для обоих связываемых полей **числового типа** должны быть одинаковыми.

Способы создания связей в MsA

1. Использование Мастера подстановки для создания связей между таблицами

2. Создание связей «вручную»:

- выделить в одной (главной) таблице ключевое поле (первичный ключ) и перетащить в соответствующее поле с такими же значениями (внешний ключ) другой (подчиненной) таблицы.

В открывшемся диалоговом окне «Изменение связей»:

1. установить флажок
   - ✔ Обеспечение целостности данных

При необходимости, в этом же окне можно установить каскадные операции:

- ✔ каскадное обновление связанных полей
- ✔ каскадное удаление связанных записей

2. нажать кнопку **Создать**

Замечание: перед работой в окне «Схема данных» все открытые таблицы необходимо ЗАКРЫТЬ!!!!

Каскадные операции

Режим обеспечения целостности данных может быть **ослаблен** включением каскадных операций двух видов: ***операции каскадного обновления*** и ***операции каскадного удаления.***

Если установлен ☑ «Каскадное обновление связанных полей», то любые изменения в значении ключевого поля в главной таблице, ведут к автоматическому обновлению соответствующих значений во всех связанных записях в подчиненной таблице.

При установке ☑ «Каскадное удаление связанных записей» при удалении записи из главной таблицы обеспечивается автоматическое удаление связанных записей в подчиненных таблицах.

Глава 8.

Создание и работа с БД посредством графического интерфейса СУБД.

Технологии работы с базой данных в MS Access.

Отбор данных с помощью фильтра

Фильтр – это набор условий применяемых для отбора подмножества записей в таблице.

В **MsA** существуют следующие фильтры :

- ***фильтр по выделенному фрагменту;***
- ***обычный фильтр;***
- ***расширенный фильтр.***

§ 8.3. Проектирование запросов в MsA

Понятие запроса. Виды запросов.

Запрос (query) – механизм выбора и представления информации из БД. Запрос направляется пользователем или программой в СУБД для поиска отдельных записей в БД.

Виды запросов:

- запросы на выборку;
- активные запросы.

Способы формирования запросов:

- с помощью запросов по образцу (QBE – Query By Example)
- с помощью последовательности SQL-инструкций специального структурированного языка запросов SQL (Structured Query Language).

Запрос на выборку – задает вопрос базе данных и отображает полученный ответ виде динамического набора данных.

Может быть *однотабличным* (в качестве исходной информации используются данные одной таблицы) и *многотабличным* (в качестве исходной информации используются данные нескольких таблиц).

Формирование условий отбора.

Условие отбора – это выражение, которое состоит из **операторов сравнения** и сравниваемых **операндов**.

В качестве **операндов** могут быть выбраны некоторые ***заданные значения*** и ***идентификаторы*** – ссылки на значение поля, элемента управления или свойства (имена полей, таблиц, запросов, форм, отчетов и т.д.).

Идентификаторы необходимо заключать в квадратные скобки. Если надо указать ссылку на поле в конкретной таблице, форме, отчете, то прописывается полное имя поля в виде:

[Имя таблицы]![Имя поля].

Список **операторов** используемых при задании выражений:

1. Операторы сравнения:

= (равно)

< > (не равно)

> (больше)

>= (не меньше)

< (меньше)

<= (не больше)

BETWEEN – позволяет задать диапазон значений.

Синтаксис:

Between *«Выражение»* ***And*** *«Выражение»*

(например: BETWEEN 10 And 20 означает тоже, что и логическое выражение >=10 AND <=20).

IN – позволяет задавать используемый для сравнения список значений (операндом является список, заключенный в круглые скобки).

Например: IN("Брест", "Минск", "Гродно") означает тоже самое, что и логическое выражение "Брест" OR "Минск" OR "Гродно".

2. Логические операторы:

AND – логическое **И** или **конъюнкция**, т.е. ***логическое умножение***. Объединяет два и более условий и возвращает истинное значение только при выполнении всех условий
(например, от 10 и до 20: >=10 AND <=20).

OR – логическое **ИЛИ**, эквивалент логического сложения – **дизъюнкции**. Связывает два или больше условий, но возвращает истинный результат при выполнении хотя бы одного условия (например, до 50 или свыше 100: <50 OR >100).

NOT – отрицание
(например: Is Not Null – поле, содержащее какое-либо значение).

3. Оператор LIKE – проверяет соответствие текстового или Memo-поля по заданному шаблону символов:

Символы шаблона	*Соответствие в выражении*
?	Любой один текстовый символ
*	Соответствует любой цифре или любому символу
#	Любая одна цифра
[список знаков]	Любой один знак в «списке знаков»
[!список знаков]	Любой один знак, который не входит в «список знаков»

Формирование условий отбора для полей с типом данных Дата/Время

Для того, чтобы сообщить Access, что вводится дата (или время), необходимо значение даты (или времени) заключить в знак #.

Например: #15.10.19#.

Сложные критерии выборки

Для выборки записей по условию, которое задается для *нескольких полей* таблицы или по *нескольким условиям для одного поля*, применяются два вида запросов:

«И-запросы» (выбор записей только при условии выполнения всех условий);

«ИЛИ-запросы» (выбор записей при выполнении хотя бы одного из условий).

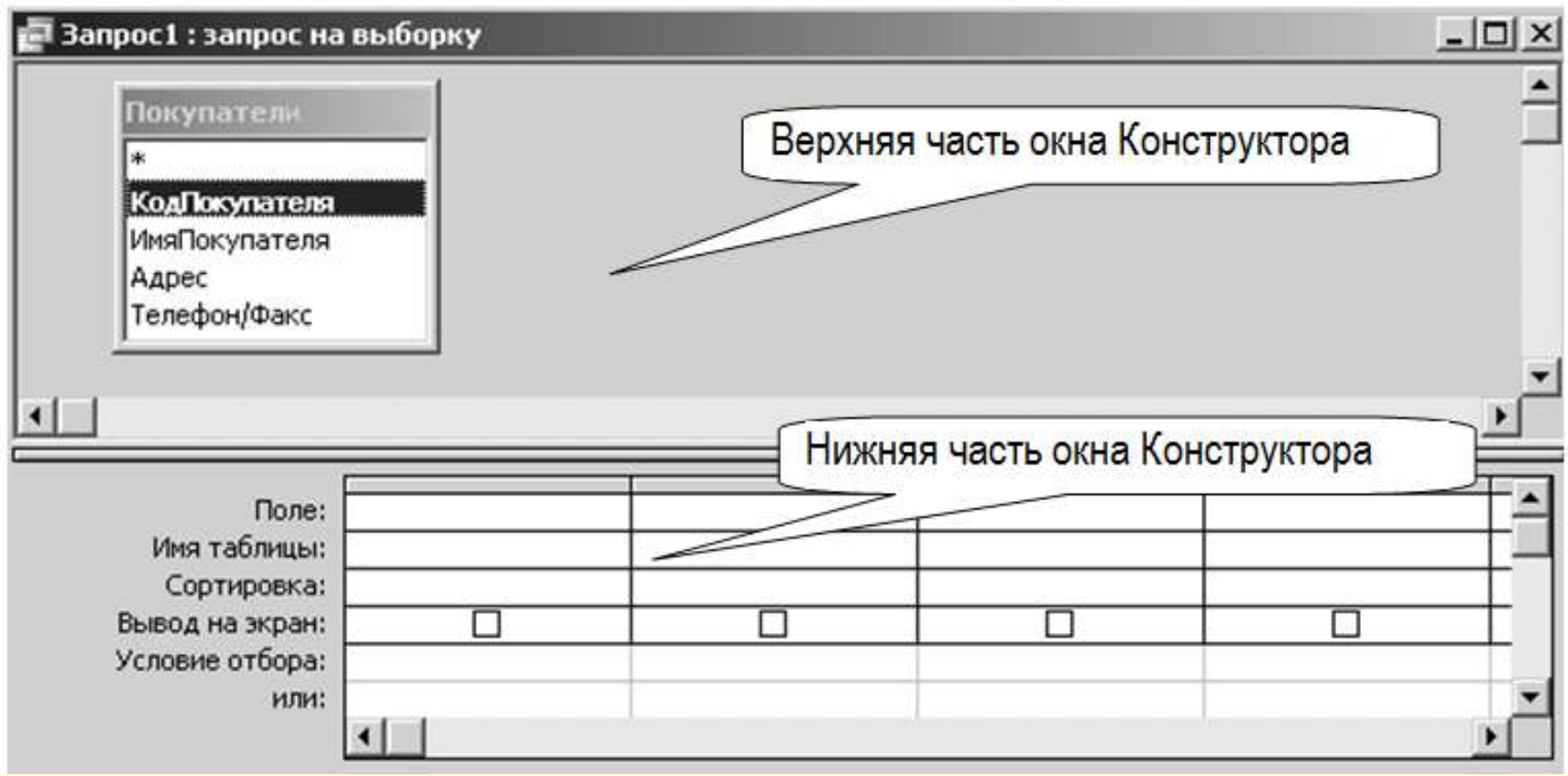
При задании «***ИЛИ-запроса***» каждое условие выборки должно размещаться на **отдельной** строке *Бланка запроса*.

При задании «***И-запроса***» каждое условие выборки должно размещаться на **одной** строке, но в разных полях Бланка запроса.

Замечание: Эти операции для условий отбора по *одному полю* могут быть заданы явно с помощью операторов **OR** и **AND** соответственно.

Технология создания простых запросов.

Создание запросов на выборку с помощью Конструктора

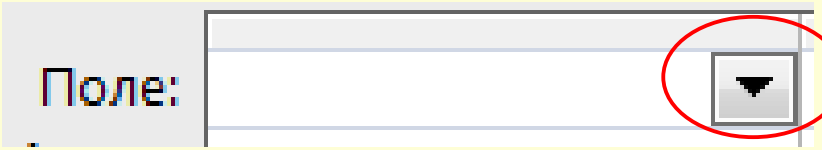


Этапы формирования запроса в Конструкторе:

1. в верхней панели отобразить таблицы (или другие запросы), участвующие в текущем запросе с помощью окна «Добавление таблицы»;
2. выбрать поля результатов запроса;
3. указать критерии для выборки данных, группировки, сортировки и вывода данных на экран;
4. сохранить запрос: ***п.м. Файл → Сохранить запрос***;

Замечание: имя запроса должно быть информативным и **отличаться** от имени любого другого объекта в исходной БД.

Способы включения полей таблицы в бланк QBE запроса:

1. Непосредственно в ячейке строки «Поле» бланка QBE. 

(Alt + ↓).

2. Перетаскивание поля из верхней части окна запроса из таблицы в бланк QBE в нужную ячейку.

3. Двойной щелок по выбранному полю в таблице.

Замечание: для включения **всех** полей в бланк QBE нужно выбрать пункт * , при этом отключить режим вывода на экран для дополнительных полей, по которым задается критерий отбора.

Выполнение запроса из окна Конструктора:

в ACCESS 2003: на панели инструментов нажать кнопку «**Запуск**» ! или выполнить команду **Запуск** из пункта меню **Запрос**;

в ACCESS 2007/2010:

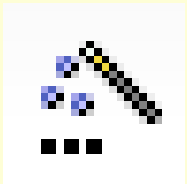
на вкладке ленты **Конструктор** в группе инструментов **Результаты** запустить команду

! ***Выполнить***.

Замечание1: Просмотр результата запроса происходит в режиме таблицы.

Замечание2: Сохраненный запрос может быть выполнен в любой момент нажатием кнопки **Открыть** в окне БД.

Построитель выражений – предназначен для составления математических, логических и других выражений при изменении свойств полей таблиц, при задании условий отбора и создании вычисляемых полей в запросах. Этот инструмент обеспечивает простой доступ к именам полей и элементов управления в БД, а также ко многим встроенным функциям, которые можно включать в выражения.



Запросы с параметром (параметрические запросы) –

это запросы, в которых критерий отбора может задать сам пользователь, введя нужный параметр при вызове запроса. Такой запрос каждый раз при выполнении требует ввода определенных параметров.

Преимущества параметрического запроса:

- не нужно постоянно модифицировать запрос в режиме Конструктора;
- использование в формах и отчетах, т.к. каждый раз при их открытии MsA запрашивает у пользователей требуемый параметр.

Чтобы установить параметр нужно вместо конкретных данных в бланк QBE в строку условие отбора ввести имя или фразу, заключенную в квадратные скобки.

То что заключено в ([ ]) MsA рассматривает, как параметр.

Пример: Выборка по полю типа Дата/время значений, попадающих в интервал, границы которого вводятся с клавиатуры:

Between [Введите начальную дату:] And [Введите конечную дату:]

>= [Введите начальную дату:] And <= [Введите конечную дату:]

Замечание1: если необходимо изменить тип данных параметра, нужно выполнить команду ***Запрос → Параметры*** и в диалоговом окне **Параметры запроса** ввести имена каждого параметра в столбец «Параметры» в том виде в каком они введены в бланк QBE и задать **Тип данных** из раскрывшегося списка.

По умолчанию Тип параметра - *Текстовый*.

Замечание2: если при выполнении запроса с вычисляемыми полями появляется окно для ввода значения параметра, значит в формуле данного поля некорректно указана ссылка на какой-либо объект БД.

Глава 8.

Создание и работа с БД посредством графического интерфейса СУБД.

Технологии работы с базой данных в MS Access.

§ 8.3. Проектирование запросов в MsA

Вычисляемые запросы.

В MsA можно задать вычисление над любыми полями таблицы и сделать вычисляемое значение новым полем в наборе записей.

Результаты вычислений не хранятся в таблице (т.е. не создают полей в исходных таблицах БД), а каждый раз вычисляются при запуске запроса.

Вычисляемое поле добавляется в бланк QBE в строку **Поле**. В пустую ячейку вводится выражение, которое может включать:

- ***всевозможные встроенные функции MsA,***
- ***арифметические операции с использованием полей таблицы.***

Вычисляемому полю обязательно присваивается **ИМЯ** (в противном случае, ему по умолчанию присваивается имя **Выражение1**), ставится двоеточие, затем задается (в т. ч. с помощью *Построителя*) вычисляемое выражение.

Правила формирования вычисляемого запроса в Конструкторе:

1. В окне «Схема данных» все добавляемые объекты БД (таблицы или другие запросы) должны быть взаимосвязаны!!!

2. В бланк QBE обязательно первым должно быть включено **ключевое поле** таблицы (при однотабличном запросе) или **ключевое поле подчиненной таблицы** (при многотабличном запросе).

3. Для использования в текущем запросе при вычислении нового поля ссылки (идентификатора) на ранее созданное вычисляемое поле, необходимо перед этим СОХРАНИТЬ запрос, чтобы в Построителе отобразились все поля текущего запроса для возможного их выбора в новых выражениях.

4. Создание нового текстового поля, как результата сцепления других текстовых полей, выполняется с использованием знака

&.

Установка свойств полей запроса.

В общем случае поля, выводимые в наборе записей запроса наследуют свойства, заданные ранее.

Можно изменить некоторые свойства выделенного поля (или нового вычисляемого поля), находясь в окне Конструктора запроса: ***п.м. Вид*** → ***Свойства*** или команду ***Свойства*** из **КЗМ**.

Измененные свойства можно просмотреть в режиме *Таблица*.

Свойства поля запроса:

- Описание объекта (сообщение в строке состояния Окна запроса);
- Формат поля (представление данных на экране);
- Число десятичных данных (для числовых данных);
- Маска ввода;
- Подпись поля (заголовок столбца).

Использование встроенных функций.

Функции категории «Дата/время»:

Date( ) – возвращает текущую системную дату.

Day(дата) – возвращает значение дня месяца от 1 до 31.

Month(дата) – возвращает значение ***номера*** месяца от 1 до 12.

MonthName(Month(дата)) – возвращает название месяца с *Января* по *Декабрь*.

Year(дата) – возвращает значение года от 100 до 9999.

Weekday(дата) – по умолчанию возвращает целое число от 1 (Воскресенье) до 7 (Суббота), соответствующее дню недели.

WeekdayName(Weekday(дата;2)) – возвращает название дня недели с Понедельника по Воскресенье.

Hour(дата) – возвращает целое число от 0 до 23, представляющее значение часа в дате.

DatePart("интервал", дата) – возвращает число дней, недель, месяцев и т.д. в соответствии с значением аргумента интервал для указанной даты.

"q" – определение квартала (значение от 1 до 4);

"ww" – определение номера недели в году (значение от 1 до 53).

DateAdd("интервал"; число; дата) – возвращает новую дату, равную указанной дате, увеличенной на указанное **число** в соответствии с установленной ***маской***, указанной в аргументе "интервал". Возвращает данные типа ***Дата/время***.

Значения маски:

Значение	*Описание*	*Значение*	*Описание*
yyyy	Год	w	День недели
q	Квартал	ww	Неделя
m	Месяц	h	Часы
y	Дней в году	n	Минуты
d	День	s	Секунды

DateDiff("интервал", дата1, дата2) – определяет интервал между двумя датами в указанных единицах измерения (днях, месяцах, кварталах, годах). Здесь значение аргумента "интервал" имеет то же значение, что и в функции DateAdd.

Причем: “дата1” – начальная дата, “дата2” – конечная дата, причем дата1 ≤ дата2.

Замечание: Запрос на выборку данных, в котором в критериях отбора используются функции из категории Дата/время, после сохранения преобразуется в вычисляемый запрос.

Функции категории «Управление»:

IIf(условие; еслиИстина; еслиЛожь) – возвращает один из двух аргументов в зависимости от результата вычисления выражения.

Аргумент	*Назначение*
условие	Выражение, значение которого нужно вычислить
еслиИстина	Значение или выражение, возвращаемые, если значением выражения является "Истина" (1)
еслиЛожь	Значение или выражение, возвращаемые, если значением выражения является "Ложь" (0)

Функции категории «Текстовые»:

Format("выражение"; инструкция форматир-ия) – возвращает строку, содержащую выражение, отформатированное согласно инструкциям форматирования.

Для выражений даты/времени можно применять следующие символы в инструкции форматирования:

Символ	*Описание*
c	Полный формат даты
dd	День месяца (от 1 до 30)
ddd	Первые две буквы названия дня недели (от **Пн** до **Вс**)
dddd	Полное название дня недели (от **Понедельник** до **Воскресенье**)
w	День недели (от 1 до 7)
ww	Неделя года (от 1 до 53)
mm	Месяц года (от 1 до 12)
mmm	Первые три буквы названия месяца (от **Янв** до **Дек**)
mmmm	Полное название месяца (от **Январь** до **Декабрь**)
q	Квартал года (от 1 до 4)
y	День года (от 1 до 366)
yy	Последние две цифры года (от 01 до 99)
yyyy	Возвращает 4-ёх-значное значение года (от 100 до 9999)

Глава 8.

Создание и работа с БД посредством графического интерфейса СУБД.

Технологии работы с базой данных в MS Access.

§ 8.3. Проектирование запросов в MsA

Итоговые запросы

– это запросы, которые выполняют вычисления в группах записей.

Для создания итогового запроса в окне *Конструктора* запросов, вызывается команда

п.м. Вид → Групповые операции

(или нажимается кнопка ***Групповые операции*** на панели инструментов).

После этого в Бланке запроса появится новая строка ***Групповые операции***, в которой для соответствующего поля указывается тип выполняемого вычисления из раскрывающегося списка установок групповых операций.

Установки групповых операций:

Групповая установка	***Результат***
Группировка	Определяет группы записей, для которых рассчитываются итоговые значения.
Выражение	Определяет в запросе вычисляемое поле. Создание вычисляемого поля позволяет использовать в выражении несколько функций.
Условие	Задает условия отбора для поля, не используемого для определения группы. После выбора параметра «Условие» поле делается скрытым (снимается флажок в строке «Вывод на экран»).

Групповые функции:

Групповая функция	Результат
Sum	Сумма значений поля
Avg	Среднее значение поля
Min	Минимальное значение поля
Max	Максимальное значение поля
Count	Число значений в поле (не считая пустые)
StDev	Среднеквадратичное отклонение значений поля
Var	Дисперсия значений поля
First	Возвращают значение поля из первой записи результирующего набора запроса
Last	Возвращают значение поля из последней записи результирующего набора запроса

Замечание: При вычислении статистических функций не учитываются записи, содержащие пустые (**Null**) значения.

Например, функция **Count** возвращает количество всех непустых полей (т. е. полей, не содержащих значения **Null**).

Виды итоговых запросов:

I. Обобщающие запросы для одной группы записей

При использовании опции группировка записи группируются на основе одинаковых значений поля, где указана данная опция и MsA в этом случае выполняет вычисления отдельно для каждой группы.

Пример1: *Подсчитать количество перевозок и общий вес груза в каждый пункт назначения.*

Поле:	***Пункт назначения***	***Пункт назначения***	***Вес груза***
Групповая операция:	**Группировка**	**COUNT**	**SUM**

Пример2: *Подсчитать количество автомобилей разных типов в парке машин.*

Поле:	***Тип автомобиля***	***Тип автомобиля***	
Групповая операция:	**Группировка**	**COUNT**	

II. Обобщающие запросы для нескольких групп записей

Можно произвести расчеты над сгруппированными данными из нескольких полей и из нескольких таблиц.

Замечание: Последовательность размещения полей в бланке запроса определяет порядок вложения групп: в первую очередь группировка будет выполнена по крайнему левому полю.

Пример1: *Подсчитать для каждого типа машины за каждый день общее количество ездок и максимальную величину пробега (км).*

Поле:	***Тип автомобиля***	***Дата перевозки***	***Число ездок***	***Пробег***
Групповая операция:	**Группировка**	**Группировка**	**SUM**	**MAX**

III. Обобщающие запросы по всем записям

– запросы с участием групповых функций без использования опции группировка.

Замечание: *результат выборки – одна запись!*

Пример1: *Вывести статистику по полю «Вес груза» (максимальное, минимальное, среднее и общее значения).*

Поле:	***Макс-ый вес: Вес груза***	***Миним-ый вес: Вес груза***	***Средний вес: Вес груза***	***Общий вес: Вес груза***
Групповая операция:	**MAX**	**MIN**	**AVG**	**SUM**

Пример2: *Вывести общую стоимость всех перевозок.*

Поле:	***Общая стоимость перевозок: Стоимость перевозок***
Групповая операция:	**SUM**

IV. Группировка с использованием критериев

– группировка записей в обобщающем запросе с установкой ограничений на число обработанных или отображенных записей.

IV.a. применение критериев отбора для поля, обработанного групповой операцией

– ограничение числа сгруппированных записей.

Пример1: *Определить кол-во перевозок за каждый день не позднее 01.01.2019 г.*

Поле:	***Дата перевозки***	***Кол-во перевозок: Дата перевозки***
Групповая операция:	**Группировка**	**COUNT**
Условие отбора:	**< #01.01.2019#**	

Пример2: *Определить кол-во водителей, имеющих первую категорию.*

Поле:	***Категория***	***Кол-во водителей: Категория***
Групповая операция:	**Группировка**	**COUNT**
Условие отбора:	**“первая”**	

IV.b. применение критериев отбора для поля после его обработки групповой функцией

Пример1: *Определить для каждой машины общую стоимость топлива и вывести записи, для которых значения общей стоимости превышает N руб.*

Поле:	***Номер машины***	***Стоимость топлива***
Групповая операция:	**Группировка**	**SUM**
Условие отбора:		**> N**

Пример2: *Определить кол-во перевозок за каждый месяц определенного года, значение которого водится с клавиатуры.*

Поле:	***Номер перевозки***	***Месяц: Format([Перевозки]![Дата перевозки];"mmmm")***	***Год: Year([Перевозки]![Дата перевозки])***
Групповая операция:	**COUNT**	**Группировка**	**Группировка**
Условие отбора:			**[Введи год]**

IV.c. применение критериев отбора для поля перед его обработкой групповой операцией

Пример1: *Определить общее количество ездок в каждый пункт назначения, значения числа ездок в которые не менее 4-ех.*

Поле:	*Пункт назначения*	*Число ездок*	*Число ездок*
Групповая операция:	Группировка	SUM	Условие
Условие отбора:			>= 4

IV.d. применение критериев отбора для поля, не обработанного групповой операцией

Пример1: *Определить количество и максимальную грузоподъемность машин, возраст которых более 10 лет.*

Поле:	***Номер машины***	***Грузоподъемность***	***Возраст машины***
Групповая операция:	**COUNT**	**MAX**	**Условие**
Условие отбора:			**>10**

V. Запросы с использованием опции ВЫРАЖЕНИЕ

– для создания вычисляемого поля с помощью выражения, включающего групповые функции.

Пример1: *Определить для каждой машины разницу между максимальным и минимальным значением числа ездок.*

Поле:	***Номер машины***	***Размах числа ездок: Max([Запрос_Пробег]![Число ездок])-Min([Запрос_Пробег]![Число ездок])***
Групповая операция:	**Группировка**	**Выражение**

Пример2: *Определить кол-во перевозок за каждый день заданного с клавиатуры квартала определенного года.*

Поле:	*Дата перевозки*	*Кол-во перевозок: Дата перевозки*	*Квартал: DatePart("q"; [Перевозки]![Дата перевозки])*	*Год: Year([Перевозки]![Дата перевозки])*
Групповая операция:	**Группировка**	**COUNT**	**Выражение**	**Выражение**
Вывод на экран	☑	☑	☑	☑
Условие отбора:			**[Введи квартал]**	**2019**

VI. Запросы с участием итоговых запросов

Пример1: *Вывести информацию и перевозках, вес груза которых больше среднего значения по данному полю.*

Поле:	***Перевозки****	***Вес груза (кг)***
Вывод на экран	☑	☐
Условие отбора:		**>[Лекция 3_1]![Средний вес]**

Замечание: *Таблица «Перевозки» и запрос «Лекция 3_1» располагаются в окне «Схема данных» без связи!!!!*

Перекрестные запросы

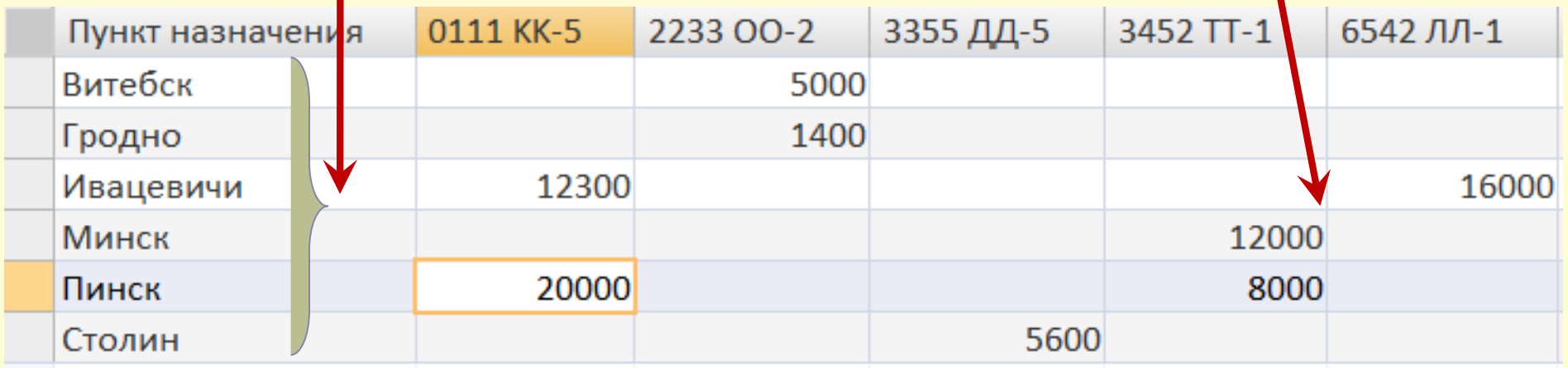
– обеспечивают создание результирующей таблицы на основе расчетов, полученных при анализе группы таблиц.

С помощью данного вида запроса можно сгруппировать большой объем информации и представить его в удобном для восприятия виде.

В перекрестном запросе подсчитываются статистические расчеты (средние величины, суммы и т.д.) по значению одного поля таблицы, после чего результаты вычислений группируются в виде таблицы **по двум наборам данных**: один из которых определяет заголовки столбцов (т.е. находится в верхней строке), а другой – заголовки строк (т.е. находится в левом столбце таблицы).

Т.О., данные сгруппированы как по горизонтали, так и по вертикали.

Пример1: *Подсчитать общий вес груза, перевезенного на каждой машине в каждый пункт назначения.*



Пункт назначения	0111 КК-5	2233 ОО-2	3355 ДД-5	3452 ТТ-1	6542 ЛЛ-1
Витебск		5000			
Гродно		1400			
Ивацевичи	12300				16000
Минск				12000	
Пинск	20000			8000	
Столин			5600		

СПОСОБЫ проектирования перекрестных запросов:

- *с помощью **Мастера**;*
- *в режиме **Конструктора**.*

Последовательность действий при создании перекрестного запроса (*Пример1*) с помощью **Мастера**:

в версии ACCESS 2003: находясь в окне БД выбрать вкладку **Запросы**; нажать кнопку **Создать**;

в версии ACCESS 2007/2010: выбрать пиктограмму  Мастер запросов на вкладке ленты **Создать** в группе инструментов **Другие** (или в группе **Запросы**);

- в появившемся диалоговом окне «Новый запрос» выбрать опцию ***Перекрестный запрос*** и нажать **ОК**;

- в появившемся окне «Создание перекрестных таблиц» выбрать из списка таблицу-источник (таблица «*Перевозки*»). Нажать **Далее**.
- на втором шаге Мастера выбрать с помощью кнопки  поле, значения которого будут использованы в качестве ***заголовков строк*** (поле «*Пункт назначения*»). Нажать **Далее**.
- на третьем шаге Мастера выбрать поле, значения которого будут использованы в качестве ***заголовков столбцов*** (поле «*Номер машины*»). Нажать **Далее**.

- на следующем шаге Мастера выбрать сначала поле, по которому будут производиться вычисления на пересечении строк и столбцов (поле «*Вес груза*»), а затем необходимую операцию (Сумма). На вопрос "Вычислить итоговое значение для каждой строки?" ответить «Нет". Нажать **Далее**.
- на последнем шаге Мастера ввести **имя** запроса и выбрать один из дальнейших вариантов действий: либо перейти в режим таблицы для просмотра результата, либо перейти в режим Конструктора для возможного изменения структуры запроса. Нажать **Готово**.

Порядок создания перекрестного запроса в режиме **Конструктора** (*Пример1*) :

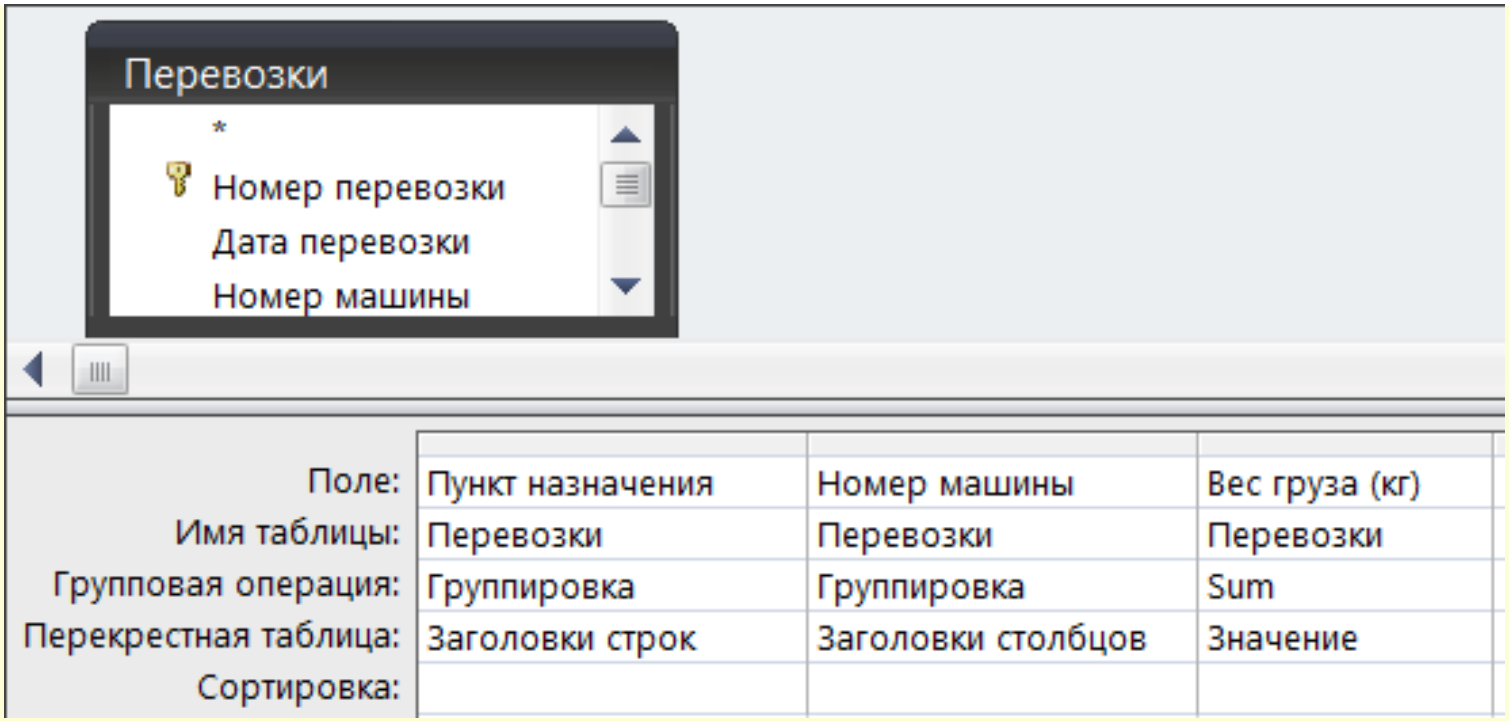
- сначала создать обычный запрос-выборку (в окне БД на вкладке ***Запросы*** команда – Создать запрос в режиме *Конструктора*);
- добавить таблицу-источник в окно схемы данных бланка Конструктора;
- далее необходимо преобразовать тип данного запроса на **Перекрестный**:

в версии ACCESS 2003: находясь в окне Конструктора из пункта меню ***Запрос*** выбрать команду ***Перекрестный***;

в версии ACCESS 2007/2010: выбрать пиктограмму  Перекрестный на вкладке ленты **Конструктор** в группе инструментов **Тип запроса;**

- для полей, значения которых будут заголовками строк, выбрать в строке Перекрестная таблица опцию ***Заголовки строк*** и оставить в строке Групповая операция значение Группировка;
- для полей, значения которых будут представлены в запросе в качестве заголовков столбцов, выбрать в строке Перекрестная таблица опцию ***Заголовки столбцов*** и оставить в строке Групповая операция значение Группировка;

➢для расчетных полей, значения которых будут находиться на пересечении строк и столбцов, выбрать в строке Перекрестная таблица опцию ***Значение***, а в строке Групповая операция из списка выбрать нужную статистическую функцию.



Поле:	Пункт назначения	Номер машины	Вес груза (кг)
Имя таблицы:	Перевозки	Перевозки	Перевозки
Групповая операция:	Группировка	Группировка	Sum
Перекрестная таблица:	Заголовки строк	Заголовки столбцов	Значение
Сортировка:			

Замечание: Опции «Заголовки столбцов» и «Значение» указываются только ОДИН раз.

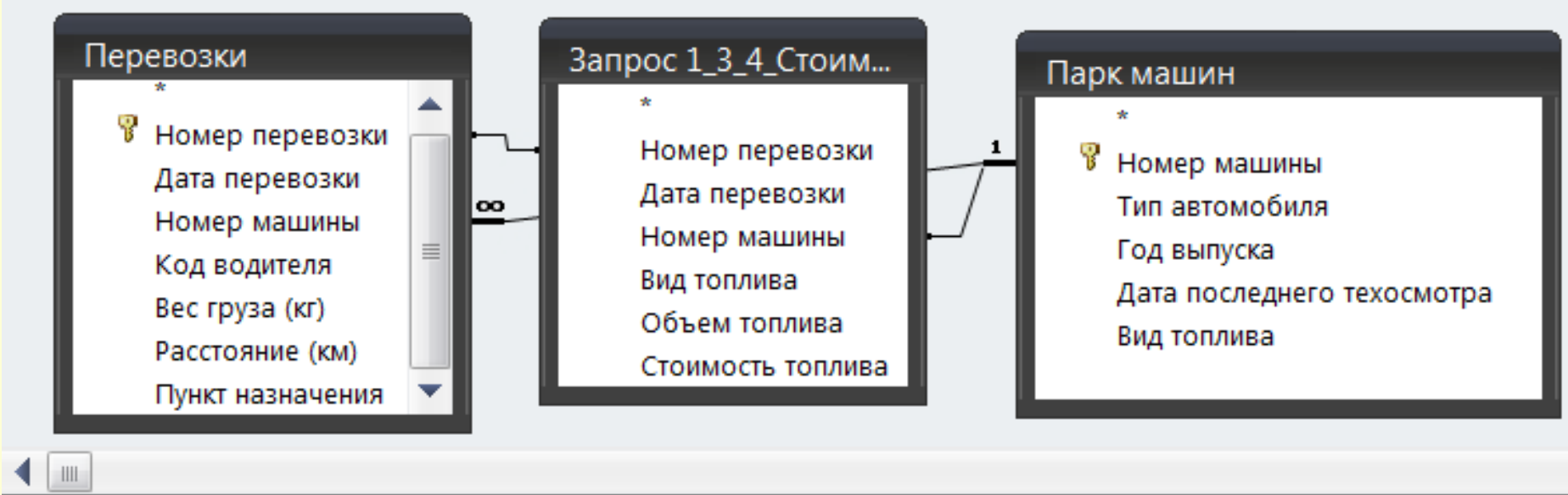
Возможно дополнительное добавление опции «Заголовки строк». При этом отображаемая информация может быть:

1. уточнена (Перекрестные запросы с заголовками строк из разных таблиц)

или

2. подведены **ИТОГИ** по строкам (Перекрестные запросы с итоговым столбцом).

Пример2 (Перекрестные запросы с заголовками строк из разных таблиц): Вывести общую стоимость топлива, затраченного каждым типом автомобиля с разными видами топлива в каждый пункт назначения.

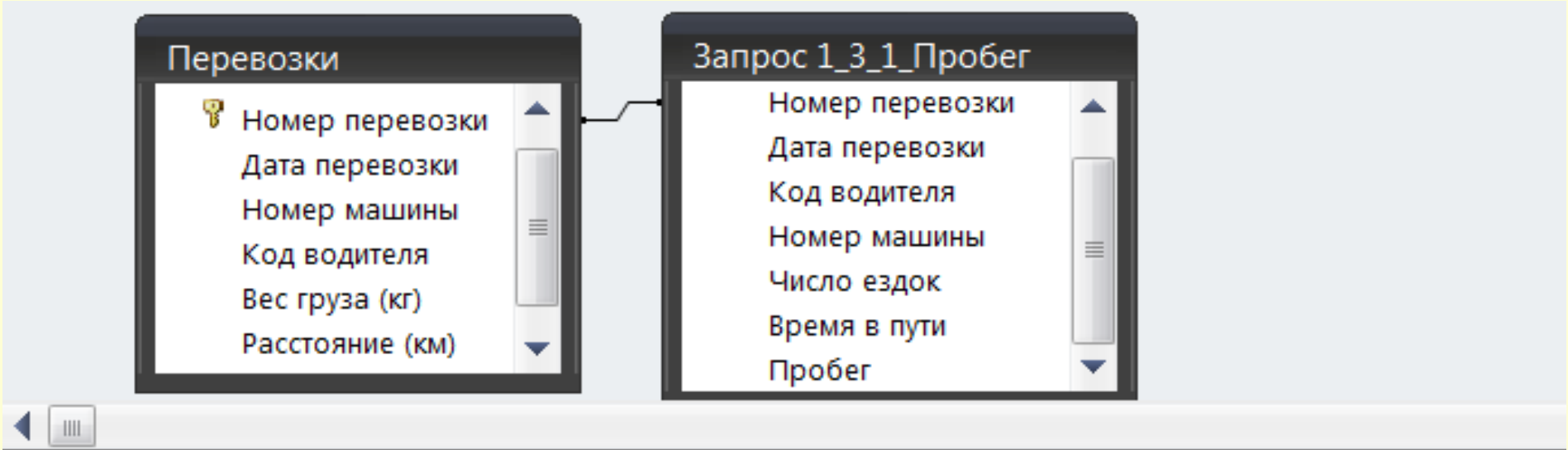


Поле:	Тип автомобиля	Вид топлива	Пункт назначения	Стоимость топлива
Имя таблицы:	Парк машин	Парк машин	Перевозки	Запрос 1_3_4_Стоимость топлива
Групповая операция:	Группировка	Группировка	Группировка	Sum
Перекрестная таблица:	Заголовки строк	Заголовки строк	Заголовки столбцов	Значение

Результат запроса:

Тип автомобиля	Вид топлива	Витебск	Гродно	Ивацевичи	Минск	Пинск	Столин
Газ	Бензин А92			1167950		1863750	
Газ	Бензин А95			1530000			
Зил	Бензин А80						594000
Зил	Газ				2851200	1056000	
Зил	Диз.топливо	2860000	1100000				
Камаз	Диз.топливо		1218000		3900000	3150000	750000
Маз	Биотопливо						552000

Пример4 (Перекрестный запрос с критериями отбора): Вывести за каждый день для каждой машины общее число ездок. В итоговом столбце отобразить максимальное значение пробега за каждый день.



Поле:	Дата перевозки	Номер машины	Число ездок	Пробег
Имя таблицы:	Перевозки	Перевозки	Запрос 1_3_1_Пробег	Запрос 1_3_1_Пробег
Групповая операция:	Группировка	Группировка	Sum	Max
Перекрестная таблица:	Заголовки строк	Заголовки столбцов	Значение	Заголовки строк

Результат запроса:

Дата перевозки	Max-Пробег	0111 КК	2233 ОО	3355 ДД	3452 ТТ	6542 ЛЛ	6654 ДЛ	7821 АА	7896 ГГ
01.01.2019	300	1							
02.02.2019	1200				3				
02.03.2019	1600							2	
02.11.2019	812								1
03.11.2019	2600		2				1		
05.11.2019	1200					4			
06.11.2019	640	2							1
10.11.2019	1500	5							
11.11.2019	1000		1						
12.11.2019	3240				4				
13.11.2019	600			1					
14.11.2019	2100								7
15.11.2019	1000							1	

Замечание1: Заголовок итогового столбца возможно переименовывать в бланке QBE:

Максимальный пробег: Пробег
Запрос 1_3_1_Пробег
Max
Заголовки строк

Замечание2: Сам итоговый столбец в режиме выполнения запроса возможно перемещать в конец результирующей таблицы, удерживая его за заголовок столбца.

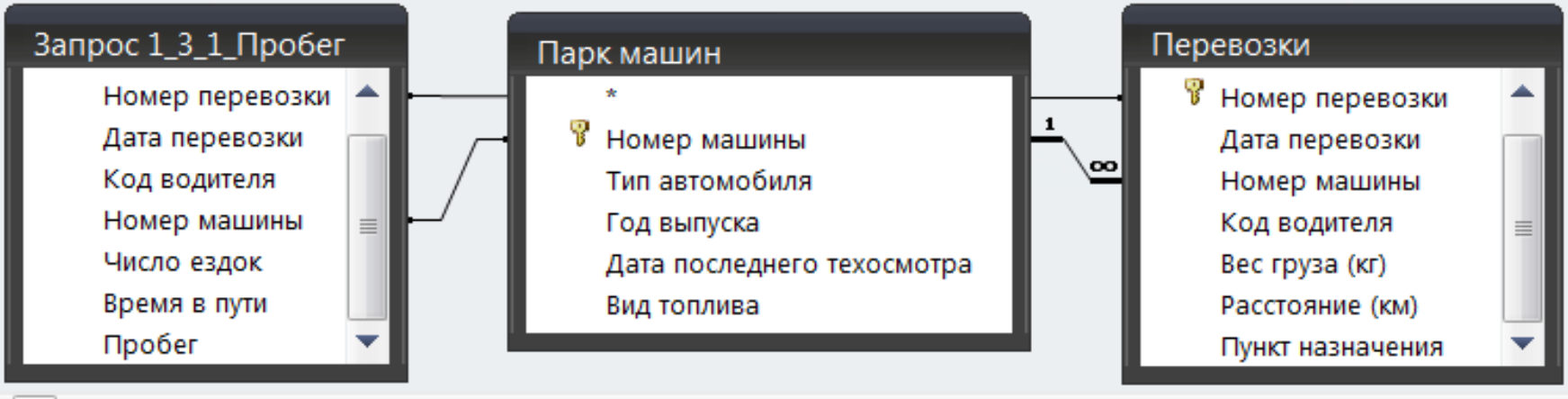
Дата перевозки	0111 КК	2233 ОО	3355 ДД	3452 ТТ	6542 ЛЛ	6654 ДЛ	7821 АА	7896 ГГ	Максимальный пробег
01.01.2019	1								300
02.02.2019				3					1200
02.03.2019							2		1600
02.11.2019								1	812
03.11.2019		2				1			2600
05.11.2019					4				1200
06.11.2019	2							1	640
10.11.2019	5								1500
11.11.2019		1							1000
12.11.2019				4					3240
13.11.2019			1						600
14.11.2019								7	2100
15.11.2019							1		1000

Условия отбора в перекрестном запросе

Условия могут устанавливаться:

- для любого ***нового*** поля;
- для поля ***Заголовки строк***;
- для поля ***Заголовки столбцов***.

Пример5 (Перекрестный запрос с критериями отбора для нового поля): Вывести максимальное время в пути в каждый пункт назначения каждой машиной, вид топлива которых «Диз.топливо». В итоговом столбце отобразить общий пробег каждой машины.

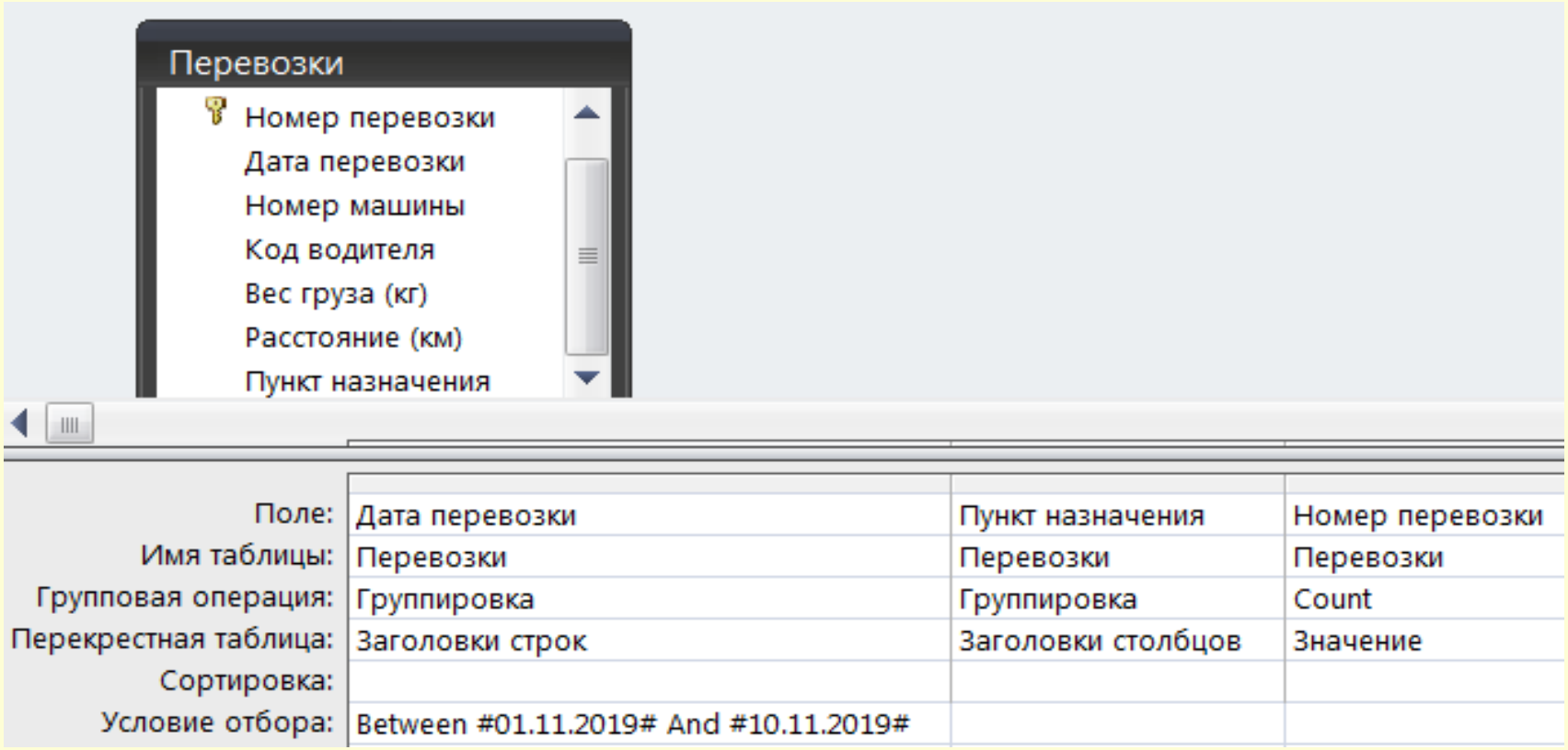


Поле:	Пункт назначения	Номер машины	Время в пути	Общий пробег: Пробег	Вид топлива
Имя таблицы:	Перевозки	Перевозки	Запрос 1_3_1_Пробег	Запрос 1_3_1_Пробег	Парк машин
Групповая операция:	Группировка	Группировка	Max	Sum	Условие
Перекрестная таблица:	Заголовки столбцов	Заголовки строк	Значение	Заголовки строк	
Сортировка:					
Условие отбора:					"Диз.топливо"

Результат запроса:

Номер машины	Витебск	Гродно	Минск	Пинск	Столин	Общий пробег
2233 ОО-2	50	19,23				3600
7821 АА-1			26,67			2600
7896 ГГ-2		13,53		35	8,33	3412

Пример6 (Перекрестный запрос с критериями отбора для поля «Заголовки строк»): Вывести количество перевозок в каждый пункт назначения в разрезе дат, которые попадают в интервал с ДАТА1 по ДАТА2.



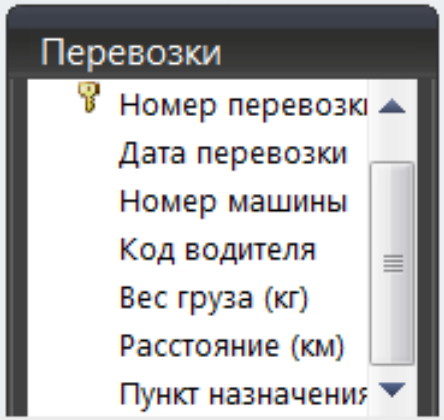
Поле:	Дата перевозки	Пункт назначения	Номер перевозки
Имя таблицы:	Перевозки	Перевозки	Перевозки
Групповая операция:	Группировка	Группировка	Count
Перекрестная таблица:	Заголовки строк	Заголовки столбцов	Значение
Сортировка:			
Условие отбора:	Between #01.11.2019# And #10.11.2019#		

Результат запроса:

	Неделя	Витебск	Гродно	Ивацевичи	Минск	Пинск	Столин
	44		6000				
	45	5000		23800			7500
	46				15500	56400	5600

Использование встроенных функций и условий отбора в перекрестном запросе

Пример7: Вывести за каждую неделю ноября 2019 года общий вес груза, перевезенного в каждый пункт назначения.



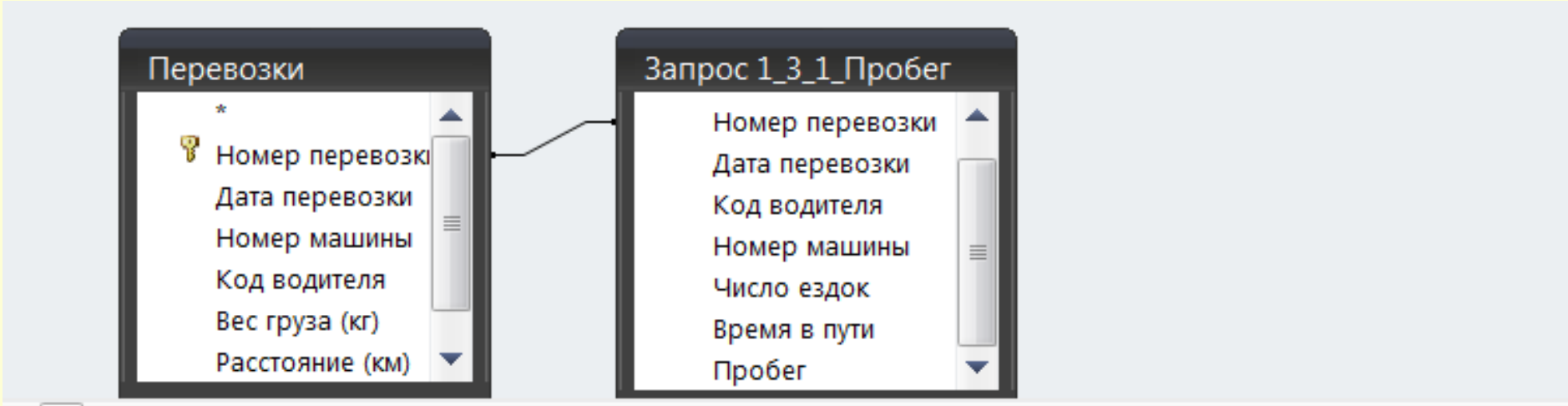
Поле:	Пункт назначения	Неделя: DatePart("ww";[Перевозки]![Дата перевозки])	Вес груза (кг)
Имя таблицы:	Перевозки		Перевозки
Групповая операция:	Группировка	Группировка	Sum
Перекрестная таблица:	Заголовки столбцов	Заголовки строк	Значение
Сортировка:			
Условие отбора:		Month([Дата перевозки])=11 And Year([Дата перевозки])=2019	

Результат запроса:

Дата перевозки	Витебск	Гродно	Ивацевичи	Пинск	Столин
02.11.2019		1			
03.11.2019	1				1
05.11.2019			1		
06.11.2019			1		1
10.11.2019				2	

Параметры в перекрестном запросе

Пример8: Вывести за каждый день недели заданного с клавиатуры месяца 2019 года общее число ездок каждой машиной.



Поле:	Номер машины	День недели: Format(	Число ездок	Месяц: Month([Перевозк	Год: Year([П
Имя таблицы:	Перевозки		Запрос 1_3_1_Пробег		
Групповая операция:	Группировка	Группировка	Sum	Условие	Условие
Перекрестная таблица:	Заголовки строк	Заголовки столбцов	Значение		
Сортировка:					
Условие отбора:				[Введи НОМЕР месяца]	2019

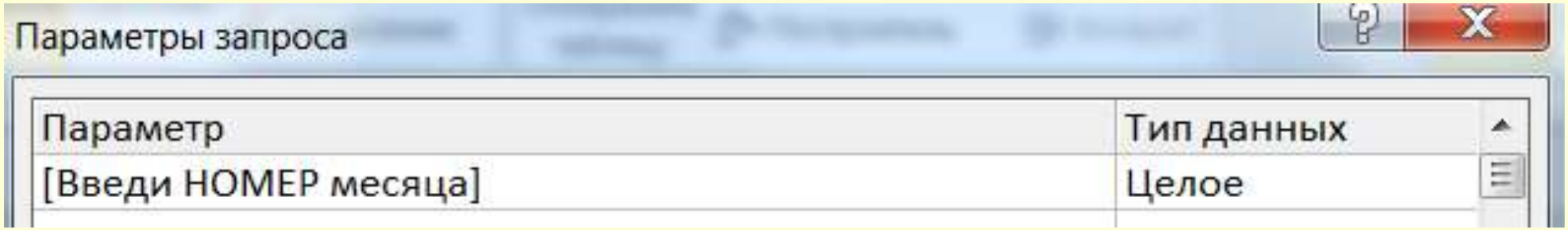
Вычисляемые поля:

День недели: Format([Перевозки]![Дата перевозки];"dddd")
Месяц: Month([Перевозки]![Дата перевозки])
Год: Year([Перевозки]![Дата перевозки])

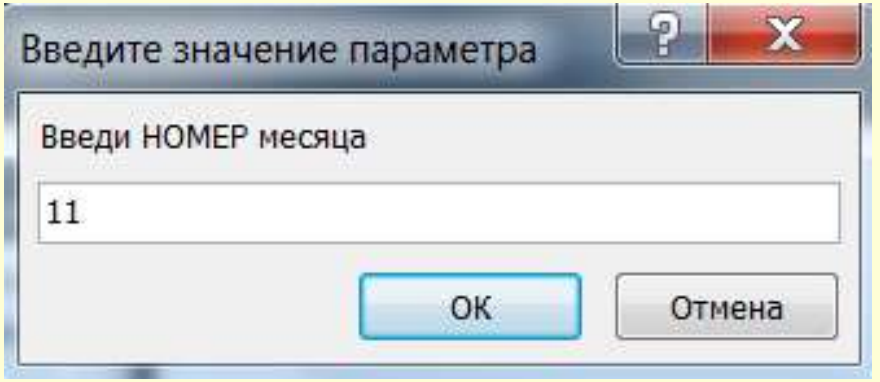
Замечание: для определения **типа данных** параметра необходимо его добавить в окно «Параметры запроса», которое открывается следующим образом:

в версии ACCESS 2003: выбрать команду ***Параметры*** из пункта меню ***Запрос***.

в версии ACCESS 2007/2010: выбрать пиктограмму ***Параметры*** на вкладке ленты **Создание** в группе инструментов **Показать или скрыть** (или в группе **Запросы**).



Выполнение запроса:



Результат запроса:

Номер машины	воскресенье	вторник	пятница	среда	суббота	четверг
0111 КК-5	5			2		
2233 ОО-2	3					
3355 ДД-5				1		
3452 ТТ-1		4				
6542 ЛЛ-1		4				
6654 ДЛ-1	1					

Глава 8.

Создание и работа с БД посредством графического интерфейса СУБД.

Технологии работы с базой данных в MS Access.

§ 8.3. Проектирование запросов в MsA

Технологии создания активных запросов

Активные запросы (или запросы-действия) выполняют определенные действия над извлеченными данными и вносят изменения непосредственно в саму БД.

ЗАМЕЧАНИЯ:

1. Визуально эти запросы можно отличить от других по восклицательному знаку (!), расположенному слева от названия запроса в окне БД.

2. Результат выполнения активного запроса следует искать на вкладке ***Таблицы*** окна БД, т.к. такие запросы связаны с корректированием или созданием таблиц.

3. Действия запросов, вносящих изменения, **необратимы**, поэтому желательно создавать резервные копии обрабатываемых таблиц.

Различают следующие запросы-действия:

- *запросы на создание таблицы*
- *запросы на обновление*
- *запросы на добавление*
- *запросы на удаление*

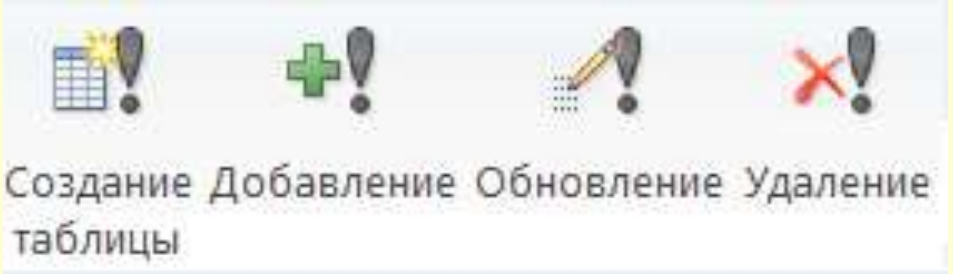
Последовательность действий при создании активного запроса:

1. создать запрос на выборку (в режиме *Конструктора*);

2. затем преобразовать его в активный запрос:

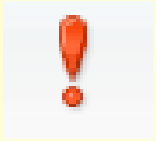
в версии ACCESS 2003: с помощью соответствующей команды в п.м. Запрос.

в версии ACCESS 2007/2010: на вкладке ленты **Конструктор** в группе инструментов **Тип запроса**.



3. перейти в **режим просмотра** 

результирующей таблицы и **просмотреть** записи, выбранные в запросе;

4. **выполнить** запрос: кнопка **Запуск** 

5. **проверить** внесенные изменения в режиме просмотра результирующей таблицы.

Запрос на создание таблицы

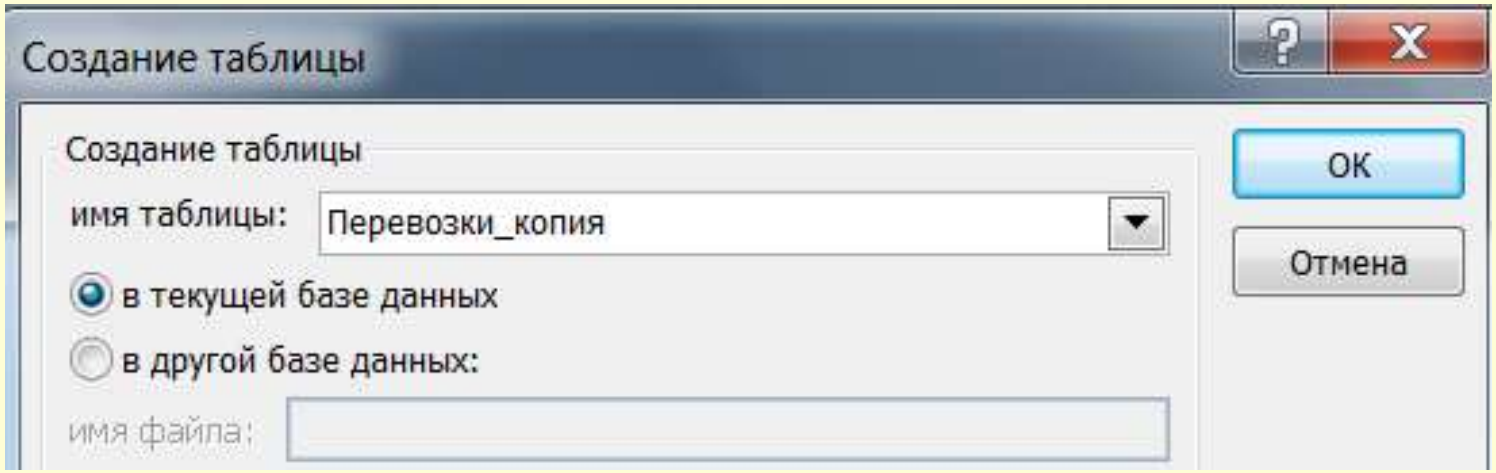
– позволяет создавать **новые** таблицы, основанные на результатах запроса. Все записи, выбранные запросом, копируются в новую таблицу БД, создаваемую запросом данного вида.

Замечание: Запросы на создание таблиц обычно используются для создания **резервных копий** и **архивов** данных или для **экспортирования** выбранных данных в файлы другого формата.

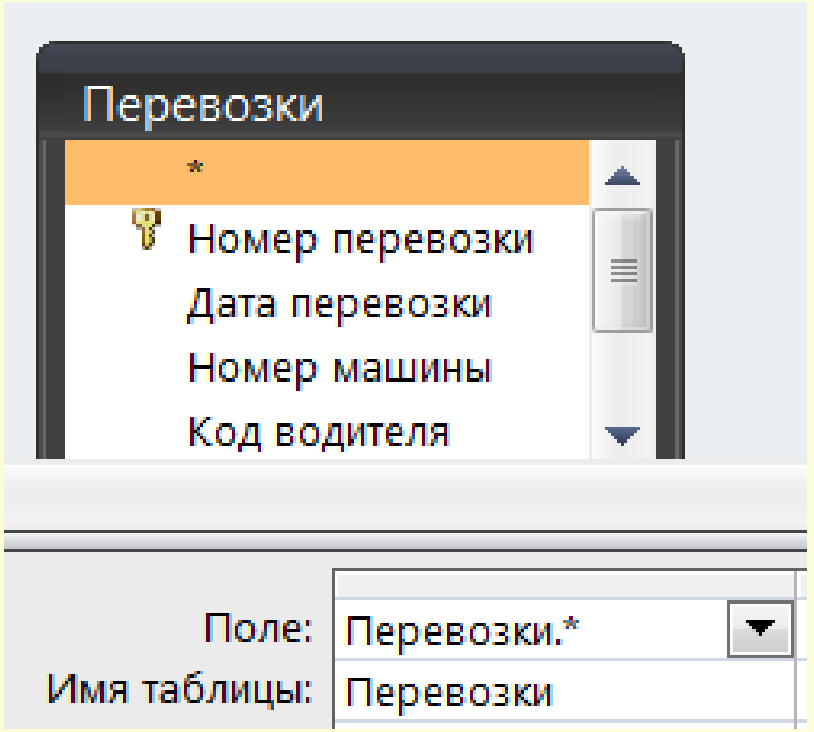
Пример1: Создать **<u>копию</u>** таблицы «Перевозки» с помощью активного запроса.

Порядок создания:

1. создается обычный запрос-выборка (источник данных – таблица «Перевозки»);
2. в окне *Конструктора* запроса вызывается команда ***п.м. Запрос → Создание таблицы*** (2003);
3. в появившемся окне «Создание таблицы» указывается имя создаваемой таблицы и БД, в которую ее следует поместить:

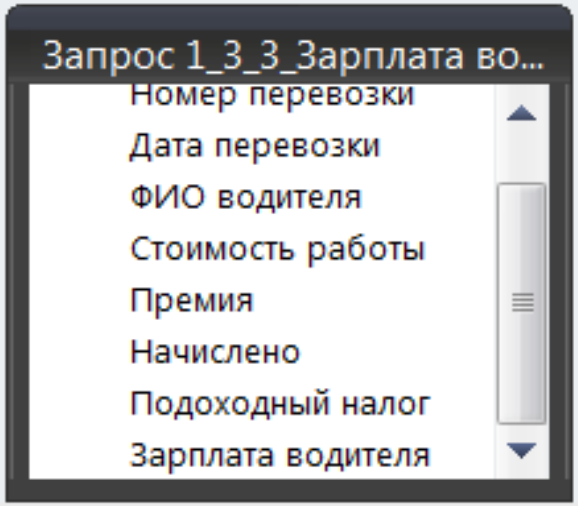


4. затем из списков полей в бланк запроса помещаются поля, которые должны быть в создаваемой таблице:



5. при необходимости задаются условия отбора записей:

Пример2: Создать таблицу «Ведомость зарплаты» за заданный месяц текущего года.



Вычисляемые поля:

Месяц: Month([Запрос 1_3_3_Зарплата водителей]![Дата перевозки])

Год: Year([Запрос 1_3_3_Зарплата водителей]![Дата перевозки])

Поле:	Запрос 1_3_3_Зарплата водителей.*	Месяц: Month([Запрос 1_3_3_	Год: Year([Запрос 1_3_3_
Имя таблицы:	Запрос 1_3_3_Зарплата водителей		
Сортировка:			
Вывод на экран:	☑	☐	☐
Условие отбора:		11	Year(Date())

Запрос на обновление

– корректирует все записи, удовлетворяющие определенному условию.

Такие запросы изменяют данные в существующих таблицах.

Замечание: Запросы на обновление применяются в тех случаях, когда нужно произвести однотипные изменения в каком-либо наборе данных.

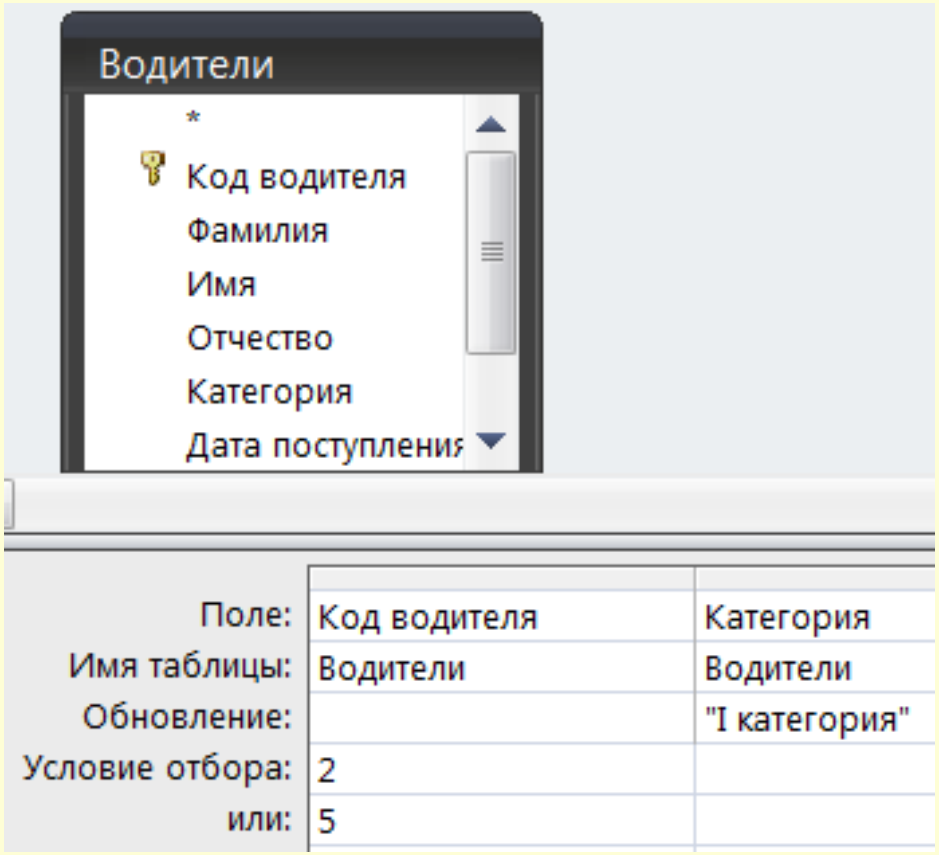
Пример3: Оформить изменение с 1-ой категории на 2-ую категорию у двух сотрудников.

Порядок создания:

1. создается обычный запрос-выборка (источник данных – таблица «Водители»);

2. в окне *Конструктора* запроса вводится команда ***п.м. Запрос → Обновление*** (2003), после чего в бланке QBE появляется новая строка **Обновление**;

3. затем для полей, подлежащих изменению, в строке Обновление задаются выражения, значения которых будут новыми значениями обновляемых полей.



Запрос на удаление

– позволяет удалить все записи, которые удовлетворяют условиям, определенным в запросе.

Замечание: запросы на удаление могут использоваться в двух случаях:

1. при удалении записей из одной таблицы (как правило, из подчиненной);

2. при удалении из нескольких взаимосвязанных таблиц.

Пример4 (удаление записей из подчинённой таблицы): Удалить из таблицы «Перевозки» записи за прошлый год.

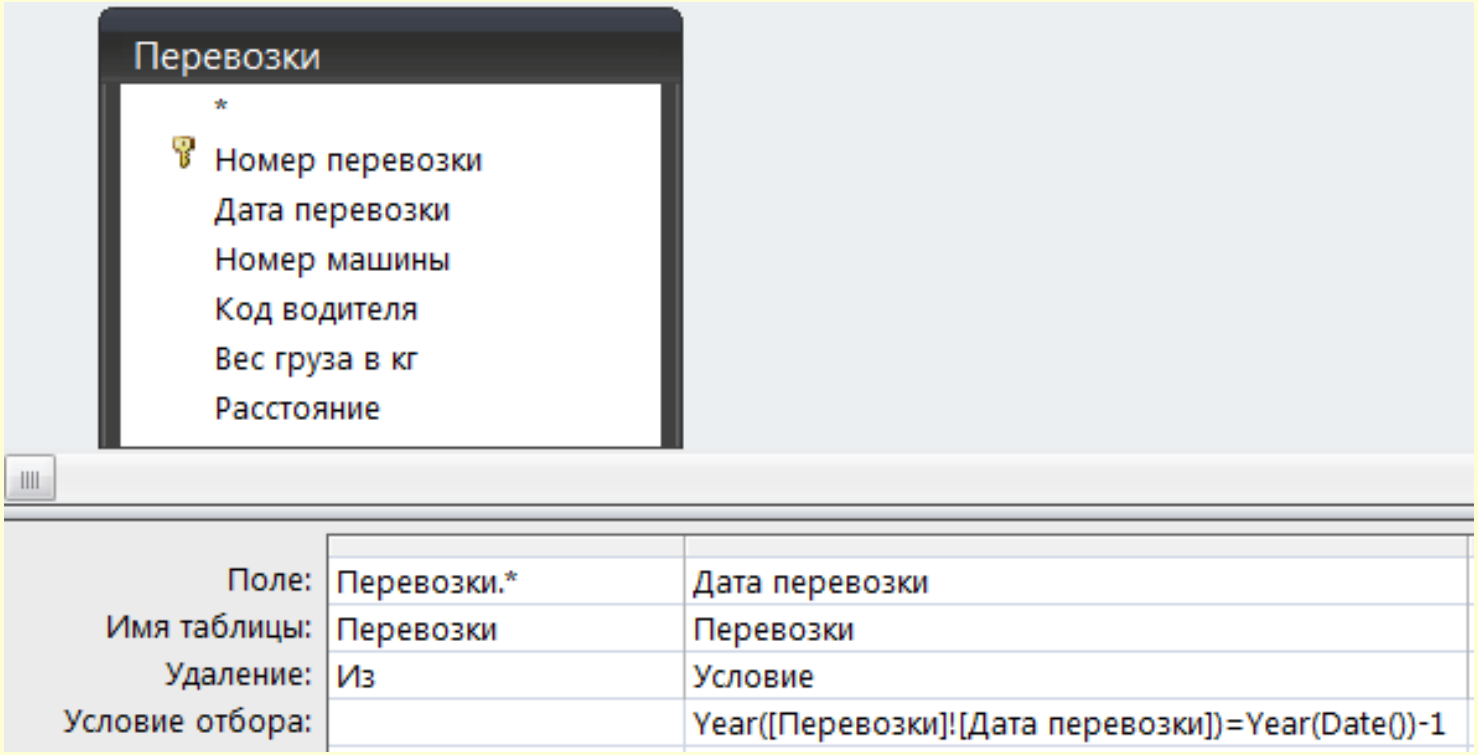
Порядок создания:

1. создается обычный запрос-выборка (источник данных – таблица «Перевозки»);

2. в окне *Конструктора* запроса вводится команда ***п.м. Запрос → Удаление*** (2003), после чего в бланке QBE появляется новая строка **Удаление**;

3. сначала в строку ***Поле*** бланка QBE помещается имя таблицы (с помощью перетаскивания символа звездочки (*) из списка полей таблицы), из которой будут удаляться записи: в строке ***Удаление*** для этого поля появится значение **ИЗ**;

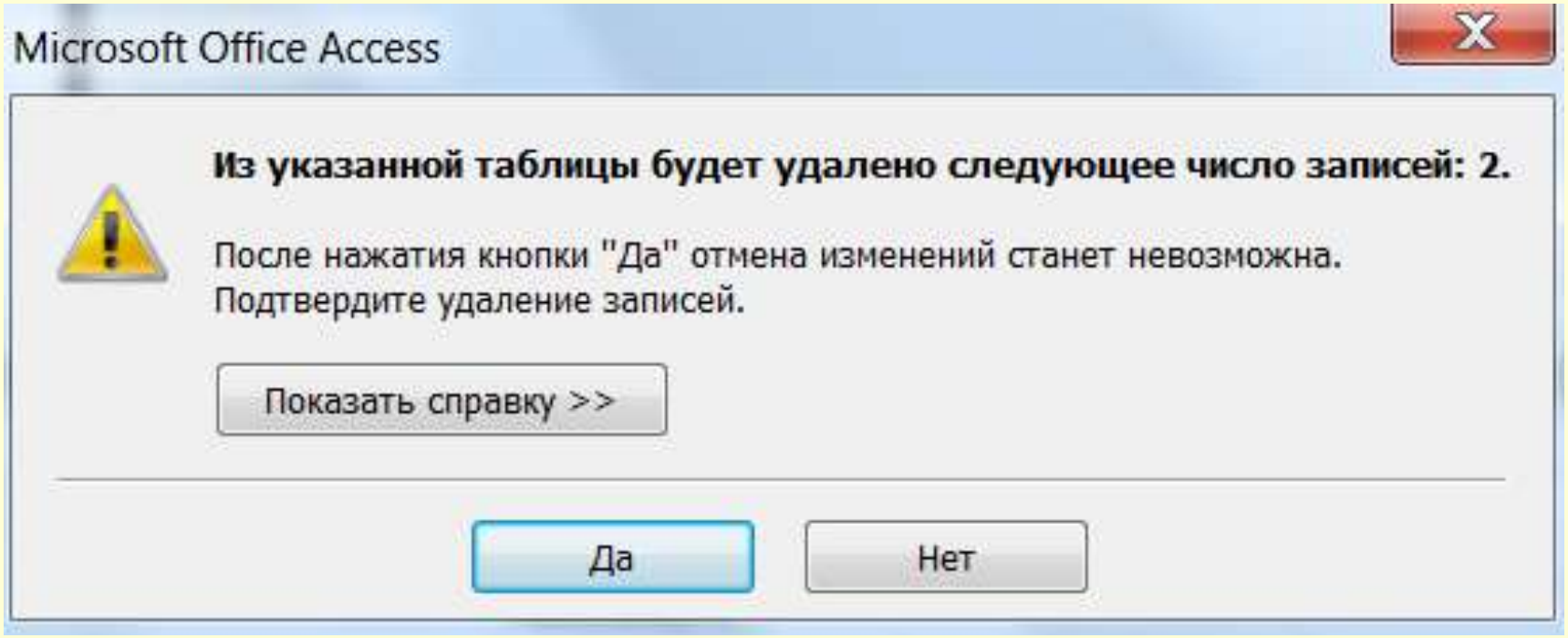
4. затем в строку ***Поле*** бланка QBE добавляются только те поля таблицы, по которым будут задаваться условия на удаление записей: в строке ***Удаление*** для этих полей появится значение **УСЛОВИЕ**.



