

ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
Федеральное государственное образовательное бюджетное учреждение
высшего образования
«Поволжский государственный университет телекоммуникаций и информатики»
КОЛЛЕДЖ СВЯЗИ

ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ
УЧЕБНОЕ ПОСОБИЕ

для специальностей:

- 09.02.03 – Программирование в компьютерных системах
- 09.02.04 – Информационные системы (по отраслям)
- 09.02.06 – Сетевое и системное администрирование
- 09.02.07 – Информационные системы и программирование

Составил
преподаватель Терехин Д.В.

Самара
2019

Рассмотрено на заседании П(Ц)К

Утверждаю

«Информационные системы и технологии»

Зам. директора по УВР

Протокол №__ от _____ 2019г.

_____ Логвинов А.В.

Председатель П(Ц)К _____ Шомас Е.А.

«__» _____ 2019г.

Терехин Д.В. Основы проектирования баз данных, Учебное пособие., Самара КС ПГУТИ, 2019г. – 5.3 п. л.

СОДЕРЖАНИЕ

Раздел 1. Основы теории проектирования баз данных	4
Тема 1.1 Определение и назначение баз данных. Область применения баз данных	4
Тема 1.2 Логические модели баз данных. Типы взаимосвязей в моделях баз данных.....	6
Тема 1.3 Нормализация баз данных. Средства доступа к данным	10
Тема 1.4 Этапы проектирования баз данных. Системы управления базами данных.....	12
Раздел 2. Использование СУБД ACCESS для создания баз данных.	18
Тема 2.1 Основные характеристики и возможности СУБД ACCESS.....	18
Тема 2.1 Основные характеристики и возможности СУБД ACCESS.....	23
Тема 2.3 Индексирование: понятие индекса, типы индексных файлов	30
Тема 2.4 Создание, активация и удаление индекса. Переиндексирование	31
Тема 2.5 Обработка данных в базе	35
Тема 2.6 Изменение данных в БД средствами запроса. Установка и отмена фильтра.....	39
Раздел 3. Организация запросов SQL	41
Тема 3.1 Исторические аспекты. Структура и типы данных языка SQL	41
Тема 3.2 Операторы языка SQL	46
Раздел 4. Организация интерфейса с пользователем	57
Тема 4.1. Понятие объекта, свойства и характеристики объекта.	57
Тема 4.2. Хранимые процедуры и триггеры.	69
Тема 4.3. Формирование и вывод отчетов.....	74
Список использованных источников.....	85

Раздел 1. Основы теории проектирования баз данных

Тема 1.1 Определение и назначение баз данных. Область применения баз данных

База данных (БД) – это поименованная совокупность структурированных данных, относящихся к определенной предметной области. Под предметной областью будем понимать некоторую часть реального мира, информация о которой представлена в базе данных. Система управления базами данных

Система управления базами данных (СУБД) – это комплекс программных средств, предназначенных для создания и сопровождения базы данных. Можно выделить комплекс основных понятий, с помощью которых описываются принципы построения баз данных. Рассмотрим эти понятия.

Информационный объект – это описание некоторого реального объекта, явления, процесса, события в виде совокупности логически связанных атрибутов. Например, информационный объект СТУДЕНТ может быть представлен в базе данных с помощью следующих атрибутов: Код студента, Фамилия, Имя, Отчество, Год рождения. Это можно записать так:

СТУДЕНТ (Код студента, Фамилия, Имя, Отчество, Год рождения).

Между объектами предметной области могут существовать связи, имеющие различный содержательный смысл. Эти связи могут быть *обязательными* или *факультативными*.

Если вновь порожденный объект оказывается по необходимости связанным с каким-либо объектом предметной области, то между этими двумя объектами существует обязательная связь. В противном случае связь является факультативной (необязательной).

Совокупность объектов предметной области и связей между ними характеризует (типовую) структуру предметной области.

Множество объектов предметной области, значения атрибутов объектов и связи между ними могут изменяться во времени. Изменения могут сводиться к появлению новых или исключению из рассмотрения некоторых существующих объектов в предметной области, установлению новых или разрушению существующих связей между ними. Поэтому с каждым моментом времени можно сопоставить некоторое состояние предметной области.

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

Процесс создания информационной модели начинается с определения концептуальных требований будущих пользователей БД.

Требования отдельных пользователей интегрируются в едином «обобщенном представлении», которое называют **концептуальной моделью** данной предметной области.

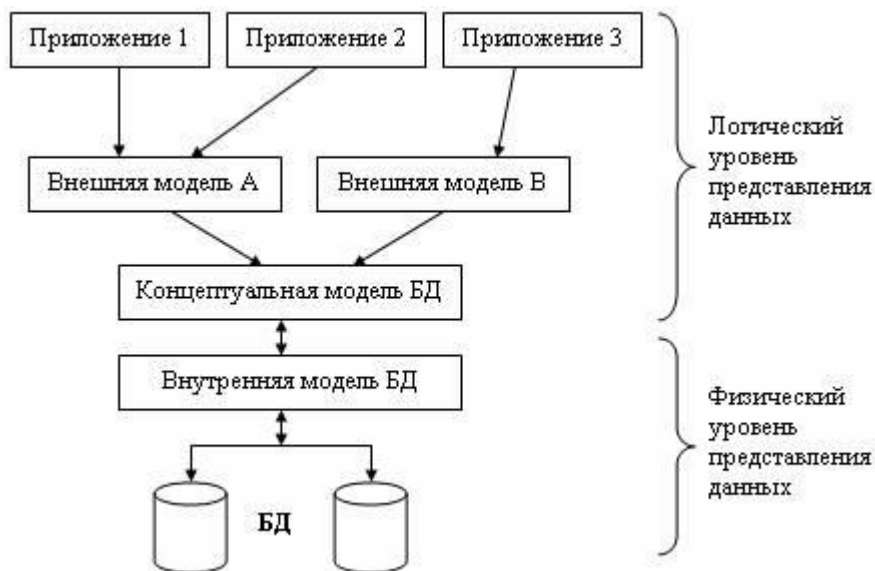


Рисунок 1.1 – Концептуальная модель БД

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

При этом усилия разработчика должны быть направлены в основном на структуризацию данных, принадлежащих будущим пользователям БД, и выявление взаимосвязей между ними.

Возможно, что отраженные в концептуальной модели взаимосвязи между объектами окажутся впоследствии нереализуемыми средствами выбранной СУБД. Это потребует изменения концептуальной модели. Версия концептуальной модели, которая может быть реализована конкретной СУБД, называется **логической моделью**.

Логическая модель отражает логические связи между атрибутами объектов вне зависимости от их содержания и среды хранения и может быть реляционной, иерархической или сетевой. Таким образом, логическая модель отображает логические связи между информационными данными в данной концептуальной модели.

Различным пользователям в информационной модели соответствуют различные подмножества ее логической модели, которые называются **внешними моделями** пользователей. Таким образом, внешняя модель пользователя представляет собой отображение концептуальных требований этого пользователя в логической модели и соответствует тем представлениям, которые пользователь получает о предметной области на основе логической модели. Следовательно, насколько хорошо спроектирована внешняя модель, настолько полно и точно информационная модель отображает предметную область и настолько полно и точно работает автоматизированная система управления этой предметной областью.

Логическая модель отображается в физическую память, которая может быть построена на электронных, магнитных, оптических, биологических или других принципах.

Внутренняя модель предметной области определяет размещение данных, методы доступа и технику индексирования в данной логи-ческой модели и иначе называется физической моделью.

Информационные данные любого пользователя в БД должны быть независимы от всех других пользователей, т. е. не должны оказывать влияния на существующие внешние модели. Это первый уровень независимости данных. С другой стороны, внешние модели пользователей никак не связаны с типом физической памяти, в которой будут храниться данные, и с физическими методами доступа к этим данным. Это положение отражает второй уровень независимости данных.

Тема 1.2 Логические модели баз данных. Типы взаимосвязей в моделях баз данных

Организация данных рассматривается с позиций той или иной модели данных. Модель данных является ядром любой базы данных. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

Модель данных – совокупность структур данных, ограничений целостности и операций манипулирования данными. Модели используются для представления данных в информационных системах.

Различают три типа моделей данных, которые имеют множества допустимых информационных конструкций:

- *иерархическая;*
- *сетевая;*
- *реляционная.*

Иерархическая модель данных

Иерархическая структура представляет совокупность элементов, связанных между собой по определенным правилам. Объекты, связанные иерархическими отношениями, образуют ориентированный граф (перевернутое дерево), вид которого представлен на рисунке 1.2:

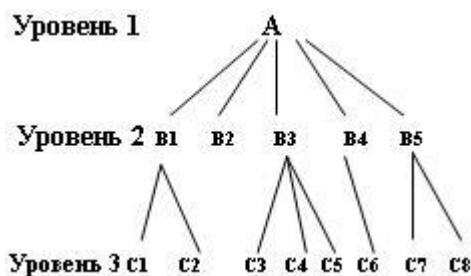


Рисунок 1.2 – Иерархическая модель БД

Основные понятия иерархической структуры

Это – узел, уровень и связь.

Узел – это совокупность атрибутов данных, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа. Каждый узел на более низком уровне **связан** только с одним узлом, находящимся на более высоком уровне.

Иерархическое дерево имеет только одну вершину (корень дерева), не подчиненную никакой другой вершине и находящуюся на самом верхнем (первом) **уровне**. Зависимые (подчиненные) узлы находятся на втором, третьем и т.д. уровнях. К каждой записи базы данных существует только один (иерархический) путь от корневой записи. Например, как видно из рисунка, для записи С4 путь проходит через записи В3 к А.

Пример иерархической структуры:

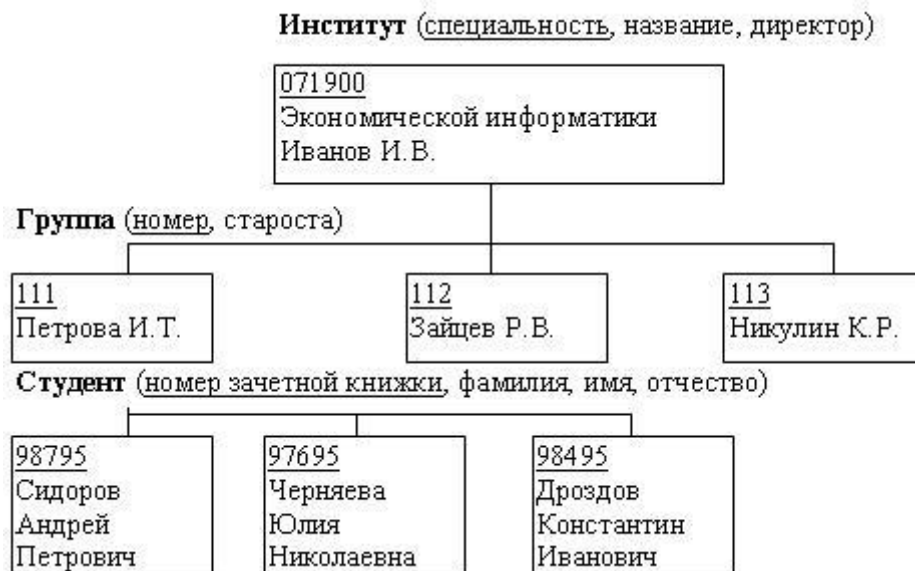


Рисунок 1.3 – Пример иерархической структуры

Сетевая модель данных

В сетевой структуре при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим элементом.

На рисунке 4 изображена сетевая структура базы данных в виде графа.

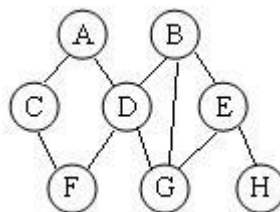


Рисунок 1.4 – сетевая структура БД

Пример сетевой структуры:



Рисунок 1.5 – Пример сетевой структуры БД

Примером сложной сетевой структуры может служить структура базы данных, содержащей сведения о студентах, участвующих в научно-исследовательских работах (НИРС). Возможно участие одного студента в нескольких НИРС, а также участие нескольких студентов в разработке одной НИРС. Графическое изображение описанной в примере сетевой структуры состоит только из двух типов записей.

Реляционная модель данных

Понятие *реляционный* (англ. *relation* – отношение) связано с разработками известного американского специалиста в области систем баз данных Е.Кодда.

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая *реляционная таблица* представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы – один элемент данных;
- все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
- каждый столбец имеет уникальное имя (заголовки столбцов являются названиями полей в записях);
- одинаковые строки в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.

Отношение – это плоская таблица, содержащая *N* столбцов, среди которых нет одинаковых. *N* – это *степень отношения*, или *арность* отношения. *Столбец* отношения соответствует атрибуту сущности. **Кортеж** – строка отношения (соответствует записи в таблице).

Пример реляционной модели

№ личного дела	Фамилия	Имя	Отчество	Дата рождения	Группа
16493	Сергеев	Петр	Михайлович	01.01.90	112
16593	Петрова	Анна	Владимировна	15.03.89	111
16693	Антохин	Андрей	Борисович	14.04.90	112

Отношения представлены в виде *таблиц*, строки которых соответствуют кортежам или *записям*, а столбцы – атрибутам отношений, доменам, *полям*.

Поле, каждое значение которого однозначно определяет соответствующую запись, называется *простым ключом* (ключевым полем).

Если записи однозначно определяются значениями нескольких полей, то такая таблица базы данных имеет *составной ключ*. В примере ключевым полем таблицы является "№ личного дела".

Давайте теперь рассмотрим то, как могут быть связаны таблицы в реляционных базах данных. Сразу скажу, что всего существует **три вида связей** между таблицами баз данных:

- связь один к одному;

- связь один ко многим;
- связь многие ко многим.

Рассмотрим, как такие связи между таблицами могут быть реализованы в реляционных базах данных.

Связь один к одному – самая редко встречаемая связь между таблицами. В 97 случаях из 100, если вы видите такую связь, вам необходимо объединить две таблицы в одну. Таблицы будут связаны один к одному тогда, когда одному объекту таблицы А соответствует один объект таблицы Б, и одному объекту таблицы Б соответствует один объект таблицы А.

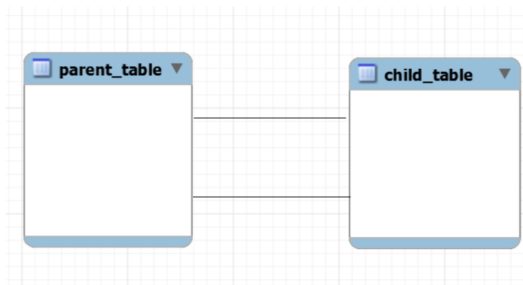
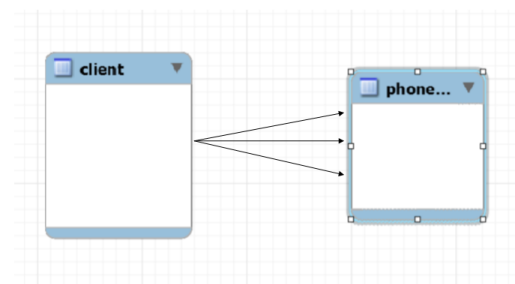
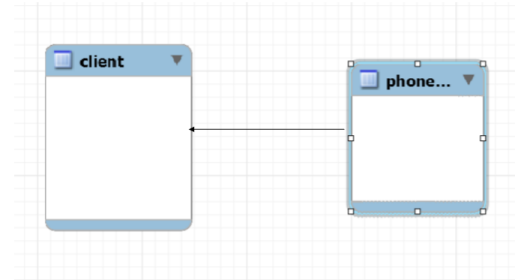


Рисунок 1.6 – Связь один к одному

Связь один ко многим в реляционных базах данных реализуется тогда, когда объекту А может принадлежать или же соответствовать несколько объектов Б, но объекту Б может соответствовать только один объект А.



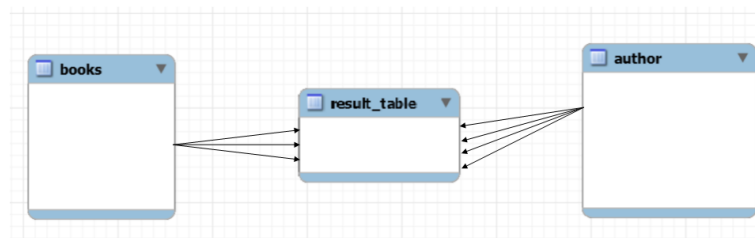
У одного клиента может быть несколько номеров



Но за номером закреплён лишь один клиент

Рисунок 1.7 – Связь один ко многим

Связь многие ко многим реализуется в том случае, когда несколькими объектами из таблицы А может соответствовать несколько объектов из таблицы Б, и в тоже время несколькими объектами из таблицы Б соответствует несколько объектов из таблицы А.



Одна книга могла быть написана несколькими авторами.
Автор мог написать несколько книг.

Рисунок 1.8 – Связь многие ко многим

Тема 1.3 Нормализация баз данных. Средства доступа к данным

Основными объектами любой базы данных являются таблицы. Таблицы базы данных создаются таким образом, чтобы каждая из них содержала информацию об одном информационном объекте. Между таблицами должны быть установлены реляционные связи. Установка таких связей делает возможным выполнение одновременной обработки данных из нескольких таблиц.

Ключ – это столбец (может быть несколько столбцов), добавляемый к таблице и позволяющий установить связь с записями в другой таблице. Существуют ключи двух типов: первичные и вторичные или внешние.

Первичный ключ реляционной таблицы – это поле или группа полей, которые позволяют однозначно определить каждую запись (строку) в таблице.

Первичный ключ должен обладать двумя свойствами:

- *однозначная идентификация записи* – запись должна однозначно определяться значением ключа;
- *отсутствие избыточности* – никакое поле нельзя удалить из ключа, не нарушая при этом свойства однозначной идентификации записи.

Для установки связей между таблицами используются *ключевые поля*. Можно связать две реляционные таблицы, если ключ одной связываемой таблицы ввести в состав ключа другой таблицы (возможно совпадение ключей).

Можно ключевое поле одной связываемой таблицы ввести в структуру другой таблицы так, что оно в этой таблице не будет ключевым. В этом случае это поле называется **внешним ключом**.

Внешний (вторичный) ключ - это одно или несколько полей (столбцов) в таблице, содержащих ссылку на поле или поля первичного ключа в другой таблице. Внешний ключ определяет способ объединения таблиц.

Из двух логически связанных таблиц одну называют таблицей первичного ключа или главной таблицей, а другую таблицей вторичного (внешнего) ключа или подчиненной таблицей. СУБД позволяют сопоставить родственные записи из обеих таблиц и совместно вывести их в форме, отчете или запросе.

Существует *три типа первичных ключей*: ключевые поля счетчика (счетчик), простой ключ и составной ключ.

- *Поле счетчика (Тип данных «Счетчик»)*. Тип данных поля в базе данных, в котором для каждой добавляемой в таблицу записи в поле автоматически заносится уникальное числовое значение.

- *Простой ключ*. Если поле содержит уникальные значения, такие как коды или инвентарные номера, то это поле можно определить, как первичный ключ. В качестве ключа можно определить любое поле, содержащее данные, если это поле не содержит повторяющиеся значения или значения Null.

- *Составной ключ*. В случаях, когда невозможно гарантировать уникальность значений каждого поля, существует возможность создать ключ, состоящий из нескольких полей. Чаще всего такая ситуация возникает для таблицы, используемой для связывания двух таблиц многие – ко – многим.

Необходимо еще раз отметить, что в поле первичного ключа должны быть только уникальные значения в каждой строке таблицы, т.е. совпадение не допускается, а в поле вторичного или внешнего ключа совпадение значений в строках таблицы допускается.

Если возникают затруднения с выбором подходящего типа первичного ключа, то в качестве ключа целесообразно выбрать поле счетчика.

Если в таблице существуют поля, значения каждого из которых однозначно определяют записи, то эти поля могут быть приняты в качестве **альтернативных ключей**. Например, если в качестве первичного ключа выбрать идентификационный номер налогоплательщика, то номер паспорта будет альтернативным ключом.

Каждому альтернативному ключу присваивается уникальный целый номер. Этот ключ указывается с помощью размещения справа от каждого атрибута ключа заключенных в скобки букв АК с номером альтернативного ключа, например, (АК1). Отдельный атрибут может входить в качестве компоненты в более чем один альтернативный ключ. Атрибут первичного ключа может служить также частью альтернативного ключа.

ПРИМЕР:

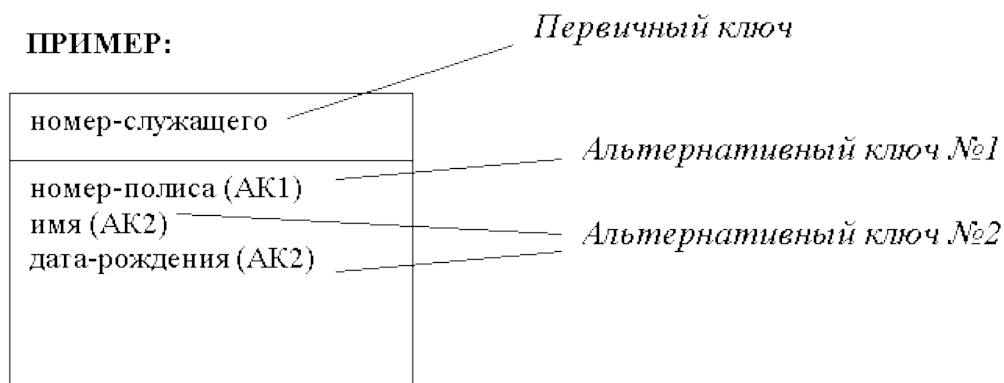


Рисунок 1.9 – Примеры ключей

Нормализация таблиц при проектировании БД

При проектировании структуры новой БД определяют сущности (объекты, явления) предметной области, которые должны найти свое отражение в базе данных. В конечном итоге анализ предметной области должен привести к созданию *эскиза* БД. Сначала желательно изобразить *сущности* и *связи* между ними. Как правило, каждой сущности в БД соответствует *таблица*. Затем - в эскизе второго порядка - для каждой таблицы БД приводится список *атрибутов* - *полей* записи.

На этом этапе процесс проектирования структур БД является процессом творческим, неоднозначным, с другой стороны, узловые его моменты могут быть формализованы.

Одной из таких формализаций является требование, согласно которому реляционная база данных должна быть нормализована. Процесс *нормализации* имеет своей целью устранение избыточности данных и заключается в приведении к третьей нормальной форме (3НФ).

Первая нормальная форма (1НФ) требует, чтобы каждое поле таблицы БД:

- было неделимым;
- не содержало повторяющихся групп.

