МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра охраны труда и окружающей среды

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«14» 02 2018 г.

ПРОЕКТИРОВАНИЕ СТРУКТУР БАЗ ДАННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Методические указания к проведению
лабораторных и практических работ
для студентов направлений подготовки 20.03.01, 20.04.01
«Техносферная безопасность»

Курск 2018

УДК 371.64/.69:004
Составители: И.О. Кирильчук, А.В. Гнездилова

Рецензент
Кандидат технических наук, доцент *Г.П. Тимофеев.*

Проектирование структур баз данных информационных систем: методические указания к проведению лабораторных и практических работ / Юго-Зап. гос. ун-т; сост.: И.О. Кирильчук, А.В. Гнездилова. Курск, 2018. 27с.

Излагаются основные положения теории реляционных баз данных, и даются навыки грамотного планирования баз данных систем обработки информации.

Методические указания предназначены для студентов направлений подготовки 20.03.01, 20.04.01 Техносферная безопасность.

Текст печатается в авторской редакции

Подписано в печать 14.02.18. Формат 60х84 1/16.
Усл. печ. л. 1,56.Уч.-изд.л.1,42. Тираж 30 экз. Заказ 1070. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Цель работы:

– изучение основных положений теории реляционных баз данных; получение навыков грамотного планирования баз данных информационных систем.

1 Общие положения

1.1 Структурные элементы базы данных

Целью любой информационной системы является хранение и обработка данных о каких-либо объектах. В широком смысле слова база данных – это совокупность сведений о конкретных объектах. При создании базы данных (БД) в основном преследуется цель упорядочить данные по различным признакам, чтобы иметь возможность быстро извлекать нужную информацию. В современной технологии баз данных предполагается, что создание БД, ее поддержка, управление, а также доступ пользователей к самим данным осуществляется с помощью специальных программных продуктов – *систем управления базами данных* (СУБД).

В терминологии баз данных описываемые объекты часто называют *сущностями* (entity), а сами данные – *атрибутами* (attribute). Объект (или сущность) – это нечто, существующее в реальном мире и различимое, что имеет название и имеется способ отличать один подобный объект от другого. Сущностью может являться ВУЗ, студент, ручка, аудитория и т.д. Кроме того, помимо физического предмета сущностью могут являться и абстрактные вещи, которые, тем не менее, могут быть описаны: лекция, телевизионная передача, правовые нормы, правила поведения и т.д. Группа всех подобных объектов образует *набор объектов*. Набором объектов для объекта "сотрудник" будут являться все сотрудники, работающие на предприятии.

Атрибут – это некоторый показатель, характеризующий объект и принимающий для каждого конкретного объекта индивидуальное значение (текстовое, числовое и т.п.). Если в качестве объекта рассмотреть сотрудника предприятия, то этот объект будет иметь атрибуты имени, фамилии и отчества, принимающие текстовые

значения, числовой атрибут номера подразделения, в котором он работает, атрибут даты рождения, имеющий тип, описывающий дату и время, и многие другие атрибуты. Атрибут может являться также и набором объектов. Например, атрибутом работника является номер подразделения, в котором он работает. С другой стороны, подразделение является экземпляром набора подразделений и описывается набором атрибутов: начальник подразделения, количество сотрудников, адрес, специализация и т.п. В свою очередь, некоторые из атрибутов подразделения сами могут являться экземплярами набора объектов.

Помимо понятий сущность и атрибут, при работе с базами данных часто используются понятия поле, запись, таблица. *Поле* (field) – это минимальная неделимая структура организации данных. Для каждого поля базы данных разработчик должен задать следующие значения:

- уникальное в пределах отношения *имя* (name);
- *тип данных*, которые будут храниться в поле (числовой, символьный, логический и т.д.);
- максимальная *длина* (или *размер*), который могут иметь данные, хранящиеся в поле;
- дополнительные характеристики (например, для числовых данных – точность, для символьных – формат, и т.д.).

Множество логически связанных полей образуют *запись* (record). Запись является ничем, как строкой таблицы. Экземпляр записи – это отдельная реализация записи, содержащая конкретные значения ее полей. Множество экземпляров записи одной структуры образуют *таблицу* (table). При описании таблицы указывается последовательность расположения полей и их основные характеристики (имя, длина и точность). Количество записей в таблице может меняться.