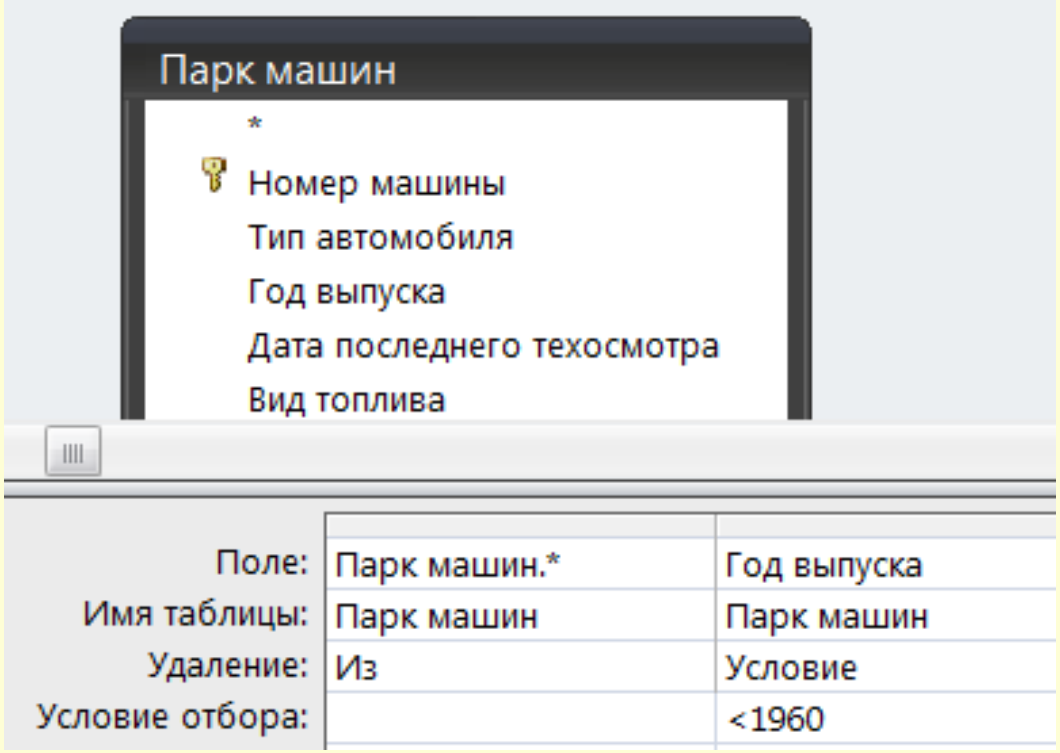
Предварительный просмотр в режиме таблицы:

Номер перевоз	Перевозки.Да	Номер машин	Код водите	Вес груза в к	Расстояние
1	01.01.2019	0111 КК-5	4	4500	15
2	02.02.2019	3452 ТТ-1	3	8000	200

Выполнение запроса:



Пример5 (удаление записей из взаимосвязанных таблиц): Удалить из таблицы «Парк машин» автомобили, год выпуска которых до 1960 г. в связи с их списанием.



Замечание_1: Если между таблицами установлена связь с обеспечением целостности данных и ***С каскадным удалением записей***, то создается запрос на удаление записей только из главной таблицы.

!!! В результате выполнения запроса будут удалены записи по условию из главной таблицы и связанные с ней записи из подчинённой таблицы.

Замечание_2: Если между таблицами установлена связь с обеспечением целостности данных, но ***БЕЗ каскадного удаления записей***, то прежде составляется запрос на удаление записей из подчиненной таблицы, а затем – из главной.

В этом случае порядок удаления записей из подчиненной таблицы:

1. в окне Конструктора запроса вводится команда п.м. Запрос → Удаление, после чего появляется в бланке QBE новая строка Удаление;

2. затем из списка полей подчиненной таблицы в бланк запроса переносится символ *, после чего в строке ***Удаление*** для этого поля отображается значение **Из**;

3. далее из списка полей главной таблицы переносятся поля, участвующие в условии отбора удаляемых записей, и для них в строке ***Удаление*** появляется значение **Условие**, что позволяет задать условия отбора удаляемых записей в строке ***Условие отбора***.

Замечание_3: Запросы на удаление часто используются для архивации записей. При этом запускается запрос на создание таблицы, который копирует выбранные записи в новую таблицу, и выполняется запрос на удаление, который удаляет записи, скопированные на предыдущем шаге из первой таблицы.

Запрос на добавление

– используется для добавления данных в какую-либо таблицу.

Такие запросы могут копировать данные в другие таблицы одной и той же БД или в таблицы других БД. Таблицы необязательно должны иметь идентичную структуру, но требуется, чтобы типы полей одной таблицы **соответствовали** типам полей другой таблицы.

Порядок создания:

1. в окне *Конструктора* запроса в качестве источника запроса указывается таблица, из которой добавляются записи в другую таблицу (записи таблицы-источника должны содержать такие же поля, что и пополняемая таблица БД);

2. вводится команда ***п.м. Запрос → Добавление;***

3. в появившемся окне «**Добавление**» следует указать имя пополняемой таблицы и БД, в которой она находится, после чего в бланке запроса появляется новая строка ***Добавление***;

4. затем следует перенести из списка полей таблицы-источника те поля, которые совпадают с полями пополняемой таблицы. Их имена Access автоматически укажет в строке ***Добавление*** как имена полей пополняемой таблицы.

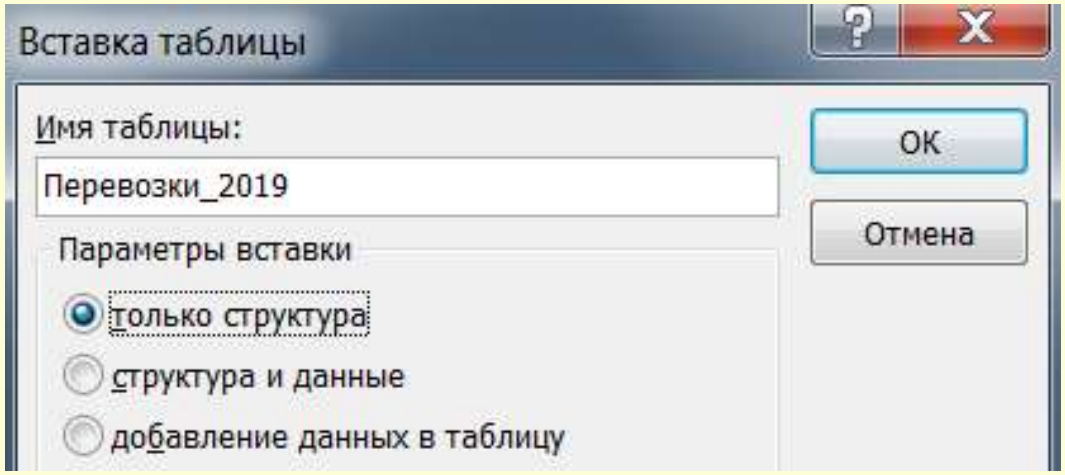
Пример6: Дополнить таблицу «Перевозки_2019» данными из таблицы «Перевозки» записями за прошлый год.

!!! Предварительно создать структуру таблицы «Перевозки_2019»,состоящую из таких же полей, как в таблице «Перевозки».

Порядок выполнения:

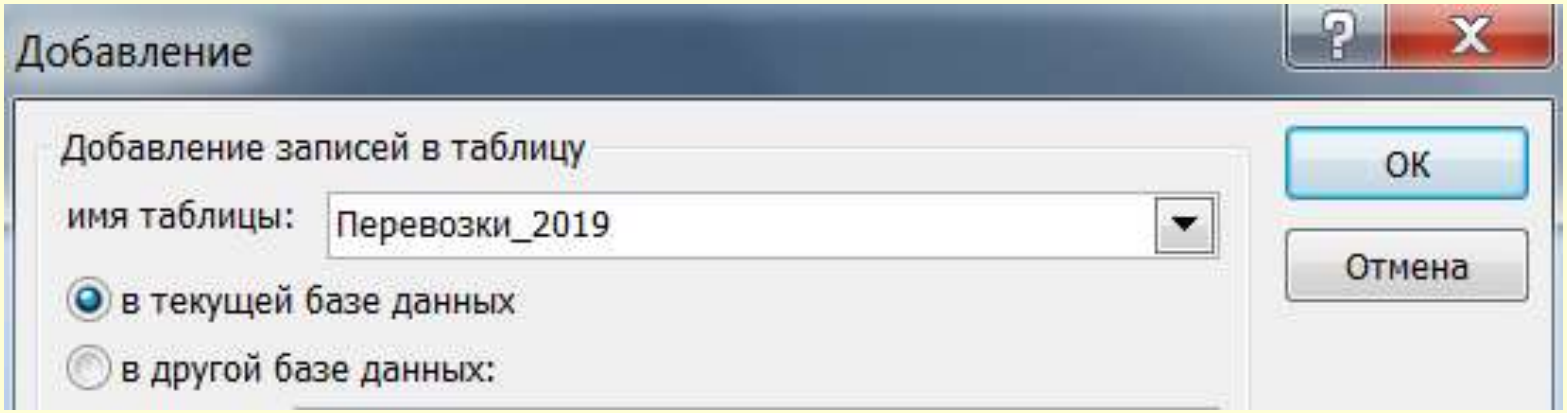
1. В окне БД скопировать в буфер обмена таблицу «Перевозки», затем вызвать команду ***Вставить*** и в появившемся окне «Вставка таблицы» задать имя новой таблице и выбрать переключатель

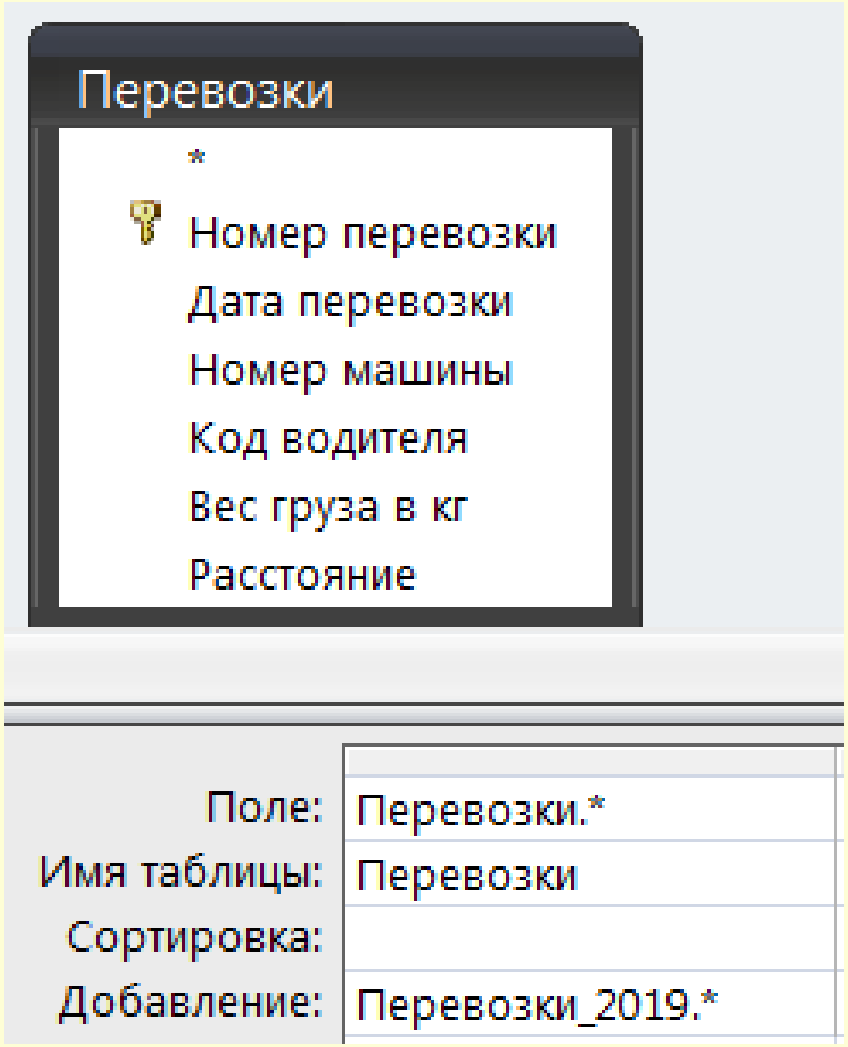
⊙ ***только структура***



2. в окне *Конструктора* запроса в качестве источника указывается табл. «Перевозки»;

3. вводится команда ***п.м. Запрос → Добавление:***




Перевозки
*
Номер перевозки
Дата перевозки
Номер машины
Код водителя
Вес груза в кг
Расстояние
Поле:
Перевозки.*
Имя таблицы:
Перевозки
Сортировка:
Добавление:
Перевозки_2019.*

Глава 8.

Создание и работа с БД посредством графического интерфейса СУБД.

Технологии работы с базой данных в MS Access.

§ 8.4. Проектирование форм в MsA

Определение и возможности форм

Понятие **«Форма»** в БД означает структурированное окно, экранное поле либо независимый элемент интерфейса с заранее установленными областями для ввода либо изменения информации; визуальный шаблон, который упорядочивает предоставляемые формой данные, что позволяет лучше их организовывать и просматривать.

Форма дает возможность вводить и просматривать информацию в БД.

Как объект БД форма используется:

- *для ввода данных в таблицу, их корректировки;*
- *в качестве специального окна диалога для выбора;*
- *в виде кнопочной формы для открытия других объектов БД.*

Источником данных для формы могут являться записи таблицы либо запроса.

В форме может быть представлена информация из нескольких связанных таблиц.

Представление данных в формах может быть различным: поля располагаются в виде столбца или таблицы.

Форма облегчает восприятие информации – это гибкий способ ее представления.

Инструментарий для форм ничем не отличается от табличного: данные в формах можно ***искать, заменять, сортировать, фильтровать***, словом, делать с ними то же самое, что и в таблицах.

Способы создания форм:

- **Автоформа** (полностью автоматизированное средство);
- с помощью **Мастера** (в режиме диалога);
- в режиме **Конструктора**;
- **Диаграмма** – создание формы с диаграммой.
- **Сводная таблица** – создание формы со сводной таблицей MS Excel.

Создание Автоформы

в версии ACCESS 2003:

1 способ: в окне БД на вкладке ***Таблицы*** выбрать таблицу, для которой создается форма и вызвать команду ***п. м. Вставка*** → ***Автоформа*** (или кнопку «Новый объект» на панели инструментов).

2 способ:

- в окне БД на вкладке ***Формы*** нажать кнопку **Создать**;
- в окне «Новая форма» задать режим –
  - Автоформа: в столбец,
  - Автоформа: ленточная,
  - Автоформа: табличная,
  - пр.;
- выбрать из раскрывающегося списка в качестве источника данных для формы таблицу или запрос.

в версии ACCESS 2007/2010:

находясь в окне БД в списке объектов ***Таблицы*** (в области переходов – левая часть окна приложения) выделить таблицу, для которой создаётся форма, и выбрать на вкладке ленты ***Создание*** в группе инструментов ***Формы*** пиктограмму «Формы»

Создание формы с помощью Мастера форм:

1) в версии ACCESS 2003: находясь в окне БД на вкладке ***Формы*** нажать кнопку **Создать**; в окне **Новая форма** задать режим **Мастер форм**;

1) в версии ACCESS 2007/2010: выбрать пиктограмму  ***Мастер форм*** на вкладке ленты **Создание** в группе инструментов **Формы** (может находится в дополнительном меню **Другие формы**);

2) следовать шагам *Мастера* в окне «Создание форм»:

на 1-м шаге:

- сначала выбрать из раскрывающегося списка в качестве источника данных для формы таблицу или запрос,
- затем выбрать все поля для формы посредством кнопки **>>** ,
- нажать кнопку *Далее*;

на 2-м шаге: выбрать внешний вид формы (например, «в один столбец»), нажать кнопку *Далее*;

на 3-м шаге: выбрать стиль оформления формы (например, «Стандартная»), нажать кнопку *Далее*;

на 4-м шаге: задать имя форме и выбрать один из вариантов дальнейших действий: либо открыть форму для просмотра данных, либо переключиться в режим *Конструктора* для дальнейшего её редактирования, нажать кнопку *Готово*.

Замечание: Просматривать записи в форме можно, используя кнопки внизу окна формы:

Создание формы в режиме Конструктора:

в версии ACCESS 2003: находясь в окне БД на вкладке ***Формы*** нажать кнопку **Создать**; в окне **Новая форма** задать режим **Конструктор**.

в версии ACCESS 2007/2010: выбрать пиктограмму  ***Конструктор форм*** на вкладке ленты **Создание** в группе инструментов **Формы**).

Процесс создания формы состоит:

- в размещении объектов в форме (текстовой информации; полей ввода; кнопок управления; рисунков или других графических изображений);
- определении свойств для размещенных объектов, а также связанных с ними событий и выполняемых действий;
- определении свойств самой формы как объекта (размера, возможности использования формы либо для просмотра данных, либо для их ввода и редактирования и пр.).

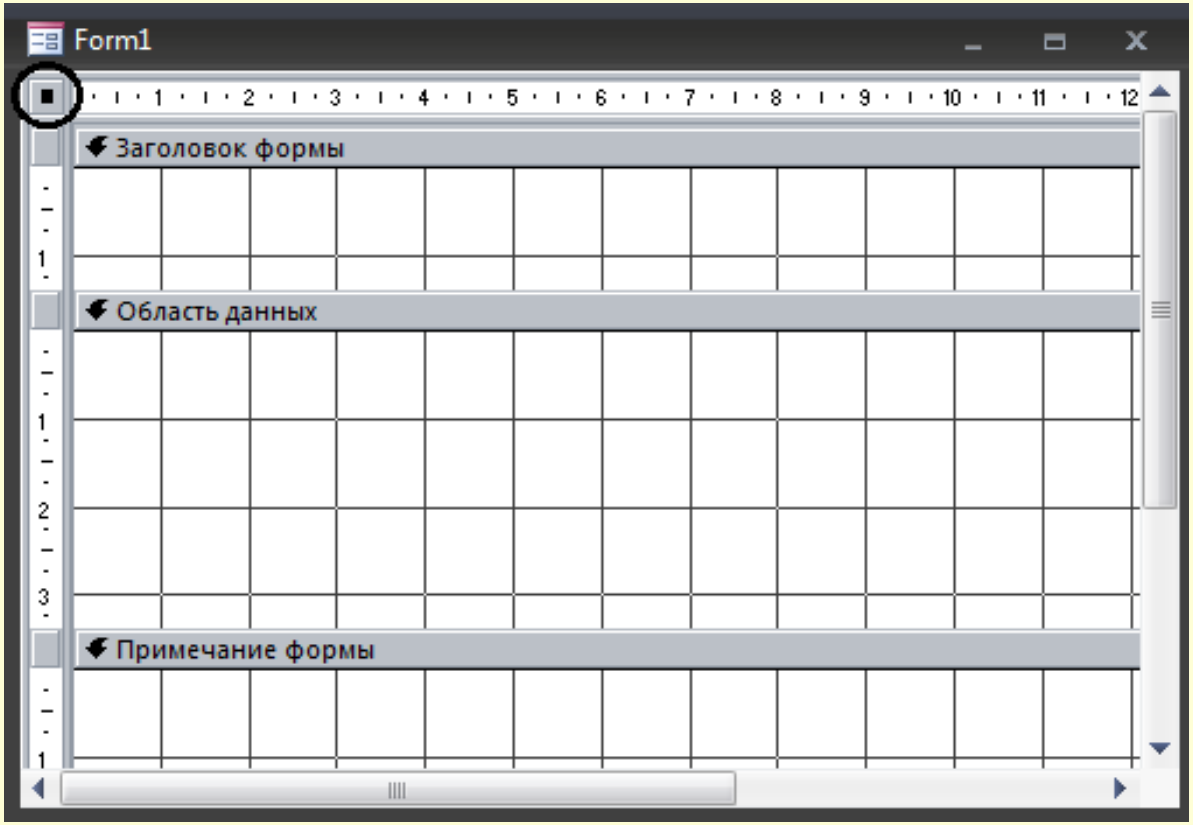
Структура формы

В режиме *Конструктора* может производиться настройка **областей** и **объектов** формы.

Области окна формы:

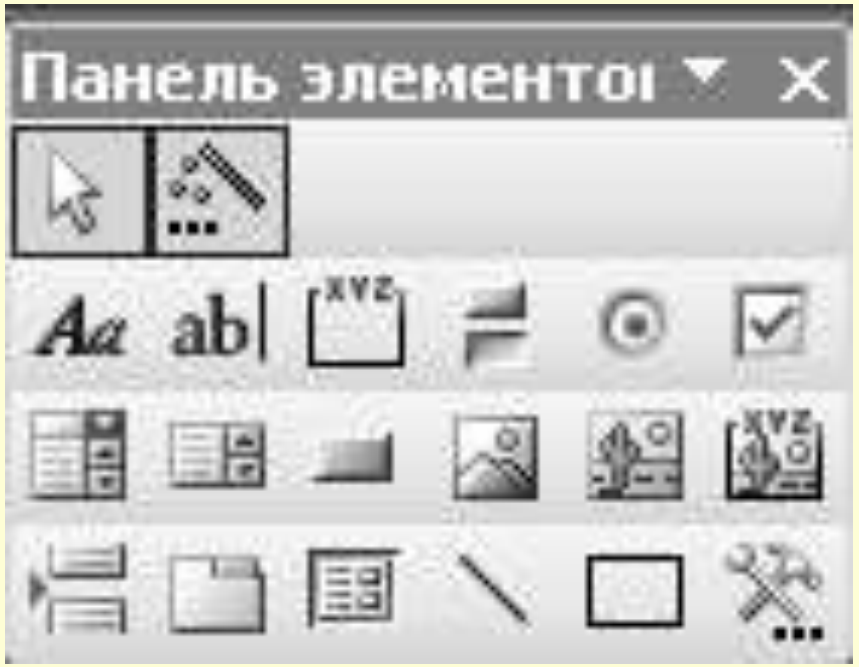
- область заголовка;
- область верхнего колонтитула;
- область данных;
- область нижнего колонтитула;
- область примечаний.

Замечание: для отображения областей нужно выбрать соответствующие команды из контекстного меню, вызываемого щелчком правой кнопки мыши по области формы.



Объекты в форме размещаются с помощью ***Панели элементов управления (ПЭУ)***, которая вызывается:

- в версии ACCESS 2003 из пункта меню ***Вид*** или пиктограммы на панели инструментов



➢в версии ACCESS 2007/2010 располагается на вкладке ленты ***Конструктор***:

Для размещения на форме **нового** ЭУ его нужно выбрать щелчком мыши на ПЭУ, перейти на бланк конструктора, при этом курсор должен принять форму крестика +, и нарисовать прямоугольник, удерживая левую кнопку мыши. В этом прямоугольнике будет размещен созданный ЭУ.

В Access существуют следующие типы элементов управления (ЭУ):

Выбор объектов. Позволяет выделить определённый элемент управления или группу ЭУ, если держать нажатой клавишу Shift.

Мастера. Если эта кнопка нажата, то при размещении на форме ЭУ запускается мастер, помогающий задать параметры элемента.

Надпись. Описательный текст.

Текстовое поле для ввода и редактирования текста.

Группа переключателей.

Выключатель. Может быть в двух состояниях: включено и выключено.

Переключатель. Несколько переключателей обычно объединяются в группу и позволяют выбрать одно из взаимоисключающих значений, например муж или жен.

Флажок. Используется для включения и отключения параметра.

Поле со списком. В поле можно ввести новое значение или выбрать существующее из списка.

Список. Содержит значения, из которых можно сделать выбор.

Кнопка. Используется для выполнения определенного действия или ряда действий.

Рисунок. Не меняется при переходе от одной записи к другой.

Свободная рамка объекта. Может отображать объект OLE.

Присоединённая рамка объекта. Отображает объекты OLE, хранящиеся в записях таблиц, например, рисунки, фотографии.

Разрыв страницы.

Набор вкладок. Позволяет разделить форму на несколько вкладок.

Подчинённая форма/отчёт. В форму добавляется информация из дополнительной таблицы.

Линия.

Прямоугольник.

Другие элементы, в том числе ActiveX. Позволяют добавить формам и отчетам ещё некоторые функциональные возможности.

Свойства и события объектов формы

Как и форма в целом, так и каждый из ее элементов обладает свойствами, которые позволяют определить его внешний вид, размер, местоположение в форме, режим ввода/вывода, а также привязать к элементу выражение, макрос или программу (т.е. встроенные события, которые выполняются при наступлении связанных с элементом действий).

Для того чтобы получить доступ к свойствам и событиям объекта формы необходимо выделить нужный объект, а затем:

в версии ACCESS 2003: выбрать команду ***Свойства*** из пункта меню **Вид** или нажать соответствующую кнопку на панели инструментов.	в версии ACCESS 2007/2010: выбрать пиктограмму ***Страница свойств*** на вкладке ленты **Конструктор** в группе инструментов **Сервис**. 

Замечание 1: для отображения окна свойств выбирается команда ***Свойства*** из контекстного меню, вызываемого щелчком правой кнопки мыши по выделенному объекту.

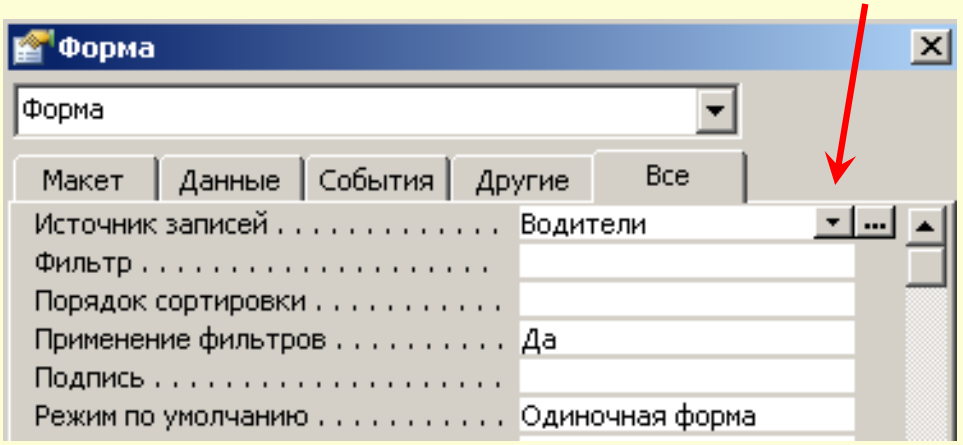
Замечание 2: для отображения окна свойств самой формы, необходимо выбрать команду ***Свойства*** из контекстного меню, вызываемого щелчком правой кнопки мыши по кнопке , расположенной в левом верхнем углу формы на пересечении горизонтальной и вертикальной линеек  .

В окне ***«Свойства»*** все свойства разбиты по группам на вкладках *Макет, Данные, События, Другие.*

Размещение элементов управления "Поле" в форме

При создании формы в режиме *Конструктора*, необходимо связать её с источником данных.

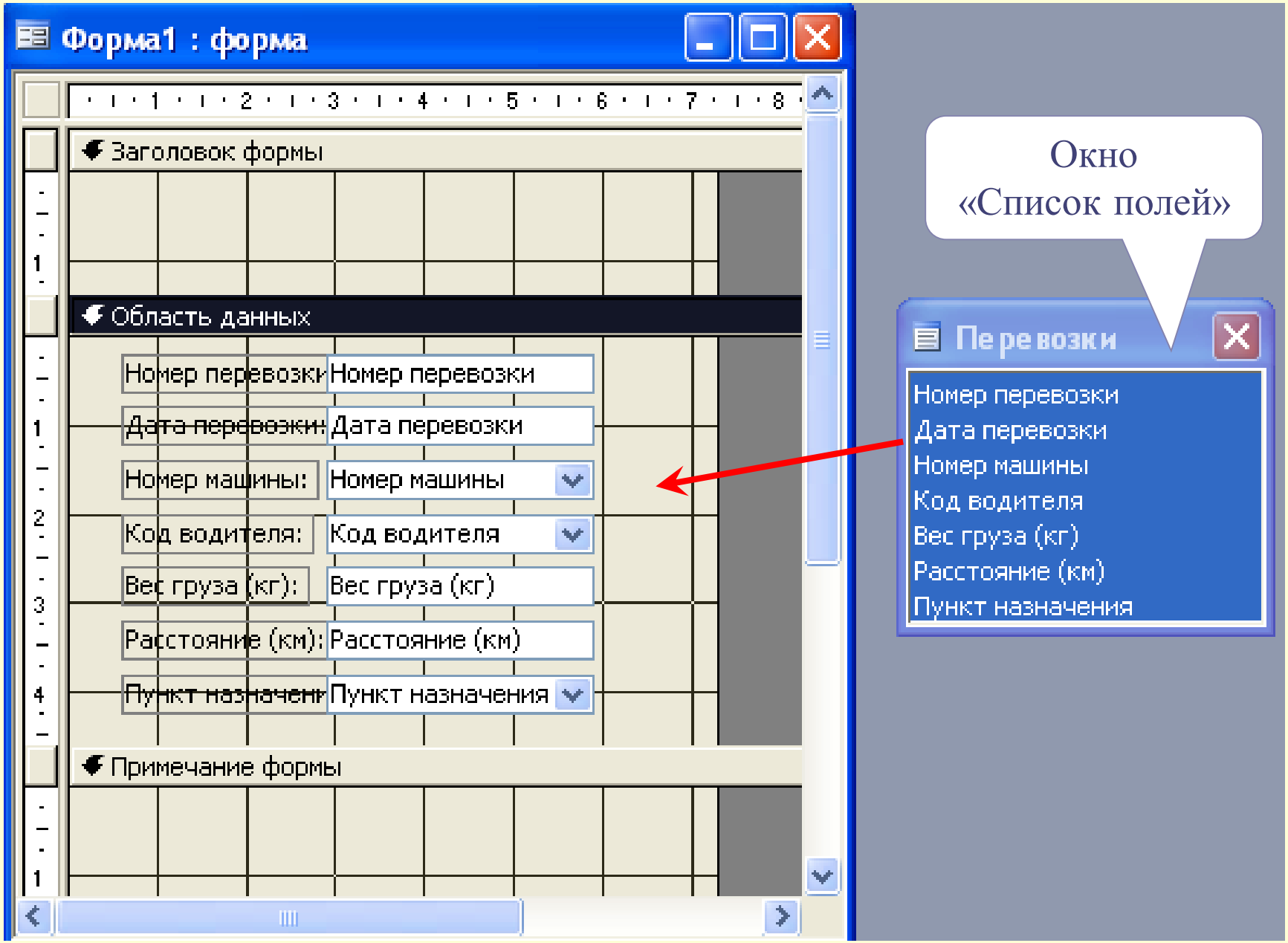
Для этого, в ***свойстве*** формы на вкладке *Данные* в строке Источник записей с помощью элемента управления "список" ( ) выбрать источник данных, т.е. таблицу или запрос.



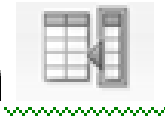
Далее в окне **«Список полей»** отобразятся поля текущего источника данных, которые необходимо переместить методом "перетащить и бросить" в область данных формы.

Замечание: для одновременного выделения всех полей используется клавиша **SHIFT**, а для нескольких отдельных полей – клавиша **CTRL**.

Такие поля называются ***присоединенными***, поскольку они связаны с данными таблиц или запросов.


Форма1 : форма
Заголовок формы
Область данных
Номер перевозки
Номер перевозки
Дата перевозки:
Дата перевозки
Номер машины:
Номер машины
Код водителя:
Код водителя
Вес груза (кг):
Вес груза (кг)
Расстояние (км):
Расстояние (км)
Пункт назначени
Пункт назначения
Примечание формы
Окно «Список полей»
Перевозки
Номер перевозки
Дата перевозки
Номер машины
Код водителя
Вес груза (кг)
Расстояние (км)
Пункт назначения

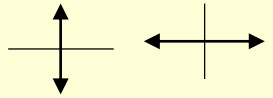
Замечание: для открытия окна ***«Список полей»***:

в версии ACCESS 2003 вызывается соответствующая команда из пункта меню **Вид** или **пиктограмма**  на панели инструментов.	в версии ACCESS 2007/2010 выбирается пиктограмма  Добавить поля, которая располагается на вкладке ленты **Конструктор** в группе инструментов **Сервис**.

Кроме присоединенных полей, существуют **свободные** поля, которые, например, могут использоваться для отображения результатов вычислений или для приёма данных, вводимых пользователем.

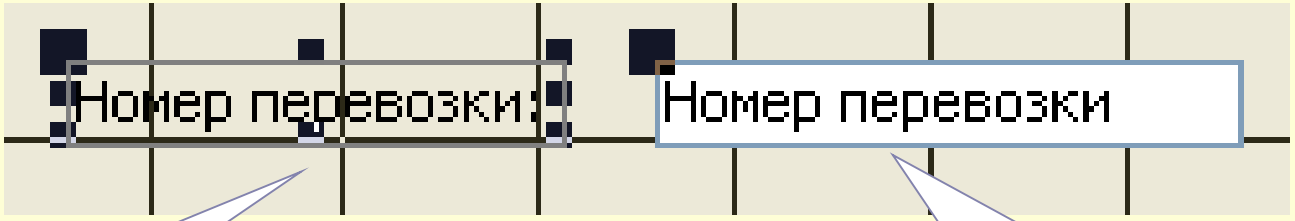
Возможности редактирования формы в режиме Конструктора:

1. изменение размеров **окна** *Конструктора* с помощью стрелок ↔, ↕ на границах окна;

2. изменение размеров **области формы** (область заголовка, данных, примечания) на границах области с помощью стрелок: 

3. изменение размеров **выделенного ЭУ**, используя маркеры размеров, расположенные на углах и серединах границ элемента управления;

4. передвижение при необходимости уже имеющихся выделенных полей формы, если два отдельных ЭУ (надпись и поле ввода) образуют связанную **группу**: подвести указатель к границе до появления «ладони»  (или символа ), зажать кнопку мыши, перетащить в другое место и отпустить;



Надпись

Поле ввода

5. передвижение при необходимости **отдельно** *надписи* или *поля ввода* с помощью маркера перемещения – квадратика ■ – в верхнем левом углу элемента, на котором указатель мыши отображается как «указательный палец»  ;

6. изменение внешнего вида элемента управления (цвет шрифта, границы, фона).

Пример1: Создать с помощью ***Мастера*** форму в один столбец для таблицы «Водители».

В созданной форме в режиме ***Конструктора*** добавить вычисляемые поля:

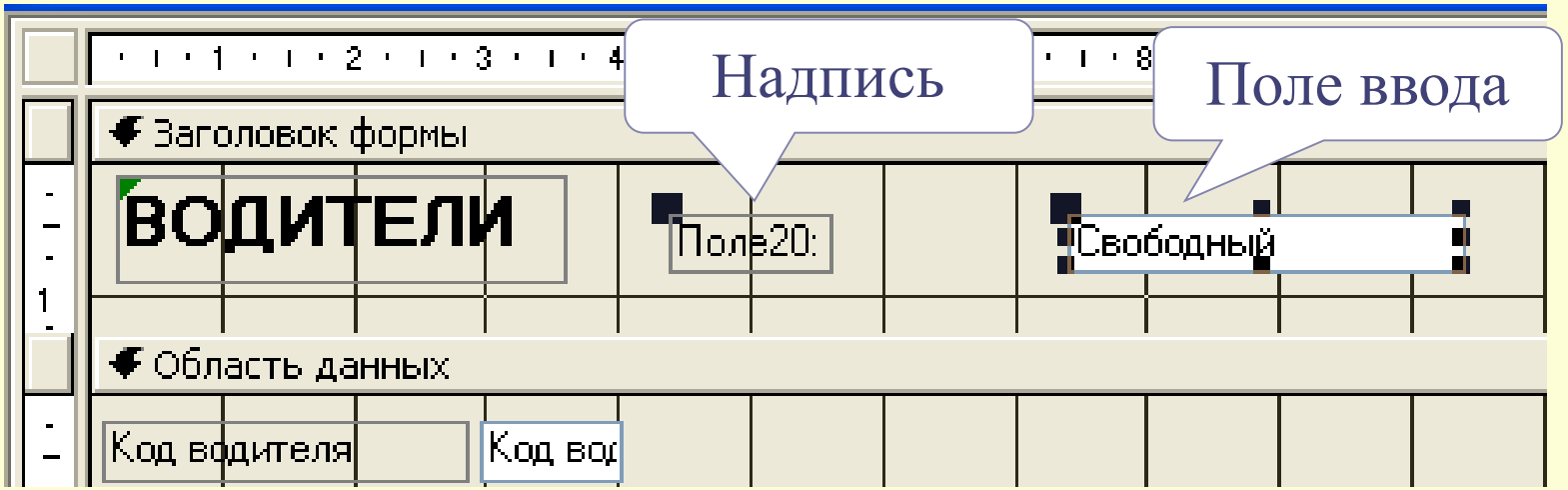
- в области заголовка – поле текущей даты,
- в области данных – поле для отображения дня недели даты рождения водителя.

Порядок выполнения:

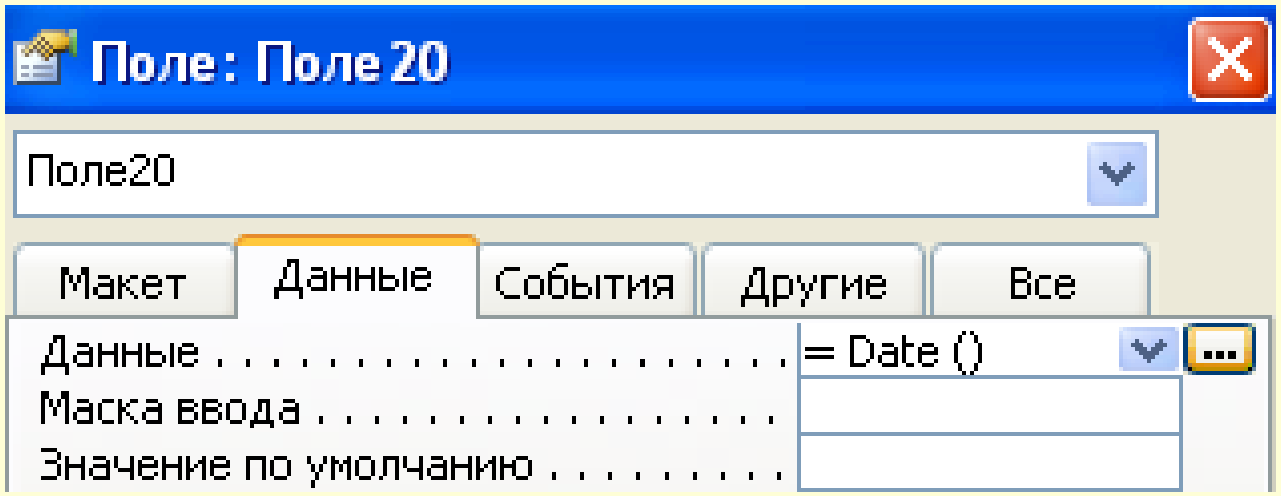
1. Последовательность работы с Мастером форм см. выше в п. 5.2.2.
2. После создания формы перейти в режим Конструктора.

3. Для создания свободных (вычисляемых) полей выполнить следующее:

3.1. выбрать на **Панели элементов управления** элемент **Поле** (ab|), нажать мышью место в нужной области формы, в которой предполагается разместить данный элемент. В форме появится связанный объект, состоящий из выделенного поля ввода и его надписи:



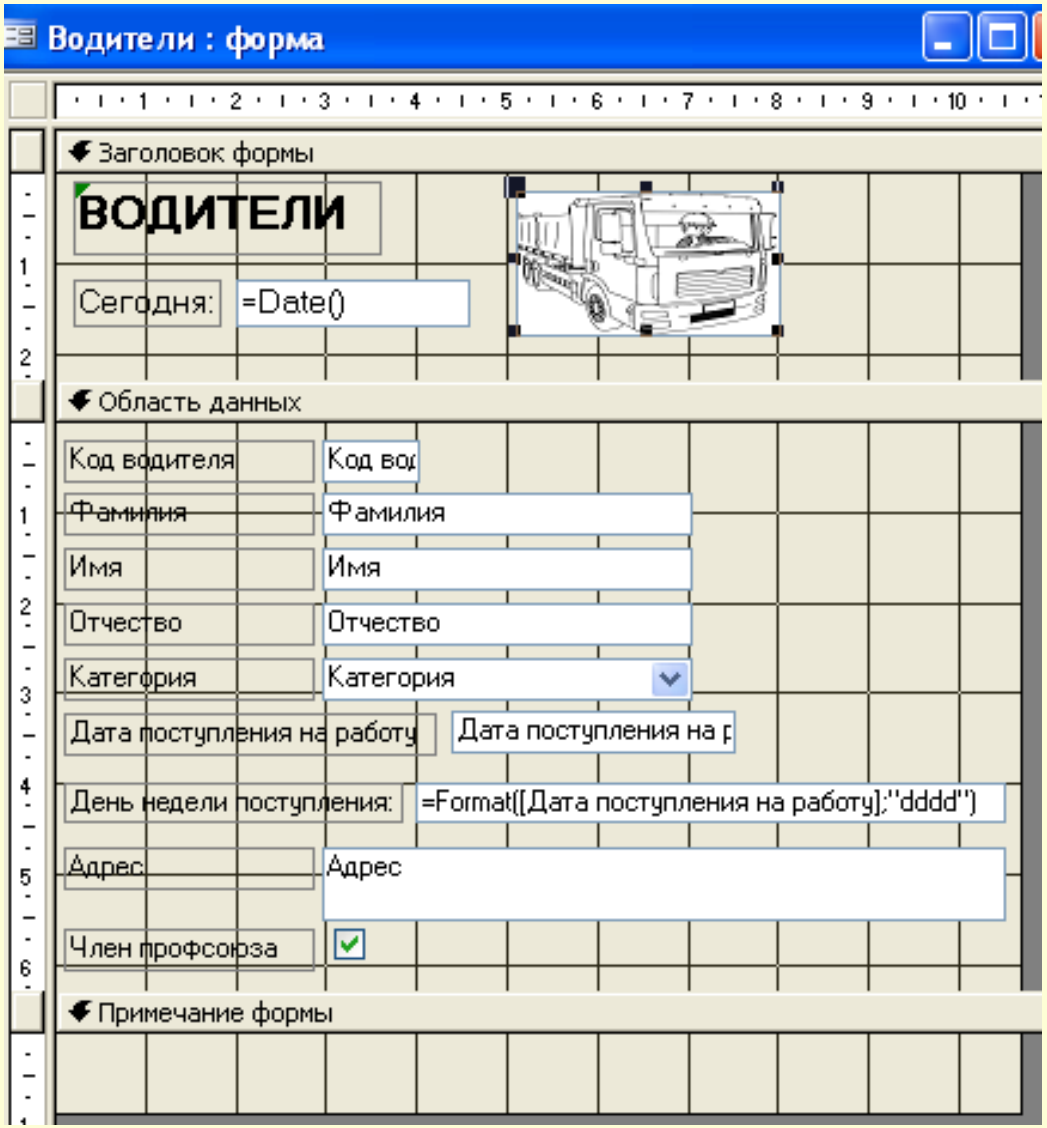
3.2. далее для выделенного нового поля ввода необходимо открыть окно «Свойства», в котором на вкладке ***Данные*** выбрать свойство *Данные*. Для добавления выражения с использованием встроенных функций целесообразно использовать ***Построитель выражений***, который можно открыть кнопкой … в правой части строки свойства *Данные*:

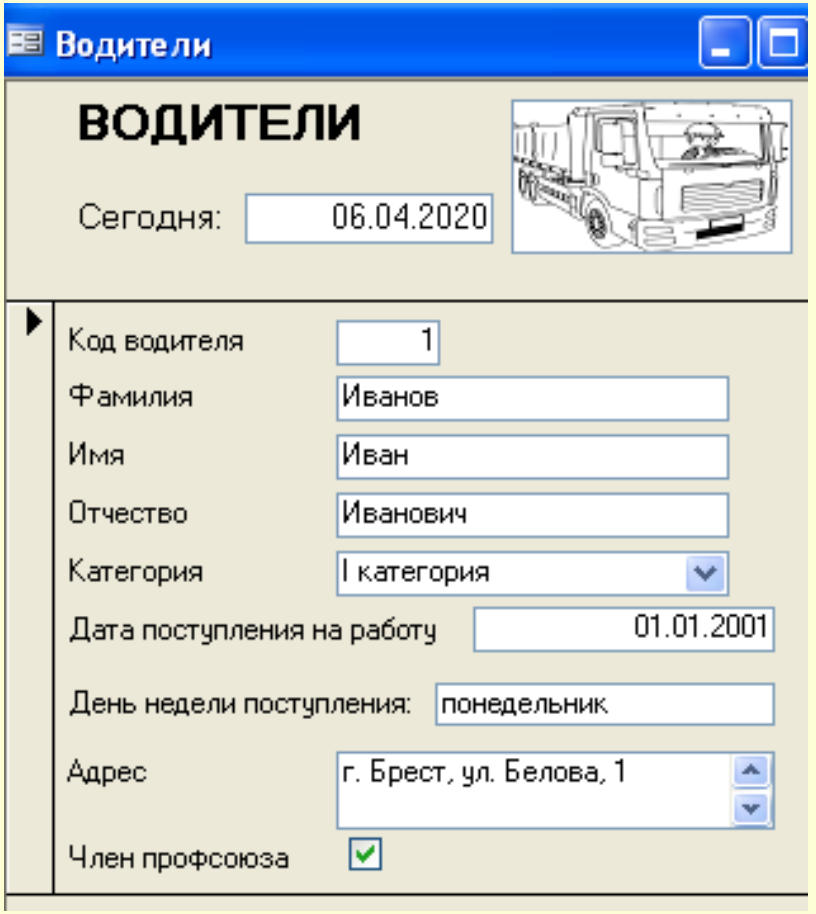


3.3. затем выделить надпись к полю ввода, открыть окно «Свойства» для этого элемента, чтобы ввести в свойстве ***Подпись*** подходящий текст с названием к полю ввода;

3.4. закрыть окно свойств и откорректировать ширину поля и надписи по содержимому;

3.5. перейти в режим **Формы** и просмотреть записи, используя кнопки внизу окна формы.


Водители : форма
Заголовок формы
ВОДИТЕЛИ
Сегодня:
=Date()
Область данных
Код водителя
Код вод
Фамилия
Фамилия
Имя
Имя
Отчество
Отчество
Категория
Категория
Дата поступления на работу
Дата поступления на р
День недели поступления:
=Format([Дата поступления на работу];"dddd")
Адрес
Адрес
Член профсоюза
Примечание формы


Водители
ВОДИТЕЛИ
Сегодня:
06.04.2020
Код водителя
1
Фамилия
Иванов
Имя
Иван
Отчество
Иванович
Категория
I категория
Дата поступления на работу
01.01.2001
День недели поступления:
понедельник
Адрес
г. Брест, ул. Белова, 1
Член профсоюза

Разработка многотабличных (составных) форм:

Многотабличная форма создается для работы на основе нескольких взаимосвязанных (как правило, связью "Один-ко-многим") таблиц и может состоять из одной формы или из основной и одной или нескольких подчиненных вложенных форм. Такой вид формы может быть создан в режиме *Конструктора* или с помощью *Мастера форм.* Однако в Access наиболее технологичным является способ первоначального создания форм с помощью Мастера и доработка их в режиме Конструктора.

Пример1: Создать многотабличную форму для просмотра и ввода записей в таблицы «Водители» и «Перевозки»

Порядок создания многотабличной формы с помощью Мастера:

1) <u>в версии ACCESS 2003</u>: находясь в окне БД на вкладке ***Формы*** нажать кнопку **Создать**; в окне **Новая форма** задать режим **Мастер форм**;

1) <u>в версии ACCESS 2007/2010</u>: выбрать пиктограмму  ***Мастер форм*** на вкладке ленты **Создание** в группе инструментов **Формы** (может находится в дополнительном меню **Другие формы**);

2) следовать шагам Мастера в окне «Создание форм»:

на 1-м шаге:

сначала выбрать из раскрывающегося списка "Таблицы и запросы" в качестве источника данных для формы **главную** таблицу-справочник «Водители» и выбрать все поля этой таблицы в область "Выбранные поля" посредством кнопки **>>**,

затем на этом же шаге выбрать **подчиненную** таблицу «Перевозки» и выбрать все поля этой таблицы в область "Выбранные поля" посредством кнопки **>>**,

нажать кнопку ***Далее***;

на 2-м шаге:

выбрать вид представления данных – ⊙Подчинённые формы, нажать ***Далее***;

на 3-м шаге:

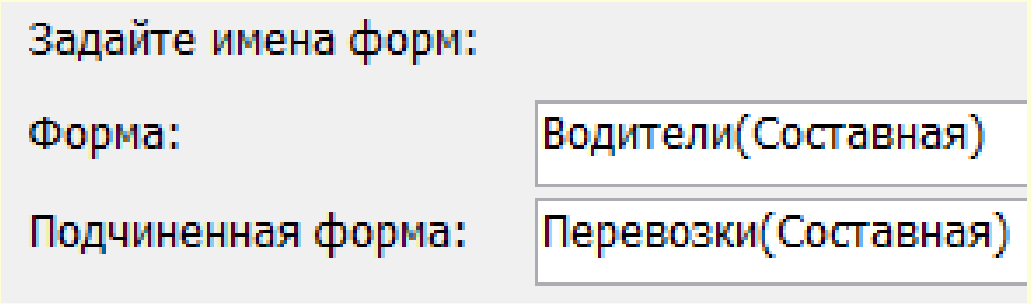
выбрать внешний вид подчиненной формы, например, "Табличный", нажать ***Далее***;

на 4-м шаге:

выбрать стиль оформления формы, например, "Стандартная", нажать ***Далее***;

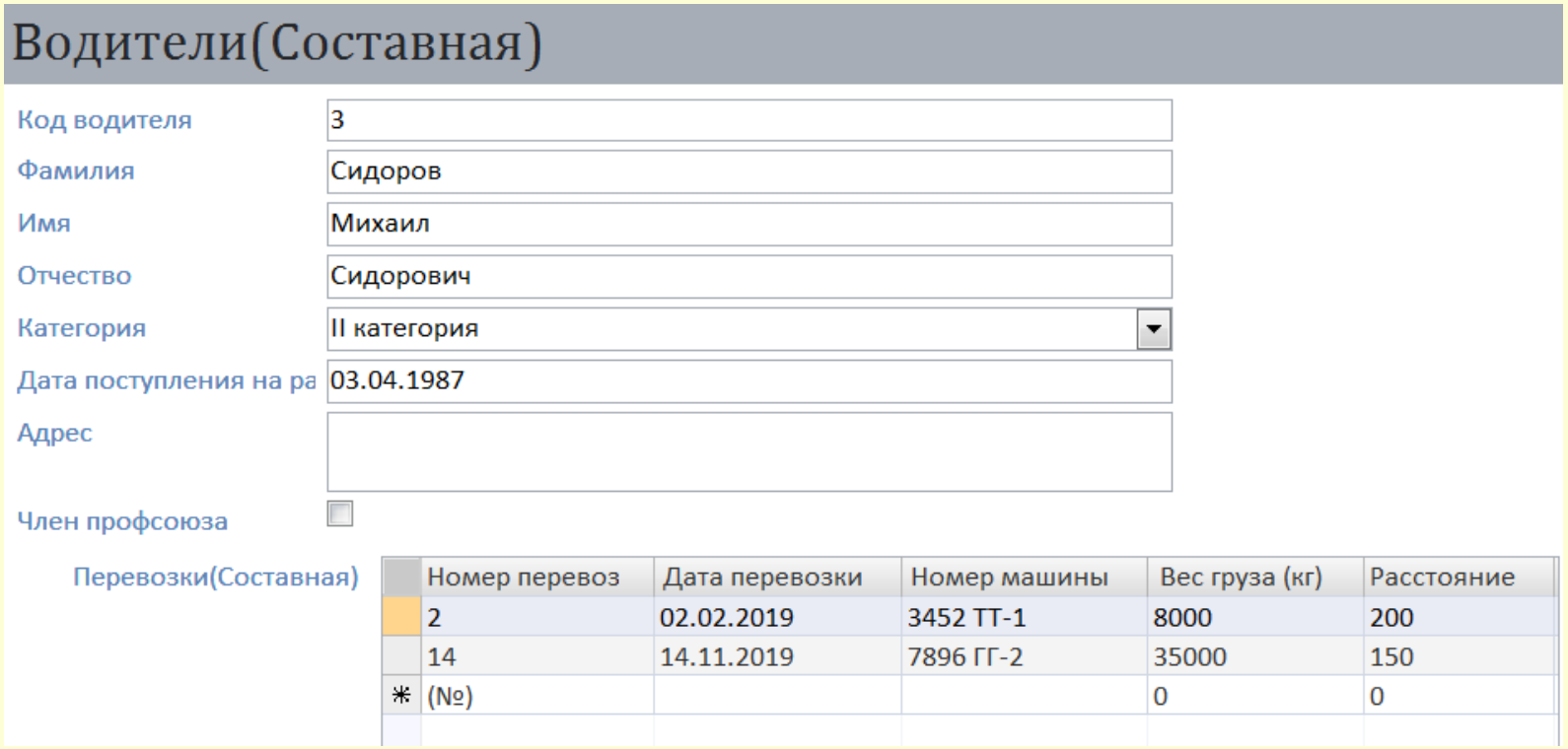
на 5-м шаге:

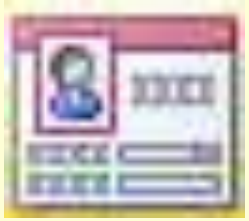
задать имя двум формам: основной и подчиненной, ***например***:



На этом же шаге выбрать один из вариантов дальнейших действий: либо открыть форму для просмотра данных, либо переключиться в режим *Конструктора* для дальнейшего её редактирования, нажать кнопку ***Готово***.

В результате получим вид формы, в которой можно просматривать информацию о каждом водителе в отдельности, а также информацию о его перевозках, т.е. связанных записей в подчиненной таблице:



Замечание: в версии ACCESS 2007/2010 составную форму для **главных** таблиц (участвующих в связях в отношении "ОДИН") можно создавать автоматически, выделив её в *списке объектов БД* и выбрав на вкладке ленты ***Создание*** в группе инструментов Формы пиктограмму **Формы** 

Глава 8.

Создание и работа с БД посредством графического интерфейса СУБД.

Технологии работы с базой данных в MS Access.

§ 8.5. Проектирование отчетов в MsA

Определение и возможности отчетов

«Отчет» в БД – это объект, предназначенный для формирования выходного документа, который может быть распечатан, т.е. отображение информации из БД в виде, удобном для ее восприятия и анализа пользователем.

Этот объект может создаваться как в табличной, так и в свободной форме.

В отчете можно группировать и сортировать данные, осуществлять расчеты в строках и проводить итоговые вычисления над группами строк и над всеми строками с использованием статистических функций.

Отчет может основываться на таблице или запросе и представлять сложные зависимости между различными наборами данных.

Замечание: Источником сведений для отчета может быть только одна таблица (или один запрос). Если данные для отчета находятся в нескольких таблицах, нужно сначала создать запрос на основе этих таблиц, который и будет являться источником данных.

Способы создания отчетов

- ***Автоотчет*** (автоматизированное средство);
- с помощью ***Мастера*** (в режиме диалога);
- в режиме ***Конструктора***;
- ***Диаграмма*** – создание отчета с диаграммой;
- ***Почтовые наклейки.***

Создание АВТООТЧЕТА в версии MsA 2003:

1 способ:

- ✓ в окне БД на вкладке Таблицы (вкладке Запросы) выделить таблицу (запрос), на основании которой (которого) будет создаваться отчет;
- ✓ вызвать команду п.м. Вставка → Автоотчет (или кнопку «Новый объект» на ПИ).

2 способ:

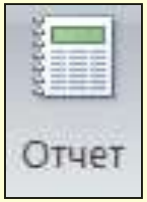
✓ в окне БД на вкладке Формы нажать кнопку **Создать**;

✓ в окне «Новая форма» задать режим – Автоотчет (в столбец, ленточный);

✓ выбрать из раскрывающегося списка в качестве источника данных для формы таблицу или запрос.

Создание АВТООТЧЕТА в версии MsA 2007/2010:

- находясь в окне БД в списке объектов Таблицы или Запросы (в области переходов – левая часть окна приложения), выделить таблицу или запрос, для которой(ого) создаётся отчет;
- выбрать на вкладке ленты ***Создание*** в группе инструментов **Отчеты** пиктограмму Отчет



Порядок создания отчета с помощью Мастера:

1) в версии ACCESS 2003: находясь в окне БД на вкладке ***Отчеты*** нажать кнопку **Создать**; в окне **Новый отчет** задать режим **Мастер отчетов** и выбрать из раскрывающегося списка в качестве источника данных для отчета **таблицу** или **запрос**;

1) в версии ACCESS 2007/2010: выбрать пиктограмму  ***Мастер отчетов*** на вкладке ленты **Создание** в группе инструментов **Отчеты**;

2) следовать шагам ***Мастера*** в окне «Создание отчета»:

на 1-м шаге – выбрать поля для отчета (кроме полей выбранной таблицы, например, можно добавлять числовые поля из связанных таблиц для подведения итогов);

на 2-м шаге – указать поле, по которому должна осуществляться группировка записей в отчете;

на 3-м шаге – выбор порядка сортировки в группах и задания итогов по группам, используя кнопку Итоги;

на 4-м шаге – выбрать макет отчета;

на 5-м шаге – выбрать стиль для отчета;

на 6-м шаге – задать имя отчету (можно именем таблицы-источника).

Порядок создания отчета в режиме Конструктора:

1) в версии ACCESS 2003: находясь в окне БД на вкладке ***Отчеты*** нажать кнопку **Создать**; в окне **Новый отчет** задать режим **Констуктор** и выбрать из раскрывающегося списка в качестве источника данных для отчета **таблицу** или **запрос**;

1) в версии ACCESS 2007/2010: выбрать пиктограмму  ***Конструктор отчетов*** на вкладке ленты **Создание** в группе инструментов **Отчеты**;

В режиме Конструктора может производиться настройка областей и объектов отчета.

Замечание: если нет каких-либо областей нужно выбрать

в версии ACCESS 2003:

- команду из ***п.м. Вид → Заголовок /примечание отчета***;
- команду из ***п.м. Вид → Колонтитулы***

в версии ACCESS 2007/2010:

на вкладке ленты **Упорядочить** в группе инструментов ***Отображение*** пиктограммы

- Заголовок /примечание отчета 
- Колонтитулы страницы 

Структурно вся информация в отчете (окне Конструктора) разбивается на разделы, предназначенные для отражения конкретной информации. При печати отчетов разделы размещаются на страницах в определенном порядке.

Области (разделы) окна отчета:

✓ **заголовок отчета** (печатается один раз в начале отчета на первой странице перед верхним колонтитулом и может удерживать: герб фирмы, название отчета, дату …);

✓**область верхнего колонтитула** (печатается в верхней части каждой страницы отчета и может удерживать заголовки столбцов);

✓ **область заголовка группы** (печатается в начале каждой новой группы записей и удерживает имя конкретной группы);

✓ **область данных** (удерживает данные отчета, источником которых являются значения записей);

✓**область примечания группы** (печатается после последней записи каждой группы и выводит результаты вычислений в разрезе групп);

✓ **область нижнего колонтитула** (печатается в нижней части каждой страницы отчета и может удерживать номера страниц);

✓ **область примечаний** (печатается один раз в конце отчета на последней странице и может удерживать подсчеты над всеми записями отчета).

Все сведения отчета (как и формы) содержатся в элементах управления (ЭУ). ЭУ представляют собой объекты, используемые для ***отображения данных***, ***выполнения действий*** или в качестве инструментов (графики, рисунки и пр.), которые дают возможность представить данные отчета (или формы) в более привлекательном виде и сделать более понятным их назначение.

ЭУ могут быть ***связанными***, ***свободными*** или ***вычисляемыми***.

Связанный ЭУ присоединен к полю базовой таблицы или запроса и используются для отображения, ввода или обновления значений из полей БД. **Свободные** ЭУ не имеют источника данных и используются для вывода на экран линий, прямоугольников и рисунков. Для **вычисляемого** ЭУ в качестве источника данных вводится выражение, в котором могут быть использованы данные из поля базовой таблицы или запроса для формы или отчета, а также данные другого ЭУ формы или отчета.

Объекты в отчете (как и на форме) размещаются с помощью ***Панели элементов управления (ПЭУ).*** Доступ к свойствам и событиям объектов отчета аналогичен доступу к свойствам и событиям объектов формы, который рассмотрен в **пункте 5.2.3. Лекции № 9 «Проектирование форм».**

Там же рассмотрены: способы вызова ПЭУ; организация связи отчета (по аналогии с формой) с источником данных; способы размещения и возможности редактирования ЭУ в отчете (по аналогии с формой).

Пример2: С помощью **Мастера** создать ОТЧЕТ по запросу «Стоимость перевозки», предусмотрев группировку записей по полю Номер машины. Для каждой группировки вычислить общую и максимальную стоимость перевозки. В режиме **Конструктора** добавить в *Заголовок отчета* надпись со своей ФИО.

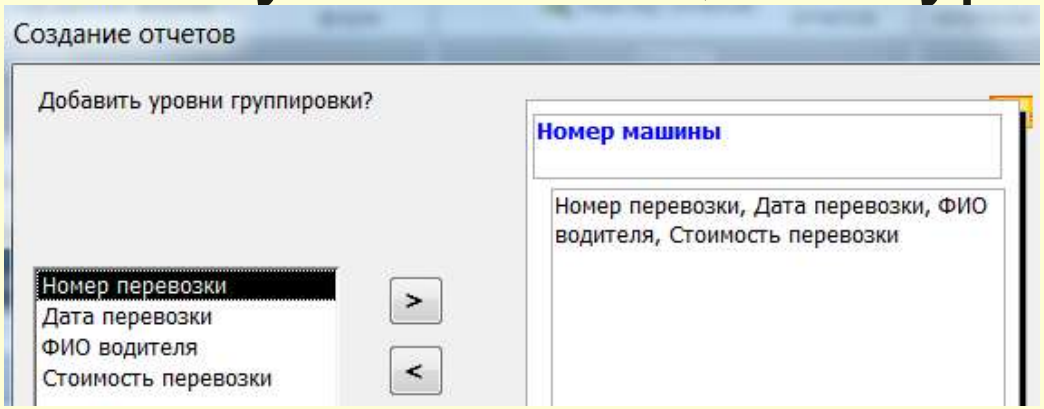
Последовательность действий:

1. в окне БД на вкладке **Отчеты** нажать кнопку ***Создать***;

2. в окне «Новый отчет» задать режим **Мастер отчетов** и выбрать из раскрывающегося списка в качестве источника данных для формы запрос «Стоимость перевозок», ОК;

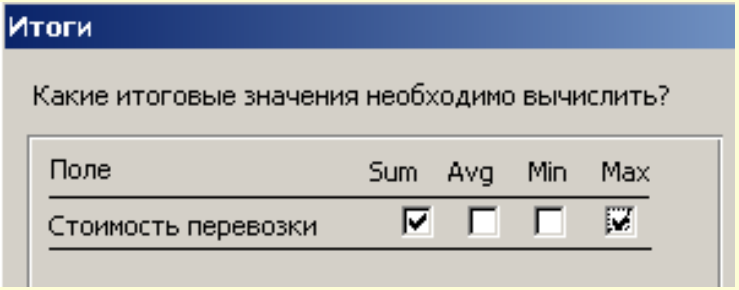
<u>на 1-м шаге</u> в появившемся окне «Создание отчетов» выбрать все поля посредством кнопки с изображением **>>** ;

<u>на 2-м шаге</u> в окне «Создание отчетов» оставить параметры по умолчанию, т.е. уровень группировки по полю «Номер машины»;



на 3-м шаге активизировать окно для подсчета итогов с помощью кнопки 

в котором оставить режим Показать ⊙данные и итоги, включить флажки под заданными функциями и нажать ОК:



на 4-м шаге выбрать вид макета для отчета (например, ступенчатый) и вид ориентации ⊙альбомная;

на 5-м шаге выбрать стиль для отчета (например, строгий);

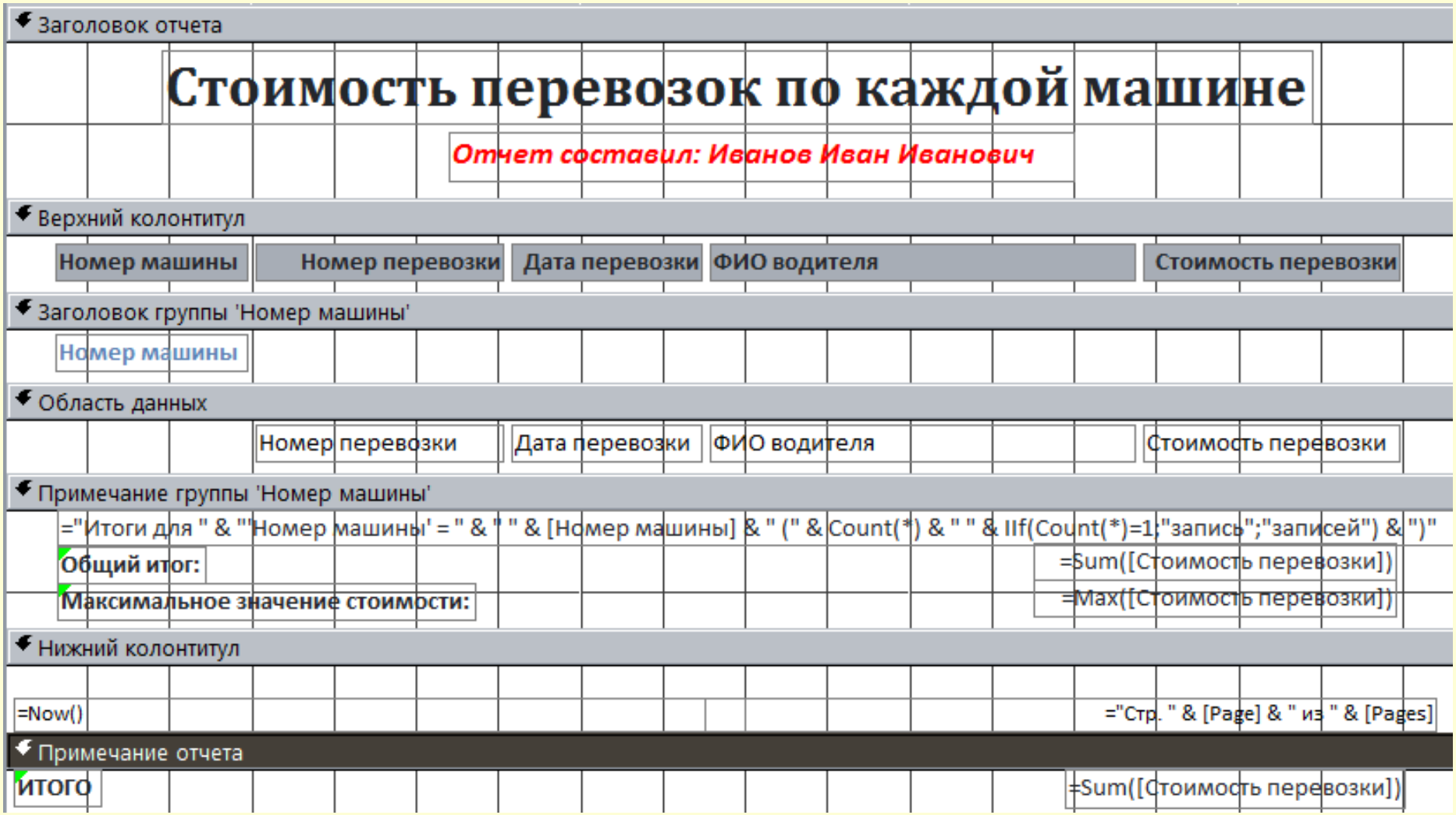
на 6-м шаге задать имя отчету, указанное в задании, и перейти к редактированию отчета, выбрав соответствующий переключатель:

⊙ **Изменить макет отчета.**

Для добавления текстового поля в заголовок отчета, необходимо выполнить следующее:

- на ПЭУ выбрать элемент Надпись Aa ;
- начертить с помощью мыши прямоугольник в заголовке отчета и напечатать нужный текст, отформатировав его с помощью соответствующих пиктограмм на панели инструментов «Формат».

В результате получим следующий вид отчета в режиме **Конструктора**:



В результате получим следующий вид части отчета в режиме **просмотра (представления)**:

Стоимость перевозок по каждой машине

Отчет составил: Иванов Иван Иванович

Номер машины	Номер перевозки	Дата перевозки	ФИО водителя	Стоимость перевозки
0111 КК-5				
	10	10.11.2019	Жуков Антон Михайлович	1 863 953,58
	8	06.11.2019	Петров Петр Петрович	795 302,05
	1	01.01.2019	Жуков Антон Михайлович	372 828,30
Итоги для 'Номер машины' = 0111 КК-5 (3 записей)				
Общий итог:				3 032 083,93
Максимальное значение стоимости:				1 863 953,58
2233 ОО-2				
	11	10.11.2019	Брунькин Артур Львович	1 100 135,62
	6	03.11.2019	Жуков Антон Михайлович	2 860 307,98
Итоги для 'Номер машины' = 2233 ОО-2 (2 записей)				
Общий итог:				3 960 443,60
Максимальное значение стоимости:				2 860 307,98

Пример3: Создать **АВТООТЧЕТ** по запросу «Пробег». В режиме **Конструктора** внести следующие изменения в структуру отчета:

- добавить уровни группировки записей по полю ***Номер машины;***
- для каждой выделенной группы подсчитать итоговые (суммарные) значения всех вычисляемых в запросе полей;

<u>Последовательность действий:</u>

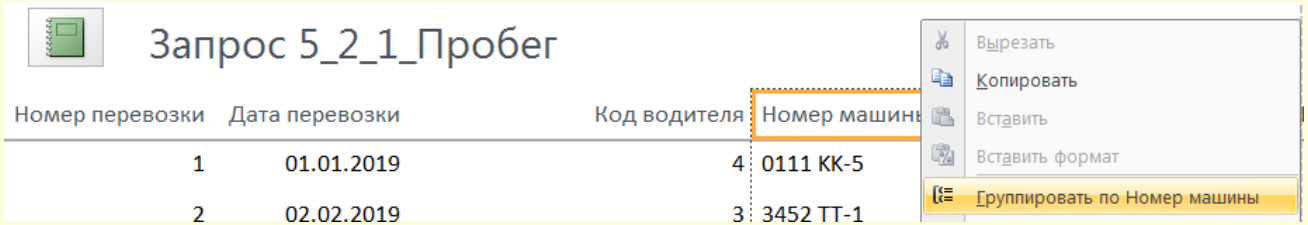
1. Порядок создания автоотчета см. в пунктах 6.2.1. для MsA-2003 (2 способ – Автоотчет: ленточный) и 6.2.2. для MsA-07/10 текущей презентации;

Для добавления уровней группировки необходимо:

в версии MsA 2007/2010:

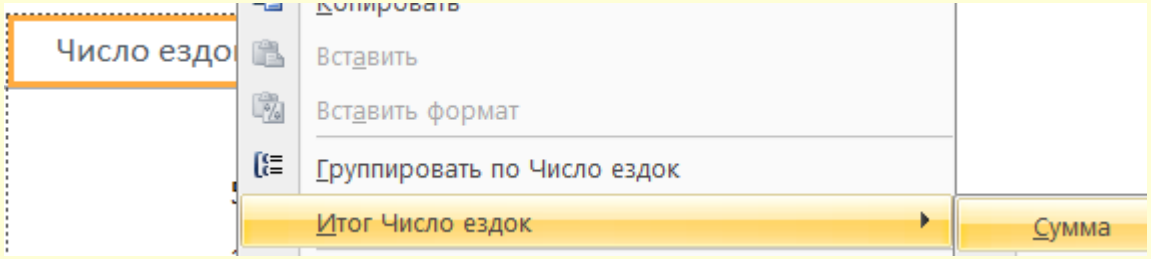
2. перейти в **режим Макета** отчета; 

3. щелкнуть правой кнопкой заголовок поля ***Номер машины***, по которому требуется выполнить группировку и в контекстном меню выбрать команду **Группировать**



Замечание: Access добавит уровень группировки и создаст заголовок группы.