Неделимость поля означает, что значение поля не должно делиться на более мелкие значения. Например, если в поле "Подразделение" содержится название факультета и название кафедры, требование неделимости не соблюдается и необходимо из данного поля выделить или название факультета, или кафедры в отдельное поле.

Повторяющимися являются поля, содержащие одинаковые по смыслу значения. Например, если требуется получить статистику сдачи экзаменов по предметам, можно создать поля для хранения данных об оценке по каждому предмету. Однако в этом случае мы имеем дело с повторяющимися группами.

Вторая нормальная форма (2НФ) требует, чтобы все поля таблицы зависели от первичного ключа, то есть, чтобы первичный ключ однозначно определял запись и не был избыточен. Те поля, которые зависят только от части первичного ключа, должны быть выделены в составе отдельных таблиц.

Третья нормальная форма (3НФ) требует, чтобы значение любого поля таблицы, не входящего в первичный ключ, не зависело от значения другого поля, не входящего в первичный ключ.

Тема 1.4 Этапы проектирования баз данных. Системы управления базами данных

В процессе разработки базы данных можно выделить несколько этапов.

Анализ предметной области. На этом этапе формируется задание по созданию базы данных. В нем подробно описывается состав базы данных, назначение и цели ее создания, а также

перечисляется, какие виды работ предполагается осуществлять в этой базе данных (отбор, изменение данных, печать или вывод отчета и т. д.).

Разработка схемы данных БД. На этом этапе рассматривается, из каких информационных объектов должна состоять база данных, какими атрибутами описывается каждый объект. Затем определяется структура реляционных таблиц с указанием свойств полей и связей между таблицами.

Синтез компьютерной модели объекта, предполагающий выполнение следующих типовых операций:

- 1) создание файла базы данных;
- 2) создание базовых таблиц;
- 3) создание экранных форм для ввода, редактирования и просмотра данных в таблицах;
- 4) заполнение таблиц данными;
- 5) работа с созданной базой данных:
 - сортировка, фильтрация и поиск записей в таблицах;
 - отбор данных из таблиц в соответствии с заданными критериями отбора;
 - выполнение обработки данных (удаление, добавление, изменение данных, выполнение вычислений);
 - подготовка отчетов.

Рассмотрим подробно **этап разработки схемы данных**. Разработку схемы данных обычно выполняют в следующей последовательности:

- 1) составляется общий список полей, отражающий атрибуты таблиц базы данных;
- 2) поля общего списка распределяются по базовым таблицам;
- 3) в соответствии со свойствами данных определяются свойства каждого поля;
- 4) в каждой таблице выделяется ключевое поле;
- 5) определяются связи между таблицами.

Проектирование базы данных на основе модели типа объект—отношение

Имеется целый ряд методик создания информационно-логических моделей. Одна из наиболее популярных в настоящее время методик при разработке моделей использует ERD (Entity-Relationship Diagrams).

В русскоязычной литературе эти диаграммы называют «объект — отношение» либо «сущность — связь». Модель ERD была предложена Питером Пин Шен Ченом в 1976 г. К настоящему времени разработано несколько ее разновидностей, но все они базируются на графических диаграммах, предложенных Ченом. Диаграммы конструируются из небольшого числа компонентов. Благодаря наглядности представления они широко используются в CASE-средствах (Computer Aided Software Engineering).

Рассмотрим используемую терминологию и обозначения.

Сущность (Entity) — реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению.

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа (сущности). Каждая сущность должна обладать некоторыми свойствами:

- иметь уникальное имя; причем к этому имени должна всегда применяться одна и та же интерпретация (определение сущности). И наоборот: одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются ею через связь;
- обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности.

Сущность может быть независимой либо зависимой. Признаком зависимой сущности служит наличие у нее наследуемых через связь атрибутов.

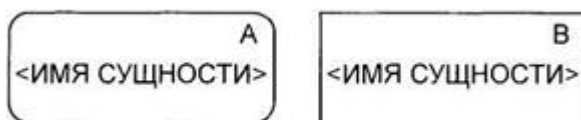


Рисунок 1.10 – Графическое обозначение сущностей: А – зависимая, В - независимая

Каждая сущность может обладать любым количеством связей с другими сущностями модели.

Связь (Relationship) — поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Одна из участвующих в связи сущностей — независимая, называется родительской сущностью, другая — зависимая, называется дочерней или сущностью-потомком.

Как правило, каждый экземпляр родительской сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров дочерней сущности.

Каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности-потомка может существовать только при существовании сущности-родителя.

Связи дается имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи. Имя каждой связи между двумя данными сущностями должно быть уникальным, но имена связей в модели не обязаны быть уникальными. Каждая связь имеет определение. Определение связи образуют соединением имени сущности-родителя, имени связи, выражения степени связи и имени сущности-потомка.

НОТАЦИЯ	ОБОЗНАЧЕНИЕ СВЯЗИ
IDEF1	—● Много
	—· Один
IE	—◀ Много
	—+ Один
Любая	- - - - - Необязательная
	— Обязательная

Рисунок 1.11 – Обозначение связей

Например, связь продавца с контрактом может быть определена следующим образом:

- продавец может получить вознаграждение за один или более контрактов;
- контракт должен быть инициирован ровно одним продавцом.

На диаграмме связь изображается отрезком (ломаной).

Концы отрезка с помощью специальных обозначений указывают степень связи. Кроме того, характер линии — штриховая или сплошная, указывает обязательность связи.

Атрибут — любая характеристика сущности, значимая для рассматриваемой предметной области. Он предназначен для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности.

#	<атрибут_1>	#	— ключевой атрибут
*	<атрибут_2>	*	— обязательный атрибут
o	<атрибут_3>	o	— необязательный атрибут

Рисунок 1.12 – Графическое отображение характеристики

Атрибут представляет тип характеристик (свойств), ассоциированных со множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, пар предметов и т. д.)

Экземпляр атрибута — это определенная характеристика конкретного экземпляра сущности. Экземпляр атрибута определяется типом характеристики (например, «Цвет») и ее значением (например, «лиловый»), называемым значением атрибута. В ER-модели атрибуты ассоциируются с конкретными сущностями. Каждый экземпляр сущности должен обладать одним конкретным значением для каждого своего атрибута.

Атрибут может быть либо обязательным, либо необязательным. Обязательность означает, что атрибут не может принимать неопределенных значений (null values). Атрибут может быть либо описательным (т. е. обычным дескриптором сущности), либо входить в состав уникального идентификатора (первичного ключа).

Уникальный идентификатор — это атрибут или совокупность атрибутов и/или связей, однозначно характеризующая каждый экземпляр данного типа сущности. В случае полной идентификации экземпляр данного типа сущности полностью идентифицируется своими собственными ключевыми атрибутами, в противном случае в идентификации участвуют также атрибуты другой сущности — родителя.



Рисунок 1.13 - Идентификация

Каждый атрибут идентифицируется уникальным именем, выражаемым грамматическим оборотом существительного, описывающим представляемую атрибутом характеристику. Атрибуты изображаются в виде списка имен внутри блока ассоциированной сущности, причем каждый атрибут занимает отдельную строку. Атрибуты, определяющие первичный ключ, размещаются наверху списка и выделяются знаком «#».

Каждая сущность должна обладать хотя бы одним возможным ключом. Возможный ключ сущности — это один или несколько атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. При существовании нескольких возможных ключей один из них обозначается в качестве первичного ключа, а остальные — как альтернативные ключи.

В настоящее время на основе подхода Чена создана методология IDEF1X, которая разработана с учетом таких требований, как простота изучения и возможность автоматизации. IDEF1X-диаграммы используются рядом распространенных CASE-средств (в частности, ERwin, Design/IDEF).

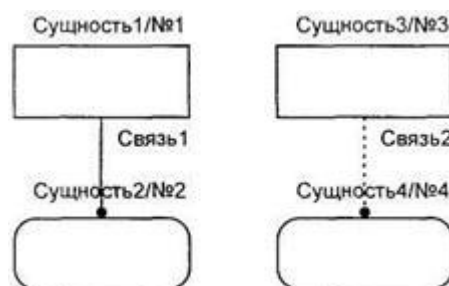


Рисунок 1.14 – Методология IDEF1X

Сущность в методологии IDEF1X называется независимой от идентификаторов или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности

Каждой сущности присваивается уникальное имя и номер, разделяемые косой чертой «/» и помещаемые над блоком.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется идентифицирующей, в противном случае — не идентифицирующей.

Идентифицирующая связь между сущностью-родителем и сущностью-потомком изображается сплошной линией

Штриховая линия изображает не идентифицируемую связь.

Сущность-потомок в не идентифицирующей связи будет независимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.

Связь может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которое может существовать для каждого экземпляра сущности-родителя). В IDEF1X могут быть выражены следующие мощности связей:

- каждый экземпляр сущности-родителя может иметь ноль, один или более связанных с ним экземпляров сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой.

Сущности могут иметь также внешние ключи (Foreign Key).

При идентифицирующей связи они используются в качестве части или целого первичного ключа, при не идентифицирующей — служат не ключевыми атрибутами.

В списке атрибутов внешний ключ отмечается буквами FK в скобках.

В результате получается информационно-логическая модель, которая используется рядом распространенных CASE-средств, таких, как ERwin, Design/IDEF. В свою очередь, CASE-технологии имеют высокие потенциальные возможности при разработке баз данных и информационных систем, а именно, увеличение производительности труда, улучшение качества программных продуктов, поддержка унифицированного и согласованного стиля работы.

Раздел 2. Использование СУБД ACCESS для создания баз данных

Тема 2.1 Основные характеристики и возможности СУБД ACCESS.

Основные требования к СУБД

1. Непротиворечивость данных. Не должно быть такой ситуации, когда заказывается отсутствующий на складе товар или в результате ошибки ввода информация о покупателе в заказе не соответствует данным картотеки покупателей. Такое требование называется требованием целостности. Целостность базы данных подразумевает поддержание полной, непротиворечивой и адекватно отражающей предметную область информации.

С требованием целостности данных связано понятие транзакции.

Транзакция — это последовательность операций над БД, рассматриваемых как единое целое (то есть или все, или ничего). Например, при оформлении заказа на определенный товар в системе нужно выполнить такие операции: регистрацию заказа и резервирование определенного количества товара, а также уменьшение данного товара на складе. Если на любом этапе изменения данных произойдет сбой, то целостность БД будет нарушена. Для предотвращения подобных нарушений вводится транзакция «Оформление заказа», в которой над БД либо должны произвестись все необходимые операции (товар продан, уменьшен его запас на складе), либо должен произойти возврат к исходному состоянию (товар не продан, его количество на складе не изменилось).

2. Актуальность хранимых данных. В любой момент времени информация, содержащаяся в БД, должна быть современной.

3. Многоаспектное использование данных - поступление информации из различных источников в единую БД и возможность ее использования любым отделом предприятия в соответствии с правами доступа и функциями.

4. Возможность модификации системы - возможность ее расширения и модификации данных, а также дополнение новыми функциями без ущерба для системы в целом.

5. Надежность - целостность БД не должна нарушаться при технических сбоях.

6. Скорость доступа - обеспечение быстрого доступа к требуемой информации.

СУБД осуществляют взаимодействие между БД и пользователями системы, а также между БД и прикладными программами, реализующими определенные функции обработки данных.

СУБД обеспечивают надежное хранение больших объемов данных сложной структуры во внешней памяти компьютера и эффективный доступ к ним.

К основным функциям СУБД относятся:

- непосредственное управление данными во внешней и оперативной памяти и обеспечение эффективного доступа к ним в процессе решения задач;
- поддержание целостности данных и управление транзакциями;

- ведение системного журнала изменений в БД для обеспечения восстановления БД после технического или программного сбоя;
- реализация поддержки языка описания данных и языка запросов;
- обеспечение безопасности данных;
- обеспечение параллельного доступа к данным нескольких пользователей.

СУБД Access, ее характеристика, возможности.

Приложение Microsoft Access – это настольная система управления реляционными базами данных (СУБД), предназначенная для работы на автономном персональном компьютере (ПК) или локальной вычислительной сети под управлением семейства операционных систем Microsoft Windows (Windows 2000, Windows XP и Windows Server 2003). СУБД Microsoft Access обладает мощными, удобными и гибкими средствами визуального проектирования объектов с помощью Мастеров, что позволяет пользователю при минимальной предварительной подготовке довольно быстро создать полноценную информационную систему на уровне таблиц, запросов, форм и отчетов.

К **основным возможностям** СУБД Microsoft Access можно отнести следующие:

- Проектирование базовых объектов – двумерные таблицы с полями разных типов данных.
- Создание связей между таблицами, с поддержкой целостности данных, каскадного обновления полей и каскадного удаления записей.
- Ввод, хранение, просмотр, сортировка, изменение и выборка данных из таблиц с использованием различных средств контроля информации, индексирования таблиц и аппарата алгебры логики.
- Создание, модификация и использование производных объектов (запросов, форм и отчетов).

Интерфейс пользователя MS Access – это комплекс программ, который реализует диалог в процессе работы пользователя с приложением Access.

Главное окно приложения Microsoft Access состоит из следующих областей:

- строка заголовка;
- строка меню;
- панель инструментов;
- окно базы данных;
- строка состояния.

К основным характеристикам СУБД Access относятся:

- графический многооконный интерфейс, позволяющий в диалоговом режиме создавать таблицы, формы, запросы, отчеты и макросы;

- наличие развитых диалоговых средств конструирования (таблиц, форм, запросов, отчетов, макросов);
- наличие специальных инструментальных средств, автоматизирующих работу, называемых мастерами;
- наличие средств соблюдения целостности данных на уровне базы данных;
- наличие средств обеспечения безопасности данных (защита паролем, ограничение уровня доступа и возможность работы в многопользовательских системах);
- использование объектной технологии OLE для внедрения в базу данных объектов различной природы (текстов, электронных таблиц, рисунков);
- наличие собственного языка программирования Visual Basic, объектно-ориентированного языка для разработки приложений пользователя, обладающего свойством автономности от СУБД и переносимости в другие приложения MS Office;
- полная поддержка языка запросов SQL;
- возможность интеграции данных из разных СУБД.

Приведем некоторые из средств Microsoft Access, существенно упрощающие разработку приложений.

1. Процедуры обработки событий и модули форм и отчетов. На встроенном языке VBA можно писать процедуры обработки событий, возникающих в формах и отчетах. Процедуры обработки событий хранятся в модулях, связанных с конкретными формами и отчетами, в результате чего код становится частью макета формы или отчета. Кроме того, существует возможность вызова функции VBA свойством события.

2. Свойства, определяемые в процессе выполнения. С помощью макроса или процедуры обработки событий можно определить практически любое свойство формы или отчета в процессе выполнения в ответ на возникновение события в форме или отчете.

3. Модель событий. Модель событий, похожая на используемую в языке Microsoft Visual Basic, позволяет приложениям реагировать на возникновение различных событий, например, нажатие клавиши на клавиатуре, перемещение мыши или истечение определенного интервала времени.

4. Использование обработки данных с помощью VBA. С помощью языка VBA можно определять и обрабатывать различные объекты, в том числе, таблицы, запросы, поля, индексы, связи, формы, отчеты и элементы управления.

5. Построитель меню. Предназначен для помощи при создании специальных меню в приложениях. Кроме того, специальные меню могут содержать подменю.

6. Улучшенные средства отладки. Помимо установки точек прерывания и пошагового выполнения программ на языке VBA, можно вывести на экран список всех активных процедур. Для

этого следует выбрать команду Вызовы в меню Вид или нажать кнопку [Вызовы] на панели инструментов.

7. Процедура обработки ошибок. Помимо традиционных способов обработки ошибок возможно использование процедуры обработки события Errоr для перехвата ошибок при выполнении программ и макросов.

8. Улучшенный интерфейс защиты. Команды и окна диалога защиты упрощают процедуру защиты и смены владельца объекта.

9. Программная поддержка механизма OLE. С помощью механизма OLE можно обрабатывать объекты из других приложений.

10. Программы-надстройки. С помощью VBA можно создавать программы-надстройки, например, нестандартные мастера и построители. Мастер — средство Microsoft Access, которое сначала задает пользователю вопросы, а затем создает объект (таблицу, запрос, форму, отчет и т.д.) в соответствии с его указаниями.

Диспетчер надстроек существенно упрощает процедуру установки программ-надстроек в Microsoft Access.

Мастера Access

Access позволяет даже малоподготовленному пользователю создать свою БД, обрабатывать данные с помощью форм, запросов и отчетов, проводить анализ таблиц БД и выполнять ряд других работ. Практически для любых работ с БД в Access имеется свой мастер, который помогает их выполнять.

- **Мастер по анализу таблиц** позволяет повысить эффективность базы данных за счет нормализации данных. Он разделяет ненормализованную таблицу на две или несколько таблиц меньшего размера, в которых данные сохраняются без повторения.

- **Мастера по созданию форм и отчетов** упрощают и ускоряют процесс создания многотабличных форм и отчетов. Новые форма и отчет могут наследовать примененный к таблице-источнику записей фильтр. Мастера по разработке форм и отчетов автоматически создают инструкцию SQL, определяющую источник записей для формы или отчета, поэтому отпадает необходимость в создании запроса.

Для изменения вида формы, отчета или отдельных элементов может быть использован мастер, вызываемый кнопкой [Автоформат].

- **Мастер подстановок** создает в поле таблицы раскрывающийся список значений из другой таблицы для выбора и ввода нужного значения. Для создания такого поля со списком достаточно в режиме конструктора таблицы выбрать тип данных этого поля — Мастер подстановок. Мастер подстановок можно вызвать в режиме таблицы командой меню Вставка\Столбец

подстановок. Созданный в данном поле таблицы список наследуется при включении этого поля в форму.

- **Мастера по импорту/экспорту** позволяют просматривать данные при импорте/экспорте текста или электронных таблиц, а также при экспорте данных Microsoft Access в текстовые файлы.
- **Мастер защиты** при необходимости эвакуирует данные, для чего создает новую базу данных, копирует в нее все объекты из исходной базы данных, снимает все права, присвоенные членам группы пользователей, и шифрует новую базу данных. После завершения работы мастера администратор может присвоить новые права доступа пользователям и группам пользователей.
- **Мастер по разделению базы данных** позволяет разделить ее на два файла, в первый из которых помещаются таблицы, а во второй — запросы, формы, отчеты, макросы и модули. При этом пользователи, работающие в сети, имея общий источник данных, смогут устраивать формы, отчеты и другие объекты, используемые для обработки данных, по своему усмотрению.

Типы данных СУБД

При создании структуры таблицы необходимо определить тип полей, которые будут в ней использоваться. Чтобы задать тип данных, которые будут храниться в определенном поле, необходимо щелкнуть, находясь в режиме *Конструктора* таблиц, левой кнопкой мыши в столбце *Тип данных* соответствующей строки, после чего выбрать в списке требуемый тип.

В таблице приведены все имеющиеся в Access типы данных.

Название типа	Описание	Объем данных
Текстовый	Текст – одна строка	До 255 байт
Поле МЕМО	Текст – несколько строк	До 65535
Числовой	Число	1,2,4 или 8 байт
Дата время	Дата и/или время	8 байт
Денежный	Денежные суммы (числа с четырьмя десятичными знаками)	8 байт
Счетчик	Уникальное целое число, обозначающее порядковый номер записи в таблице	4 байта
Логический	Логические значения	1 бит
Поле объекта OLE	Объекты других приложений Windows	До 1 Гбайт
Гиперссылка	Ссылка на файл (в данном компьютере, в локальной сети, в Интернет)	До 2048 символов

Ниже приведено описание основных типов данных, используемых в таблицах СУБД Access.

- **Текстовый.** Используется для хранения строковых данных с длиной до 255 символов.
- **Поле МЕМО.** Применяется для хранения текста, превышающего 255 символов и/или состоящего из нескольких строк (т.е. текста, в котором встречается символ возврат каретки).
- **Числовой.** Используется для обработки чисел различного размера и формата.

- **Дата/время.** Применяется при обработке значений календарных дат и времени. Этот тип данных позволяет хранить значения в нескольких форматах. По умолчанию устанавливается Полный формат даты.
- **Денежный.** Используется для хранения денежных значений и предусматривает точность до 4 знаков после десятичной запятой и до 15 знаков перед запятой.
- **Счетчик.** Применяется для описания поля, используемого в качестве первичного ключа, и генерирует для каждой новой записи в таблице уникальное значение формата Длинное целое.
- **Логический.** Предназначен для хранения и обработки логических значений, т.е. Ложь или Истина.

Тема 2.1 Основные характеристики и возможности СУБД ACCESS

При открытии Access откроется вкладка **создать**. На вкладке "Создание" можно создать новую базу данных несколькими способами.

- **Пустая база данных.** Если хотите, вы можете начать с нуля. Это хороший вариант, если у вас есть особые требования к проектированию или у вас есть необходимые данные, которые нужно разместить или добавить.
- **Шаблон, установленный в Access** Вы можете использовать шаблон, если вы запускаете новый проект и хотите начать с головного офиса. Access сопровождается несколькими шаблонами, установленными по умолчанию.
- **Шаблон с сайта Office.com.** В дополнение к шаблонам, поставляемым с Access, много других шаблонов доступно на сайте Office.com. Для их использования даже не нужно открывать браузер, потому что эти шаблоны доступны на вкладке **Создать**.

Добавление объектов в базу данных

При работе с базой данных в нее можно добавлять поля, таблицы и части приложения.

Части приложения — это функция, позволяющая использовать несколько связанных объектов базы данных как один объект. Например, часть приложения может состоять из таблицы и формы, основанной на ней. С помощью части приложения можно одновременно добавить в базу данных таблицу и форму.

Также можно создавать запросы, формы, отчеты, макросы — любые объекты базы данных, необходимые для работы.

Создание базы данных с помощью шаблона

В Access есть разнообразные шаблоны, которые можно использовать как есть или в качестве отправной точки. *Шаблон* — это готовая к использованию база данных, содержащая все таблицы, запросы, формы, макросы и отчеты, необходимые для выполнения определенной задачи. Например, существуют шаблоны, которые можно использовать для отслеживания вопросов, управления

контактами или учета расходов. Некоторые шаблоны содержат примеры записей, демонстрирующие их использование.

Если один из этих шаблонов вам подходит, с его помощью обычно проще и быстрее всего создать необходимую базу данных. Однако если необходимо импортировать в Access данные из другой программы, возможно, будет проще создать базу данных без использования шаблона. Так как в шаблонах уже определена структура данных, на изменение существующих данных в соответствии с этой структурой может потребоваться много времени.

1. Если база данных открыта, нажмите на вкладке **Файл** кнопку **Заккрыть**. Откроется вкладка **Создать**.

2. На вкладке **Создать** доступно несколько наборов шаблонов.