1.2 Реляционная модель данных

Модель данных (data model) включает в себя структуры данных, операции их обработки и ограничения целостности. За все время существования компьютеров для обработки данных было разработано

множество различных моделей данных. Наибольшее распространение получила реляционная модель данных, характеризующаяся простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

В 1970 г. сотрудником фирмы IBM доктором Э. Коддом было доказано, что любой набор данных может быть представлен в виде простых двумерных таблиц, называемых в математике *"отношением"* ("relation"). Разработанная Коддом теория реляционных баз данных описывает правила эффективной организации данных и управления ими.

Отметим, что для описания одних и тех же объектов в теории баз данных вообще и в реляционной теории баз данных в частности, а также при практической работе с СУБД используются разные термины. В таблице 1 приведено соответствие описания объектов в терминах разных областей.

Таблица 1

Соответствие терминов, применяемых при проектировании БД

Теория БД	Реляционные БД	СУБД
Сущность (Entity)	Отношение (Relation)	Таблица (Table)
Кортеж (Tuple)	Запись (Record)	Строка (Row)
Атрибут (Attribute)	Поле (Field)	Столбец, колонка (Column)

Итак, каждая *реляционная таблица* представляет собой *двумерный массив* и обладает следующими свойствами:

- любой элемент таблицы является минимальным элементом данных;
- элементы в пределах одного столбца имеют одинаковый тип (числовой, символьный и т.д.);
- все столбцы одной таблицы должны иметь уникальное имя;
- в таблице отсутствуют совпадающие строки;

– порядок следования строк и столбцов в таблице может быть произвольным.

Пример реляционной таблицы, содержащей номер личного дела сотрудника, его имя, фамилию, отчество, дату рождения и идентификационный номер подразделения, в котором он работает, приведен в таблице 2.

Таблица 2

Пример реляционной таблицы

№ дела	Фамилия	Имя	Отчество	Дата рожд.	№ подр.
19200	Иванов	Петр	Сергеевич	14.02.70	27
19133	Кузнецов	Владимир	Павлович	23.11.65	01
20457	Иванов	Игорь	Сергеевич	02.06.80	05
20458	Сидорова	Галина	Алексеевна	06.08.74	27
19987	Жуков	Петр	Николаевич	22.03.69	14

Как видно, каждый из атрибутов расположен в отдельном столбце, а каждая строка содержит набор атрибутов, описывающих конкретный экземпляр объекта. Одним из принципов реляционной теории является требование к минимальности элементов таблицы. Это означает, что каждое поле таблицы должно являться отдельным атомарным значением для конкретной предметной области. Под атомарным значением понимается значение, неделимое на более простые составляющие.

Однако следует отметить, что в некоторых случаях в качестве атомарного значения могут рассматриваться агрегированные объекты. Например, в одной предметной области адрес может являться атомарным значением и не разбиваться на более мелкие составляющие, тогда как в другой предметной области адрес будет представлен набором более конкретных значений: страна, район, город, улица, дом и квартира. Если в первом случае для адреса отводится единственный столбец таблицы, то во втором случае будет отведено шесть столбцов.

Доступ к конкретному элементу отношения может быть получен с указанием адреса этого элемента в формате A[i, j], где A – элемент данных, i – номер строки, j – номер столбца отношения.

Количество атрибутов в отношении определяет *порядок* (или *степень*) этого отношения. Порядок отношения, приведенного в таблице 2, равен 6.

Множество всех значений A[i, j] при постоянном i и всех возможных j образует *кортеж* или, попросту, строку таблицы.