4. щелкнуть правой кнопкой заголовок поля ***Число ездок***, по которому требуется выполнить группировку и в контекстном меню выбрать команду **Итог**, затем выбрать необходимую функцию – **Сумма**



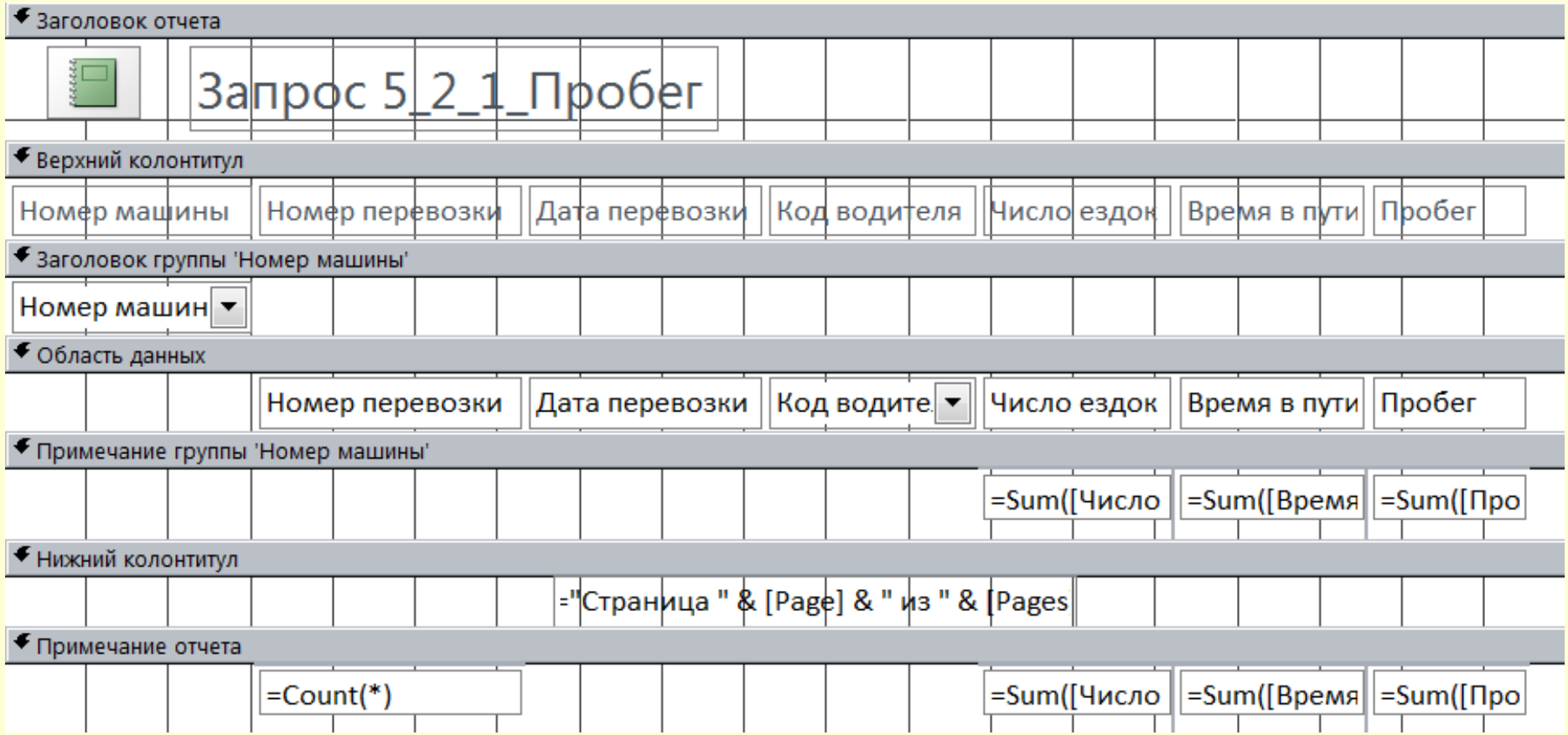
Аналогично итоговые поля создаются для атрибутов ***Время в пути*** и ***Пробег***.

В результате получим следующий вид отчета в режиме **просмотра (представления)**:

Запрос 5_2_1_Пробег

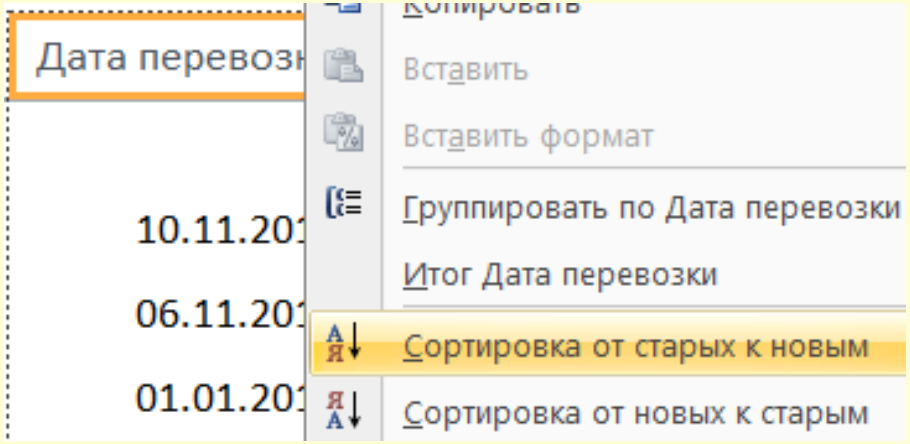
Номер машины	Номер перевозки	Дата перевозки	Код водителя	Число ездок	Время в пути	Пробег
0111 КК-5						
	10	10.11.2019	4	5	30,00	1500
	8	06.11.2019	2	2	12,80	640
	1	01.01.2019	4	1	6,00	300
				8	48,80	2440
2233 ОО-2						
	11	10.11.2019	5	1	19,23	1000
	6	03.11.2019	4	2	50,00	2600
				3	69,23	3600

В результате получим следующий вид отчета в режиме **Конструктора**:



Замечание1: Если в отчете содержатся уровни группировки, Access добавит колонтитулы групп (если их нет) и поместит итоговое значение в каждый колонтитул. Access добавит в колонтитул отчета такой элемент управления, как вычисляемый текст, в котором подводится общий итог.

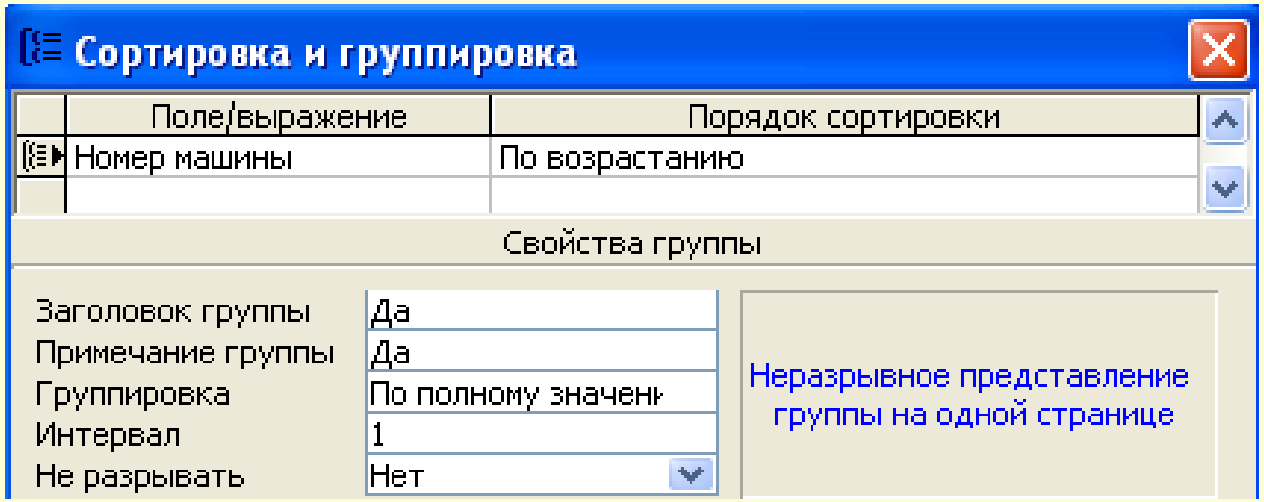
Замечание2: ***для сортировки по выбранному полю необходимо*** щелкнуть правой кнопкой заголовок поля и в контекстном меню выбрать команду **Сортировать**



Для добавления уровней группировки необходимо:

в версии MsA 2003:

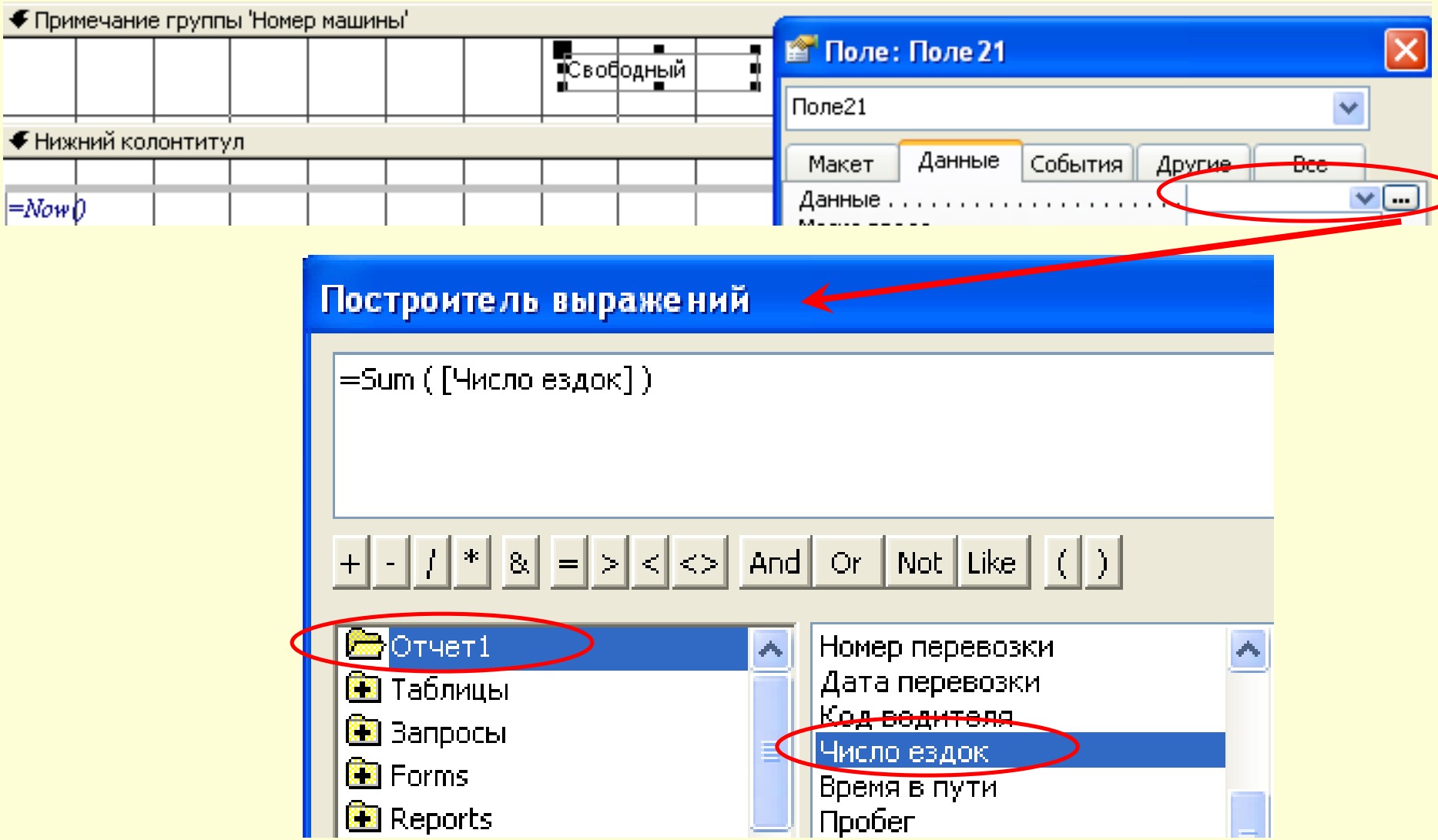
2. перейти в **режим Конструктора** отчета;
3. выбрать команду ***п.м. Вид → Сортировка и группировка***;
4. в появившемся одноименном окне выбрать поле для группировки и свойства группы:



Замечание: Access добавит в окне **Конструктора** два новых раздела: ***Заголовок группы*** и ***Примечание группы***.

5. в *Заголовок группы* переместить ЭУ-надпись ***Номер машины*** из области Верхний колонтитул и ЭУ-поле ***Номер машины*** из Области данных;

6. в области *Примечание группы* добавить три свободных ЭУ-поле, удалить для них надписи и ввести в эти поля выражения для подсчета суммарных значений по заданным полям;

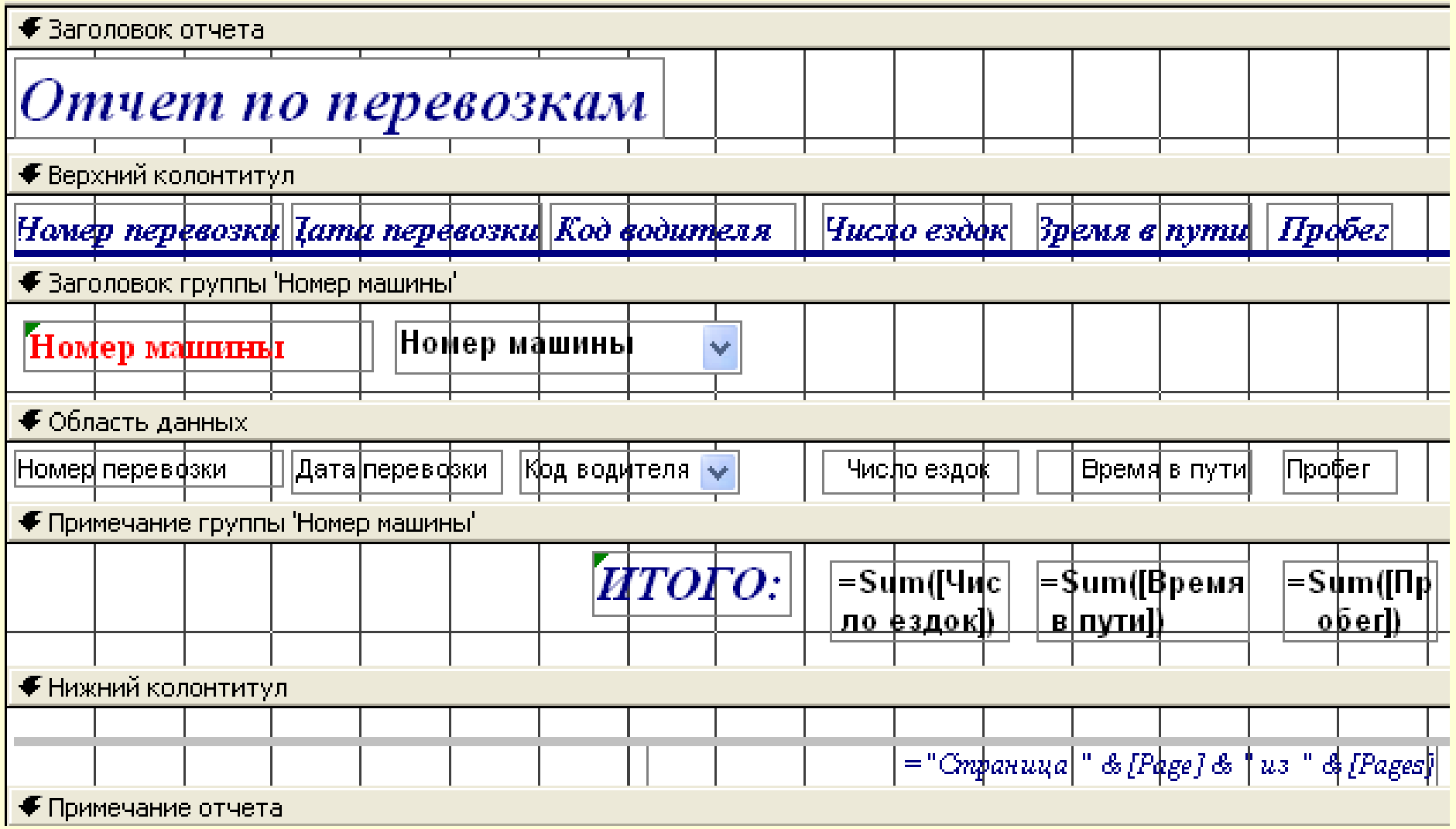

Примечание группы 'Номер машины'
Свободный
Нижний колонтитул
=Now()
Поле: Поле21
Поле21
Макет
Данные
События
Другие
Все
Данные . . . . . .
Построитель выражений
=Sum ([Число ездок])
+ - / * & = > < <> And Or Not Like ( )
Отчет1
Таблицы
Запросы
Forms
Reports
Номер перевозки
Дата перевозки
Код водителя
Число ездок
Время в пути
Пробег

6. в области *Примечание группы* добавить три свободных ЭУ-поле, удалить для них надписи и ввести в эти поля выражения для подсчета суммарных значений по заданным полям;

7. устранить пустоты между ЭУ в области верхнего колонтитула и области данных путем перемещения ЭУ вдоль области;

8. установить, при необходимости, нужные параметры страницы командой из п.м. Файл Параметры страницы.

В результате получим следующий вид отчета в режиме **Конструктора**:



В результате получим следующий вид отчета в режиме **Просмотра**:

Отчет по перевозкам

Номер перевозки	*Дата перевозки*	*Код водителя*	*Число ездок*	*Время в пути*	*Пробег*
Номер машины	**0111 КК-5**				
10	10.11.2019	4	5	30,00	1500
8	06.11.2019	2	2	12,80	640
1	01.01.2019	4	1	6,00	300
		ИТОГО:	**8**	**48,8**	**2440**
Номер машины	**2233 ОО-2**				
11	10.11.2019	5	1	19,23	1000
6	03.11.2019	4	2	50,00	2600
		ИТОГО:	**3**	**69,23**	**3600**

Глава 8.

Создание и работа с БД посредством графического интерфейса СУБД.

Технологии работы с базой данных в MS Access.

§ 8.6. Автоматизация работы с БД в MsA: макросы. Управление БД.

Определение и возможности макросов

«Макрос» – это набор из одной и нескольких макрокоманд, которые обеспечивают последовательность операций и применяются для автоматизации их выполнения.

Макрокоманда – это инструкция, ориентированная на выполнение определенного действия.

Макросы оперируют объектами MsA (запросами, формами, отчетами) и объединяют разрозненные действия с ними в единую программу действий, направленных на решение задачи пользователя, полностью отстраняя его от её решения, а также применяются для управления запуском приложений при открытой БД.

В MsA макросы можно рассматривать как упрощенный язык программирования, на котором программа записывается в виде списка макрокоманд для выполнения.

Макрокоманды можно классифицировать по назначению:

- макрокоманды для работы с данными в формах и отчетах (например, *НайтиЗапись*);
- макрокоманды выполнения (например, *ОткрытьЗапрос*);
- макрокоманды импорта/экспорта (например, *ОтправитьОбъект*);
- макрокоманды для работы с объектами БД (например, *КопироватьОбъект*);
- другие (например, *Сообщение*).

Макросы различаются по структуре и разделяются на:

- ***простые*** или ***линейные*** (состоящие из последовательности макрокоманд, выполняющихся одна за другой);
- ***групповые*** (набор логически связанных макросов, сохраняющихся под общим именем);
- ***макросы с условием*** (в которых отдельная макрокоманда или набор команд выполняются в зависимости от выполнения, которое задается логическим выражением);
- ***макросы с циклом*** (в котором осуществляется многократное выполнение другого макроса).

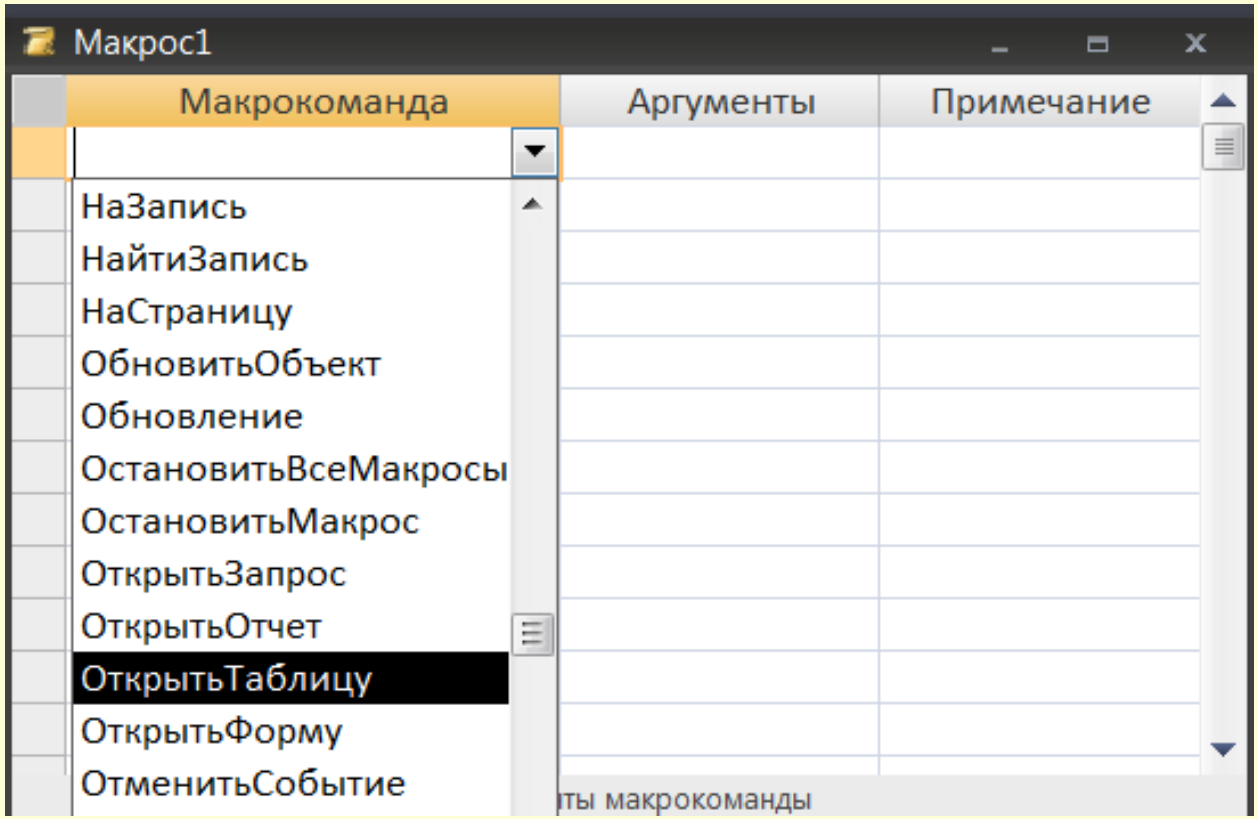
Макрос создается с помощью **Конструктора макросов**.

Порядок создания макросов:

1. в версии ACCESS 2003: находясь в окне БД на вкладке ***Макросы*** нажать кнопку **Создать**.

1. в версии ACCESS 2007/2010: выбрать пиктограмму  ***Макрос*** на вкладке ленты **Создание** в группе инструментов **Другие** (или в группе **Макросы и код**).

2. в окне диалога «Макрос» в ячейке столбика «Макрокоманда» раскрыть список макрокоманд, которые могут быть использованы в макросе (в списке расположены имена типовых процедур);



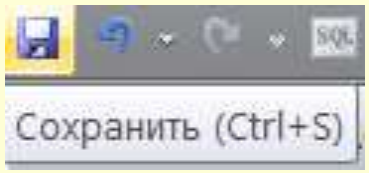
3. выбрать имя макрокоманды;

4. по желанию ввести текст комментария к макрокоманде (столбец ***Примечание***);

5. в нижней части окна макроса указать аргументы (константа, переменная или выражение которые являются источником данных для макрокоманды, процедуры или метода);

6. для включения в макрос других макрокоманд перейти на другую строку и повторить действия пп. 2 – 5;

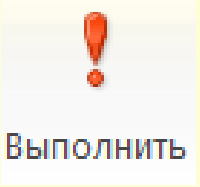
7. сохранить макрос: ***п.м. Файл → Сохранить как***… (в MsA 2003); команда ***Сохранить*** на панели быстрого доступа (в MsA 2007/2010).



Способы запуска макроса:

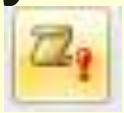
1 способ: из окна макроса:

- кнопкой «Запуск» (!) на панели инструментов (в MsA 2003);
- кнопкой «Выполнить» на вкладке ленты **Конструктор** в группе инструментов **Сервис** (в MsA 2007/2010);



2 способ: из окна БД на вкладке **Макросы** выбрать имя макроса и дважды щелкнуть по нему;

3 способ: в окне БД:

- командой в ***п.м. Сервис → Макрос → Выполнить макрос...*** (в MsA 2003);
- кнопкой ***«Выполнить макрос»*** на вкладке ленты **Работа с базами данных** в группе инструментов **Макрос** (в MsA 2007/2010)  ;

открыть окно «Запуск макроса» → в поле «Имя макроса» ввести или выбрать имя макроса → «ОК»;

4 способ: для запуска макроса, который входит в группу макросов, нужно руководствоваться общим синтаксисом:

Имя Группы макросов. Имя макроса

Понятие события в MsA

В MsA имеется возможность организовать выполнение макросов, используя механизм расширенной обработки событий (т.е. результатов действия пользователя).

MsA распознает определенные события, к которым может привязываться запуска макроса.

Событие (event) – это изменение состояния объекта БД, в момент возникновения которого можно изменить стандартный порядок обработки объекта и определить свою, нестандартную, реакцию.

Примеры событий: нажатие кнопок и клавиш интерфейса; перемещение мыши; открытие объекта (формы, отчета); …

События делятся на 8 категорий:

1. события ***окна формы, отчета*** (например, Открытие);
2. события ***данных*** (например, Изменение);
3. события ***фокуса ввода*** (например, Вход);
4. события ***клавиатуры*** (например, Нажатие клавиши);
5. события ***мыши*** (например, Нажатие кнопки);

6. события ***печати*** (например, Страница);
7. события ***ошибки***;
8. события ***таймера***.

В MsA только формы и отчеты являются объектами, для которых определены события.

Примеры некоторых событий в Access:

- Включение (Activate)
- Загрузка (Load)
- Выгрузка (Unload)
- Вход (Enter)
- Выход (Exit)
- Нажатие кнопки (Click)
- Двойное нажатие кнопки (DblClick)
- Открытие (Open)
- Закрытие (Close)
- Печать (Print)
- Удаление (Delete)
- Форматирование (Format)
- …

Для обработки событий разрабатываются макросы, которые классифицируются следующим образом:

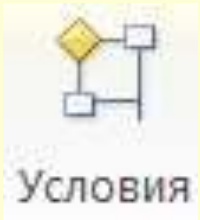
1. макросы, связанные с событиями *элементов управления в форме*;
2. макросы, связанные с событиями *раздела формы*;
3. макросы, связанные с событиями *формы*;
4. макросы, связанные с событиями *раздела отчета*;
5. макросы, связанные с событиями *отчета*.

Применение условий в макросе

Условие определяет порядок передачи управления при выполнении макроса между макрокомандами. Условие задается логическим выражением.

Для ввода условий команд в макрос нужно:

- находясь в окне макроса выбрать команду ***Условие*** из ***п.м. Вид*** (в MsA 2003);
- кнопкой «Условия» на вкладке ленты **Конструктор** в группе инструментов **Показать или скрыть** (в MsA 2007/2010);

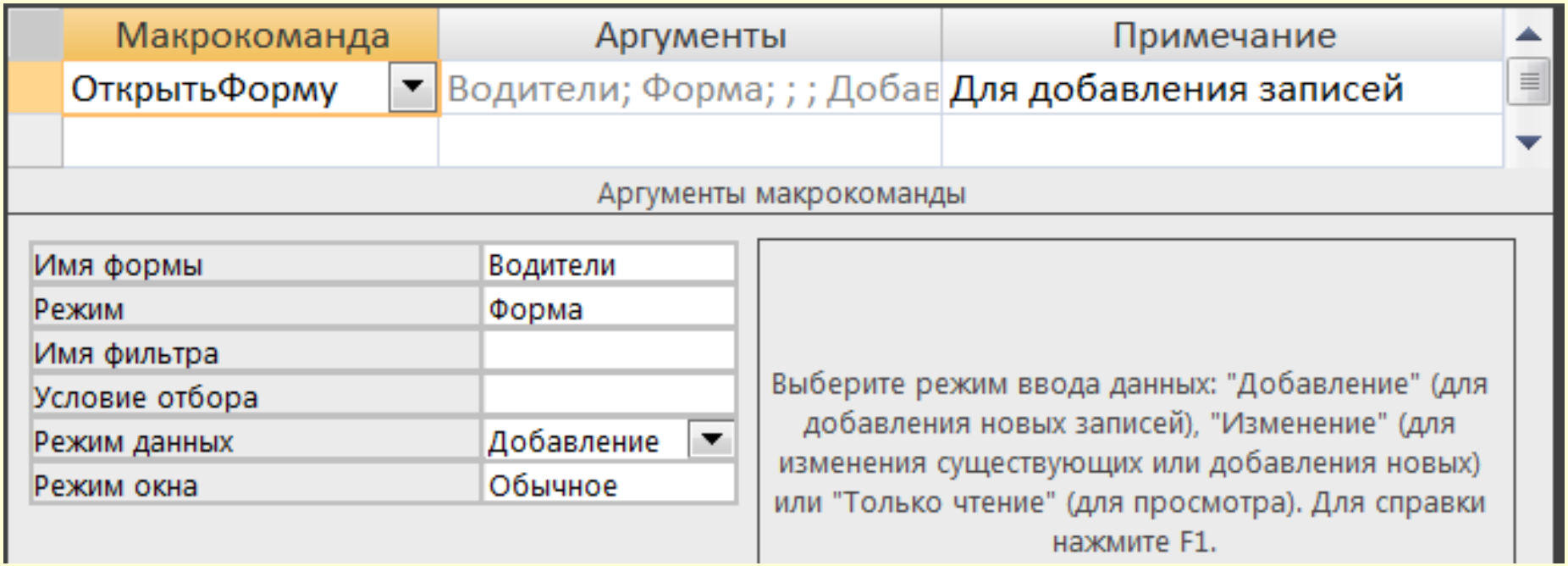


в результате в окне макроса слева выводится третий столбик «Условие» для записи условий.

Если условие «истина», то выполняется макрокоманда в строке записи этой макрокоманды.

Макросы, не связанные с событиями

Пример1. Создать макрос, позволяющий автомтически открывать форму “Водители” в режиме добавления записей:

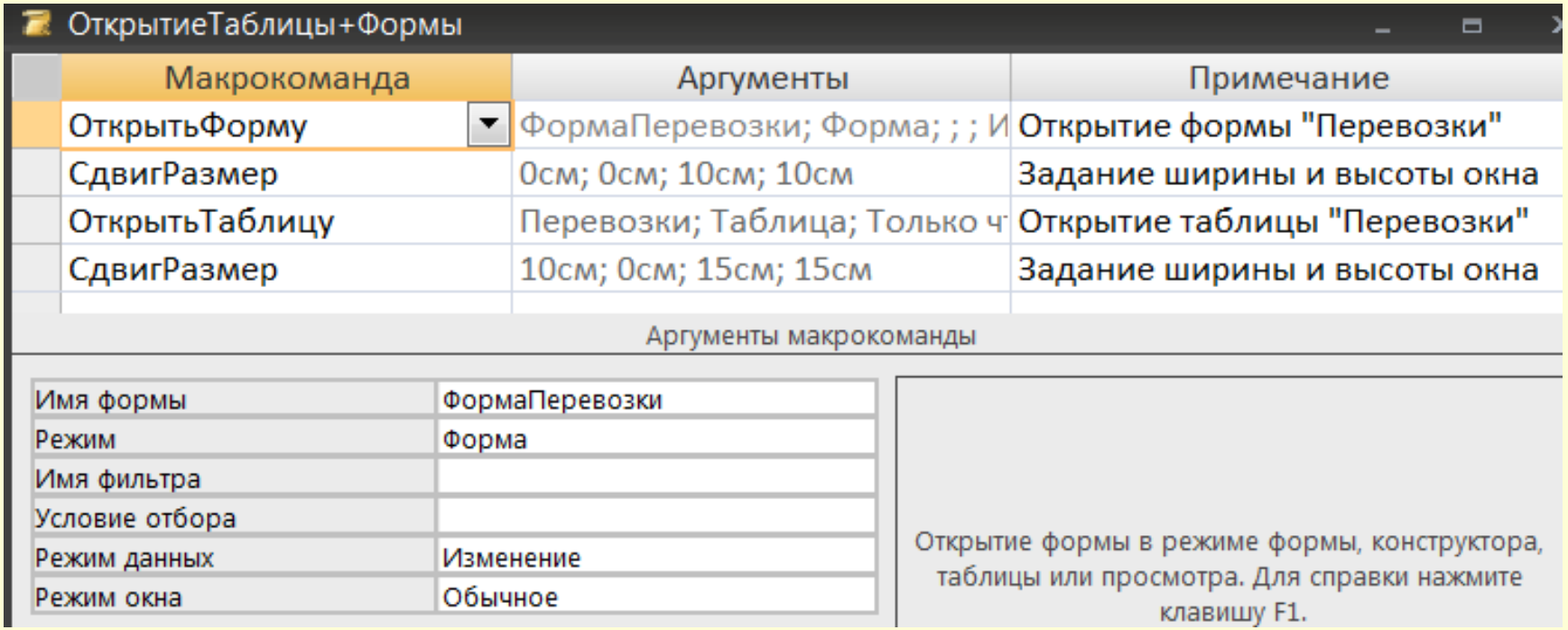


Пример2. Создать макрос, позволяющий отобразить на экране одновременно таблицу “Перевозки” для внесения изменений и форму, созданную на ее основе для просмотра этих изменений.

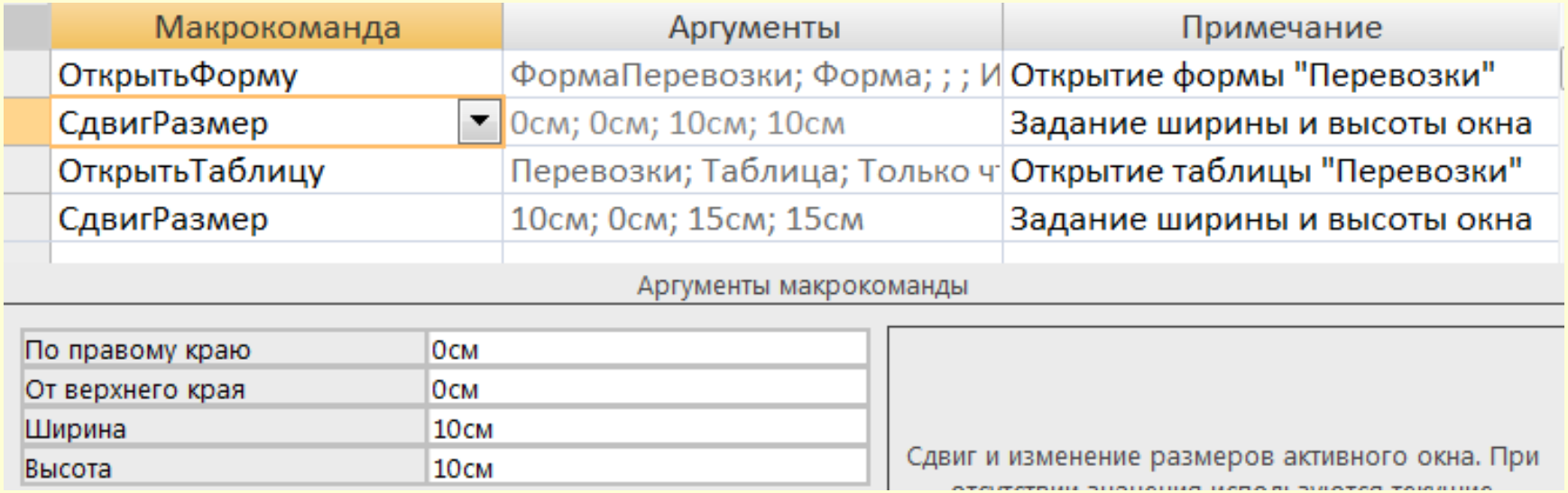
Порядок выполнения:

В окне Конструктора макросов проектируем макрос с ***линейной*** структурой, предварительно сделав окно БД неразвернутым полностью в окне приложения MsA:

1.


ОткрытиеТаблицы+Формы
Макрокоманда
Аргументы
Примечание
ОткрытьФорму
ФормаПеревозки; Форма; ; ; И
Открытие формы "Перевозки"
СдвигРазмер
0см; 0см; 10см; 10см
Задание ширины и высоты окна
ОткрытьТаблицу
Перевозки; Таблица; Только ч
Открытие таблицы "Перевозки"
СдвигРазмер
10см; 0см; 15см; 15см
Задание ширины и высоты окна
Аргументы макрокоманды
Имя формы
ФормаПеревозки
Режим
Форма
Имя фильтра
Условие отбора
Режим данных
Изменение
Режим окна
Обычное
Открытие формы в режиме формы, конструктора, таблицы или просмотра. Для справки нажмите клавишу F1.

2.


Макрокоманда
Аргументы
Примечание
ОткрытьФорму
ФормаПеревозки; Форма; ; ; И
Открытие формы "Перевозки"
СдвигРазмер
0см; 0см; 10см; 10см
Задание ширины и высоты окна
ОткрытьТаблицу
Перевозки; Таблица; Только ч
Открытие таблицы "Перевозки"
СдвигРазмер
10см; 0см; 15см; 15см
Задание ширины и высоты окна
Аргументы макрокоманды
По правому краю
0см
От верхнего края
0см
Ширина
10см
Высота
10см
Сдвиг и изменение размеров активного окна. При

3.

Макрокоманда	Аргументы	Примечание
ОткрытьФорму	ФормаПеревозки; Форма; ; ; И	Открытие формы "Перевозки"
СдвигРазмер	0см; 0см; 10см; 10см	Задание ширины и высоты окна
ОткрытьТаблицу	Перевозки; Таблица; Только ч	Открытие таблицы "Перевозки"
СдвигРазмер	10см; 0см; 15см; 15см	Задание ширины и высоты окна

Аргументы макрокоманды

Имя таблицы	Перевозки
Режим	Таблица
Режим данных	Только чтение

4.

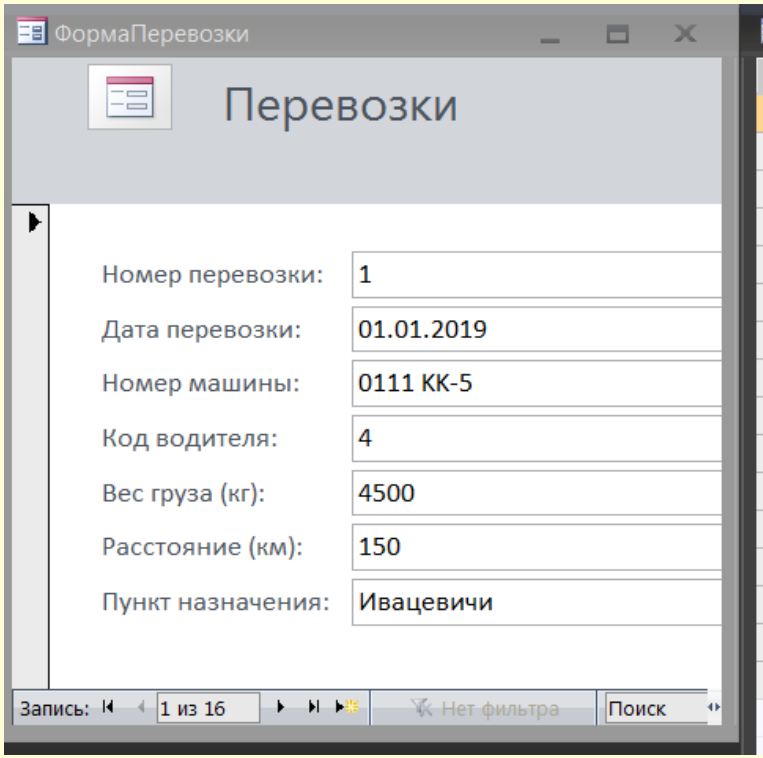
Макрокоманда	Аргументы	Примечание
ОткрытьФорму	ФормаПеревозки; Форма; ; ; И	Открытие формы "Перевозки"
СдвигРазмер	0см; 0см; 10см; 10см	Задание ширины и высоты окна
ОткрытьТаблицу	Перевозки; Таблица; Только ч	Открытие таблицы "Перевозки"
СдвигРазмер	10см; 0см; 15см; 15см	Задание ширины и высоты окна

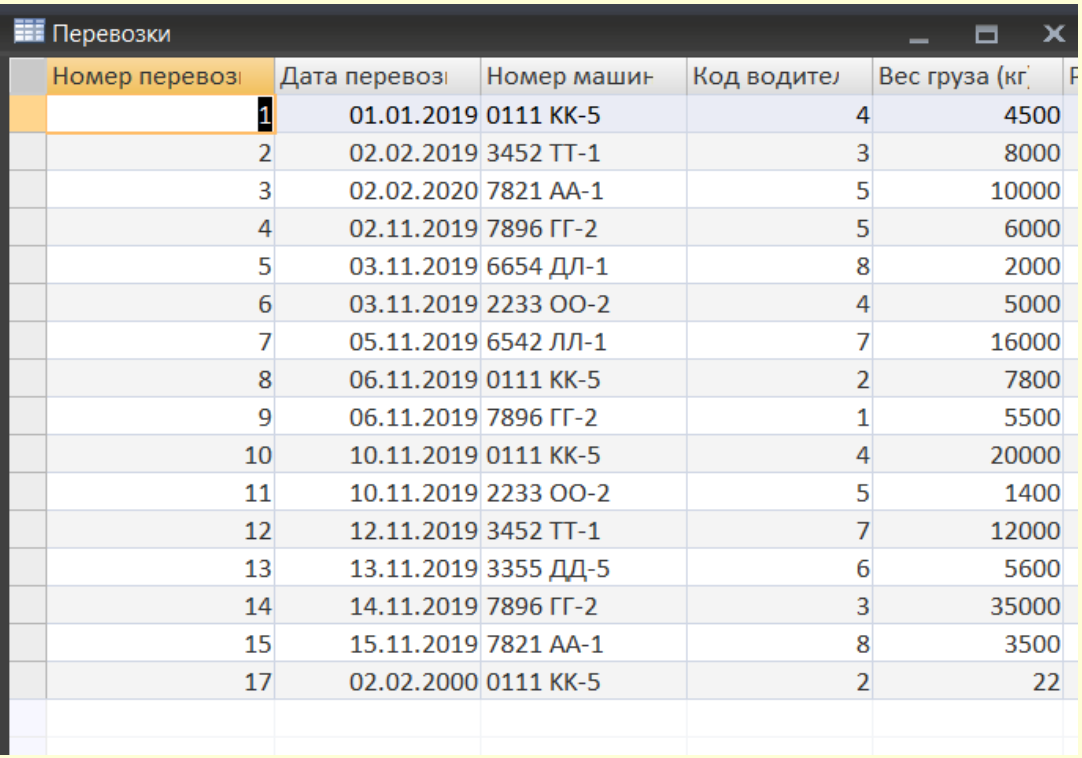
Аргументы макрокоманды

По правому краю	10см
От верхнего края	0см
Ширина	15см
Высота	15см

Сдвиг и изменение размеров активного окна. При

Результат выполнения:





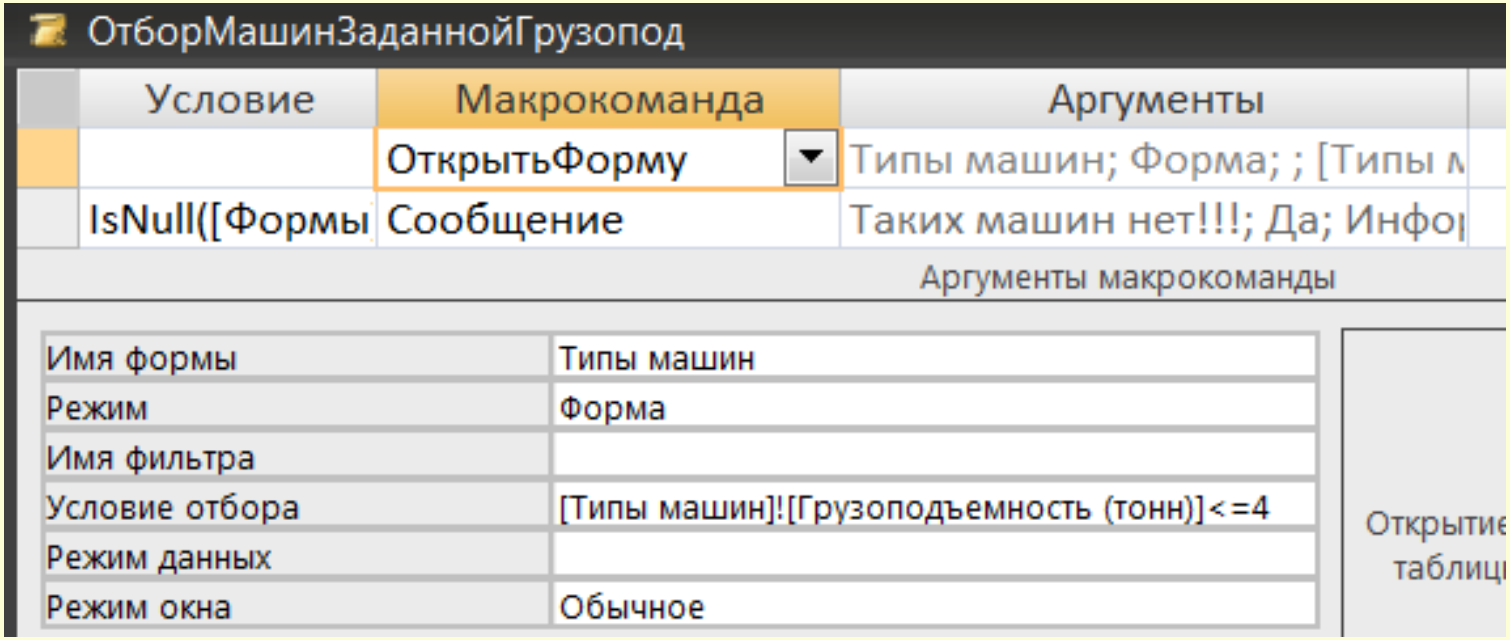
Перевозки

Номер перевоз	Дата перевоз	Номер машин	Код водите	Вес груза (кг
1	01.01.2019	0111 КК-5	4	4500
2	02.02.2019	3452 ТТ-1	3	8000
3	02.02.2020	7821 АА-1	5	10000
4	02.11.2019	7896 ГГ-2	5	6000
5	03.11.2019	6654 ДЛ-1	8	2000
6	03.11.2019	2233 ОО-2	4	5000
7	05.11.2019	6542 ЛЛ-1	7	16000
8	06.11.2019	0111 КК-5	2	7800
9	06.11.2019	7896 ГГ-2	1	5500
10	10.11.2019	0111 КК-5	4	20000
11	10.11.2019	2233 ОО-2	5	1400
12	12.11.2019	3452 ТТ-1	7	12000
13	13.11.2019	3355 ДД-5	6	5600
14	14.11.2019	7896 ГГ-2	3	35000
15	15.11.2019	7821 АА-1	8	3500
17	02.02.2000	0111 КК-5	2	22

Пример3. Создать макрос, позволяющий отображать в форме “Типы машин” только машины с грузоподъемностью не более 4-ёх тонн.

Порядок выполнения:

В окне Конструктора макросов проектируем макрос с УСЛОВИЕМ:

1. 

2.

	Условие	Макрокоманда
		ОткрытьФорму
	IsNull([Формы]![Типы машин]![Тип автомобиля])	Сообщение

Аргументы макрокоманды

Сообщение	Таких машин нет!!!
Сигнал	Да
Тип	Информационное
Заголовок	Сообщение

Вывод окна, со

3.

	Условие	Макрокоманда
		ОткрытьФорму
	IsNull([Формы]![Типы машин]![Тип автомобиля])	Сообщение
	...	Закрыть
	...	ОстановитьМакрос
		ПрименитьФильтр

Аргументы макрокоманды

Тип объекта	Форма
Имя объекта	Типы машин
Сохранение	Подсказка

4.

Условие	Макрокоманда
	ОткрытьФорму
IsNull([Формы]![Типы машин]![Тип автомобиля])	Сообщение
...	Закрыть
...	ОстановитьМакрос
	ПрименитьФильтр

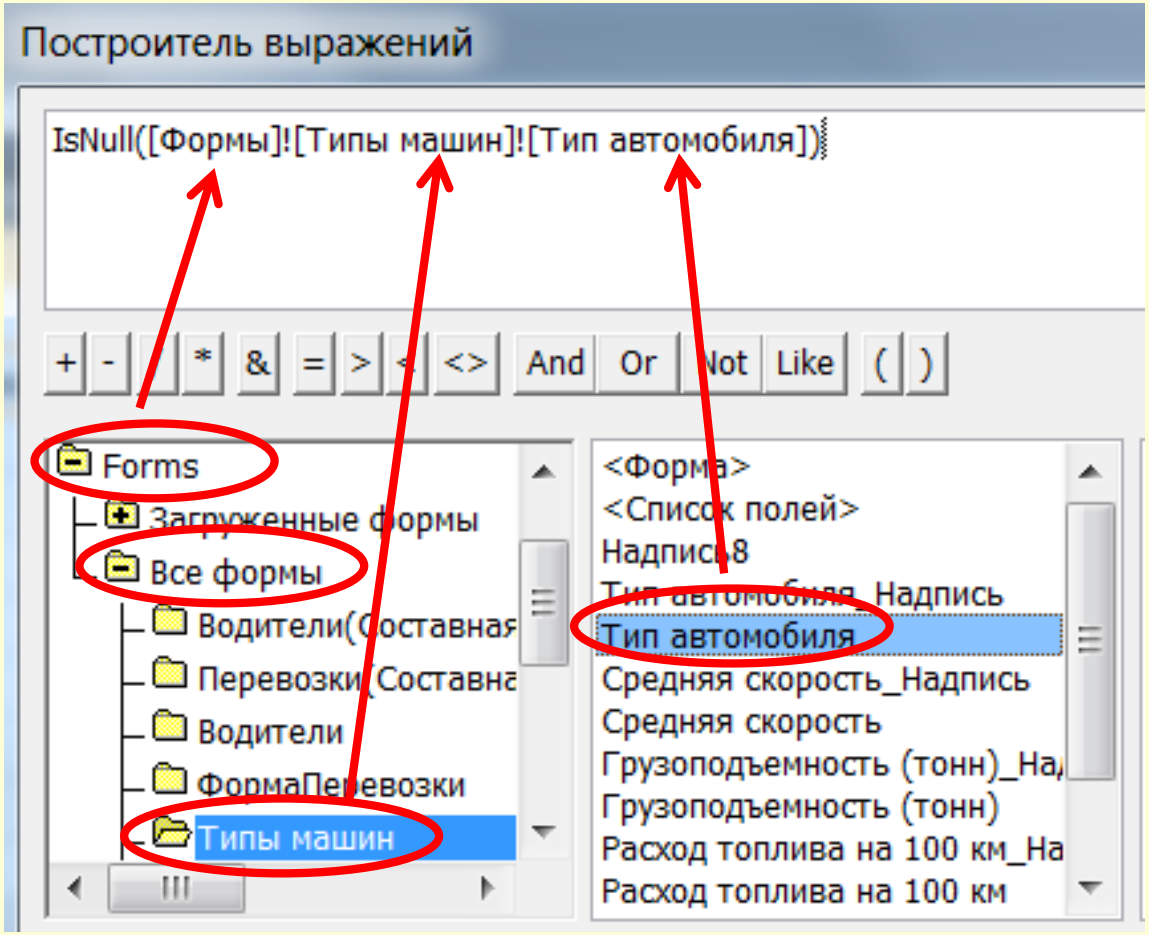
5.

Условие	Макрокоманда
	ОткрытьФорму
IsNull([Формы]![Типы машин]![Тип автомобиля])	Сообщение
...	Закрыть
...	ОстановитьМакрос
	ПрименитьФильтр

Аргументы макрокоманды

Имя фильтра	
Условие отбора	[Типы машин]![Грузоподъемность (тонн)]<=4
Имя элемента	

Замечание1: для ввода выражения в столбец ***Условие*** окна Конструктора макроса или в аргумент ***Условие отбора*** выбранной макрокоманды можно пользоваться **Построителем выражений**.



Замечание2: многоточие в столбце **Условие** задается для тех макрокоманд, выполнение которых должно следовать в случае ***истинности*** условия, заданного в макрокоманде накануне.

Макросы, связанные с событиями

1. Макросы, связанные с событиями ЭУ в форме

Пример4. Создать ***<u>групповой</u>*** макрос, отдельные макросы которого позволяют закреплять действие за кнопками формы “Надбавки” – просматривать сведения о водителях разных категорий или всех одновременно.

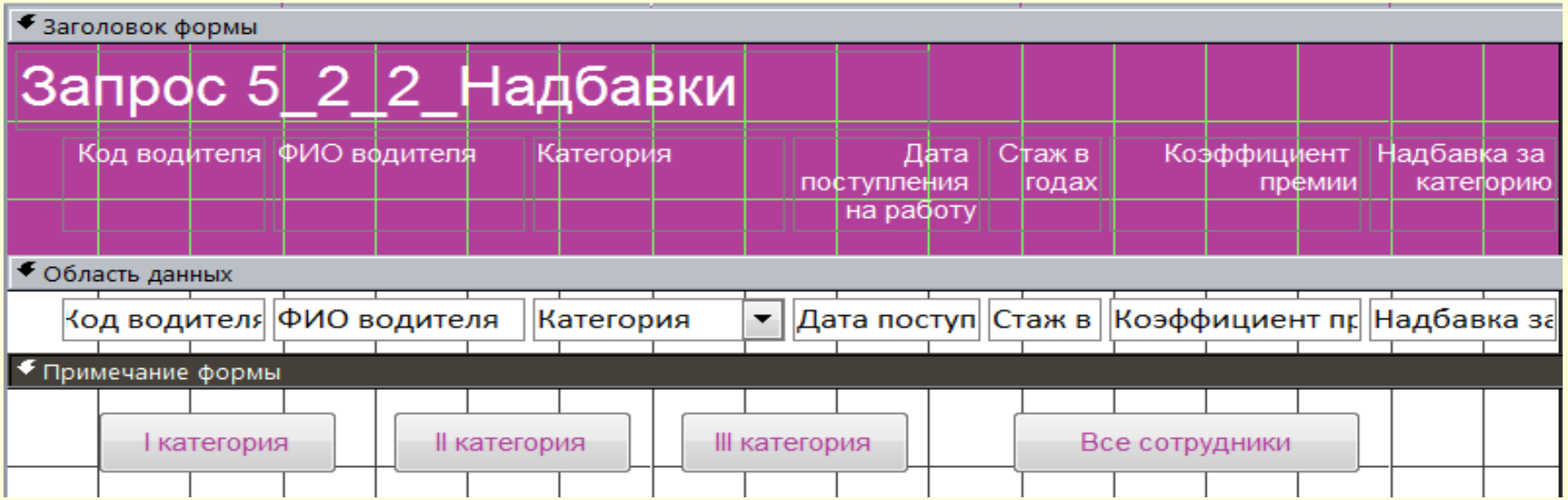
<u>Порядок выполнения:</u>

1. С помощью ***Мастера форм*** создаем ленточную форму для запроса “5_2_2_Надбавки”;

2. Открываем созданную форму в режиме *Конструктора*.

3. Выключаем на панели ЭУ (если она включена) кнопку “Мастера” (в MsA 2003) или кнопку на вкладке ленты **Конструктор** в группе инструментов **Элементы управления** (в MsA 2007/2010) 

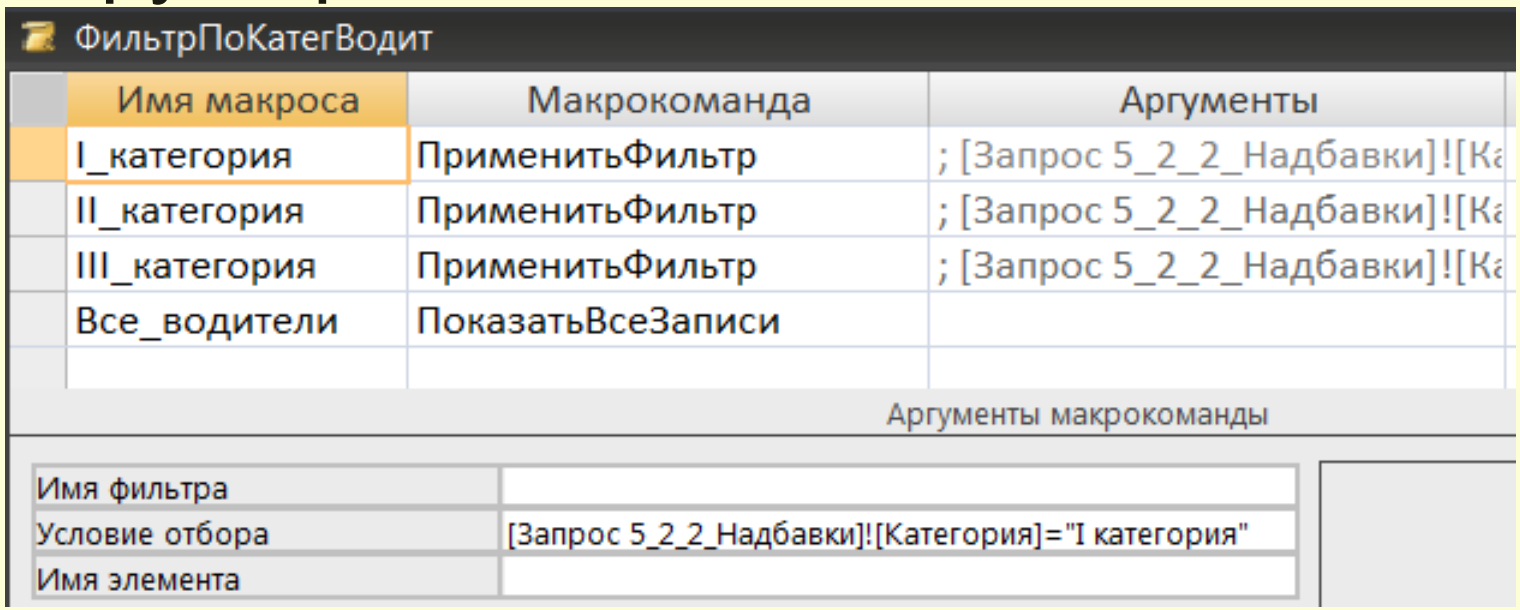
4. Вставляем четыре кнопки в раздел «Примечание формы», пользуясь ЭУ «Кнопка», изменив по ходу добавления их подписи либо непосредственно щелкнув в область ЭУ, либо задав нужные текстовые выражения в свойстве «Подпись» в окне свойств выделенного ЭУ на вкладке **Все**.



5. Создаем макрос в режиме *Конструктора*:

5.1. добавляем столбец “Имя макроса” с помощью команды ***Имена макросов*** в ***п.м. Вид*** (в MsA 2003) или с помощью кнопки на вкладке ленты **Конструктор** в группе инструментов **Показать или скрыть** (в MsA 2007/2010)

5.2. вводим следующие макросы в окно Конструктора:

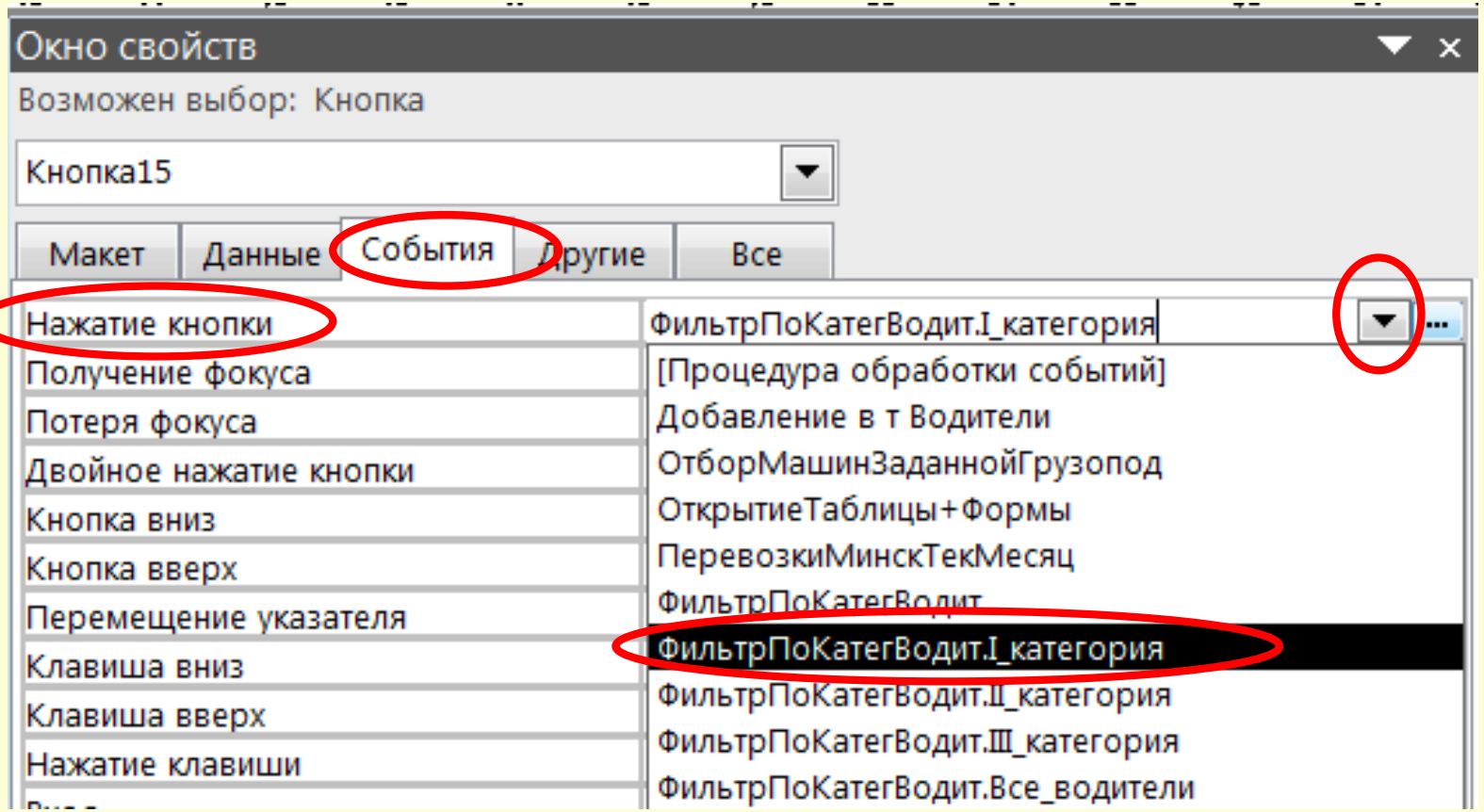


6. Связываем макрос с событием ЭУ “Кнопка”:

6.1. открываем форму, созданную по запросу “5_2_2_Надбавки” в режиме *Конструктора*;

6.2. открываем окно **Свойств** для кнопки «I категория»;

6.3. выбираем на вкладке **События** нужное имя макроса в свойстве «Нажатие кнопки»:



6.4. связь остальных макросов выполняется аналогично.

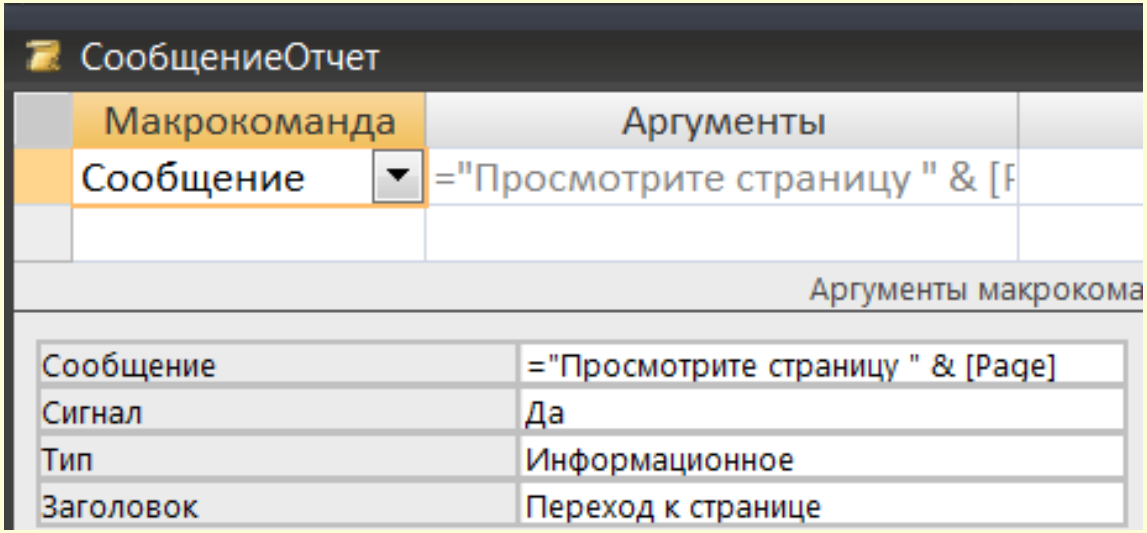
7. Проверка работы макроса выполняется в режиме ***Просмотра формы*** после нажатия каждой кнопки в отдельности!!!

2. Макросы, связанные с событиями отчета

Пример5. Создать макрос, позволяющий при просмотре отчета “Перевозки” в случае перехода к новой странице выводить сообщение, содержащее номер этой страницы.

Порядок выполнения:

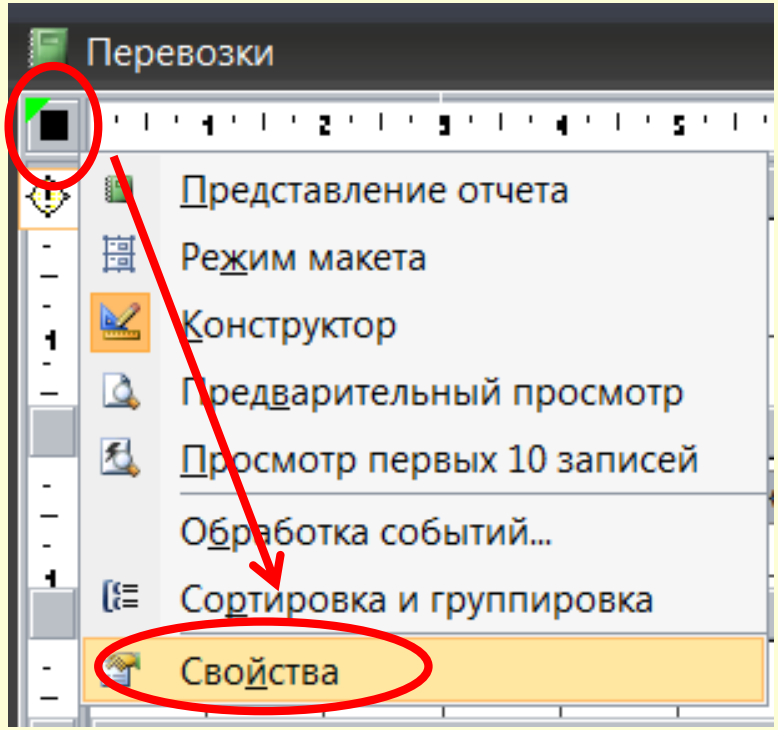
1. Создаем макрос в режиме *Конструктора*:



2. Связываем макрос с событием отчета:

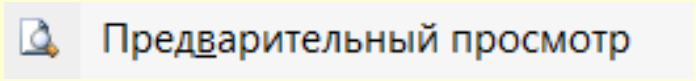
2.1. открываем отчет, созданный по таблице “Перевозки” в режиме *Конструктора*;

2.2. открываем окно **Свойств** отчета (командой из контекстного меню по кнопке на пересечении вертикальной и горизонтальной линеек);

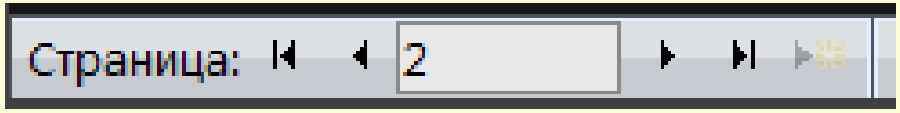


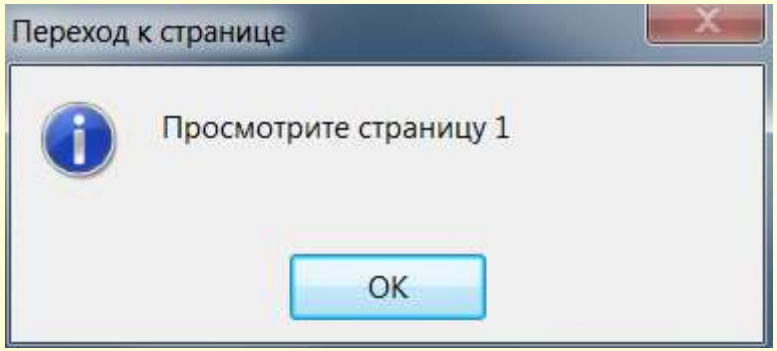
2.3. выбираем на вкладке **События** нужное имя макроса в свойстве «Страница»

3. Проверка работы макроса выполняется в режиме ***Предварительного просмотра отчета***



после нажатия кнопки перехода на новую страницу





Создание управляющей (кнопочной) формы

Обычно, когда пользователю БД нет необходимости просматривать всю структуру базы, для работы с приложением MsA создаётся специальная форма, которая носит название ***Главная кнопочная форма*** (Main Switchboard) и содержит кнопки, позволяющие выбирать основные функции или подсистемы приложения.

Главная кнопочная форма создается с целью навигации по базе данных, т.е. она может использоваться в качестве главного меню БД.

Кнопочная форма представляет собой страницу, содержащую кнопки, которые позволяют открывать другие страницы, таблицы, запросы, диалоговые окна и формы, просматривать и печатать отчеты и выполнять другие действия.

Такую форму можно создать, как обычно, с помощью *Конструктора* форм, а можно использовать специальный мастер, который называется *Диспетчер кнопочных форм* (Swithboard Manager).

Порядок создания кнопочной формы:

1. Создать пустую (т.е. без указания источника данных) форму в режиме *Конструктора*.

2. Добавить в область данных текст заголовка с помощью ЭУ «Надпись», изменив при необходимости её оформление (цвет шрифта, фона, границы, размеры и т.д.)

3. На Панели ЭУ активизировать кнопку «Мастера».

4. Далее добавлять на форму необходимые ЭУ: кнопки, вкладки, поля со списком, и т.д.

5. Сохранить форму.

Пример6: добавить ЭУ «Кнопка» для открытия формы.

а) выбрать ЭУ «Кнопка» на панели ЭУ и начертить с помощью мыши прямоугольник в области данных – загрузится **Мастер создания кнопок**.

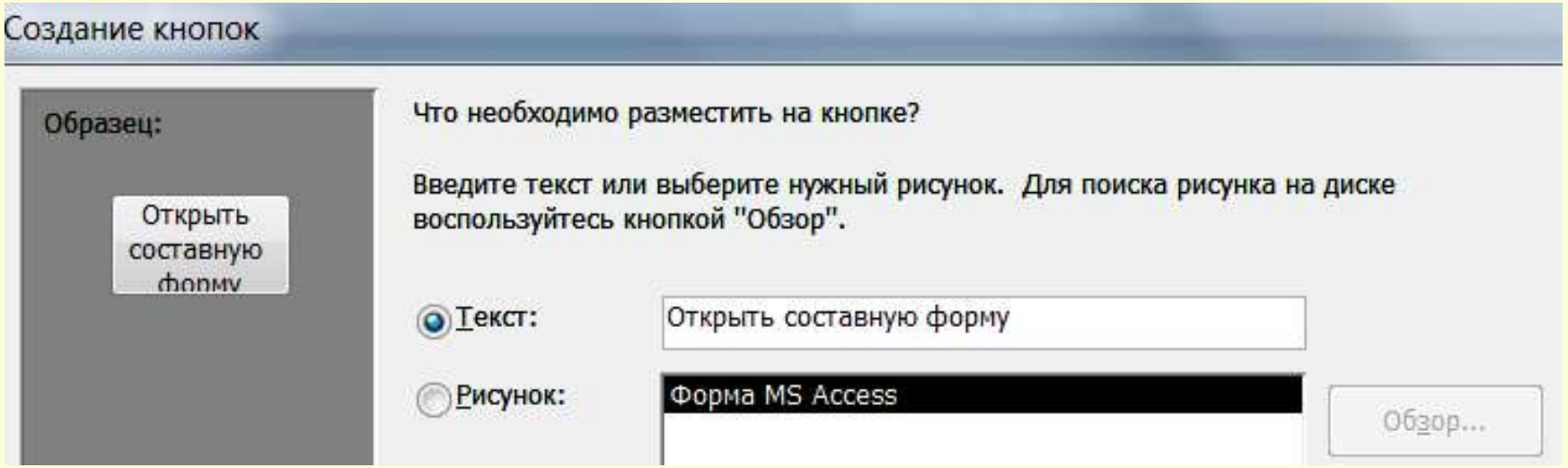
б) в окне «Создание кнопок» выполнить следующие действия:

на 1-м шаге выбрать необходимую опцию из списка *Категории* «Работа с формой», а из списка *Действия* – конкретное действие «Открыть форму», нажать кнопку **Далее**;

на 2-м шаге выбрать открываемый объект, например, составную (многотабличную) форму, нажать кнопку **Далее**;

на 3-м шаге выбрать один из двух вариантов отображения данных в форме – *открыть форму для отобранных записей* или *с выводом всех записей*, нажать кнопку **Далее**;

на 4-м шаге выбрать, что будет размещено на кнопке: текст или рисунок, нажать кнопку **Далее**;



на 5-м шаге выбрать имя для кнопки, нажать кнопку **Готово**.

Замечание1: Можно упорядочить и красиво размещать кнопки на форме (предварительно выделив их всех или их часть при нажатой клавише **SHIFT**) с помощью команд выравнивания (по верхнему краю, по нижнему краю и пр.), и команд определения размера (по самому высокому, по самому широкому и пр.), которые можно найти в контекстном меню для выделенных объектов или

- из ***п.м. Формат → Выровнять / Размер*** (в MsA 2003);
- в группе инструментов ***Выравнивание / Размер*** на вкладке ленты **Упорядочить** (в MsA 2007/2010).

Замечание2: в Мастере создания кнопок может не быть обработок некоторых действий (например, открытие таблицы), поэтому необходимо создавать для этого макрос, который затем можно найти в категории "Разное" в ***Мастере*** для выбора действия, выполняемого при ***нажатии кнопки***.

Глава 8.

Создание и работа с БД посредством графического интерфейса СУБД.

Технологии работы с базой данных в MS Access.

§ 8.6. Автоматизация работы с БД в MsA: модули VBA

Определение модуля, как объекта БД

«Модуль» – это объект, который включает программы, написанные на языке программирования Visual Basic, которые могут разрабатываться пользователем для реализации нестандартных процедур при создании приложения.

Модуль состоит из процедур – совокупностей команд языка ***Visual Basic for Application*** (VBA).

Знакомство с VBA

Visual Basic для приложений (VBA – Visual Basic for Applications) является инструментальным средством разработки приложений в среде основных компонентов Microsoft Office (Word, Excel, PowerPoint, Access, FrontPage, Outlook) и может использоваться именно как средство разработки приложений, а не только в качестве инструмента настройки пользовательского интерфейса и редактирования макросов.