3. Выберите шаблон, который вы хотите использовать.

4. Приложение Access предложит имя файла базы данных в поле **имя файла** — при желании вы можете изменить имя файла. Выберите соответствующую папку, куда вы сохраните свою базу данных.

5. Нажмите кнопку **Создать**.

Access создаст базу данных на основе выбранного шаблона, а затем откроет ее. Для многих шаблонов при этом отображается форма, в которую можно начать вводить данные. Если шаблон содержит примеры данных, вы можете удалить каждую из этих записей, щелкнув область маркировки (затененное поле или полосу слева от записи) и выполнив действия, указанные ниже.

На вкладке **Главная** в группе **Записи** нажмите кнопку **Удалить**.

6. Щелкните первую пустую ячейку в форме и приступайте к вводу данных. Для открытия других необходимых форм или отчетов используйте область навигации. Некоторые шаблоны содержат форму навигации, которая позволяет перемещаться между разными объектами базы данных.

Создание базы данных без использования шаблона

Если вы не заинтересованы в использовании шаблона, вы можете создать базу данных, создав собственные таблицы, формы, отчеты и другие объекты базы данных. В большинстве случаев это состоит из следующих элементов:

- Ввести, вставить или импортировать данные в таблицу, которая создается вместе с базой данных, и повторить эту процедуру для новых таблиц, которые создаются с помощью команды **Таблица** на вкладке **Создание**.

- Импортировать данные из других источников, при этом создав таблицы.

Создание пустой базы данных

1. На вкладке **Файл** щелкните **Создать** и выберите вариант **Пустая база данных**.

2. В поле **Имя файла** введите имя файла. Чтобы сохранить файл в другой папке, отличной от используемой по умолчанию, нажмите кнопку **Поиск расположения для размещения базы данных** (рядом с полем **Имя файла**), откройте нужную папку и нажмите кнопку **ОК**.

3. Щелкните **Создать**.

Access создаст базу данных с пустой таблицей "Таблица1" и откроет ее в режиме таблицы. Курсор будет помещен в первую пустую ячейку столбца **Щелкните для добавления**.

4. Чтобы добавить данные, начните вводить их или вставьте из другого источника.

Ввод данных в режиме таблицы — это очень похоже на работу на листе Excel. При вводе данных будет создана структура таблицы. При добавлении нового столбца в таблицу в таблице определяется новое поле. Microsoft Access автоматически задает тип данных каждого поля в зависимости от введенных данных.

Если на этом этапе вводить данные в таблицу "Таблица1" не нужно, нажмите кнопку **Заккрыть**. Если вы внесли изменения в таблицу, будет предложено сохранить их. Нажмите кнопку **Да**, чтобы сохранить изменения, кнопку **Нет**, чтобы не сохранять их, или кнопку **Отмена**, чтобы оставить таблицу открытой.

Добавление таблицы

Добавление таблиц к существующей базе данных осуществляется командами группы **Таблицы** на вкладке **Создать**.

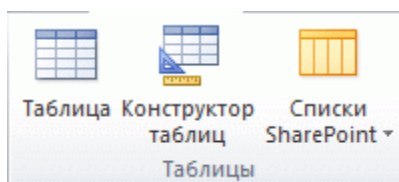


Рисунок 2.1 – Создание таблиц

Создание таблицы в режиме таблицы. В режиме таблицы можно начать ввод данных сразу, структура таблицы при этом будет создаваться автоматически. Полям присваиваются имена с последовательными номерами ("Поле1", "Поле2" и т. д.), а тип данных автоматически задается с учетом вводимых данных.

1. На вкладке **Создание** в группе **Таблицы** нажмите кнопку **Таблица**.

Access создаст таблицу и выделит первую пустую ячейку в столбце **Щелкните для добавления**.

2. На вкладке **Поля** в группе **Добавление и удаление** выберите нужный тип поля. Если нужный тип поля не отображается, нажмите кнопку **Другие поля**.

3. Откроется список часто используемых типов полей. Выберите необходимый тип поля, и Access добавит в таблицу новое поле в точке вставки.

Поле можно переместить путем перетаскивания. При этом в таблице появляется вертикальная полоса вставки, указывающая место, где будет расположено поле.

4. Чтобы добавить данные, начните вводить их в первую пустую ячейку или вставьте из другого источника.

5. Для переименования столбца (поля) дважды щелкните его заголовок и введите новое название.

Присвойте полям значимые имена, чтобы при просмотре области **Список полей** было понятно, что содержится в каждом поле.

6. Чтобы переместить столбец, щелкните его заголовок для выделения столбца и перетащите столбец в нужное место. Можно выделить несколько смежных столбцов и перетащить их одновременно. Чтобы выделить несколько смежных столбцов, щелкните заголовок первого столбца, а затем, удерживая нажатой клавишу SHIFT, щелкните заголовок последнего столбца.

Создание таблицы в режиме конструктора. В режиме конструктора сначала следует создать структуру таблицы. Затем можно переключиться в режим таблицы для ввода данных или ввести данные, используя другой способ, например, вставить данные из буфера обмена или импортировать их.

1. На вкладке **Создание** в группе **Таблицы** нажмите кнопку **Конструктор таблиц**.

2. Для каждого поля в таблице введите имя в столбце **Имя поля**, а затем в списке **Тип данных** выберите тип данных.

3. При желании можно ввести описание для каждого поля в столбце **Описание**. Это описание будет отображаться в строке состояния, когда в режиме таблицы курсор будет находиться в данном поле. Описание также отображается в строке состояния для любых элементов управления в форме или отчете, которые создаются путем перетаскивания этого поля из области **списка полей**, и любых элементов управления, которые создаются для этого поля при использовании мастера отчетов или мастера форм.

4. Когда все необходимые поля будут добавлены, сохраните таблицу:

- На вкладке **Файл** щелкните **Сохранить**.

5. Вы можете в любой момент ввести данные в таблицу, переключив ее в режим таблицы и щелкнув первую пустую ячейку.

Задание свойств полей в режиме конструктора.

Независимо от способа создания таблицы рекомендуется проверить и задать свойства полей. Хотя некоторые свойства доступны в режиме таблицы, другие можно настроить только в режиме конструктора. Чтобы перейти в режим конструктора, в области навигации щелкните таблицу правой кнопкой мыши и выберите пункт **Конструктор**. Чтобы отобразить свойства поля, щелкните его в сетке конструктора. Свойства отображаются под сеткой конструктора в области **Свойства поля**.

Щелкните свойство поля, чтобы просмотреть его описание рядом со списком **Свойства поля**. Более подробные сведения можно получить, нажав кнопку справки.

В следующей таблице описаны некоторые наиболее часто изменяемые свойства полей.


Свойство	Описание
Размер поля	Для текстовых полей это свойство указывает максимально допустимое количество знаков, сохраняемых в поле. Максимальное значение 255. Для числовых полей это свойство определяет тип сохраняемых чисел («Длинное целое», «Двойное с плавающей точкой» и т.д.). Для более рационального хранения данных рекомендуется выделять для хранения данных наименьший необходимый размер памяти. Если требуется, это значение позже можно изменить.
Формат поля	Это свойство определяет формат отображения данных. Оно не влияет на фактические данные, сохраняемые в этом поле. Вы можете выбрать встроенный формат или задать собственный.
Маска ввода	Это свойство используется для определения общего шаблона для ввода любых данных в поле. Это позволяет обеспечить правильный ввод и нужное количество знаков для всех данных. Для получения справки по созданию маски ввода нажмите кнопку  справа от поля свойства.
Значение по умолчанию	Это свойство позволяет задать стандартное значение, которое будет отображаться в этом поле при добавлении новой записи. Например, для поля Дата/время в котором необходимо записывать дату добавления записи, в качестве значения по умолчанию можно ввести “ Date() ”(без кавычек)
Обязательное поле	Это свойство указывает, обязательно ли вводить значение в поле. Если для него задано значение Да , невозможно будет добавить запись, если в это поле не введено значение.

Схема данных в Access

СУБД MS Access позволяет **строить схему базы** данных и автоматически поддерживает первичные и внешние ключи.

Схема базы данных (или просто схема данных) является графическим образом БД. В ней определяются и запоминаются связи между таблицами. Это позволяет Access автоматически использовать связи при конструировании форм, запросов, отчетов. Схема данных отображается в специальном окне Схема данных, где таблицы представлены списками полей, а связи - линиями между полями в связанных таблицах.

Создать схему данных можно двумя способами:

1. Первый способ предполагает наличие *нормализованных таблиц*, связывание которых осуществляется в окне Схема данных. Создание схемы данных начинается с размещения в окне всех таблиц, которые должны быть включены в схему. Далее можно приступать к определению попарных связей между ними.

Устанавливая связи между парой таблиц, надо выделить в главной таблице уникальное ключевое поле (ПК в каждой таблице отображается жирным шрифтом), по которому устанавливается связь. Далее при нажатой левой кнопке мыши это поле перетаскивается на соответствующее поле подчиненной таблицы. Если устанавливается связь по составному ключу, то

необходимо выделить все поля, входящие в составной ключ главной таблицы, и перетащить их на одно из полей связи в подчиненной таблице.

После установления связи открывается диалоговое окно Изменение связи, в котором для ключевого поля главной таблицы определяется поле связи подчиненной таблицы. Для каждого поля составного ключа главной таблицы связь с полем подчиненной таблицы должна быть установлена отдельной строкой. Кроме того, в окне Изменение связи для каждой связи можно задать параметр Обеспечение целостности данных, после чего устанавливаются опции Каскадное обновление связанных полей и Каскадное удаление связанных записей. При этом Access автоматически установит тип связи **1:M** (в схеме обозначается как 1:∞). Если таблицы содержат данные, не отвечающие требованиям целостности, связь **1:M** не будет установлена, и Access в этом случае выводит соответствующее сообщение.

2. Второй способ - получение схемы данных из одной ненормализованной таблицы с помощью Мастера анализа таблиц.

Ненормализованные таблицы содержат поля с повторяющимися значениями. Для таких таблиц расходуется дополнительная память, не может быть обеспечен однократный ввод и корректировка данных в полях с повторяющимися значениями, велика вероятность возникновения ошибок.

С помощью Мастера анализа таблиц Access позволяет автоматически или вручную выполнить анализ таблицы и разделить ее на несколько взаимосвязанных таблиц, в которых данные не будут дублироваться. Для полученных таблиц Мастер анализа таблиц создает схему данных, в которой устанавливаются связи 1:M и определяются параметры обеспечения целостности. После процедуры анализа для исходной таблицы будут получены нормализованные взаимосвязанные таблицы.

Однако Мастер анализа таблиц далеко не во всех случаях может правильно выделить повторяющиеся данные в отдельную таблицу. При работе с Мастером предусматривается интерактивное участие пользователя в корректировке предложений по разделению таблицы.

Модификация структуры базы данных

Модификация самой структуры БД включает в себя возможности расщепления, объединения таблиц; изменения первичных ключей и структуры связей между таблицами.

1. Изменение первичных ключей базы данных.

При работе с базой может возникнуть необходимость изменения первичных ключей некоторых таблиц. Так, если бы мы в таблице "Преподаватели" определили первичный ключ по полю "Фамилия", он бы не всегда однозначно идентифицировал записи в таблице, т.к. в таблице может быть несколько преподавателей с фамилиями "Петров", "Сидоров" и т.д.

Выходом из этой ситуации является или выбор "более уникального" поля, или же вводится поле типа "Счетчик", значение которого никогда не повторяется. В нашей таблице таким полем является "Номер_Преподавателя", поскольку у каждого преподавателя свой уникальный номер.

Переопределение первичного ключа производится или сбросом определения ключа в описании индексов (команда "Индекс" меню "Вид" выводит список индексов БД, в ней активизируем строку "Первичный ключ" для поля, являющегося первичным ключом, и нажимаем кнопку "Del" клавиатуры); а затем установив новый первичный ключ (командой "определить ключ" меню "Правка"). Либо же активизацией поля будущего первичного ключа, и щелчком на кнопке "Первичный ключ" панели инструментов.

Для определения ключа можно выделить несколько полей одновременно. Для этого удерживаем нажатой клавишу "Ctrl" клавиатуры и щелкаем по области маркировки требуемых строк. При невозможности переопределения выводится информационное окно Access.

2. Модификация структуры связей базы данных

Возможности определения и модификации структуры связей базы данных были рассмотрены в предыдущей лабораторной работе, и особенных затруднений не представляют.

Сложная структура связей базы, особенно с множественными связями, при установленных требованиях каскадного обновления и удаления, связанных полей, приводит к замедлению работы Access. Поэтому после создания базы и какого-то периода ее эксплуатации рекомендуется пересмотреть структуру таблиц и их связей в целях оптимизации.

Из видов модификации базы данных иногда может потребоваться расщепление одной таблицы на несколько, или же объединение группы таблиц в одну. рассмотрим их применение по отношению к базе STUD.

3. Расщепление таблиц

Расщепление таблиц может потребоваться в том случае, если какая-то из таблиц проекта содержит редко используемую группу полей. Например, в

запросах используются основные признаки объекта, а более детальная информация бывает востребована редко: в итогах, сводках, и т.д.

Имеет смысл расщепить такую таблицу на две или более частей, связав их по какому-либо признаку объекта.

Также повышает производительность системы расщепление таблицы, если к части данных необходимо ограничить доступ пользователей. К конфиденциальным данным можно отнести информацию о адреса заказчиков, зарплате сотрудников, деталях контрактов и т.д. В том случае закрытая информация обособливается в отдельные таблицы, и доступ к ним пользователей ограничивается.

Можно использовать присоединение данных через сеть, ограничив доступ пользователей к данным средствами сетевой ОС.

Простейший метод расщепления таблицы - создание ее копии, удаление из обеих таблиц лишних полей; настройка связей – и занесение модифицированных таблиц обратно в проект.

Тема 2.3 Индексирование: понятие индекса, типы индексных файлов

Если записей в таблице много, то найти нужную запись бывает очень трудно. Поиск данных производится методом перебора, то есть просматриваются все записи таблицы от первой записи до последней записи, что приводит к большим затратам времени. Чтобы облегчить поиск данных в таблице, используют индексы.

Индекс, иногда его называют указатель, представляет собой порядковый номер записи в таблице. Индекс строится по значениям одного поля или по значениям нескольких полей.

Индекс, построенный по значениям одного поля, называется **простым**, а по значениям двух и более полей — **сложным**. Во время построения индекса записи в таблице сортируются по значениям поля (или полей) будущего индекса. Затем первой строке таблицы присваивается индекс номер один, второй строке — индекс номер два и т. д. до конца таблицы.

Как простой, так и сложный индекс имеют свой тип (Type). Первичный (Primary) индекс (ключ) — это поле или группа полей, однозначно определяющих запись, то есть значения первичного индекса уникальны (не повторяются). В реляционной базе данных каждая таблица может иметь только один первичный ключ. Внешних ключей у таблицы может быть много, и они будут иметь один из типов:

- **Candidate** — кандидат в первичный ключ или альтернативный ключ. Он обладает всеми свойствами первичного ключа.
- **Unique** (уникальный) — допускает повторяющиеся значения в поле, по которому он построен, но на экран будет выводиться только одна первая запись из группы записей с одинаковым значением индексного поля.
- **Regular** (регулярный) — не накладывает никаких ограничений на значения индексного поля и на вывод записей на экран. Индекс только управляет порядком отображения записей. Это наиболее популярный тип индекса.

Взаимосвязь между таблицами осуществляется по индексам, которые называют **ключами**.

Построенный индекс хранится в специальном индексном файле. Если индексный файл хранит только один индекс, то он называется одноиндексным и имеет расширение .idx. Индексные файлы, которые хранят много индексов, называются мультииндексными и имеют расширение .cdx. Каждый индекс, который хранится в мультииндексном файле, называется тегом. Каждый тег имеет свое уникальное имя.

Мультииндексные файлы бывают двух типов: просто мультииндексные файлы (о них рассказано выше) и структурные мультииндексные файлы. Структурный мультииндексный файл имеет одинаковое имя с таблицей, которой он принадлежит (отличие только в расширении файла), и обладает следующими свойствами:

- автоматически открывается со своей таблицей;
- его нельзя закрыть, но можно сделать не главным.

Одна таблица может иметь много индексных файлов как одноиндексных, так и мультииндексных. В старших версиях FoxPro используются мультииндексные файлы.

Тема 2.4 Создание, активация и удаление индекса. Переиндексирование

Создание индекса

Создать индекс можно двумя способами.

1. С помощью команды:

```
INDEX ON <индексное выражение> TO <idx-файл> | TAG <имя тега> [OF <cdx-файл>]  
[FOR <условие>]  
[COMPACT]  
[DESCENDING]  
[UNIQUE]  
[ADDITIVE]  
[NOOPTIMIZE]
```

Назначение опций:

<индексное выражение> — имя поля (или полей), по значениям которого надо построить индекс. При построении сложного индекса имена полей перечисляются через знак + (плюс). Если сложный индекс построен по:

- числовым полям, то индекс строится по сумме значений полей;
- символьным полям, то индекс строится сначала по значению первого поля, а при повторяющихся значениях первого поля — по значениям второго поля; при повторяющихся значениях первого и второго полей — по значениям третьего поля и т. д.;
- по полям разных типов, то сначала значения полей приводят к одному типу, как правило символьному, а затем строят индекс.

Длина индексного выражения не должна превышать 254 символа.

TO <idx-файл> — указывается имя одноиндексного файла.

TAG <имя тега> [OF <cdx-файл>] — указывается имя тега в мультииндексном файле. Если используется опция *[OF <cdx-файл>]*, то создаваемый тег помещается в указанный мультииндексный файл, а если требуемый мультииндексный файл отсутствует, то будет построен

структурный мультииндексный файл. Если опция [OF <cdx-файл>] опущена, то созданный тег будет помещен в текущий мультииндексный файл.

FOR <условие>— устанавливает режим отбора в индекс тех записей таблицы, которые удовлетворяют <условию>.

COMPACT— управляет созданием компактного одноиндексного файла. В старших версиях FoxPro не используется.

DESCENDING— строит индекс по убыванию. По умолчанию используется построение индекса по возрастанию (ASCENDING). Для одноиндексных файлов можно построить индекс только по возрастанию. Если перед использованием команды INDEX ON... подать команду SET COLLATE, то можно построить одноиндексный файл по убыванию.

UNIQUE— строит уникальный индекс. Если индексное поле (поля) содержит повторяющиеся значения, то в индекс попадает только одна первая запись и остальные записи будут не доступны.

ADDITIVE— вновь создаваемый индексный файл не закрывает уже открытые к этому моменту времени индексные файлы. Если опция опущена, то вновь создаваемый индексный файл закрывает все ранее открытые индексные файлы.

2. С помощью Главного меню:

В этом случае индекс создается либо при создании таблицы, либо при модификации структуры таблицы. Для этого в диалоговой панели Table Designer надо выбрать вкладку Index.

Каждый индекс описывается одной строкой в окне диалоговой панели Table Designer.

В графе *Name* указывается имя тега мультииндексного файла. Если ранее открыт один из мультииндексных файлов, то вновь построенный индекс помещается в открытый мультииндексный файл. Если индекс строится одновременно с созданием табличного файла или табличный файл не имеет мультииндексных файлов, то вновь построенный индекс помещается в автоматически создаваемый структурный мультииндексный файл.

В графе *Type*, снабженной раскрывающимся списком, указывается один из допустимых типов индекса. Если индекс строится для таблицы, входящей в состав базы данных, то возможны четыре значения: Primary, Candidate, Unique и Regular. Если индекс строится для свободной таблицы, то в раскрывающемся списке отсутствует значение Primary.

В графе *Expression* перечисляются имена полей, по значениям которых надо построить индекс. Если строится сложный индекс, то удобнее воспользоваться строителем выражений, который запускается нажатием кнопки, расположенной справа от поля ввода.

В графе *Filter* можно задать логическое условие и построить индекс не для всех записей таблицы, а только для записей, удовлетворяющих условию фильтра. Эта графа также снабжена строителем выражений. Содержание и внешний вид обоих строителей одинаковые.

В раскрывающемся списке *String* приведены допустимые строковые функции. Аналогично в раскрывающихся списках *Math*, *Logical* и *Date* приведены допустимые математические, логические функции и функции даты. Нужная функция из этих раскрывающихся списков выбирается щелчком левой кнопки мыши. Имена полей (список *Fields*) и имена переменных (список *Variables*) выбираются с помощью двойного щелчка левой кнопки мыши. Получившееся в результате выражение помещается в окно *Expression*.

В раскрывающемся списке *From Table* указано имя таблицы, из которой берутся поля для построения индекса. При желании можно заказать любую таблицу из текущей базы данных и для построения индекса взять любое поле.

Открытие индексного файла

Открыть индексный файл можно только в том случае, если ранее открыт соответствующий табличный файл. В противном случае будет выдано сообщение об ошибке. Для открытия индексного файла надо подать команду:

```
SET INDEX TO [список индексных файлов]  
[ORDER <выр.N> | <idx-файл>][TAG<имя тега>][OP<cdx-файл>]]  
[ASCENDING | DESCENDING]  
[ADDITIVE]
```

Назначение опций такое же, как в командах **USE** и **INDEX ON**.

Открыть существующий индексный файл можно одновременно с открытием табличного файла командой **USE** (см. п. 2.2 «Открытие таблицы»).

Для закрытия всех индексных файлов надо подать одну из команд: либо **SET INDEX TO** без опций, либо **CLOSE INDEX**.

Замена текущего индекса

Для каждой таблицы одновременно может быть открыто несколько индексных файлов, но текущим (активным) будет только один индекс. По умолчанию принято, что текущим будет первый по порядку индекс в том индексном файле, имя которого указано первым в списке имен индексных файлов команды **USE** или команды **SET INDEX TO**.

Текущим можно сделать любой индекс из текущего индексного файла с помощью команды **SET ORDER TO**

```
[<выр.N1> | <idx-файл> | [TAG <имя тега> [OF <cdx-файл>]]  
[IN <выр.N2> | <выр.C ]  
[ASCENDING | DESCENDING]
```

Назначение опций:

<выр.N1>— задает текущий индекс по его порядковому номеру в мультииндексном файле.

<idx - файл>— делает текущим одноиндексный файл.

TAG <имя тега> [OF <cdx - файл>]— задает текущий индекс по имени тега из указанного мультииндексного файла. Если опция [OF <cdx-файл>] опущена, то тег выбирается из текущего мультииндексного файла.

IN <выр.N2>— указывает номер рабочей области, в которой находится индексный файл. Опция используется в том случае, если табличный файл открыт в одной рабочей области, а индексный файл — в другой рабочей области.