Как уже говорилось, реляционные таблицы связываются друг с другом. Чтобы иметь возможность связать таблицы, необходимо иметь какой-то идентификатор, который позволял бы уникально идентифицировать любую строку таблицы. Таким идентификатором является *ключ* (key). Ключом называется множество атрибутов, задание значений которых позволяет однозначно определить значения остальных атрибутов таблицы. В качестве ключа для отношения, приведенного в таблице 2, может выступать атрибут "№ дела" или набор атрибутов "Фамилия", "Имя" и "Отчество". В первом случае в качестве ключа выступает единственный атрибут – "№ дела" (номер личного дела у каждого работника свой). Во втором случае ключ является сложным и состоит из трех атрибутов. Если в качестве ключа выбрать только фамилию, то возможна ситуация, когда по одной фамилии нельзя будет однозначно идентифицировать человека – это бывает в случае, когда на предприятии работают однофамильцы или родственники. Чтобы добиться уникальности, в ключ входит еще имя и отчество. Но и здесь не всегда можно гарантировать, что ключ будет уникален. Поэтому рекомендуется в качестве ключа выбирать заведомо неповторяющиеся атрибуты, например номер паспорта.

Множество атрибутов отношения является *возможным ключом* отношения тогда и только тогда, когда выполняются два независимых от времени условия:

– **Уникальность**. В любой момент времени никакие два кортежа не имеют одинаковых значений для всех атрибутов, входящих в возможный ключ. Остальные атрибуты могут иметь произвольные значения. Например, если в качестве возможного ключа выбран номер

личного дела, то необходимо гарантировать, что в отношении никогда не появится двух личных дел с одинаковыми номерами.

– **Минимальность**. Ни один из атрибутов не может быть исключен из ключа без нарушения уникальности возможного ключа. Это означает, что в ключ разрешается вносить только минимальный набор атрибутов, который позволяет идентифицировать каждую строку отношения. То есть нет необходимости помещать в ключ и номер личного дела, и номер паспорта. При создании возможного ключа на основе атрибутов "Фамилия", "Имя" и "Отчество" исключение любого из атрибутов резко снижает уникальность ключа. Кроме того, не стоит вносить в возможный ключ повторяющийся атрибут, не увеличивающий уникальность ключа. То есть нельзя включать дополнительно к номеру личного дела еще и фамилию служащего.

Один из возможных ключей отношения может быть выбран в качестве *первичного ключа*. Остальные возможные ключи, если оно есть, принимаются за *альтернативные*.

1.3 Проектирование реляционных баз данных

Реляционная база данных – это совокупность отношений, в которых хранится вся информация баз данных. Для пользователя такая база данных представляется набором двумерных таблиц, что облегчает понимание структуры данных и управление ими. Таблицы реляционной базы данных связаны между собой *отношениями*.

Замечание. Не нужно путать *отношения* (relationship) между таблицами, определяющие правила связывания данных, с термином *отношение* (relation), обозначающим таблицу. Если последний используется в теории реляционных баз данных при описании сущности, то первый – при практическом проектировании баз данных. Оба термина (relationship и relation) переводятся с английского одинаково – *отношение*.

Требования к проектированию реляционных баз данных в самом общем виде можно свести к нескольким правилам. Рассмотрим их.

– Каждая таблица в базе данных имеет уникальное имя в пределах этой базы данных. Это позволяет уникально идентифицировать данные в таблице.

– Все строки в таблице однотипны, т.е. количество, набор и последовательность полей в каждой строке одной таблицы одинаковы. Кроме того, количество значений в строке также фиксировано. Иначе говоря, в каждой позиции таблицы на пересечении любых строки и столбца имеется только одно значение (в т.ч. и пустое). Множественные поля и группы недопустимы.

– Строки таблицы обязательно отличаются друг от друга хотя бы единственным значением. То есть в любой момент времени в таблице не присутствует двух одинаковых строк, что позволяет однозначно идентифицировать каждую из них.

– Каждому столбцу таблицы присваивается уникальное в пределах таблицы имя. Любой из столбцов служит для хранения данных строго определенного типа (даты, числового, символьного, денежного и т.п.). Кроме этого столбец несет еще и смысловую нагрузку. Предположим, что имеется столбец, в котором указывается фамилия сотрудника. Хотя она и используется для хранения символьных данных, все же нельзя хранить в этом столбце имя или отчество.

– При обращении к данным можно свободно обращаться к любой строке или столбцу таблицы. Сами данные не налагают никаких ограничений на последовательность обращений к ним. Кроме того, ни одна строка таблицы не зависит от значений, хранимых в другой строке.