Объекты и семейства VBA

VBA является **объектно-ориентированным языком**, предоставляющим возможности визуального программирования и содержит иерархию объектов, каждому из которых соответствует свой набор ***методов*** и ***свойств***.

Объекты представляют собой фундаментальные «строительные» блоки – почти все, что делается в среде VBA, включает модификацию объектов.

Т.о., **объект (object)** – это программный элемент, который имеет свое отображение на экране, содержит некоторые переменные, определяющие его свойства, и некоторые методы для управления объектом.

Класс (class) представляет собой описание совокупности однотипных объектов. Класс как новый тип данных, позволяющий создавать новые переменные этого типа - объекты (экземпляры класса).

Свойство (property) – это отдельная характеристика объекта или класса. Свойство объекта может принимать определенное значение.

Метод (method) представляет собой процедуру (или функцию) объекта или класса. Методы определяют «поведение» объекта.

Событие (event). Объект может реагировать на определенные события, происходящие в процессе работы приложения и влияющие на объект. Реакцией объекта на произошедшее событие может быть выполнение объектом некоторых заданных действий – выполнение специальной процедуры, которая называется **процедурой обработки события.**

Объект, например форма, после получения сообщения о событии приступает к обработке события (реакция объекта на событие).

Каждый объект может реагировать только на предварительно зафиксированные события.

Любому событию объекта может быть назначена некоторая процедура его обработки!!!

Понятие СОБЫТИЯ и перечень некоторых событий приведен в § 7.4. Лекции 11.

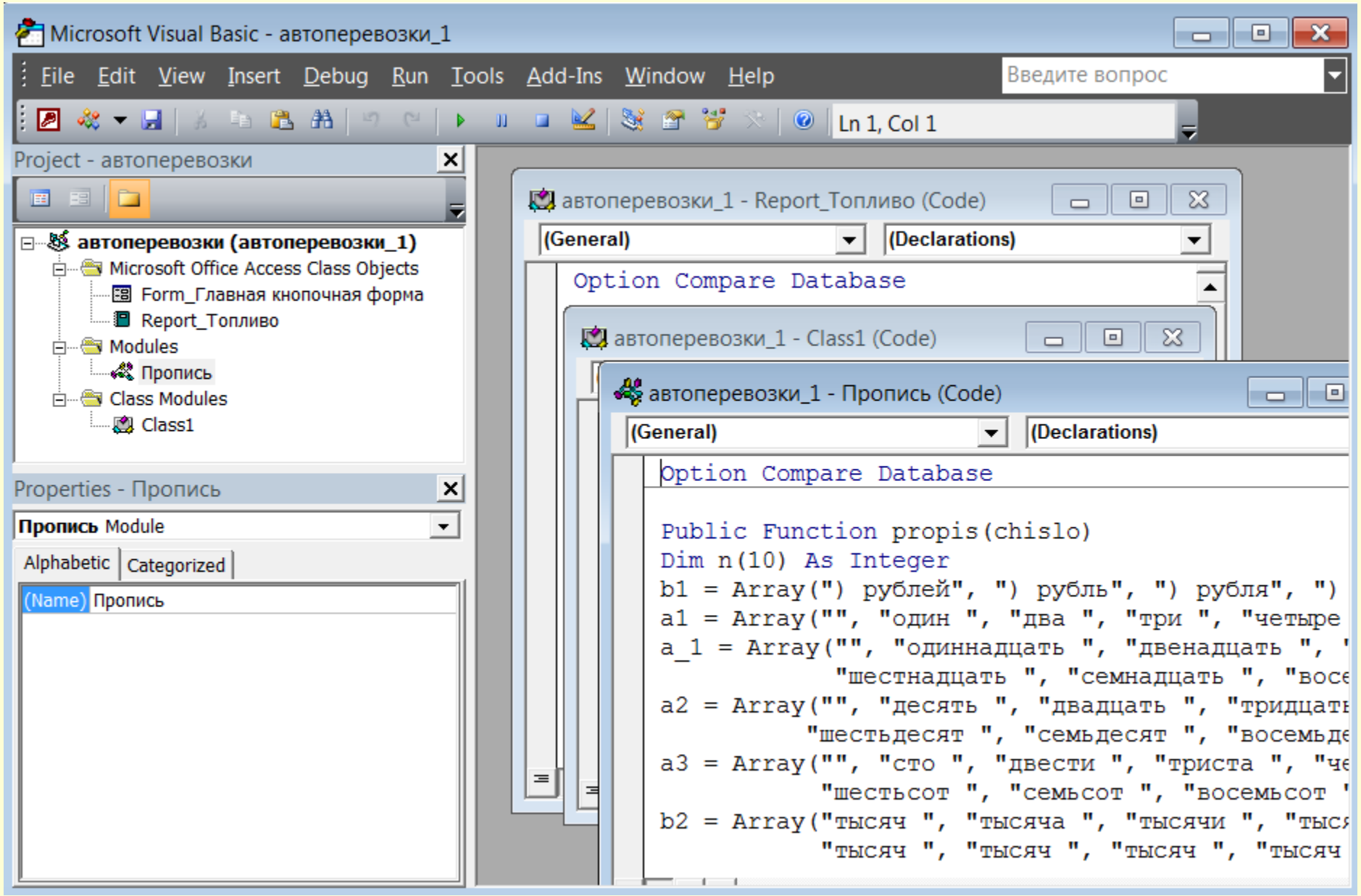
Семейством (collection) называется упорядоченный набор однотипных объектов – экземпляров одного класса. Семейство тоже является объектом.

Объектной моделью (object model) называется совокупность взаимосвязанных объектов, описывающих программную систему.

Инструментальная среда VBA

В VBA языком программирования является Visual Basic, а инструментальная среда программирования реализована в виде **редактора** Visual Basic (VB), который является отдельным приложением и может активизироваться из любого приложения MS Office клавишами Alt+F11.

Также редактор VB активизируется из приложения MsA в процессе создания модуля или просмотра модулей.


Microsoft Visual Basic - автоперевозки_1
File Edit View Insert Debug Run Tools Add-Ins Window Help
Введите вопрос
Ln 1, Col 1
Project - автоперевозки
автоперевозки (автоперевозки_1)
Microsoft Office Access Class Objects
Form_Главная кнопочная форма
Report_Топливо
Modules
Пропись
Class Modules
Class1
Properties - Пропись
Пропись Module
Alphabetic
Categorized
(Name) Пропись
автоперевозки_1 - Report_Топливо (Code)
(General)
(Declarations)
Option Compare Database
автоперевозки_1 - Class1 (Code)
автоперевозки_1 - Пропись (Code)
(General)
(Declarations)
Option Compare Database
Public Function propis(chislo)
Dim n(10) As Integer
b1 = Array(") рублей", ") рубль", ") рубля", ")
a1 = Array("", "один ", "два ", "три ", "четыре
a_1 = Array("", "одиннадцать ", "двенадцать ",
"шестнадцать ", "семнадцать ", "боси
a2 = Array("", "десять ", "двадцать ", "тридцать
"шестьдесят ", "семьдесят ", "восемьде
a3 = Array("", "сто ", "двести ", "триста ", "че
"шестьсот ", "семьсот ", "восемьсот
b2 = Array("тысяч ", "тысяча ", "тысячи ", "тыся
"тысяч ", "тысяч ", "тысяч ", "тысяч

В **проекте** MsA могут храниться такие объекты, как формы, отчеты, макросы и модули, ссылки на страницы доступа к данным, а также ссылки на объекты, хранящиеся в БД на SQL-сервере (таблицы, представления, диаграммы БД и хранимые процедуры).

Т.е. проект содержит все связанные с приложением объекты, относящиеся к конкретной БД, причём проект сохраняется в одном файле с самой БД.

Модули VBA

Код VBA в приложении MsA содержится в модулях. Модули являются объектами Access, такими же, как таблицы, запросы, формы, отчеты, страницы и макросы, о чем свидетельствует ярлык на панели объектов в окне БД.

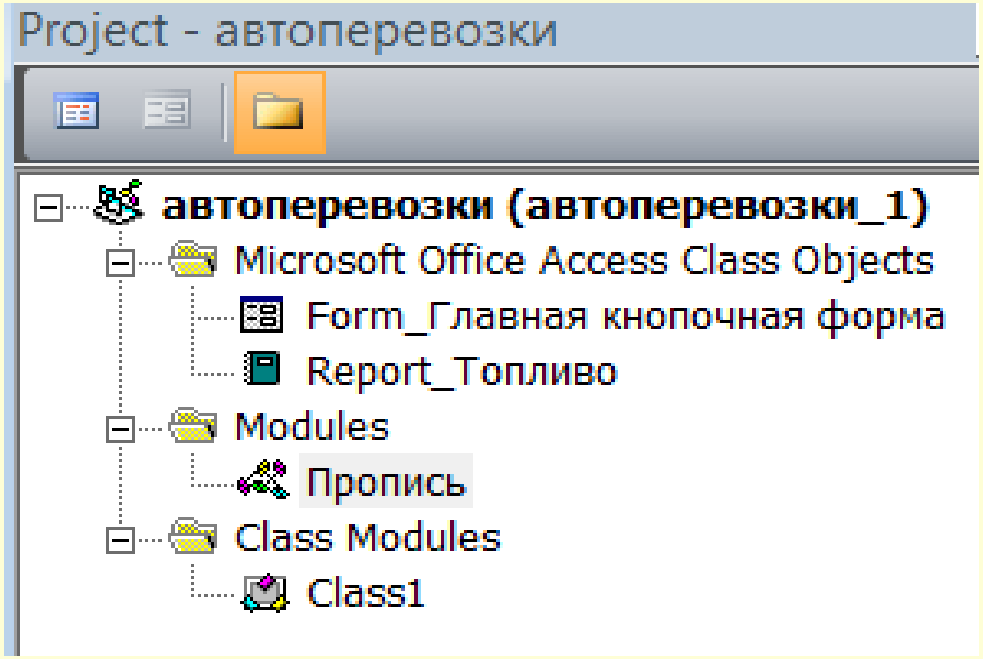
Основное содержание модулей – это процедуры на языке VBA.

Существуют два типа модулей:

- ***стандартные модули;***
- ***модули класса.***

В окне Project редактора VB объекты приложений делятся на три группы:

- ***Microsoft Access Class Objects*** – объекты классов Access, которые включают модули форм и отчетов (модули объектов);
- ***Modules*** – стандартные модули;
- ***Class Modules*** – модули классов или модули пользовательских классов.



***Модуль объекта* (формы, отчета)** – это модуль класса, содержащий программы всех процедур обработки событий, возникающих в конкретном объекте (форме, отчете) или в его элементах управления (ЭУ).

Модуль формы НЕ создается сразу при создании новой формы. Он создается и связывается с формой, как только производится попытка создать первую процедуру обработки событий для этой формы или одного из ЭУ формы (или в случае нажатия кнопки *Программа* (*Code*) в окне Конструктора формы.

Стандартные модули содержат общие процедуры, которые могут использоваться в разных местах приложения: при обработке событий в разных объектах, для вычисления значений в разных запросах или формах, вызываться из других модулей и т.д. Эти процедуры НЕ связаны с конкретным объектом (формой, отчетом).

Стандартные модули могут использоваться другими приложениями MsA, т.к. в общих процедурах нет ссылок на конкретные объекты данного приложения (формы, отчеты).

Стандартные модули могут использоваться для объявления глобальных переменных, констант, типов и внешних процедур.

Модуль класса отличается от стандартного модуля тем, что, кроме процедур, он содержит описание объекта и используется для создания классов (объектов).

Отдельные модули класса, расположенные на вкладке **Модули** окна БД, содержат описание класса (объекта), созданного пользователем.

Модуль – это именованная единица, состоящая из одной или нескольких процедур и раздела описаний, в котором объявляются переменные, константы и пользовательские типы данных, а также устанавливаются параметры компилятора.

Создание модуля в VBA:

1. Создание **стандартного** модуля (или модуля класса):

в версии MsA 2003:

1 способ: в окне БД на вкладке **Модули** нажать кнопку **Создать**;

2 способ:

- открывается редактор VB: ***п.м. Сервис ⇒ Макрос ⇒ Редактор VB (или ALT+F11);***
- создается новый модуль для текущей БД: ***п.м. Insert ⇒ Module (Class Module)*** ;

в версии MsA 2007/2010:

- командой ***Модуль*** (***Модуль класса***) на вкл. ленты **Создание** в группе инстр-ов **Другие** в списке команд **Макрос**

2. Создание ***модуля формы (отчета):***

1 способ: в режиме *Конструктора* формы (или отчета) выбрать свойство "Да" в поле **«Наличие модуля»** на вкладке "Все" в окне свойств объекта Форма (или Отчет);

2 способ:

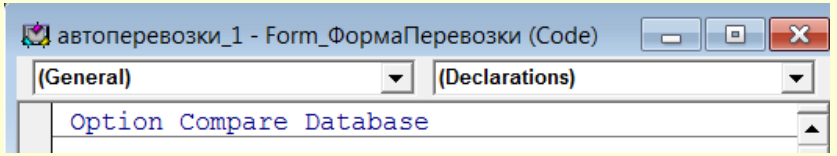
в версии MsA 2003:

кнопкой "Программа" на панели инструмен-тов в режиме *Конструктора* формы/отчета;

в версии MsA 2007/2010:

командой ***Просмотреть код*** на вкл. ленты **Конструктор** в группе инструментов **Сервис.**

Замечание: После создания нового модуля в области объявления глобальных переменных окна редактора кода по умолчанию отображается верхняя строка **Option Compare Database**, которая указывает на то, что для сравнения текстовых строк используется параметр БД. Для модуля приложения может использоваться один из способов сравнения (option compare) строк: binary; text; database.



Возможно добавление строки **Option Explicit** (явное объявление переменных) – эта директива запрещает использование необъявленных переменных. В этом случае каждая переменная должна объявляться до ее использования.

```
'выше глобальных переменных и констант могут быть только декларации:
Option Explicit      'принудительное объявление переменных
Option Base 1        'нижняя граница объявляемых массивов начинается с 1
Option Compare Text 'сравнение текста без учета регистра
'глобальная переменная - первой строкой, выше всех процедур
Public MyVariable As String
'далее процедуры и функции
Sub main()
```

Состав программного кода VBA:

Переменные – используются для сохранения значений величин, изменяющихся в процессе выполнения программы.

Каждая переменная имеет имя, по которому к ней обращаются. Присвоение значения для переменной осуществляется посредством оператора присваивания. В левой части оператора присваивания имя переменной, а в правой – значение или выражение:

X = 200 * 0.8 / 70

Y = "Петров Семен Иванович"

Z= X / 80+30

Константы – служат для сохранения некоторых постоянных значений.

Комментарии – служат для краткого описания по тексту программы:

‘ комментарии

Замечание: В языке VB действуют следующие соглашения на имена процедур, переменных и констант:

- должны начинаться с буквы;
- могут включать буквы, цифры и символы подчеркивания;
- не должны включать знаки препинания или пробелы;
- не должны совпадать с ключевыми словами языка VB.

Объявление констант:

[Private | Public] Const <Имя_конст> As <тип = значение>

Объявление переменных:

Dim <Имя переменной> [As <Тип данных>]

Замечание: Служебные слова ***Private*** и ***Public*** задают область видимости объектов:

- **Private** делает объект доступным только внутри данного модуля;
- **Public** делает объект доступным из другого модуля.

Типы данных, которые можно хранить в переменных VBA:

Тип данных	Занимает байт в памяти	Пределы значений
Byte	1	Целые числа от **0** до **255**
Boolean	2	**True** или **False**
Integer	2	Целые числа от (**-32768**) до **32767**
Long	4	Целые числа от (**-2147483648**) до **2147483647**
Single	4	От (**–3.402823E+38**) до (**–1.401298E-45**) и от **1.401298E-45** до **3.402823E+38**
Double	8	От **±1.79769313486232E+308** до **±4.94065645841247E-324**
Decimal	12	От **±79228162514264337593543950335** без десятичных знаков до **±7,9228162514264337593543950335** с 28-ю знаками после запятой
Currency	8	От (**–922337203685477.5808**) до **922337203685477.5807**
Date	8	От **01.01.100** до **31.12.9999**(не надо путать с датами в Excel - 01.01.1900 до 31.12.2078)
String	10(+длина строки)	От **0** до **65400** символов для фиксированных строк и чуть более 2 млрд. для строк переменной длины
Object	4	Любой объект
Array	Определяется кол-вом и размером элементов	
Variant	от 16-ти	Любой из встроенных типов данных

Массив – это переменная, в которой хранится одновременно несколько значений одинакового типа. Формальное определение массива таково: он представляет собой совокупность однотипных индексированных переменных.

Прежде чем использовать массив, нужно обязательно объявить его с помощью оператора Dim и указать при этом тип хранящихся в массиве значений.

!!! Все значения в массиве обязаны принадлежать к одному типу данных.

Синтаксис оператора объявления массива:

Dim <*имяМассива*> (<*размер1*>,<*размер2*>,...) **As** <*типДанных*>

где указанные в скобках величины <*размер1*>, <*размер2*> и т.д. задают размеры массива – количество индексов и максимально допустимое значение для каждого конкретного индекса. При этом индексирование элементов массива по умолчанию начинается с нуля.

Так, объявление

Dim Array1 (9) As Integer

определяет одномерный массив из 10 элементов, являющихся переменными *целого* типа, а объявление

Dim Array2 (4, 9) As Variant

определяет двумерный массив из пятидесяти (5x10) элементов, являющихся переменными универсального типа *Variant*.

Ссылки на объекты

Кроме обычных переменных, в VB часто встречаются переменные, представляющие собой ссылку на объект. Применение переменной-объекта отличается от использования обычных переменных: нужно не только объявить такую переменную, но и назначить ей соответствующий объект с помощью специального оператора **SET**. Синтаксис этого объявления и назначения:

Dim <имяПеременной> As Object

Set <имяПеременной> = <ссылкаНаОбъект>

Пример:

```
Dim MyBase As Database
Set MyBase = CurrentDb()
```

Оператор – это наименьшая единица VBA – кода, предназначен для определения переменной, установки параметров или выполнения какого-либо действия в программе.

Процедура – это отдельная единица программного кода VBA, которую можно вызвать по имени для выполнения определенных действий.

Процедура может иметь параметры и в результате выполнения последовательности инструкций изменять их значения.

Понятие и типы процедур. Порядок создания.

Процедура – это самостоятельная замкнутая программная единица, включающая совокупность операторов описания локальных данных процедуры и операторов, которые выполняются в ней. *Процедура обрабатывает событие, если ее результат – это реакция на какое-либо действие (например, щелчок мыши).*

Виды процедур:

- ***Подпрограммы*** (Sub)
- ***Функции*** (Function)

Процедура-подпрограмма выполняет действия, но не возвращает значение.

Замечание: Если подпрограмма не содержит параметров, то ее оператор должен включать пустой набор круглых скобок. Существуют два типа процедур - процедуры свойств и процедуры обработки событий. Процедуры, которые можно связать с выполнением самых различных событий, например, с открытием формы или отчета, со щелчком мышью по кнопке в форме и так далее, называют **процедурами обработки событий**.

Синтаксис процедуры-подпрограммы:

Sub *<имяПроцедуры>(<аргумент1>,<аргумент2>,…)*
<оператор_1>
<оператор_2>
…
End Sub

Пример1 (процедуры-подпрограммы): текст макроса, сформированного мастером построения ЭУ «Кнопка» в режиме Конструктора формы:

```
Private Sub Кнопка1_Click()
On Error GoTo Err_Кнопка1_Click
   Dim stDocName As String
   Dim stLinkCriteria As String
   stDocName = ChrW(1058) & ChrW(1080) & ChrW(1087) & ChrW(1099) & ChrW(32) & ChrW(1084) &
ChrW(1072) & ChrW(1096) & ChrW(1080) & ChrW(1085)
   DoCmd.OpenForm stDocName, , , stLinkCriteria
Exit_Кнопка1_Click:
   Exit Sub
Err_Кнопка1_Click:
   MsgBox Err.Description
   Resume Exit_Кнопка1_Click
End Sub
```

Процедура-функция либо вычисляет какое-либо значение и возвращает результат вычисления, либо тестирует что-либо, тогда в результате – ИСТИНА или ЛОЖЬ.

Синтаксис процедуры-функции:

Function
<имяФункции>(<аргумент1>,<аргумент2>,…)
<оператор_1>
<оператор_2>
…
<имяФункции> = <возвращаемоеЗначение>
End Function

Замечание: Функция чаще всего имеет, по крайней мере, один аргумент, заключенный в круглые скобки. Позволяется задавать до 29 аргументов, разделяя их запятыми.

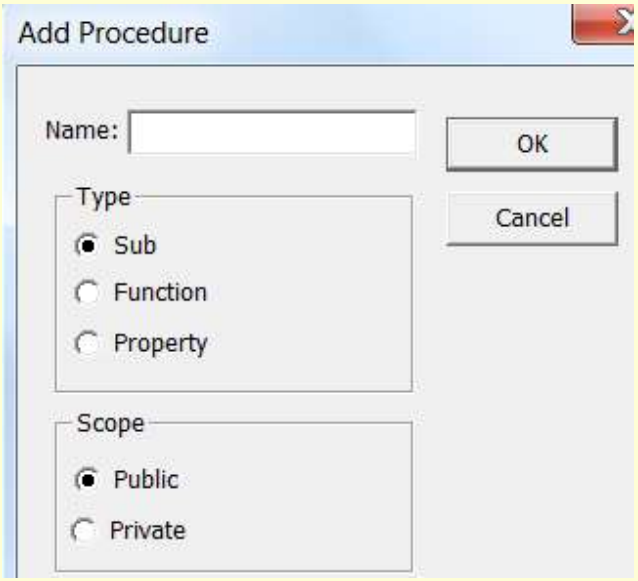
Пример2 (процедуры-функции): вычисление надбавки в зависимости от категории водителя:

```
Public Function NadbCateg(cat)
BV = 25
If cat = "I категория" Then
NadbCateg = 1 * BV
    ElseIf cat = "II категория" Then
    NadbCateg = 1.5 * BV
        Else
        NadbCateg = 2 * BV
End If
End Function
```

Порядок создания процедуры в модуле в редакторе VB:

1. выбрать команду ***п.м. Insert ⇒ Procedure;***
2. в появившемся диалоговом окне **Add Procedure**:

- задать в поле **Name** – *имя функции;*
- выбрать *тип процедуры* из возможных типов (Sub, Function, Property);
- определить *область видимости* из возможных вариантов (Private, Public);



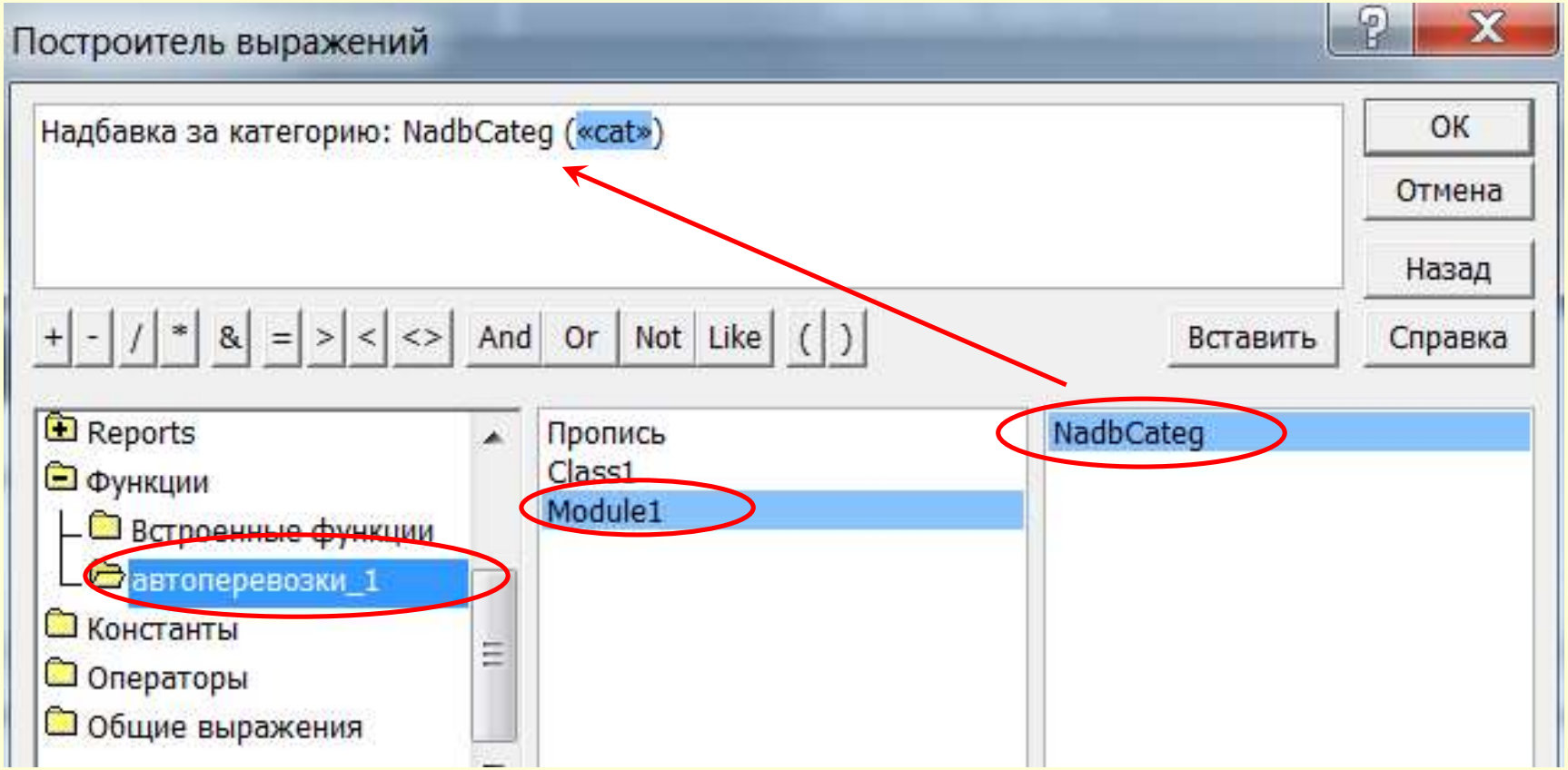
3. после выполнения п.2 в окне Code Window (окно просмотра исходного кода VBA) появятся ключевые слова начала и конца процедуры, в теле которых задать текст программы;

4. выполнить отладку кода: ***п.м. Debag*** $\Rightarrow$ ***Compile VBAProject***;

5. сохранить модуль вместе с файлом БД: ***п.м. File*** $\Rightarrow$ ***Save.***

Замечание1: Чтобы вернутся в приложение MsA, не закрывая редактор VB, используется комбинация клавиш ALT+F11.

Замечание2: Вариант использования функции, созданной в VBA – создание вычисляемого поля в запросе:



Управляющие конструкции языка VBA

Условный оператор If.,.Then...

– это общепринятый в языках программирования оператор управления вычислениями, который позволяет выбирать и выполнять действия в зависимости от истинности некоторого условия.

Имеется два варианта ***синтаксиса***:

- *в одну строку;*
- *в форме блока.*

Краткая форма оператора ***в одну строку***:

If <*условие*> **Then** <*оператор*>

Полная форма оператора ***в одну строку***:

If <*условие*> **Then** <*оператор1*> **Else** <*оператор2*>

В ***блочной*** форме ***краткое*** ветвление:

If <*условие*> **Then**
<*оператор1*>
<*оператор2*>
…
End If

Замечание: условие обязательно во всех вариантах и может быть числовым или строковым выражением.

В ***блочной*** форме ***полное*** ветвление имеет вид:

If <*условие1*> **Then**
<*БлокОператоров1*>
ElseIf <*условие2*> **Then**
<*БлокОператоров2*>
Else
<*ИначеОператоры*>
End If

Замечание: *БлокОператоров1* и *БлокОператоров2* – это последовательности из одного или нескольких операторов, по крайней мере одна из которых должна быть непустой.

Оператор выбора Select Case

– в зависимости от значения выражения выбирает и исполняет одну из последовательностей операторов.

Синтаксис:

Select Case *ПроверяемоеВыражение*
Case *СписокВыражений_1*
БлокОператоров_1
…
Case *СписокВыражений_n*
БлокОператоров_n
Case Else
ИначеОператоры
End Select

ПроверяемоеВыражение должно присутствовать обязательно и может быть произвольным выражением с числовым или строковым значением.

СписокВыражений представляет собой одно или несколько выражений, разделенных запятыми. При выполнении оператора проверяется, соответствует ли хотя бы один из элементов этого списка проверяемому выражению.

Эти элементы ***СпискаВыражений*** могут иметь одну из форм:

<выражение>

– задает отдельные значения,

<выражение1> To <выражение2>

– задает диапазоны значений,

Is <ЛогическийОператор> <выражение>

– задает области значений.

Пример:

```
Select  Case  Col_vo
    Case  50 To 99
    Skidka = 0.05
        Case  100
        Skidka = 0.06
            Case  Is > 100
            Skidka = 0.1
Case Else Skidka = 0
End Select
```

Встроенные функции языка VBA

Из категории арифметических функций:

Int(n) – возвращает целое число, меньшее или равное n как для положительных, так и для отрицательных чисел.

Fix(n) – возвращает целое число, меньшее или равное n для положительных чисел и большее или равное n для отрицательных чисел.

Например:

Int(99,8) = 99. Int(-99,8) = -100. Int(-99,2) = -100.

Fix(99,2) = 99. Fix(-99,8) = -99. Fix(-99,2) = -99.

Из категории функций обработки строк:

Asc(C) – возвращает **код** первого символа строки символов ***C*** по кодировке ASCII (т.е. каждый символ имеет собственный уникальный номер во внутренней памяти компьютера в виде числа)

Например: Asc(“A”)=65; Asc(“a”)=97; Asc(“Apple”)=65

Замечание: Чтобы просмотреть список кодов, которые распознает VBA, и соответствующих им символов, необходимо открыть справку в VBA и найти раздел Character sets

Chr(n) – возвращает **символ** соответствующий коду ***n*** ASCII (в качестве аргумента могут быть числа от 0 до 255)

Например: Chr(34) - это кавычки “

LTrim(C) – удаляет пробелы слева из строки символов ***C***

RTrim(C) – удаляет пробелы справа из строки символов ***C***

Mid(C, n1, n2) – возвращает ***n2*** символов из строки ***C***, начиная с n1 символа.

Например: Для извлечения подстроки из строки текста: если переменная C="холодная зима", то Mid(C,10,4) = "зима".

Left(C, n) – возвращает **n** символов с левого конца строки ***C***

Right(C, n) – возвращает **n** символов с правого конца строки ***C***

UCase(C) – преобразует строчные буквы в прописные

LCase(C) – преобразует прописные буквы в строчные

Len(C) – подсчитывает число символов в строке символов C

Из категории функций преобразования типов:

Val(C) – функция преобразования строки **C** в число.

Str(N) – функция преобразования числа N в строку.

Глава 9.

Введение в язык структурированных запросов SQL

Назначение языка SQL

Язык SQL (Structured Query Lanquaqe) – структурированный язык запросов для работы с базами данных (разработан фирмой IBM в начале 70-х гг XX столетия).

Язык SQL утвержден Американским национальным институтом стандартов (ANSI) и Международной организацией стандартизации (ISO) в качестве официального стандарта для реляционных БД и не зависит от специфики компьютера.

Существуют две ***формы*** языка SQL:

- **Интерактивный** SQL применяется непосредственно в БД для выполнения определенных действий над данными. Вводятся определенные команды, после выполнения которых тут же выводятся выходные данные (результат).
- **Встроенный** (программный) SQL – это включение команд языка SQL в программы, которые написаны на другом языке программирования.

Статический SQL – разновидность программного SQL, предназначенная для встраивания SQL-операторов в текст программы на языке программирования высокого уровня.

Динамический SQL – разновидность программного SQL, предназначенная для встраивания SQL-операторов в текст программы на языке программирования высокого уровня, допускающая динамическое формирование и выполнение запросов во время работы программы.

API-интерфейсы – разновидность программного SQL, основанная на использовании библиотек функций, разработанных для обеспечения связи прикладной программы с СУБД посредством выполнения SQL-запросов.

Язык SQL имеет определенный набор **команд**, которые позволяют осуществлять:

- организацию данных;
- изменение данных;
- чтение данных;
- управление доступом к данным;
- совместное использование данных;
- обеспечение целостности данных;
- обращение к базам данных в прикладных программах.

Структура SQL-команды

Команда – это инструкция, которая дается базе данных SQL. Каждая команда SQL начинается с ключевого слова – **глагола**, описывающего действие, выполняемое командой (*например*, CREATE – создать).

В команде может быть одно или несколько предложений.

Предложение описывает данные, с которыми работает команда, или содержит уточняющую информацию о действии, выполняемом командой.

Каждое предложение начинается с ключевого слова (*например*, WHERE – где).

Одни предложения в команде являются обязательными, а другие – нет.

Некоторые предложения могут содержать дополнительные ключевые слова, выражения.

Пример SQL-команды:

Выражения в SQL используются для выполнения операций над значениями, которые считаны из БД, или для выбора информации из БД.

Выражения представляют собой определенную последовательность полей, констант, функций, соединенных операторами.

Для указания конкретных значений данных используются ***константы***. Различают следующие виды констант:

- *константы с фиксированной запятой,*
- *константы с плавающей запятой,*
- *строковые константы (должны быть заключены в одинарные кавычки),*
- *отсутствующее значение (NULL).*

Соглашения по синтаксису команд:

[ ] – квадратные скобки – часть команды, которую при желании можно опустить;

{ } – фигурные скобки – означают, что конструкции, заключенные в эти скобки, должны рассматриваться как целые синтаксические единицы;

... – многоточие – указывает на то, что непосредственно предшествующая ему синтаксическая единица может повторяться один или более раз;

| – прямая черта – означает наличие выбора из двух или более возможностей.

Например, обозначение ASC|DESC указывает, можно выбрать один из терминов ASC или DESC;

Примечания:

1) Access не учитывает разрывы строк в инструкции SQL. Несмотря на это, каждое предложение рекомендуется начинать с новой строки, чтобы инструкцию SQL было удобно читать как тому, кто ее написал, так и всем остальным.

2) Каждая инструкция SELECT заканчивается точкой с запятой (;). Точка с запятой может стоять как в конце последнего предложения, так и на отдельной строке в конце инструкции SQL.

Типы операторов SQL

Команды SQL (инструкции, операторы) делят на следующие группы:

1. Язык определения данных (Data Definition Lanquaqe – DDL).

Он дает возможность создавать различные объекты БД и переопределять их структуру, *например*, создавать или удалять таблицы.

Примеры операторов: Create, Alter, Drop

2. Язык манипулирования данными (Data Manipulation Lanquaqe – DML).

Он дает возможность манипулировать данными внутри объектов реляционной БД.

Примеры операторов: Update, Insert, Delete.

3. Язык запросов к данным (Data Query Lanquaqe – DQL).

Он используется для построения запросов к реляционным БД.

Примеры операторов: Select.

4. Язык управления данными (Data Control Language – DCL). Он позволяет осуществлять контроль над возможностью доступа пользователей к данным внутри БД, а также определяет набор прав пользователей при работе с объектами БД.
Примеры операторов: Grant, Revoke.

5. Язык администрирования данных (Data Administration Language – DAL). Он дает возможность выполнять аудит и анализ операций внутри БД.
Примеры операторов: Start, Audit.

Типы данных SQL

Понятие тип данных в БД полностью адекватно понятию типа данных в языках программирования. Тип данных определяет:

- ✓ каков размер области памяти, занимаемой объектом;
- ✓ как интерпретируется эта память;
- ✓ какие действия можно выполнять с этим объектом.

В SQL используются следующие основные типы данных, форматы которых могут несколько различаться для разных СУБД:

а) ***символьные*** типы данных – содержат буквы, цифры и специальные символы:

✓ **CHAR** или **CHAR(n)** – символьные строки фиксированной длины. Длина строки определяется параметром **n**. CHAR без параметра соответствует CHAR(1). Для хранения таких данных всегда отводится **n** байт вне зависимости от реальной длины строки;

✓ **VARCHAR(n)** – символьная строка переменной длины. Для хранения данных этого типа отводится число байт, соответствующее реальной длине строки;

б) ***целые*** типы данных – поддерживают только целые числа (дробные части и десятичные точки не допускаются). Над этими типами разрешается выполнять арифметические операции и применять к ним агрегирующие функции (определение максимального, минимального, среднего и суммарного значения столбца реляционной таблицы):

✓ **INTEGER** или **INT** – целое, для хранения которого отводится, как правило, 4 байта. (Замечание: число байт, отводимое для хранения того или иного числового типа данных зависит от используемой СУБД и аппаратной платформы.) Интервал значений от – 2147483647 до + 2147483648;

✓ **SMALLINT** – короткое целое (2 байта), интервал значений от – 32767 до +32768;

в) ***вещественные*** типы данных – описывают числа с дробной частью:

- ✓ **FLOAT** и **SMALLFLOAT** – числа с плавающей точкой (для хранения отводится обычно 8 и 4 байта соответственно);
- ✓ **DECIMAL(p)** – тип данных аналогичный **FLOAT** с числом значащих цифр **p**;
- ✓ **DECIMAL(p,n)** – аналогично предыдущему, **p** – общее количество десятичных цифр, **n** – количество цифр после десятичной запятой;

г) ***денежные*** типы данных – описывают, естественно, денежные величины. Если система такого типа данных не поддерживает, то используется DECIMAL(p,n):

✓ **MONEY(p,n)** – все аналогично типу DECIMAL(p,n). Вводится только потому, что некоторые СУБД предусматривают для него специальные методы форматирования;

д) ***дата и время*** – используются для хранения даты, времени и их комбинаций. Большинство СУБД умеет определять интервал между двумя датами, а также уменьшать или увеличивать дату на определенное количество времени:

- ✓ **DATE** – тип данных для хранения даты;
- ✓ **TIME** – тип данных для хранения времени;
- ✓ **DATETIME** – тип данных для хранения моментов времени (год + месяц + + день + часы + минуты + секунды + доли секунд);

е) ***двоичные*** типы данных – позволяют хранить данные любого объема в двоичном коде (оцифрованные изображения, исполняемые файлы и т.д.). Определения этих типов наиболее сильно различаются от системы к системе, часто используются ключевые слова:

- ✓**BINARY,**
- ✓**BYTE,**
- ✓**BLOB.**

Глава 10.
Организация данных с помощью языка SQL.
Язык DDL.

Конструкции языка SQL для организации данных в БД

Создание таблиц

Перед созданием таблицы БД необходимо:

1. создать структуру таблицы, т.е. определить необходимый перечень полей и отношений между ними;
2. присвоить имена выбранным полям;
3. определить тип каждого поля (символьное, числовое, логическое и т.д.);
4. задать размер полей.

Таблицы создаются командой

CREATE TABLE

Эта команда определяет **имя** таблицы и саму таблицу в виде описания набора **имен столбцов**, указанных в определенном порядке. Она также определяет **типы данных** и **размеры** столбцов. Каждая таблица должна иметь, по крайней мере, 1 столбец.

***Замечания*:** 1) Команда CREATE TABLE создает *пустую* таблицу. 2) Для данных символьного типа «размер» указывать обязательно, т.к. по умолчанию – это 1 (один символ).

Синтаксис команды:

CREATE TABLE <имя_таблицы>
(<имя_столбца><тип_данных>[(<размер>)],
<имя_столбца><тип_данных>[(<размер>)]…);

Значение аргумента ***«размер»*** зависит от типа данных. Если он не указывается, то СУБД будет устанавливать значение автоматически.

Пример1. Создать таблицу-справочник клиентов:

CREATE TABLE Клиенты
(КодКлиента integer,
НазваниеКлиента char(40),
ДатаРождения date,
Адрес char(30),
Телефон char(13),
PRIMARY KEY (КодКлиента))

Удаление таблиц

Удалить можно только **пустую** таблицу. Заполненная таблица с находящимися в ней строками не может быть удалена, т.е. таблица перед удалением должна быть очищена.

Синтаксис команды:

DROP TABLE <имя_таблицы>

Корректировка таблицы

Для изменения таблицы, после того как она была создана, используется команда:

ALTER TABLE

Команда может добавлять столбцы к таблице, удалять столбцы, изменять их размеры, а также добавлять или удалять ограничения.

Эта команда не часть стандарта ANSI, поэтому в разных системах она имеет разные возможности.

Синтаксис команды для добавления столбца в таблицу:

ALTER TABLE <имя_таблицы> **ADD** <имя_столбца> <тип_данных>,<размер>;

Синтаксис команды для удаления столбца из таблицы:

ALTER TABLE <имя_таблицы> **DROP COLUMN** <имя_столбца>

Введение ограничений

При создании или изменении таблицы можно устанавливать ограничения на значения, вводимые в поля. В этом случае SQL будет отклонять любые данные, которые нарушают заданные ограничения.

Ограничение может устанавливаться как отдельно для каждого конкретного столбца, так и на всю таблицу в целом.

Для определения ограничения на столбец, специальное выражение вставляется в конец имени столбца после типа данных и перед запятой.

Синтаксис для команды CREATE TABLE с учетом ограничений:

CREATE TABLE <имя_таблицы>
(<имя_столбца><тип_данных><ограничение_таблицы>,
<имя_столбца><тип_данных><ограничение_таблицы >…);

С помощью ограничений (CONSTRAINT) можно устанавливать индексы для полей таблицы, первичный и внешний ключи, определять отношение и целостность данных.

1) Чтобы предохранить поле таблицы от ввода в него пустых (NULL) значений, нужно при создании таблицы указать для соответствующего столбца ключевое выражение **NOT NULL**.

2) Ключевое слово **UNIQUE** определяет ограничение: значение столбца должно быть уникальным.

3) Если указано ограничение **CHECK (<условие>)**, то будет проверяться проверка условия на вводимое в поле значение.

4) Ключевое слово **DEFAULT** задает значение по умолчанию.

Индексирование

Для манипулирования со значением строк таблицы предназначены **ИНДЕКСЫ**. *Индексирование* – это упорядочение записей по ключу (алфавиту, хронологии, в порядке возрастания или убывания). Для индексного поля создается упорядоченный список значений для этого поля. В таблице данных строки не упорядочены. Для поиска строки с заданным значением поля-ключа программа последовательно просматривает все записи таблицы, строка за строкой, пока не встретит строку с заданным значением поля, а это долгий путь. Индекс же сразу находит запись по значению поля-ключа.

Индекс (индексный файл) создается по команде:

CREATE INDEX <имя_индекса> **ON** <имя_таблицы> (<имя_столбца> [,<имя_столбца>]…);

Пример2. Создать индекс по полю «ФИО» таблицы «Список»:

CREATE INDEX ФИО ON Список (ФИО);

Замечание: Таблица индексов (индексный файл), созданная командой CREATE INDEX, для пользователя ***невидима***. SQL сам автоматически обращается к таблице индексов по мере надобности.

Создание первичных ключей

Первичный ключ создается с помощью выражения **PRIMARY KEY**.

Таблица может содержать только один первичный ключ.

Существует два способа создания первичного ключа:

1 способ – в виде ограничения на столбец (для простого первичного ключа);

2 способ – в виде ограничения на таблицу (для простого или составного первичного ключа).

Примеры создания первичных ключей:

1-й способ:

```
CREATE TABLE Продавцы
(КодПродавца   integer
PRIMARY KEY,
ИмяПродавца   char (10),
Город         char (10),
Комиссионные  float);
```

2-й способ:

```
CREATE TABLE Заказы
(КодПокупки          integer,
СуммаПокупки         double,
ДатаПокупки          date,
КодКлиента           integer,
КодПродавца          integer,
PRIMARY KEY (КодКлиента,
КодПродавца));
```

Создание внешних ключей

(создание схемы данных)

Внешний ключ создается с помощью выражения **FOREIGN KEY**. Назначение FOREIGN KEY – это ограничение допустимых значений поля множеством значений первичного ключа, ссылка на который указывается при описании данного ограничения. Как и для первичного, для внешнего ключа также существует два способа его назначения:

1-й способ – в виде ограничения на столбец:

```
CREATE TABLE Заказы
(КодПокупки      integer,
СуммаПокупки     double,
ДатаПокупки      date,
КодКлиента       integer REFERENCES Клиенты (КодКлиента),
КодПродавца      integer REFERENCES Продавцы (КодПродавца));
```

2-й способ – в виде ограничения на таблицу:

```
    CREATE TABLE Заказы
    (КодПокупкиinteger,
    СуммаПокупки     double,
    ДатаПокупки      date,
    КодКлиента       integer,
    КодПродавца      integer,
    FOREIGN KEY (КодКлиента) REFERENCES Клиенты (КодКлиента),
    FOREIGN KEY (КодПродавца) REFERENCES Продавцы
(КодПродавца));
```

Примеры, приведенные на предыдущем слайде, определяют поля *КодКлиента* и *КодПродавца* как **внешние** ключи таблицы ***Заказы***, ссылающиеся на первичные ключи *КодКлиента* и *КодПродавца* таблиц ***Клиенты*** и ***Продавцы*** соответственно в базе данных, состоящей из трех таблиц:

Замечание: Если в родительской таблице у первичного ключа уже указано ограничение PRIMARY KEY, то при задании ограничения FOREIGN KEY, накладываемого на таблицу или на столбцы, можно в скобках не указывать список столбцов первичного ключа.

Манипулирование данными

Ввод значений в таблицы

Все строки в SQL вводятся с использованием команды модификации **INSERT**.

Предложение INSERT имеет следующий формат:

INSERT INTO <имя_таблицы> [(столбец [,столбец] ...)]

VALUES (<значение> [, (<значение>] ...);

Пример3: Так, например, чтобы ввести строку в таблицу Продавцов, можно использовать следующее условие:

INSERT INTO Продавцы
VALUES (11, 'Браун', 'Лондон', 0.12);

Пример4: Можно также указывать имена столбцов, куда конкретно необходимо вставить значения. Это позволяет вставлять имена в любом порядке. Например:

INSERT INTO Продавцы (ИмяПродавца, Комиссионные, КодПродавца, Город)
VALUES ('Браун',0.12, 11, 'Лондон');

Удаление строк из таблицы

Можно удалять строки из таблицы командой модификации – **DELETE**. Она может удалять только введённые **строки**, а не индивидуальные значения полей. Предложение DELETE имеет следующий формат:

DELETE FROM <имя_таблицы>
[**WHERE** логическое_условие];

Пример5: Чтобы удалить все содержание таблицы Продавцы:

DELETE FROM Продавцы;

Пример6: Чтобы определить какие строки будут удалены, можно использовать условие. Например, чтобы удалить продавца *с кодом 13* из таблицы:

```
DELETE FROM Продавцы
WHERE КодПродавца = 13;
```

Изменение значений полей таблицы

Можно изменить в строке таблицы некоторые или все значения командой модификации – **UPDATE**. Эта команда содержит предложение UPDATE, в котором указано имя используемой таблицы и предложение SET, которое указывает на изменение, которое нужно сделать для определенного столбца.

Предложение UPDATE имеет формат:

UPDATE <имя_таблицы>
SET столбец_1 = значение_1 [, столбец_2 = значение_2 ...]
[**WHERE** логическое_условие]

Здесь *столбец_1* – название первого из обновляемых столбцов, *значение_1* – присваиваемое ему значение (константа или результат вычислений).

Обновляемых столбцов может быть несколько.

Замечание:

1) Если необязательная секция **WHERE** пропущена, то обновляются все записи таблицы. Если эта секция присутствует, то будут обновляться только те записи, для которых *логическое_условие* будет истинным.

2) Составные условия формируются с помощью логических связок AND, OR, NOT.

Пример7: Так, например, чтобы изменить фамилию и номер группы студента, идущего в БД под идентификатором 120, можно указать в условии значение первичного ключа:

UPDATE Студенты
SET ФИО = ‘Петровский П.П.’, Группа = Ф-30
WHERE КодСтудента = 120

Пример8: Чтобы увеличить все цены товаров в Таблице1 на 10%:

UPDATE Таблица1
*SET Цена = Цена * 1.1*

Глава 11.

Организация запросов в формате SQL.

Язык DML.

Извлечение информации из таблицы

Синтаксис оператора SELECT

Для извлечения информации из таблиц применяется команда **SELECT** – наиболее частая команда при работе с реляционной БД. Этот оператор обладает большими возможностями по созданию структуры ВЫХОДНОЙ информации, указанию источников ВХОДНОЙ информации, способа упорядочения выходной информации, формированию новых значений и т.п.

SELECT [Предикат] {*|таблица.*| [таблица.] поле1[,[таблица.] поле2[, …]]}

[**AS** псевдоним1[, псевдоним2[, …]]]

FROM Выражение[, ...] [**IN** Внешняя_БД]

[**WHERE** Условие_отбора]

[**ORDER BY** поле1 [ASC | DESC] [, поле2 [ASC | DESC]] ...]

[**GROUP BY** Список_полей_группировки]

[**HAVING** Условие_группировки]

Аргументы оператора SELECT

Предикат – задает ограничения на возвращаемые записи:

ALL – все записи (по умолчанию);

DISTINCT – все записи без их дублирования;

DISTINCTROW – полностью различающиеся записи по всем полям;

TOP – возврат заданного числа или процента записей в диапазоне, соответствующем фразе ORDER BY.

Таблица – имя таблицы-источника, из которой берутся записи.

Поле1, Поле2… – имена полей, используемых при отборе (порядок их следования определяет выходную структуру выборки данных).

Псевдоним1, Псевдоним2… – новые заголовки столбцов результата выборки данных.

FROM – определяет выражение, используемое для задания источника формирования выборки (обязательно присутствует в каждом операторе).

Внешняя_БД – имя БД с таблицами, которые указаны с помощью аргумента *Выражение*, если они не находятся в текущей БД.

[WHERE…] – определяет условие отбора записей в выражении *Условие_отбора* (необязательный аргумент).

[ORDER BY…] – определяет поля, по которым выполняется упорядочение выходных записей. Упорядочение возможно как по возрастанию (ASC), так и по убыванию (DESC) значения выбранного поля (необязательный аргумент).

[GROUP BY…] – определяет поля для формирования групп, по которым возможно вычисление групповых итогов; порядок их следования определяет виды итогов (старший, промежуточный и т.п.). Максимум полей для группировки – 10. (необязательный аргумент).

[HAVING…] – определяет условия отбора записей для сгруппированных данных (необязательный аргумент).

В самой простой форме (простейшие запросы), в команде SELECT указываются столбцы, которые должны быть выведены в результате запроса, и имя таблицы из которой они будут извлекаться.

Пример1: Выводим значения полей *Номер машины, Тип автомобиля* и *Год выпуска* из таблицы *Парк машин*:

```
SELECT [Номер машины], [Тип автомобиля], [Год выпуска]
FROM [Парк машин];
```

Задание условий выборки

Предложение **WHERE** может содержать выражения, связанные логическими операторами, с помощью которых задаются условия выборки:

Оператор	Назначение
And	логическое И или конъюнкция (логическое умножение)
Or	логическое ИЛИ дизъюнкция (включающее Or)
Not	отрицание
Eqv	проверка логической эквивалентности выражений
Imp	логическая импликация выражений
Xor	логическое ИЛИ (исключающее Or)

Кроме того, могут использоваться операторы для построения условий:

LIKE – выполняет сравнение строковых значений;

BETWEEN…AND – выполняет проверку на диапазон значений;

IN – выполняет проверку выражения на совпадение с любым из элементов списка;

IS – проверка значения на **Null** (пусто).

Отбор записей по части значения поля

Ключевое слово LIKE используется для определения шаблона, в соответствии с которым выполняется отбор записей. Слева от ключевого слова LIKE пишется имя поля, по значениям которого выполняется отбор записей, а справа – шаблон. Для составления шаблона допускается использование следующих символов:

% – заменяет любое количество символов слева или справа от комбинации букв;

_ – заменяет один символ. Допускается многократное использование символа «_».

Пример2: Выбираем перевозки, расстояние в которых более 100 км:

```
SELECT [Номер перевозки], [Дата перевозки], Расстояние
FROM Перевозки
WHERE Перевозки.Расстояние>100;
```

Пример3: Выбираем перевозки, выполненные в январе 2019 г.:

```
SELECT Перевозки.*
FROM Перевозки
WHERE Перевозки.[Дата перевозки] Between #1/1/2019# And #1/31/2019#;
```

Пример4 (И-запрос): Выбираем перевозки, *расстояние* в которых более *100* км:

```
SELECT Перевозки.*
FROM Перевозки
WHERE Перевозки.[Вес груза (кг)]>=100 AND Перевозки.[Пункт назначения] In ("Минск","Пинск");
```

Пример5 (ИЛИ-запрос): Выбираем машины с типом *Маз* или *Камаз*:

```
SELECT [Типы машин].*
FROM [Типы машин]
WHERE [Тип автомобиля]="Маз" Or [Тип автомобиля]="Камаз";
```

Замечание1: Если идентификатор (*например,* поле таблицы) содержит пробелы или специальные знаки, он должен быть заключен в **прямоугольные** скобки.

Групповые функции SQL

Групповые функции необходимы для определения статистических данных на основе наборов числовых значений:

Функция	Назначение
Avg	вычисляет среднее арифметическое набора чисел, содержащихся в указанном поле запроса;
Count	вычисляет количество выделенных записей в запросе;
Min, Max	возвращают минимальное и максимальное значения из набора в указанном поле запроса;
StDev, StDevPs	возвращают среднеквадратическое отклонение генераль-ной совокупности и выборки для указанного поля в запросе;
Sum	возвращает сумму значений в заданном поле запроса;
Var, VarPs	возвращают дисперсию распределения генеральной совокупности и выборки для указанного поля в запросе.

Пример6: ***Выводим список машин с перевезенным общим весом***:

```
SELECT [Номер машины], Sum([Вес груза (кг)]) AS ОбщийВес
FROM Перевозки
GROUP BY [Номер машины]
```

Результат выборки:

	Номер машины	ОбщийВес
	0111 КК-5	32322
	2233 ОО-2	6400
	3355 ДД-5	5600
	3452 ТТ-1	20000
	6542 ЛЛ-1	16000
	6654 ДЛ-1	2000
	7821 АА-1	13500
	7896 ГГ-2	46500

Пример7: **Выводим количество перевозок и среднее расстояние для водителя, код которого задаем с клавиатуры**:

```
SELECT [Код водителя], Count([Код водителя]) AS КоличествоПеревозок, Avg([Расстояние (км)]) AS СреднееРасстояние
FROM Перевозки
GROUP BY [Код водителя]
HAVING [Код водителя]=[Введи код водителя] ;
```

Результат выборки:

Код водителя	КоличествоПеревозок	СреднееРасстояние
2	2	1191

Подзапросы

В инструкцию SELECT может быть вложена другая инструкция SELECT, SELECT...INTO, INSERT...INTO, DELETE или UPDATE. Различают ***основной*** и ***подчиненные*** запросы, которые являются вложенными в основной запрос.

Подчиненный запрос можно использовать вместо выражения в списке полей инструкции SELECT или в предложениях WHERE и HAVING.

Существуют три типа подчиненных запросов.

Первый тип – сравнение выражения с результатом подчиненного запроса:

сравнение (ANY | ALL | SOME) (инструкция)

Ключевые слова:

ANY – каждый (сравнение с каждым элементом подчиненной выборки).

ALL – все (сравнение со всеми элементами подчиненной выборки).

SOME – некоторые (сравнение с некоторыми элементами подчиненной выборки).

Пример8: ***Выводим только те записи из таблицы «Перевозки», в которых значение веса груза больше каждого значения веса для водителя с кодом 7***:

*SELECT * FROM Перевозки WHERE [Вес груза (кг)] > ANY*
(SELECT [Вес груза (кг)] FROM Перевозки WHERE Перевозки.[Код водителя]=7)

Второй тип – выражение, которое должно быть найдено в наборе записей, являющихся результатом выполнения подчиненного запроса:

выражение [NOT] IN (инструкция)

Пример9: ***Выводим из таблицы «Парк машин», информацию о машинах, которые осуществляли перевозки в Минск***:

*SELECT * FROM [Парк машин] WHERE [Номер машины] IN*

(SELECT [Номер машины] FROM Перевозки WHERE [Пункт назначения] = "Минск")

Пример10: ***Выводим информацию о машинах, которые еще не участвовали в перевозках***:

*SELECT * FROM [Парк машин] WHERE [Номер машины] NOT IN*

(SELECT [Номер машины] FROM Перевозки)

Третий тип – инструкция SELECT, заключенная в круглые скобки, с предикатом EXISTS (с необязательным зарезервированным словом NOT) в логическом выражении для определения, должен ли подчиненный запрос возвращать какие-либо записи.

[NOT] EXISTS (инструкция)

Пример11: ***Запрос из Примера10***:

SELECT FROM [Парк машин] WHERE NOT EXISTS*

*(SELECT * FROM Перевозки WHERE [Парк машин].[Номер машины]=Перевозки.[Номер машины])*

Операции соединения таблиц

Оператор языка SQL **JOIN** предназначен для соединения двух или более таблиц БД по совпадающему условию. Этот оператор существует только в реляционных БД. Именно благодаря JOIN реляционные БД обладают такой мощной функциональностью, которая позволяет вести не только хранение данных, но и их, хотя бы простейший, анализ с помощью запросов.

Для соединения двух таблиц оператор SQL JOIN имеет следующий синтаксис:

SELECT *Имена_Столбцов (1..N)*
FROM *Таблица1* **JOIN** *Таблица2* **ON** *Условие*

После одного или нескольких звеньев с оператором JOIN может следовать необязательная секция WHERE или HAVING, в которой, также, как в простом SELECT-запросе, задаётся условие выборки.

Общим для всех СУБД является то, что в этой конструкции вместо JOIN может быть указано:

INNER JOIN – *внутреннее соединение*,

LEFT OUTER JOIN,

RIGHT OUTER JOIN,

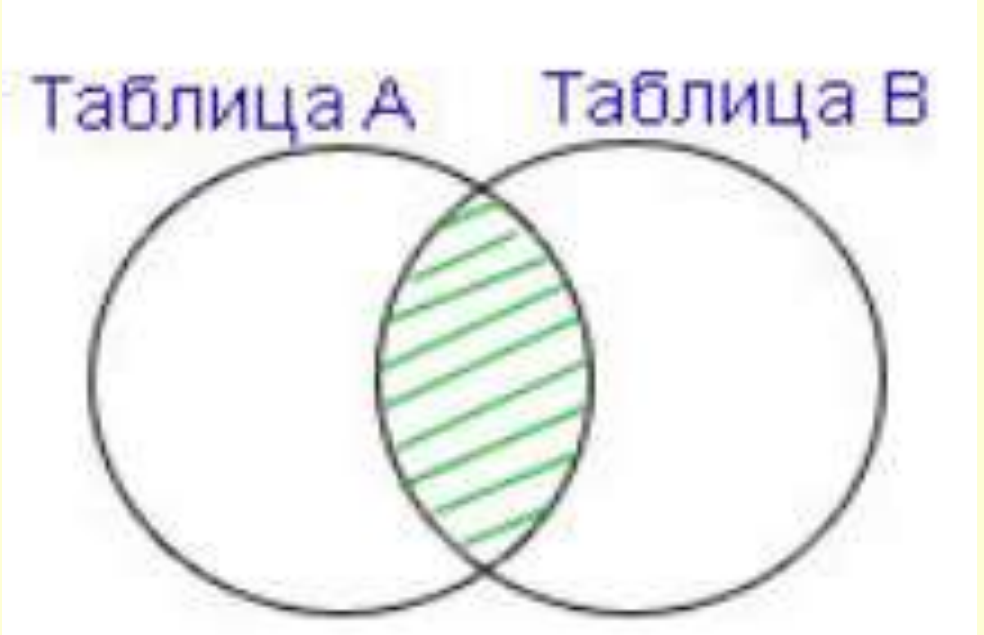
FULL OUTER JOIN,

} ***внешние соединения***

Замечание: Запрос можно писать без ключевого слова OUTER.

INNER JOIN (внутреннее соединение)

Запрос с оператором INNER JOIN предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON. То же самое делает и просто JOIN. Обычно используется для объединения записей, которые есть и в первой и во второй таблице, т. е. получения пересечения таблиц.

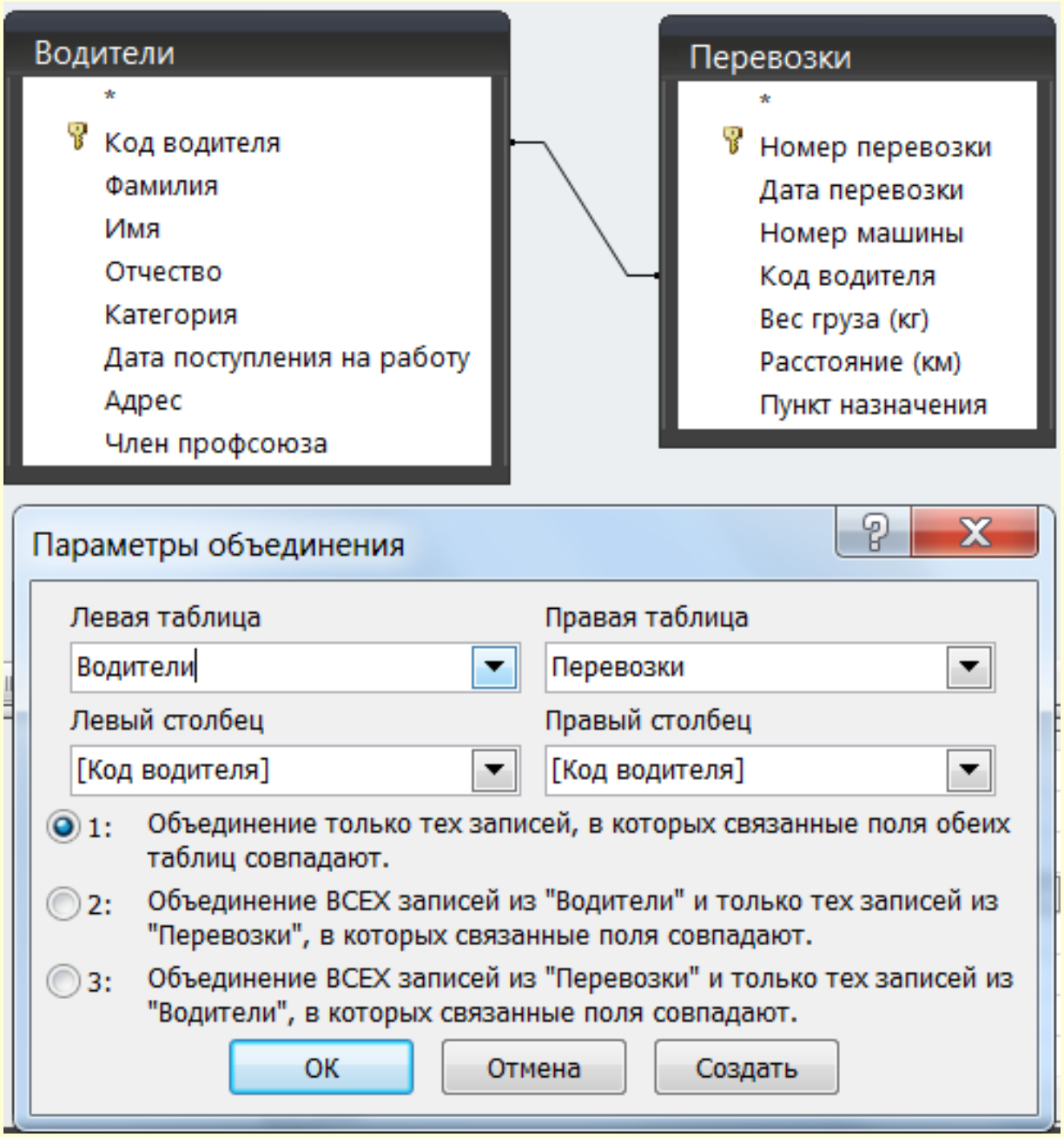


Пример12: ***Выводим записи (симметричное соединение двух таблиц), в которых значение их ключей связи (поле «Код водителя») представлено в обеих таблицах (т.е., не выводятся записи про водителя, который не выполнил ни одной перевозки, но есть в справочнике «Водители»)***:

SELECT Перевозки., Водители.* FROM Водители INNER JOIN Перевозки*

ON Водители.[Код водителя] = Перевозки.[Код водителя];

Замечание: В режиме *Конструктора запросов* (Пример12) линия объединения между таблицами в **Схеме данных** будет отображаться обычной линией связи *(без замыкающих стрелок на концах)* и параметры этого объединения *(окно вызывается соответствующей командой из контекстного меню)* будут выглядеть следующим образом:


Водители
*
Код водителя
Фамилия
Имя
Отчество
Категория
Дата поступления на работу
Адрес
Член профсоюза
Перевозки
*
Номер перевозки
Дата перевозки
Номер машины
Код водителя
Вес груза (кг)
Расстояние (км)
Пункт назначения
Параметры объединения
Левая таблица
Водители
Правая таблица
Перевозки
Левый столбец
[Код водителя]
Правый столбец
[Код водителя]
1: Объединение только тех записей, в которых связанные поля обеих таблиц совпадают.
2: Объединение ВСЕХ записей из "Водители" и только тех записей из "Перевозки", в которых связанные поля совпадают.
3: Объединение ВСЕХ записей из "Перевозки" и только тех записей из "Водители", в которых связанные поля совпадают.
OK
Отмена
Создать

LEFT OUTER JOIN

(левое внешнее соединение)

Запрос с оператором **LEFT OUTER JOIN** предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после **ON**, и дополняются записями из *первой* по порядку (**левой**) таблицы, даже если они не соответствуют условию.

У записей **левой** таблицы, которые не соответствуют условию, значение столбца из правой таблицы будет неопределённым (NULL).

Замечание: Левая таблица та, которая идет перед написанием ключевых слов [LEFT | RIGHT | INNER] JOIN, правая таблица – после них.

Пример13: Выводим все записи таблицы «Топливо» и соответствующие им записи таблицы «Парк машин»:

*SELECT [Парк машин].[Номер машины], [Парк машин].[Тип автомобиля], Топливо.**

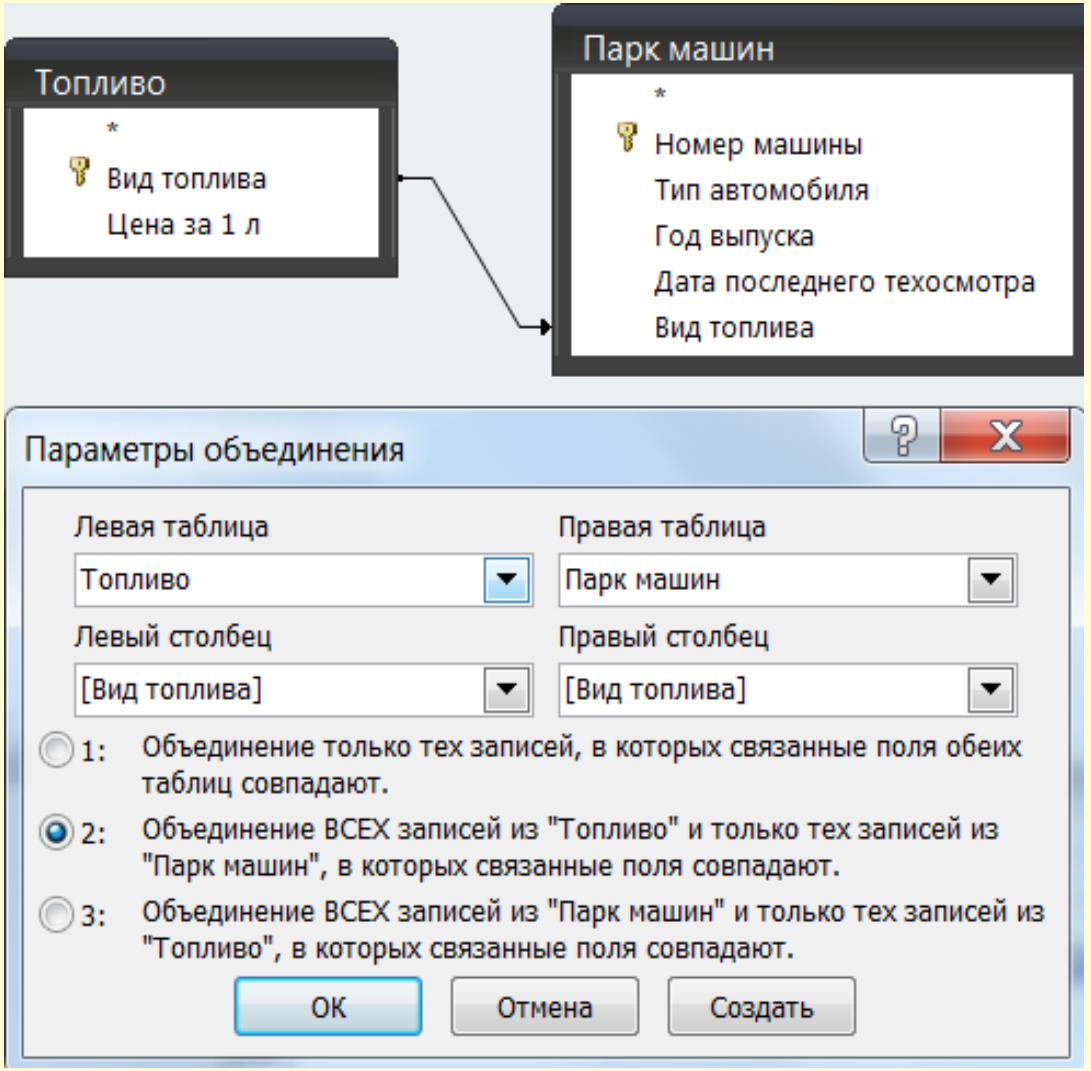
FROM Топливо LEFT JOIN [Парк машин] ON Топливо.[Вид топлива] = [Парк машин].[Вид топлива];

Замечание1: В результирующей таблице значения столбцов таблицы «Парк машин» имеют значения NULL для записи с видом топлива Бензин АИ-95, т.к. эта запись имеет идентификатор «Вид топлива», значения которого нет в таблице «Парк машин» (т.е. нет машин в парке, заправляемых данным видом топлива).

Результат запроса:

	Номер машины	Тип автомоби	Вид топлива	Цена за 1 л
	0111 КК-5	Газ	Бензин А92	1,68
	6542 ЛЛ-1	Газ	Бензин А92	1,68
	5566 ЛЛ-1	Газ	Бензин А92	1,68
			Бензин А95	1,78
	3355 ДД-5	Маз	Биотопливо	2,00
	6654 ДЛ-1	Зил	Биотопливо	2,00
	3452 ТТ-1	Зил	Газ	0,94
	2233 ОО-2	Зил	ЭктоДизель	1,78
	7821 АА-1	Камаз	ЭктоДизель	1,78
	7896 ГГ-2	Камаз	ЭктоДизель	1,78
*				

Замечание2: В режиме *Конструктора запросов* (Пример13) линия объединения между таблицами в **Схеме данных** будет отображаться односторонней стрелкой и параметры этого объединения будут выглядеть следующим образом:



RIGHT OUTER JOIN

(правое внешнее соединение)

Запрос с оператором **RIGHT OUTER JOIN** предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после **ON**, и дополняются записями из второй по порядку (**правой**) таблицы, даже если они не соответствуют условию.

У записей **правой** таблицы, которые не соответствуют условию, значение столбца из правой таблицы будет неопределённым.

***Пример14:* Выводим все записи таблицы «Перевозки» и соответствующие им записи таблицы «Парк машин»**:

```
SELECT КопияПеревозки.[Номер перевозки],
КопияПеревозки.[Дата перевозки],
КопияПаркМашин.[Номер машины],
КопияПаркМашин.[Тип автомобиля]
FROM КопияПаркМашин RIGHT JOIN
КопияПеревозки ON КопияПаркМашин.[Номер
машины]=КопияПеревозки.[Номер машины];
```

Замечание1: В результирующей таблице значения столбцов таблицы «Парк машин» имеют значения NULL для записи с номером машины 0222 АВ-3, т.к. эта запись имеет идентификатор, значения которого нет в таблице «Парк машин» (т.е. нет такой машины в парке, на которой осуществлялась перевозка).

Замечание2: Запрос был выполнен над копиями исходных таблиц БД «Автоперевозки», между которыми установлены связи БЕЗ обеспечения целостности данных, поскольку в противном случае, было бы невозможным добавление в подчиненную таблицу записи со значением внешнего ключа, которого нет в ключевом поле главной таблицы!!!

Результат запроса:

Номер перевози	Дата перевози	Номер машины	Тип автомоби
8	06.11.2019	0111 КК-5	Газ
17	02.03.2020	0111 КК-5	Газ
18	02.03.2020		
11	10.11.2019	2233 ОО-2	Зил
13	13.11.2019	3355 ДД-5	Маз
2	02.02.2019	3452 ТТ-1	Зил
12	12.11.2019	3452 ТТ-1	Зил
5	03.11.2019	6654 ДЛ-1	Зил
3	02.02.2020	7821 АА-1	Камаз
15	15.11.2019	7821 АА-1	Камаз
4	02.11.2019	7896 ГГ-2	Камаз
9	06.11.2019	7896 ГГ-2	Камаз
14	14.11.2019	7896 ГГ-2	Камаз
*	(№)		

FULL OUTER JOIN

(полное внешнее соединение)

Запрос с оператором **FULL OUTER JOIN** предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию, указанному после ON, и дополняются записями

из первой (**левой**)

и второй (**правой**) таблиц,

даже если они не

соответствуют условию.

У записей, которые не соответствуют условию, значение столбцов из другой таблицы будет NULL (неопределённым).

Соединение более двух таблиц

Реляционные БД должны подчиняться требованиям целостности и неизбыточности данных, в связи с чем данные об одном бизнес-процессе могут содержаться не только в одной, двух, но и в трёх и более таблицах. В этих случаях для анализа данных используются цепочки соединённых таблиц. При помощи оператора **JOIN** в одном запросе можно соединить большое число таблиц. В таких запросах за одной секцией соединения следует другая, причём каждый следующий JOIN соединяет со следующей таблицей таблицу, которая была второй в предыдущем звене цепочки.

Таким образом, синтаксис SQL запроса для соединения более двух таблиц следующий:

```
SELECT ИМЕНА_СТОЛБЦОВ (1..N)
    FROM ИМЯ_ТАБЛИЦЫ_1 JOIN ИМЯ_ТАБЛИЦЫ_2
        ON УСЛОВИЕ
        JOIN ИМЯ_ТАБЛИЦЫ_3
            ON УСЛОВИЕ
        ...
    JOIN ИМЯ_ТАБЛИЦЫ_M
            ON УСЛОВИЕ
```

Пример15: ***Выводим номера перевозок машинами, грузоподъемность которых более 4 тонн***:

SELECT Перевозки.[Номер перевозки], Перевозки.[Номер машины], [Парк машин].[Год выпуска], [Типы машин].[Грузоподъемность (тонн)]

FROM [Типы машин] INNER JOIN ([Парк машин] INNER JOIN Перевозки ON [Парк машин].[Номер машины]=Перевозки.[Номер машины]) ON [Типы машин].[Тип автомобиля]=[Парк машин].[Тип автомобиля]

WHERE [Типы машин].[Грузоподъемность (тонн)])>4;

Глава 12.

Процедурный SQL

Язык SQL является очень мощным языком манипулирования данными, однако для решения сложных задач обработки данных ему не хватает управляющих конструкций, имеющихся в универсальных языках программирования.

В связи с этим многие СУБД имеют процедурные расширения этого языка, которые представляют собой полноценный язык программирования, поддерживающий возможность использования в нем операторов SQL.