Текущим индекс также можно сделать с помощью диалоговой панели Table Designer, переместив строку описания нужного индекса на первое место.

Перестройка индексных файлов

При внесении изменений в большие таблицы тратится много времени, так как при внесении каждого изменения заново перестраиваются все открытые индексные файлы. Для экономии времени индексные файлы закрывают и вносят изменения в таблицу. Однако в этом случае возникает несоответствие между обновленной таблицей и индексными файлами. Для устранения указанного несоответствия надо заново перестроить индексные файлы. После открытия всех индексных файлов, принадлежащих измененному табличному файлу, надо подать команду **REINDEX**. Команда действует на все индексные файлы, открытые в текущей рабочей области. **Переиндексирование** можно также выполнить, подав из Главного меню команду **Table -> Rebuild Indexes**.

Преобразование одноиндексного файла в тег

Если табличному файлу принадлежит один или несколько одно-индексных файлов, то их можно скопировать как теги в мульти-индексный файл. Для этих целей используют команду

COPY INDEXES <имена idx - файлов> | ALL [TO <cdx - файл>]

Опция ALL указывается в том случае, если надо скопировать все одноиндексные файлы. При этом список имен <имена idx-файлов> не указывается. Тегам присваиваются имена одноиндексных файлов. При копировании нескольких одноиндексных файлов их имена перечисляются через запятую. Если опция TO опущена, то одноиндексные файлы копируются в текущий мультииндексный файл. Если опция TO содержит имя несуществующего мультииндексного файла, то он создается.

Допустима и обратная операция, то есть один тег преобразуется (копируется) в одноиндексный файл с помощью команды:

COPY TAG <список имен тегов> {OF <cdx-файл>} TO <idx-файл>]

Предварительно мульти индексный файл должен быть открыт. Можно скопировать отдельные теги, указав <список имен тегов>, либо все теги, используя опцию ALL.

Тема 2.5 Обработка данных в базе

Запросы создаются пользователем для выборки необходимых ему данных из одной или нескольких связанных таблиц и представления выбранных данных также в виде таблицы. Запрос может формироваться двумя способами:

- с помощью запросов по образцу (QBE — Query By Example);
- с помощью инструкций SQL (Structured Query Language — язык структурированных запросов). SQL — специализированный язык, предназначенный для организации запросов, обновления и управления реляционными базами данных.

Практически все типы запросов в Access можно создать визуально. Исключение составляют сквозные запросы (SQL pass-through) — это запросы из других приложений, запросы на изменение структуры данных и запросы объединения.

Также визуально можно построить запросы добавления, удаления, обновления и создания таблиц.

Отметим также, что одной из наиболее сильных сторон Access являются фильтры. Фильтры строятся с помощью запросов или установкой критериев. Для облегчения этой задачи используют параметрические запросы.

В Access может быть создано несколько видов запросов:

- **запрос на выборку**— выбирает данные из взаимосвязанных таблиц и других запросов. Результатом его является таблица, которая существует до закрытия запроса. Таблицу с результатами запроса можно использовать для работы с данными в таблицах, на которых построен запрос;
- **запрос на создание таблицы**— основан на запросе выборки, но в отличие от него результат запроса сохраняется в новой таблице;
- **запросы на обновление, добавление, удаление**— являются запросами действия, в результате выполнения которых изменяются данные в таблицах.

Основные принципы конструирования запроса заложены в технике конструирования **запроса на выборку**, являющегося основой всех видов запросов.

Запрос на выборку позволяет достаточно просто выбрать данные из одной или нескольких взаимосвязанных таблиц. Результаты запроса отображаются в виде таблицы.

При конструировании запроса достаточно выделить и перетащить необходимые поля из таблиц, представленных в схеме данных запроса, в бланк запроса и ввести условия отбора записей.

Результаты выполнения запроса выводятся в режиме таблицы. Несмотря на то, что поля результирующей таблицы принадлежат, как правило, нескольким таблицам базы данных, с ними можно работать так, как если бы они принадлежали одной таблице. Можно менять данные в таблице результатов запроса на выборку и сделанные изменения будут внесены в базовые таблицы. Для создания запроса надо выбрать в окне базы данных закладку **Запрос** и нажать кнопку **Создать**.

В открывшемся окне **Новый запрос** из предложенных вам типов запросов **конструктор, простой запрос, перекрестный запрос, повторяющиеся записи, записи без подчиненных**, выбрать **конструктор**. В окне **Добавление таблицы** выбрать используемые в запросе таблицы и нажать кнопку **Добавить**. Затем кнопкой **Заккрыть** выйти из окна **Добавление таблицы**. В результате появится окно конструктора запросов — **<Имя запроса>: запрос на выборку**.

Окно конструктора запросов разделено на две панели. Верхняя панель содержит схему данных запроса, которая включает выбранные для данного запроса таблицы. Таблицы представлены списками полей. Нижняя панель является бланком запроса по образцу — QBE, который нужно заполнить.

Условия отбора записей, сортировка и фильтрация данных

Сортировка – это упорядочение записей таблицы (списка) по определенному столбцу в порядке возрастания или убывания. Для выполнения сортировки необходимо:

1. Выделить одну ячейку таблицы.
2. Выбрать команду **Сортировка и Фильтр** в группе **Редактирование** на вкладке **Главная**.
3. Из раскрывающегося списка выберите способ сортировки – от минимального к максимальному или от максимального к минимальному. Нажатие на соответствующую кнопку приведет к выполнению команды.

Если необходимо провести сортировку по определенным параметрам, тогда:

1. Вызовите команду **Настраиваемая сортировка** в группе **Редактирование** на вкладке **Главная**.
2. В появившемся диалоговом окне *Сортировка* выберите столбец, в котором нужно отсортировать данные, и порядок сортировки. Если вы хотите выполнить сортировку сразу по нескольким столбцам, заполните поля *Затем по*, *В последнюю очередь по*. Чтобы дополнить список сортируемых столбцов, нажмите кнопку *Добавить уровень*.

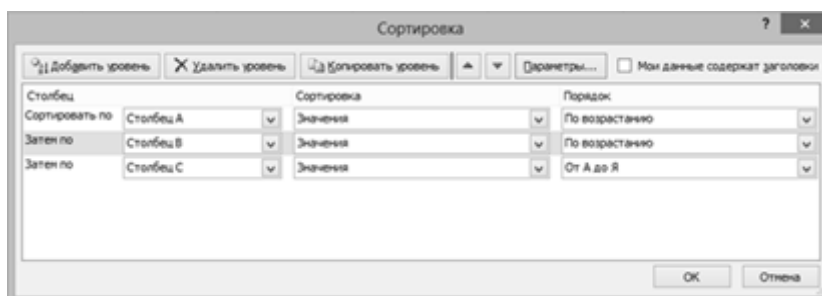


Рисунок 2.2 – Окно сортировки

Нажмите **ОК**.

Фильтрация данных – это быстрый и простой способ найти нужные для работы данные в диапазоне ячеек или таблице. Например, с помощью фильтра можно просмотреть только указанные значения, наибольшие и наименьшие либо повторяющиеся значения. Отфильтровав данные в

диапазоне ячеек или таблице, можно применить фильтр повторно, чтобы обновить результаты, либо очистить фильтр, чтобы отобразить все данные.

При помощи автофильтра можно создать фильтры трех типов: по списку значений, по формату или по условиям. Все они являются взаимоисключающими в пределах диапазона ячеек или столбца таблицы. Например, можно выполнить фильтрацию по цвету ячеек или по списку чисел, однако использовать фильтры обоих типов одновременно нельзя; точно так же необходимо выбрать один вариант, если требуется выполнить отбор по значкам или на основе настраиваемого фильтра.

Для выполнения фильтрации необходимо:

1. Выделите требуемый диапазон ячеек или просто установите курсор в произвольную ячейку диапазона, в котором будет производиться фильтрация.

2. В раскрывающемся списке **Сортировка и Фильтр** в группе **Редактирование** на вкладке **Главная** выберите команду **Фильтр**. После ее выполнения в каждой ячейке строки заголовка появятся кнопки, обозначающие раскрывающийся список. В этом списке находятся следующие команды:

– *Сортировка от минимального к максимальному* и *Сортировка от максимального к минимальному*, *Сортировка по цвету* (Пользовательская сортировка);

– *Удалить фильтр*;

– *Фильтр по цвету*;

– *Установка фильтрации*: Числовые фильтры (если в столбце числа), Текстовые фильтры (если в столбце текст), Фильтры по дате (если в столбце даты);

– Флажок *Выделить все*. Если снять этот флажок, то все столбцы перестанут отображаться;

– Флажки с названиями столбцов. Можно снимать флажки с разных столбцов, тогда они перестают отображаться.

3. После установки любого из фильтров в столбце с фильтром изменяется вид кнопки *Раскрывающийся список* – в нем появляется изображение фильтра.

4. Нажмите кнопку ОК.

Рассмотрим **Числовые фильтры**.

С помощью таких фильтров можно отобразить значения ячеек, которые принимают значения **Равно**, **Не равно**, **Больше**, **Больше или равно**, **Меньше**, **Меньше или равно**, **Между** какого-то указанного значения. Если выберите пункт **Первые 10**, то появляется окно, в котором можно выбрать какие первые 10 значений будут отображаться: 10 наибольших или 10 наименьших элементов списка или % от количества элементов. Причем количество отображаемых элементов или % можно изменять. Если выбираем пункт **Выше среднего** или **Ниже среднего**, то будут отображаться только те строки фильтруемого столбца, значения в которых выше либо ниже среднего арифметического, которое вычисляется автоматически.

Настраиваемый фильтр (Пользовательский автофильтр) позволяет указать условие фильтрации, состоящее из двух выражений, составляемых при помощи логических функций И или ИЛИ.

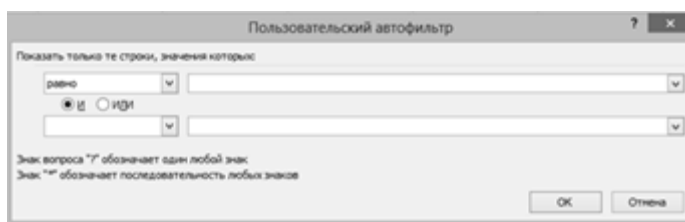


Рисунок 2.3 – Пользовательский автофильтр

Данное окно можно вызвать, выполнив команду **Настраиваемый фильтр** в раскрывающемся списке типа фильтра:

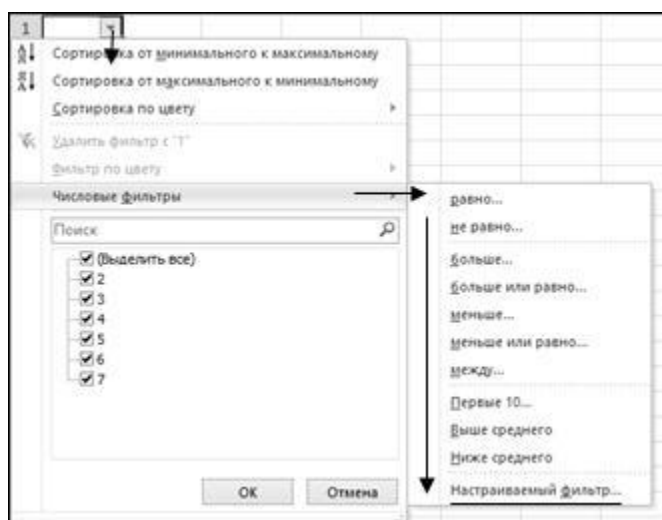


Рисунок 2.4 – Настраиваемый фильтр

С помощью **текстовых фильтров** можно выбрать такие параметры: **Равно**, **Не равно**, **Начинается с**, **Заканчивается на**, **Содержит**, **Не содержит**. Они настраиваются аналогично **Числовым фильтрам**.

Можно применять одновременно фильтрацию по нескольким столбцам. Допустим, выбираем: Наименования – начинается на букву А, затем устанавливаем фильтр по Складу 1 – больше 25.

Отменить фильтрацию можно одним из способов:

- с помощью горячих клавиш Shift+Ctrl+L;
- нажав кнопку **Фильтр | Сортировка и фильтр** на вкладке **Главная** в группе **Редактирование**.

Важно! Сортировать данные в таблицах Excel и проводить фильтрацию можно не только на вкладке **Главная**, но и с помощью команд **Сортировка**, **Фильтр**, **Дополнительно** в группе **Сортировка и Фильтр** на вкладке **Данные**.

Тема 2.6 Изменение данных в БД средствами запроса. Установка и отмена фильтра.

Изменение данных в БД средствами запроса

Запрос на обновление может быть использован для замены данных в таблицах БД. Отбор заменяемых записей (полей) производится с помощью **Запроса на выборку**, а затем в окне конструктора запросов с помощью кнопки **Обновление** на панели инструментов или команды меню **Запрос|Обновление** превращается в запрос на обновление.

Для обновления поля надо в строку **Обновление** ввести значение или выражение, определяющее новое значение поля. Такое выражение можно создать с помощью построителя выражений. После выполнения команды **Запрос|Запустили** нажатия соответствующей кнопки открывается диалоговое окно с сообщением о числе обновляемых записей и вопросом о продолжении операции обновления.

Результат выполнения запроса можно проверить переключением в режим таблица командой **Вид|Режим таблица** или нажатием кнопки **Представление запроса**. Если переключить в режим таблицы до выполнения **запроса**, то можно просмотреть старое содержимое обновляемых полей.

Запрос на добавление осуществляет добавление записей из таблицы запроса в другую таблицу. Добавляемые записи выбираются из одной или нескольких взаимосвязанных таблиц с помощью **Запроса на выборку**, который затем в окне конструктора запросов превращается в запрос на добавление кнопкой **Добавление** на панели инструментов или командой меню **Запрос|Добавление**. В открывшемся окне **Добавление** в поле **Имя таблицы** вводится имя таблицы, в которую надо добавить записи.

После выполнения этой команды в бланке запроса появляется строка **Добавление**. Для формирования добавляемых записей надо включить в бланк запроса поля, соответствующие определенным полям таблицы, в которые будет производиться добавление, и там, где необходимо, записать условия отбора полей в ячейки строки **Условие отбора**. Если в таблице, в которую добавляются записи, есть ключ, то ключевые поля обязательно должны быть включены в бланк запроса.

Для указания в строке **Добавление** имен полей таблицы-получателя надо в каждой ячейке открыть список и выбрать нужное имя. Если выбранные поля имеют одни и те же имена в обеих таблицах, имена в строку **Добавление** вносятся автоматически.

Предварительный просмотр добавляемых записей производится кнопкой **Представление запроса** на панели инструментов. Возврат в режим конструктора запросов производится этой же кнопкой.

Для добавления записей нажимается кнопка **Запуск** на панели инструментов. В открывшемся диалоговом окне сообщается о числе обновляемых записей и задается вопрос о продолжении операции добавления.

Если таблица, в которую добавляются записи, содержит ключевое поле, то добавляемые записи должны содержать такое же поле.

Запрос на удаление позволяет удалить записи из одной таблицы или из нескольких взаимосвязанных таблиц. Удаляемые записи выбираются с помощью **Запроса на выборку**, а затем в окне конструктора он превращается в **Запрос на удаление** с помощью кнопки **Удаление** на панели инструментов или команды меню **Запрос|Удаление**.

После выполнения этой команды в бланке запроса появляется строка Удаление. Символ звездочка (*) из списка полей таблицы, записи которой требуется удалить, перемещается мышкой в бланк запроса. В строке Удаление в столбце этого поля появляется значение Из. Для полей, для которых это необходимо, заполняется строка Условие отбора.

Для предварительного просмотра удаляемых записей можно нажать кнопку **Представление запроса** на панели инструментов. Для возврата в режим конструктора запроса используется эта же кнопка. Для удаления записей нажимается кнопка **Запуск** на панели инструментов.

Результаты запроса зависят от установленных в схеме БД отношений между таблицами и параметров целостности. Если для связи установлен параметр целостности Каскадное удаление связанных записей, то в результате выполнения этого запроса будут удалены все связанные записи. Если параметры целостности Каскадное удаление связанных записей не установлены, то записи удаляются только в указанных в бланке запроса таблицах вне зависимости от их логических связей.

Раздел 3. Организация запросов SQL

Тема 3.1 Исторические аспекты. Структура и типы данных языка SQL

SQL (Structured Query Language) — Структурированный Язык Запросов — стандартный язык запросов по работе с реляционными БД. Язык SQL появился после реляционной алгебры, и его прототип был разработан в конце 70-х годов в компании IBM Research. Он был реализован в первом прототипе реляционной СУБД фирмы IBM System R. В дальнейшем этот язык применялся во многих коммерческих СУБД и в силу своего широкого распространения постепенно стал стандартом «де-факто» для языков манипулирования данными в реляционных СУБД.

Первый международный стандарт языка SQL был принят в 1989 г. (далее мы будем называть его SQL/89 или SQL1). Иногда стандарт SQL1 также называют стандартом ANSI/ISO, и подавляющее большинство доступных на рынке СУБД поддерживают этот стандарт полностью. Однако развитие информационных технологий, связанных с базами данных, и необходимость реализации переносимых приложений потребовали в скором времени доработки и расширения первого стандарта SQL.

В конце 1992 г. был принят новый международный стандарт языка SQL, который в дальнейшем будем называть SQL/92 или SQL2. И он не лишен недостатков, но в то же время является существенно более точным и полным, чем SQL/89. В настоящий момент большинство производителей СУБД внесли изменения в свои продукты так, чтобы они в большей степени удовлетворяли стандарту SQL2.

В 1999 году появился новый стандарт, названный SQL3. Если отличия между стандартами SQL1 и SQL2 во многом были количественными, то стандарт SQL3 соответствует качественным серьезным преобразованиям. В SQL3 введены новые типы данных, при этом предполагается возможность задания сложных структурированных типов данных, которые в большей степени соответствуют объектной ориентации. Наконец, добавлен раздел, который вводит стандарты на события и триггеры, которые ранее не затрагивались в стандартах, хотя давно уже широко использовались в коммерческих СУБД. В стандарте определены возможности четкой спецификации триггеров как совокупности события и действия. В качестве действия могут выступать не только последовательность операторов SQL, но и операторы управления ходом выполнения программы. В рамках управления транзакциями произошел возврат к старой модели транзакций, допускающей *точки сохранения (savepoints)*. Возможность указания в операторе отката ROLLBACK точек возврата позволит откатывать транзакцию не в начало, а в промежуточную ранее сохраненную точку. Такое решение повышает гибкость реализации сложных алгоритмов обработки информации.

В настоящее время действует стандарт, принятый в 2003 году (SQL:2003) с небольшими модификациями, внесёнными позже в 2008 г.

1986 – первый вариант стандарта, принятый институтом ANSI и одобренный ISO в 1987г.

1992 – стандарт SQL 92 или SQL 2. Значительные изменения (ISO 9075); уровень Entry Level стандарта SQL-92 был принят как стандарт FIPS 127-2.

1999 – стандарт SQL 3. Добавлена поддержка регулярных выражений, рекурсивных запросов, поддержка триггеров, базовые процедурные расширения, не скалярные типы данных и некоторые объектно-ориентированные возможности.

2003 – SQL:2003. Введены расширения для работы с XML-данными, оконные функции, генераторы последовательностей и основанные на них типы данных.

2006 – SQL:2006. Функциональность работы с XML-данными значительно расширена. Появилась возможность совместно использовать в запросах SQL и XQuery.

2008 – SQL:2008. Улучшены возможности оконных функций, устранены некоторые неоднозначности стандарта SQL:2003

Структура и типы данных языка SQL

Компьютерная реализация БД определяется языками описания (**ЯОД**) и манипулирования данными (**ЯМД**). Они могут базироваться на реляционной алгебре (процедурные языки), реляционном исчислении кортежей (SQL) и доменов (QBE Query By Example) (декларативные языки).

Язык структурированных запросов **SQL** предназначен для доступа к информации и управления реляционной БД. Он является общим при работе с различными базами данных, такими как Oracle, Microsoft SQL Server, Informix, DB2, Access, MySQL.

Все СУБД, претендующие на название «реляционные», реализуют тот или иной диалект SQL: SQL*Plus корпорации Oracle; Transact-SQL для СУБД Microsoft SQL Server и др. В диалектах язык может быть дополнен операторами процедурных языков программирования.

В настоящее время, ни одна система не реализует стандарт SQL в полном объеме. Кроме того, во всех диалектах языка имеются возможности, не являющиеся стандартными. Таким образом, можно сказать, что каждый диалект – это надмножество некоторого подмножества стандарта SQL.

Язык SQL определяет:

- операторы языка, называемые командами языка SQL;
- типы данных;
- набор встроенных функций.

Выделяют две разновидности языка SQL интерактивный и вложенный. **Интерактивный SQL** позволяет конечному пользователю в интерактивном режиме выполнять SQL-операторы. Все СУБД предоставляют инструментальные средства для работы с базой данных в интерактивном

режиме. **Вложенный SQL** позволяет включать операторы SQL в код программы на другом языке программирования, например, C++.

В интерактивном языке SQL можно выделить три раздела:

1. DDL (Data Definition Language) – это язык определения данных, который включает операторы, управляющие объектами базы данных. К последним относятся таблицы, индексы, представления. Для каждой конкретной базы данных существует свой набор объектов базы данных, который может значительно расширять набор объектов, предусмотренный стандартом.

- CREATE DATABASE – создать базы данных;
- DROP DATABASE – удалить базу данных;
- CREATE TABLE – создать таблицу;
- ALTER TABLE – изменить таблицу;
- DROP TABLE – удалить таблицу;
- CREATE VIEW – создать представление;
- DROP VIEW – удалить представление.

2. DML (Data Manipulation Language) – язык манипулирования данными:

- SELECT – отобразить строки из таблиц;
- INSERT – добавить строки в таблицу;
- UPDATE – изменить строки в таблице;
- DELETE – удалить строки в таблице;

3. DCL (Data Control Language) – язык управления данными состоит из операторов контроля данных, защиты и управления данными:

- COMMIT – зафиксировать внесенные изменения;
- ROLLBACK – откатить внесенные изменения.
- GRANT – предоставить привилегии пользователю или приложению на манипулирование объектами;
- REVOKE – отменить привилегии пользователя или приложения.

В языке SQL имеется шесть скалярных типов данных, определенных стандартом. Их краткое описание представлено в таблице

Тип данных	Объявления
Символьный	CHAR VARCHAR
Битовый	BIT BIT VARYING
Точные числа	NUMERIC DECIMAL INTEGER SMALLINT
Вещественные числа	FLOAT REAL DOUBLE PRECISION
Дата/время	DATE TIME TIMESTAMP
Интервал	INTERVAL

Строковые типы:

CHARACTER или **CHAR** - символьные строки постоянной длины в n символов. При задании данного типа под каждое значение всегда отводится n символов, и если реальное значение занимает менее, чем n символов, то СУБД автоматически дополняет недостающие символы пробелами.

VARCHAR - строки символов переменной длины. Битовые типы:

BIT - строка битов постоянной длины.

BIT VARYING - строка битов переменной длины. Точные типы:

NUMERIC - точные числа, здесь p - общее количество цифр в числе, m - количество цифр слева от десятичной точки.

DECIMAL - точные числа, здесь p - общее количество цифр в числе, m - количество цифр слева от десятичной точки.

DEC - то же, что и DECIMAL.

INTEGER или **INT** - целые числа.

SMALLINT - целые числа меньшего диапазона.

Вещественные типы:

FLOAT - числа большой точности, хранимые в форме с плавающей точкой. Здесь p - число байтов, резервируемое под хранение одного числа. Диапазон чисел определяется конкретной реализацией.

REAL - вещественный тип чисел, который соответствует числам с плавающей точкой, меньшей точности, чем FLOAT.

DOUBLE PRECISION специфицирует тип данных с определенной в реализации точностью большей, чем определенная в реализации точность для REAL.

Типы даты/времени и интервал:

DATE - календарная дата.

TIME – формат времени.

TIMESTAMP(точность) - дата и время.

INTERVAL - временной интервал.

Большинство коммерческих СУБД поддерживают дополнительные типы данных, которые не специфицированы в стандарте. Так, например, практически все СУБД в том или ином виде поддерживают тип данных для представления неструктурированного текста большого объема. Этот тип аналогичен типу MEMO в настольных СУБД. Называются эти типы по-разному, например в ORACLE этот тип называется LONG, в DB2 - LONG VARCHAR, в SYBASE и MS SQL Server - TEXT.

Встроенные SQL-функции

При создании ограничений возможно использование следующих встроенных SQL-функций:

Функции, возвращающие системные переменные:

- SYSDATE. Возвращает текущие дату и время;
- USERENV. Возвращает значение, содержащее информацию о текущем сеансе.

Числовые функции:

- ABS. Возвращает абсолютное значение числа;
- MOD. Возвращает остаток от деления одного числа на другое;
- ROUND. Округляет число до указанного количества разрядов;
- SIGN. Определяет знак числа, переданного в качестве параметра;
- TRUNC. Усекает число до указанного количества разрядов.

Текстовые функции:

- CHR. Возвращает символ для заданного номера символа из таблицы ASCII;
- CONCAT3. Возвращает строку - результат конкатенации («сцепления») трех строк;
- UPPER. Возвращает строку, где все символы переведены в верхний регистр;
- LENGTH. Возвращает длину строки, измеренную в символах;
- SUBSTR. Возвращает часть строки, начинающуюся с символа с номером «а» и имеющую длину «b» символов;
- INSTR. Возвращает позицию вхождения подстроки в строку;
- LPAD. Возвращает строку, дополненную слева до необходимой длины символами из другой строки;
- RPAD. Возвращает строку, дополненную справа до необходимой длины символами из другой строки;
- LTRIM. Возвращает строку с удаленными левыми символами, встречающимися в заданной подстроке;
- RTRIM. Возвращает строку с удаленными правыми символами, встречающимися в заданной подстроке;

- REPLACE. Возвращает строку, где каждое вхождение «search_str» заменено на «replace_str».

Функции для работы с датами:

- ADD_MONTHS. Возвращает дату, увеличенную на указанное число месяцев.

Функции преобразования данных:

- TO_CHAR. Преобразует дату или число в строку символов;
- TO_LONGCHAR. Преобразует входное значение в длинную строку (LONGCHAR);
- TO_DATE. Преобразует строку в дату;
- TO_NUMBER. Преобразует строку в числовое значение.

Прочие функции:

- DECODE. Аналог последовательности вложенных операторов IF-THEN-ELSE;
- NVL. Возвращает «expr2», если «expr1» содержит значение NULL, в противном случае возвращает «expr1»;
- CALL_DATE. Функция возвращает дату - результат выполнения зарегистрированной в базе процедуры;
- CALL_NUMBER. Функция возвращает число - результат выполнения зарегистрированной в базе процедуры;
- CALL_VARCHAR. Функция возвращает строку - результат выполнения зарегистрированной в базе процедуры.

Тема 3.2 Операторы языка SQL

Оператор выбора SELECT

Язык запросов (*Data Query Language*) в *SQL* состоит из единственного оператора SELECT. Этот единственный оператор поиска реализует все *операции реляционной алгебры*. Как просто, всего один оператор. Однако писать запросы на языке *SQL* (грамотные запросы) сначала совсем не просто. Надо учиться, так же как надо учиться решать математические задачки или составлять алгоритмы для решения непростых комбинаторных задач. Один и тот же *запрос* может быть реализован несколькими способами, и, будучи все правильными, они, тем не менее, могут существенно отличаться *по* времени исполнения, и это особенно важно для больших баз данных.

Синтаксис оператора SELECT имеет следующий вид:

SELECT[ALL|DISTINCT](<Список полей>|*)

FROM <Список таблиц>

[WHERE <Предикат-условие выборки или соединения>]

[GROUP BY <Список полей результата>]

[HAVING <Предикат-условие для группы>]

[ORDER BY <Список полей, по которым упорядочить вывод>]

Здесь *ключевое слово* ALL означает, что в результирующий набор строк включаются все строки, удовлетворяющие условиям запроса. Значит, в результирующий набор могут попасть одинаковые строки. И это нарушение принципов теории отношений (в отличие от *реляционной алгебры*, где *по* умолчанию предполагается отсутствие дубликатов в каждом результирующем отношении). *Ключевое слово* DISTINCT означает, что в результирующий набор включаются только различные строки, то есть дубликаты строк результата не включаются в набор.

Символ *. (звездочка) означает, что в результирующий набор включаются все столбцы из исходных таблиц запроса.

В разделе **FROM** задается перечень исходных отношений (таблиц) запроса.

В разделе **WHERE** задаются условия отбора строк результата или условия соединения кортежей исходных таблиц, подобно *операции* условного соединения в *реляционной алгебре*.

В разделе **GROUP BY** задается *список* полей группировки.

В разделе **HAVING** задаются предикаты-условия, накладываемые на каждую группу.

В части **ORDER BY** задается *список* полей упорядочения результата, то есть *список* полей, который определяет порядок сортировки в результирующем отношении. Например, если первым полем списка будет указана Фамилия, а вторым Номер группы, то в результирующем отношении сначала будут собраны в алфавитном порядке студенты, и если найдутся однофамильцы, то они будут расположены в порядке возрастания номеров групп.

В выражении условий раздела WHERE могут быть использованы следующие предикаты:

Предикаты сравнения { =, <, >, <=, >=, <> }, которые имеют традиционный смысл.

Предикат Between A and B —принимает значения между A и B. Предикат истинен, когда сравниваемое значение попадает в заданный диапазон, включая границы диапазона. Одновременно в стандарте задан и противоположный предикат Not Between A and B, который истинен тогда, когда сравниваемое значение не попадает в заданный интервал, включая его границы.

Предикат вхождения в множество IN (множество) истинен тогда, когда сравниваемое значение входит в множество заданных значений. При этом множество значений может быть задано простым перечислением или встроенным подзапросом. Одновременно существует противоположный предикат NOT IN (множество), который истинен тогда, когда сравниваемое значение не входит в заданное множество.

Предикаты сравнения с образцом LIKE и NOT LIKE. Предикат LIKE требует задания шаблона, с которым сравнивается заданное значение, предикат истинен, если сравниваемое значение соответствует шаблону, и ложен в противном случае. Предикат NOT LIKE имеет противоположный смысл.

По стандарту в шаблон могут быть включены специальные символы:

символ подчеркивания () — для обозначения любого одиночного символа;

символ процента (%) — для обозначения любой произвольной последовательности символов;

остальные символы, заданные в шаблоне, обозначают самих себя.

Предикат сравнения с неопределенным значением IS NULL. Понятие неопределенного значения было внесено в концепции баз данных позднее. Неопределенное значение интерпретируется в реляционной модели как значение, неизвестное на данный момент времени. Это значение при появлении дополнительной информации в любой момент времени может быть заменено на некоторое конкретное значение. При сравнении неопределенных значений не действуют стандартные правила сравнения: одно неопределенное значение никогда не считается равным другому неопределенному значению. Для выявления равенства значения некоторого атрибута неопределенному применяют специальные стандартные предикаты:

<имя атрибута>IS NULL и <имя атрибута> IS NOT NULL.

Если в данном кортеже (в данной строке) указанный атрибут имеет неопределенное значение, то предикат IS NULL принимает значение "Истина" (TRUE), а предикат IS NOT NULL — "Ложь" (FALSE), в противном случае предикат IS NULL принимает значение "Ложь", а предикат IS NOT NULL принимает значение "Истина".

Введение Null-значений вызвало необходимость модификации классической двужанной логики и превращения ее в трехзначную. Все логические операции, производимые с неопределенными значениями, подчиняются этой логике в соответствии с заданной таблицей истинности:

A	B	Not A	$A \wedge B$	$A \vee B$
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	TRUE
TRUE	Null	FALSE	Null	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	Null	TRUE	FALSE	Null
Null	TRUE	Null	Null	TRUE
Null	FALSE	Null	FALSE	Null
Null	Null	Null	Null	Null

Предикаты существования EXISTS и несуществования NOT EXISTS. Эти предикаты относятся к встроенным подзапросам, и подробнее мы рассмотрим их, когда коснемся вложенных подзапросов.

В условиях поиска могут быть использованы все рассмотренные ранее предикаты.

Отложив на время знакомство с группировкой, рассмотрим детально первые три строки оператора SELECT:

SELECT — ключевое слово, которое сообщает СУБД, что эта команда — запрос. Все запросы начинаются этим словом с последующим пробелом. За ним может следовать способ выборки — с удалением дубликатов (**DISTINCT**) или без удаления (**ALL**, подразумевается по умолчанию). Затем следует список перечисленных через запятую столбцов, которые выбираются запросом из таблиц, или символ '*' (звездочка) для выбора всей строки. Любые столбцы, не перечисленные здесь, не будут включены в результирующее отношение, соответствующее выполнению команды. Это, конечно, не значит, что они будут удалены или их информация будет стерта из таблиц, потому что запрос не воздействует на информацию в таблицах — он только показывает данные.

FROM — ключевое слово, подобно **SELECT**, которое должно быть представлено в каждом запросе. Оно сопровождается пробелом и затем именами таблиц, используемых в качестве источника информации. В случае если указано более одного имени таблицы, неявно подразумевается, что над перечисленными таблицами осуществляется операция декартова произведения. Таблицам можно присвоить имена-псевдонимы, что бывает полезно для осуществления операции соединения таблицы с самой собою или для доступа из вложенного подзапроса к текущей записи внешнего запроса (вложенные подзапросы здесь не рассматриваются).

Все последующие разделы оператора **SELECT** являются необязательными.

Самый простой запрос **SELECT** без необязательных частей соответствует просто декартову произведению. Например, выражение

SELECT * FROM R1, R2

соответствует декартову произведению таблиц **R1** и **R2**. Выражение

SELECT R1.A, R2.B FROM R1, R2

соответствует проекции декартова произведения двух таблиц на два столбца **A** из таблицы **R1** и **B** из таблицы **R2**, при этом дубликаты всех строк сохранены, в отличие от операции проектирования в *реляционной алгебре*, где при проектировании по умолчанию все дубликаты кортежей уничтожаются.

WHERE — ключевое слово, за которым следует предикат — условие, налагаемое на запись в таблице, которому она должна удовлетворять, чтобы попасть в выборку, аналогично операции селекции в *реляционной алгебре*.

Агрегатные функции языка SQL

В стандарте языка SQL определено несколько агрегатных функций:

- **count** — возвращает количество значений в указанном столбце;
- **sum** — возвращает сумму значений в указанном столбце;
- **avg** — возвращает усредненное значение в указанном столбце;
- **min** — возвращает минимальное значение в указанном столбце;

- **max** — возвращает максимальное значение в указанном столбце.

В качестве операнда данных функций может использоваться наименование только одного столбца, и все они возвращают единственное значение, С функциями `sum` и `avg` могут использоваться только числовые поля. С функциями `count`, `max` и `min` могут использоваться как числовые, так и символьные поля. При вызове всех перечисленных выше функций, кроме функции `count(*)`, осуществляется исключение всех пустых значений и только после этого операция применяется к оставшимся значениям столбца, Функция `count(*)` призвана осуществлять подсчет всех строк таблицы независимо от того, какие значения в них находятся.

Агрегатные функции нельзя использовать в предложении `where`, потому что предикаты оцениваются в терминах одиночной строки, а агрегатные функции — в терминах групп строк.

Группирование результатов

Выбираемые пользователем данные могут быть подвергнуты различного рода анализу и обобщению, и язык SQL имеет для этого свои средства. Только что рассмотренные агрегатные функции являются примером таких средств. В приведенных выше запросах агрегатные функции применялись ко всей таблице и выдавали сводные данные на основании обработки значений всего выделенного столбца. Такие результаты обычно размещаются в конце отчета и сжимаются в итоговую строку.

Однако часто встречаются ситуации, когда в отчет необходимо поместить и промежуточные результаты, опирающиеся на вычисления обобщенных групповых значений. Для применения агрегатных функций в подобных случаях предполагается предварительная операция группировки. Суть операции группировки состоит в том, что все множество строк таблицы разбивается на группы, в каждой из которых собираются строки, имеющие одинаковые значения атрибутов, которые заданы в списке группировки. Обработка такой информации реализуется путем применения агрегатных функций уже к каждой отдельной группе и выдаче полученных итогов.

В языке SQL для осуществления операции группировки в оператор `select` включается фраза **group by**. Запрос, в котором присутствует фраза `group by`, называется *группирующим запросом*, а столбцы, перечисленные в этой фразе, называются *группирующими столбцами*.

Стандарт языка требует, чтобы предложение `select` и фраза `group by` были тесно связаны между собой. Если в запросе должна использоваться группировка, то каждый элемент списка в предложении `select` должен иметь единственное значение для всей группы. К тому же в предложении `select` могут включаться в этом случае только следующие типы данных:

- имена столбцов;
- агрегатные функции;
- константы;
- выражения, состоящие из перечисленных выше элементов.

Таким образом, предложение `group by` позволяет определять подмножество значений в особом поле в терминах другого поля и применять функцию агрегата к подмножеству. Это дает возможность объединять поля и агрегатные функции в едином предложении `select`.

Вложенные запросы

Стандарт языка позволяет в тело одного оператора `select` внедрять другой оператор `select`. В такой ситуации можно рассматривать внешний и внутренний (внедряемый) операторы запроса. Обычно внутренний запрос генерирует значение, которое проверяется в предикате внешнего запроса (в предложении `where` или `having`), определяющего, верно оно или нет. Если внутренний оператор запроса помещен в предложения `where` и `having` внешнего оператора `select`, то создается ситуация вложенных запросов (подзапросов).

В сочетании с другими возможностями оператора выбора, такими как группировка, возможность вкладывать запросы внутрь друг друга представляет собой мощное средство для достижения нужного результата. Во фразе `from` оператора `select`, если при формировании запроса требуется более чем один экземпляр некоторой таблицы, допустимо применять синонимы к именам таблицы. Синонимы задаются с использованием ключевого слова `as`:

```
FROM RI AS A, RI AS B.
```

Подзапросы могут быть нескольких видов:

- скалярный подзапрос, возвращающий единственное значение;
- строковый подзапрос, возвращающий несколько значений в виде одной строки;
- табличный подзапрос, возвращающий данные в виде таблицы.

Подзапрос может указываться непосредственно после операторов сравнения (`!=`, `<`, `>`, `<=`, `>=`) и его текст должен быть заключен в скобки.

Пример:

```
SELECT ФИО, Этап, Начисления  
FROM R  
WHERE Начисления > (SELECT avg(Начисления) FROM R)
```

Многотабличные запросы

При работе с базами данных потребности пользователей не ограничиваются только реализацией простых запросов данных из одной таблицы. Во многих случаях для получения ответа на запрос необходимо объединить информацию из нескольких исходных таблиц. Для того чтобы осуществить такое объединение в результирующей таблице, необходимо выполнить операцию соединения, при которой объединение информации из двух таблиц происходит посредством образования пар связанных строк, выбранных из каждой таблицы. Таблицам можно присвоить имена-псевдонимы, что бывает полезно для осуществления операции соединения таблицы с самой собою и в ряде других ситуаций.

Если в операторе select указано более одного имени таблицы, неявно подразумевается, что над перечисленными таблицами осуществляется операция декартова произведения. Самый простой запрос select такого рода без необязательных частей выглядит следующим образом:

```
SELECT * FROM R1, R2;
```

и соответствует декартову произведению таблиц *R1* и *R2*. Выражение

```
SELECT R1.A, R2.B FROM R1, R2;
```

соответствует проекции декартова произведения двух таблиц на два столбца *A* из таблицы *R1* и *B* из таблицы *R2*.

Пример

```
SELECT R3.ФИО, R3.Этап FROM R1, R3
```

```
WHERE R1.Отдел = '03' AND
```

```
R1.ФИО = R3.ФИО AND
```

```
R3.Этап = 'Этап_3'
```

Операторы DML - операторы манипулирования данными:

- SELECT - отобразить строки из таблиц
- INSERT - добавить строки в таблицу
- UPDATE - изменить строки в таблице
- DELETE - удалить строки в таблице
- COMMIT - зафиксировать внесенные изменения
- ROLLBACK - откатить внесенные изменения

Примеры использования операторов манипулирования данными

SELECT – выбрать данные из таблицы.Пр. Выбрать все строки из таблицы поставщиков, удовлетворяющих нек-ому условию (ключевое слово *WHERE...*):

```
SELECT *
```

```
FROM P
```

```
WHERE P.PNUM > 2;
```

INSERT - вставка строк в таблицу. Пр. Вставка одной строки в таблицу:

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

UPDATE - обновление строк в таблице. Пр. Обновление нескольких строк в таблице:

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

DELETE - удаление строк в таблице. Пр. Удаление нескольких строк в таблице:

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

Пр. Удаление всех строк в таблице:

```
DELETE FROM P;
```

Операторы DDL (Data Definition Language) - операторы определения объектов БД:

- CREATE SCHEMA - создать схему базы данных
- DROP SHEMA - удалить схему базы данных
- CREATE TABLE - создать таблицу.
- ALTER TABLE - изменить таблицу
- DROP TABLE - удалить таблицу
- CREATE DOMAIN - создать домен
- ALTER DOMAIN - изменить домен
- DROP DOMAIN - удалить домен
- CREATE COLLATION - создать последовательность
- DROP COLLATION - удалить последовательность
- CREATE VIEW - создать представление
- DROP VIEW - удалить представление

Пример применения операторов DDL:

(1)create table dolznost (kod_dolz char(6) not null, name_dolz varchar(30) not null)

(2)alter table dolznost add constraint PK_kod_dolz primary key(kod_dolz)

(3)**DROP TABLE dolznost**

Операторы создания и удаления индексов

Операторы создания и удаления индекса имеют следующий формат.

Создать индекс:

CREATE [UNIQUE] [CLUSTERED | NONCLUSTERED] INDEX <имя_индекса>

ON <Имя таблицы> (<Имя столбца> [ASC/DESC] [, <Имя столбца> [ASC/DESC] ...])

Где UNIQUE — создает уникальный индекс для таблицы или представления. Уникальным является индекс, в котором не может быть двух строк с одним и тем же значением ключа индекса. Кластеризованный индекс представления должен быть уникальным. Ядро СУБД не позволяет создать уникальный индекс по столбцам, уже содержащим повторяющиеся значения. При попытке создания такого индекса выдается сообщение об ошибке. Прежде чем создавать уникальный индекс по такому столбцу или столбцам, необходимо удалить все повторяющиеся значения. Столбцы, используемые в уникальном индексе, должны иметь свойство NOT NULL, т. к. при создании индекса значения NULL рассматриваются как повторяющиеся.

CLUSTERED — Создает индекс, в котором логический порядок значений ключа определяет физический порядок соответствующих строк в таблице. На нижнем (конечном) уровне такого индекса хранятся действительные строки данных таблицы. Для таблицы или представления в каждый момент времени может существовать только один кластеризованный индекс. Представление с уникальным кластеризованным индексом называется индексированным. Создание уникального кластеризованного индекса физически материализует представление. Уникальный

кластеризованный индекс для представления должен быть создан до того, как для этого же представления будут определены какие-либо другие индексы. Создавайте кластеризованные индексы до создания любых некластеризованных. При создании кластеризованного индекса все существующие некластеризованные индексы таблицы перестраиваются. Если аргумент CLUSTERED не указан, создается некластеризованный индекс.

NONCLUSTERED — создание индекса, задающего логическое упорядочение для таблицы. Логический порядок строк в некластеризованном индексе не влияет на их физический порядок. Для каждой таблицы можно создать до 999 некластеризованных индексов, независимо от того, каким образом они создаются: неявно с помощью ограничений PRIMARY KEY и UNIQUE или явно с помощью инструкции CREATE INDEX. Для индексированных представлений некластеризованные индексы могут создаваться только в случае, если уже определен уникальный кластеризованный индекс.

По умолчанию, используется значение NONCLUSTERED.

Удалить индекс:

DROP INDEX <Имя индекса>

Если в операторе CREATE INDEX используется квалификатор UNIQUE, то уникальность значений индекса автоматически поддерживается системой. Для каждого из ключевых столбцов можно указать порядок следования значений: по возрастанию — ASC (используется по умолчанию) и по убыванию — DESC.

Встроенный SQL

Назначение встроенного SQL – комбинировать достоинства языков, давая возможность разрабатывать сложные процедурные программы, которые обращаются к БД через SQL, скрывая от пользователя сложности работы с таблицами в процедурном языке. Встраиваемые SQL-операторы могут вставляться статически или динамически. Программа со встроенным SQL представляет собой смесь SQL-операторов и кода на базовом (HOST) языке программирования, поэтому ее нельзя просто скомпилировать с помощью компилятора базового языка.

Особенности создания программ.