На таком языке можно писать процедуры и функции, постоянно хранящиеся в БД и исполняемые в среде СУБД.

К сожалению, на настоящий момент ситуация такова, что каждая СУБД поддерживает свой собственный язык процедурного расширения SQL, что усложняет задачи переносимости программного обеспечения (например: процедурный язык PL/SQL поддерживается СУБД Oracle, PLG/SQL – в СУБД PostgreSQL, Transact-SQL – в Microsoft SQL Server).

Ярчайшими представителями процедурной идеи выступают:

- **хранимые процедуры (stored procedures);**
- **функции;**
- **триггеры (triggers).**

Это подпрограммы, заранее подготовленные разработчиком БД, которые компилируются и хранятся как самостоятельный исполняемый код в системном каталоге БД. Процедуры и функции могут вызываться с любого места инструкции SQL (с помощью оператора COLL/EXEC или программы, написанной на высокоуровневом языке), а триггеры запускаются СУБД автоматически по определенному событию без вмешательства извне.

Элементы процедурного SQL

Основной программной единицей является блок – совокупность операторов, заключенная в операторные скобки **BEGIN ... END.**

Задача составного оператора **BEGIN ... END** – указание начала и конца программного блока для логического объединения двух или более операторов, которые должны выполняться как единое целое.

При выполнении процедурных действий в блоке, как правило, необходимы переменные для хранения промежуточных значений.

Переменные

Наиболее распространенный способ хранения данных, используемых в промежуточных вычислениях, реализуется при посредничестве переменных.

Переменные позволяют размещать в памяти значения различных типов (целые и вещественные числа, символы, текстовые строки).

Можно выделить три категории переменных:

1. **пользовательские** переменные – предназначены для решения локальных задач интерактивного SQL, они доступны исключительно в рамках отдельной сессии.

Для создания пользовательской переменной используется формат, в котором она (переменная) должна начинаться с символа @ и состоять из буквенно-цифровых символов. Есть два способа присвоить значение пользовательской переменной.

Первый способ – использование ключевого слова **SET** и операторов присваивания «:=» и «=»:

SET @variable_name := value;

Примеры:

```
SET @counter := 100;
SET @a: = ‘Hello, SQL!’;
SET @b = @c * 1.5
```

Второй способ – использование ключевого слова **SELECT** и оператор присваивания «:=»:

SELECT @variable_name := value;

Пример использования пользовательской переменной для хранения самого последнего идентификатора, сгенерированного столбцом AUTO_INCREMENT:

SELECT @id:=LAST_INSERT_ID();

Замечание: Пользовательская переменная, определенная одним клиентом, не видна другим клиентам.

2. **системные** переменные – это переменные, основное назначение которых заключается в информировании оператора об особенностях конфигурации и работы БД и для изменения параметров работы сервера. При это различают:

- ***глобальные*** системные переменные – оказывают влияние на все операции сервера;
- ***сеансовые*** системные переменные – оказывают влияние лишь на операции текущего клиента.

Все глобальные переменные имеют префикс в виде **двух** символов @, а их значения извлекаются с помощью оператора **SELECT**.

Пример извлечения количества подключений к SQL Server со времени запуска программы:

SELECT @@CONNECTIONS

Среди наиболее часто применяемых системных переменных можно отметить следующие:

@@**ERROR** – содержит номер ошибки, возникшей при выполнении последнего оператора T-SQL в текущем соединении. Если ошибка не обнаружена, содержит 0. Значение этой системной переменной переустанавливается после выполнения каждого очередного оператора.

@@IDENTITY – содержит последнее идентификационное значение, вставленное в базу данных в результате выполнения последнего оператора INSERT. Если в последнем операторе INSERT не произошла выработка идентификационного значения, системная переменная @@IDENTITY содержит NULL.

@@ROWCOUNT – одна из наиболее широко используемых системных переменных, которая возвращает информацию о количестве строк, затронутых последним оператором. Обычно применяется для контроля ошибок, отличных от тех, которые относятся к категории ошибок этапа прогона программы.

3. **Локальные** переменные – используемые в коде хранимых процедур (функций, триггеров).

Объявление локальных переменных образует секцию объявления, которая начинается ключевым словом **DECLARE:**

DECLARE @Имя_Переменной Тип_Данных [,

@Имя_Переменной Тип_Данных, …]

Пример объявления локальной переменной UStr, которая хранит до 16 символов Unicode:

DECLARE @UStr varchar(16)

Замечания: 1. Используемые для переменных типы данных в точности совпадают с существующими в таблицах. 2) В одной команде DECLARE через запятую может быть перечислено несколько переменных.

Пример описания и результата выполнения скрипта (в SQL Server Management Studio):

```
1  DECLARE @name NVARCHAR(20), @age INT;
2  SET @name='Tom';
3  SET @age = 18;
4  PRINT 'Name: ' + @name;
5  PRINT 'Age: ' + CONVERT(CHAR, @age);
```

Messages

```
Name: Tom
Age: 18
```

Хранимые процедуры

Хранимые процедуры в SQL — это аналог функций в других языках программирования. Хранимые процедуры могут выполнять действия над данными автоматически: вывод данных, удаление, изменение.

Особенностью процедур является то, что есть возможность передавать аргументы, (так же как и функциям в других языка) и выводить различные данные в зависимости от аргумента. Также, процедура является сущностью SQL, которую создают один раз, а затем вызывают, передавая аргументы.

Создание процедур в SQL

Чтобы создать процедуру необходимо воспользоваться оператором

CREATE PROCEDURE.

После оператора следует указать имя процедуры, а затем в круглых скобках аргументы, если они имеются, вместе с указанием типа данных каждого аргумента:

CREATE PROCEDURE name_procedure (arg1 datatype, arg2 datatype,...).

Пример процедуры, показывающий список покупателей из города, который передается в качестве аргумента:

```
CREATE PROCEDURE get_customers (city_arg VARCHAR(45))
BEGIN
    SELECT cname, city
    FROM customers
    WHERE city = city_arg;
END ;
```

Процедура обязательно должна начинаться с BEGIN и заканчиваться END.

Внутри процедуры используется запрос SELECT, который сравнивает город со значением аргумента.

Вызов процедур в SQL

Очевидно, для использования процедуры, ее следует вызвать и передать входные аргументы, если они требуются.

Вызов хранимой процедуры в SQL производится с помощью оператора **CALL**.

Вызов созданной выше процедуры (для вывода списка покупателей, проживающих в Москве) будет произведен следующим образом:

```
CALL get_customers("Москва");
```

Триггеры

Триггеры – это хранимые процедуры специального типа, автоматически выполняющиеся при наступлении заданного события.

Триггеры не содержат явных входных и выходных параметров и не могут быть запущены явным образом.

Триггеры по своей сути представляют обработчики событий. Они выполняются при наступлении какого-либо простого действия в SQL. Такими действиями обычно являются: удаление, вставка и обновление данных.

Оператор для создания триггера следующий:

CREATE TRIGGER name_trigger

После оператора и имени триггера необходимо указать в каком случае будет срабатывать триггер. Возможно 6 вариантов:

BEFORE INSERT	AFTER INSERT
BEFORE UPDATE	AFTER UPDATE
BEFORE DELETE	AFTER DELETE

То есть триггер срабатывает либо до, либо после вставки, обновления, удаления данных из БД в SQL.

Пример триггера, который в таблице orders автоматически будет увеличивать новую цену на 20%:

```
CREATE TRIGGER Before_Update_amt
BEFORE UPDATE ON orders
FOR EACH ROW
BEGIN
    SET NEW.amt = NEW.amt * 1.2;
END ;
```

Курсоры

Курсор – это особый временный объект SQL, 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 выполнять сложные вычисления, которые трудно выразить с помощью синтаксиса инструкции SELECT.

Пакеты T-SQL

Запросом называют одну инструкцию T-SQL, а **пакетом** – их набор. Вся последовательность инструкций пакета отправляется серверу из клиентских приложений как одна цельная единица. SQL Server рассматривает весь пакет как рабочую единицу. Наличие ошибки хотя бы в одной инструкции приведет к невозможности выполнения всего пакета.

Файл сценария SQL может содержать несколько пакетов, тогда все пакеты разделяют ключевые слова терминаторов. По умолчанию этим ключевым словом является **GO**, и оно должно быть единственным в строке.

Глава 13.
Администрирование баз данных.

Распределенные транзакции и репликация данных.

Безопасность данных.

§ 13.1. Повышение производительности БД.

Оптимизация работы базы данных является весьма непростой задачей и включает в себя решение целого комплекса взаимосвязанных проблем.

Это обеспечение приемлемого быстродействия и функциональности базы данных, удобства работы пользователей, оптимизация потребляемых ресурсов, например, по критерию минимизации затрат памяти и максимизации использования сети и др.

Важнейшим аспектом оптимизации является **повышение производительности** БД.

Для повышения производительности БД можно использовать общие методы повышения быстродействия программ, такие как ***увеличение мощности аппаратных средств, конфигурирование операционной системы, оптимизация структуры внешних носителей и размещения БД на них*** и др.

Кроме того, используются специальные средства оптимизации работы базы данных, встроенные в СУБД. В частности, большинство современных реляционных СУБД, имеют в своем составе специальный компонент – **оптимизатор запросов**, позволяющий максимально быстро и эффективно обрабатывать запросы выбора и запросы манипулирования данными.

Распространенный способ оптимизации работы БД – это **сжатие базы данных**. Оно обеспечивает оптимизацию размещения объектов базы данных на внешних носителях и возвращение освободившегося дискового пространства для дальнейшего использования

Технологии сжатия данных:

1. Наиболее распространена технология сжатия на основе различий, когда некоторое значение заменяется сведениями об его отличиях от предыдущего значения.

2. Другой тип технологии сжатия основан на иерархическом сжатии данных.

Для ускоренного доступа к данным базы по запросам пользователей используются ***индексирование*** и ***хеширование***.

Индекс-средство ускорений операций поиска в таблице БД, а также других операций, требующих поиска: извлечения, корректировки, сортировки. Цель использования **индексирования** - ускорение извлечения данных за счет уменьшения количества дисковых операций ввода-вывода.

Индексирование таблиц в БД является отличным средством повышения производительности SQL-запросов. Физически все данные таблиц хранятся в файлах, которые разбиты на страницы, без соблюдения какой-либо физической структуры.

Индексы упорядочивают данные таблиц, оптимизируя время поиска данных (подобно бинарному поиску по упорядоченному массиву данных).

Другим распространенным способом упорядочения записей и доступа к данным является **хеширование-технология** быстрого прямого доступа к записи БД на основе заданного значения некоторого поля записи, как правило, ключевого.

Резервным копированием называется сохранение копии данных где-то вне основного места их хранения. Главное назначение резервного копирования – ***восстановление*** данных после их потери.

Нередко приходится слышать, что при наличии реплики БД с неё всегда можно восстановить данные, и резервное копирование не нужно. На самом деле резервное копирование позволяет решить как минимум три задачи, которые не могут быть решены при помощи реплики, да и реплику без резервной копии не инициализировать.

Во-первых, резервная копия позволяет восстановить данные после логической ошибки. Например, бухгалтер удалил группу проводок или администратор БД уничтожил табличное пространство. Обе операции абсолютно легитимны с точки зрения БД, и процесс репликации воспроизведёт их в базе-реплике.

Во-вторых, современные СУБД – весьма надёжные программные комплексы, однако изредка всё же происходит повреждение внутренних структур БД, после которого доступ к данным пропадает. Иногда такое нарушение происходит обычно при высокой нагрузке или при установке какого-нибудь обновления. Но как высокая нагрузка, так и регулярные обновления говорят о том, что БД – отнюдь не тестовая, и данные, хранящиеся в ней, ценны.

Наконец, третья задача, решение которой требует наличия резервной копии, – это клонирование базы, например, для целей тестирования.

Резервное копирование БД так или иначе базируется на одном из двух принципов:

- ✓ выборка данных с последующим сохранением в произвольном формате;
- ✓ снимок состояния файлов БД и сохранение журналов.

§ 13.2. Распределенные транзакции и репликация данных.

Транзакция — последовательность операций модификации данных в БД, переводящая ее из одного непротиворечивого состояния в другое непротиворечивое состояние.

Удаленный запрос — запрос к БД, находящихся на ресурсах локальной сети предприятия или сети Интернет.

Возможность реализации удаленной транзакции — обработка одной транзакции, состоящей из множества SQL-запросов, на одном удаленном узле.

Поддержка распределенной транзакции — обработка транзакции, состоящей из нескольких SQL-запросов, выполняемых на нескольких узлах сети (удаленных или локальных), но каждый из которых обрабатывается только на одном узле.

Распределенный запрос — запрос, при обработке которого используются данные из БД, расположенные в разных узлах сети.

Важнейшей целью создания БДявляется организация ***параллельного*** доступа многих пользователей к общим данным, которые используются совместно.

С параллельным выполнением транзакций связаны следующие основные проблемы:

- потеря обновлений;

- несвязность данных. Возникает в том случае, когда две транзакции выполняются параллельно и первая транзакция отменяется после того, как вторая транзакция уже получила доступ к несвязанным данным. В этом случае откат завершается после того, как начнется выполнение второй транзакции.

- неоднозначность поиска. Проблема состоит в том, что транзакция может считывать одни данные перед их изменением, а другие данные после их изменения, что приводит к несовместимости данных.

В сети распределенных БД необходимо управление параллельным выполнением. Управление параллельным выполнением транзакций — это координирование одновременного исполнения транзакций в распределенных БД.

Цель управления параллельным выполнением в многопользовательских БД состоит в обеспечении ***сериализации*** транзакций. Это требует, чтобы результат множества одновременно выполняемых транзакций был эквивалентен результату последовательного выполнения этих же транзакций.

Для распределенных СУБД транзакции могут выполняться на нескольких узлах, где располагаются необходимые данные.

Репликация — одна из техник масштабирования баз данных. Состоит эта техника в том, что данные с одного сервера БД постоянно копируются (реплицируются) на один или несколько других (называемые репликами).

Для приложения появляется возможность использовать не один сервер для обработки всех запросов, а несколько.

Таким образом появляется возможность распределить нагрузку с одного сервера на несколько.

НАЗНАЧЕНИЯ МЕХАНИЗМА РЕПЛИКАЦИЙ:

1. ***Надежность*** - процесс репликации значительно увеличивает надежность системы, предоставляя приложениям альтернативный доступ к данным. При потере соединения с одним из серверов, пользователь может продолжать работу с данными, используя другой.

2. ***Производительность*** - используя репликацию, можно обеспечить высокую скорость доступа к данным, так как нагрузка будет равномерно распределена между множеством баз репликации.

3. ***Возможность работы баз с базой*** - локальная информационная база позволяет пользователю работать с набором данных без наличия постоянного соединения с БД. Позже, когда установится соединение, пользователь сможет синхронизировать (обновить) объекты, если это необходимо. И внесенные им изменения попадут в БД.

4. Уменьшение сетевой нагрузки - репликация может быть использована для распределения данных между различными региональными центрами. Приложения будут обращаться не к центральному серверу, а к региональному, тем самым уменьшая нагрузку на сеть.

5. Работа всех подписчиков с общими ресурсами - репликация позволяет каждому из подписчиков работать с общими ресурсами, например регистр остатков или регистр бухгалтерии, причем для синхронизации потоков всех операций по каждому подписчику применяется контрольное перепроведение документа в эталонной БД, таким образом, на подписчике при проведении документа назначается статус "предварительно проведен", а на эталонной базе "проведен".

6. Гибкая процедура разрешения конфликтов - репликация позволяет отслеживать конфликты (под конфликтом понимаем состояние, когда 2 или более подписчиков содержат транзакции по изменению одного и того же объекта) по основным типам объектов (справочники, документы), правильно их разрешать, легировать и нотифицировать.

§ 13.3. Безопасность данных.

Под угрозой безопасности понимают все способы вторжения, провоцирующие причинение ущерба информации (искажение или разрушение), хищение информационных ресурсов, в том числе с целью сбыта. Возможны как случайные, так и преднамеренные угрозы.

Под **случайными** понимаются технические сбои, не зависящие от пользователя, – ошибки, повреждения компьютеров или носителей, природные бедствия.

Ошибки по невниманию, как правило, связаны с нарушением процесса выполнения программ, и происходят чаще всего по вине сотрудников.

Преднамеренные угрозы зачастую реализуют внешние источники вторжения, то есть не имеющие отношения к организации – источнику информации. Таких «специалистов» может заинтересовать как программное обеспечение, так и обрабатываемые им сведения.

Причем цели присвоения ресурсов могут быть разными: от банальной наживы до саботажа и дискриминации всего предприятия и впоследствии – разлада рабочего процесса.

Понятие «**безопасность баз данных**» включает в себя целый ряд инструментов, средств контроля и мер, направленных на обеспечение и поддержку конфиденциальности, целостности и доступности баз данных.

Безопасность БД должна включать в себя следующие элементы:

- ✓ Данные, хранящиеся в базе данных.
- ✓ Система управления базами данных (СУБД).
- ✓ Любые связанные приложения.
- ✓Физический сервер базы данных и/или виртуальный сервер базы данных и используемое оборудование.
- ✓ Вычислительные ресурсы и/или сетевая инфраструктура, предназначенная для доступа к БД.

Безопасность баз данных — сложная и комплексная инициатива, охватывающая все аспекты технологий и методов в области информационной безопасности.

Чем доступнее и удобнее база данных для конечных пользователей, тем уязвимее она для угроз безопасности; чем защищеннее база данных от угроз, тем сложнее получить доступ к данным.

Что касается сохранности содержимого БД, стоит выделить два основных подхода, которые призваны обеспечить безопасность данных:

1. ***Избирательное управление данными***. Достаточно гибкий способ, при использовании которого разные контрагенты могут быть наделены различными полномочиями по отношению к одному и тому же элементу.

2. ***Обязательное управление данными***. Этот способ имеет обратную схему действия, то есть БД и все объекты библиотек распределены по уровням секретности, которые подразумевают допуск к просмотру и модификации пользователей с соответствующим уровнем доступа или выше. Дабы не нарушать всю структуру секретности, способ не дает возможности субъекту с наивысшей классификацией полномочий вносить данные в категории низшей секретности.

Первостепенные критерии надежности:

1. ***Политика безопасности данных.*** Направлена на мониторинг потенциальных угроз и поиск средств для их устранения. Содержит правила и нормы пользования содержимым хранилища, свойственные конкретной организации. На основании редакции политики безопасности определяются отдельные механизмы организации защиты системы.

2. Гарантированность данных. Предполагает оправданность выбранных программных средств. Эта категория воплощает те механизмы, которые дают ход политике безопасности конкретной системы.

Вычислительная часть системы управления безопасностью контролирует допуск контрагентов к тем или иным операциям с данными, находящимися под защитой. Функции мониторинга при обращении субъекта соотносят допустимые для него действия.

Защита информации от несанкционированного доступа (НСД) –

защита информации, направленная на предотвращение получения защищаемой информации заинтересованными субъектами с нарушением установленных нормативными и правовыми документами (актами) или обладателями информации прав или правил разграничения доступа к защищаемой информации.

Для защиты от НСД, как правило, используется идентификация, аутентификация и управление доступом. В дополнение к перечисленным, могут применяться и другие методы.

Идентификация – присвоение пользователям идентификаторов (уникальных имен или меток) под которыми система «знает» пользователя.

Кроме идентификации пользователей, может проводиться идентификация групп пользователей, ресурсов АС и т. д.

Идентификация нужна и для других системных задач, например, для ведения журналов событий.

В большинстве случаев идентификация сопровождается аутентификацией.

Аутентификация – установление подлинности – проверка принадлежности пользователю предъявленного им идентификатора.

Например, в начале сеанса работы в АС пользователь вводит имя и пароль. На основании этих данных система проводит идентификацию (по имени пользователя) и аутентификацию (сопоставляя имя пользователя и введенный пароль).

Управление доступом – метод защиты информации путем регулирования использования всех ресурсов системы.

Система идентификации и аутентификации является одним из ключевых элементов инфраструктуры защиты от НСД любой информационной системы. Обычно выделяют 3 группы методов аутентификации.

1. Аутентификация по наличию у пользователя уникального объекта заданного типа. Иногда этот класс методов аутентификации называют по-английски “I have” («у меня есть»). В качестве примера можно привести аутентификацию с помощью смарт-карт или электронных USB-ключей.

2. Аутентификация, основанная на том, что пользователю известна некоторая конфиденциальная информация – “I know” («я знаю»). Например, аутентификация по паролю. Более подробно парольные системы рассматриваются далее в этом разделе.

3. Аутентификация пользователя по его собственным уникальным характеристикам – “I am” («я есть»). Эти методы также называются биометрическими.

Глава 14. Клиент-серверные базы данных. Распределенные базы данных.

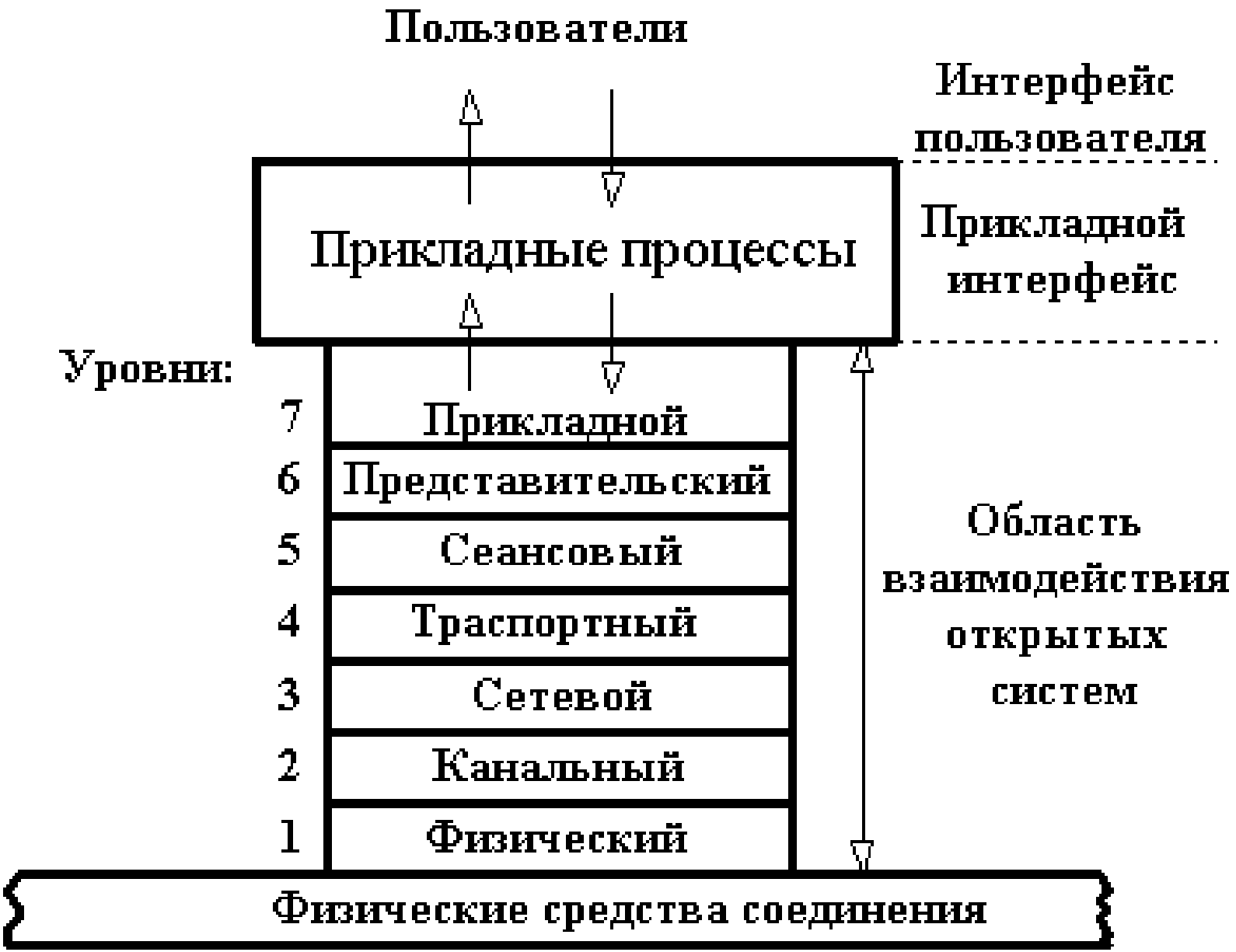
§ 14.1. Модель открытых систем.

В 1984 г. ряд международных организаций (в их число входила и организация ISO – International Standards Organization) разработали стандартную модель сетевого взаимодействия, которую назвали моделью взаимодействия открытых систем (***Open System Interconnection – OSI***).

В модели OSI все протоколы сети делятся на семь уровней:

- ***физический,***
- ***канальный,***
- ***сетевой,***
- ***транспортный,***
- ***сеансовый,***
- ***представительный,***
- ***прикладной.***

Уровни управления ЛВС:


Пользователи
Интерфейс пользователя
Прикладные процессы
Прикладной интерфейс
Уровни:
7 Прикладной
6 Представительский
5 Сеансовый
4 Траспортный
3 Сетевой
2 Канальный
1 Физический
Область взаимодействия открытых систем
Физические средства соединения

Физический уровень имеет дело с передачей битов информации по физическим каналам связи. Такими каналами могут быть, например, коаксиальный кабель, витая пара, оптоволоконный кабель. На этом уровне стандартизируются характеристики электрических сигналов, уровни напряжения и тока, тип кодировки информации, скорость передачи сигналов, а также типы разъемов и назначение каждого контакта.

Канальный уровень обеспечивает надежную передачу данных через физический канал. Канальный уровень оперирует блоками данных, называемыми кадрами. Основной задачей канального уровня является прием кадра из сети и отправка его в сеть, при выполнении чего осуществляется физическая адресация передаваемых сообщений, контролируется соблюдение правил использования физического канала, выявляются неисправности, управляется потоками информации.

Для реализации протоколов канального уровня используется специальное оборудование: *концентраторы, мосты, коммутаторы.*

Сетевой уровень служит для образования единой системы, объединяющей несколько сетей, которые могут быть различной топологии и обладать произвольной структурой. Главной задачей сетевого уровня является маршрутизация. Сети соединяются между собой специальными устройствами, называемыми ***маршрутизаторами***, которые собирают данные о топологии межсетевых соединений и на ее основании пересылают пакеты информации из одной сети в другую. Последовательность маршрутизаторов, через которые проходит пакет, называется ***маршрутом***, а выбор маршрута – ***маршрутизацией***. Примером протокола сетевого уровня является протокол межсетевого взаимодействия ***IP*** стека TCP/ IP.

Транспортный уровень предназначен для оптимизации передачи данных от отправителя к получателю с той степенью надежности, которая требуется.

Основная задача транспортного уровня – обнаружение и исправление ошибок в сообщениях, пришедших с описанных выше уровней. Начиная с транспортного уровня, все дальнейшие протоколы реализуются ***программным*** обеспечением компьютера, обычно включаемого в состав сетевой операционной системы. Примером транспортного протокола является TCP стека TCP/ IP.

Сеансовый уровень устанавливает соединение между сервером и клиентом и управляет им, при необходимости восстанавливает разорванное соединение. На этом уровне устанавливаются правила начала и завершения взаимодействия и определяется, какая из сторон является активной в данный момент, а какая принимает данные.

Представительный уровень выполняет преобразование данных между устройствами с различными форматами данных, не меняя при этом содержания. На этом уровне происходят шифрование и дешифрование данных, благодаря которым обеспечивается секретность передаваемого сообщения.

Прикладной уровень является пользовательским интерфейсом для работы с сетью. Данный уровень непосредственно взаимодействует с пользовательскими прикладными программами, предоставляя им доступ в сеть. С помощью протоколов указанного выше уровня пользователи сети получают доступ к разделяемым ресурсам, таким, как файлы, принтеры, гипертекстовые Web-страницы, электронная почта и т.д.

Замечание: Группа программных, программно-аппаратных или просто аппаратных модулей, составляющих каждый уровень, формируется так, чтобы все модули уровня для выполнения возложенного на них функционала обращались с запросами только к модулям соседнего НИЖЕЛЕЖАЩЕГО уровня.

Для взаимодействия между уровнями одного и то же узла используются интерфейсы, определяющие набор функций, которые нижележащий уровень предоставляет уровню, расположенному выше. Для обмена сообщениями между модулями одного и того же уровня, но расположенными на разных узлах, разрабатываются протоколы.

Протокол – это формализованные правила, с помощью которых осуществляется сетевое взаимодействие расположенных в разных узлах модулей одного и того же уровня.

Интерфейс – это формализованные правила, с помощью которых взаимодействуют находящиеся в одном узле модули, реализующие протоколы соседних уровней.

§ 14.2. Клиент-серверные базы данных

Наиболее эффективную работу с централизованной БД обеспечивает архитектура ***клиент/сервер***.

Централизованная БД характеризуется тем, что полностью находится на центральном компьютере, к которому пользователи (клиенты) обращаются за информацией с помощью своих компьютеров.

Компьютеры называемые **клиентами**, занимаются обработкой прикладных программ. Компьютеры, называемые **серверами**, занимаются обработкой БД.

Централизованная архитектура СУБД

Тип компьютеров, используемых в качестве ***клиентов*** может быть разным – это могут быть большие ЭВМ или микрокомпьютеры. Однако, как правило, функции клиентов выполняют почти всегда ПК.

В роли ***сервера*** может выступать компьютер любого типа, но по экономическим причинам функции сервера чаше всего также выполняют ПК, но имеющие более высокую производительность.

На сервере сети размещается БД и устанавливается мощная серверная СУБД – ***сервер баз данных***.

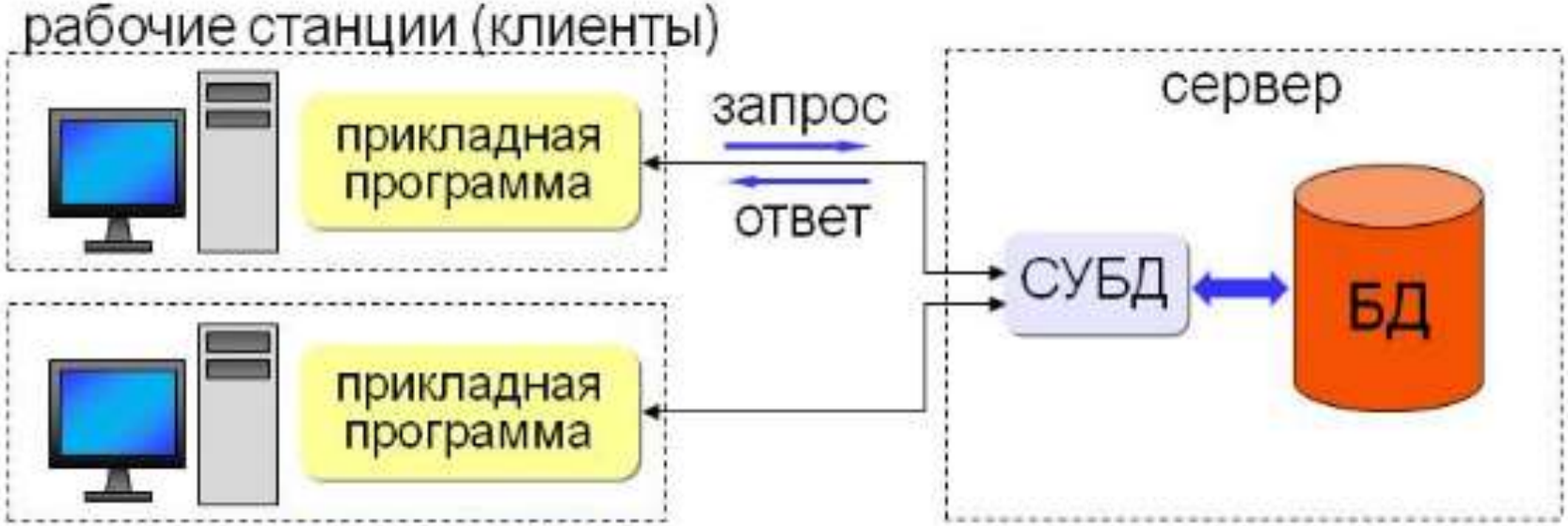
Сервер БД – это программный компонент, обеспечивающий хранение больших объемов информации, ее обработку и представление ее пользователям в сетевом режиме. На компьютере-клиенте приложение-клиент формирует запрос к БД. Серверная СУБД обеспечивает интерпретацию запроса, его выполнение, формирование результата запроса и пересылку его по сети на клиентский компьютер.

Клиентское приложение интерпретирует его необходимым образом и представляет пользователю. Клиентское приложение может также посылать запрос на обновление БД и серверная СУБД внесет необходимые изменения в БД.

Примечание: Полноценное архитектурное решение «клиент-сервер» появилось на свет одновременно с всеобщим признанием концепции открытых систем, задачей которой является стандартизация аппаратных и программных интерфейсов.

Схема архитектуры клиент/сервер:

Центральным звеном большинства клиент-серверных БД выступает СУБД.



На сегодняшний день клиент-серверные СУБД стали наиболее востребованной архитектурой в проектах БД, что произошло благодаря следующим достоинствам:

1) повышение производительности системы за счет перераспределения обязанностей между сервером и клиентами;

2) централизованное хранение данных приводит к повышению целостности и непротиворечивости данных, т.к. все ограничения проверяются в одном месте;

3) существенно упрощается решение задачи обеспечения безопасности данных;

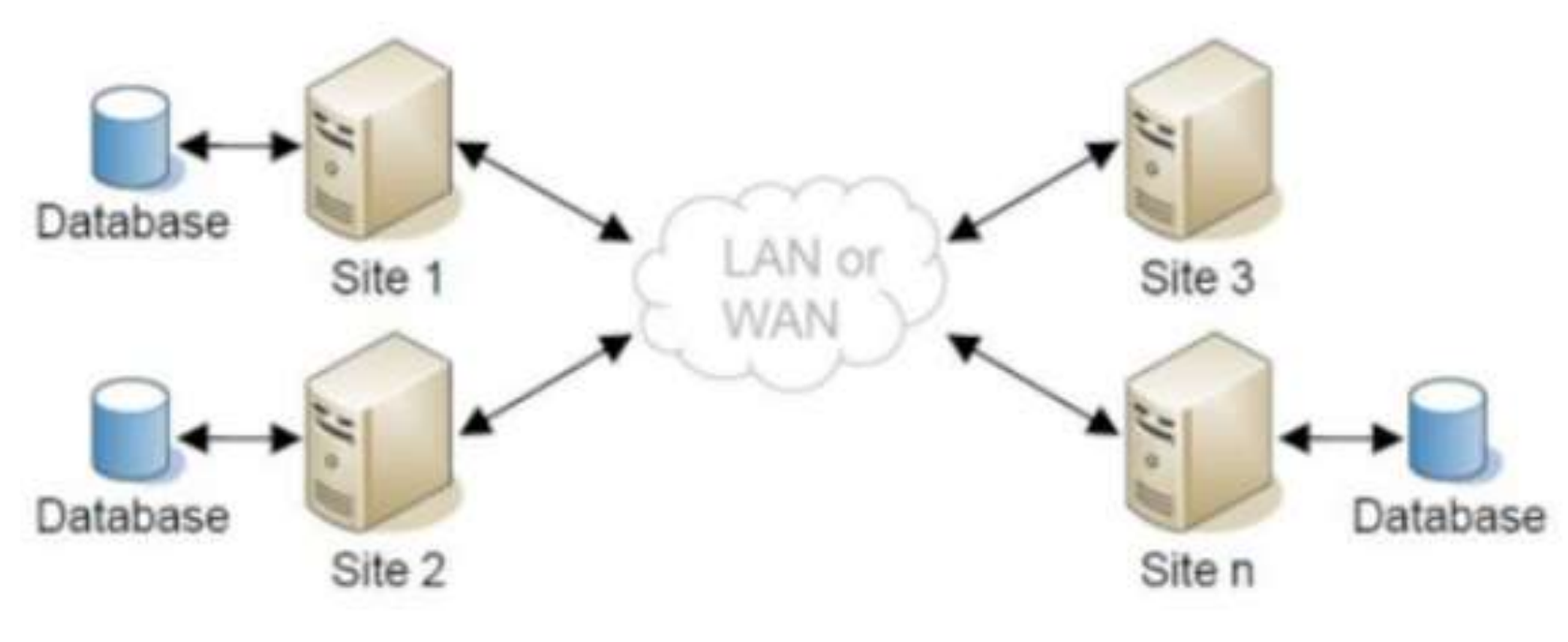
4) снижается нагрузка на сеть предприятия, т.к. обмен между клиентом и сервером сводится к передаче SQL-запроса и возврату запрашиваемых данных.

Работа клиент-серверных систем: клиентский компьютер отправляет серверу запрос, построенный на основе языка SQL, а сервер возвращает клиентскому компьютеру результаты ее выполнения, например, выборку определенных данных.

Примеры клиент-серверных СУБД: *MySQL, InterBase (компании Embarcadero), Oracle, Informix компании IBM), Sybase, SQL Server (компании Microsoft), DB2, NetWare SQL, PostgreSQL* и т.д.

§ 14.3. Распределенные базы данных

Распределенная БД (distributed database) представляет собой набор логически связанных между собой данных, которые физически распределены по нескольким компьютерам (узлам) в некоторой компьютерной сети.



Система управления распределенными базами данных (СУРБД) – это программыный комплекс, предназначенный для управления состоит из единой логической базы данных, разделенной на некоторое количество фрагментов. Каждый фрагмент БД сохраняется на одном или нескольких компьютерах, которые соединены между собой линиями связи и каждый из которых работает под управлением отдельной СУБД. Любой из сайтов способен независимо обрабатывать запросы пользователей, требующие доступа к локально сохраняемым данным (что создает определенную степень локальной автономии), а также способен обрабатывать данные, сохраняемые на других компьютерах сети.

Состав распределенной СУБД:

1. Стандартная СУБД, предназначенная для управления локальными данными на сайте.
2. Программное обеспечение, предназначенное для организации сетевого взаимодействия сайтов между собой.
3. Глобальный системный каталог, содержащий служебную информацию о параметрах, специфичных для распределенной системе.
4. Собственно СУРБД, выступающая управляющим элементом над всеми перечисленными выше компонентами.

Характерные аспекты проектирования распределенных БД:

1. Фрагментация – разбиение БД или таблицы бД на несколько частей и хранение этих частей на разных узлах РБД (распределение по сайтам системы). Каждый фрагмент БД хранится на одном или нескольких компьютерах (узлах, sites), которые соединены между собой коммуникационной сетью и каждый из которых работает под управлением отдельной СУБД.

Наиболее простой случай фрагментации предполагает, что таблицы БД размещаются по тем сайтам, на которых они окажутся наиболее востребованными.

2. Репликация – создание и хранение копий одних и тех же данных на разных узлах РБД. Может комбинироваться с фрагментацией.

Репликация позволяет пользователям, работающим за разными компьютерами, обмениваться изменениями, вносимыми в БД, и осуществлять одновременный доступ к собственным копиям. Изменения, вносимые в основную реплику, распределяются по всем реплицируемым объектам – таблицам, формам, запросам, отчетам, макросам, модулям. В создаваемом наборе копий БД одна копия является **основной** репликой (копией) – именно в нее разрешается вносить изменения.

3. Распределение – рациональное распределение фрагментов по сайтам. Связано с особенностями использования хранящихся в БД данных.

Распределенные ограничения целостности – ограничения, для проверки выполнения которых требуется обращение к другому узлу РБД.

Распределенные запросы – это запросы на чтение, обращающиеся более чем к одному узлу РБД.

Распределенные транзакции – команды на изменение данных, обращающиеся более чем к одному узлу РБД.

Глава 15.

Интерактивная аналитическая обработка OLAP.

Многомерные базы данных и хранилища данных.

§ 15.1. Многомерные БД

Понятие многомерной БД

Многомерной называют базу данных, в которой данные организованы не в виде множества связанных двумерных таблиц, а в виде упорядоченных многомерных массивов.

В многомерных СУБД информация является логически целостной.

Это уже не просто наборы строковых и числовых значений, а целостные структуры типа «кому, что и в каком количестве было продано на данный момент времени».

Основные свойства, присущие многомерным БД:

- ✓ **Агрегируемость данных** - означает рассмотрение информации на различных уровнях ее обобщения. В информационных системах степень детальности представления информации для пользователя зависит от его уровня: аналитик, пользователь, управляющий, руководитель.
- ✓ **Историчность данных** - предполагает обеспечение высокого уровня статичности собственно данных и их взаимосвязей, а также обязательность привязки данных ко времени.
- ✓ **Прогнозируемость данных** - подразумевает задание функций прогнозирования и применение их к различным временным интервалам.

Преимуществами многомерных баз данных являются:

- ✓ поиск и извлечение данных производится значительно быстрее, чем в реляционных базах, поскольку многомерная база данных денормализована и содержит заранее вычисленные агрегаты;
- ✓ более простая процедура встраивания функций в многомерную модель данных;
- ✓ стоимость поддержки многомерной базы данных в среднем ниже, чем у реляционной.

В качестве недостатков многомерных БД можно выделить:

- ✓ сложность изменения структуры данных;
- ✓ неэффективное использование памяти.

§ 15.2. Интерактивная аналитическая обработка OLAP

В основе концепции OLAP лежит принцип многомерного представления данных.

OLAP (On Line Analytical Processing – интерактивная аналитическая обработка данных) – один из способов представления и анализа данных. При этом информация представляется в виде многомерного куба с возможностью произвольного манипулирования ею. Многомерные модели рассматривают данные либо как факты с соответствующими численными параметрами, либо как текстовые измерения, которые характеризуют эти факты.

В многомерной БД параметры представляют свойства факта, который пользователь хочет изучить. Основной таблицей хранилища данных является таблица фактов. Обычно такие таблицы содержат сведения об объектах или событиях, совокупность которых будет в дальнейшем анализироваться.

Таблицы измерений содержат практически неизменяемые данные. **Измерение** – упорядоченный набор значений, принимаемых конкретным параметром и соответствующих одной из граней гиперкуба. Измерения качественно описывают исследуемый процесс, они могут быть числовыми, но в любом случае это данные дискретные, т. е. их значения принадлежат ограниченному набору.

Факты (меры) – это данные, которые количественно описывают процесс, они непрерывны, т. е. могут принимать бесконечное количество значений. Факты представляют субъект – некий шаблон или событие, которые необходимо проанализировать. В большинстве многомерных моделей данных факты однозначно определяются комбинацией значений измерений; факт существует только тогда, когда ячейка для конкретной комбинации значений не пуста.

Параметры – свойства факта. Параметры состоят из двух компонентов: 1) численная характеристика факта, например, цена или доход от продаж; 2) формула, обычно простая агрегативная функция, скажем, сумма, которая может объединять несколько значений параметров в одно. Свойство и формула выбираются таким образом, чтобы представлять осмысленную величину для всех комбинаций уровней агрегирования.

Концепция OLAP систем основана на следующих понятиях:

1) **Многомерный куб** (n – число измерений многомерного куба) – это многомерная структура, состоящая из множества ячеек и хранящая взаимосвязанные данные, описывающие предметную область или её составляющую.

Ячейка является атомарной структурой куба и при отображении располагается внутри него.

Многомерный куб может создаваться и храниться как на физическом уровне представления данных (многомерные OLAP системы), так и на концептуальном уровне (виртуально) с помощью схем «звезда» или «снежинка» (реляционные OLAP системы).

2) **Показатель q_j** (j = 1...p, p – количество показателей) – это типизированная величина, которая является предметом анализа (*например, показателями являются величина закупок, величина продаж, посещаемость сайта и т.д.).*

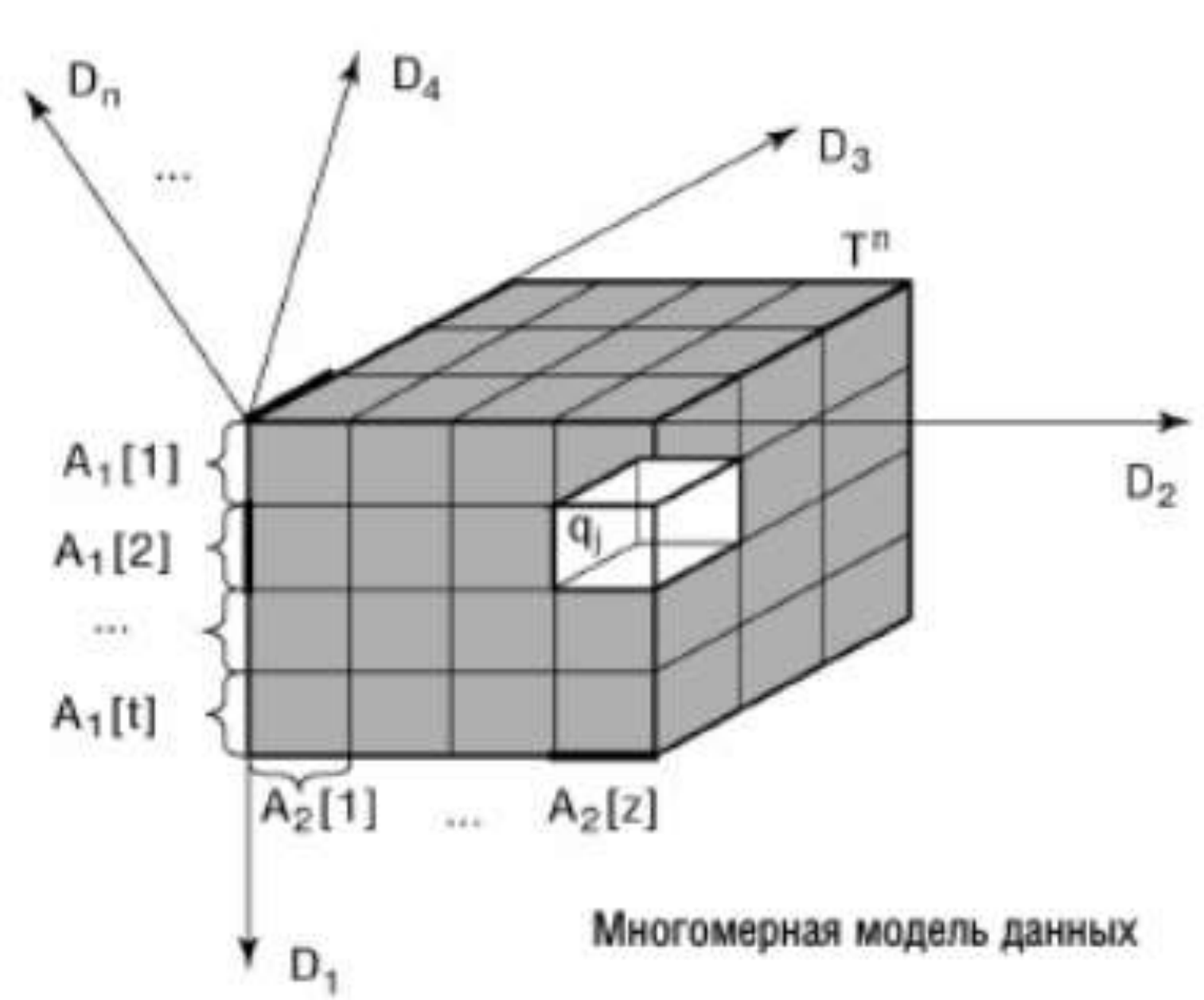
Один многомерный куб может содержать несколько показателей, причём каждому конкретному значению показателя соответствует единственная ***ячейка*** многомерного куба.

3) **Измерение D_i** (i = 1...n) – это множество объектов (называемых членами измерения или значениями измерения) одного или нескольких типов, организованных в виде иерархической структуры и обеспечивающих информационный контекст типизированного показателя. *Например, измерение «товар» может состоять из следующих значений: «бытовая техника», «компьютеры», «мобильные телефоны» и т.д.*

Значения, связанные с измерением, характеризуют какое либо классификационное свойство сущностей предметной области.

Совокупность измерений образует множество измерений $D = \{D_1, D_2, \ldots, D_n\}$. Измерение принято отображать в виде ***ребра*** многомерного куба, а члены измерения – в виде точек или участков, откладываемых на ребре.

Множество всех измерений куба образует ***систему координат*** представляемого пространства данных. Значения измерений играют роль индексов, находящихся в ячейках куба. Особенностью измерений является их иерархическая структура, которая используется для ***агрегации*** и ***детализации*** значений показателей.



Многомерная модель данных

ОСНОВНЫЕ ТИПЫ OLAP СИСТЕМ

Техники обработки и варианты хранения информации в БД классифицируют системы OLAP на несколько видов:

- системы **ROLAP**, работающие с реляционными БД, в которых данные сгруппированы в табличной форме. В системах такого типа возможна аналитика информации в виде чисел и текстов.
- **MOLAP** системы – многомерные системы, в которых данные в ходе обработки структурируются в OLAP кубы на специализированных серверах.
- **HOLAP** системы – «смешанные» системы, в которых объединены алгоритмы многомерной структуризации данных в форме кубов и размещения их в реляционных таблицах.

ОСОБЕННОСТИ И СТРУКТУРА OLAP СИСТЕМ

OLAP системы включают ключевые компоненты:

- **базу данных (БД)** –- источник, из которого берется информационный материал для обработки. Тип БД определяется разновидностью OLAP системы и порядком выполнения действий OLAP сервера. Чаще всего пользуются реляционными и многомерными БД и хранилищами данных;
- **OLAP сервер** – ядро системы, с помощью которого проводится обработка многомерных структур данных, и обеспечивается связь между БД и пользователями систем;
- **приложения для работы пользователей**, в которых формируются запросы и визуализируются полученные ответы.

Специфика обработки данных OLAP системами состоит в построении многомерных, то есть имеющих большое количество связей между отдельными элементами, массивов информации.

Для формирования таких массивов OLAP система собирает данные из различных источников (например, из хранилищ данных, из информационных систем управления предприятием (ERP) или из системы взаимодействия с клиентами (CRM)).

После этого информация обрабатывается на OLAP сервере и передается в пользовательские приложения.

§ 15.3. Хранилище данных (Data Warehouse)

Хранилище данных представляет собой предметно-ориентированный, интегрированный, неизменчивый и поддерживающий хронологию набор данных, организованный для целей поддержки принятия решений.

Хранилища данных позволяют эффективнее, быстрее и качественнее предоставлять данные для систем их аналитической обработки, чем обычные СУБД.

Предметная ориентированность означает, что данные в хранилище объединены в соответствии с областями, которые они описывают, а не с приложениями, которые их используют.

Интегрированность означает, что хранилище должно поддерживать совместное хранение данных различной природы, форматов и типов, отражающих различные аспекты предметной области, а не отдельные бизнес-функции. Данные содержатся внутри хранилища в его едином внутреннем формате.

Поддержка хронологии указывает на то, что хранение данных организовано с учетом даты и времени их появления, для чего каждой записи присваивается специальная метка времени (time stamp), что позволяет извлекать данные в хронологическом порядке и анализировать временные последовательности.

Неизменчивость подразумевает, что для данных в хранилище предусмотрена только операция добавления, а удалять или изменять их нельзя. Если какие-либо изменения все же необходимы, «перегружается» все хранилище целиком. Необходимость такого подхода объясняется тем, что при промышленной эксплуатации хранилища совместно с аналитическими платформами один и тот же запрос к нему, выполняемый в любое время, должен обеспечить предоставление одних и тех же данных. Очевидно, что если бы в хранилище были разрешены изменения, то два одинаковых запроса, выполняемые с некоторым интервалом, в течение которого данные могли измениться, сформируют два различных набора данных, анализ которых может привести к некорректным заключениям и выводам, что недопустимо.

Кроме собственно данных, описывающих бизнес-процессы компании, в хранилище содержатся метаданные – служебные данные, описывающие структуру хранилища, содержащие информацию о принадлежности данных к тому или иному типу или виду (измерение, атрибут или факт). С помощью метаданных формируется семантический слой, который обеспечивает визуальные средства управления данными и метаданными.

Метаданные в хранилищах разделяют на технические (обеспечивают работу самого хранилища), и бизнес-метаданные (описывают структуру данных в рамках заданной бизнес-модели).

В промышленной эксплуатации основными источниками данных для хранилищ являются OLTP-системы. Кроме этого, источниками быть любые файлы в информационной системе предприятия, где содержится структурированная информация, анализ которой, как ожидается, может дать полезные знания. Такие файлы могут иметь различные типы и форматы – электронные таблицы (Excel), настольные СУБД (Access), текст с разделителями (TXT, CSV-файлы), файлы учетных систем (1С:Предприятие, Парус) и т.д. Поэтому для хранилищ данных очень важно иметь развитые средства для загрузки и интегрирования данных из различных типов и форматов.

Лабораторная работа № 1

Тема: *Знакомство с СУБД Microsoft Access.*

Задание:

1. Просмотреть основные возможности СУБД ACCESS на примере базы данных «Борей»:

   1.1. запустить утилиту Total Commander;

   1.2. в командную строку 

скопировать спецификацию файла БД:

   W:\OF2a\PFILES\MSOFFICE\OFFICE\SAMPLES\NWIND.MDB

   1.3. нажать **ENTER**.

2. Создать базу данных в Access, на основе одного из предлагаемых шаблонов:

   2.1. выбрать команду ***п.м. Файл → Создать***;

   2.2. в окне «Область задач» выбрать опцию ***«На моем компьютере…»*** в разделе **Шаблоны**;

   2.3. в окне ***Шаблоны*** перейти на вкладку **Базы данных** и выбрать БД «Контакты»;

   2.4. в появившемся окне «Файл новой БД» указать путь для её сохранения: R:\БД;

   2.5. следовать шагам *Мастера* создания БД;

   2.6. закрыть главную кнопочную форму;

   2.7. развернуть окно БД и изучить организацию связей между таблицами в окне Схемы данных.

3. Найти и изучить в системе помощи (клавиша F1) информацию по следующим темам:

1 способ (с использованием оглавления) – в окне «Справка Access» выбрать гиперссылку **Оглавление**, далее передвигаться по следующим разделам: *Создание баз данных и объектов и работа с ними → Объекты баз данных → Работа с таблицами → Создание таблиц → Создание таблицы (MDB) →*

- ✓ *Создание таблицы при помощи мастера таблиц;*
- ✓ *Создание таблицы в режиме конструктора.*

2 способ (с использованием строки поиска) – в окне «Справка Access» в поле ввода набрать критерий поиска «Ключевое поле»  , в окне результатов выборки выбрать раздел *Определение или изменение первичного ключа (MDB)*.

4. Создать новую базу данных в MS Access:

   4.1. выбрать команду ***п.м. Файл → Создать***;

   4.2. в окне «Область задач» выбрать опцию ***«Новая БД…»*** в разделе **Создание файла**;

   4.3. выбрать путь для сохранения файла новой БД R:\БД, задать имя БД – ***База1***.

5. В новой БД создать таблицу с помощью **Мастера таблиц**:

   5.1. в окне БД в режиме отображения объектов «Таблицы» выбрать опцию Создание таблицы с помощью мастера:

   - ✓ на 1-ом шаге мастера выбрать в группе ⊙ ***Деловые*** образец таблицы **«Задачи»**, включить все поля в новую таблицу 

, переименовать любое из полей;

✓ на 2-ом шаге мастера задать имя таблице, оставить выбор способа определения ключа
⊙ ***…автоматически***…;

✓ на 3-ем шаге мастера оставить вариант дальнейших действий после создания таблицы:
⊙ ***Ввести данные непосредственно в таблицу***, **ГОТОВО**;

5.2. в режиме таблицы ввести произвольные две-три записи.

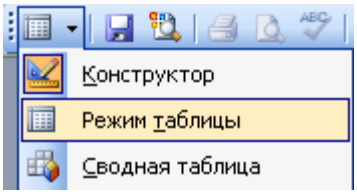
6. Изучить информацию по использованию свойства поля **«Маска ввода»**.

7. В новой БД создать таблицу «Курсы валют» в режиме **Конструктора**:

7.1. создать структуру таблицы:

Имя поля	***Тип данных***	***Свойства поля***	
		название св-ва	***значение св-ва***
Дата	Дата/время	Формат поля	Краткий формат даты
		Маска ввода	запустить мастер по созданию масок ввода [...]: ✓ на 1-ом шаге выбрать **Краткий формат даты**; ✓ на 2-ом шаге в качестве заполнителя выбрать символ «*» и проверить работу маски ввода в поле ввода «Проба»; **ГОТОВО**.
КурсЕвро	Денежный	Формат поля	Евро
		Число десятичных знаков	2
Курс$	Денежный	Формат поля	С разделителями разрядов
		Число десятичных знаков	2
КурсРуб	Денежный	Формат поля	С разделителями разрядов
		Число десятичных знаков	2

7.2. определить ключевым полем поле **Дата** с помощью пиктограммы  на панели инструментов или команды ***п.м. Правка → Ключевое поле***:



7.3. сохранить таблицу;

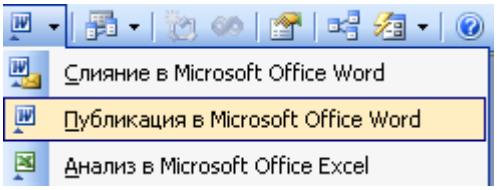
7.4. перейти в режим таблицы и заполнить не менее чем семью записями.

7.5. закрыть таблицу.

8. Созданную таблицу **«Курсы валют»** экспортировать в текстовый редактор Word следующим образом:

8.1. в окне БД в режиме отображения объектов «Таблицы» выбрать таблицу;

8.2. выбрать пункт меню ***Публикация в MS Office Word*** в группе команд «Связи с Office», вызываемых соответствующей пиктограммой на панели инструментов;



8.3. сохранить полученный документ в папку R:\СТиБД как отчет по выполненной работе, добавив в начале документа тему и номер лабораторной работы.

9. Созданную в п. 5 задания таблицу **«Задачи»** экспортировать в текстовый редактор Word с помощью буфера обмена:

9.1. в окне БД в режиме отображения объектов «Таблицы» выбрать и открыть таблицу;

9.2. выделить все данные таблицы (***п.м. Правка → Выделить все записи***);

9.3. скопировать таблицу в буфер (***п.м. Правка → Копировать***);

9.4. перейти в созданный в п. 8.3. документ-отчет;

9.5. вставить таблицу из буфера (***п.м. Правка → Вставить***);

9.6. пересохранить отчет.

Лабораторная работа № 2

Тема: *Проектирование учебной базы данных.*

Задание: 1. Создать средствами СУБД ***MS Access*** базу данных «Автоперевозки», имеющую следующую структуру:

Таблица 1 «Водители»
🗝 **КодВодителя**
Фамилия
Имя
Отчество
Категория
ДатаПоступленияНа-Работу
Адрес
Телефон

Таблица 2 «Типы машин»
🗝 **ТипАвтомобиля**
СредняяСкорость
Грузоподъемность (тонн)
РасходТоплива (на 100 км)

Таблица 3 «Топливо»
🗝 **ВидТоплива**
ЦенаЗаЛитр

Таблица 6 «Маршруты»
🗝 **КодПунктаНаз-начения**
ПунктНазначения
Область
Район
Расстояние (км)

Таблица 4 «Парк машин»
🗝 **НомерМашины**
ТипАвтомобиля
ГодВыпуска
ДатаПоследне-гоТехосмотра
ВидТоплива

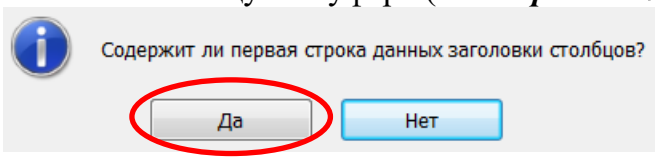
Таблица 5 «Перевозки»
🗝 **НомерПере-возки**
ДатаПеревозки
НомерМашины
КодВодителя
ВесГруза (кг)
ПунктНазначения

🗝 – ключевое поле

1.1. создать файл новой базы данных (**БД**) на рабочем диске R:\БД\;

1.2. из лабораторной работы № 1, ***импортировать*** справочные таблицы (таблицы № 1 и № 2), располагаемые на соответствующем рабочем листе рабочей книги ***Список.xls*** в отдельные соответствующие таблицы «Типы машин» и «Топливо» в созданную БД:

- ✓ выделить таблицу на рабочем листе Excel без названия, но с заголовками столбцов;
- ✓ скопировать таблицу в буфер (***п.м. Правка → Копировать***);
- ✓ в окне БД перейти на вкладку объектов «Таблицы»;
- ✓ вставить таблицу из буфера (***п.м. Правка → Вставить***);
- ✓



1.3. в режиме ***Конструктора*** откорректировать структуры импортированных в п. 1.2. таблиц: добавить новые поля, изменить при необходимости типы данных и задать дополнительные свойства уже имеющихся полей. Пересохранить таблицы с измененными структурами и **переименовать**. В режиме таблицы заполнить значения новых **полей** произвольными данными.

Замечание: Убрать значение «0» в свойстве ***«Значение по умолчанию»*** для всех полей числового типа!!!

Таблица «Типы машин»

Поле	***Тип***	***Свойства***
ТипАвтомобиля	текстовый	длина 10
СредняяСкорость	числовой	целое, положительное значение
Грузоподъемность (тонн)	числовой	двойное с плавающей точкой, с тремя десятичными знаками
РасходТоплива (на100км)	числовой	двойное с плавающей точкой, с двумя десятичными знаками

Таблица «Топливо»

Поле	***Тип***	***Свойства***
ВидТоплива	текстовый	длина 15
ЦенаЗаЛитр	денежный	денежный формат, с 2 десятич. знаками, положительное значение

1.4. в режиме ***Конструктора*** создать следующие таблицы, задав указанные типы данных и свойства полей:

Таблица «Водители»

Поле	***Тип***	***Свойства***
КодВодителя	числовой	значение целое, положительное
Фамилия	текстовый	длина 30
Имя	текстовый	длина 10
Отчество	текстовый	длина 20
Категория	текстовый	длина 10, фиксированный список значений*: *первая, вторая, высшая*
ДатаПоступленияНаРаботу	дата/время	краткий формат даты, маска ввода по шаблону
Адрес	текстовый	длина 50
Телефон	текстовый	маска ввода: \(900") "900\-00\-00;0;#
ЧленПрофсоюза	логический	тип элемента управления (вкл. Подстановка): *флажок*

**** - заполняется с помощью инструмента «Мастер подстановок» (см. Инструкцию ниже)***

Таблица «Парк машин»

Поле	***Тип***	***Свойства***
НомерМашины	текстовый	длина 10, маска ввода 00\-00\ >LL\-0;0;#
ТипАвтомобиля	текстовый	длина 30, заполняется значениями из таблицы «Типы машин»**
ГодВыпуска	числовой	целое, маска ввода 0000;;#
ДатаПоследнегоТехосмотра	дата/время	краткий формат даты, маска ввода по шаблону
ВидТоплива	текстовый	длина 15, заполняется значениями из таблицы «Топливо»**

***** - выполняется с помощью инструмента «Мастер подстановок» (см. Инструкцию ниже)***

Таблица «Перевозки»

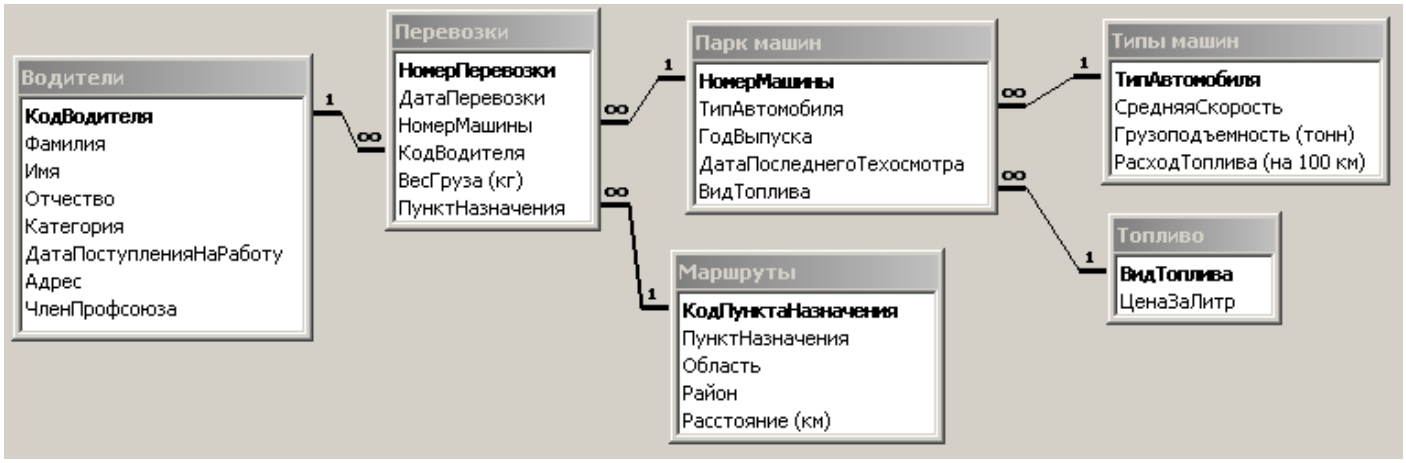
Поле	***Тип***	***Свойства***
НомерПеревозки	счетчик	
ДатаПеревозки	дата/время	краткий формат даты, маска ввода по шаблону, значение по умолчанию: текущая дата
НомерМашины	текстовый	длина 10, заполняется значениями из таблицы «Парк машин»**
КодВодителя	числовой	значение целое, положительное, заполняется значениями из таблицы «Водители»**
ВесГруза (кг)	числовой	длинное целое, положительное значение
ПунктНазначения	текстовый	длина 20, заполняется значениями из таблицы «Маршруты»**

***** - выполняется с помощью инструмента «Мастер подстановок» (см. Инструкцию ниже)***

Таблица «Маршруты»

Поле	***Тип***	***Свойства***
КодПунктаНазначения	числовой	целое
ПунктНазначения	текстовый	длина 40
Область	текстовый	длина 40
Район	текстовый	длина 40
Расстояние (км)	числовой	длинное целое, положительное значение

1.5. Создать связи между таблицами:



1.6. Заполнить созданные в п. 1.4. таблицы произвольными данными:

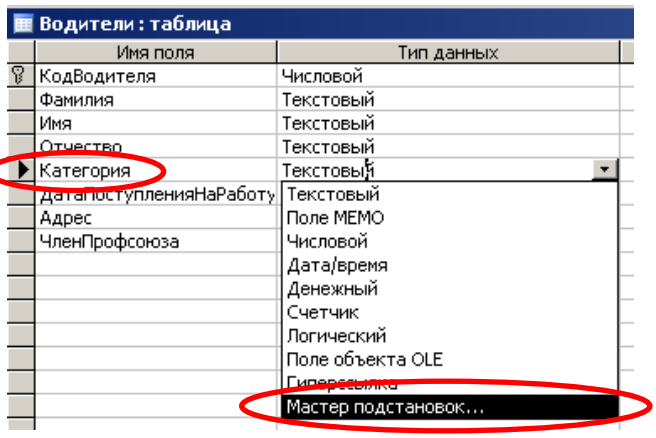
- ✓ в таблице «Маршруты» – 7 записей *(в том числе 2-3 записи с наименованием ПунктаНазначения, заканчивающимся на «ин»)*;
- ✓ в таблице «Водители» – 7 записей;
- ✓ в таблице «Парк машин» – 10 записей;
- ✓ в таблице «Перевозки» – 25 записей, **причем в столбце «ДатаПеревозки» даты вводить (изменять значение текущей даты, установленной по умолчанию) за разные временные периоды (по одной-две даты в разных кварталах) с 2017 по 2021 год !!!**

2. Оформить отчет, в котором разместить скриншот схемы данных и содержимое всех таблиц.

ИНСТРУКЦИЯ по работе с инструментом «Мастер подстановок»

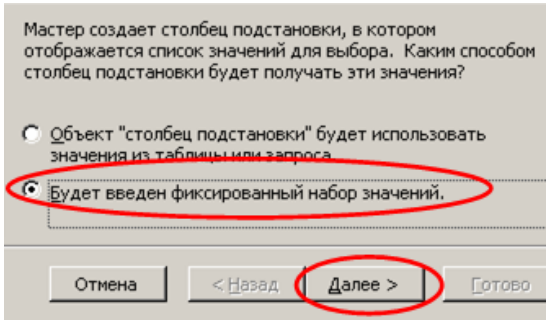
1. Создание **фиксированного** списка значений атрибута (*на примере поля «Категория» в таблице «Водители»*):

1.1. в режиме конструктора таблицы выбрать инструмент «***Мастер подстановок***» в столбце **Тип данных**:

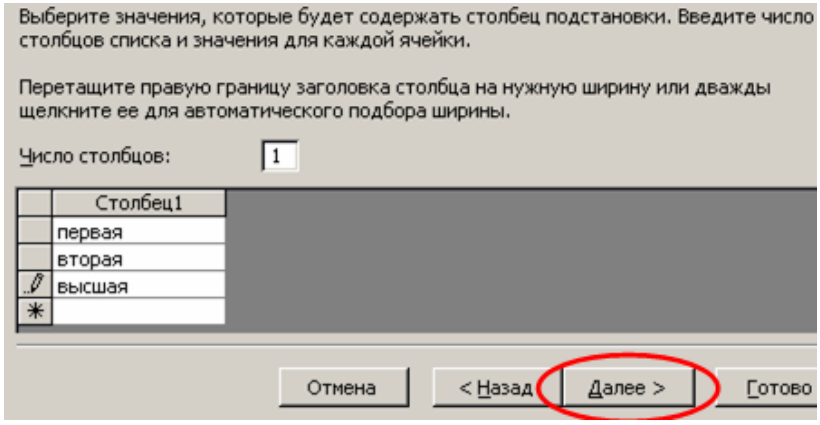


1.2. следовать шагам мастера:

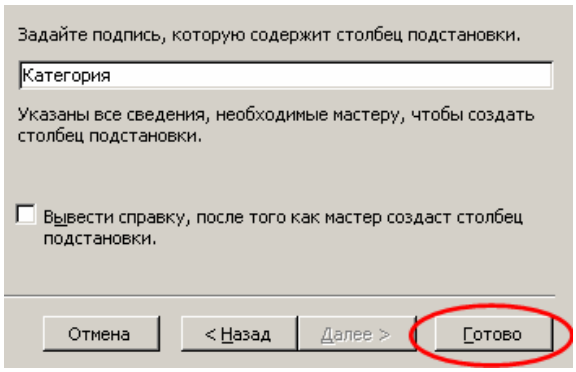
1 шаг:



2 шаг:

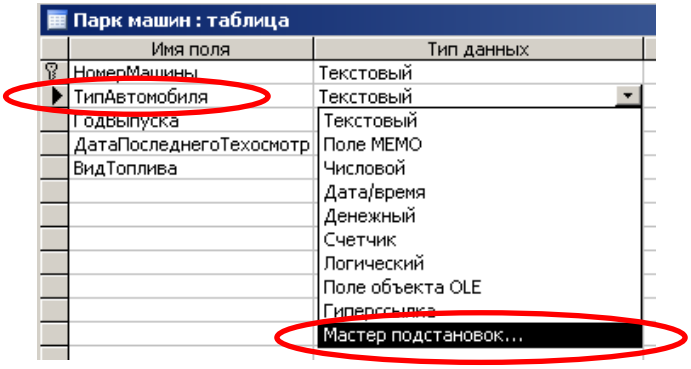


3 шаг:



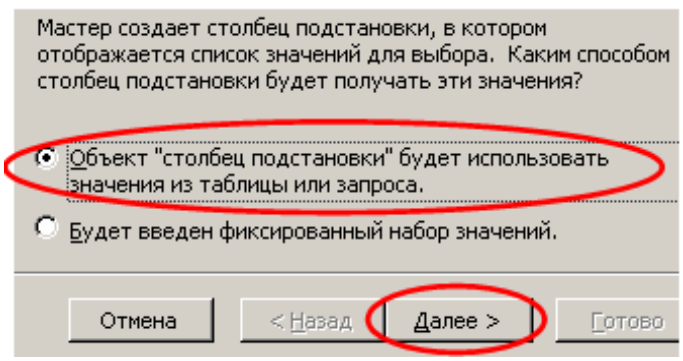
2. Создание **выпадающего** списка значений для поля внешнего ключа (*например, по полю ТипАвтомобиля для связанных таблиц «Парк машин» и «Типы машин»*):

2.1. в режиме конструктора **подчиненной** таблицы выбрать инструмент «***Мастер подстановок***» для поля внешнего ключа:

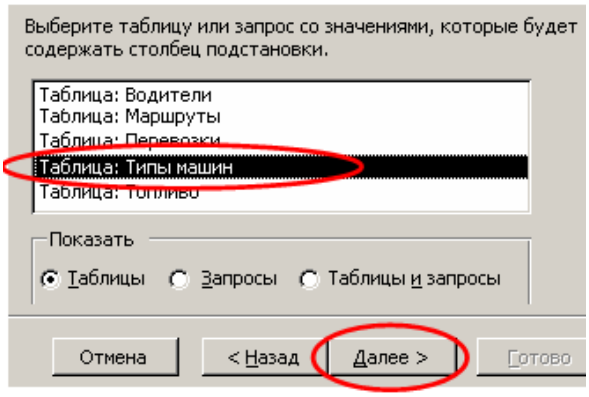


2.2. следовать шагам мастера:

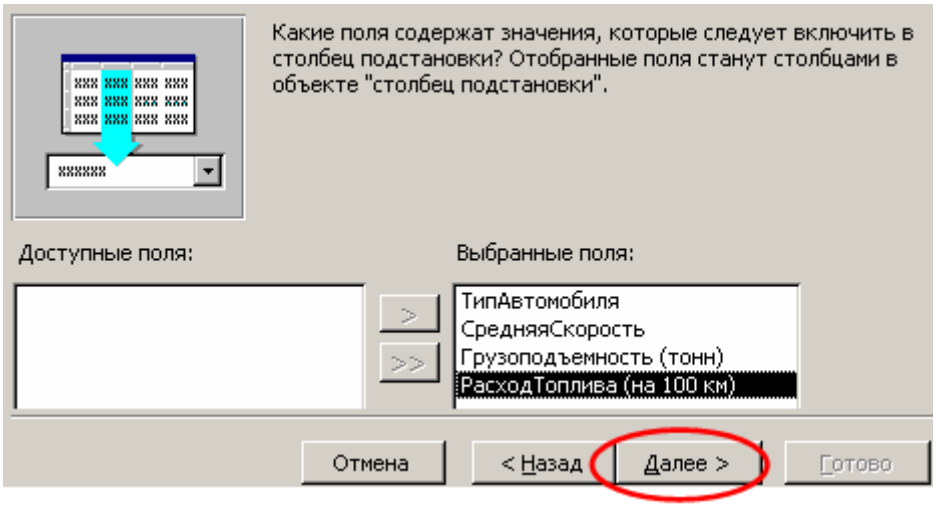
1 шаг:



2 шаг:

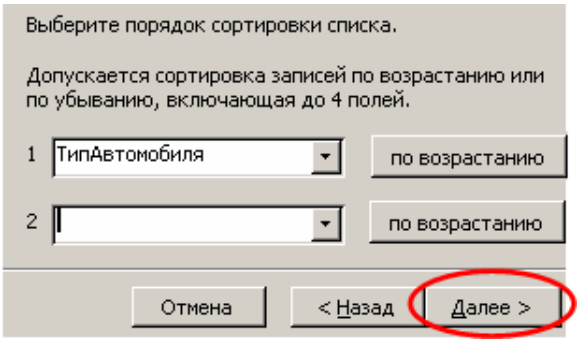


3 шаг:



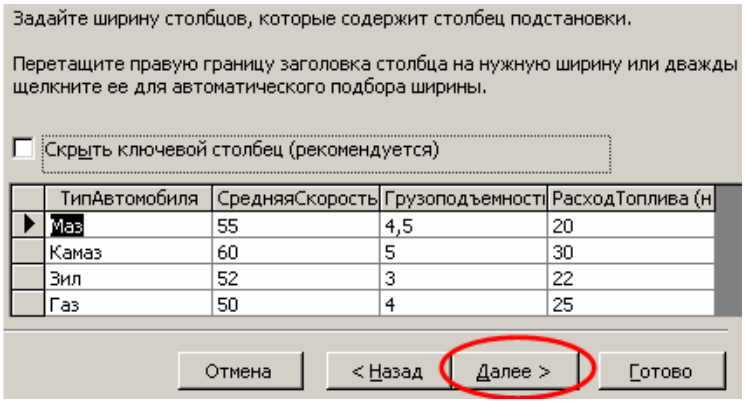
Замечание: НЕ обязательно в столбец подстановки включать все поля главной таблицы!!!

4 шаг:

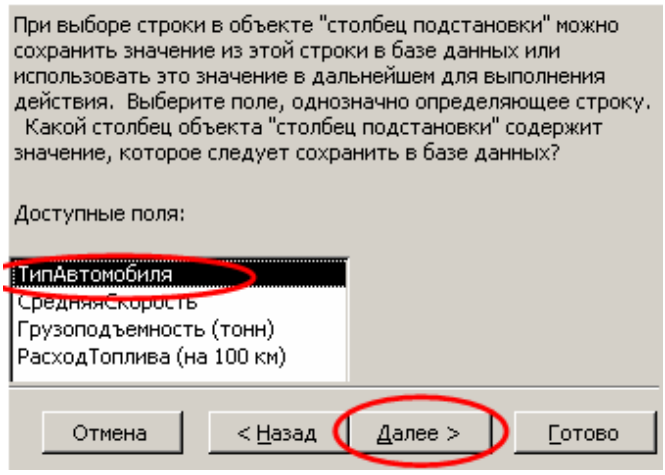


Замечание: НЕ обязательно выполнять сортировку!!!

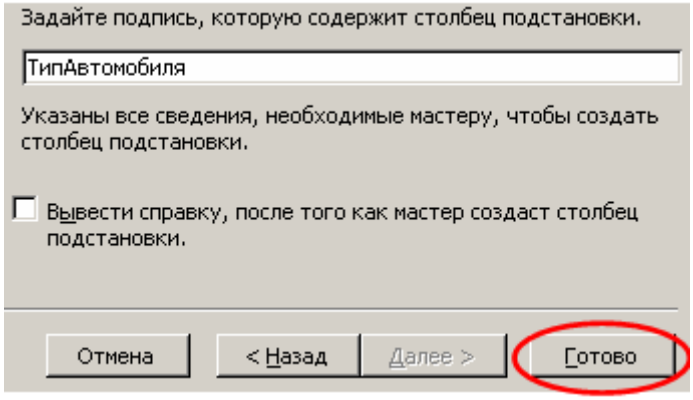
5 шаг:

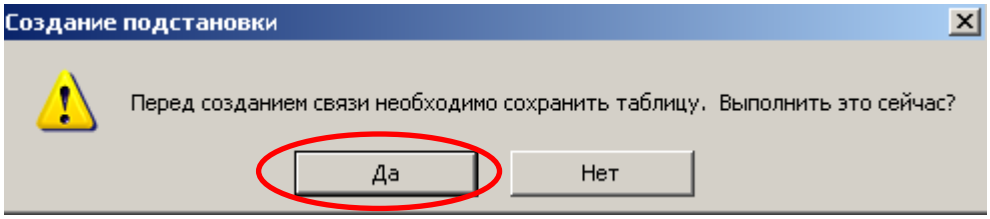


6 шаг:



7 шаг:





Примеры масок ввода

Маски ввода используются в полях (в таблицах и запросах), в текстовых полях и в полях со списком (в формах) для форматирования данных и управления вводимыми значениями. Маска ввода состоит из текстовых символов (таких как точки, тире, скобки), разделяющих пустые интервалы, предназначенные для заполнения. Свойство «Маска ввода» (InputMask) состоит из текстовых и специальных символов, определяющих тип значений, которые могут быть введены в данную позицию. В основном маски ввода используются в текстовых полях и полях Дата/Время, а также в числовых и денежных.

В приведенной ниже таблице указаны описания некоторых масок ввода и примеры значений, которые в них могут быть введены.

Описание маски ввода	***Примеры значений***
(000) 000-0000	(206) 555-0248
(999) 999-9999!	(206) 555-0248
	( ) 555-0248
(000) AAA-AAAA	(206) 555-TELE
#999	- 20
	2000
>L????L?000L0	GREENGR339M3
	МАЙ Р 452Ю7
>L0L 0L0	Т2Ф 8М4
00000-9999	98115-3007
>L<?????????????	Мария Иван
ISBN 0-&&&&&&&&&-0	ISBN 1-55615-507-7
	ISBN 0-13-964262-5
>LL00000-0000	ВФ51392-0493

Описание маски ввода может состоять из трех разделов, разделенных знаком точка с запятой, например, (999) 000-0000!;0;" ".

Раздел	***Значение***
Первый	Сама маска ввода.
Второй	Указывает, следует ли сохранять текстовые символы: **0** = текстовые символы сохраняются вместе с введенными значениями **1 или пустое значение** = сохраняются только введенные символы
Третий	Символ, выводящийся в маске ввода на месте пустых символов. Допускается использование любого символа; для отображения пробела, следует ввести " " (прямые кавычки,

	пробел, прямые кавычки). Если данный раздел описания оставить пустым, то для представления пустых символов используется символ подчеркивания (_).

В приведенной ниже таблице указано, как Microsoft Access интерпретирует символы, содержащиеся в первой части описания в свойстве «Маска ввода». Чтобы включить в маску текстовые константы, отличные от представленных в таблице, в том числе символы и пробелы, следует просто ввести их в нужную позицию. Чтобы включить один из следующих символов в качестве текстовой константы, необходимо перед ним ввести символ обратной косой черты \.

Символ	***Описание***
0	Цифра (от 0 до 9, ввод обязателен; символы плюс [+] и минус [-] не допускаются).
9	Цифра или пробел (ввод не обязателен; символы плюс и минус не допускаются).
#	Цифра или пробел (ввод не обязателен; пустые символы преобразуются в пробелы, допускаются символы плюс и минус).
L	Буква (от A до Z или от A до Я, ввод обязателен).
?	Буква (от A до Z или от A до Я, ввод не обязателен).
A	Буква или цифра (ввод обязателен).
a	Буква или цифра (ввод необязателен).
&	Любой символ или пробел (ввод обязателен).
C	Любой символ или пробел (ввод необязателен).
. , : ; - /	Десятичный разделитель и разделители тысяч, значений дат и времени. (Отображаемый символ зависит от настроек языка и стандартов на панели управления Windows.)
<	Указывает перевод всех следующих символов на нижний регистр.
>	Указывает перевод всех следующих символов на верхний регистр.
!	Указывает заполнение маски ввода справа налево, а не слева направо.

Заполнение маски символами всегда происходит слева направо. Восклицательный знак в маске ввода можно помещать в любую позицию.

\ Указывает ввод любого следующего символа в качестве текстовой константы. Используется для отображения всех перечисленных в данной таблице символов как текстовых констант (например, \A выводится как символ «A»).

Значение ***Пароль***, заданное для свойства «Маска ввода», создает поле для ввода пароля. Любой символ, введенный в поле, сохраняется как символ, но отображается как звездочка (*).

Лабораторная работа № 3 (1)

Тема: *Отбор данных в MsA с помощью фильтра и простых запросов на выборку.*

Задание:

Часть I. Работа с фильтрами в MsA

Можно пользоваться **ИНСТРУКЦИЕЙ** по выполнению лабораторной работы.

1. Организовать выборку данных из таблиц учебной БД «Автоперевозки», используя разные способы создания фильтров:

Замечание: После выполнения фильтров №№ 1.1. – 1.4., таблицы закрывать **БЕЗ** сохранения их макета или структуры (см. в инструкции)!!!

Использование режима «Изменить фильтр»:

1.1. Выбрать в таблице «Парк машин» автомобили определённого типа (значение задается самостоятельно), год выпуска которых позже **N** (значение **N** задается самостоятельно).

1.2. Выбрать записи в таблице «Маршруты», для которых значения по атрибуту ***Расстояние*** попадают в определенный диапазон: более **N1** и менее **N2** (границы диапазона – числа **N1** и **N2** – задать самостоятельно, причем **N1< N2**).

1.3. Выбрать записи в таблице «Маршруты», для которых значения по атрибуту ***Расстояние*** попадают в интервалы: не более **N1** и не менее **N2** (границы интервалов – числа **N1** и **N2** – задать самостоятельно, причем **N1< N2**).

Использование режима «Фильтр по выделенному»:

1.4. Выбрать водителей в соответствующей таблице, чья фамилия начинается на заданную произвольно букву.

Использование режима «Расширенный фильтр»:

1.5. Выбрать автомобили в таблице «Парк машин», вид топлива которых «Бензин*» и дата последнего техосмотра которых ранее ДД.ММ.ГГГГ – задать самостоятельно. Сохранить фильтр как запрос с названием «Фильтр_4_1_5».

Часть II. Запросы-выборки в MsA

2. Создать следующие **запросы-выборки** из таблиц учебной БД «Автоперевозки» (*в окне БД на вкладке «Запросы» командой «Создать в режиме Конструктора»*), сохраняя их с уникальными именами, указывающими номер лаб_работы, номер запроса и краткое содержание (*например, «**Запрос_4_2_1_ЧленыПроф**»*):

2.1. *запрос по текстовому полю по таблице «Водители»,* выводящий водителей, которые являются членами профсоюза;

2.2. *запрос по текстовому полю по таблицам «Перевозки» и «Маршруты»,* выводящий записи о перевозках в пункты назначения, название которых заканчиваются на «ин»;

2.3. *запрос по оператору «И» (И-запрос) по таблицам «Перевозки» и «Маршруты»,* выводящий перевозки, расстояние которых не менее ***N*** км. и вес груза более ***N1*** кг. (границы интервалов ***N*** и ***N1*** задать самостоятельно);

2.4. *запрос по оператору «ИЛИ» (ИЛИ-запрос) по таблице «Типы машин»*, выбирающий автомобили, тип которых МАЗ и КАМАЗ;

2.5. *параметрический запрос по таблицам «Парк машин» и «Топливо»*, выводящий машины, заправляемые топливом, вид которого вводится с клавиатуры;

2.6. *запрос по полю типа Дата/время (И-запрос) по таблице «Перевозки»*, выводящий перевозки, осуществленные в первом квартале 2017 года – использовать проверку условия попадания даты во временной диапазон (с … по…) с операторами сравнения;

2.7. *запрос по полю типа Дата/время (И-запрос) по таблице «Водители»*, выводящий список водителей, поступивших на работу в месяце ***M*** года ***Y*** (***M*** и ***Y*** задать самостоятельно) – использовать **ВСТРОЕННЫЕ ФУНКЦИИ** из категории Дата/Время;

2.8. *параметрический запрос по полю типа Дата/время по временному диапазону (И-запрос) по таблице «Перевозки»,* выводящий перевозки, осуществленные в определенном периоде, границы которого вводятся с клавиатуры – использовать проверку условия попадания даты во временной диапазон (*с … по…*) с операторами сравнения;

2.9. *параметрический запрос по полю типа Дата/время по таблице «Водители»*, выводящий список водителей, поступивших на работу в определенный год, значение которого вводится с клавиатуры – использовать встроенные функции из категории Дата/Время;

2.10. *многотабличный запрос по таблицам «Водители», «Парк машин», «Перевозки» и «Маршруты»* включающий поля всех связанных таблиц БД (***расшифровка*** значений всех ***внешних*** ключей).

3. Из соответствующих таблиц учебной БД «Автоперевозки» выбрать ***перевозки*** по заданным ниже критериям в виде отдельных запросов на выборку. При необходимости использовать **ВСТРОЕННЫЕ ФУНКЦИИ** категорий «Дата/время» и «Текстовые».

Замечание: Возможно **изменять** значения в некоторых записях таблиц БД (участвующих в запросах) под конкретные критерии отбора для того, чтобы результат запроса был не пустым!!!

3.1.	машинами с видом топлива ***Газ*** и ***Бензин АИ-95*** или в пункты назначения, ***названия*** которых начинаются на букву ***«К»***
3.2.	за ***02.11.2016*** и ***12.01.2017*** водителями первой и высшей категорий
3.3.	перевезенных машинами типа ***«МАЗ» до 01.12.2016*** и ***позже 10.01.2017***, вес груза которых ***от 500 до 1 000*** кг.
3.4.	с весом груза ***до N1*** и ***свыше N2*** (N1 и N2 задаются как параметр, причём N1<N2) или с водителями, фамилии которых содержат буквы ***«ов»***
3.5.	дата перевозки которых – ***1-ое полугодие 2017 г.*** или в пункты назначения, количество букв в названии которых вводится как параметр
3.6.	за ***первую декаду месяца (номер которого задается как параметр) 2017 г.*** в пункт назначения ***Кобрин*** и ***Пинск***
3.7.	машинами с грузоподъемностью ***не долее N1*** и ***не менее N2*** (N1 и N2 задать произвольными значениями, причём N1<N2) и давность которых – ***более 30 дней***
3.8.	выполненные в ***выходные дни*** машинами, номера которых содержат цифры «***33***»
3.9.	машинами с разными видами топлива ***кроме «Газ»*** в пункты назначения, названия которых начинаются на буквы ***от «А» до «К»***
3.10.	в пункты назначения ***Высокое, Лунинец*** и ***Пинск*** за определённый **квартал (**вводится как параметр) определённого **года (**вводится как параметр)
3.11.	машинами, название типа которых заканчивается на «**АЗ**» за ***текущий месяц текущего года***
3.12.	за ***2016 год*** с водителями, поступившими на работу ранее, чем за два года до текущей даты

Оформить отчет по данному пункту в виде текстового документа. Можно воспользоваться следующим **БЛАНКОМ** отчета, в который добавлять «скриншоты» бланка конструктора каждого запроса с результатом его выборки.

Примеры выражений для условий отбора

Следующие выражения для условий отбора могут быть введены:

- в строку «Условие отбора» фильтра;
- в строку «Условие отбора» запроса в режиме конструктора;
- в свойстве: «Условие на значение».

Поле	***Выражение***	***Заказы, отобранные в запросе***
Город получателя	"Москва"	Заказы, отправленные в Москву
Город получателя	"Москва" Or "Васюки"	Заказы, отправленные в Москву или Васюки
Дата исполнения	=#2.02.2022#	Заказы, отправленные 2-фев-2022
Дата исполнения	Between #5-янв-2022# And #10-янв-2022#	Заказы отправленные не раньше 5-янв-2022 и не позже 10-янв-2022
Страна получателя	In ("Чили", "Перу", "Россия")	Заказы, отправленные в Чили или Перу или Россию
Страна получателя	Not "США"	Заказы, отправленные во все страны, кроме США
Дата размещения	< Date( ) - 30	Заказы с давностью более 30 дней
Дата размещения	Year([Дата размещения])=2021	Заказы за 2021 г.
Дата размещения	DatePart("q",[Дата размещения])=4	Заказы за полный квартал по календарю
Дата размещения	Year([Дата размещения])=Year(Now()) And Month([Дата размещения])=Month(Now())	Заказы за текущий месяц текущего года
Название получателя	Like "С*"	Заказы, отправленные клиентам, названия фирм которых начинаются на букву «С»
Название получателя	Like "*ия"	Заказы, отправленные клиентам, названия фирм которых заканчиваются буквами "ия"
Название получателя	Like "[А-Г]*"	Заказы, отправленные клиентам, названия фирм которых начинаются с букв от А до Г включительно
Количество дней	Between 1 And 10	Количество дней не должно быть меньше 1 и больше 10

Лабораторная работа № 3(2)

Тема: *Вычисляемые запросы в MsA.*

Задание:

Создать следующие **вычисляемые запросы** для учебной БД «Автоперевозки»:

1. *многотабличный запрос (таблицы «Перевозки», «Типы машин», «Парк машин», «Маршруты»), в котором вывести поля* ***НомерПеревозки, ДатаПеревозки, КодВодителя, НомерМашины*** *и создать новые вычисляемые поля:*

ЧислоЕздок = ВесГруза / Грузоподъемность (тонн) / 1 000

ЧислоЕздок: IIf([Перевозки]![ВесГруза(кг)]/[Типы машин]![Грузоподъемность(тонн)] / 1000 - Int([Перевозки]![ВесГруза(кг)] / [Типы машин]![Грузоподъемность(тонн)] / 1000)>0,5; Round([Перевозки]![ВесГруза(кг)] / [Типы машин]![Грузоподъемность(тонн)] / 1000); Round([Перевозки]![ВесГруза(кг)] / [Типы машин]![Грузоподъемность(тонн)] / 1000)+1)

ВремяВПути = Расстояние * 2 * ЧислоЕздок / СредняяСкорость (округленное до одного знака после запятой – использовать функцию **Round( )** – *см. конспект лекций*);

Пробег = ЧислоЕздок * Расстояние * 2;

!!! Запрос сохранить с именем «Пробег»

2. *запрос по таблице «Водители», в котором вывести поля* ***КодВодителя, Категория, ДатаПоступленияНаРаботу*** *и создать новые вычисляемые поля:*

ФИО водителя = Фамилия + Имя + Отчество *– см. конспект лекций;*

СтажРаботы (лет) *– см. конспект лекций;*

%ПремииЗаСтаж =

Стаж работы в годах	***% премии***
от 10 до 20 лет	***10%***
от 20 до 30 лет	***20%***
свыше 30 лет	***30%***

Замечание*: в свойствах вычисляемого поля установить «Процентный» формат*

НадбавкаЗаКатегорию =

Категория	***Надбавка за категорию***
первая	***1 · БВ***
вторая	***1,5 · БВ***
высшая	***2 · БВ***

БВ – базовая величина, размер которой вводится как параметр.

!!! Запрос сохранить с именем «Надбавки»

Замечание*: в последующих запросах в свойствах всех вычисляемых полей денежного вида установить формат «С разделителями разрядов» и двумя точками после запятой.*

3. *запрос на основе предыдущих запросов «Пробег», «Надбавки» и таблицы «Водители», в котором вывести поля* ***НомерПеревозки, ДатаПеревозки, КодВодителя, ФИО водителя, Категория, НадбавкаЗаКатегорию*** *и создать новые вычисляемые поля:*

СтоимостьРаботы = ВремяВПути * ЦенаЧасаРаботыВодителя (цена 1 часа работы водителя вводится как параметр);

ПремияЗаСтаж = %ПремииЗаСтаж * СтоимостьРаботы;

Надбавка = 50% от СтоимостиРаботы, если перевозка осуществлялась в выходные дни;

Начислено = СтоимостьРаботы + ПремияЗаСтаж + Надбавка + НадбавкаЗаКатегорию;

ПодоходныйНалог = 13% * Начислено;

ПрофВзнос = 1% от суммы «Начислено» (для членов профсоюза);

ЗарплатаВодителя = Начислено – ПодоходныйНалог – ПрофВзнос.

!!! Запрос сохранить с именем «ЗарплатаВодителей»

4. *запрос на основе запроса «Пробег» и таблиц «Перевозки», «Парк машин», «Топливо», в котором вывести поля* ***НомерПеревозки, ДатаПеревозки, НомерМашины, ВидТоплива*** *и создать новые вычисляемые поля:*

ОбъемТоплива = Пробег * РасходТоплива / 100;

СтоимостьТоплива = ОбъёмТоплива * ЦенаЗаЛитр.

!!! Запрос сохранить с именем «СтоимостьТоплива»

5. *запрос на основе запросов «Зарплата водителей», «Стоимость топлива» и таблицы «Перевозки», в котором вывести поля* ***НомерПеревозки, ДатаПеревозки, НомерМашины, ФИО водителя*** *и создать новые вычисляемые поля:*

СтоимостьПеревозки = ЗарплатаВодителя + СтоимостьТоплива.

!!! Запрос сохранить с именем «СтоимостьПеревозки»

6. *запрос по таблице «Парк машин», в котором вывести поля* ***НомерМашины, ГодВыпуска, ДатаПоследнегоТехосмотра*** *и создать новые вычисляемые поля:*

ВозрастМашины (лет) *– см. конспект лекций*

ДавностьПрохожденияТехосмотра (лет)

ДавностьПрохожденияТехосмотра (месяцев)

ДавностьПрохожденияТехосмотра (дней)

Лет: DateDiff("yyyy";[Дата1];[Дата2])-
Iif([Дата2]<DateAdd("yyyy";DateDiff("yyyy";[Дата1];[Дата2]);[Дата1]);1;0)

Месяцев: DateDiff("m";[Дата1];[Дата2])-[Лет]*12-
Iif([Дата2]<DateAdd("m";DateDiff("m";[Дата1];[Дата2]);[Дата1]);1;0)

Дней: DateDiff("d";DateAdd("m";[Лет]*12+[Месяцев];[Дата1]);[Дата2])

Замечание: *Дата1 – [Дата последнего техосмотра],*

Дата2 = [Текущая дата]

ДатаСледующегоТехосмотра =

Возраст машины (полных лет)	***Периодичность прохождения техосмотра***
до 10	***2 года***
не менее 10 лет	***1 год***

Сообщение = вывести комментарий в виде текста «Срочно пройти техосмотр!» для тех машин, ДатаСледующегоТехосмотра у которых ранее текущей даты, а для машин, у которых ДатаСледующегоТехосмотра попадает в текущий месяц текущего года вывести сообщение «Готовиться к техосмотру».

!!! Запрос сохранить с именем «ТехОсмотр»

Оформить отчет в виде текстового документа (в котором указать номер и тему лабораторной работы, в колонтитулах – ФИО, № группы) одним из способов:

1. воспользоваться **БЛАНКОМ** отчета;

2. в текстовом документе отображать «скриншоты» бланка Конструктора каждого запроса с формулировкой его условия и результат его выполнения. Выражения вычисляемых полей, которые не просматриваются в окне QBE, копировать после картинки бланка Конструктора.

Инструкция к выполнению лабораторной работы № 4

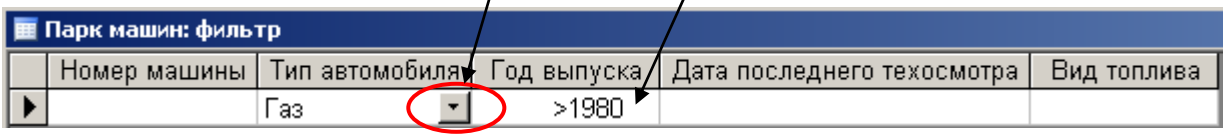
Тема: *Отбор данных в MsA с помощью фильтра и простых запросов на выборку.*

Часть I. Работа с фильтрами в MsA

Использование режима «Изменить фильтр»:

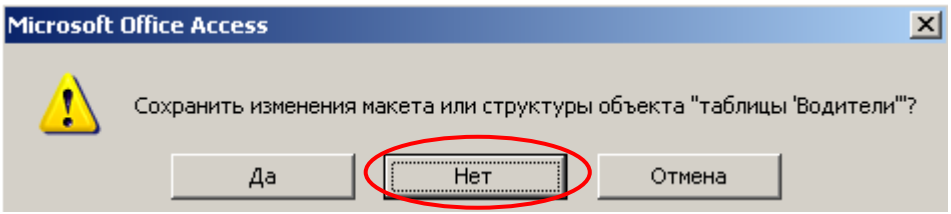
Пример_1. *Выбрать в таблице «Парк машин» автомобили определённого типа (значение задается самостоятельно), год выпуска которых позже* ***N*** *(значение* ***N*** *задается самостоятельно).*

- Открыть таблицу.
- Выбрать команду ***п.м. Записи → Фильтр → Изменить фильтр*** (или соответствующая пиктограмма на панели инструментов).
- Задать критерии отбора записей непосредственно в соответствующих полях либо выбором из ***раскрывающего списка значений***, либо вводя критерий с клавиатуры с ***использованием логических операторов***:



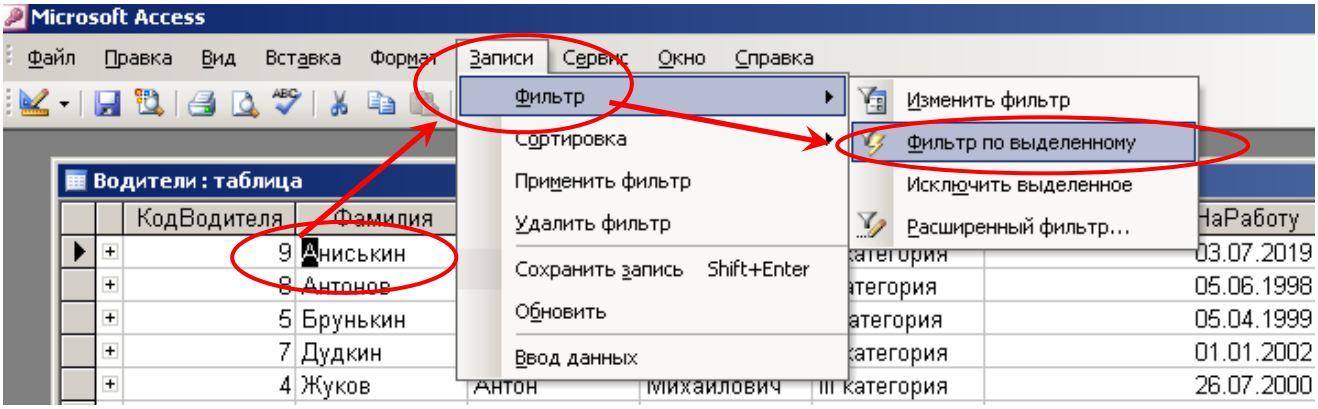
Замечание: в случае неверного задания критериев отбора можно очистить бланк фильтра с помощью пиктограммы на панели инструментов .

- Выполнить фильтр с помощью команды ***п.м. Фильтр → Применить фильтр*** или соответствующая пиктограмма на панели инструментов (*в MsA>2003 команда* ***Фильтр по форме*** *на вкл. «Сортировка и фильтр» на ленте «Главная»*).
- Просмотреть результат выполнения фильтра.
- Отключить фильтр с помощью пиктограммы «Удалить фильтр» на панели инструментов .
- Закрыть таблицу **БЕЗ** сохранения макета или структуры таблицы:



Использование режима «Фильтр по выделенному»:

Пример_2. *Выбрать водителей, чьи фамилии начинаются на букву «А»:*

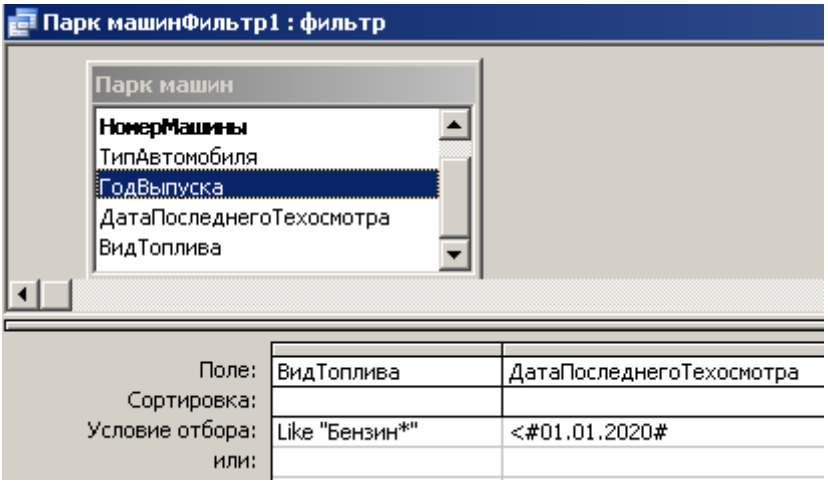


- Открыть таблицу.
- Задать критерии отбора записей, выделив **фрагмент** **значения** в заданном поле (столбце).
- Выбрать команду ***п.м. Записи → Фильтр → Фильтр по выделенному*** или соответствующая пиктограмма на панели инструментов  (*в MsA>2003 команда **Выделение** на вкл. «Сортировка и фильтр» на ленте «Главная»*).
- Просмотреть результат выполнения фильтра.
- Отключить фильтр с помощью пиктограммы «Удалить фильтр» на панели инструментов  .
- Закрыть таблицу **БЕЗ** сохранения макета или структуры таблицы.

Использование режима «Расширенный фильтр»:

Пример_3. *Выбрать автомобили в таблице «Парк машин», вид топлива которых начинается со слова «Бензин»* ***И*** *дата последнего техосмотра которых ранее ДД.ММ.ГГГГ (границу интервала задать самостоятельно). Сохранить фильтр как запрос!!!*

- Открыть таблицу.
- Выбрать команду ***п.м. Записи → Фильтр → Расширенный фильтр***.
- В открывшемся окне фильтра перенести в строку «Поле» нижней части бланка необходимые для задания критериев отбора поля из текущей таблицы (указанной в верхней части бланка) и задать в строке «Условие отбора» критерии отбора записей:

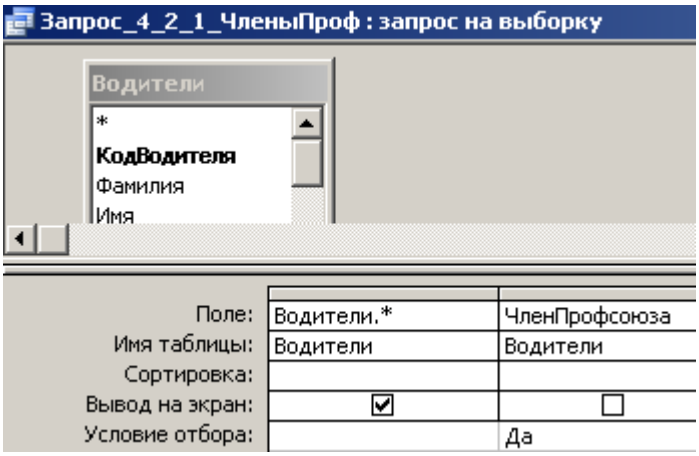


- Выполнить фильтр с помощью команды ***п.м. Фильтр → Применить фильтр*** (или соответствующая пиктограмма на панели инструментов ).
- Просмотреть результат выполнения фильтра.
- Вернуться в бланк фильтра ***п.м. Записи → Фильтр → Расширенный фильтр.***
- Сохранить фильтр как запрос с помощью команды ***п.м. Файл → Сохранить как запрос → ввести имя запроса.***

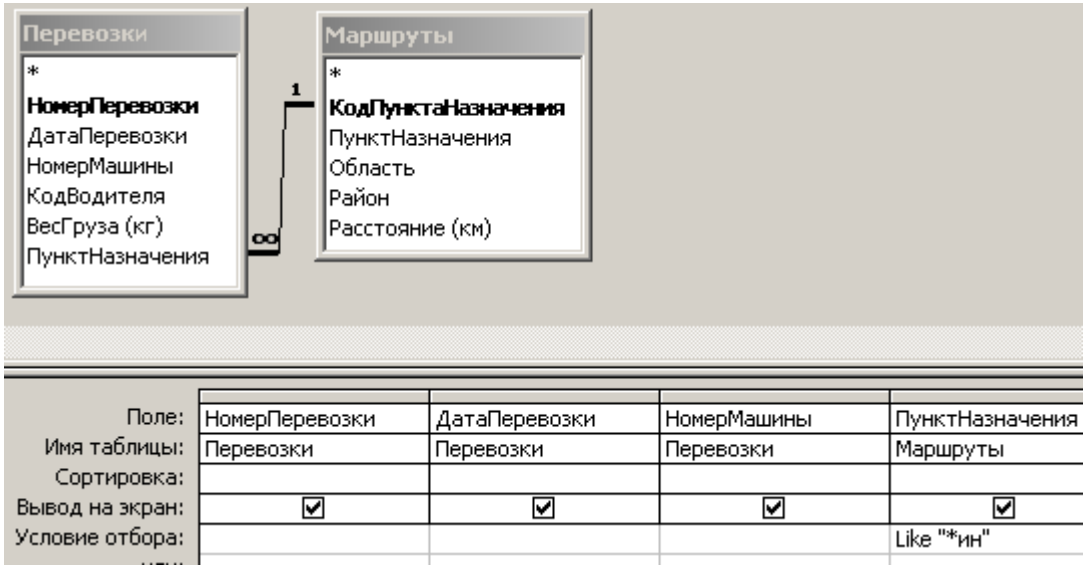
Часть II. Запросы-выборки в MsA

2. Создать следующие **запросы-выборки** из БД «Автоперевозки»:

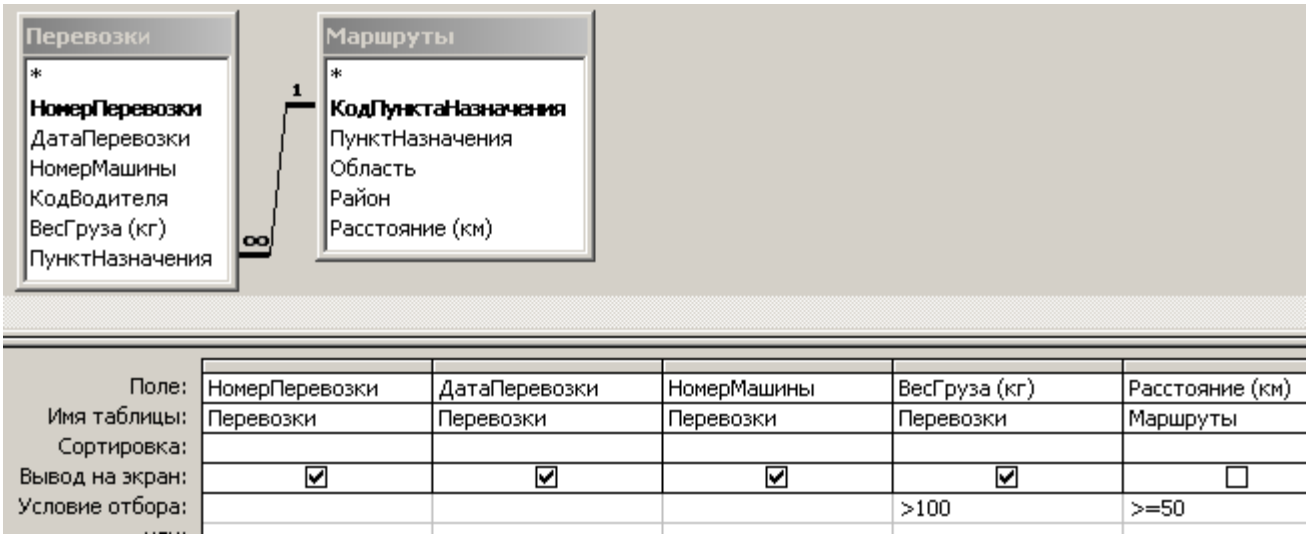
2.1. *запрос по тестовому полю (по таблице «Водители»)*, выводящий водителей, которые являются членами профсоюза:



2.2. *запрос по текстовому полю (по таблицам «Перевозки» и «Маршруты»)*, выводящий записи о перевозках в пункты назначения, название которых заканчиваются на «ин»:

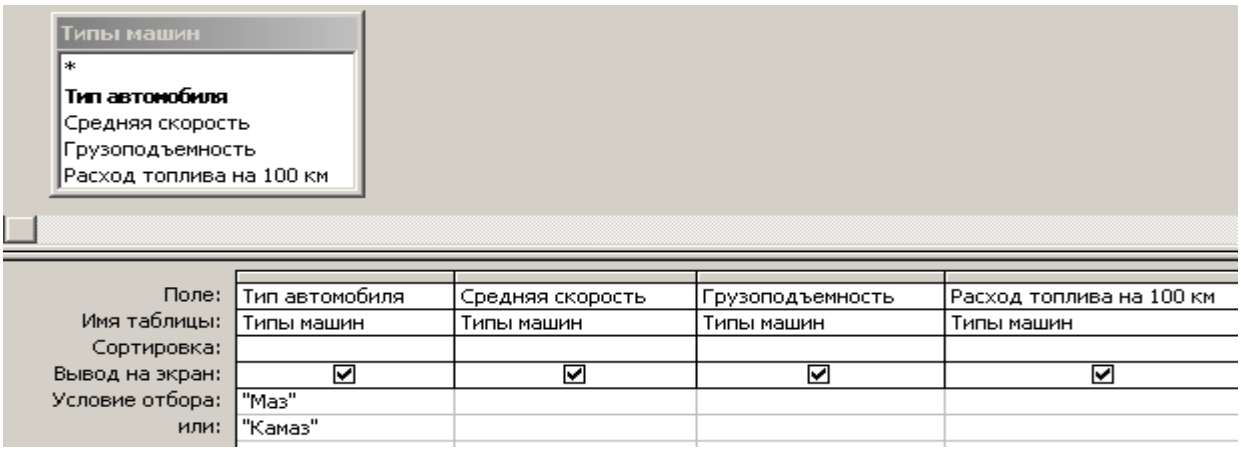


2.3. *запрос по оператору «И» (по таблицам «Перевозки» и «Маршруты»)*, выводящий перевозки, расстояние которых не менее ***N*** км. и вес груза более ***N1*** кг. (границы ***N*** и ***N1*** задать самостоятельно):

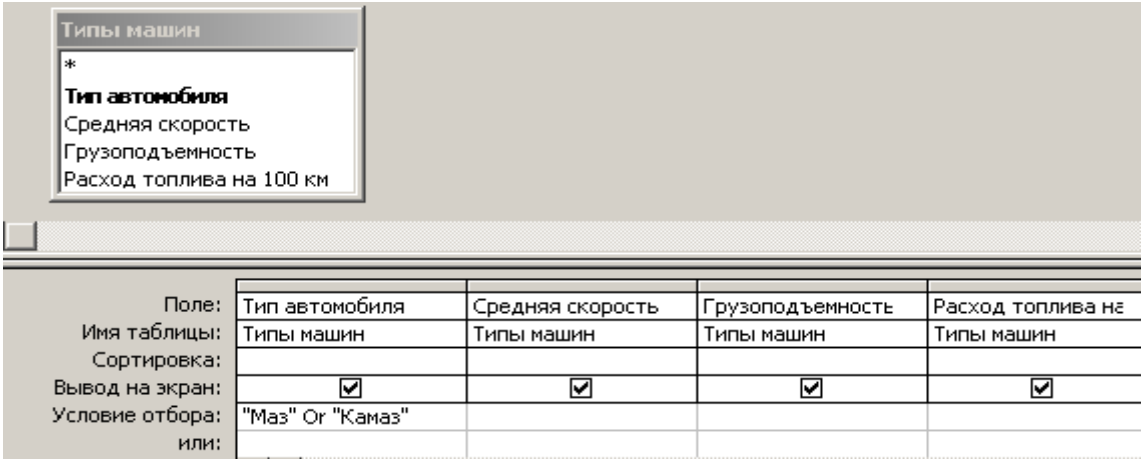


2.4. *<u>запрос по оператору «ИЛИ»</u> (по таблице «Типы машин»)*, выбирающий автомобили, тип которых МАЗ и КАМАЗ:

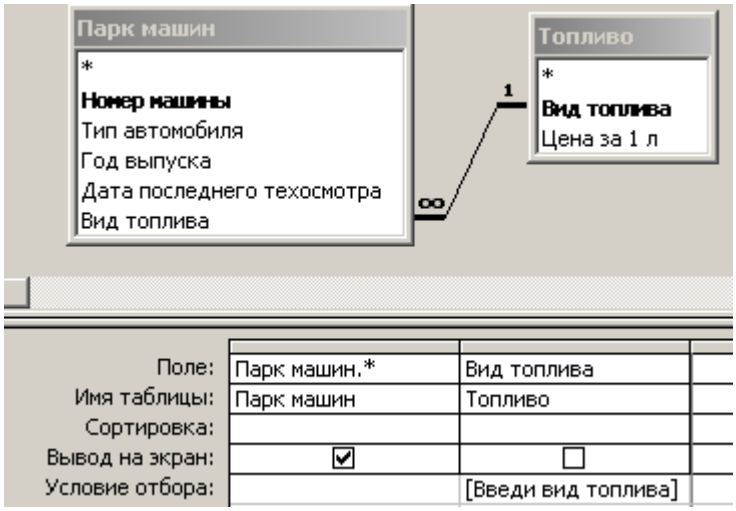
I способ:



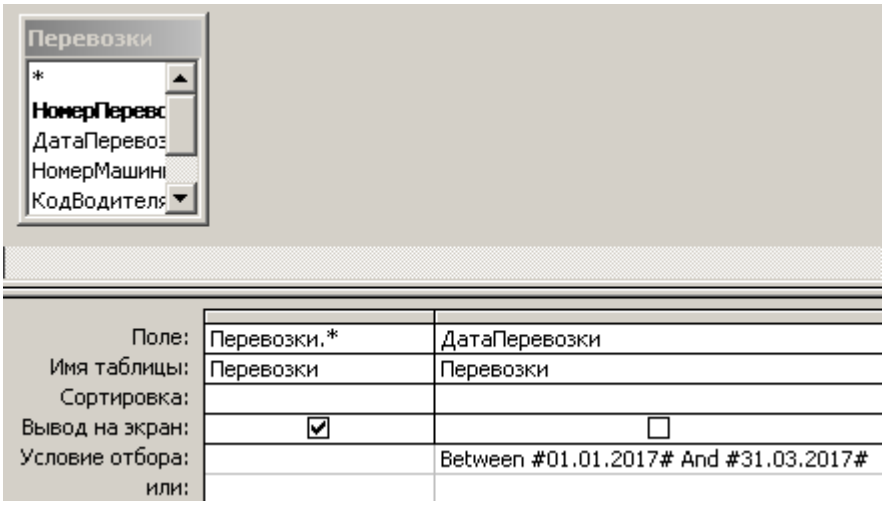
II способ:



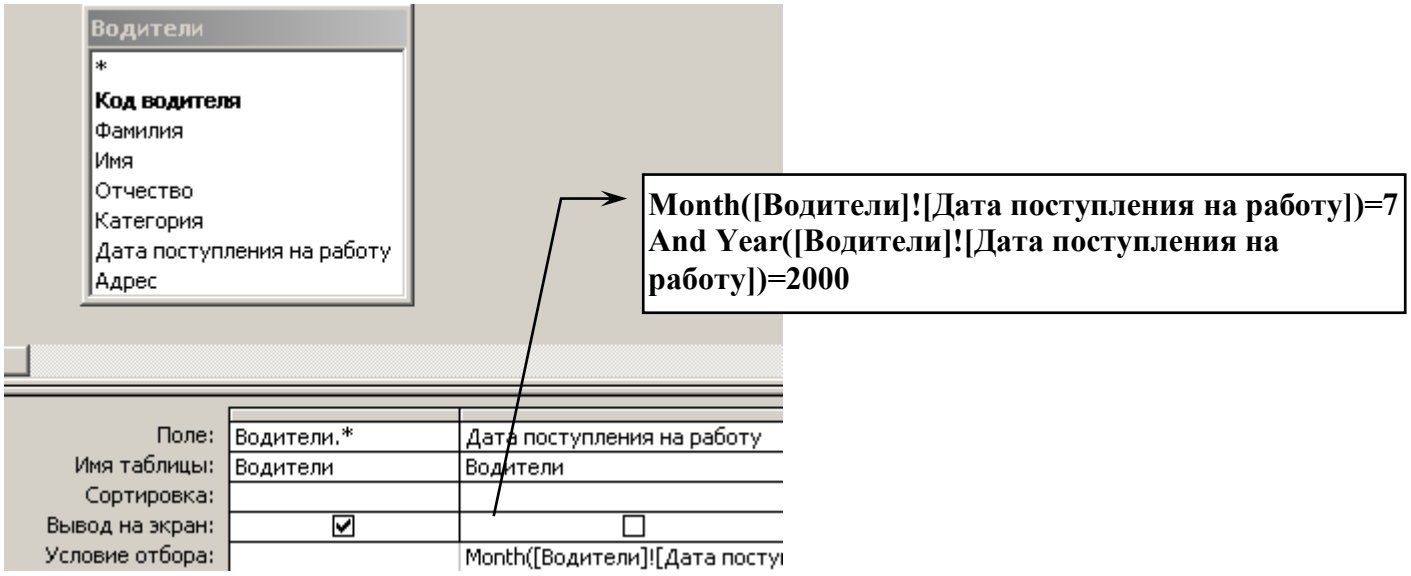
2.5. *<u>параметрический запрос</u> (по таблицам «Парк машин» и «Топливо»)*, выводящий перевозки, в которых машины заправлялись топливом, вид которого вводится с клавиатуры:



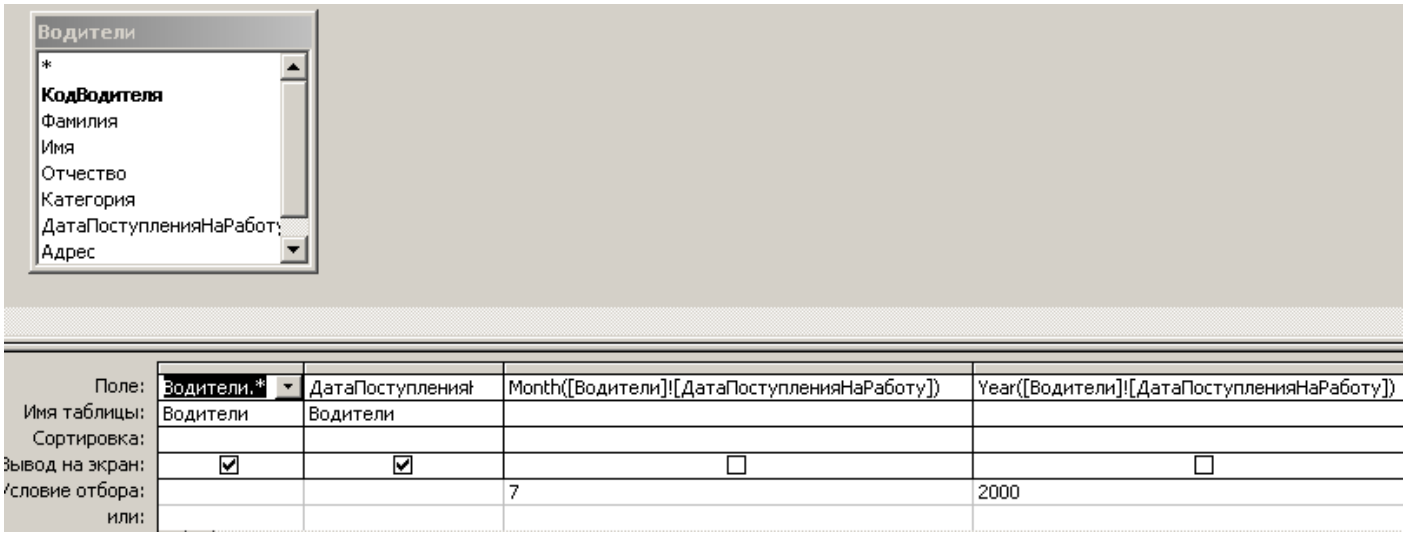
2.6. *<u>запрос по полю типа Дата/время</u> (по таблице «Перевозки»)*, выводящий перевозки, осуществленные в первом квартале 2017 года – использовать операторы сравнения:



2.7. *<u>запрос по полю типа Дата/время</u> (по таблице «Водители»)*, выводящий список водителей, поступивших на работу в ***июле*** месяце ***2000*** года – использовать встроенные функции из категории Дата / Время:

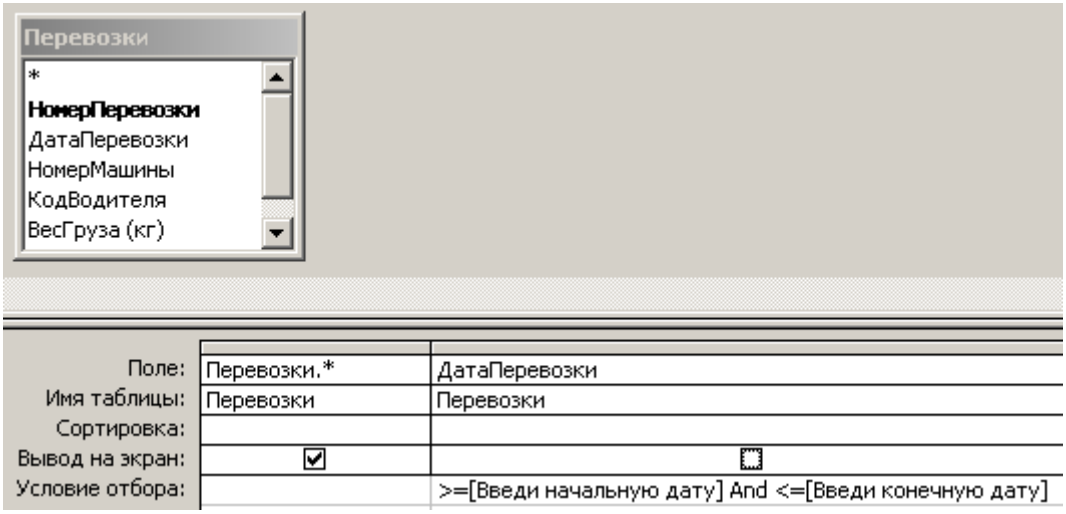


<u>Замечание:</u> После того, как запрос будет сохранен, закрыт и повторно открыт в режиме *Конструктора*, вид его будет другим:

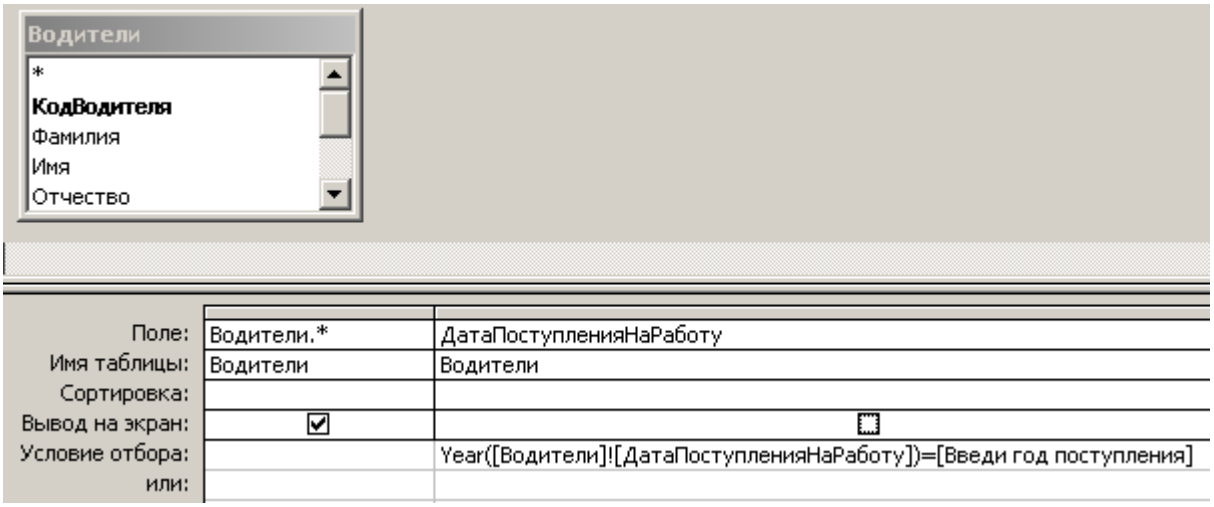


2.8. *<u>параметрический запрос по полю типа Дата/время по временному диапазону (И-запрос)</u> по таблице «Перевозки»,* выводящий перевозки, осуществленные в

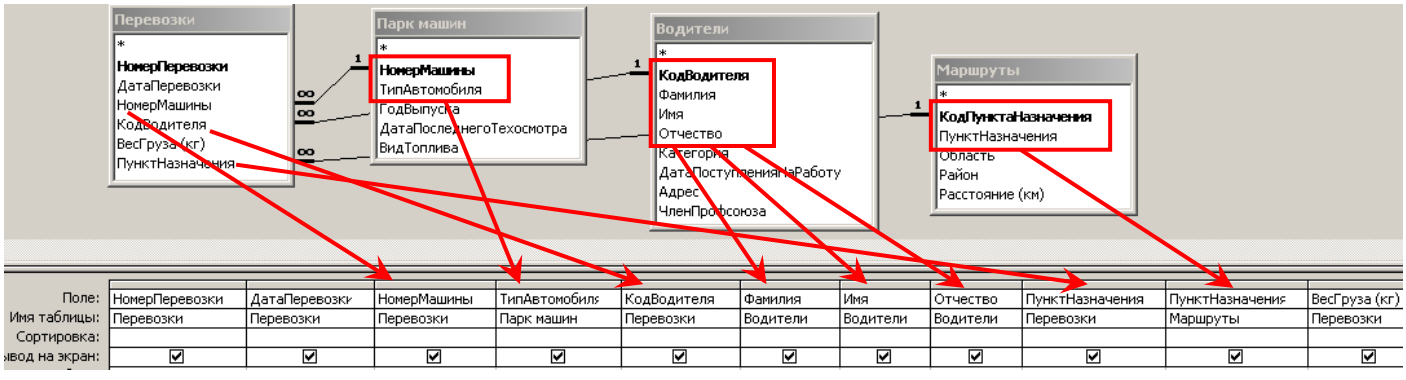
определенном периоде, границы которого вводятся с клавиатуры – использовать проверку условия попадания даты во временной диапазон (*с … по…*) с операторами сравнения;



2.9. *<u>параметрический запрос по полю типа Дата/время</u> по таблице «Водители»*, выводящий список водителей, поступивших на работу в определенный год, значение которого вводится с клавиатуры – использовать встроенные функции из категории Дата/Время;



2.10. *<u>многотабличный запрос</u> по таблицам «Водители», «Парк машин», «Перевозки» и «Маршруты»* включающий поля всех связанных таблиц БД (***расшифровка*** значений всех ***внешних*** ключей).



Функции из категории «Дата/Время»

Date( ) – возвращает текущую системную дату.

Day(дата) – возвращает значение дня месяца от 1 до 31.

Month(дата) – возвращает значение месяца от 1 до 12.

MonthName(Month(дата)) – возвращает **название** месяца с Января по Декабрь.

Year(дата) – возвращает значение года от 100 до 9999.

Weekday(дата) – по умолчанию возвращает целое **число** от 1 (Воскресенье) до 7 (Суббота), соответствующее дню недели.

Weekday(дата;2) – по умолчанию возвращает целое **число** от 1 (Понедельник) до 7 (Воскресенье), соответствующее дню недели.

WeekdayName(Weekday(дата;2)) – возвращает **название** дня недели с Понедельника по Воскресенье.

Hour(дата) – возвращает целое число от 0 до 23, представляющее значение часа в дате.

DatePart("интервал", дата) – возвращает число дней, недель, месяцев и т.д. в соответствии с значением аргумента **интервал** для указанной даты.

"q" – определение квартала (значение от 1 до 4);

"ww" – определение номера недели в году (значение от 1 до 53).

DateAdd("интервал"; число; дата) – возвращает новую дату, равную указанной дате, увеличенной на указанное **число** в соответствии с установленной маской, указанной в аргументе "**интервал**". Возвращает данные типа ***Дата/время***.

Значения маски:

Значение	***Описание***	***Значение***	***Описание***
yyyy	Год	w	День недели
q	Квартал	ww	Неделя
m	Месяц	h	Часы
y	Дней в году	n	Минуты
d	День	s	Секунды

DateDiff("интервал", дата1, дата2) – определяет интервал между двумя датами в указанных единицах измерения (днях, месяцах, кварталах, годах). Здесь значение аргумента "интервал" имеет то же значение, что и в функции DateAdd.

Замечание: “дата1” – начальная дата, “дата2” – конечная дата, причем дата1 $\leq$ дата2.

Функции из категории «Управление»

Функция **Iif(условие; еслиИстина; еслиЛожь)** – возвращает один из двух аргументов в зависимости от результата вычисления выражения. Находится в категории "Управление".

Аргумент	***Назначение***
условие	Выражение, значение которого нужно вычислить
еслиИстина	Значение или выражение, возвращаемые, если значением выражения является "Истина" (1)
еслиЛожь	Значение или выражение, возвращаемые, если значением выражения является "Ложь" (0)

Функции из категории «Текстовые»

Функция **Format("выражение"; инструкция форматирования)** – возвращает строку, содержащую выражение, отформатированное согласно инструкциям форматирования. Находится в категории "Текстовые".

Для выражений даты/времени можно применять следующие символы в инструкции форматирования:

Символ	***Описание***
c	Полный формат даты
dd	День месяца (от 1 до 30)
ddd	Первые две буквы названия дня недели (от **Пн** до **Вс**)
dddd	Полное название дня недели (от **Понедельник** до **Воскресенье**)
w	День недели (от 1 до 7)
ww	Неделя года (от 1 до 53)
mm	Месяц года (от 1 до 12)
mmm	Первые три буквы названия месяца (от **Янв** до **Дек**)
mmmm	Полное название месяца (от **Январь** до **Декабрь**)
q	Квартал года (от 1 до 4)
y	День года (от 1 до 366)
yy	Последние две цифры года (от 01 до 99)
yyyy	Возвращает 4-ёх-значное значение года (от 100 до 9999)

Mid(C, n1, n2) – возвращает **n2** символов из строки **C**, начиная с **n1** символа.

Например: Для извлечения подстроки из строки текста:
если переменная C="холодная зима", то Mid(C,10,4) = "зима"

Left(C, n) – возвращает **n** символов с левого конца строки **C.**

Right (C, n) – возвращает **n** символов с правого конца строки **C.**

Len(C) – возвращает число символов в строке **C.**

Лабораторная работа № 3 (3)

Тема: *Итоговые и перекрестные запросы в MsA.*

Задание:

Часть I. Проектирование итоговых запросов

Создать следующие **итоговые запросы** для учебной БД «Автоперевозки»:

1. Обобщающие запросы для одной группы записей

1.1. Подсчитать количество перевозок за каждый день.

!!! Запрос сохранить с именем «КолПерЗаКаждДень»

1.2. Подсчитать общий пробег (км) для каждой машины.

!!! Запрос сохранить с именем «ПробегМашин»

1.3. Подсчитать по каждому водителю количество ездок, общую стоимость работ, максимальный пробег и минимальное время в пути.

!!! Запрос сохранить с именем «Стат-каПоВодит»

1.4. Определить количество водителей, имеющих одну и ту же категорию и самую раннюю дату поступления на работу в разрезе категорий.

!!! Запрос сохранить с именем «КолВодитПоКатег»

Внимание: далее запросы сохранять с уникальными именами по аналогии с предыдущими пунктами!!!

2. Обобщающие запросы для нескольких групп

2.1. Подсчитать для каждого водителя за каждый день суммарный вес груза, общий пробег и общую стоимость перевозок.

2.2. Подсчитать количество ездок и среднее значение пробега в каждый пункт назначения каждой машиной.

3. Обобщающие запросы по всем записям

3.1. Вывести статистику по полю «Пробег» (максимальное, минимальное, среднее и общее значения).

3.2. Вывести статистику по полю «Стоимость перевозки» (максимальное, минимальное, среднее и общее значения).

4. Группировка с использованием критериев

4.1. Применение критериев отбора для поля, обработанного групповой операцией (ограничение числа сгруппированных записей)

4.1.1. Определить, какое количество машин заправляются видом топлива «Газ».

4.1.2. Подсчитать количество перевозок машинами типа «ВАЗ» в пункты назначения, названия которых начинаются на букву, заданную с клавиатуры.

4.2. применение критериев отбора для поля после его обработки групповой функцией

4.2.1. Вывести ФИО водителей, суммарное время в пути которых превышает N часов (значение N задать самостоятельно).

4.2.2. Вывести в разрезе каждого месяца текущего года количество машин разного типа, которым необходимо пройти техосмотр.

4.2.3. Определить общее количество ездок и общую стоимость перевозок в каждый пункт назначения за каждый квартал определённого года, значение которого задать как параметр.

4.3. применение критериев отбора для поля перед его обработкой групповой операцией

4.3.1. Вывести для каждого водителя максимальное время в пути из всех возможных значений, которые превышают 10 часов.

4.4. применение критериев отбора для поля, не обработанного групповой операцией

4.4.1. Определить количество водителей, поступивших на работу в году, значение которого вводится как параметр.

4.4.2. Определить количество членов профсоюза.

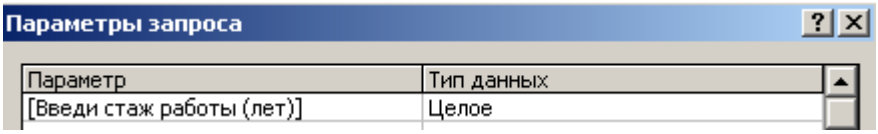
4.4.3. Определить общую сумму уплаченных членами профсоюзов взносов за последние пять лет.

4.4.4. Определить количество водителей высшей категории, стаж которых более **N** лет, где N вводится с клавиатуры как параметр.

Замечание1: Для определения **типа данных** параметра, устанавливаемого в качестве критерия отбора для расчетного поля «Стаж работы (лет)» из вычисляемого запроса «Надбавки», необходимо его (параметр) добавить в окно **«Параметры запроса»**, открываемое следующим образом:

в версии ACCESS 2003: выбрать команду ***Параметры*** из пункта меню ***Запрос***.

в версии ACCESS 2007/2010: выбрать пиктограмму ***Параметры*** на вкладке ленты **Создание** в группе инструментов **Показать или скрыть** (или в группе **Запросы**).



4.4.5. Определить, какое количество машин имеет грузоподъемность не более **N** тонн (значение N задать самостоятельно).

5. Запросы с использованием опции ВЫРАЖЕНИЕ

5.1. Вычислить разницу между максимальным и минимальным значением стоимости работы водителей.

5.2. Вычислить размах сумм премий для разных категорий водителей.

5.3. Вывести статистику по полю «Время в пути» за каждый день определенного (задается произвольно) месяца текущего года.

6. Запросы на основании итоговых запросов

6.1. Вывести информацию о перевозках машинами, чей пробег больше среднего значения по полю «Пробег» в два раза (это запрос-выборка, НЕ итоговый!!!).

6.2. Подсчитать количество перевозок, стоимость которых более самого минимального значения по полю «Стоимость перевозки» на **N** руб. (значение **N** задать самостоятельно).

Часть II. Проектирование перекрестных запросов

Создать следующий перекрестный запрос с помощью Мастера:

П.1. Вывести за каждый день каждой машиной перевезённый общий вес груза.

Создать следующие перекрестные запросы в режиме Конструктора:

1. Простые перекрестные запросы

П.2. Вывести для каждого водителя за каждый день общую стоимость работы.

Замечание2: Для определения **типа данных** параметров, вызываемых из вычисляемого запроса «Надбавки», необходимо их (параметры) добавить в окно **«Параметры запроса»**:

2. Перекрестные запросы с заголовками строк из разных таблиц

П.3. Вывести за каждый день каждым водителем общее число ездок в разные пункты назначения.

3. Перекрестные запросы с итоговым столбцом

П.4. Вывести общий пробег в каждый пункт назначения разными типами автомобилей. В итоговом столбце вывести максимальное время в пути для каждого типа автомобиля.

4. Перекрестные запросы с критериями отбора

П.5. Вывести общую стоимость перевозок разными типами автомобилей водителями, стаж которых от **N1** до **N2** лет (значения **N1** и **N2** задать самостоятельно). В итоговом столбце вывести максимальное число ездок для каждого водителя.

5. Перекрестные запросы с использованием встроенных функций, с критериями отбора и с параметрами

П.6. Вывести общую стоимость перевозок в каждый пункт назначения за каждый день (вывести значение даты и названия дня недели) текущей недели.

П.7. Вывести общее число ездок каждым водителем за каждый квартал определённого года, значение которого вводится как параметр.

П.8. Вывести зарплату каждого водителя (отобразить поле ФИО водителя) за каждый месяц (отобразить название месяца) заданного года. В итоговом столбце вывести максимальное значение затраченного времени в пути каждого водителя.

Оформить отчет в виде текстового документа (в котором указать номер и тему лабораторной работы, в колонтитулах – ФИО, № группы) одним из способов:

1. воспользоваться **БЛАНКОМ** отчета;

2. в текстовом документе отображать «скриншоты» бланка Конструктора каждого запроса с формулировкой его условия.

Выражения вычисляемых полей, которые не просматриваются в окне QBE, копировать после картинки бланка Конструктора.

Копировать также результаты запросов!!!

Лабораторная работа № 3 (4)

Тема: *Активные запросы в MsA.*

Задание:

Создать следующие **активные запросы** для учебной БД «Автоперевозки»:

1. Запрос на создание таблицы

1.1. Создать с помощью активного запроса для таблицы «Водители» копию с названием «Водители_Копия».

!!! Запрос сохранить с именем «Созд_тВодителиКопия»

Внимание: далее запросы сохранять с уникальными именами по аналогии!!!

1.2. Создать с помощью активного запроса для таблицы «Перевозки» копию с названием «Перевозки_Копия».

1.3. Создать с помощью активного запроса для таблицы «Парк машин» копию с названием «ПаркМашинКопия».

1.4. Создать новую таблицу «Перевозки_2020», в которую поместить все записи о перевозках за прошлый год из таблицы «Перевозки».

Замечание: В окне БД (на вкладке «Таблицы») просмотреть ВСЕ созданные таблицы и в режиме *Конструктора* установить в них ключевые поля!!!

2. Запросы на обновление

2.1. С помощью активного запроса в таблице «Водители_Копия» изменить категорию (в сторону повышения) для двух водителей, у которых она одинаковая.

!!! Просмотреть результат после обновления значений полей в таблице «Водители_Копия».

2.2. С помощью активного запроса изменить (увеличить) в таблице «Топливо» цены на все виды топлива на 1%.

!!! Просмотреть результат после обновления значений полей в таблице «Топливо».

2.3. В схеме данных между таблицами «Топливо» и «Парк машин» изменить связь по полю ***Вид топлива,*** добавив к обеспечению *Целостности данных* операцию *Каскадного обновления связанных полей*.

Создать запрос-обновление, в котором выполнить изменение названия вида топлива с «Дизтопливо» на «Экто Diesel». !!! Просмотреть результат после обновления значений полей в двух взаимосвязанных таблицах.

3. Запросы на удаление

3.1. Удалить из таблицы «Перевозки_Копия» все записи о перевозках за прошлый год.

3.2. В схему данных добавить две таблицы «Водители_Копия», «Перевозки_Копия» и установить между ними связь по полю ***Код водителя*** с обеспечением *Целостности данных* и **с установкой** операции *Каскадного удаления связанных записей*.

С помощью активного запроса удалить из таблицы «Водители_Копия» одного из сотрудников (например, в связи с его увольнением), по которому есть связанные

значения в поле внешнего ключа в таблице «Перевозки_Копия». !!! Просмотреть результат после удаления записей из двух взаимосвязанных таблиц.

3.3. В схеме данных добавить две таблицы «ПаркМашинКопия», «Перевозки_Копия» и установить между ними связь по полю ***Номер машины*** с обеспечением *Целостности данных*, но **БЕЗ установки** операции *Каскадного удаления связанных записей*.

С помощью активного запроса из таблицы «Перевозки_Копия» удалить (например, по причине списания) автомобили с годом выпуска = N (где N задать по своему усмотрению, но так, чтобы для машины с таким номером были связанные значения в поле внешнего ключа в таблице «Перевозки_Копия»). !!! Просмотреть результат после удаления записей из подчинённой таблицы.

Затем отдельным активным запросом удалить эти машины из главной таблицы «ПаркМашинКопия». !!! Просмотреть результат после удаления записей из главной таблицы.

4. Запросы на добавление

4.1. Создать структуру таблицы «Перевозки_Архив», состоящую из таких же полей, как в таблице «Перевозки»: в окне БД ***скопировать*** в буфер обмена таблицу «Перевозки», затем вызвать команду ***Вставить*** и в появившемся окне «Вставка таблицы» задать имя новой таблице и выбрать переключатель ⊙ только структура.

С помощью активного запроса добавить в таблицу «Перевозки_Архив» записи о перевозках из таблицы «Перевозки» за текущий месяц текущего года.

Оформить отчет в виде текстового документа (в котором указать номер и тему лабораторной работы, в колонтитулах – ФИО, № группы) следующим образом: в текстовом документе отображать «скриншоты» бланка *Конструктора* каждого запроса с формулировкой его условия.

Выражения вычисляемых полей, которые не просматриваются в окне QBE, копировать после картинки бланка Конструктора.

Лабораторная работа № 4

Тема: Работа с БД в MsA: проектирование форм и отчетов

Задание:

Просмотреть работу форм и формирование отчетов в БД «Борей»:

- запустить утилиту Total Commander;
- в командную строку 

скопировать спецификацию файла БД:

W:\OF2a\PFILES\MSOFFICE\OFFICE\SAMPLES\NWIND.MDB

Часть I. Проектирование форм

1.1. С помощью средства **«Автоформа»** создать ***ФОРМУ*** для таблицы *«Перевозки»* и дополнить в режиме формы таблицу двумя записями.

1.2. С помощью **Мастера** форм создать для таблицы «Типы машин» ***Табличную*** форму, а для таблицы «Парк машин» – форму ***В один столбец***.

1.3. С помощью **Мастера** создать ***СОСТАВНУЮ ФОРМУ*** по таблицам *«Водители»* и *«Перевозки»*, в которой отображать для каждого водителя все осуществленные им перевозки. С помощью созданной формы добавить в штат одного сотрудника и две выполненные им перевозки за произвольные даты.

1.4. Для запроса "Стоимость перевозок" из л.р. № 6 в режиме **Конструктора** создать форму для просмотра данных. Используя Панель элементов управления (ПЭУ) включить в форму следующие элементы:

В Заголовке:

- ✓ Надпись (название формы).
- ✓ Текущую дату (как вычисляемое поле).

Для вставленных ЭУ изменить формат, шрифт, тип выравнивания и размер данных, а также изменить цвет, границы и внешнее оформление.

Напоминание:

В области данных:

- ✓ Отобразить все поля указанной таблицы на фоне ЭУ «Прямоугольник», или рисунка (для выравнивания выделенной группы ЭУ: ***п.м. Формат → Выровнять***).
- ✓ Добавить вычисляемое поле «День недели перевозки», в котором вывести название дня недели для указанного поля.
- ✓ Для ЭУ **«Поле»** изменить шрифт и высоту букв, а затем установить размеры самих полей по размеру данных (***п.м. Формат → Размер***).

Замечание2: для выделения одновременно нескольких ЭУ (например, только ЭУ «Поле» без их подписей) используется клавиша **SHIFT**.

- ✓ Для ЭУ **«Подпись поля»** убрать границы и изменить элементы форматирования шрифта.

В области примечаний:

- ✓ Добавить вычисляемое поле для вывода общей стоимости по всем перевозкам.
- ✓ Добавить надпись с собственными ФИО.

Замечание3: Для организации формы для просмотра установить значение «НЕТ» для следующих свойств формы: *«Ввод данных», «Разрешить изменения», «Разрешить удаления», «Разрешить обновления».*

Часть II. Проектирование отчетов

2.1. Для таблицы «Топливо» создать **Автоотчет (***Автоотчет: в столбец***).**

2.2. С помощью **Мастера** создать ***ОТЧЕТ (табличный)*** по таблице *«Перевозки»*, предусмотрев группировку записей по полю *Номер машины*. Для каждой группировки вычислить общий вес груза и максимальное расстояние.

!!! Перейти в режим **Конструктора** и изучить (и при необходимости ***отредактировать***) структуру и расположение элементов управления в разделах отчета.

Замечание3: Для отчетов с представлением данных в табличной форме с большим количеством отображаемых полей в области данных рекомендуется настраивать вид ориентации – **АЛЬБОМНЫЙ**.

2.3. Для запроса «Зарплата водителей» создать **Автоотчет (***Автоотчет: ленточный***).** В режиме **Конструктора** выполнить следующие изменения в структуре отчета:

- ✓ добавить уровни группировки записей по полю **ФИО водителя**;
- ✓ для каждой выделенной группы подсчитать итоговые (суммарные) значения всех вычисляемых в запросе полей;
- ✓ добавить в ***Примечание*** отчета надпись с ФИО исполнителя (собственное ФИО);
- ✓ изменить элементы форматирования (гарнитура и высота шрифта, начертание, цвет, обрамление и т.д.) для ЭУ «Поле» и «Подпись поля» по выбору;
- ✓ оформить нумерацию страниц.

2.4.**** В режиме **Конструктора** создать отчет в произвольной форме, выводящий список машин, дата следующего техосмотра которых попадает в заданный с клавиатуры месяц определенного года. В отчет вывести все характеристики автомобиля, включая даты последнего и предстоящего техосмотра.

Замечание4: Для выполнения задания необходимо предварительно создать ***параметрический*** запрос, который будет являться источником данных для отчета.

**** – Дополнительное задание (необязательное).

Продемонстрировать работу преподавателю.