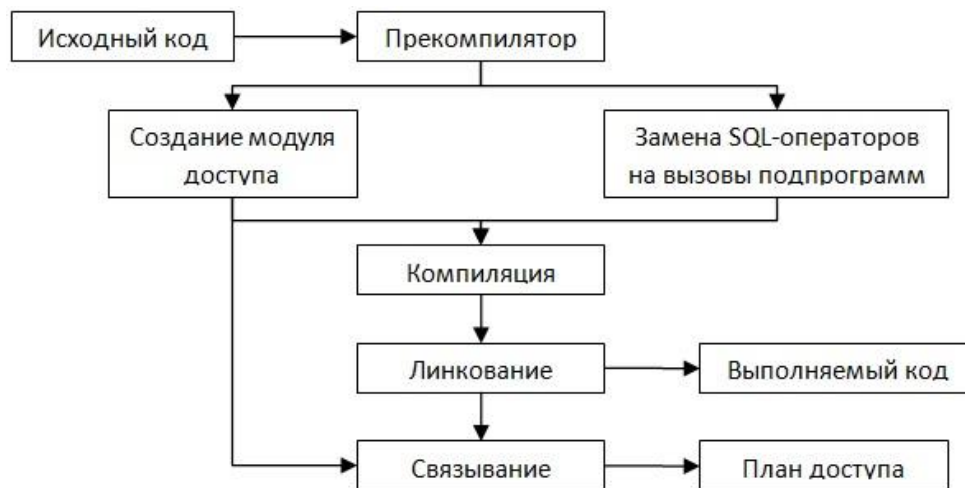


Рисунок 3.1 Алгоритм создания программ

1) Прекомпилятор

Исходная программа со встроенным SQL подается на вход прекомпилятора (препроцессора) SQL, представляющего собой специальную программу. Для каждого базового языка, поддерживаемого СУБД, необходим свой прекомпилятор. Препроцессор создает на выходе две файла. Первый представляет собой исходный текст программы, из которого удалены SQL-операторы. Препроцессор заменяет их на вызовы «закрытых» функций СУБД на этапе исполнения. Второй файл содержит копии всех SQL-операторов, встроенных в программу, и часто называется модулем запросов к БД.

2) Компоновка

Первый файл компилируется обычным компилятором базового языка в объектный код. Объектные модули, созданные компилятором, поступают на вход компоновщика. Он связывает их с различными библиотечными и «закрытыми» функциями доступа к БД и на выходе создает исполняемую программу.

3) План выполнения

Второй файл передается в специальную программу BIND. Она исследует, анализирует, проверяет, оптимизирует SQL-операторы и для каждого создает план выполнения. Результатом работы этой программы является объединенный план выполнения всех встроенных SQL-операторов, обычно сохраняющийся в БД под именем прикладной программы. Из-за изменений в БД план выполнения может устареть и при следующем запуске может быть произведена перекомпиляция SQL-операторов и создан новый план выполнения запроса.

3. Особенности встраивания SQL-операторов

Особенность встраивания SQL-операторов внутрь кода на HOST-языке заключается в том, что они предваряются словами-флагами, позволяющими далее разделить текст программы на две части для дальнейшей обработки:

EXEC SQL <SQL-оператор>

4. Использование переменных HOST-языка

Части программы с SQL-операторами и HOST-языки взаимодействуют в основном с помощью переменных, иногда называемых базовыми. SQL-92 определяет эквиваленты SQL-типов переменных для ряда языков.

Правила объявления переменных:

- Должны быть объявлены в ограничителях **BEGIN DECLARE SECTION|END DECLARE SECTION** с применением синтаксиса HOST-языка. Число таких секций и места их расположения не ограничиваются.

- Должны быть совместимыми по типу данных с их функциями в SQL-операторе.
- Должны иметь значение к тому моменту времени, когда они используются в SQL-операторе, независимо от того, что сам SQL-оператор может присваивать значение.

Пример объявления и использования:

EXEC SQL BEGIN DECLARE SECTION;

Var Id_num :Integer;

SalesPerson :Packed Array[1..10] Of Char;

Loc :Packed Array[1..10] Of Char;

Comm :Real;

EXEC SQL END DECLARE SECTION;

Если указанные переменные содержат значения, которые необходимо вставить в таблицы, то SQL-оператор может иметь следующий вид:

EXEC SQL INSERT INTO SalesPeople VALUES (:Id num, :SalesPerson, :Loc, :Comm);

Раздел 4. Организация интерфейса с пользователем

Тема 4.1. Понятие объекта, свойства и характеристики объекта.

Создание экранной формы: свойства, события и методы

Формы - диалоговый графический интерфейс для работы пользователя с базой данных. Формы предназначены для ввода и просмотра взаимосвязанных данных БД на экране в удобном виде, который может соответствовать привычному для пользователя документу. Формы можно распечатывать, а также применять для создания панелей управления в приложении.

Любая форма, с помощью которой вы хотите просматривать, вводить или редактировать записи таблиц БД, должна быть предварительно сконструирована. В процессе подготовительной работы по разработке формы определяется, из каких таблиц нужно отображать данные, какие именно поля должны быть представлены в форме, нужны ли вычисляемые поля, какие графические элементы, служащие для ее оформления, будут использоваться (линии, поясняющие текст, рисунки).

Основы создания формы

Однотабличная форма может быть создана пользователем в режиме Конструктора форм или с помощью Мастера. В первом случае создание начинается с пустой формы и конструирование полностью возлагается на пользователя. Для создания однотабличной формы целесообразно использовать Мастер форм или команды Автоформа.

Чтобы начать создание формы, надо в окне базы данных выбрать закладку Формы и нажать кнопку Создать. Открывающееся диалоговое окно Новая форма представляет возможность выбрать один из режимов создания формы:

- Конструктор,
- Мастер форм,
- Автоформа: в столбец,
- Автоформа: ленточная,
- Автоформа: табличная,
- Диаграмма,
- Сводная таблица

Формы, которые удовлетворяют любому, даже самому требовательному вкусу, можно создать с помощью Конструктора.

Эффективным способом работы является быстрое создание форм с помощью Мастера форм и дальнейшее их совершенствование с помощью Конструктора. Мастер форм может создавать форму для одной таблицы и для нескольких взаимосвязанных таблиц.

При выборе только одной таблицы могут быть созданы формы:

- В один столбец,

- Ленточная
- Табличная.

Форма В один столбец выводит в виде колонок для просмотра данные только одной записи, поля которой расположены в нужном порядке.

Ленточная форма выводит одну и более записей в зависимости от того, сколько можно уместить их на экране.

Табличная форма выводит данные обычным табличным способом, но в отличие от таблиц может выбирать поля для вывода.

Мастер форм позволяет пользователю определить, какие пол таблицы включаются в форму, и выбрать стиль ее оформления. Выбор таблицы для создания формы может быть произведен как в окне Новая форма, так и в первом диалоговом окне мастера Создание форм.

Команды **Автоформа**: в столбец, Автоформа: ленточная и Автоформа: табличная создают для заданной таблицы формы, которые отличаются от форм, создаваемых мастером, тем, что включают все поля таблицы и не предоставляют возможности выбора стиля оформления. Эти команды, не вступая в диалог с пользователем и не отображая формы в режиме конструктора, выводят ее на экран в режиме формы, то есть заполненную значениями из таблицы.

Заметим, что таблица, для которой строится форма, выбирается в окне Новая форма. Форма, созданная мастером, также, как и форма, созданная любой командой Автоформа, может быть отредактирована в соответствии с требованиями пользователя. Редактирование выполняется в режиме Конструктора форм.

Последние опции — Сводная таблица и Диаграмма — позволяют создавать достаточно специализированные по своим задачам формы и активно используют OLE-технологии.

Элементы управления

Все сведения в форме или отчете содержатся в элементах управления.

Элементы управления — это объекты формы или отчета, которые служат для вывода данных на экран, выполнения макрокоманд или оформления формы, или отчета.

Например, поле можно использовать для вывода данных на экран в форме или отчете, кнопку — для открытия другой формы или отчета, а линию или прямоугольник — для разделения и группировки элементов управления с тем, чтобы они лучше воспринимались пользователем. В Microsoft Access существуют следующие типы элементов управления, которые содержатся на панели элементов в режиме конструктора формы или режиме конструктора запроса.

1. надпись,
2. поле,
3. группа,
4. выключатель,

5. переключатель,
6. флажок,
7. поле со списком,
8. список,
9. кнопка,
10. рисунок,
11. свободная рамка объекта,
12. присоединенная рамка объекта,
13. разрыв страницы,
14. набор вкладок,
15. подчиненная форма/отчет,
16. линия,
17. прямоугольник
18. и дополнительные элементы ActiveX.

Элементы управления могут быть

- связанными,
- свободными
- или вычисляемыми.

Связанный элемент управления присоединен к полю базовой таблицы или запроса. Такие элементы управления используются для отображения, ввода или обновления значений из полей базы данных.

Для вычисляемого элемента управления в качестве источника данных используется выражение. В выражении могут быть использованы данные из поля базовой таблицы или запроса для формы или отчета, а также данные другого элемента управления формы или отчета. Для свободного элемента управления источника данных не существует. Свободные элементы управления используются для вывода на экран данных, линий, прямоугольников и рисунков.

Надписи предназначены для отображения в форме или отчете описательных текстов, таких, как заголовки, подписи или краткие инструкции. В надписях не выводятся значения полей или выражений; они всегда являются свободными и не меняются при переходе от записи к записи. Надпись может быть присоединена к другому элементу управления (такую надпись называют подписью). Например, поле создается с присоединенной надписью, которая содержит подпись этого поля. Эта надпись появляется как заголовок столбца в форме в режиме таблицы. Надпись, созданная с помощью инструмента «Надпись», размещается отдельно и не присоединяется ни к какому элементу управления. Такие надписи используются для отображения

разных сведений, например, заголовков формы или отчета, а также для вывода поясняющего текста. Надписи, не присоединенные к элементам управления, не отображаются в режиме таблицы.

Поля используются в форме или отчете для отображения данных из таблицы, запроса или инструкции SQL. Поле такого типа называют присоединенным, потому что оно связано с данными в поле в источнике данных. Кроме того, существуют свободные поля. Например, можно создать свободное поле для отображения результатов вычислений или для приема данных, вводимых пользователем. Содержимое свободного поля нигде не сохраняется.

Группа используется в форме или отчете для вывода ограниченного набора параметров. Группа делает выбор параметра простым и наглядным. В каждый момент времени в группе может быть выбран только один параметр. Группа состоит из рамки группы и набора флажков, переключателей или выключателей.

При присоединении группы к полю к нему присоединяется только рамка группы, а не находящиеся в ней флажки, выключатели или переключатели. Пользователь не должен определять свойство Данные (ControlSource) для каждого элемента управления в группе. Вместо этого следует задать в свойстве Значение параметра (OptionValue) каждого флажка, выключателя или переключателя число, являющееся допустимым для поля, к которому присоединена рамка группы. При выборе параметра в группе Microsoft Access вводит в поле значение, равное значению свойства Значение параметра (OptionValue) выбранного элемента.

В свойстве Значение параметра (OptionValue) требуется задавать число, так как значением группы может быть только числовое, а не текстовое значение. Microsoft Access сохраняет это число в базовой таблице.

Группа может быть также связана с выражением или быть свободной. Свободные группы применяются в специальных диалоговых окнах для принятия данных, вводимых пользователем, и для выполнения действий, основанных на этих данных.

Выключатель может быть использован в форме или отчете как отдельный элемент управления, в котором отображаются значения логического поля из базовой таблицы, запроса или инструкции SQL.

Когда пользователь нажимает кнопку выключателя, присоединенного к логическому полю, Microsoft Access отображает значение в базовой таблице в формате, который определяется значением свойства поля Формат поля (Format) («Да»/«Нет», «Истина»/«Ложь» или «Вкл»/«Выкл»). Выключатели особенно удобны при использовании в группах. В такой группе легко видеть, какой из выключателей нажат.

Переключатель может быть использован в форме или отчете как элемент управления, в котором отображаются значения логического поля из базовой таблицы, запроса или инструкции SQL. Когда пользователь выбирает переключатель, присоединенный к логическому полю, Microsoft

Access отображает значение в базовой таблице в формате, который определяется значением свойства поля Формат поля (Format) («Да»/«Нет», «Истина»/«Ложь» или «Вкл»/«Выкл»).

Переключатели обычно используются в группе для отображения набора параметров, из которых необходимо выбрать один.

Флажок может быть использован в форме или отчете как отдельный элемент управления, в котором отображаются значения логического поля из базовой таблицы, запроса или инструкции SQL.

При установке или снятии флажка, присоединенного к логическому полю, Microsoft Access отображает значение в базовой таблице в формате, который определяется значением свойства поля Формат поля (Format) («Да»/«Нет», «Истина»/«Ложь» или «Вкл»/«Выкл»). Кроме того, флажки включаются в группу для отображения набора выбираемых значений.

Во многих случаях удобнее выбрать значение из списка, чем вводить конкретное значение с клавиатуры по памяти.

Поле со списком позволяет выбрать любой из этих способов ввода значения, не требуя при этом значительного места в форме. Поле со списком является комбинацией двух элементов: поля и раскрывающегося списка. Значение, выбранное или введенное в присоединенное поле со списком, вставляется в поле, к которому присоединено поле со списком.

В поле со списком список состоит из строк с данными. Строки содержат один или несколько столбцов, с заголовками или без заголовков. Если поле со списком, содержащим нескольких столбцов, является присоединенным, то сохраняется значение одного из столбцов. Свободное поле со списком позволяет сохранять значение, используемое в другом элементе управления. Например, с помощью свободного поля со списком можно ограничить значения, отбираемые в другом поле со списком или в специальном диалоговом окне. Свободное поле применяется также для поиска записи с помощью значения, выбранного или введенного в поле со списком.

Поля со списком имеют свойство Ограничиться списком (LimitToList), которое определяет, допускается ли ввод в поле любых значений или только совпадающих с одним из значений списка. Если в форме достаточно свободного места и требуется, чтобы список постоянно находился на экране, а также если требуется ограничить вводимые данные имеющимся списком, вместо поля со списком можно использовать список.

Списки. Во многих случаях удобнее выбрать значение из списка, чем вводить конкретное значение по памяти. Кроме того, выбор из списка позволяет быть уверенным, что введенное значение является допустимым.

Список состоит из строк с данными. Строки содержат один или несколько столбцов, которые могут быть снабжены заголовками. Если список из нескольких столбцов является присоединенным, то сохраняется значения одного из столбцов.

Свободный список позволяет хранить значение, используемое в другом элементе управления. Например, с помощью свободного списка можно ограничить значения, отбираемые в другом списке или в специальном диалоговом окне. Свободный список применяется также для поиска записи с помощью значения, выбранного в списке.

В тех случаях, когда в форме недостаточно места для отображения списка, или если наряду с выбором из списка требуется вводить новые значения с клавиатуры, вместо списка следует использовать поле со списком.

Кнопки используются в формах для выполнения определенного действия или ряда действий. Например, можно создать в форме кнопку, открывающую другую форму. Чтобы кнопка выполняла какое-либо действие, следует создать макрос или процедуру обработки события и связать их со свойством кнопки Нажатие кнопки (OnClick).

Мастер кнопок позволяет создавать кнопки более 30 разных типов. При создании кнопки с помощью мастера для нее определяется процедура обработки события. Текст надписи на кнопке задается в качестве значения свойства Подпись (Caption). Чтобы поместить на кнопку рисунок, следует указать его в свойстве кнопки Рисунок (Picture).

Рисунки, свободные и связанные рамки объекта. В форму или отчет Microsoft Access можно добавлять объекты или части объектов, созданные в других приложениях, например, рисунок, созданный в Microsoft Paint, электронную таблицу, созданную в Microsoft Excel, или текстовый документ, созданный в Microsoft Word. Существует возможность вставлять все содержимое файла или только некоторую выделенную часть.

Способ вставки рисунка или объекта зависит от того, какой объект предполагается создать: присоединенный или свободный. Присоединенный объект хранится в таблице. При переходе к новой записи в форме или отчете отображается другой объект. Например, таким способом удобно хранить фотографии всех сотрудников фирмы. Свободный объект является частью структуры формы или отчета. При переходе к новой записи объект не изменяется.

Подчиненная форма — это форма, находящаяся внутри другой формы. Первичная форма называется главной формой, а форма внутри формы называется подчиненной формой. Комбинацию «форма/подчиненная форма» часто называют также иерархической формой или комбинацией «родительской» и «дочерней» форм.

Подчиненная форма удобна для вывода данных из таблиц или запросов, связанных отношением «один ко многим».

Главная форма и подчиненная форма в этом типе форм связаны таким образом, что в подчиненной форме выводятся только те записи, которые связаны с текущей записью в главной форме. Например, когда главная форма отображает тип «Напитки», подчиненная форма отображает только те товары, которые входят в тип «Напитки».

При использовании формы с подчиненной формой для ввода новых записей текущая запись в главной форме сохраняется при входе в подчиненную форму. Это гарантирует, что записи из таблицы на стороне «многие» будут иметь связанную запись в таблице на стороне «один». Это также автоматически сохраняет каждую запись, добавляемую в подчиненную форму. Подчиненная форма может быть выведена в режиме таблицы, как на предшествующей иллюстрации, или она может быть выведена как простая или ленточная форма. Главная форма может быть выведена только как простая форма.

Главная форма может содержать любое число подчиненных форм, если каждая подчиненная форма помещается в главную форму. Имеется также возможность создавать подчиненные формы двух уровней вложенности. Это означает, что можно иметь подчиненную форму внутри главной формы, а другую подчиненную форму внутри этой подчиненной формы. Например, можно иметь главную форму, в которой выводятся данные о клиентах, подчиненную форму, выводящую данные о заказах и другую подчиненную форму, которая отображает то, что заказано.

Подчиненным отчетом называют отчет, вставленный в другой отчет. При комбинировании отчетов один из отчетов является главным. Главный отчет может быть как присоединенным, так и свободным, т. е. не базирующимся на таблице, запросе или инструкции SQL. Свободный главный отчет может служить контейнером нескольких не связанных между собой отчетов, которые требуется объединить.

Главный отчет связывают с таблицей, запросом или инструкцией SQL в тех случаях, когда в него требуется вставить подчиненные отчеты, в которых выводятся данные, связанные с данными в главном отчете. Например, в главном отчете могут быть выведены все записи о продажах за год, а в подчиненном отчете — итоговые данные, например суммы продаж за каждый квартал. В главном отчете могут также содержаться данные, являющиеся общими для двух или нескольких подчиненных отчетов. В этом случае области данных выводятся в подчиненных отчетах. В главный отчет наряду с подчиненными отчетами включают также подчиненные формы, причем число таких подчиненных форм не ограничивается. Более того, главный отчет может содержать подчиненные формы или отчеты двух уровней вложенности. Например, в отчете может содержаться подчиненный отчет, который в свою очередь содержит подчиненную форму или подчиненный отчет.

Элементы управления Разрывы страниц, Линии, Прямоугольники используются для оформления форм и отчетов. Элемент управления Набор вкладок используется для представления нескольких

страниц данных в одном наборе. Это особенно удобно при работе со многими элементами управления, которые могут быть распределены на две или более категорий. Например, элемент управления Набор вкладок может быть использован в форме Сотрудники, чтобы отделить общие сведения от личных.

Элементы HTML и ActiveX позволяют добавить формам и отчетам еще некоторые функциональные возможности.

Последовательность загрузки таблиц базы данных

При разработке форм, обеспечивающих загрузку взаимосвязанных таблиц базы данных следует придерживаться определенных требований к последовательности их загрузки в соответствии со схемой данных. Эти требования можно сформулировать следующим образом:

- Независимо могут загружаться таблицы, которые не подчинены каким-либо другим таблицам в одно-многозначных связях.
- Таблицы, подчиненные каким-либо другим таблицам, могут загружаться либо одновременно с ними, либо после загрузки главных таблиц, в противном случае не могут установиться связи загружаемых в подчиненную таблицу записей с записями главных таблиц
- В базу данных сначала загружаются из соответствующих документов справочные данные, а затем учетные.

Перед конструированием форм для загрузки базы данных Access, необходимо проводить подготовительную работу по определению этапов загрузки.

Этапы загрузки базы данных и проектирования форм

В процессе определения этапов загрузки базы данных и проектирования форм целесообразно выполнить приведенную ниже последовательность действий:

1. Определение документов-источников внемашинной сферы, содержащие необходимые данные для загрузки таблиц базы данных.
2. Определение таблиц-объектов загрузки на отдельных этапах ввода данных и соответствующего документа-источника.
3. Определение последовательности этапов загрузки.
4. Определение подсхемы данных для каждого этапа загрузки (схемы данных), необходимой для построения экранной формы ввода документа.

В подсхему данных могут входить:

- Таблица-объект загрузки.
- Таблица, связанная с таблицей-объектом загрузки и содержащая данные только для отображения (вывода) в форме.
- Таблица, главная относительно загружаемой, позволяющая группе вводимые (выводимые) записи.

5. Определение общей структуры экранной формы, т.е. ее макета в соответствии со структурой входного документа и подсхемой данных. При этом:

- Для многотабличной (составной) формы выбирается таблица, которая будет источником записей основной части этой формы, и задается звание формы.
- Выбираются таблицы, которые будут источником записей подчиненных форм, включаемых в составную форму, и определяется названиям каждой подчиненной формы.
- Распределяется пространство формы для размещения ее основной части и включаемых подчиненных форм.
- Если подчиненная форма в свою очередь имеет включаемую (подчиненную 2-го уровня) форму, для нее выполняется работа для любой составной формы.
- Определение состава размещаемых реквизитов для каждой из частей составной формы.

При этом:

- Ключевые поля таблицы-источника основной части составной формы надо вводить в основную часть формы.
- Предусмотреть в подчиненной форме ключевые поля таблицы-источника подчиненной формы, которых нет в таблице-источнике основной части.

После выполнения перечисленных пунктов осуществляется конструирование экранной формы средствами Access.

Разработка многотабличных форм

Составная многотабличная форма создается для работы с несколькими взаимосвязанными таблицами. Многотабличная форма может состоять из основной части и одной или нескольких подчиненных включаемых форм, т. е. быть составной. Подчиненная форма может быть построена на основе как подчиненной, так и главной таблицы относительно таблицы - источника основной части формы.

Многотабличная форма может не иметь включаемых форм. В этом случае в форму кроме полей таблицы, на основе которой она строится, добавляются поля из связанной с ней таблицы, являющейся главной относительно основной таблицы формы. Многотабличная форма может быть создана в режиме конструктора или с помощью мастера форм.

Технология разработки формы для загрузки двух связанных таблиц.

Технология разработки многотабличной формы включает определение требований к создаваемой форме и сам процесс создания формы средствами Access.

При определении требований к составной форме целесообразно выполнить:

- определение подсхемы данных для соответствующего этапа загрузки,
- определение общей структуры экранной формы, т. е. ее макета в соответствии со структурой входного документа и подсхемой данных,

- определение состава размещаемых реквизитов для каждой из частей составной формы.

После выполнения перечисленных пунктов надо осуществить конструирование экранной формы средствами Access.

Рассмотрим технология разработки на примере составной формы для одновременной загрузки и работы с данными двух таблиц — ГРУППА и СТУДЕНТ.

Подготовка к созданию составной формы

Осуществим подготовку к созданию формы для загрузки данных в две таблицы — ГРУППА и СТУДЕНТ (объект загрузки), которые связаны в схеме данных одно-многочисленными отношениями.

Документом-источником такой формы является "Список студентов группы". С этого документа будут загружаться одновременно две таблицы: ГРУППА и СТУДЕНТ.

Определение подсхемы данных для составной формы

Поскольку объект загрузки ГРУППА—СТУДЕНТ не подчиняется в схеме данных другим таблицам, подсхема, необходимая для построения формы, не должна включать других таблиц.

Определение общей структуры составной формы

Определим общую структуру составной формы, которую назовем СПИСОК ГРУППЫ.

Для того чтобы обеспечить удобный ввод данных с документа, в форме предусмотрим основную часть с реквизитами группы и подчиненную с записями о студентах группы. Подчиненную форму назовем СПИСОК СТУДЕНТОВ.

Таким образом, форму СПИСОК ГРУППЫ определяют:

- тип формы — многотабличная,
- источник записей для основной части формы — таблица ГРУППА,
- включаемая подчиненная форма СПИСОК СТУДЕНТОВ.

Форму СПИСОК СТУДЕНТОВ определяют:

- тип формы — подчиненная, многозаписевая,
- источник записей — таблица СТУДЕНТ.

Размещение реквизитов основной и подчиненной форм

Размещение реквизитов в основной части формы и подчиненной форме должно соответствовать входному документу "Список студентов группы".

В основной части составной формы СПИСОК ГРУППЫ вверху разместим названия реквизитов, соответствующие полям таблицы ГРУППА:

- номер группы (НГ - ключ),
- количество студентов (КОЛ),
- средний балл группы при поступлении (ПБАЛЛ).

Расчетный реквизит ПБАЛЛ включаем в форму, т. к. она будет использоваться не только для загрузки, но и для просмотра данных.

В подчиненной форме СПИСОК СТУДЕНТОВ разместим в качестве заголовков столбцов многозаписевой формы названия реквизитов, соответствующих полей таблицы СТУДЕНТ:

- номер студента в группе (НС),
- фамилия И.О. (ФИО),
- год рождения (ГОДР),
- адрес (АДРЕС),
- средний балл при поступлении (ПБАЛЛ).

Замечание. Ключевое поле НГ не включено в подчиненную форму, т. к. НГ включен в основную часть формы.

Создание формы для двух таблиц с помощью мастера

Создадим средствами Access форму для одновременной загрузки и корректировки двух таблиц ГРУППА и СТУДЕНТ в базе данных «*Учебный процесс*».

Определение таблиц и полей для основной и включаемой частей формы

В окне базы данных выбираем закладку **Форма** и нажимаем кнопку **Создание формы с помощью мастера**. В открывшемся окне **Создание форм** выберем для таблицы ГРУППА поля, включаемые в форму. Выберем также вторую таблицу СТУДЕНТ и ее поля. Эта таблица будет источником записей подчиненной формы, связанных с записью отображаемой в основной части формы.

Выбор типа формы

В следующем окне **Создание форм** уже будет выделена таблица ГРУППА для создания основной части формы. Поскольку эта таблица является главной в схеме данных относительно другой таблицы СТУДЕНТ, представляется возможность выбора типа формы с включением подчиненной формы или создание связанной формы. Выберем первый вариант — **Подчиненные формы**.

Далее предоставляется возможность выбрать внешний вид подчиненной формы — **Ленточный** или **Табличный**. Выберем **Ленточный** для получения многозаписевой подчиненной формы и вывода в ней подписей полей (из структуры таблиц) вместо их имен. В следующем окне выберем стиль оформления **Стандартный**.

Назначение имени формы и ее открытие

В последнем окне **Создание форм** введем имя составной формы — СПИСОК ГРУППЫ — и подчиненной формы — СПИСОК СТУДЕНТОВ.

Выберем также дальнейшие действия мастера — *Открыть форму для просмотра или ввода данных*.

После завершения работы мастера выводится форма с данными из таблиц базы данных. При этом в подчиненной форме выводятся все записи, связанные с текущей записью основной части формы.

Созданная мастером форма может использоваться для работы с данными и без доработки. В случае необходимости можно отобразить данные подчиненной формы в виде таблицы с именами полей. Для этого в режиме формы выполняется команда **Вид → Таблица подчиненной формы**, работающая как переключатель. Курсор предварительно устанавливается на подчиненной форме.

Если в окне **Создание форм** выбрать дальнейшие действия мастера — *Изменить макет формы*, мастер закончит свою работу выводом формы в режиме конструктора.

Редактирование формы в режиме конструктора

Изменение надписей и размещения полей

В окне базы данных на закладке **Формы** выберем для редактирования созданную мастером многотабличную форму СПИСОК ГРУППЫ и нажмем кнопку **Конструктор**. Если форма была открыта ранее, то для перехода в режим конструктора достаточно нажать кнопку **Представление формы** на панели конструктора форм.

В окне конструктора форм указано имя формы СПИСОК ГРУППЫ, представлены размещенные в области данных поля основной части формы, а также рамка для подчиненной формы, в которой приводится ее имя СПИСОК СТУДЕНТОВ.

Произведем доработку формы СПИСОК ГРУППЫ, используя технику редактирования. Введем в область заголовка полное название формы СПИСОК СТУДЕНТОВ ГРУППЫ, которое будет выводиться при распечатке формы и будет соответствовать форме документа. Формирование текста в области заголовка подробно было рассмотрено выше при конструировании однотабличной формы.

Отмечая курсором мыши и перетаскивая отмеченный элемент, разместим поля так, как будет удобно для пользователя. Уточним подписи полей, шрифт и размеры полей и подписей, заголовков формы. Для изменения размера любого элемента надо расширять или сжимать рамочку этого элемента.

Создание кнопок для перехода к другой записи

Для того чтобы в форме переходить к следующей и предыдущей записи таблицы, создадим соответствующие *кнопки управления* в основной части многотабличной формы.

Нажмем на панели элементов Access кнопку **Мастер элементов**, а затем **Кнопка**. После переноса кнопки курсором мыши в нужное место и вычерчивания ее рамки запустится мастер кнопок.

В окне **Создание кнопки** выберем действие, которое необходимо выполнять при нажатии кнопки. Например, в рамке *Категории* выберем **Переходы по записям**, в рамке *Действия*

— *Предыдущая запись*. В следующем окне отмечаем *Рисунок*, *Показать все рисунки* и выбираем подходящий рисунок из списка, например, *Стрелка вверх (черная)*.

После нажатия **Готово** кнопка с выбранным рисунком встраивается в форму.

Аналогичные действия производятся для встраивания кнопки перехода к последующей записи таблицы — *Стрелка вниз (черная)* и кнопки закрытия формы — **STOP**.

Переход в режим формы и загрузка таблиц

После завершения создания многотабличной формы перейдем из режима конструктора в режим формы, нажав кнопку **Представление формы** на панели инструментов или выполнив команду меню **Вид → Форма**.

Если форма была закрыта, то для начала сеанса работы с данными через форму необходимо в окне базы данных выбрать закладку **Формы** и, отметив форму СПИСОК ГРУППЫ, нажать кнопку **Открыть**. Заметим, что в режиме формы в названии формы отображается ее заголовок — СПИСОК СТУДЕНТОВ ГРУППЫ.

Полученная многотабличная форма СПИСОК СТУДЕНТОВ ГРУППЫ обеспечивает одновременную загрузку и работу с данными двух таблиц ГРУППА и СТУДЕНТ.

В процессе загрузки сначала вводятся значения реквизитов группы: номер группы, количество студентов и средний балл группы. Заметим, что ввод ключевого поля — номер группы — всегда обязателен для создания записи в таблице ГРУППА. Причем эта запись создается независимо от того, будет ли введен список студентов группы. Другие два реквизита группы могут вводиться и позже при просмотре данных через форму, если не введены дополнительные ограничения на их значения.

Далее вводятся реквизиты студентов. При этом ввод номера студента в группе всегда обязателен для образования записи в таблице СТУДЕНТ, который вместе с введенным в основную часть формы номером группы образует уникальный ключ в этой таблице. Запись о студенте сохраняется при переходе к очередной строке в подчиненной форме.

Для перехода к записям другой группы используются созданные кнопки со стрелками вверх, вниз. Для перемещения по списку студентов используется строка, образованная Access, со счетчиком записей и кнопками перехода. Для завершения работы с формой нажмем созданную в форме кнопку **STOP**.

Тема 4.2. Хранимые процедуры и триггеры. Обеспечение достоверности, целостности и непротиворечивости данных. Каскадные воздействия

Хранимые процедуры

При работе с *SQL Server* пользователи могут создавать собственные процедуры, реализующие те или иные действия. *Хранимые процедуры* являются полноценными объектами базы

данных, а потому каждая из них хранится в конкретной базе данных. Непосредственный вызов *хранимой процедуры* возможен, только если он осуществляется в контексте той *базы данных*, где находится процедура.

Типы хранимых процедур

В SQL Server имеется несколько типов хранимых процедур.

- Системные хранимые процедуры предназначены для выполнения различных административных действий. Практически все действия по администрированию сервера выполняются с их помощью. Можно сказать, что системные хранимые процедуры являются интерфейсом, обеспечивающим работу с системными таблицами, которая, в конечном счете, сводится к изменению, добавлению, удалению и выборке данных из системных таблиц как пользовательских, так и системных баз данных. Системные хранимые процедуры имеют префикс `sp_`, хранятся в системной базе данных и могут быть вызваны в контексте любой другой базы данных.

- Пользовательские хранимые процедуры реализуют те или иные действия. Хранимые процедуры – полноценный объект базы данных. Вследствие этого каждая хранимая процедура располагается в конкретной базе данных, где и выполняется.

- Временные хранимые процедуры существуют лишь некоторое время, после чего автоматически уничтожаются сервером. Они делятся на локальные и глобальные. Локальные временные хранимые процедуры могут быть вызваны только из того соединения, в котором созданы. При создании такой процедуры ей необходимо дать имя, начинающееся с одного символа `#`. Как и все временные объекты, хранимые процедуры этого типа автоматически удаляются при отключении пользователя, перезапуске или остановке сервера. Глобальные временные хранимые процедуры доступны для любых соединений сервера, на котором имеется такая же процедура. Для ее определения достаточно дать ей имя, начинающееся с символов `##`. Удаляются эти процедуры при перезапуске или остановке сервера, а также при закрытии соединения, в контексте которого они были созданы.

Создание, изменение и удаление хранимых процедур

Создание хранимой процедуры предполагает решение следующих задач:

- определение типа создаваемой хранимой процедуры: временная или пользовательская. Кроме этого, можно создать свою собственную системную хранимую процедуру, назначив ей имя с префиксом `sp_` и поместив ее в системную базу данных. Такая процедура будет доступна в контексте любой базы данных локального сервера;
- планирование прав доступа. При создании хранимой процедуры следует учитывать, что она будет иметь те же права доступа к объектам базы данных, что и создавший ее пользователь;

- определение параметров хранимой процедуры. Подобно процедурам, входящим в состав большинства языков программирования, хранимые процедуры могут обладать входными и выходными параметрами;

- разработка кода хранимой процедуры. Код процедуры может содержать последовательность любых команд SQL, включая вызов других хранимых процедур.

Создание новой и изменение имеющейся хранимой процедуры осуществляется с помощью следующей команды:

```
{CREATE | ALTER } PROC[EDURE] имя_процедуры [;номер]
[{ @имя_параметра тип_данных } [VARYING ] [=default][OUTPUT] ][,...n]
[WITH { RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION } ]
[FOR REPLICATION]
AS
sql_оператор [...n]
```

Удаление хранимой процедуры осуществляется командой:

```
DROP PROCEDURE {имя_процедуры} [...n]
```

Выполнение хранимой процедуры

Для выполнения хранимой процедуры используется команда:

```
[[ EXECUTE] имя_процедуры [;номер]
[[@имя_параметра=]{значение | @имя_переменной}
[OUTPUT ]][DEFAULT ]][,...n]
```

Если вызов хранимой процедуры не является единственной командой в пакете, то присутствие команды EXECUTE обязательно. Более того, эта команда требуется для вызова процедуры из тела другой процедуры или триггера.

Использование ключевого слова OUTPUT при вызове процедуры разрешается только для параметров, которые были объявлены при создании процедуры с ключевым словом OUTPUT.

Когда же при вызове процедуры для параметра указывается ключевое слово DEFAULT, то будет использовано значение по умолчанию. Естественно, указанное слово DEFAULT разрешается только для тех параметров, для которых определено значение по умолчанию

Из синтаксиса команды EXECUTE видно, что имена параметров могут быть опущены при вызове процедуры. Однако в этом случае пользователь должен указывать значения для параметров в том же порядке, в каком они перечислялись при создании процедуры. Присвоить параметру значение по умолчанию, просто пропустив его при перечислении нельзя. Если же требуется опустить параметры, для которых определено значение по умолчанию, достаточно явного указания имен параметров при вызове хранимой процедуры. Более того, таким способом можно перечислять параметры и их значения в произвольном порядке.

Отметим, что при вызове процедуры указываются либо имена параметров со значениями, либо только значения без имени параметра. Их комбинирование не допускается.

Понятие триггера

Триггер – это подпрограмма, похожая на процедуру БД, автоматически вызываемая СУБД при изменении, удалении или добавлении записи в таблице. К триггерам невозможно обратиться из программы, передать им параметры или получить от них результат. Наиболее часто триггеры применяются для поддержания ссылочной целостности и каскадных операций в БД. Ссылочные спецификации, определяющие каскадные действия при удалении и обновлении и создаваемые при объявлении таблиц, также реализуются через триггеры, однако текст этих триггеров не редактируется.

Назначение триггеров

- Предотвращение изменения (например, предотвратить счета от изменений после того как они разосланы).
- Журналирование изменения (например, хранить копии старых данных).
- Аудит изменений (например, вести лог пользователей и ролей, участвующих в изменениях).
- Фиксация изменений (например, обеспечить, чтобы все изменения датировались по часам сервера, а не клиента).
- Реализация бизнес-правил.
- Репликация данных (например, хранить записи всех изменений, которые будут отправлены в другой базе данных более поздней версии).
- Повышение производительности (например, обновление сальдо после каждой детали сделки, для ускорения запросов).

Объявление триггеров

```
CREATE TRIGGER <имя триггера> {BEFORE|AFTER}  
{DELETE|INSERT|UPDATE [OF <столбцы>]} ON <имя  
таблицы> REFERENCING {OLD {[ROW]|TABLE [AS] <старое имя>} NEW {[ROW|TABLE]  
[AS] <новое имя>}} [FOR EACH {STATEMENT|ROW [WHEN <предикат>]}}]  
[BEGIN ATOMIC]  
<SQL-операторы>  
[END]
```

Ключевые слова

- **BEFORE|AFTER** – время запуска триггера – до | после операции обновления.
- **DELETE|INSERT|UPDATE** [**OF** <столбцы>] = событие срабатывания триггера.
- **FOR EACH ROW** – для каждой строки (строчный триггер, тогда и **WHEN**).

- FOR EACH STATEMENT – для всей команды (действует по умолчанию).
- REFERENCING – позволяет присваивать до 4-х псевдонимов старым и | или новым строкам и | или таблицам, к которым могут обращаться триггера.

Ограничения триггеров

Тело триггера не может содержать операторов:

- Определения, удаления и изменения объектов БД (таблиц, доменов и т.п.)
- Обработки транзакций (COMMIT, ROLLBACK)
- Подключения и отключения к БД (CONNECT, DISCONNECT)

Особенности применения

- Триггер выполняется после применения всех других (декларативны) проверок целостности и целесообразен тогда, когда критерий проверки достаточно сложен. Если декларативные проверки отклоняют операцию обновления, то до выполнения триггеров дело не доходит. Триггер работает в контексте транзакции, а ограничение FK нет.

- Если триггер вызывает дополнительную модификацию своей базовой таблицы, то чаще всего это не приводит к его рекурсивному выполнению, однако это следует уточнять. В СУБД SQL Server 2005 предусмотрена возможность указания рекурсии до 255 уровней с помощью ключевого слова OPTION (MAXRECURSIV 3).

- Триггеры обычно не выполняются при обработке больших двоичных столбцов (BLOB). Следует помнить, что всякий раз при обновлении данных СУБД автоматически создает так называемые триггерные виртуальные таблицы, которые в различных СУБД носят разные название. В InterBase и Oracle – Это New и Old. В SQL Server – Inserted и Deleted. Причем при изменении данных создаются обе. Эти таблицы имеют то же количество столбцов, с теми же именами и доменами, что и обновляемая таблица. В СУБД SQL Server 2005 предусмотрена возможность указания таблицы, включая временную, в которую следует вставить данные с помощью ключевого слова OUTPUT Inserted.ID,... INTO @.

- В ряде СУБД допустимо объявлять триггеры для нескольких действий одновременно. Для реализации разных реакций на различные действия в Oracle предусмотрены предикаты Deleting, Inserting, Updating, возвращающие True для соответствующего вида обновления.

- В СУБД Oracle можно для триггеров Update указать список столбцов (After Update Of), что обеспечит вызов триггера только при изменении значений только этих столбцов.

- Для каждого триггерного события может быть объявлено несколько триггеров (в Oracle 12 триггеров на таблицу) и обычно порядок их запуска определяется порядком создания. В некоторых СУБД, например, InterBase, порядок запуска указывается с помощью дополнительного ключевого слова POSITION . В общем случае считается, что первоначально должны выполняться триггеры для каждой команды, а затем – для каждой строки.

- Триггеры можно встраивать друг в друга. Так SQL Server допускает 32 уровня вложения (с помощью глобальной переменной @@NextLevel можно определить уровень вложения).

Недостатки триггеров

- 1 Сложность. Размещение некоторых действий над данными в БД усложняет ее проектирование, реализацию и администрирование.
- 2 Скрытность функциональных возможностей от пользователя. Трудно производить модернизацию приложения, когда скрыты некоторые возможности.
- 3 Влияние на производительность. При небольшом числе триггеров увеличивается время обработки данных.

Изменение и удаление триггеров

- Для удаления триггера используется оператор DROP TRIGGER <имя триггера>
- Для изменения триггера используется оператор ALTER TRIGGER <имя триггера>...
- Отключение триггеров

В ряде случаев, например, при пакетной загрузке, триггеры требуется отключать. В ряде СУБД предусмотрены соответствующие возможности. В Oracle и SQL Server ключевые слова DISABLE|ENABLE, в InterBase INACTIVE|ACTIVE в операторе ALTER TRIGGER.

Тема 4.3. Формирование и вывод отчетов

Отчет – это объект базы данных, который предназначен для вывода информации из баз данных, прежде всего на принтер. Отчеты позволяют выбрать из баз данных нужную пользователю информацию, оформлять ее в виде документа, перед выводом на печать просмотреть на экране. Источником данных для отчета может служить таблица или запрос. Кроме данных, полученных из таблиц, в отчете могут отображаться вычисляемые поля, например, итоговые суммы.

Виды отчетов:

- *Одноколонный отчет* (в столбец) – длинный столбец текста, содержащий надписи полей, их значения из всех записей таблицы или запроса;
- *Многоколонный отчет* – создается из отчета в одну колонку и позволяет вывести данные отчета в несколько колонок;
- *Табличный отчет* – отчет, имеющий табличную форму;
- *Отчет с группировкой данных и подведением итогов* - создается из табличного отчета объединением данных в группы с подсчетом итогов;
- *Перекрестный отчет* – строится на основе перекрестных запросов и содержит итоговые данные;
- *Составной отчет* – отчет, имеющий сложную структуру, включающий один или несколько отчетов;

- *Составной документ* - отчет, полученный слиянием документов с Word;
- *Почтовые наклейки* – специальный тип многоколонного отчета, предназначенный для печати имен и адресов групп;

- *Отчет по связанным таблицам* – отчет, позволяющий выводить данные из нескольких таблиц, имеющих связь один – ко - многим.

Структура отчета.

Отчет может состоять из следующих разделов:

- *Заголовок отчета* – отображается только на первой странице и определяет высоту области заголовка отчета, который может содержать текст, графику и другие элементы управления;
- *Верхний колонтитул* – выводится наверху каждой страницы; как правило, содержит заголовки столбцов;
- *Заголовок группы* – печатается перед первой записью, входящей в группу; обычно содержит итоговые данные по записям, входящим в группу;
- *Область данных* – предназначена для отображения записей источника данных отчета;
- *Область примечания группы* – отображается после обработки последней записи группы; обычно содержит итоговые данные по записям, входящими в группу;
- *Нижний колонтитул* – выводится внизу каждой страницы; содержит, например, номер страницы отчета, дату печати отчета и т.д.;
- *Раздел примечаний* - отображается только внизу последней странице; помещают поля с итоговыми значениями по всем записям, включенным в отчет.

Отчет может быть создан с помощью пиктограмм группы *Отчеты* вкладки СОЗДАНИЕ:

- пиктограммы *Отчет* – создать можно простой отчет на основе данных текущего запроса или таблицы, к которому можно добавить дополнительные компоненты, например группы или итоги;
- пиктограммы *Пустой отчет* – создать можно пустой отчет в который можно добавить поля и элементы управления;
- пиктограммы *Мастер отчет* – программное средство, которое позволяет создавать отчет в диалоге с пользователем;
- пиктограммы *Конструктора* – создание нового пустого отчета в режиме конструктора.

Обычно при создании отчета используют следующие способы: Мастер отчетов позволяет ускорить процесс создания отчета, работа в нем производится в пошаговом режиме в диалоге с пользователем и доработать созданный мастером отчет можно в режиме Конструктора.

Раздел *Область данных* может содержать вычисляемые поля, предназначенные для отображения в отчетах значений выражений на основе исходных данных.

В режиме Конструктора доступны такие свойства отчета как группировка, сортировка и поиск данных, которые вызываются пиктограммой Группировка группы *Группировка и итоги*. В диалоговом окне *Группировка, Сортировка и итоги* можно определить поле или выражение, по которому будут сгруппированы данные, а также сгруппированные данные можно отсортировать по возрастанию или убыванию. *Access* позволяет группировать данные двумя способами – по категориям и по диапазону значений, которое может быть как числовым, так и алфавитным.

Создание отчета с помощью мастера

Отчеты создаются при выборе объекта Отчеты в главном окне базы данных и нажатии на кнопку Создать, расположенную на панели инструментов окна базы данных. В появившемся диалоговом окне Новый отчет, необходимо выбрать режим создания отчета и указать источник данных.