Лабораторная работа № 5 (1)

Тема: **Автоматизация работы с БД в MS ACCESS: Макросы. Управление БД.**

Задание:

В режиме *Конструктора* создать ***ГЛАВНУЮ КНОПОЧНУЮ ФОРМУ (ГКФ)*** для управления работой в БД «Автоперевозки»:

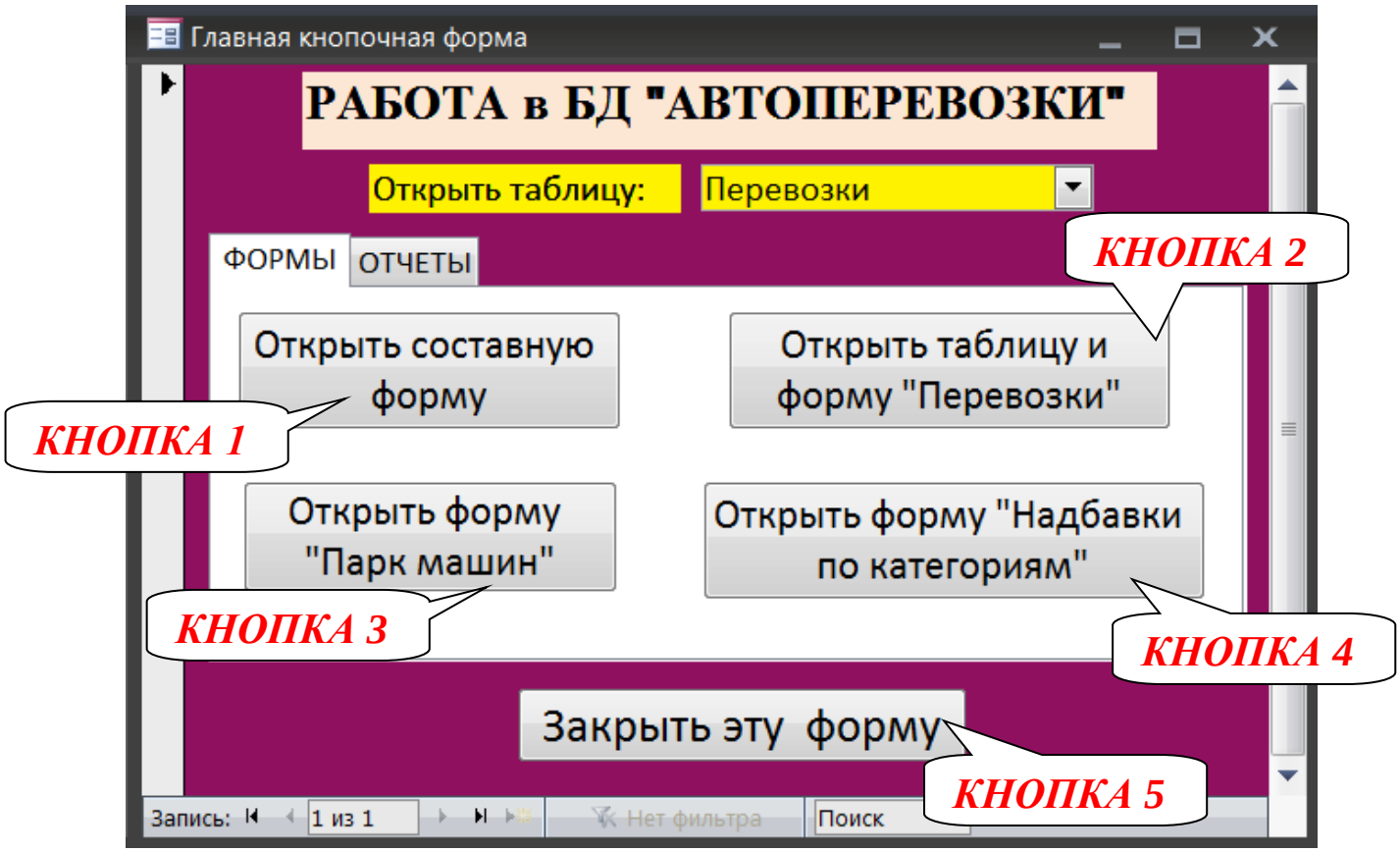
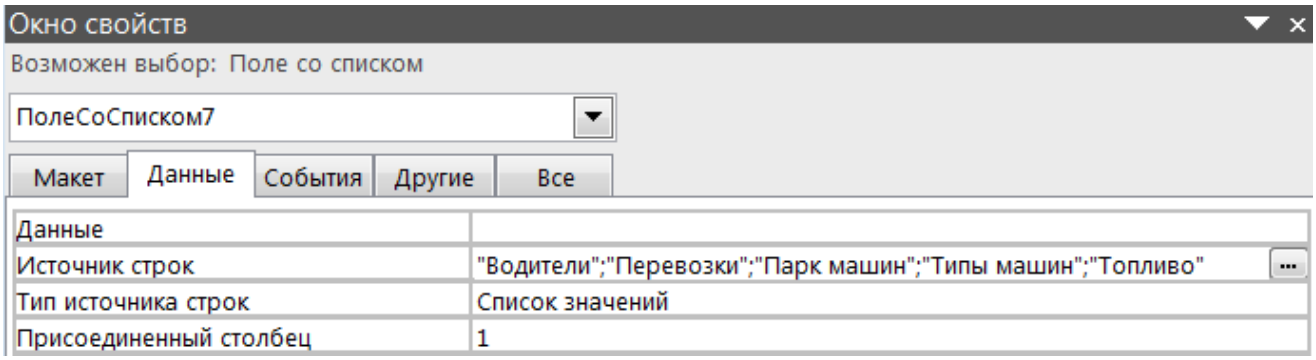


Рисунок 1 – ***Главная кнопочная форма***

!!! Сохранить форму с названием: «Главная кнопочная форма»

В области данных ГКФ (*для которой изменить цвет заливки*) разместить следующие элементы управления (ЭУ), применив к ним различные возможности форматирования:

1. ЭУ «**НАДПИСЬ**» для вывода названия ГКФ: ***Работа в БД «Автоперевозки»***.
2. ЭУ «**ПОЛЕ СО СПИСКОМ**» для выбора одной из таблиц БД для её открытия.



Для работы данного ЭУ создать ***макрос с УСЛОВИЕМ*** (*связанного с событиями ЭУ в форме*) с названием «ОткрТаблиц_ГКФ», в котором прописать макрокоманды открытия таблиц, названия которых фигурируют в качестве источника строк ЭУ «Поле со списком»:

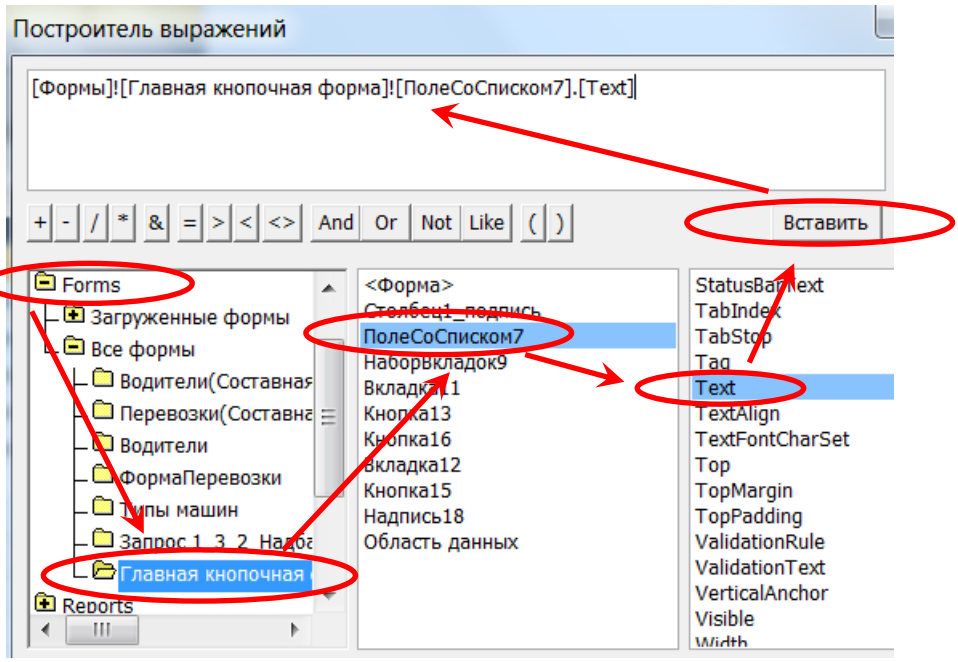
ОткрТаблиц_ГКФ

Условие	Макрокоманда
[Формы]![Главная кнопочная форма]![ПолеСоСписком7].[Text]="Водители"	ОткрытьТаблицу
[Формы]![Главная кнопочная форма]![ПолеСоСписком7].[Text]="Перевозки"	ОткрытьТаблицу
[Формы]![Главная кнопочная форма]![ПолеСоСписком7].[Text]="Парк машин"	ОткрытьТаблицу
[Формы]![Главная кнопочная форма]![ПолеСоСписком7].[Text]="Типы машин"	ОткрытьТаблицу
[Формы]![Главная кнопочная форма]![ПолеСоСписком7].[Text]="Топливо"	ОткрытьТаблицу

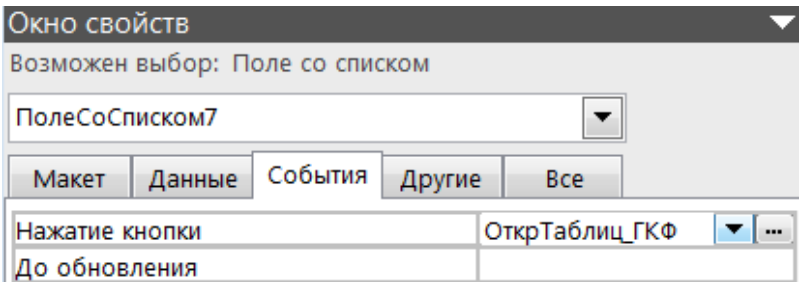
Аргументы макрокоманды

Имя таблицы	Водители
Режим	Таблица
Режим данных	Изменение

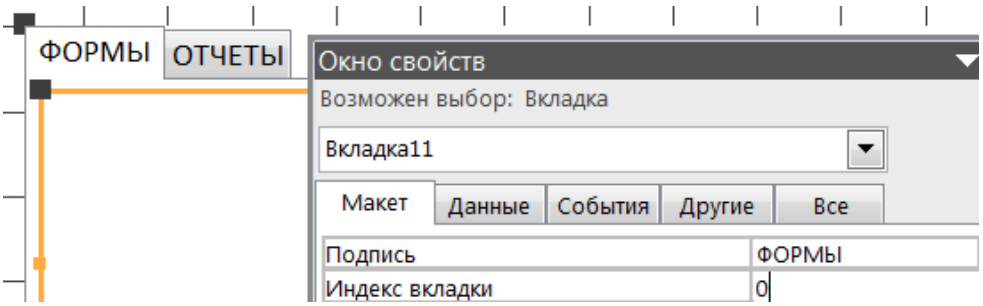
Для формирования условий макроса необходимо пользоваться ***Построителем выражений***:



Для связи макроса с событием ЭУ «Поле со списком» необходимо вызвать окно **Свойств** для ЭУ и выбрать на вкладке **События** нужное имя макроса в свойстве «Нажатие кнопки»:



3. ЭУ «**ВКЛАДКА**» для размещения кнопок, открывающих часто используемые формы и отчеты.



Сначала создать следующие макросы:

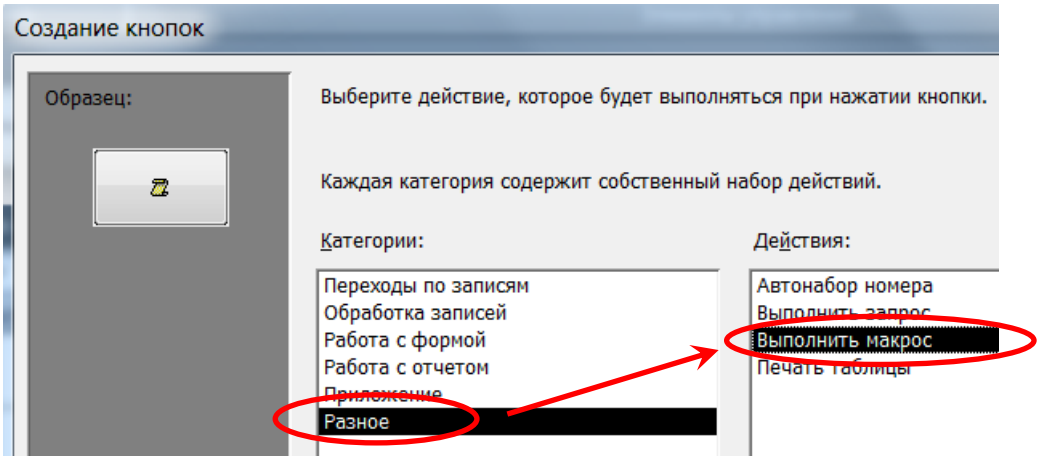
Простой макрос (с макрокомандой выполнения) для открытия ***составной формы*** (категория действия – *Работа с формой*), созданной в п. 1.3. лабораторной работы № 4.

На вкладке **«Формы»** добавить следующие ЭУ «**КНОПКИ»** при ***активизированной кнопке «Мастера»***  :

3.1. Кнопку (кнопка_1 на рис. 1) для открытия ***составной формы*** (категория действия – *Работа с формой*), созданной в п. 1.3. лабораторной работы № 4.

3.2. Кнопку (кнопка_2 на рис. 1) для запуска ***макроса линейной структуры*** (*не связанного с событиями*), позволяющего отобразить на экране одновременно таблицу «Перевозки» для внесения изменений и форму, созданную на ее основе в п. 1.1. лабораторной работы № 4 для просмотра этих изменений.

Действия на первом шаге при создании таких кнопок, когда активизирована кнопка «Мастера»  на Панели ЭУ:



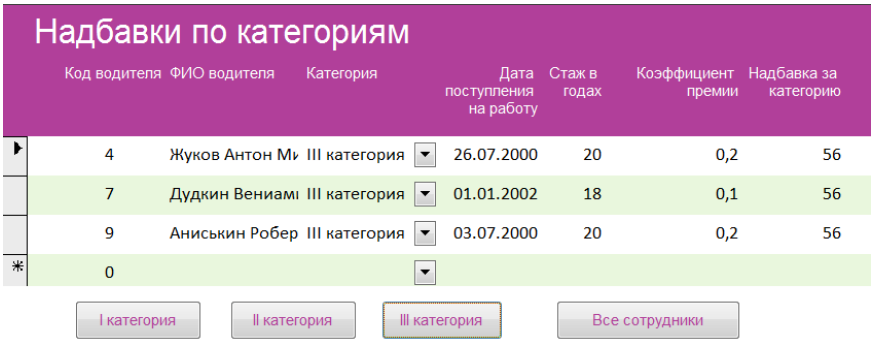
3.3. Кнопку (кнопка_3 на рис. 1) для запуска ***макроса с УСЛОВИЕМ*** (*не связанного с событиями*), позволяющего открывать форму (в один столбец), созданную по таблице «Парк машин» п. 1.2. лаб_раб № 4, отображающую **только** машины с видом топлива «Газ» и годом выпуска позже **N** года (значение **N** задать самостоятельно), а при отсутствии таковых записей должно выводится соответствующее сообщение.

3.4. Кнопку (кнопка_4 на рис. 1) для открытия формы «Надбавки», которую нарисовать в режиме отключенной кнопки «Мастера» на Панели ЭУ. Далее выполнить следующее:

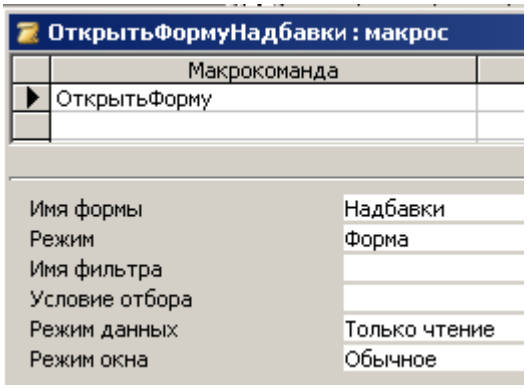
3.4.1. с помощью мастера форм создать ЛЕНТОЧНУЮ форму для запроса «5_2_Надбавки» (из лаб_раб. № 3);

3.4.2.создать ***групповой макрос,*** *например, с названием «ВыборКатегории»,* (*связанный с событиями ЭУ в форме*), отдельные макросы которого позволяют закреплять действия за кнопками формы «Надбавки» – просматривать сведения о надбавках водителей разных категорий или всех одновременно;

3.4.3. добавить четыре кнопки на форму «Надбавки» в режиме активизированной кнопки «Мастера»  на Панели ЭУ с привязкой каждого отдельного макроса из группового, созданного в пункте задания выше:



3.4.4. создать макрос для открытия формы «Надбавки» с названием «ОткрытьФормуНадбавки»:



3.4.5. привязать макрос «ОткрытьФормуНадбавки» в событие «Нажатие кнопки» для кнопки_4 в ГКФ.

На вкладке **«Отчеты»** добавить следующие ЭУ «**КНОПКИ»** при ***активизированной кнопке «Мастера»***:

3.5. Кнопку для открытия ***отчета «Зарплата водителей»***, созданного в п. 2.3. лабораторной работы № 4 (категория действия – *Работа с отчетом*).

3.6.**** Кнопку для запуска макроса (*связанного с событиями отчета)*, который открывает ***отчет*** по таблице ***«Перевозки»*** (созданный в п. 2.2. лаб_работы № 4) и, в случае его предварительного просмотра перед печатью и перехода к следующей странице, выводит сообщение, содержащее номер этой страницы.

4. ЭУ «**КНОПКА**» для закрытия главной кнопочной формы (действие «Закрыть форму» в категории *Работа с формой*).

**** – Дополнительное задание (необязательное).

Продемонстрировать работу преподавателю.

Лабораторная работа № 5 (2)

Тема: **Автоматизация работы с БД в MS ACCESS: Модули VBA.**

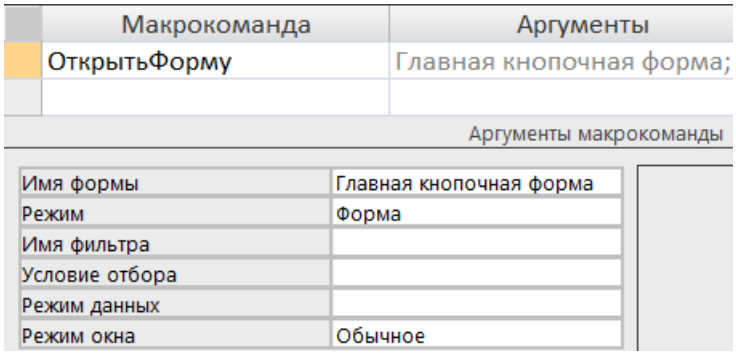
Задание:

1. Просмотреть модули форм и отчетов в БД «Борей». Изучить стандартный модуль «Заставка» и проанализировать работу функций OpenStartup, HideStartupForm, CloseForm.

2. Создать макрос, который позволяет выводить на экран ГЛАВНУЮ КНОПОЧНУЮ ФОРМУ (созданную в лаб_работе № 5(1)) при открытии БД «Автоперевозки».

Порядок выполнения:

2.1. В режиме *Конструктора* создать макрос:



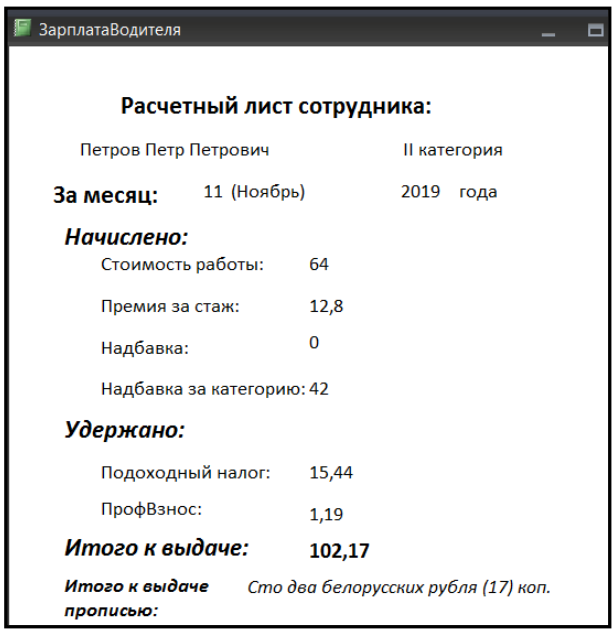
2.2. Нажать кнопку **Сохранить** и в диалоговом окне «Сохранение» ввести название макроса: **AutoExec**.

2.3. Проверить работу макроса: закрыть БД и открыть её снова.

3. Создать ***стандартный модуль*** VBA с названием «MyProcedure», в котором создать процедуру-функцию «PremiaStag» с использованием многострочного условного оператора **If…Then** для определения процента премии за стаж по условию вычисляемого поля «%премии за стаж» в запросе «Надбавки» лаб_работы № 3(2).

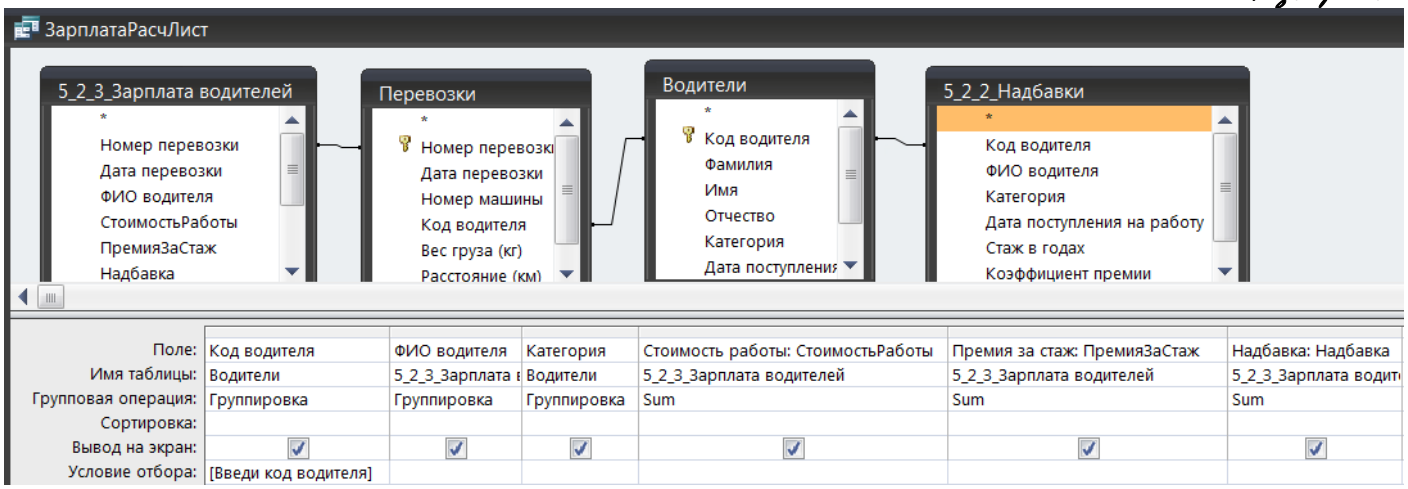
Дополнить запрос «Надбавки» новым вычисляемым полем с использованием пользовательской функции «PremiaStag».Сравнить результаты вычислений !!!

4. Создать **ОТЧЕТ** для выдачи расчетного листка отдельному водителю за заданный месяц определенного года:



Порядок выполнения:

4.1. В режиме *Конструктора* создать итоговый параметрический запрос «ЗарплатаРасчЛист», выводящий для конкретного водителя за указанный месяц заданного года суммарные значения всех рассчитанных в запросе ЗарплатаВодителей (из лаб_работы № 3(2)) полей и поля «Надбавка за категорию», рассчитанного в запросе Надбавки:



Поле:	Код водителя	ФИО водителя	Категория	Стоимость работы: СтоимостьРаботы	Премия за стаж: ПремияЗаСтаж	Надбавка: Надбавка
Имя таблицы:	Водители	5_2_3_Зарплата в	Водители	5_2_3_Зарплата водителей	5_2_3_Зарплата водителей	5_2_3_Зарплата водит
Групповая операция:	Группировка	Группировка	Группировка	Sum	Sum	Sum
Сортировка:						
Вывод на экран:	☑	☑	☑	☑	☑	☑
Условие отбора:	[Введи код водителя]					

продолжение запроса:

Поле:	Надбавка за категорию: Надбавка за категорию	Итого начислено: Начислено	Подоходный налог: ПодоходныйНалог	ПрофВзнос: ПрофВзнос
Имя таблицы:	5_2_2_Надбавки	5_2_3_Зарплата водителей	5_2_3_Зарплата водителей	5_2_3_Зарплата водителей
Групповая операция:	Sum	Sum	Sum	Sum
Сортировка:				
Вывод на экран:	☑	☑	☑	☑
Условие отбора:				

продолжение запроса:

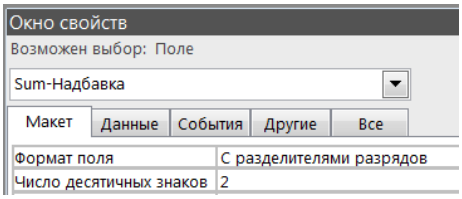
Поле:	ПрофВзнос: ПрофВзнос	Итого к выдаче: [5_2_3_Зарплата водителей]![Зарплата водителя]	Месяц: Month([Перевозки]![Дата перевозки])	Год: Year([Перевозки]![Дата перевозки])
Имя таблицы:	5_2_3_Зарплата водител			
Групповая операция:	Sum	Sum	Группировка	Группировка
Сортировка:				
Вывод на экран:	☑	☑	☑	☑
Условие отбора:			[Введи НОМЕР месяца]	[Введи год]

4.2. Создать ***<u>стандартный модуль</u>*** VBA с названием «Пропись», в который добавить код процедуры-функции propis( ) для вывода в отчете суммы к выдаче прописью (код функции см. в отдельном прикрепленном одноименном файле).

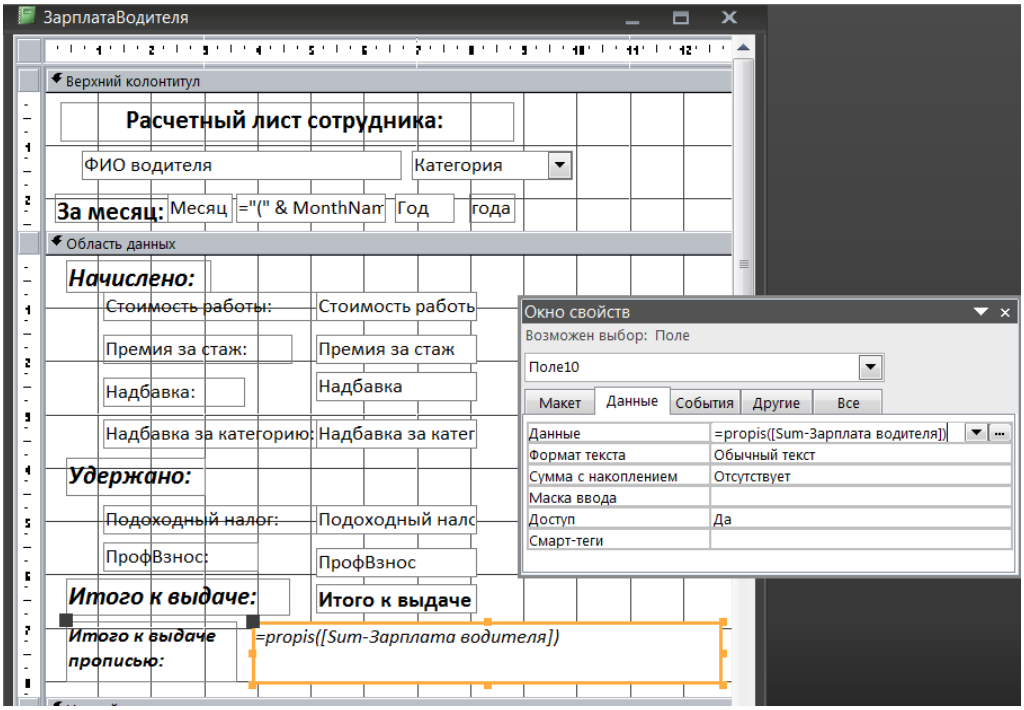
Использование встроенных функций языка VBA см. в **конспекте лекций**.

4.3. Создать **отчет** в режиме *Конструктора*, для источника записей которого взять запрос «ЗарплатаРасчЛист», сформированный в п. 4.1. задания.

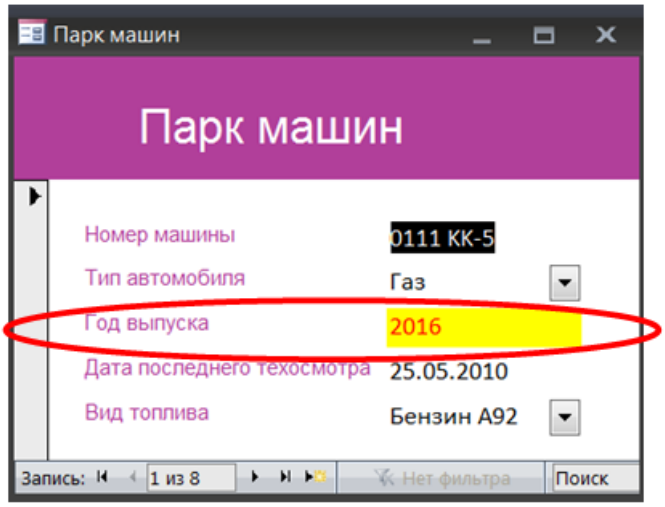
!!! Обязательно изменить свойства всех расчетных итоговых полей:



Использовать созданную пользовательскую функцию прописи числа в свойстве «Данные» нового свободного ЭУ «Поле», добавленного в нижнюю часть области данных отчета (***замечание:*** *аргумент функции propis( ) – поле текущего отчета!!!*).

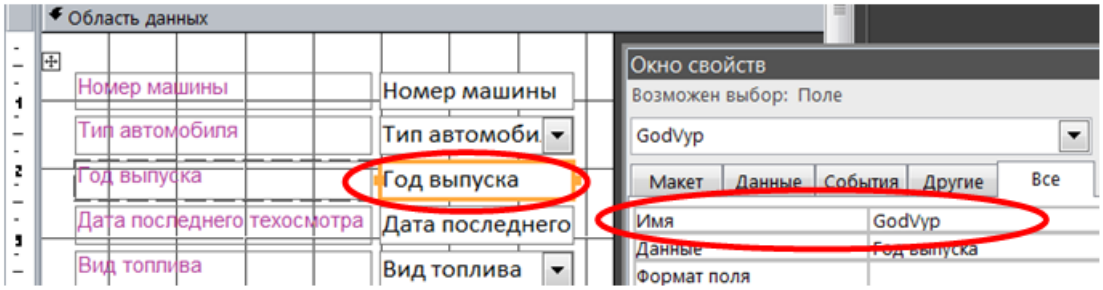


5. Создать ***<u>модуль для формы</u>*** «Парк машин», сформированной в п.1.2. лаб_работы № 4 с использованием оператора выбора **<u>Select Case</u>** для изменения свойств (цвет текста и цвет заливки) поля «Год выпуска» при открытии формы и переходе на новую запись:

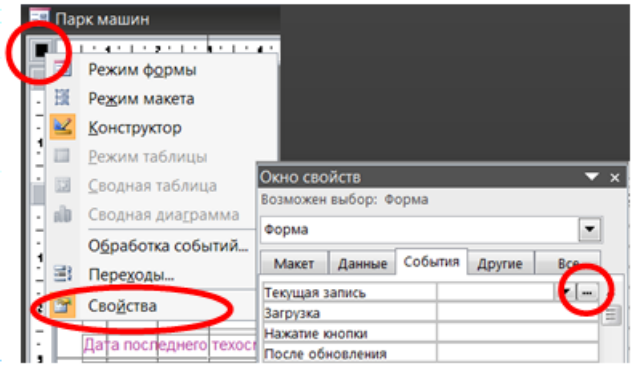


Порядок выполнения:

5.1. В режиме *Конструктора* формы «Парк машин» изменить свойство **<u>Имя</u>** (GodVyp) для ЭУ Поле «Год выпуска»:



5.2. Открыть окно свойств самой формы и в свойстве *«Текущая запись»* на вкладке **События** вызвать ***Построитель*** (кнопка с многоточием) и выбрать опцию **Программы** из окна «Построить».



Автоматически откроется редактор VB и будет создан модуль формы «Form_Парк машин», в котором появится ключевые слова начала и конца процедуры – обработчика события (обработки текущей записи – Current):

```
Private Sub Form_Current()

End Sub
```

5.3. Внутри ключевых слов задать тело процедуры:

```
Private Sub Form_Current()
Dim curGodVyp As Integer, lngBlack As Long
Dim lngRed As Long, lngYellow As Long, lngWhite As Long

   If Not IsNull(Me!GodVyp.Value) Then
      curGodVyp = Me!GodVyp.Value
   Else
      Exit Sub
   End If

   lngRed = RGB(255, 0, 0)
   lngBlack = RGB(0, 0, 0)
   lngYellow = RGB(255, 255, 0)
   lngWhite = RGB(255, 255, 255)

   Select Case curGodVyp
   Case Is > 2015
      Me!GodVyp.ForeColor = lngRed
      Me!GodVyp.BackColor = lngYellow
   Case Else
      Me!GodVyp.ForeColor = lngBlack
      Me!GodVyp.BackColor = lngWhite
   End Select
```

5.3. Добавить в текущую процедуру в конструкции оператора **Select Case** блок с проверкой попадания года выпуска машины в интервал от N1 до N2 (границы интервала задать самостоятельно), при этом добавить новые цвета заливки фона и шрифта с помощью функции RGB( ). Выполнить проверку работы процедуры в Режиме формы.

6. Аналогично с предыдущим пунктом задания создать ***Модуль формы*** для **Главной кнопочной формы** с обработчиком события при её **ЗАГРУЗКЕ** – выводом окна сообщения:

```
Private Sub Form_Load()
MsgBox "Сейчас открылась Главная кнопочная форма", 48 + vbOKOnly, "Окошко приветствия"
End Sub
```

Оформить отчет, в котором вывести коды всех созданных процедур (п. № 3 и п. № 5.3. задания). Описать аргументы функции ***MsgBox*** *в п. 6 задания.*

Продемонстрировать работу преподавателю.

Лабораторная работа № 6

Тема: **Проектирование реляционной базы данных.**

Задание:

1. Составить план разработки проекта реляционной базы данных (РБД) для заданной предметной области. Базу данных следует рассматривать как часть будущей информационной системы «Реализация товаров в магазине “XXX”», предназначенной для учета продаж товаров по ВАРИАНТУ (выбирается по двум последним цифрам логического имени):

№ вар.	*Тема:*	*№ вар.*	*Тема:*
1	*Магазин обуви*	*11*	*Магазин автомобильных запчастей*
2	*Магазин оргтехники*	*12*	*Магазин продуктов питания*
3	*Магазин «Садовод»*	*13*	*Книжный магазин*
4	*Магазин антикварных товаров*	*14*	*Магазин ювелирных изделий*
5	*Магазин мебели ручной работы*	*15*	*Магазин канцелярских товаров*
6	*Магазин музыкальных инструментов*	*16*	*Магазин детской одежды*
7	*Магазин детских товаров*	*17*	*Магазин косметики и парфюмерии*
8	*Магазин спортивной одежды*	*18*	*Магазин бытовой химии*
9	*Магазин товаров бытовой техники*	*19*	*Магазин часов*
10	*Магазин женской одежды*	*20*	*Магазин офисной мебели*

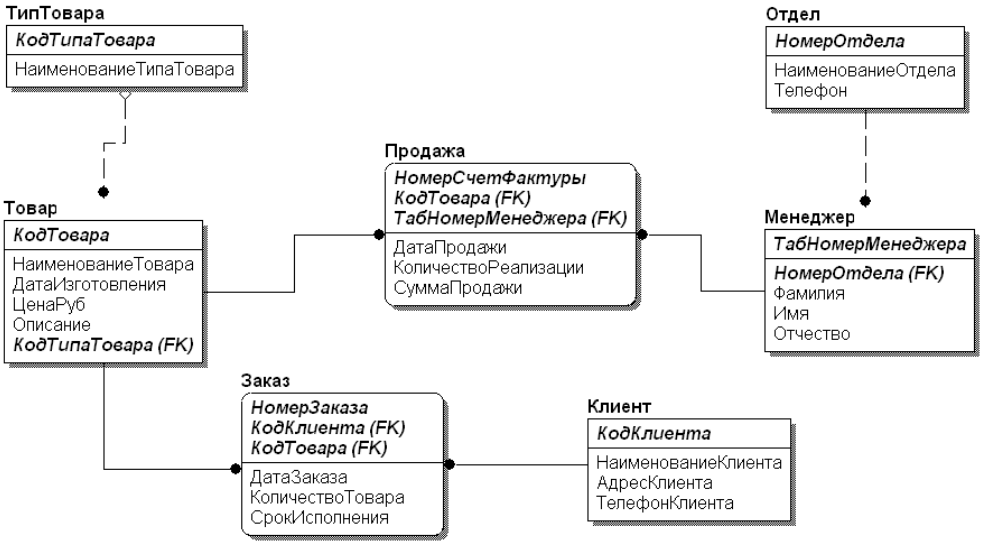
2. Выполнить анализ заданной предметной области. Сформулировать словесное описание информационных объектов.

3. Разработать концептуальную модель БД, описывающую предметную область в рамках ER-модели «сущность – связь». Для описания объектов предметной области определить не менее 7-ми сущностей с разными типами отношений между ними в рамках методологии IDEF1X.

4. Выполнить логическое проектирование реляционной БД в среде CA ERwin Process Modeler (с логико-физическим типом модели). Реализовать принудительное преобразование **одной** из связей «многие-ко-многим» *(т.е. создание новой - связывающей - таблицы и двух новых связей "один-ко-многим")* с помощью встроенного в ErWin инструмента (пошагового мастера) ***Many-To-Many Transform Wizard***.

Порядок выполнения работы

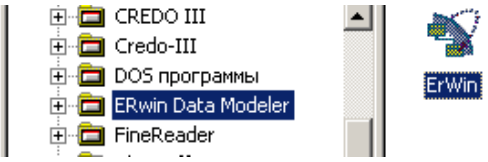
на примере АИС «Реализация средств вычислительной техники»:



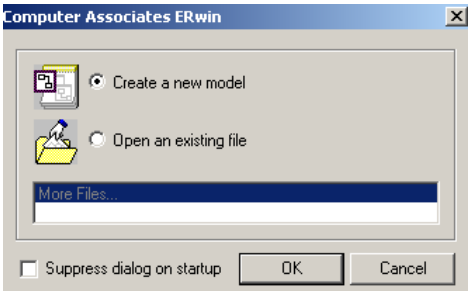
1. Ознакомиться с краткими сведениями о пакете ErWin по ССЫЛКЕ.

2. Запустить программу ErWin:

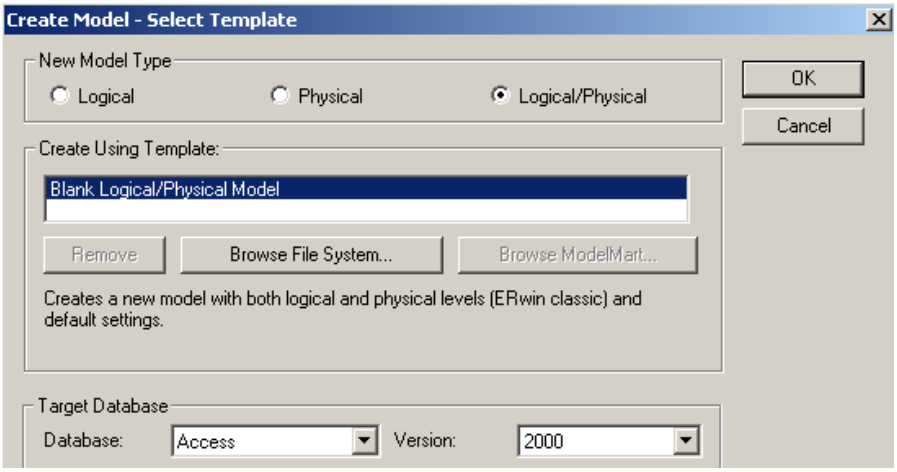
Сетевые приложения →



3. Создать новую модель данных:



Создание новой модели начинается с выбора типа модели: логическая, физическая, логико-физическая. Выбор типа новой модели осуществляется в диалоге **Create Model**. При выборе ***логико-физической*** модели в диалоге **Create Model** необходимо указать необходимую СУБД и номер ее версии (в полях Database и Version).



При активизации кнопки **OK** открывается окно для построения новой модели данных и в списке выбора для переключения между логической и физической

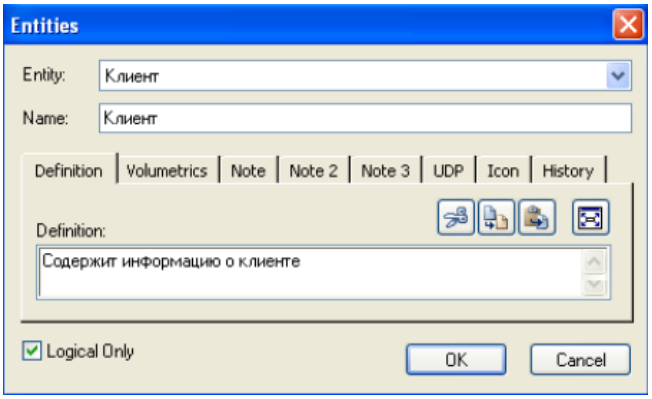
моделью (пункт ***Logical***, расположенный на панели инструментов) автоматически будет выбрана логическая модель.

4. Добавлять последовательно в окно новой модели сущности, изменяя при этом параметры шрифта для выделения её имени, ключевых полей и атрибутов.

!!! Для атрибутов задавать типы данных по смыслу.

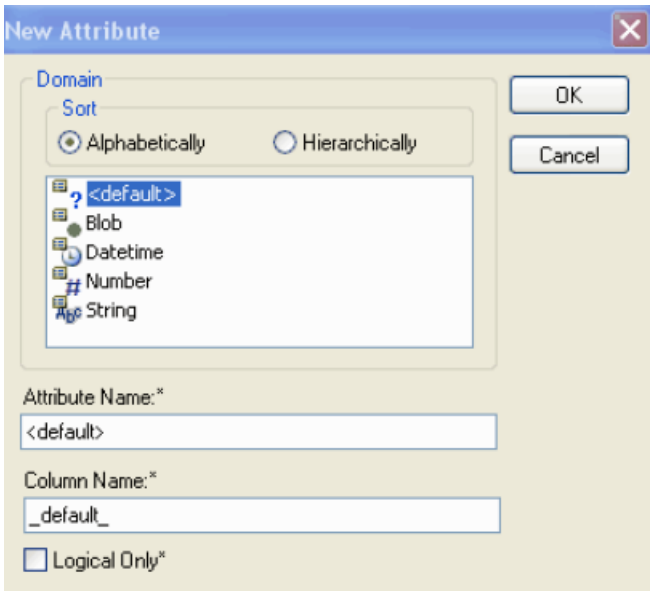
Для внесения сущности в модель необходимо щелкнуть левой кнопкой мыши по кнопке сущности  , расположенной на панели инструментов, и затем щелкнуть один раз в поле проектирования модели на том месте, где необходимо расположить новую сущность. В результате в поле проектирования появится сущность с именем E/1 по умолчанию.

Определение (задание) имени сущности осуществляется через диалог ***Entities***, который открывается через пункт **Entity Properties** (свойства сущности) контекстного меню по щелчку правой кнопкой мыши по выделенной сущности или пункта главного меню ***Model → Entitie***.



Определение атрибутов сущностей и их характеристик осуществляется в диалоговом окне ***Attributes***, которое открывается с помощью пункта **Attributes…** контекстного меню и позволяет ввести данные об атрибутах выбранной сущности:

При активизации кнопки **New…** диалогового окна **Attributes** открывается диалог **New Attribute**, позволяющий добавить новый атрибут для выделенной сущности:

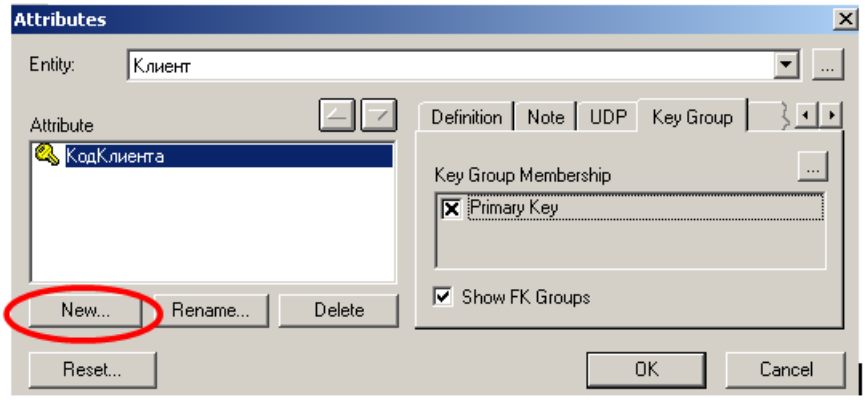


В диалоговом окне **New Attribute** указывается имя нового атрибута (поле ***Attribute Name***), имя, соответствующее ему в физической модели колонки (поле ***Column Name***) и домен (тип колонки на уровне физической модели).

При активизации кнопки **OK** новый атрибут добавляется в список атрибутов

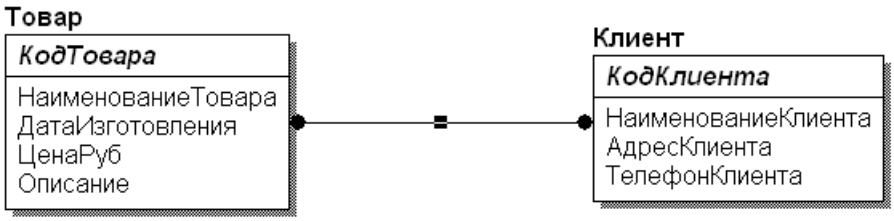
выделенной сущности, отражающийся в поле ***Attribute*** диалогового окна **Attributes**.

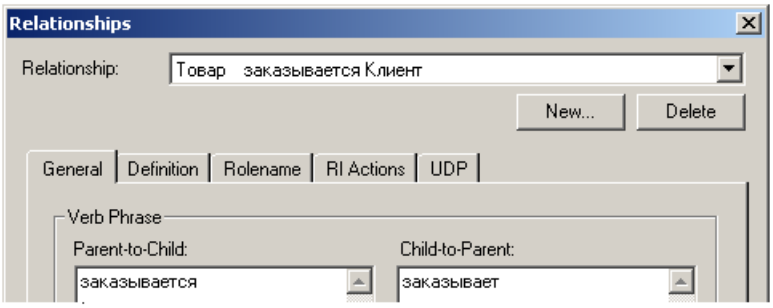
Для определения первичного ключа выделенной сущности необходимо перейти на вкладку ***General*** и сделать пометку в окне выбора *Primary Key*.



5. Определить связи между сущностями.

5.1. Задать связь «многие-ко-многим», например, между сущностями «Товар» и «Клиент»:

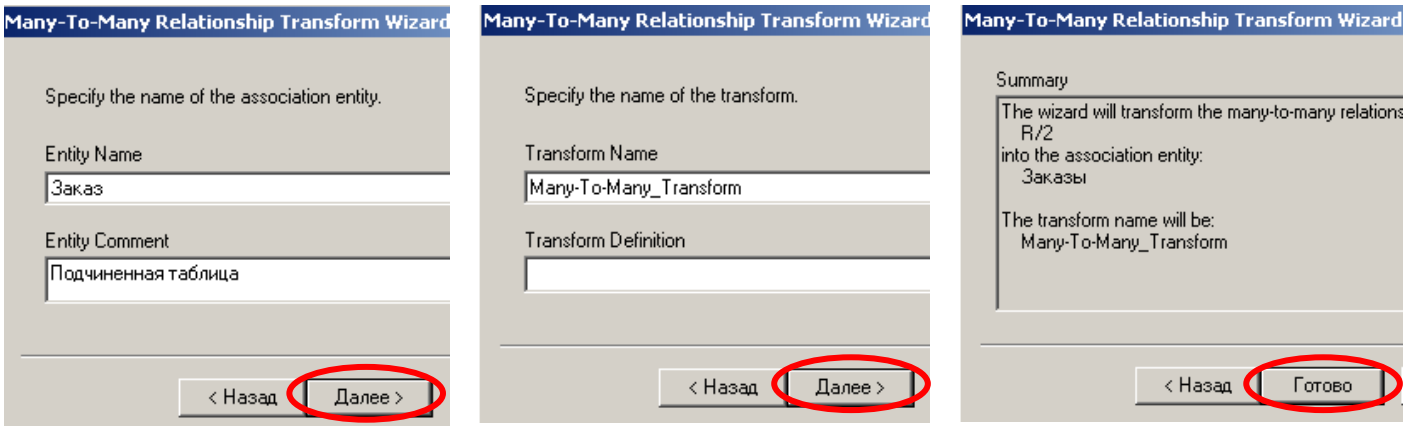




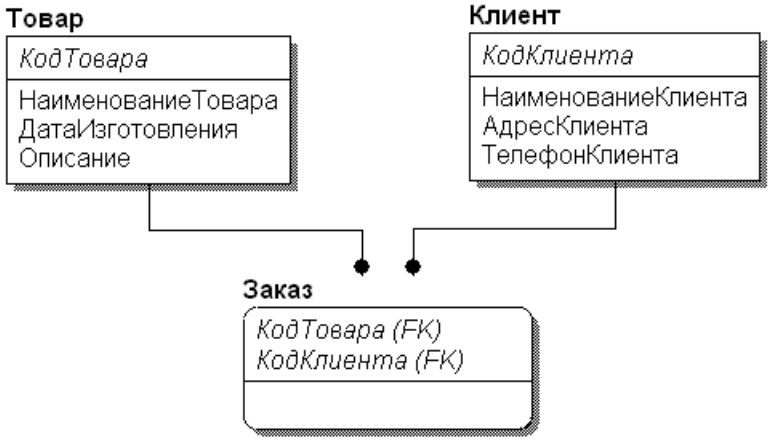
5.2. Принудительно преобразовать эту связь к двум новым связям "один-ко-многим":

- выбрать пункт контекстного меню ***Create Association Entity,*** в результате чего откроется диалог **Many-To-Many Transform Wizard:**

- диалог ***Many-To-Many Transform Wizard*** предлагает выполнить четыре шага для преобразования связи. Для перехода к следующему шагу необходимо щелкнуть по кнопке **Далее**. На втором и третьем шаге следует задать имя вновь создаваемой таблицы и имя преобразования:



- По окончании выполнения действий диалога ***Many-To-Many Transform Wizard*** на диаграмме будет добавлена новая таблица с заданным пользователем именем, а связь "многие-ко-многим" заменится идентифицирующими связями "один-ко-многим" от старых таблиц к новой таблице. При этом новые связи автоматически получат соответствующие имена от связи "многие-ко-многим":



5.3. Задать другие виды связей между сущностями,

определив для них мощность (Cardinality) и имя (Verb Phrase):

- ✓ *Идентифицирующая связь*
- ✓ *Неидентифицирующая связь*
- ✓ *Необязательная неидентифицирующая связь*

6. Определить для каждой связи правила ссылочной целостности, т.е. в окне свойств связи (двойным щелчком на линии связи) и на вкладке ***«RI Actions»*** установить требуемые ограничения при различных действиях с БД: *добавлении, удалении, изменении записей*.
Можно устанавливать следующие действия:

- **NONE** – ничего не делать;
- **RESTRICT** – при наличии связанных записей действие запрещено;
- **CASCADE** – действие распространяется на связанные записи;
- **NO ACTION** – аналогично NONE;
- **SET** – установить значение внешнего ключа в NULL.

Замечание: Просмотреть шаблоны триггеров и расширенный код, зависящий от выбранной СУБД на вкладке ***«Relationships Templates»*** контекстного меню **Relationships**.

7. Оформить отчет в следующем виде:

7.1. Сначала – описание предметной области, например:

БД должна хранить информацию: • о **продуктах**, поставляемых в магазин • об ежедневной **продаже** продуктов • о **заказах на поставку** продуктов • о **поставщиках** продуктов	1. Сущность «**Продукты**» • *Код продукта* – уникальный идентификатор, ключевой атрибут • *Продукт* – название продукта • *Единица измерения* – литры, килограммы, штуки и т.п. • *Срок хранения в днях* – для определения даты окончания срока годности продукта • *Условия хранения* – температура, влажность и т.п.
2.Сущность «**Поставщики**» • *Код поставщика* – уникальный идентификатор, ключевой атрибут • *Поставщик* – название организации или ФИО физического лица • *Код города* – город, где находится поставщик (для поиска) • *Адрес* – улица и дом (а также квартира – для физического лица) • *ФИО директора* • *Телефон* • *Факс*	3. Сущность «**Продажи**» • *Дата продажи* • *Код продукта* – какой именно продукт был продан • *Количество* – сколько продано этого продукта в тех единицах измерения, которые указаны для этого продукта в сущности Продукт • *Цена продажи* – цена при продаже за единицу продукта

7.2. Затем – «скриншоты» логической и физической ER-модели. Связи в логической модели дополнить описанием установленной мощности (Cardinality) и заданным именем (Verb Phrase).

7.3. Далее – «скриншоты» установленных правил ссылочной целостности для каждой связи.

Лабораторная работа № 7
Тема: Организация данных с помощью языка SQL. Проектирование индивидуальной БД.

Задание:

1. На основании логической модели БД, описанной в лаб_работе № 6, выполнить физическое проектирование индивидуальной БД в рамках реляционной модели базы данных с помощью языка SQL [*сначала с помощью языка определения данных (DDL)*. Провести операции изменения структуры таблиц и данных с помощью *языка манипулирования данными (DML)*]:

1.1. Создать основные таблицы БД (с использованием команды **CREATE TABLE**) с определением:

- ✓ подходящих типов данных для заданных столбцов;
- ✓ ограничений для некоторых (определенных самостоятельно) полей таблиц, например, определение ***уникальности*** значений или требования ***НЕ содержания пустых значений***;
- ✓ *первичных* ключевых полей (в таблицах-справочниках определить **простой** ключ, в подчиненной – **составной**);
- ✓ *внешних* ключей для определения связей между таблицами.

Замечание: *После создания таблиц открыть* ***Схему данных*** *и просмотреть созданные связи между таблицами!!!*

1.2. Создать дополнительную таблицу БД, не входящую в состав логической модели (с использованием команды **CREATE TABLE**) с тремя столбцами произвольной структуры без определения первичных и внешних ключей.

1.3. Внести изменения в структуру дополнительной таблицы, созданной в п. 1.2., удалив один из столбцов (с использованием команды **ALTER TABLE**).

1.4. Удалить дополнительную таблицу, созданную в п. 1.2. (с использованием команды **DROP TABLE**).

1.5. Внести изменения в структуру одной из основных таблиц, созданных в п. 1.1., дополнив её одним-двумя полями (с использованием команды **ALTER TABLE**). ***Например***, возможно добавить в таблицу «Товары» (в описанном выше примере БД) поля *Сорт, Макет, Описание, ЕдиницаИзмерения* и пр.

1.6. Создать индекс по одному из полей ***любой*** главной таблицы (с использованием команды **CREATE INDEX**). ***Например***, возможно создать индекс по полю НаименованиеТовара в таблице «Товары» (в описанном выше примере БД).

1.7. Заполнить данными созданные таблицы индивидуальной БД (с использованием команды **INSERT INTO**): сначала главные таблицы по 5-6 записей, затем подчиненную таблицу 15-тью записями.

Например, добавление записи в таблицу «Товары» (в описанном выше примере БД):

```
INSERT INTO Товары
VALUES (12, 'Стол', '20.02.2019', 'Мебель', 'Китай', 200.12, 2, 'Размер ВхШхГ: 80x1000x1200')
```

1.8. Удалить ОДНУ запись из подчинённой таблицы по заданному самостоятельно условию (с использованием команды **DELETE FROM**).

1.9. Сформулировать самостоятельно и внести изменения (с использованием команды **UPDATE**) в значения определенного атрибута таблицы по заданному условию-ограничению обновляемых записей (желательно использовать составное условие)**.**

Оформить отчет, в котором вывести:

- ✓ *коды всех созданных SQL-запросов,*
- ✓ *по одному коду-SQL добавления одной из записей в каждую таблицу БД,*
- ✓ *отобразить «скриншот» Схемы данных, полученной в результате создания физической модели,*
- ✓ *данные всех таблиц.*

Лабораторная работа № 8

Тема: Организация запросов в формате SQL в индивидуальной БД.

Задание:

1. **Сформулировать** и **создать** следующие простые **запросы-выборки** в формате **SQL** из таблиц индивидуальной БД (созданной в лаб. раб. № 7), причем источником данных каждого запроса должна быть ОДНА таблица:

1.1. запрос с выводом N% (N задать самостоятельно) записей таблицы, в которой провести сортировку по любому текстовому полю;

1.2. *запрос по текстовому полю* с использованием оператора **Like** для отбора записей по части значения поля;

1.3. *запрос по текстовому полю* с использованием оператора **In** для отбора записей с проверкой совпадения с любыми 3-мя или 4-мя элементами списка;

1.4. *запрос по полю типа Дата/время* с использованием оператора **BETWEEN...AND** для отбора записей с проверкой условия попадания даты во временной интервал (с… по…);

1.5. *запрос по оператору «И» (И-запрос)* для отбора записей, удовлетворяющих одновременно двум условиям для разных полей числового типа таблицы;

1.6. *запрос по оператору «ИЛИ» (ИЛИ-запрос)* для отбора записей, удовлетворяющих хотя бы одному из двух заданных условий для разных полей (с разными типами данных) таблицы;

2. Просмотреть ВСЕ созданные в лабораторной работе № 2 запросы в режиме SQL в СУБД Ms Access. **Сформулировать** и **создать** по аналогии **вычисляемые запросы** в формате **SQL** для таблиц индивидуальной БД с учетом операций соединения таблиц, если источником запроса являются несколько таблиц:

2.1. запрос с использованием ***арифметических*** операторов для ***числовых*** полей (например, вычисление *Стоимости* как Цена*Количество);

2.2. запрос с использованием ***арифметических*** операторов для полей с типом данных ***Дата*** (например, вычисление *Количества дней*, как разности между двумя датами, одна из которых может быть текущей);

2.3. параметрический запрос (например, вычисление *Стоимости* или *цены в валюте*, курс которой вводится с клавиатуры);

2.4. запрос с использованием функции ***Iif( ),*** применимой к любому числовому полю таблицы с как минимум двумя проверяемыми интервалами (например, вычисление *Скидки к цене* товара в зависимости от проданного количества).

3. Просмотреть ВСЕ созданные в лабораторной работе № 3(3) запросы в режиме SQL в СУБД Ms Access. **Сформулировать** и **создать** по аналогии **итоговые запросы** (по одному из каждой темы) в формате **SQL** для ***подчиненной*** таблицы индивидуальной БД (использовать при этом максимально разные статистические функции):

3.1. обобщающий запрос для одной группы записей;

3.2. обобщающий запрос для нескольких групп записей;

3.3. обобщающий запрос по всем записям;

3.4. запрос с применением критериев отбора для поля, обработанного групповой операцией (ограничение числа сгруппированных записей);

3.5. запрос с применением критериев отбора для поля после его обработки групповой функцией);

3.6. запрос с применением критериев отбора для поля, не обработанного групповой операцией.

4. **Сформулировать** и **создать подчиненные запросы** (по одному каждого типа) в формате **SQL** для любых таблиц индивидуальной БД.

5. **Сформулировать** и **создать запросы** на использование операций **соединения** любых таблиц индивидуальной БД в формате **SQL**: внутреннее соединение, левое и правое внешние соединения.

*Оформить отчет, в котором вывести **смысловые** формулировки и коды всех созданных SQL-запросов и «скриншоты» результирующих данных.*

Лабораторная работа № 9

Тема: Работа с программируемыми объектами SQL в индивидуальной БД.

Задание:

1. **Сжать** индивидуальную БД (записать размер БД до и после сжатия):

в версии MsA 2003:

- ✓ *открытую* БД: **п.м. Сервис → Служебные программы → Сжать и восстановить БД;**
- ✓ *неоткрытую* БД: **п.м. Сервис → Сжать и восстановить БД →** указать путь и имя сжимаемой БД → указать путь и имя сжатой БД.

в версии MsA 2007/2010:

- ✓ открыть БД → вызвать кнопкой «Office» (или п.м. Файл) главное меню приложения → **Параметры Access** → Текущая база данных → в группе настроек «Параметры приложений» установить флажок ☑ Сжимать при закрытии → **ОК** (после чего закрыть и открыть БД).

в версии MsA 2010/2013/(?2016):

- ✓ команда **Сжать и/или восстановить БД** на вкладке ленты **Работа с базами данных** в группе инструментов **Сервис**.

2. **Создать** резервную копию уже сжатой БД.

3. Загрузить и установить бесплатную версию **SQL Server 2005 Express Edition**.

4. Загрузить и установить **MS SQL Server Management Studio**.

5. Преобразовать индивидуальную БД (созданную в лаб_работе № 7 средствами SQL в MS ACCESS) в базу **SQL Server**:

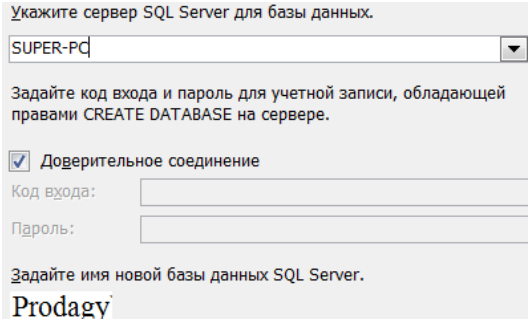
в версии MsA 2003 / 2007 / 2010:

с помощью мастера преобразования, входящего в поставку

- ✓ MS Access 2003: ***п. м. Сервис → Служебные программы → Мастер преобразования в формат SQL Server***
- ✓ MS Access 2007/2010: команда **SQL Server** на вкладке ленты **Работа с базами данных** в группе инструментов **Переместить данные:**

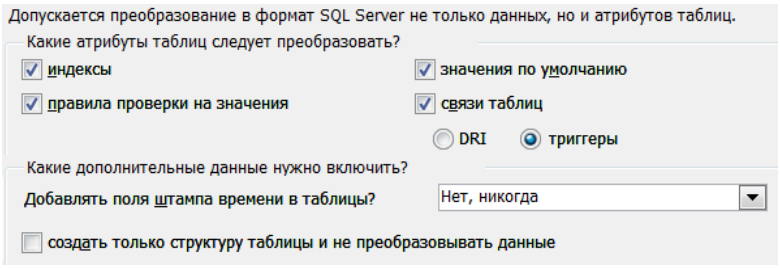
1 шаг:

2 шаг:

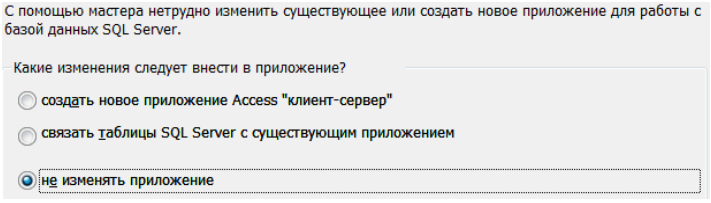


3 шаг: выбрать ВСЕ таблицы

4 шаг:



5 шаг:



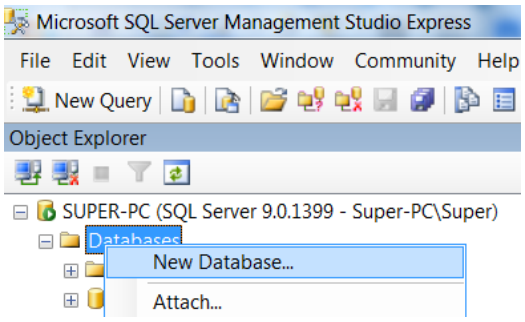
6 шаг: **Готово**.

Последним окном при работе мастера преобразования в формат MS SQL Server будет окно **ОТЧЕТА**, для которого вызвать контекстное меню (правой кнопкой мыши) в любом месте и выбрать пункт **Экспорт**. Сохранить отчет в текстовом формате или HTML.

в версии MsA 2007 / 2010 / 2013 :

с помощью **ODBS** (*технологии, которая позволяет приложениям обмениваться данными, несмотря на различия в применяемых СУБД*):

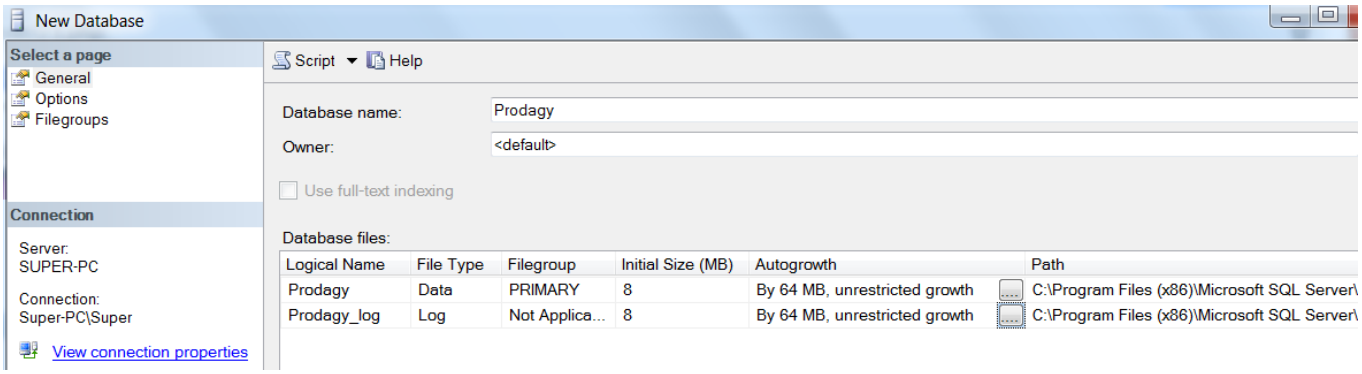
5.1. Открыть SQL Server Management Studio.



5.2. Создать новую БД: нажать правую кнопку мыши на узел **Databases**. Затем в появившемся контекстном меню выбрать пункт ***New Database***. В появившемся окне **New Database** в поле ***Database*** ввести название новой БД (*Например, Shop или Prodagy*). Следующее поле ***Owner*** задает владельца базы данных (по умолчанию оно имеет значение <default>, то есть владельцем будет тот, кто создает эту БД). Оставить это поле без изменений.

Далее идет таблица для установки общих настроек БД. Она содержит две строки - первая для установки настроек для главного файла, где будут храниться данные, и вторая строка для конфигурации файла логгирования.

В частности, можно установить следующие настройки:



Logical Name: *логическое имя, которое присваивается файлу базы данных.*

File Type: *есть несколько типов файлов, но, как правило, основная работа ведется с файлами данных (ROWS Data) и файлом лога (LOG)*

Filegroup*: обозначет группу файлов. Группа файлов может хранить множество файлов и может использоваться для разбиения базы данных на части для размещения в разных местах.*

Initial Size (MB): *устанавливает начальный размер файлов при создании (фактический размер может отличаться от этого значения).*

Autogrowth/Maxsize: *при достижении базой данных начального размера SQL Server использует это значение для увеличения файла.*

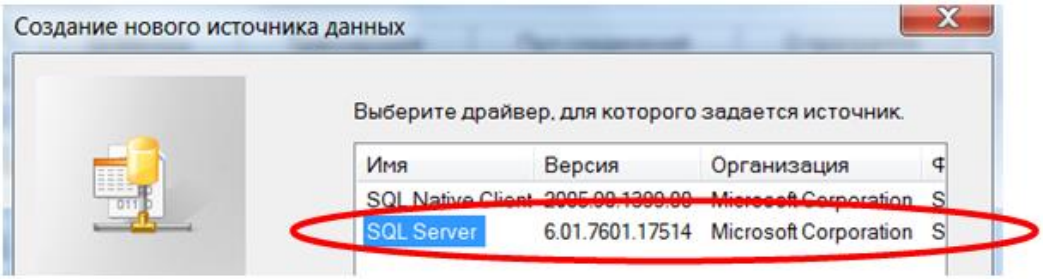
Path*: каталог, где будут храниться базы данных.*

File Name: *непосредственное имя физического файла. Если оно не указано, то применяется логическое имя.*

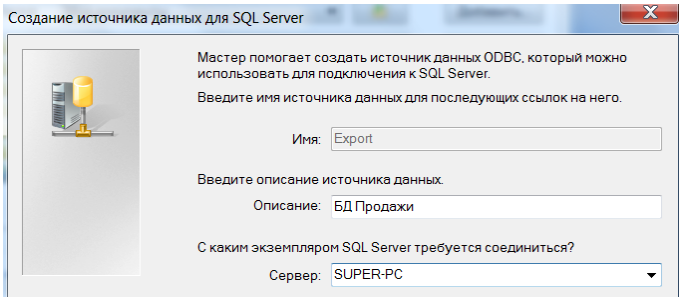
После ввода названия БД нажать **ОК** – БД будет создана и появится среди баз данных сервера.

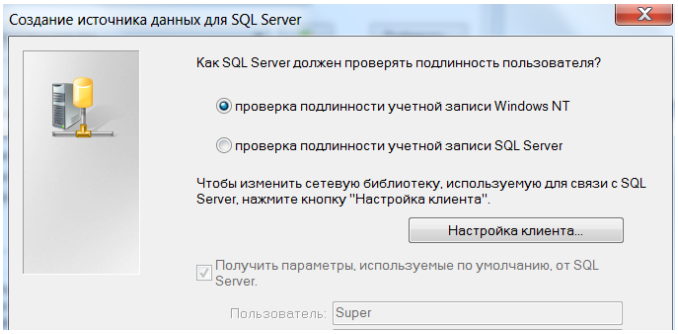
5.3. Добавить **источник данных ODBC**:

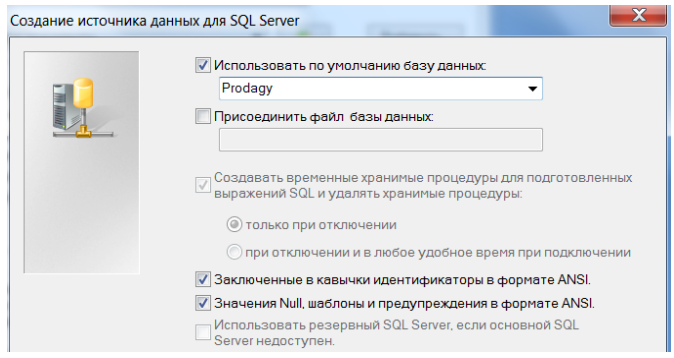
- ✓ нажать кнопку **Пуск** и выбрать пункт **Панель управления**;
- ✓ на панели управления дважды щелкнуть элемент **Администрирование**;
- ✓ в диалоговом окне «Администрирование» дважды щелкнуть элемент **Источники данных (ODBC)**;
- ✓ в открывшемся диалоговом окне «Администратор источников данных ODBC» открыть вкладку ***Файловый DSN*** и нажать кнопку **Добавить**;

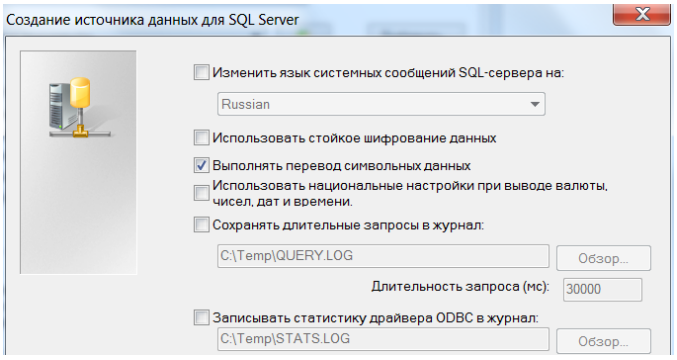


- ✓ выбрать драйвер, который нужно использовать, и **Далее**;
- ✓ ввести имя файлового источника данных (*например, Export*) и нажать кнопку **Далее**;
- ✓ нажать кнопку **Готово** и задать некоторые опции в последующих диалоговых окнах:

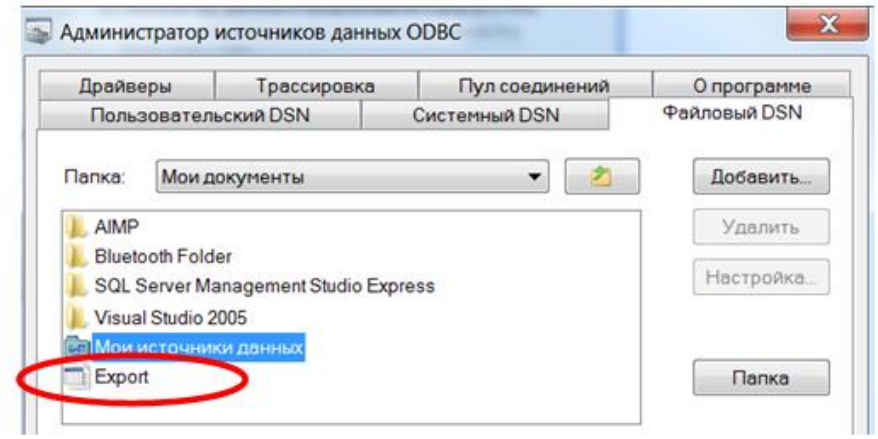






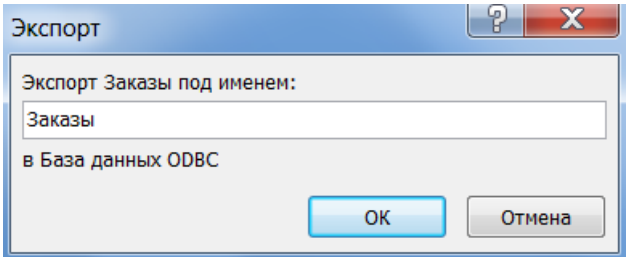


- ✓ нажать кнопки **Готово**→ **ОК**

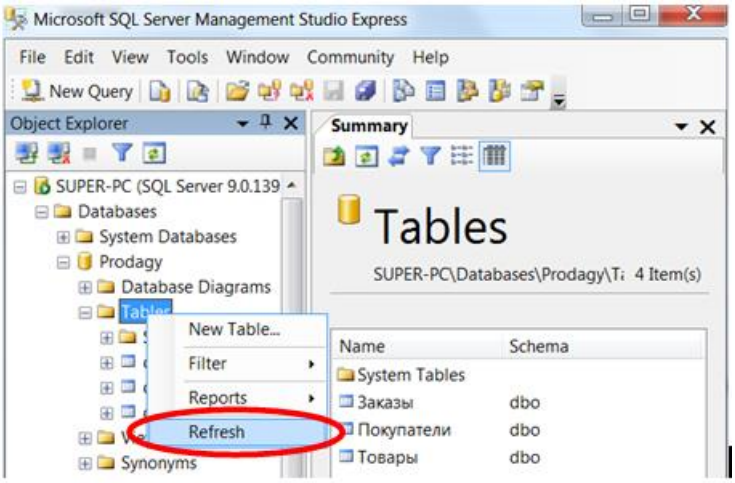


5.4. Экспортировать **ВСЕ** таблицы индивидуальной БД в БД **SQL Server:**

- ✓ в окне БД приложения MS ACCESS в списке объектов выделить таблицу (например, таблицу ***Заказы***);
- ✓ вызвать команду **База данных ODBS** на вкладке ленты **Внешние данные** в группе инструментов **Экспорт** в списке команд **Дополнительно**;



- ✓ в появившемся окне **«Выбор источника данных»** на вкладке Файловый источник данных выбрать созданный в п. 5.3. текущего задания источник (*Export*);
- ✓ появившееся окно «Экспорт – База данных ODBS» закрыть без изменений;
- ✓ открыть ***SQL Server Management Studio*** и обновить содержимое узла **Tables** (команда ***Refresh*** в контекстном меню).



6. Создать и выполнить запрос-выборку к любой таблице индивидуальной БД в **SQL Server** используя ***SQL Server Management Studio:***

```
USE prodagy

SELECT Товары.*
FROM Товары
WHERE (((Товары.[Цена за ед (руб)])>1000));
```

Results | Messages

	код това...	наименован...	единица измере...	цена за ед (р...
1	12	мука	кг	1950,00
2	15	сахар	кг	2100,00
3	16	сок	л	2000,00
4	18	печенье	уп	2500,00
5	19	йогурт	л	1980,00

7. Выполнить следующие ***системные хранимые процедуры*** (*используются для выполнения действий с системным каталогом или получения системной информации и начинаются с префикса sp_*):

7.1. формирования отчета об именах учетных записей и соответствующих им в каждой БД пользователях:

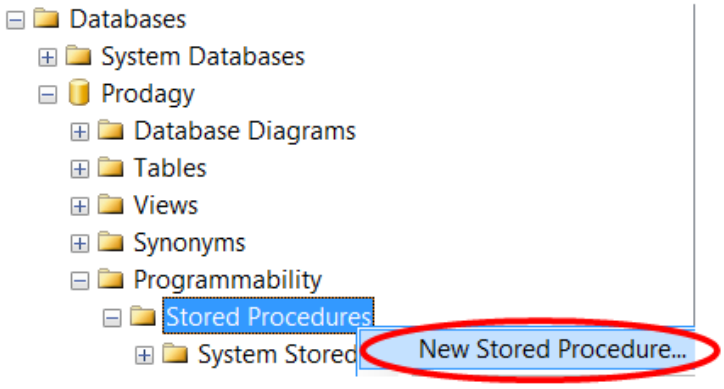
```
EXEC sp_helplogins;
```

7.2. вывода списка БД и их размера:

```
EXEC sp_databases;
```

8. Создать и выполнить ***хранимую процедуру*** для извлечения данных из любой таблицы индивидуальной БД:

в окне ***SQL Server Management Studio:***



Например:

```
USE prodagy;
GO
CREATE PROCEDURE Product
AS
BEGIN
    SELECT Наименование AS Продукт, [цена за ед (руб)], [дата поступления]
    FROM Товары
END;
GO
```

выполнение процедуры:

```
USE prodagy;
EXEC Product
```

9. Создать ***функцию*** для анализа, например, минимального запаса в таблице «Товары» индивидуальной БД с использованием условного оператора **IF..ELSE** или оператора **CASE**: если товара менее 2 единиц, то значение функции – сообщение «низкий запас», если товара от 3 до 10, то значение функции – сообщение «нормальный запас», в противном случае – «избыток продукции». Использовать созданную функцию в запросе к таблице «Товары».

10. Создать ***триггер*** (и протестировать его работу) на добавление записи в любую таблицу индивидуальной БД. Данный триггер в случае успешного добавления данных должен выводить в окне сообщений «Запись добавлена»:

сначала создать триггер:

```
USE prodagy;
GO

CREATE TRIGGER INSERT_INDICATION
ON Товары
AFTER INSERT
AS
BEGIN
SET NOCOUNT ON;
PRINT 'Запись добавлена'
END
GO
```

затем запрос на добавление одной или нескольких записей:

```
USE prodagy;

INSERT INTO Товары
VALUES (22, 'нектар', 'л', 600,'05/01/2020', 24, '05/04/2020');
```

Оформить отчет, в котором вывести коды всех созданных SQL-запросов и «скриншоты» результирующих данных.

Лабораторная работа № 10

***Тема:* Освоение способов доступа к данным из прикладных программ.**

Задание:

1. Создать ODBC-запрос, позволяющий импортировать данные из БД на рабочий лист MS Excel.

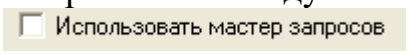
1.1. Скопировать базу данных Товарообороты (***Товарообороты.mdb***) на рабочий диск R:\.

1.2. Открыть базу данных и создать вычисляемый запрос с именем ***Стоимость поставки***, в который:

- включить **все поля *кроме*** *Дата поставки* из таблицы *Поставки*, поле Название из таблицы *Магазины*, поля Наименование, Сорт, Цена за единицу из таблицы *Товары*;
- добавить новое поле *Стоимость поставки*, рассчитав его следующим образом: ***Количество * Цена за единицу***.

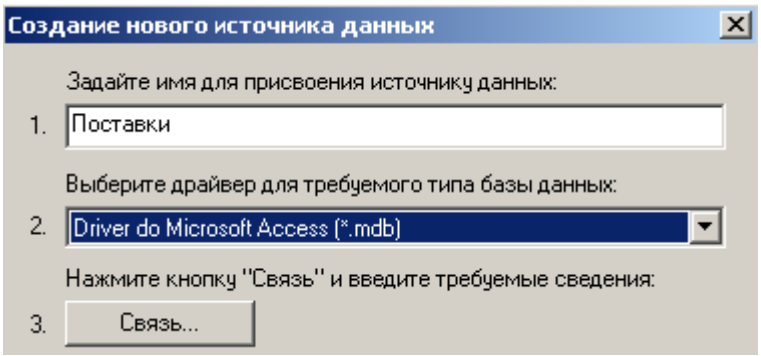
1.3. Открыть табличный процессор EXCEL, в котором выполнить импорт данных из запроса ***Стоимость поставки***, для чего:

1.3.1. выбрать команду из п.м. **Данные → Импорт внешних данных → Создать запрос**;

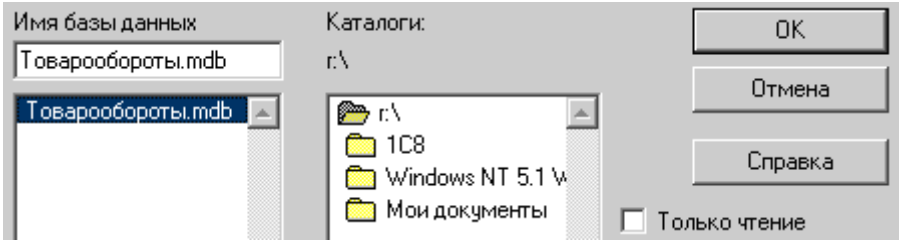
1.3.2. в окне «Выбор источника данных» на вкладке Базы данных из списка выбрать команду *<Новый источник данных>* → убрать флажок 

→ кнопка **ОК** ;

1.3.3. в окне «Создание нового источника данных» задать описательное имя источника данных (*например*, ***Поставки***), а также выбрать драйвер для требуемого типа базы данных (Driver do Microsoft Access):

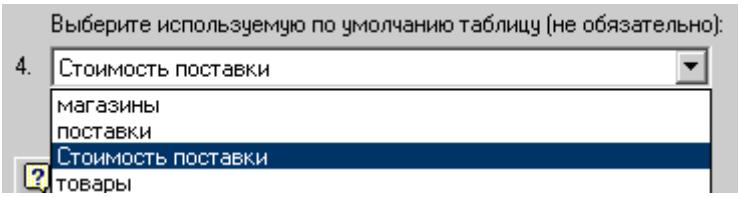


1.3.4. в текущем окне нажать кнопку **Связь** → в появившемся диалоговом окне «Установка драйвера ODBC для MsA» нажать кнопку **Выбрать** → в окне «Выбор базы данных» выделить необходимую БД → кнопка **ОК** :



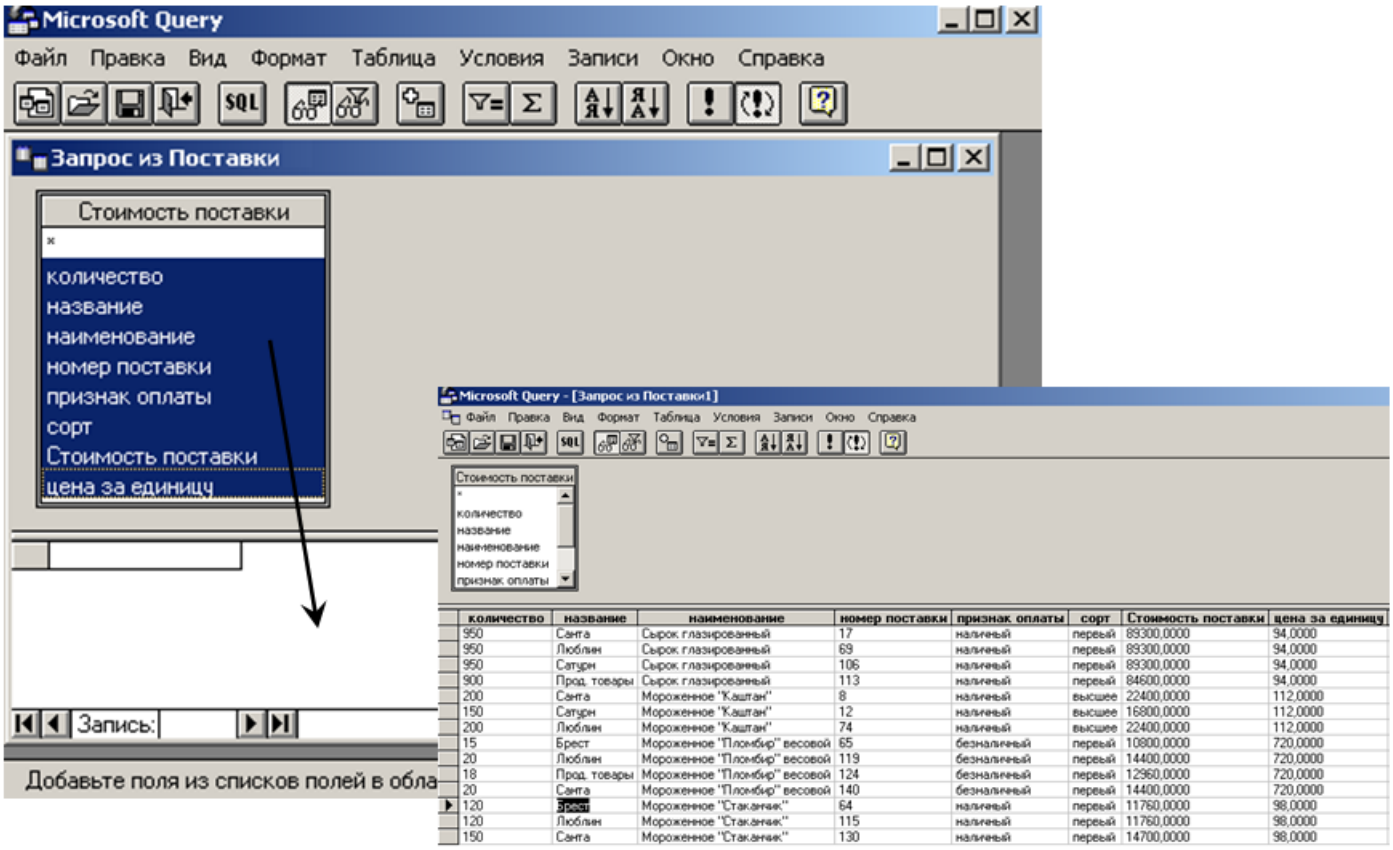
1.3.5. в окне «Установка драйвера ODBC для MsA» нажать **ОК**;

1.3.6. в окне «Создание нового источника данных» выбрать запрос ***Стоимость поставки*** → кнопка **ОК**:



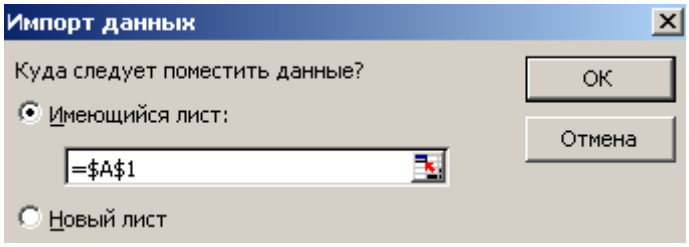
1.3.7. в окне «Выбор источника данных» нажать **ОК** ;

1.3.8. в появившемся окне «Microsoft Query» включить все поля из запроса ***Стоимость поставки***, для чего необходимо в списке полей запроса выделить все поля при помощи клавиши **SHIFT** и перетащить их в нижнюю часть бланка запроса:



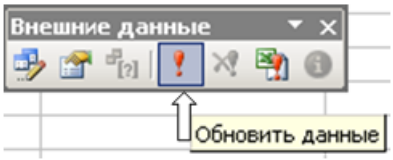
1.3.9. сохранить запрос с именем «Запрос – стоимость поставки», выбрав крманду из п. м. **Файл → Сохранить запрос →** R:\КИТ\... (обратить внимание на местонахождение файла-запроса);

1.3.10. импортировать данные на рабочий лист Excel, выбрав команду из п.м. **Файл → Вернуть данные**, либо кнопка  на панели инструментов:



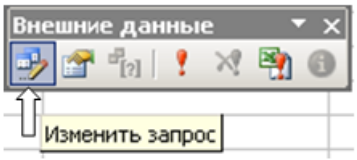
1.4. Перейти в Ms Access и добавить новую запись о поставке в соответствующую таблицу.

1.5. Перейти в Ms Excel на рабочий лист с импортированными данными → выбрать команду ***Обновить данные*** на панели инструментов «Внешние данные» (или из контекстно-зависимого меню) → в окне «Обновление данных» нажать **ОК**.



1.6. Отредактировать имеющийся запрос, с целью получения из БД информации, соответствующей определенному магазину, для чего:

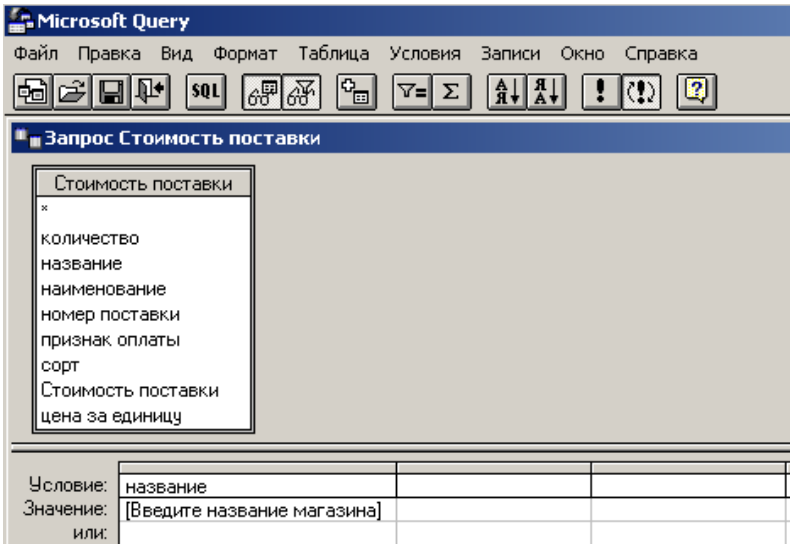
1.6.1. на панели инструментов «Внешние данные» выбрать команду ***Изменить запрос***:



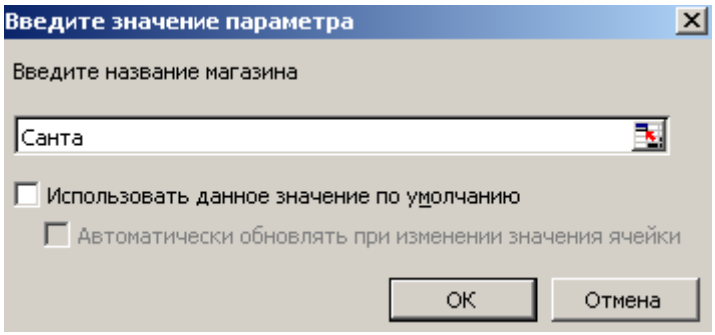
1.6.2. отобразить на экране область условий, выбрав команду из п.м. **Вид** → **Условия**;

1.6.3. Изменить запрос на параметрический:

- в поле ***Условие*** выбрать поле **Название** из списка полей;
- в поле ***Значение*** задать параметр: [Введите номер магазина]:



1.6.4. Нажать на панели инструментов копку → ввести название магазина, имеющийся в БД, *например*:



1.6.5. проверить работу запроса.

1.7. Выполнить обновление результата работы запроса, для чего:

- поместить курсор в таблицу данных;
- нажать на панели инструментов копку ;
- выбрать другое название магазина;
- выбрать команду из п.м. **Данные → Обновить данные**.

ВОПРОСЫ К ЗАЧЕТУ

для специальностей
"Экономика электронного бизнеса" (1 - 28 01 01),
"Электронный маркетинг" (1- 28 01 02)
по дисциплине ***«Базы данных»***

1. Основные понятия: информация, данные, экономическая информация. Виды и структурные единицы экономической информации.
2. Определение и свойства системы. Экономические информационные системы, их функции, классификация.
3. Состав информационного обеспечения, информационная база.
4. Основные понятия: данные, обработка данных, база данных (БД). Единица информации в БД. Информационная модель данных, её состав.
5. Определение базы данных. Преимущества базы данных. Роль баз данных и их сферы применения.
6. Схема взаимодействия пользователей с БД. Понятия интегрированности и разделимости данных. Компоненты системы баз данных.
7. Определение жизненного цикла базы данных и его этапы.
8. Концептуальное проектирование. Логическое проектирование. Физическое проектирование.
9. Сущность моделирования. Классификация моделей. Определение модели данных. Реализация модели данных.
10. Концептуальная модель данных.
11. Логические модели данных: иерархическая, сетевая, реляционная, объектная, объектно-ориентированная, объектно-реляционная. Их достоинства и недостатки.
12. Физические модели данных.
13. Определение системы управления базами данных (СУБД). Реляционные СУБД.
14. Определение и свойства сущности. Определение возможного ключа. Первичный ключ и альтернативные ключи.
15. Синтаксис и семантика выражений реляционной алгебры.
16. Теоретико-множественные реляционные операции объединения, пересечения, разности и декартова произведения. Примитивные и не примитивные реляционные операции.
17. Связь типа «один к одному».
18. Связь типа «один ко многим». Внешний ключ и его свойства.
19. Связь типа «многие ко многим». Преобразование связи «многие ко многим» в две связи «один ко многим» и связующую таблицу.
20. Понятие нормализации данных.
21. Приведение сущности к первой нормальной форме.
22. Приведение сущности ко второй нормальной форме. Виды аномалий в данных, устраняемые после приведения ко второй нормальной форме.
23. Приведение сущности к третьей нормальной форме. Виды аномалий в данных, устраняемые после приведения к третьей нормальной форме.
24. Нормальная форма Кодда-Бойса. Приведение сущности к четвертой нормальной форме.
25. Приведение сущности к пятой нормальной форме.

26. Понятие целостности данных. Целостность таблиц.
27. Целостность внешних ключей (ссылочная целостность).
28. Основные и дополнительные правила ссылочной целостности. Целостность типов данных.
29. Общая характеристика СУБД Microsoft Access (MsA). Объекты MsA: их назначение и особенности, общие принципы и способы работы с ними.
30. Типы данных, обрабатываемые в MsA.
31. Таблица как объект БД. Понятие записи и поля таблицы. Ключевые поля в таблицах. Их виды. Назначение и использование.
32. Свойства полей: Маска ввода, Условие на значение, Значение по умолчанию. Примеры.
33. Отбор данных с помощью фильтров. Условие отбора. Оператор Like. Сохранение фильтра.
34. Схема данных в MsA. Условия и способы создания связей между таблицами. Межтабличные связи. Типы отношений между таблицами.
35. Целостность данных в MsA. Каскадные операции.
36. Запрос как объект БД в MsA. Виды запросов. Способы создания запроса в Microsoft Access. Условие отбора.
37. Формирование условий отбора: операторы сравнения, логические операторы.
38. Формирование условий отбора для полей с типом данных Дата/Время. Сложные критерии выборки.
39. Технология создания простых запросов в режиме Конструктора в MsA. Вид бланка QBE. Способы включения полей. Выполнение и сохранение запроса.
40. Запрос на выборку данных в MsA: назначение, виды. Параметрический запрос: назначения, особенности, преимущества.
41. Вычисляемое поле в вычисляемом запросе в MsA: правила и способы создания. Установка свойств полей запроса.
42. Использование встроенных функций в вычисляемом поле в MsA. Функции даты/времени. Примеры.
43. Использование встроенных функций в вычисляемом поле в MsA. Функция Format( ). Функция Iif( ). Примеры.
44. Создание итоговых запросов в MsA. Установки групповых операций. Групповые функции. Виды итоговых запросов.
45. Перекрестный запрос в MsA: назначение, правила и способы создания. Условия отбора и параметры в перекрестном запросе.
46. Технологии создания активных запросов в MsA: запрос на обновление данных; запрос на удаление данных, запрос на создание таблицы; запрос на добавление данных.
47. Форма как объект БД в MsA: назначение и возможности. Способы проектирования форм. Виды форм, создаваемые мастером.
48. Режим Конструктора форм в MsA. Области формы. Типы элементов управления, их виды и назначение.
49. Свойства и события формы и её объектов в MsA. Свойства элементов управления. Создание кнопок управления в форме.
50. Отчет как объект БД в MsA: назначение и возможности. Способы проектирования отчетов.
51. Окно Конструктора отчетов в MsA. Области (разделы) окна отчета. Типы элементов управления, их виды и назначение.

52. Макрос как объект БД в MsA: назначение и возможности. Классификация макрокоманд.
53. Понятие события в MsA. Категории и примеры событий.
54. Классификация макросов для обработки событий. Применение условий в макросе.
55. Модуль как объект БД в MsA. Visual Basic for Application (VBA): возможности, объекты и семейства языка.
56. Модули VBA: назначение, типы. Состав программного кода VBA.
57. Понятие и типы процедур в VBA. Синтаксис процедуры-функции и процедуры-подпрограммы.
58. Порядок создания функции пользователя в VBA. Использование пользовательской функции в запросах.
59. Управляющие конструкции языка VBA: Условный оператор If./.Then... Примеры.
60. Управляющие конструкции языка VBA: Оператор выбора Select Case. Примеры.
61. Назначение, функциональные возможности, достоинства, формы языка структурированных запросов SQL. Язык SQL в СУБД.
62. Структура SQL-команды. Типы данных SQL. Константы. Преобразование данных.
63. Категории операторов в SQL. Логические связки NOT, AND, OR.
64. Операторы IN, BETWEEN, LIKE, IS NULL.
65. Выражения. Использование метасимволов «%» и «_» с оператором LIKE.
66. Соглашения по записи форматов SQL-команд.
67. Определение данных в языке SQL. Конструкции языка SQL для организации данных в БД.
68. Создание доменов. Создание реляционных таблиц. Синтаксис оператора CREATE TABLE.
69. Изменение структуры таблицы (добавление и удаление полей, создание и удаление индексов). Удаление таблицы.
70. Манипулирование данными. Ввод данных в таблицу. Обновление, удаление записей в таблице.
71. Операции соединения таблиц. Создание индексов, представлений.
72. Манипулирование данными в языке SQL: конструкции языка.
73. Язык запросов к данным в SQL. Выборка данных. Синтаксис (аргументы) оператора SELECT.
74. Фильтрование данных с помощью предложения WHERE. Логические условия для построения условий выборки.
75. Упорядочение набора данных с помощью ORDERBY. Сортировка строк, использование вычисляемых полей. Выражения, SQL-функции.
76. Подведение итогов с помощью функций агрегирования. Группировка строк и подсчет итоговых данных. Групповые функции SQL. Предложения GROUPBY и HAVING.
77. Параметры в запросах. Особенности создания параметрических запросов.
78. Использование подзапросов. Вложенные подзапросы, коррелирующие подзапросы. Использование операторов ANY, ALL, EXISTS.
79. Выборка данных из нескольких таблиц. Реализация операций реляционной алгебры средствами языка SQL.

80. Операции соединения таблиц в запросах. Внутреннее и внешнее соединение таблиц.
81. Процедурный SQL: назначение, основные программируемые объекты БД.
82. Элементы процедурного SQL (переменные, переменные определяемые пользователем, системные переменные, переменные в хранимой процедуре).
83. Составной оператор BEGIN..END. Условные операторы. Оператор-селектор CASE. Циклы.
84. Хранимые процедуры и функции. Триггеры. Курсоры.
85. Повышение производительности БД: индексирование, оптимизация запросов. Резервное копирование и восстановление данных. Полная архивная копия БД.
86. Удаленные запросы, распределенные транзакции. Проблемы параллельного выполнения транзакций.
87. Назначение механизма репликации (тиражирования) данных.
88. Политика безопасности. Правила защиты БД.
89. Средства защиты данных от несанкционированного доступа: идентификация, аутентификация, пользовательские роли, механизм привилегий. Безопасный доступ к данным.
90. Модель взаимодействия отрытых систем. Классификация клиент-серверных систем.
91. Клиент-серверные СУБД. Модели распределения функций.
92. Понятие и состав распределенной БД, аспекты её проектирования. Предпосылки децентрализации.
93. Система управления распределённой БД. Аспекты проектирования распределённых БД.
94. Системы «клиент-сервер» как частный случай распределенных систем. Преимущества и недостатки распределённых БД.
95. Классификация клиент-серверных систем.
96. Клиент-серверные базы данных. Примеры клиент-серверных СУБД.
97. Понятие многомерной базы данных. Размещение информации и схема адресации. Назначение технологий разработки (извлечения) данных.
98. Хранилище данных. Подготовка данных, предназначенных для хранения в хранилищах данных.
99. Особенности хранилищ данных по сравнению с операционными базами данных. Магазины (витрины данных).
100. Интерактивная аналитическая обработка OLAP. Требования к OLAP-инструментам. Основные типы OLAP-систем. OLAP-куб.

_____ *(группа _____)*

(ФИО)

База данных:_____

1. Связи между таблицами:

2.1. Запрос на выборку:

Запрос на выборку:

Поле					
Имя таблицы					
Условие отбора					
или:					

_____

_____

_____

2.2. Вычисляемый запрос:

Запрос на выборку:

Поле					
Имя таблицы					
Условие отбора					
или:					

Вычисляемые поля:

_____

_____

_____

_____

_____

2.3. Итоговый запрос:

Запрос на выборку:				
Поле				
Имя таблицы				
Групповая операция				
Условие отбора				
или:				

2.4. Перекрестный запрос:

Перекрестный запрос:					
Поле					
Имя таблицы					
Групповая операция					
Перекрестная таблица					
Условие отбора					

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ
Первый проректор БрГТУ
_____ М.В. Нерода
« »_____2022 г.

Регистрационный № УД-_____ /уч.

Базы данных

Учебная программа учреждения высшего образования
по учебной дисциплине для специальностей**:**

1– 28 01 02 «Электронный маркетинг»,
1 – 28 01 01 «Экономика электронного бизнеса»

2022 г.

Учебная программа составлена на основе образовательных стандартов

для специальности «Электронный маркетинг» ОСВО 1 – 28 01 02 – 2021,

для специальности «Экономика электронного бизнеса» ОСВО 1 – 28 01 01 – 2021,

(название образовательного стандарта)

типовой учебной программы «Базы данных», утвержденной Министерством образования Республики Беларусь 21.11.2019, регистрационный № ТД-Е.840/тип.,

типовой учебной программы «Базы данных», утвержденной Министерством образования Республики Беларусь 01.06.2021, регистрационный № ТД-I.1569/тип.

и учебных планов специальностей.

СОСТАВИТЕЛЬ:

Гучко И.М., старший преподаватель кафедры информатики и прикладной математики

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:

Кафедрой информатики и прикладной математики

Заведующий кафедрой _____ С.И. Парфомук

(протокол № 4 от 14.12.2022);

Методической комиссией экономического факультета

Председатель методической комиссии _____

подпись *Инициалы, фамилия*

(протокол №____от _____);

Научно-методическим советом
Брестского государственного технического университета
(протокол №____от _____);

1. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Место учебной дисциплины.

Учебная программа по учебной дисциплине «Базы данных» разработана для студентов, обучающихся по специальностям 1 – 28 01 02 «Электронный маркетинг», 1 – 28 01 01 «Экономика электронного бизнеса» в соответствии с требованиями образовательных стандартов ОСВО, типовых учебных планов вышеуказанных специальностей, типовой учебной программы «Базы данных», утвержденной Министерством образования Республики Беларусь 01.06.2021, регистрационный № ТД-I.1569/тип. и типовой учебной программы «Базы данных», утвержденной Министерством образования Республики Беларусь 21.11.2019, регистрационный № ТД-Е.840/тип. Является учебной дисциплиной государственного компонента для специальности 1 – 28 01 02 «Электронный маркетинг» и дисциплиной компонента учреждения высшего образования для специальности 1 – 28 01 01 «Экономика электронного бизнеса».

В течение последних лет в связи с бурным развитием информационных технологий стало возможным использование новых способов хранения и обработки информации в информационных системах, в которых важнейшим компонентом являются базы данных (БД). Структурированная, систематизированная информация легко анализируется и обрабатывается, а при условии хранения в базе, постоянно обновляется и дополняется, что позволяет говорить о её неизменной актуальности.

Подготовка современного специалиста требует уверенного владения современными технологиями баз данных, которые широко применяются для повышения эффективности обработки данных во всех сферах бизнеса и экономической деятельности.

Изучение учебной дисциплины «Базы данных» является составной частью профессионального обучения, призвано дать студентам с квалификацией «маркетолог-программист» и «экономист-программист» профессиональные технологические знания и умения по выбранной специальности, воспитывать производственную и технологическую культуру, формировать умения применять на практике полученные теоретические знания в области проектирования, создания и ведения баз данных.

Цель, задачи учебной дисциплины.

Цель изучения дисциплины:

- ✓ получить представления об исследованиях и современных разработках в области технологий баз данных;
- ✓ овладеть основами моделирования и проектирования современных БД;
- ✓ получить практические навыки, связанные с технологиями манипулирования данными;
- ✓ приобрести реальные навыки работы с базами данных с ориентацией на решение задач экономики электронного бизнеса и маркетинга.

Задачи дисциплины:

- ✓ изучение основных понятий баз данных, систем управления базами данных (СУБД), моделей данных;

- ✓ изучение основ реляционной алгебры, реляционной модели данных, основных принципов и особенностей разработки логических и физических моделей данных;
- ✓ изучение процесса проектирования и создания реляционных бах данных;
- ✓ освоение приемов работы с технологическими и инструментальными средствами построения моделей данных, создания физических объектов БД;
- ✓ изучение способов управления, сопровождения и администрирования баз данных и обеспечения безопасности данных;
- ✓ овладение методами и спецификой работы клиент-серверных механизмов взаимодействия с базами данных;
- ✓ освоение механизмов взаимодействия с базами данных при решении прикладных задач;
- ✓ освоение навыков самостоятельного поиска, изучения и анализа технической литературы по данной учебной дисциплине.

Требования к уровню освоения содержания учебной дисциплины.

В результате изучения учебной дисциплины «Базы данных» формируются следующие компетенции:

базовая профессиональная (для специальности 1 – 28 01 02 «Электронный маркетинг»):

БПК-13: Проектировать, создавать и администрировать информационные базы данных для информационного обеспечения программных комплексов и систем;

универсальные:

УК-1: Владеть основами исследовательской деятельности, осуществлять поиск, анализ и синтез информации;

УК-5: Обладать навыками саморазвития и совершенствования в профессиональной деятельности;

специализированные (для специальности 1 – 28 01 01 «Экономика электронного бизнеса»):

СК-8: Проектировать, создавать и администрировать информационные базы данных для информационного обеспечения программных комплексов и систем;

СК-9: Работать в пакетах прикладных программ, автоматизирующих бизнес-процессы электронного бизнеса.

В результате изучения учебной дисциплины «Базы данных» обучаемый должен:

знать:

- ✓ о представлениях экономической информации в автоматизированных информационных системах;
- ✓ организацию информационно-справочной работы с использованием БД;
- ✓ основные понятия баз данных;
- ✓ основы организации и функционирования баз данных;
- ✓ о концепциях моделирования данных, принципах организации баз данных и их проектирования;
- ✓ о назначении, архитектуре, функциональных возможностях современных систем управления базами данных (СУБД) и направлениях их развития;

- ✓ основные конструкции языка создания и манипулирования данными SQL;
- ✓ способы обеспечения безопасности данных;
- ✓ основы функционирования распределенных и многопользовательских баз данных;

уметь:

- ✓ строить информационную модель предметной области;
- ✓ проектировать базы данных реляционного типа;
- ✓ реализовать реляционную модель базы данных в используемой СУБД;
- ✓ использовать основные конструкции структурированного языка запросов;
- ✓ организовать ввод информации в БД, осуществлять поиск информации, проектировать формы, отчеты;
- ✓ сформулировать запросы к БД;
- ✓ работать с многопользовательской БД;
- ✓ профессионально эксплуатировать средства сетевых технологий;
- ✓ грамотно использовать глобальные информационные ресурсы.

владеть:

- ✓ методами, средствами и технологиями разработки информационных моделей и их программной реализации в выбранной СУБД;
- ✓ методами проектирования баз данных реляционного типа;
- ✓ теорией и стандартами языков описания и манипулирования данными, теоретическими и математическими основами построения выбранной модели данных;
- ✓ технологиями и техникой программной реализации баз данных, методами и языковыми средствами манипулирования данными, поддержания целостности, непротиворечивости и защиты информации;
- ✓ навыками макропрограммирования и управления объектами в среде СУБД;
- ✓ технологией использования распределенных баз данных в решениях прикладных задач.

Методика преподавания дисциплины «Базы данных» строится на сочетании лекций, лабораторных занятий, компьютерного тестирования и самостоятельной работы студентов, которая заключается в проработке материала лекционного курса, подготовке к лабораторным работам, выполнении домашних заданий. Чтение лекций предполагает использование мультимедийного проектора и компьютерной техники для наглядной демонстрации возможностей изучаемого программного продукта. В качестве основных методов обучения планируется использование сквозных прикладных задач, оформление отчетов по лабораторным работам и их защита.

Технической базой курса «Базы данных» является локальная вычислительная сеть университета, объединяющая все компьютерные залы. Важнейшим инструментом обучения являются имеющееся прикладное программное обеспечение (ресурсы сети) и электронное методическое обеспечение.

Освоение указанной дисциплины предполагает наличие у студентов базовых знаний, полученных в рамках изучения ряда учебных дисциплин: «Информационные технологии в маркетинге» *(для специальности 1 – 28 01 02*

«Электронный маркетинг»), «Информационные технологии в экономике» (*для специальности 1 – 28 01 01 «Экономика электронного бизнеса»),* «Основы алгоритмизации и программирования», «Основы объектно-ориентированного программирования», «Основы информационной безопасности».

В свою очередь, знания, полученные в ходе изучения дисциплины «Базы данных» могут быть востребованы при освоении последующих дисциплин: «Программирование сетевых приложений», «Проектирование информационных систем», «Распределенные системы обработки информации», «Бизнес-анализ и прототипирование программных продуктов» *(для специальности 1 – 28 01 02 «Электронный маркетинг»),* «Информационные системы корпоративного управления», применены при написании курсовых работ по специальным дисциплинам и выпускной дипломной работы, а также могут быть востребованы в будущей профессиональной деятельности.

По учебному плану для специальностей 1 – 28 01 02, 1 – 28 01 01 данной дисциплине для студентов очной формы образования отводится в 4-ом семестре 108 часов, в том числе 50 часов аудиторных занятий (16 часов лекций, 34 часа лабораторных занятий) и 58 часов неконтролируемой самостоятельной работы. Форма контроля – экзамен.

Код специальности (направления специальности)	Наименование специальности (направления специальности)	Курс	Семестр	Всего учебных часов	Количество зачетных единиц	Аудиторных часов (в соответствии с учебным планом УВО)					Академических часов на курсовой проект (работу)	Форма текущей аттестации
						Всего	Лекции	Лабораторные занятия	Практические занятия	Семинары		
1 – 28 01 01	Экономика электронного бизнеса	2	4	108	3	50	16	34				экзамен
1 – 28 01 02	Электронный маркетинг											

2. СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

2.1. Лекционные занятия, их содержание

2.1.1. Введение в теорию баз данных. Понятия, лежащие в основе концепции баз данных и систем управления базами данных.

Предмет и содержание дисциплины. Ее связь с другими дисциплинами.

Информация и данные. Экономическая информация. Виды и структурные единицы экономической информации. Определение и свойства системы. Экономические информационные системы, их функции, классификация. Состав информационного обеспечения, информационная база.

Определение базы данных (БД) и системы управления базами данных (СУБД). Преимущества базы данных. Роль баз данных и их сферы применения. Схема взаимодействия пользователей с БД. Понятия интегрированности и разделимости данных. Компоненты системы баз данных.

2.1.2. Жизненный цикл базы данных. Модели данных.

Определение жизненного цикла базы данных и его этапы. Концептуальное проектирование. Логическое проектирование. Физическое проектирование.

Сущность моделирования. Классификация моделей. Определение модели данных. Реализация модели данных. Концептуальная модель данных. Логические модели данных: иерархическая, сетевая, реляционная, объектная, объектно-ориентированная, объектно-реляционная. Их достоинства и недостатки. Физические модели данных.

2.1.3. Концептуальное проектирование БД. Операции реляционной алгебры. Типы связей между сущностями.

Определение и свойства сущности. Определение возможного ключа. Первичный ключ и альтернативные ключи. Синтаксис и семантика выражений реляционной алгебры. Теоретико-множественные реляционные операции объединения, пересечения, разности и декартова произведения. Примитивные и не примитивные реляционные операции.

Связь типа «один к одному». Связь типа «один ко многим». Внешний ключ и его свойства. Связь типа «многие ко многим». Преобразование связи «многие ко многим» в две связи «один ко многим» и связующую таблицу.

2.1.4. Логическое проектирование БД. Нормализация данных. Целостность данных.

Понятие нормализации данных. Приведение сущности к первой нормальной форме. Приведение сущности ко второй нормальной форме. Виды аномалий в данных, устраняемые после приведения ко второй нормальной форме. Приведение сущности к третьей нормальной форме.

Виды аномалий в данных, устраняемые после приведения к третьей нормальной форме. Нормальная форма Кодда-Бойса. Приведение сущности к четвертой нормальной форме. Приведение сущности к пятой нормальной форме.

Понятие целостности данных. Целостность таблиц. Целостность внешних ключей (ссылочная целостность). Основные и дополнительные правила ссылочной целостности. Целостность типов данных.

2.1.5. Назначение и функции СУБД. Физическая организация баз данных.

Физическая модель данных. Критерии выбора физической организации данных. Организация хранения данных. Представление реляционных данных. Модификация записей. Особенности представления объектов.

Понятие СУБД. Языковые и программные средства СУБД. Классификация СУБД. Классификация по универсальности, поддерживаемой модели данных, поддерживаемому режиму работы с базой данных. Механизмы доступа к данным из прикладных программ. Функциональные возможности СУБД. Возможности по созданию БД и ее актуализации, извлечению данных, созданию приложений базы данных, разработке приложений в СУБД, взаимодействию с другими системами, управлению базой данных. Производительность СУБД. Показатели производительности СУБД.

Режимы работы пользователя с СУБД. Работа через меню системы, в командном режиме, в программном режиме.

Направления развития СУБД. Расширение множества типов обрабатываемых данных. Интеграция технологий баз данных и Web-технологий.

2.1.6. Проектирование реляционных баз данных.

Инструментальные средства моделирования. Последовательность разработки БД в инструментальном средстве. Моделирование и описание сущностей. Установление связей между сущностями.

Диаграммы «сущность-связь»: модель уровня сущностей, модель данных, основанная на ключах, полная атрибутивная модель. CASE-средства для моделирования структур данных. Операции прямого и обратного проектирования. Синхронизация моделей данных.

2.1.7. Общая характеристика СУБД Ms Access.

Характеристики СУБД MS Access. Функциональные возможности СУБД Access. Характеристика БД и ее приложений, создаваемых в СУБД Access.

Пользовательский интерфейс СУБД Access. Система меню, панели инструментов, типы окон. Настройка рабочей среды в СУБД MS Access. Параметры настройки. Справочная система в СУБД. Виды справки.

Типы данных, обрабатываемых СУБД. Выражения в СУБД. Элементы выражения. Операторы. Инструментальные средства в СУБД для создания базы данных, ее приложений.

2.1.8. Создание и работа с БД посредством графического интерфейса СУБД. Технологии работы с базой данных в MS Access.

2.1.8.1. Технология создания базы данных в MS Access.

Создание файла БД в Access. Окно базы данных. Основные объекты Access. Способы их создания. Режимы работы с объектами.

2.1.8.2. Проектирование таблиц в MS Access. Создание связей между таблицами.

Таблица как объект БД. Способы создания таблиц. Создание структуры таблицы. Модификация структуры таблицы в режиме Конструктора. Свойства полей. Типы данных, обрабатываемых СУБД. Выражения в СУБД. Элементы выражения. Операторы.

Работа в режиме таблицы. Отбор данных с помощью фильтров. Способы задания фильтров. Сохранение фильтра как запроса.

Создание связей между таблицами. Типы отношений. Целостность данных. Схема данных.

2.1.8.3. Проектирование запросов в MS Access.

Понятие запроса к БД. Типы и возможности запросов. Способы создания запросов. Окно Конструктора запросов.

Формирование запросов выбора в режиме Конструктора. Условие отбора. Задание условия отбора для текстовых данных, даты, числовых данных. Задание сложного условия для выборки. Использование встроенных функций.

Проектирование параметрических запросов, запросов с вычисляемыми полями, итоговых и перекрестных запросов.

Запросы-действия, особенности их создания и выполнения.

Особенности создания многотабличных запросов. Виды объединения таблиц.

2.1.8.4. Проектирование форм в MS Access.

Форма как объект БД, её назначение. Способы проектирования форм. Элементы графического интерфейса формы. Ввод и редактирование данных с помощью форм.

Создание форм в режиме Конструктора. Структура формы. Свойства формы: макета, данных, событий. Настройка формы. Элементы управления (ЭУ), их виды и применение для форм и отчетов. Свойства элементов управления, их форматирование. Добавление элементов вычислений в форму. Работа с OLE-объектами в форме. Размещение кнопок управления в форме. Работа с базой данных по форме. Разработка многотабличной формы.

2.1.8.5. Проектирование отчетов в MS Access.

Отчет как объект БД, его назначение. Способы проектирования отчетов. Окно Конструктора отчетов. Группировка данных в отчетах. Добавление элементов с вычислениями в отчет. Изменение вида отчета.

2.1.8.6. Автоматизация работы с базой данных в MS Access. Управление базой данных.

Создание и запуск макросов. Использование макросов в формах и отчетах. Связь макроса с событием элемента управления.

Использование VBA для создания элементов приложений MS Access. Модуль как объект БД.

2.1.9. Введение в язык структурированных запросов SQL.

Назначение и общая характеристика языка SQL. Язык SQL в СУБД. Функциональные возможности языка SQL. Назначение, стандарты, достоинства.

Структура команды SQL. Типы данных. Константы. Преобразование данных. Операторы. Встроенные функции.

Команды определения данных и команды манипулирования данными в языке SQL. Предикаты. Логические связки NOT, AND, OR. Операторы IN, BETWEEN, LIKE, IS NULL. Выражения. Использование метасимволов «%» и «_» с оператором LIKE. Соглашения по записи форматов SQL-команд.

2.1.10. Организация данных с помощью языка SQL. Язык DDL.

Язык определения данных (DDL). Типы данных. Создание доменов. Создание реляционных таблиц. Синтаксис оператора CREATE TABLE.

Изменение структуры таблицы (добавление и удаление полей, создание и удаление индексов). Удаление таблицы.

Манипулирование данными. Ввод данных в таблицу. Обновление, удаление записей в таблице.

Операции соединения таблиц. Создание индексов, представлений.

2.1.11. Организация запросов в формате SQL. Язык DML.

Язык манипуляции данными (DML). Выборка данных. Запрос, синтаксис (аргументы) оператора SELECT. Задание условий выборки. Фильтрование данных с помощью предложения WHERE. Логические условия для построения условий выборки. Упорядочение набора данных с помощью ORDERBY. Сортировка строк, использование вычисляемых полей. Выражения, SQL-функции.

Подведение итогов с помощью функций агрегирования. Группировка строк и подсчет итоговых данных. Групповые функции. Предложения GROUPBY и HAVING.

Параметры в запросах. Особенности создания параметрических запросов.

Объединение таблиц в запросах. Внутреннее и внешнее соединение таблиц. Использование подзапросов. Вложенные подзапросы, коррелирующие подзапросы. Использование операторов ANY, ALL, EXISTS.

Выборка данных из нескольких таблиц. Реализация операций реляционной алгебры средствами языка SQL.

2.1.12. Процедурный SQL.

Элементы процедурного SQL (переменные, переменные определяемые пользователем, системные переменные, переменные в хранимой процедуре). Составной оператор BEGIN..END. Условные опреаторы. Оператор-селектор CASE. Циклы.

Хранимые процедуры и функции. Триггеры. Курсоры.

2.1.13. Администрирование баз данных. Распределенные транзакции и репликация данных. Безопасность данных.

Повышение производительности: индексирование, оптимизация запросов. Резервное копирование и восстановление данных. Полная архивная копия базы данных.

Удаленные запросы, распределенные транзакции. Проблемы параллельного выполнения транзакций. Назначение механизма репликации (тиражирования) данных.

Политика безопасности. Правила защиты БД. Средства защиты данных от несанкционированного доступа: идентификация, аутентификация, пользовательские роли, механизм привилегий. Модернизация программного обеспечения. Безопасный доступ к данным.

2.1.14. Клиент-серверные базы данных. Распределенные базы данных.

Модель взаимодействия отрытых систем. Классификация клиент-серверных систем. Клиент-серверные СУБД. Модели распределения функций.

Понятие распределенной базы данных. Предпосылки децентрализации. Система управления распределённой БД. Аспекты проектирования распределённых БД. Системы «клиент-сервер» как частный случай распределенных систем. Преимущества и недостатки распределённых БД. Классификация клиент-серверных систем.

2.1.15. Интерактивная аналитическая обработка OLAP. Многомерные базы данных и хранилища данных.

Требования к OLAP-инструментам. Понятие многомерной базы данных. Размещение информации и схема адресации. Назначение технологий разработки (извлечения) данных.

Хранилище данных. Особенности хранилищ данных по сравнению с операционными базами данных. Подготовка данных, предназначенных для хранения в хранилищах данных. Магазины (витрины данных). OLAP-куб.

2.2. Лабораторные занятия, их содержание

2.2.1. Проектирование реляционных баз данных.

Построение логической и физической моделей данных с помощью CASE-средств разработки информационных систем ERwin.

2.2.2. Работа в среде СУБД. Знакомство с возможностями MS Access. Проектирование таблиц.

Создание БД и ее объектов посредством графического интерфейса СУБД MS Access. Создание однотабличной базы данных. Обмен данными между приложениями. Способы создания таблиц. Создание и корректировка структуры таблицы в режиме Конструктора. Работа в режиме таблицы: поиск и замена

данных, сортировка записей, отбор данных с помощью фильтров, изменение вида таблицы.

2.2.3. Особенности проектирования многотабличных БД в MS Access. Создание связей.

Создание многотабличной индивидуальной базы данных. Создание таблиц. Свойства полей. Маска ввода. Построитель выражений.

Создание связей в БД. Заполнение таблиц данными.

2.2.4. Проектирование запросов в MS Access.

Конструирование запросов на выборку в режиме Конструктора с различными критериями отбора данных. Задание сложного условия для выборки. Создание запросов на основе других запросов.

Проектирование параметрических запросов.

Создание в запросах вычисляемых полей. Встроенные функции в Access. Построитель выражений.

Создание итоговых запросов. Особенности использования групповых установок и функций. Перекрестные запросы, способы их создания. Создание итогового столбца в перекрестном запросе.

Создание многотабличных запросов. Виды объединения таблиц.

Создание запросов на внесение изменений в БД: запрос на обновление данных; запрос на создание таблицы; запрос на добавление записей; запрос на удаление записей.

2.2.5. Проектирование форм и отчетов в MS Access.

Автоформа. Мастер создания форм. Создание форм в режиме Конструктора. Конструирование форм с различными объектами и элементами управления. Создание в форме вычисляемых полей. Конструирование составных форм. Работа с данными по форме.

Конструирование отчетов с вычислениями в строках и с общими итогами, а также отчетов с сортировкой и группировкой строк и с подведением итогов по группам.

2.2.6. Автоматизация работы с базой данных. Управление базой данных.

Конструирование и запуск макросов. Макросы. не связанные с событиями. Макросы, связанные с событиями ЭУ на форме или событием раздела формы или отчета. Использование макросов в формах и отчетах.

Использование VBA для создания элементов приложений MS Access.

Резервирование БД. Оптимизация БД. Восстановление поврежденной БД. Защита БД.

2.2.7. Организация данных с помощью языка SQL.

Создание реляционных таблиц. Изменение структуры таблицы (добавление и удаление полей, создание и удаление индексов). Удаление таблицы.

Освоение команд манипулирования данными. Ввод данных в таблицу. Обновление, удаление записей в таблице.

Операции соединения таблиц. Создание индексов, представлений.

2.2.8. Организация запросов в формате SQL.

Выборка данных таблицы по условию. Логические условия для построения условий выборки. Сортировка строк. Вычисления над полями таблицы.

Группировка строк, использование агрегатных функций.

Объединение таблиц в запросах. Вложенные подзапросы. Выборка данных из нескольких таблиц.

Управляющие запросы: создание, изменение, удаление таблиц.

2.2.9. Работа с программируемыми объектами SQL.

Разработка текста хранимых процедур и определяемых пользователем функций.

2.2.10. Освоение способов доступа к данным из прикладных программ.

Изучение основных принципов взаимодействия приложений, разработанных в архитектуре «клиент-сервер».

Изучение основных принципов работы клиентского приложения, осуществляющего доступ к базе данных по технологии ODBC (или другой технологии взаимодействия с базами данных).

2.3. Учебно-методическая карта учебной дисциплины

Дневная форма получения образования

Номер раздела, темы	***Название раздела, темы***	***лекции***	***лабораторные занятия***	***Самостоятельная работа***	***Форма контроля знаний***
1.	Введение в теорию баз данных. Понятия, лежащие в основе концепции баз данных и систем управления базами данных.	0,5		1	*Устный опрос, тест*
2.	Жизненный цикл базы данных. Модели данных.	0,5		2	*Устный опрос, тест*
3.	Концептуальное проектирование БД. Операции реляционной алгебры. Типы связей между сущностями.	1		3	*Устный опрос, тест*
4.	Логическое проектирование БД. Нормализация данных. Целостность данных.	2	4	6	*Отчет_ЛР*
5.	Назначение и функции СУБД. Физическая организация баз данных.	0,5	2	2	*Отчет_ЛР*
6.	Проектирование реляционных баз данных.	1	2	3	*Отчет_ЛР*
7.	Общая характеристика СУБД Ms Access.	0,5		2	*Устный опрос, тест*
8.	Создание и работа с БД посредством графического интерфейса СУБД. Технологии работы с базой данных в MS Access.	2	8	10	*Отчет_ЛР*
9.	Введение в язык структурированных запросов SQL.	1		2	*Устный опрос, тест*
10.	Организация данных с помощью языка SQL. Язык DDL.	2	4	6	*Отчет_ЛР*
11.	Организация запросов в формате SQL. Язык DML.	2	6	8	*Отчет_ЛР*
12.	Процедурный SQL.	1	4	5	*Отчет_ЛР*
13.	Администрирование баз данных. Распределенные транзакции и репликация данных. Безопасность данных.	1	2	3	*Отчет_ЛР*
14.	Клиент-серверные базы данных. Распределенные базы данных.	0,5	2	3	*Отчет_ЛР*
15.	Интерактивная аналитическая обработка OLAP. Многомерные БД и хранилища данных.	0,5		2	*Устный опрос, тест*
	Итого по дисциплине:	*16*	*34*	*58*	

Сокращения, использованные в учебно-методической карте:
Отчет_ЛР – Подготовка отчета по лабораторной работе и её защита.

3. ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

3.1. Перечень литературы (учебной, учебно-методической, научной, нормативной, др.)

3.1.1. Основная литература.

1. Базы данных: Учеб. для вузов / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев; Под ред. А. Д. Хомоненко. – СПб.: КОРОНА-Век, 2009. – 736 с.
2. Бекаревич, Ю. Б. Самоучитель Access 2010 / Ю. Б. Бекаревич, Н. В. Пушкина. – СПб. : БХВ-Петербург, 2011. – 432 с.
3. Быков, В. Л. Система автоматизированного проектирования баз данных Microsoft Access 2010: учебно-методическое пособие / В. Л. Быков, И. М. Гучко, А. М. Кулешова. – Брест : БрГТУ, 2014. – 75 с.
4. Грабер, М. Введение в SQL / М. Грабер. – Москва : Лори, 2019. – 378 с.
5. Грофф, Д. Р. SQL: полное руководство / Д. Р. Грофф, П. Н. Вайнберг, Э. Дж. Оппель. – Москва : Вильямс, 2018. – 960 с.
6. Гурвиц, Г. А. Microsoft Access 2010. Разработка приложений на реальном примере. / Г. А. Гурвиц. – СПб.: БХВ-СПб, 2010. – 469 с.
7. Гучко, И. М. Создание баз данных в среде СУБД Microsoft Access: пособие для самостоятельной работы студентов экономических специальностей / И. М. Гучко, Е. Н. Рубанова, И. Н. Аверина. – Брест : Изд-во БрГТУ, 2018. – 72 с.
8. Гучко, И. М., Рубанова, Е. Н. Методические указания по курсу "Компьютерные информационные технологии" 2-й раздел: "Технологии баз данных и знаний" для студентов экономических специальностей / Министерство образования Республики Беларусь, Брестский государственный технический университет, Кафедра информатики и прикладной математики ; сост.: И. М. Гучко, Е. Н. Рубанова. – Брест : БрГТУ, 2016. – 73 с.
9. Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт. – 8-е изд. – Москва : Диалектика, 2019. – 1328 с.
10. Диго, С. М. Базы данных: проектирование и использование: учебник / С. М. Диго. – Москва : Финансы и статистика, 2005. – 592 с.
11. Илюшечкин, В. М. Основы использования и проектирования баз данных : учебник для вузов / В. М. Илюшечкин. – Москва : Издательство Юрайт, 2021. – 213 с.– (Высшее образование). – ISBN 978-5-534-03617-6. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/468367 (дата обращения: 14.12.2022).
12. Кириллов, В. В. Введение в реляционные базы данных / В. В. Кириллов, Г. Ю. Громов. – Санкт-Петербург : БХВ-Петербург, 2012. – 464 с.
13. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. – Москва : Вильямс, 2017. – 1436 с.
14. Кренке, Д. Теория и практика построения баз данных. – СПб. : Питер, 2005. – 800 с.
15. Кузин, А. В. Базы данных: учеб. пособие / А. В. Кузин, С. В. Левонисова. – М. : Academia, 2012. – 320 с.

16. Кузин, А. В., Демин, В. М. Разработка баз данных в системе Microsoft Access: учебник. – М. : ИНФРА-М, 2014. –223 с.
17. Кузнецов, С. Д. Базы данных. Модели и языки: учебник / С. Д. Кузнецов. – Москва : Бином, 2008. – 720 с.
18. Левчук, Е. А. Технологии организации, хранения и обработки данных / Е. А. Левчук. – 3-е изд. – Минск : Выш. шк., 2007. – 239 с.
19. Малыхина, М. П. Базы данных: основы, проектирование, использование : учебное пособие / М. П. Малыхина. – 2-е изд. – Санкт-Петербург : БХВ–Петербург, 2006. – 528 с.
20. Марков, А. С. Базы данных: введение в теорию и методологию : учебник [рек. УМО РФ] / А. С. Марков, К. Ю. Лисовский. – Москва : Финансы и статистика, 2006. – 512 с.
21. Нестеров, С. А. Базы данных: учебник и практикум для СПО / С.А. Нестеров. – М. : Издательство Юрайт, 2019. – 230 с.
22. Одиночкина, С. В. Разработка баз данных в Microsoft Access 2010. СПб. : НИУ ИТМО, 2012. – 83 с.
23. Осипов, Д. Л. Технологии проектирования баз данных. – М. : ДМК Пресс, 2019. – 498 с.
24. Оскерко, В. С. Технологии баз данных: учеб. пособие/ В. С. Оскерко, З. В. Пунчик. Мн. : БГЭУ, 2015. – 215 с.
25. Редько, В. Н. Базы данных и информационные системы / В. Н. Редько, И. А. Бассараб. – Москва : Знание, 2011. – 602 с.
26. Сеннов, А. Access 2010. Учебный курс. – СПб.: Питер, 2010. – 288 с.
27. Смирнова, О. В. Access 2007 на практике. М.: Феникс, 2009. – 155 с.
28. Стружкин, Н. П. Базы данных: проектирование. Практикум: учеб. пособие для СПО / Н. П. Стружкин, В. В. Годин. – М. : Издательство Юрайт, 2019. – 291 с.
29. Стружкин, Н. П. Базы данных: проектирование. Учебник для академического бакалавриата / Н. П. Стружкин, В. В. Годин. – М. : Издательство Юрайт, 2019. – 477 с.
30. Сурядный, А. С. Microsoft Access 2010. Лучший самоучитель. – М. : Астрель, ВКТ, 2012. – 488 с.
31. Тарасов, С. В. СУБД для программиста. Базы данных изнутри / С. В. Тарасов. – Москва : СОЛОН-Пресс, 2015. – 320 с.
32. Тимошок, Т.В. Microsoft Office Access 2007: самоучитель / Т.В. Тимошок. – М. : Вильямс, 2008. – 464 с.
33. Туманов, В. Е. Основы проектирования реляционных баз данных / В. Е. Туманов. – Москва : Бином, 2012. – 420 с.
34. Федорова, Г. Разработка и администрирование баз данных / Г. Федорова. – Москва : Академия, 2015. – 313 с.
35. Харрингтон, Дж. Л. Проектирование реляционных баз данных. Пер. с англ. / Дж. Л. Харрингтон – М.: Лори, 2006. – 240 с.
36. Хомоненко, А. Д. Базы данных: Учеб. для вузов / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев; Под ред. А. Д. Хомоненко. – СПб.: КОРОНА-Век, 2009. – 736 с.

3.1.2. Дополнительная литература.

37. Наумов, А. Н. Системы управления базами данных и знаний / А. Н. Наумов, А. М. Вендров, В. К. Иванов. – Москва : Финансы и статистика, 2010. – 352 с.
38. Петров, В. Н. Информационные системы. СПб.: Питер, 2003. – 688 с.
39. Пирогов, В. Ю. Информационные системы и базы данных / В. Ю. Пирогов. – СПб. : БХВ-Петербург, 2009. – 528 с.
40. Роб, П., Коронел, К. Системы баз данных: проектирование, реализация и управление. – 5-е изд., перераб. и доп.: Пер. с англ. – СПб. : БХВ-Петербург, 2004. – 1040 с.
41. Советов, Б. Я. Базы данных: теория и практика : учебник для бакалавров / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. – Москва : Юрайт, 2013. – 463 с.
42. Хендерсон, К. Профессиональное руководство по SQL Server: структура и реализация / К. Хендерсон. – пер. с англ. – Москва : Вильямс, 2006. – 184 с.

3.2. Перечень средств диагностики результатов учебной деятельности

Контроль качества знаний по дисциплине «Базы данных» и средства диагностики устанавливаются УВО в соответствии с образовательным стандартом, нормативными документами Министерства образовании Республики Беларусь, а также методическими рекомендациями УМО. Для текущего контроля качества усвоения знаний рекомендуется использовать такой диагностический инструментарий, как устные опросы, проверка отчетов по лабораторным работам и тестовые задания. Оценка за ответы на устный опрос включает в себя полноту ответа, наличие аргументов, примеров из практики, актуальность исследуемой проблемы, корректность используемых методов исследования, привлечение знаний из различных областей и практикоориентированность полученных результатов.

Формой текущей аттестации по дисциплине «Базы данных» учебным планом предусмотрен экзамен. Итоговая оценка формируется на основе документов:

1. Правила проведения аттестации студентов, слушателей при освоении содержания образовательных программ высшего образования (Постановление Министерства образования Республики Беларусь от 29 мая 2012 г. № 53).
2. Положение «О внутрисеместровой аттестации студентов БрГТУ» № 11 от 30.01.2019 г.

Для диагностики результатов учебной деятельности используются следующие формы: устные опросы, защита лабораторных работ, собеседование при проведении индивидуальных и групповых консультаций и тестовые задания.

3.3. Методические рекомендации по организации и выполнению самостоятельной работы обучающихся по учебной дисциплине

Самостоятельная работа студентов по учебной дисциплине «Базы данных» приводятся в соответствии с п. 3 Положения о самостоятельной работе студентов учреждения образования «Брестский государственный технический университет», утвержденного ректором БрГТУ.

Самостоятельная работа предполагает углубленное изучение основной и дополнительной литературы, а также, выполнение лабораторных работ. Для организации самостоятельной работы студентов по учебной дисциплине, используются современные информационные технологии, размещен в сетевом доступе комплекс учебных и учебно-методических материалов (учебно-программные материалы, учебные издания для теоретического изучения дисциплины, методические указания к лабораторным работам, материалы текущего контроля и текущей аттестации, позволяющие определить соответствие учебной деятельности обучающихся требованиям образовательных стандартов высшего образования и учебно-программной документации, в т.ч. вопросы для подготовки к экзамену, задания, тесты, вопросы для самоконтроля, список рекомендуемой литературы).

На бесплатной платформе Google Classroom создается курс/класс, который позволяет делиться со студентами необходимыми материалами, позволяет предлагать и оценивать задания, планировать выполнение заданий, организовать общение студентов и преподавателя, проводить вебинары.

Также может использоваться один из облачных сервисов Google – Google формы. Сервис позволяет проводить опросы, проводить тестирования и викторины, получать обратную связь и проводить анализ результативности при проведении тестирования.

Эффективность самостоятельной работы студентов проверяется в ходе текущего и итогового контроля знаний.

При изучении учебной дисциплины рекомендуется использовать следующие формы самостоятельной работы:

- ✓ работа с технической литературой;
- ✓ составление конспектов;
- ✓ подготовка к выполнению лабораторных работ;
- ✓ составление отчетов по лабораторным работам;
- ✓ участие студентов в научно-исследовательской работе, проводимой на кафедре;
- ✓ участие студентов в конкурсах студенческих научных работ и студенческих научных конференциях;
- ✓ работа с методическими материалами.

Тема 1. Введение в теорию баз данных. Понятия, лежащие в основе концепции баз данных и систем управления базами данных.

Литература: [3.1_1], [3.1_3], [3.1_7] – [3.1_15], [3.1_17] – [3.1_21], [3.1_23] – [3.1_25], [3.1_28] – [3.1_29], [3.1_31], [3.1_33] – [3.1_41]

Тема 2. Жизненный цикл базы данных. Модели данных.

Литература: [3.1_1], [3.1_3], [3.1_8] – [3.1_15], [3.1_17] – [3.1_21], [3.1_23] – [3.1_25], [3.1_28] – [3.1_29], [3.1_33] – [3.1_41]

Тема 3. Концептуальное проектирование БД. Операции реляционной алгебры. Типы связей между сущностями.

Литература: [3.1_1] – [3.1_3], [3.1_8] – [3.1_15], [3.1_17] – [3.1_21], [3.1_23] – [3.1_25], [3.1_28] – [3.1_29], [3.1_33] – [3.1_41]

Тема 4. Логическое проектирование БД. Нормализация данных. Целостность данных.

Литература: [3.1_1] – [3.1_3], [3.1_6], [3.1_8] – [3.1_15], [3.1_17] – [3.1_21], [3.1_23] – [3.1_25], [3.1_28] – [3.1_29], [3.1_33] – [3.1_41]

Тема 5. Назначение и функции СУБД. Физическая организация баз данных.

Литература: [3.1_1] – [3.1_3], [3.1_8] – [3.1_16], [3.1_17] – [3.1_21], [3.1_24], [3.1_28] – [3.1_29], [3.1_31] – [3.1_35]

Тема 6. Проектирование реляционных баз данных.

Литература: [3.1_1], [3.1_3], [3.1_8] – [3.1_15], [3.1_17] – [3.1_21], [3.1_23] – [3.1_25], [3.1_28] – [3.1_29], [3.1_33] – [3.1_35], [3.1_38]

Тема 7. Общая характеристика СУБД Ms Access.

Литература: [3.1_1] – [3.1_3], [3.1_6] – [3.1_8], [3.1_15] – [3.1_16], [3.1_18], [3.1_22], [3.1_24], [3.1_26] – [3.1_27], [3.1_30], [3.1_32]

Тема 8. Создание и работа с БД посредством графического интерфейса СУБД. Технологии работы с базой данных в MS Access.

Литература: [3.1_1] – [3.1_3], [3.1_6] – [3.1_8], [3.1_15] – [3.1_16], [3.1_18], [3.1_22], [3.1_24], [3.1_26] – [3.1_27], [3.1_30], [3.1_32]

Тема 9. Введение в язык структурированных запросов SQL.

Литература: [3.1_4] – [3.1_5], [3.1_3], [3.1_6], [3.1_9] – [3.1_10], [3.1_12] – [3.1_13], [3.1_12] – [3.1_14], [3.1_17], [3.1_19] – [3.1_21], [3.1_23] – [3.1_24], [3.1_33] – [3.1_36], [3.1_42]

Тема 10. Организация данных с помощью языка SQL. Язык DDL.

Литература: [3.1_4] – [3.1_5], [3.1_3], [3.1_6], [3.1_9] – [3.1_10], [3.1_12] – [3.1_13], [3.1_12] – [3.1_14], [3.1_17], [3.1_19] – [3.1_21], [3.1_23] – [3.1_24], [3.1_33] – [3.1_36], [3.1_42]

Тема 11. Организация запросов в формате SQL. Язык DML.

Литература: [3.1_4] – [3.1_5], [3.1_3], [3.1_6], [3.1_9] – [3.1_10], [3.1_12] – [3.1_13], [3.1_12] – [3.1_14], [3.1_17], [3.1_19] – [3.1_21], [3.1_23] – [3.1_24], [3.1_33] – [3.1_36], [3.1_42]

Тема 12. Процедурный SQL.

Литература: [3.1_4] – [3.1_5], [3.1_3], [3.1_6], [3.1_9] – [3.1_10], [3.1_12] – [3.1_13], [3.1_12] – [3.1_14], [3.1_17], [3.1_19] – [3.1_21], [3.1_23], [3.1_34] – [3.1_36], [3.1_42]

Тема 13. Администрирование баз данных. Распределенные транзакции и репликация данных. Безопасность данных.

Литература: [3.1_9], [3.1_15], [3.1_19], [3.1_23], [3.1_36], [3.1_40]

Тема 14. Клиент-серверные базы данных. Распределенные базы данных.

Литература: [3.1_9], [3.1_15], [3.1_23], [3.1_40], [3.1_41]

Тема 15. Интерактивная аналитическая обработка OLAP. Многомерные БД и хранилища данных.

Литература: [3.1_13], [3.1_23], [3.1_34], [3.1_40], [3.1_41]

3.4. Программное обеспечение

3.3.1. Операционная система: MS Windows;
3.3.2. Пакет MS Office;
3.3.3. Реляционная СУБД: MS Access
3.3.4. Реляционная СУБД: MySQL, Microsoft SQL Server, SQLite *(или аналогичная);*
3.3.5. CASE-средство MS Visio, ERwin Data Modeler (или аналогичное);
3.3.6. Интернет-браузеры: Google Chrome, Mozilla Firefox;
3.3.7. Корпоративная платформа Google Workspace for Education.

3.4. Перечень вопросов к экзамену.

1. Основные понятия: информация, данные, экономическая информация. Виды и структурные единицы экономической информации.
2. Определение и свойства системы. Экономические информационные системы, их функции, классификация.
3. Состав информационного обеспечения, информационная база.
4. Основные понятия: данные, обработка данных, база данных (БД). Единица информации в БД. Информационная модель данных, её состав.
5. Определение базы данных. Преимущества базы данных. Роль баз данных и их сферы применения.
6. Схема взаимодействия пользователей с БД. Понятия интегрированности и разделимости данных. Компоненты системы баз данных.
7. Определение жизненного цикла базы данных и его этапы.
8. Концептуальное проектирование. Логическое проектирование. Физическое проектирование.
9. Сущность моделирования. Классификация моделей. Определение модели данных. Реализация модели данных.
10. Концептуальная модель данных.
11. Логические модели данных: иерархическая, сетевая, реляционная, объектная, объектно-ориентированная, объектно-реляционная. Их достоинства и недостатки.
12. Физические модели данных.
13. Определение системы управления базами данных (СУБД). Реляционные СУБД.
14. Определение и свойства сущности. Определение возможного ключа. Первичный ключ и альтернативные ключи.
15. Синтаксис и семантика выражений реляционной алгебры.
16. Теоретико-множественные реляционные операции объединения, пересечения, разности и декартова произведения. Примитивные и не примитивные реляционные операции.
17. Связь типа «один к одному».
18. Связь типа «один ко многим». Внешний ключ и его свойства.
19. Связь типа «многие ко многим». Преобразование связи «многие ко многим» в две связи «один ко многим» и связующую таблицу.
20. Понятие нормализации данных.
21. Приведение сущности к первой нормальной форме.
22. Приведение сущности ко второй нормальной форме. Виды аномалий в данных, устраняемые после приведения ко второй нормальной форме.
23. Приведение сущности к третьей нормальной форме. Виды аномалий в данных, устраняемые после приведения к третьей нормальной форме.
24. Нормальная форма Кодда-Бойса. Приведение сущности к четвертой нормальной форме.
25. Приведение сущности к пятой нормальной форме.
26. Понятие целостности данных. Целостность таблиц.
27. Целостность внешних ключей (ссылочная целостность).
28. Основные и дополнительные правила ссылочной целостности.

Целостность типов данных.

29. Общая характеристика СУБД Microsoft Access (MsA). Объекты MsA: их назначение и особенности, общие принципы и способы работы с ними.
30. Типы данных, обрабатываемые в MsA.
31. Таблица как объект БД. Понятие записи и поля таблицы. Ключевые поля в таблицах. Их виды. Назначение и использование.
32. Свойства полей: Маска ввода, Условие на значение, Значение по умолчанию. Примеры.
33. Отбор данных с помощью фильтров. Условие отбора. Оператор Like. Сохранение фильтра.
34. Схема данных в MsA. Условия и способы создания связей между таблицами. Межтабличные связи. Типы отношений между таблицами.
35. Целостность данных в MsA. Каскадные операции.
36. Запрос как объект БД в MsA. Виды запросов. Способы создания запроса в Microsoft Access. Условие отбора.
37. Формирование условий отбора: операторы сравнения, логические операторы.
38. Формирование условий отбора для полей с типом данных Дата/Время. Сложные критерии выборки.
39. Технология создания простых запросов в режиме Конструктора в MsA. Вид бланка QBE. Способы включения полей. Выполнение и сохранение запроса.
40. Запрос на выборку данных в MsA: назначение, виды. Параметрический запрос: назначения, особенности, преимущества.
41. Вычисляемое поле в вычисляемом запросе в MsA: правила и способы создания. Установка свойств полей запроса.
42. Использование встроенных функций в вычисляемом поле в MsA. Функции даты/времени. Примеры.
43. Использование встроенных функций в вычисляемом поле в MsA. Функция Format( ). Функция Iif( ). Примеры.
44. Создание итоговых запросов в MsA. Установки групповых операций. Групповые функции. Виды итоговых запросов.
45. Перекрестный запрос в MsA: назначение, правила и способы создания. Условия отбора и параметры в перекрестном запросе.
46. Технологии создания активных запросов в MsA: запрос на обновление данных; запрос на удаление данных, запрос на создание таблицы; запрос на добавление данных.
47. Форма как объект БД в MsA: назначение и возможности. Способы проектирования форм. Виды форм, создаваемые мастером.
48. Режим Конструктора форм в MsA. Области формы. Типы элементов управления, их виды и назначение.
49. Свойства и события формы и её объектов в MsA. Свойства элементов управления. Создание кнопок управления в форме.
50. Отчет как объект БД в MsA: назначение и возможности. Способы проектирования отчетов.
51. Окно Конструктора отчетов в MsA. Области (разделы) окна отчета. Типы элементов управления, их виды и назначение.

52. Макрос как объект БД в MsA: назначение и возможности. Классификация макрокоманд.
53. Понятие события в MsA. Категории и примеры событий.
54. Классификация макросов для обработки событий. Применение условий в макросе.
55. Модуль как объект БД в MsA. Visual Basic for Application (VBA): возможности, объекты и семейства языка.
56. Модули VBA: назначение, типы. Состав программного кода VBA.
57. Понятие и типы процедур в VBA. Синтаксис процедуры-функции и процедуры-подпрограммы.
58. Порядок создания функции пользователя в VBA. Использование пользовательской функции в запросах.
59. Управляющие конструкции языка VBA: Условный оператор If./.Then... Примеры.
60. Управляющие конструкции языка VBA: Оператор выбора Select Case. Примеры.
61. Назначение, функциональные возможности, достоинства, формы языка структурированных запросов SQL. Язык SQL в СУБД.
62. Структура SQL-команды. Типы данных SQL. Константы. Преобразование данных.
63. Категории операторов в SQL. Логические связки NOT, AND, OR.
64. Операторы IN, BETWEEN, LIKE, IS NULL.
65. Выражения. Использование метасимволов «%» и «_» с оператором LIKE.
66. Соглашения по записи форматов SQL-команд.
67. Определение данных в языке SQL. Конструкции языка SQL для организации данных в БД.
68. Создание доменов. Создание реляционных таблиц. Синтаксис оператора CREATE TABLE.
69. Изменение структуры таблицы (добавление и удаление полей, создание и удаление индексов). Удаление таблицы.
70. Манипулирование данными. Ввод данных в таблицу. Обновление, удаление записей в таблице.
71. Операции соединения таблиц. Создание индексов, представлений.
72. Манипулирование данными в языке SQL: конструкции языка.
73. Язык запросов к данным в SQL. Выборка данных. Синтаксис (аргументы) оператора SELECT.
74. Фильтрование данных с помощью предложения WHERE. Логические условия для построения условий выборки.
75. Упорядочение набора данных с помощью ORDERBY. Сортировка строк, использование вычисляемых полей. Выражения, SQL-функции.
76. Подведение итогов с помощью функций агрегирования. Группировка строк и подсчет итоговых данных. Групповые функции SQL. Предложения GROUPBY и HAVING.
77. Параметры в запросах. Особенности создания параметрических запросов.
78. Использование подзапросов. Вложенные подзапросы, коррелирующие подзапросы. Использование операторов ANY, ALL, EXISTS.
79. Выборка данных из нескольких таблиц. Реализация операций реляционной

алгебры средствами языка SQL.

80. Операции соединения таблиц в запросах. Внутреннее и внешнее соединение таблиц.
81. Процедурный SQL: назначение, основные программируемые объекты БД.
82. Элементы процедурного SQL (переменные, переменные определяемые пользователем, системные переменные, переменные в хранимой процедуре).
83. Составной оператор BEGIN..END. Условные операторы. Оператор-селектор CASE. Циклы.
84. Хранимые процедуры и функции. Триггеры. Курсоры.
85. Повышение производительности БД: индексирование, оптимизация запросов. Резервное копирование и восстановление данных. Полная архивная копия БД.
86. Удаленные запросы, распределенные транзакции. Проблемы параллельного выполнения транзакций.
87. Назначение механизма репликации (тиражирования) данных.
88. Политика безопасности. Правила защиты БД.
89. Средства защиты данных от несанкционированного доступа: идентификация, аутентификация, пользовательские роли, механизм привилегий. Безопасный доступ к данным.
90. Модель взаимодействия отрытых систем. Классификация клиент-серверных систем.
91. Клиент-серверные СУБД. Модели распределения функций.
92. Понятие и состав распределенной БД, аспекты её проектирования. Предпосылки децентрализации.
93. Система управления распределённой БД. Аспекты проектирования распределённых БД.
94. Системы «клиент-сервер» как частный случай распределенных систем. Преимущества и недостатки распределённых БД.
95. Классификация клиент-серверных систем.
96. Клиент-серверные базы данных. Примеры клиент-серверных СУБД.
97. Понятие многомерной базы данных. Размещение информации и схема адресации. Назначение технологий разработки (извлечения) данных.
98. Хранилище данных. Подготовка данных, предназначенных для хранения в хранилищах данных.
99. Особенности хранилищ данных по сравнению с операционными базами данных. Магазины (витрины данных).
100. Интерактивная аналитическая обработка OLAP. Требования к OLAP-инструментам. Основные типы OLAP-систем. OLAP-куб.

4. ПРОТОКОЛ СОГЛАСОВАНИЯ УЧЕБНОЙ ПРОГРАММЫ

Название дисциплины, с которой требуется согласование	***Название кафедры***	***Предложения об изменениях в содержании учебной программы***	***Решение, принятое кафедрой, разработавшей учебную программу (протокол, дата)***
✓ Бизнес-стратегии в сети Интернет ✓ Информационные ресурсы организации	Кафедра менеджмента		
✓ Интернет-маркетинг и электронная коммерция	Кафедра мировой экономики, маркетинга, инвестиций		

Содержание учебной программы
согласовано с выпускающими кафедрами

Заведующий кафедрой менеджмента,
кандидат экономических наук, доцент Гарчук И.М.

Заведующий кафедрой МЭМ,
кандидат технических наук, доцент Проровский А.Г.