Предпочтительнее создавать отчет с помощью мастера, так как он поможет выполнить многие этапы построения отчета:

1. На вкладке Создание нужно выбрать Мастер отчётов.
2. Переход на первый шаг работы мастера. Открывается знакомое окно для выбора полей.

Необходимо указать в нем таблицу или запрос.

3. Открывается окно второго шага, где необходимо установить уровень группировки, то есть если необходимо вывести все данные по наименованию товара, то первым надо выбрать поле Товар, если же вы отдаете предпочтение цене, первым полем будет ЦенаПоставки. Остальные поля группируются внутри первого поля.

4. На третьем шаге мастер предлагает выполнить сортировку записей в отчете. В этом же шаге производится подведение итогов, поэтому возвращаться и специально создавать итоговый запрос не надо. Кнопка Итоги откроет окно подведения итогов, где нужно поставить галочку напротив поля с указанием итоговой операции. При нажатии кнопки ОК произойдет возврат в окно сортировки.

5. Четвертый шаг определяет вид отчета.
6. Пятый шаг служит для сохранения отчета.

Доработка отчета в режиме Конструктора

В заголовок помещается текст из шапки макета отчета. После работы мастера текст заголовка можно и нужно изменить, так как, задавая имя отчету на последнем шаге, обычно придерживаются правил файловой системы, а заголовок требует выполнения правил правописания, поскольку является текстом. В зоне заголовка можно поместить небольшой сопроводительный текст. Это может быть сообщение или приказ. Зона заголовка для дополнительного текста увеличивается и так же, как в форме, выполняется с помощью инструмента Надпись. Если же текст большой, лучше экспортировать отчет в Word.

В верхний и нижний колонтитулы обычно помещают заголовки полей, номера страниц и даты. При определении содержания этих разделов следует исходить из требований к оформлению отдельных страниц отчета. Названия полей могут быть непонятными для тех, кто будет читать отчет, поэтому их можно изменять точно так же, как и присоединенные надписи в форме.

В области данных размещаются поля таблиц базы данных или запросов. Они являются связанными, поэтому изменять их нельзя.

Если в режиме Мастер проведена группировка записи по полю, окно конструктора отчетов имеет дополнительные разделы: Заголовок группы и Примечание группы. В первом, как правило, размещаются поля, по которым производится группировка. В разделе Примечание группы могут быть расположены выражения для подведения итогов по группе. Допускается до 10 уровней группировки выводимых записей.

Элементами разделов отчета, кроме полей таблиц или запросов, на которых строится отчет, являются также тексты подписей, кнопки управления, внедряемые объекты, линии, прямоугольники и т.п. Для них имеются соответствующие кнопки на панели элементов. Свойства каждого элемента, а также раздела и отчета в целом, могут быть уточнены. Технология размещения элементов и определения их свойств практически такая же, как и в конструкторе форм.

Создание краткого отчета с группировкой или сортировкой

Даже если вам впервые приходится создавать отчеты с группировкой, вы можете быстро создать простой отчет, выполнив следующие действия:

1. в области навигации выберите таблицу или запрос с записями, которые вы хотите поместить в отчет;

2. на вкладке **Создание** щелкните **Отчет**.

Access создаст простой табличный отчет, а затем отобразит его в режиме макета. Если в отчете много полей, вероятно, отчет займет больше страницы. Прежде чем применять группировку или сортировку, вам может понадобиться изменить размер столбцов или удалить нежелательные, чтобы отчет помещался на одной странице. Чтобы удалить столбец, щелкните его правой кнопкой мыши, а затем выберите **Удалить столбец**;

3. щелкните столбец, к которому вы хотите применить сортировку или группировку, затем нажмите **Группировка** [имя поля] или выберите один из параметров в меню **Сортировка**. К примеру, чтобы применить группировку к приоритетному столбцу, щелкните его правой кнопкой мыши и выберите пункт **Группировать по приоритету**.

Во время группировки Access перемещает поле группировки в крайний левый столбец, а затем группирует по нему остальные столбцы. В некоторых случаях Access добавляет в раздел колонтитула отчета общий итог;

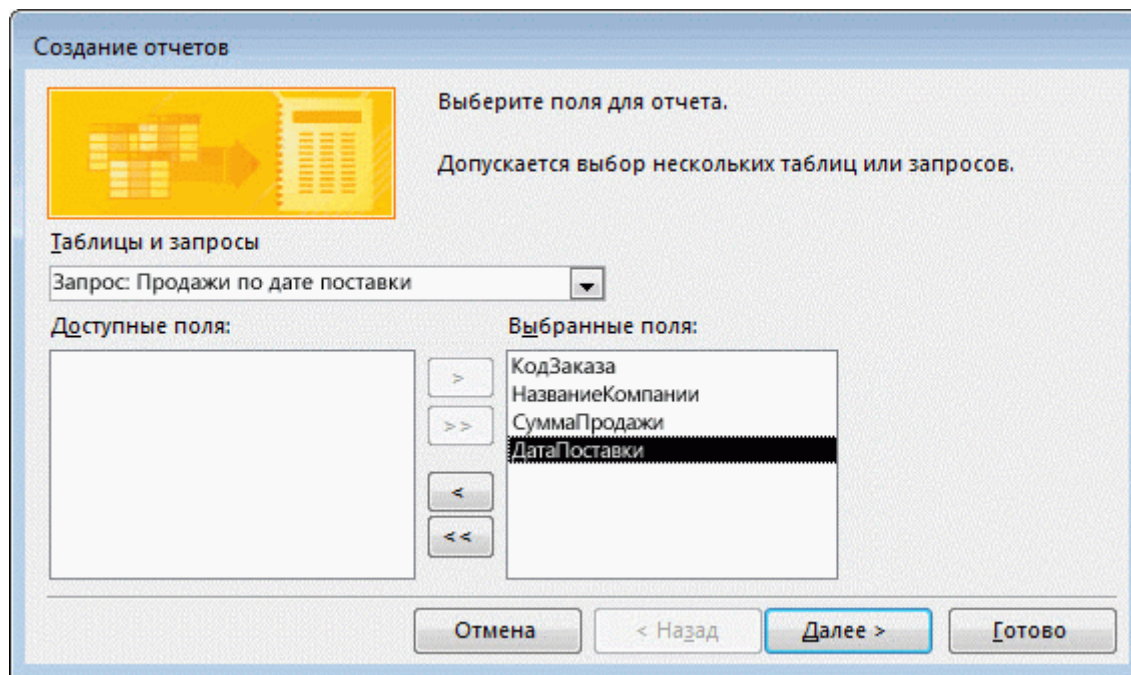
Создание отчета с группировкой при помощи мастера отчетов

Мастер отчетов задает вам вопросы и на основе ответов создает отчет. В одном из них упоминаются поля, по которым нужно выполнить группировку в отчете. После создания отчета вы можете использовать его в готовом виде или изменить в соответствии со своими потребностями. Прежде чем запускать мастер отчета, определитесь с выбором источника данных.

Запуск мастера отчетов

1. На вкладке **Создание** в группе **Отчеты** выберите **Мастер отчетов**.

Access запустит **мастер отчетов**.



2. Щелкните раскрывающийся список **Таблицы и запросы** и выберите таблицу или запрос с полями, которые вы хотите добавить в отчет.

3. Чтобы выбрать поля, дважды щелкните их в списке **Доступные поля**.

Access переместит их в список **Выбранные поля**. Вы также можете нажать кнопки, находящиеся между полями **Доступные поля** и **Выбранные поля**, чтобы добавить или удалить выбранное поле или сразу все поля.

4. Если вы хотите добавить в отчет также поля из другой таблицы или из другого запроса, щелкните еще раз раскрывающийся список **Таблицы и запросы**, выберите нужное и продолжайте добавлять поля.

5. После того, как вы завершите добавление полей, нажмите **Далее**.

Группировка записей при помощи мастера отчетов

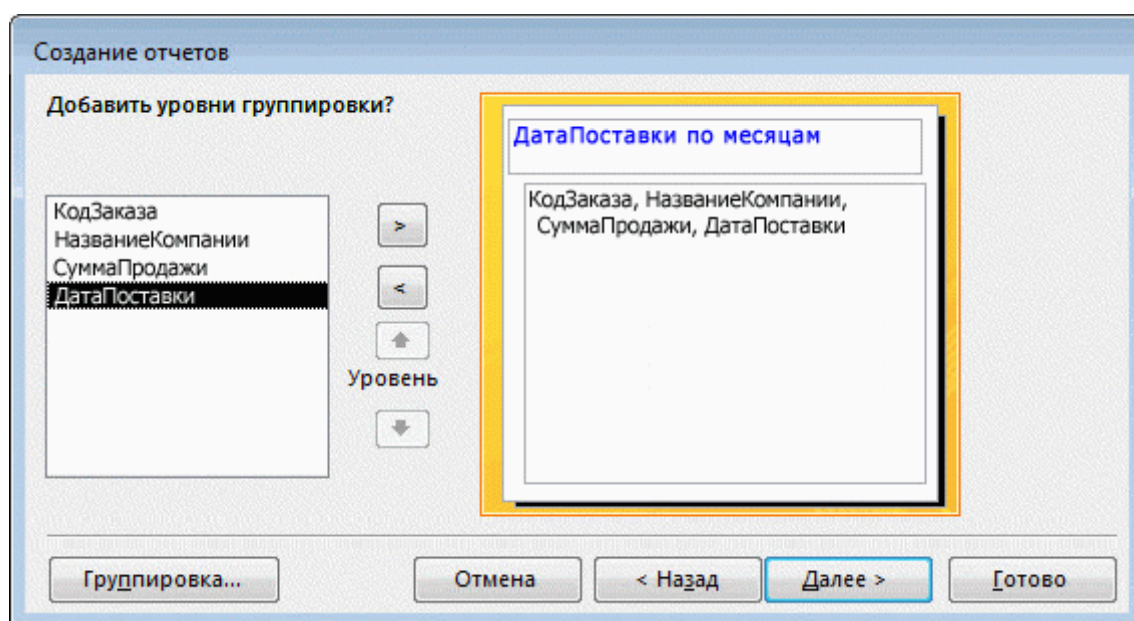
Группировка позволяет упорядочить записи по группам (например, по регионам или продавцам). Вложенные группы дают возможность легко определить отношения между группами и быстро найти нужные данные. С помощью группировки вы также можете получить сводные данные, такие как итоговые значения или проценты.

Когда вы включаете в отчет несколько таблиц, мастер проверяет отношения между ними и определяет возможный способ просмотра данных.

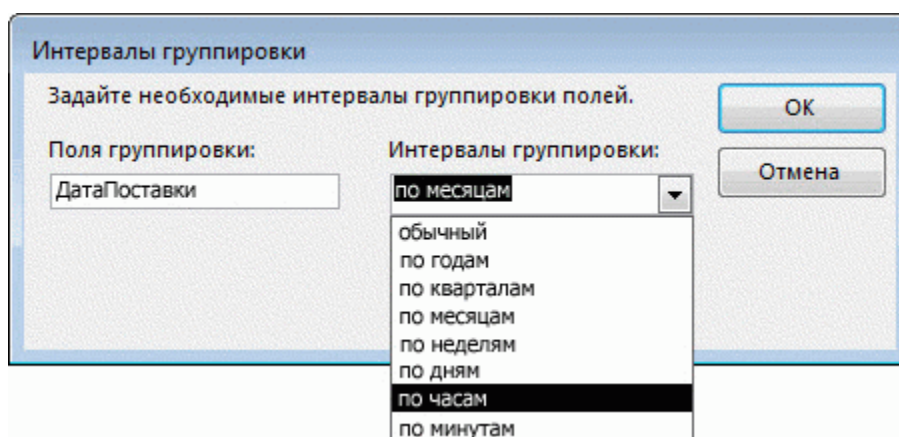
1. На странице мастера отчетов с вопросом **Добавить уровни группировки?** щелкните одно из полей в списке и нажмите **Далее**.

2. Чтобы добавить уровни группировки в отчет, дважды щелкните любое из добавляемых в отчет полей в списке.

Вы также можете удалить уровень группировки, дважды щелкнув его на отображении страницы в правой части диалогового окна. С помощью клавиш со стрелками добавьте или удалите уровни группировок. Назначьте низкий или высокий приоритет для уровня группировки, выбрав его и нажав соответствующие кнопки. Access добавляет каждый уровень группировки и показывает, что он вложен в родительский уровень группировки.



3. Нажмите кнопку **Параметры группировки**, чтобы открыть диалоговое окно **Интервалы группировки**.



4. При желании для каждого поля группировки выберите интервал группировки.

Интервал группировки позволяет настраивать способ группировки записей. На предыдущем рисунке записи группируются по полю "ДатаПоставки" с типом данных "Дата и время". Мастер

отчетов предлагает варианты, соответствующие типу поля из списка **Интервалы группировки**. Так как тип поля "ДатаПоставки" — "Дата и время", вы можете группировать по действительному значению (**обычный**), по годам, кварталам, месяцам, неделям, дням, часам и минутам. Если бы поле имело текстовый тип, вы могли бы группировать по всему полю (**обычный**) или, к примеру, по первым пяти символам. В случае числового типа данных вы можете группировать по значению (**обычный**) или по возрастанию в выбранном диапазоне.

Выбрав интервал группировки, нажмите **ОК**.

5. Чтобы перейти к следующей странице мастера, нажмите **Далее**.

Сортировка записей и подведение итогов по ним

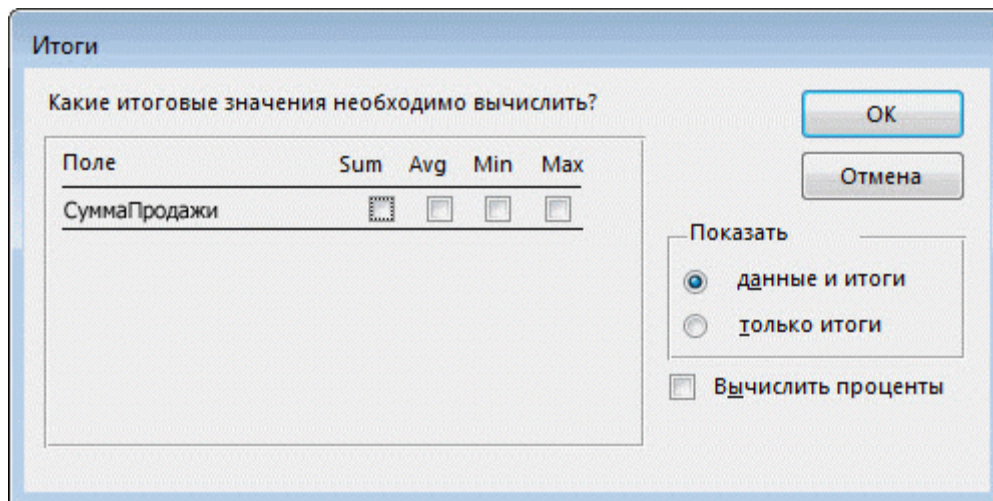
Вы можете сортировать записи в порядке возрастания или убывания по 1–4 полям.

1. Щелкните первый раскрывающийся список и выберите поля для сортировки.

Вы можете нажать кнопку в правой части списка, чтобы выбрать сортировку по возрастанию или убыванию (сортировка **по возрастанию** используется по умолчанию). При необходимости щелкните второй, третий и четвертый раскрывающийся список, чтобы выбрать дополнительные поля для сортировки.

2. Нажмите кнопку **Итоги**, если вам нужно вычислить итоговые значения числовых полей.

Обратите внимание, что кнопка **Итоги** будет доступной лишь в случае, если в разделе подробностей отчета содержится одно или несколько числовых полей. Мастер отображает доступные числовые поля.



3. Установите флажок рядом с **Sum**, **Avg**, **Min** или **Max**, чтобы включить расчеты в примечание группы.

Вы также можете выбрать для отображения подробности и сводку или только сводку. В последнем случае итоги отображаются для каждого значения поля "ДатаПоставки" (например, если вы установили флажок **Sum**), но подробности о заказе опускаются. Вы также можете выбрать для сумм отображение итоговых расчетов в процентах.

4. Нажмите кнопку **ОК**.

5. Следуйте инструкциям на остальных страницах мастера отчетов. На последней странице вы можете изменить заголовок отчета. Этот заголовок будет отображаться на первой странице отчета, и Access сохранит отчет, используя этот заголовок в качестве имени документа. В дальнейшем вы сможете изменить как заголовок, так и имя документа.

6. Нажмите кнопку **Готово**. Access автоматически сохранит отчет и отобразит его в режиме предварительного просмотра, в котором вы сможете посмотреть, как будет выглядеть распечатанный отчет.

С помощью кнопок навигации в нижней части области просмотра вы можете последовательно просматривать страницы отчета или перейти к любой его странице. Нажмите одну из кнопок навигации или введите номер нужной страницы в поле ввода номера страницы, а затем нажмите клавишу ВВОД.

В режиме предварительного просмотра вы можете увеличивать масштаб, чтобы просмотреть подробности, или уменьшать его, чтобы просмотреть расположение данных на странице. Щелкните один раз, когда курсор мыши будет указывать на отчет. Чтобы отменить эффект увеличения, щелкните еще раз. Вы также можете использовать элемент управления масштабом в строке состояния.

Предварительный просмотр отчета

Если вы хотите увидеть, как будет выглядеть отчет при печати, используйте режим предварительного просмотра. При предварительном просмотре отчета, созданного в Access, кнопки навигации по записям отображаются в расположении по умолчанию в нижней части окна Access.

Начало просмотра печати

1. Если отчет не открыт, в области навигации щелкните правой кнопкой мыши отчет, который вы хотите просмотреть, и выберите в контекстном меню команду **предварительный Просмотр**.

ИЛИ

Если отчет открыт, щелкните правой кнопкой мыши вкладку документа для отчета и выберите **предварительный Просмотр**.

2. Нажимайте кнопки перехода по записям, чтобы перемещаться между страницами отчета.

Предварительный просмотр нескольких страниц

Если отчет открыт, щелкните правой кнопкой мыши вкладку документа для отчета и выберите **предварительный Просмотр**.

1. На вкладке **предварительный Просмотр** в группе **масштаб** нажмите кнопку **две страницы**, чтобы просмотреть отчет по двум страницам.

ИЛИ

В группе **масштаб** нажмите кнопку **Дополнительные страницы** и выберите нужный вариант.

2. Нажимайте кнопки перехода по записям, чтобы перемещаться между страницами отчета.

3. Чтобы вернуться к предварительному просмотру одной страницы, в группе **масштаб** выберите **одна страница**.

Заккрытие окна предварительного просмотра

Вы можете либо напечатать отчет после предварительного просмотра, либо закрыть его, чтобы внести изменения в отчет перед печатью.

- На вкладке **предварительный Просмотр** в группе **Заккрыть предварительный Просмотр** нажмите кнопку **Заккрыть окно предварительного просмотра**.

Быстрая печать отчета

Если вы хотите напечатать все страницы отчета без изменения формата отчета, например, полей, колонтитулов и ориентации (книжная или альбомная), вы можете использовать функцию быстрой печати.

1. Если отчет не открыт, в области навигации щелкните отчет правой кнопкой мыши и выберите команду **Печать**.

ИЛИ

Убедитесь в том, что кнопка **Быстрая печать** доступна на панели быстрого доступа. Чтобы добавить кнопку на панель инструментов, щелкните стрелку вниз в конце панели инструментов и выберите пункт **Быстрая печать**.

На панели быстрого доступа нажмите кнопку **Быстрая печать**.

Печать страницы "Выбор страниц" в отчете

В отчете можно ограничить возможности печати, выбирая диапазон страниц.

1. Откройте вкладку **файл** и выберите команду **Печать**.
2. В области **Печать** параметров файла нажмите кнопку **Печать**, чтобы открыть диалоговое окно **Печать**.
3. Чтобы напечатать только некоторые данные отчета, в разделе **диапазон печати** выберите пункт **страницы**, а затем введите значения в поля **от** и **до**.

Например, чтобы напечатать только страницу 5, введите **5** в поля **С** и **По**.

Если вы владеете необходимыми разрешениями для изменения макета отчета, вы можете внести дополнительные изменения в способ печати отчета, изменив макет печати в диалоговом окне **Параметры страницы**.

Изменение параметров страницы

Диалоговое окно "**Параметры страницы**" можно использовать для изменения различных параметров печати, в том числе сверху и снизу, а также для печати только данных в отчете. В Access есть два способа отображения диалогового окна **Параметры страницы**. Вы можете запустить его в диалоговом окне **Печать** или на вкладке **Предварительный просмотр**, которое появляется при запуске предварительный просмотр.

Открытие диалогового окна "Параметры страницы" в диалоговом окне "Печать"

1. В области навигации выберите отчет, который вы хотите изменить. Вам не нужно открывать отчет, несмотря на то что вы можете сделать так, чтобы данные, которые вы хотите напечатать, были актуальными.
2. Откройте вкладку **файл** и выберите команду **Печать**.
3. В области **Печать** параметров файла нажмите кнопку **Печать**, чтобы открыть диалоговое окно **Печать**.
4. Нажмите кнопку "**Настройка**", чтобы открыть диалоговое окно "**Параметры страницы**".
5. При необходимости задайте или измените параметры в диалоговом окне.

Открытие диалогового окна "Параметры страницы" на вкладке "предварительный просмотр"

1. В области навигации щелкните правой кнопкой мыши отчет, который нужно изменить, и выберите команду **предварительный Просмотр**.

Access откроет отчет в режиме предварительного просмотра.

2. На вкладке **предварительный Просмотр** в группе **Макет страницы** нажмите кнопку **Параметры страницы**.

3. При необходимости задайте или измените параметры в диалоговом окне.

Выбор параметров печати

Если вам нужно напечатать некоторые страницы отчета или выбрать принтер, или изменить параметры печати, например, поля или столбцы, выполните действия, описанные в этом разделе.

1. В области навигации выберите отчет, который нужно распечатать.

2. Откройте вкладку **файл** и выберите команду **Печать**.

3. В области **Печать** параметров файла нажмите кнопку **Печать**, чтобы открыть диалоговое окно **Печать**.

4. При необходимости измените параметры форматирования страницы и другие параметры печати или выбор принтера, а затем нажмите кнопку **ОК**, чтобы распечатать отчет.

Список использованных источников

1. Кузнецов, С. Д. Основы баз данных / С.Д. Кузнецов. - М.: Бином. Лаборатория знаний, Интернет-университет информационных технологий, **2017**. - 488 с.
2. http://citforum.ru/database/advanced_intro/ - С. Кузнецов Базы данных. Вводный курс
3. <https://studentbank.ru/view.php?id=9582> – Организация баз данных – курс лекций