Описание столбцов таблицы называется макетом таблицы. Простейшее представление макета является ничем иным, как обычной двумерной таблицей, в которой каждая строка соответствует столбцу описываемой таблицы, а в каждой из колонок макета указываются характеристики описываемой таблицы. В таблице 3 представлен макет сущности, приведенной в таблице 2.

Макет таблицы позволяет наглядно представить, какое количество столбцов имеется в таблице и какие данные должны храниться в

каждом столбце. Заметим, что набор колонок в макете таблицы зависит от конкретной системы, в которой создается таблица. Кроме того, даже в пределах одной системы не всегда необходимо указывать все параметры.

Таблица 3

Макет таблицы

Имя поля (Field name)	Тип данных (data type)	Размер (size)	Ключ (key)	Пустое (Null)	По умолчанию (default)	Индекс (index)
№ дела	Целое	4	Да	Нет		Да
Фамилия	Текстовый	30	Нет	Нет		Да
Имя	Текстовый	30	Нет	Да		Нет
Отчество	Текстовый	30	Нет	Да		Нет
Дата рожд.	Дата/Время	8	Нет	Да	01.01.1900	Нет
№ подр.	Целое	4	Нет	Нет	42	Нет

1.4 Нормализация данных

Как было уже сказано, основная цель проектирования баз данных – это сокращение избыточности хранимых данных. Грамотно спланированная база данных обеспечивает оптимальное использование оперативной и дисковой памяти, предоставляет удобный механизм изменения данных и обеспечивает высокую их целостность. В неудачно спланированной базе данных может иметь место дублирование данных. Пользователи должны следить за всеми копиями данных, т.е. при изменении одной копии необходимо исправить и другие копии. В противном случае целостность данных нарушается. Например, если сотрудники предприятия учитываются в профсоюзе, отделе кадров и в отделе, где они работают, то необходимо следить, чтобы изменения отображались во всех местах. Если сотрудница выходит замуж и меняет фамилию, а информация изменяется лишь в одном месте, то получится, что в организации работает два разных человека. Эту и многие другие проблемы при проектировании базы данных можно решить с помощью нормализации.

Нормализация – это процесс приведения структур данных в состояние, обеспечивающее лучшие условия выборки, включения, изменения и удаления данных. Это достигается разбиением одной большой таблицы на две более мелкие таблицы. Конечной целью нормализации является получение такого проекта базы данных, в котором *каждый факт появляется лишь в одном месте*, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных.

Прежде чем приступить к описанию нормализации, нужно сказать об универсальном отношении. Таблица, в которую включены все интересующие атрибуты, называется *универсальным отношением*. При использовании универсального отношения база данных будет состоять из единственной таблицы, в которой станет располагаться вся информация. Количество атрибутов в таком отношении может быть очень большим. В таблице 4 приведен пример универсального отношения, содержащего информацию об итогах сдачи сессии студентами.

Таблица 4

Пример универсального отношения

Предмет	Преподаватель	Оценка	Фамилия	Имя	Отчество	ДатаРождения	МестоРождения	Группа
ГИС	Кирильчук	4	Иванов	Петр	Ильич	02.03.95	Курск	ТБ-11
ГИС	Кирильчук	3	Рогова	Ирина	Алексеевна	12.08.94	Льгов	ТБ-11
БЖД	Барков	4	Иванов	Петр	Ильич	02.03.95	Курск	ТБ-11
Экология	Юшин	5	Рогова	Наталья	Алексеевна	06.05.95	Льгов	ТБ-11
Экология	Юшин	4	Зуев	Игорь	Иванович	30.07.95	Курск	ТБ-21

Начинающий разработчик может использовать универсальное отношение, приведенное в таблице 4, в качестве завершенной БД. Действительно, зачем разбивать таблицу на несколько мелких, если и в

одной таблице можно с успехом хранить всю необходимую информацию. Однако применение универсального отношения связано с рядом проблем:

– **Избыточность**. Информация во многих столбцах многократно повторяется. Чем больше данных будет храниться в базе данных, тем больше информации дублируется и тем выше непроизводительные затраты.

– **Потенциальная противоречивость** (аномалии обновления). Вследствие наличия множества копий одних и тех же данных возможна ситуация, когда одна часть избыточных копий данных будет изменена, а другая – нет. Например, если для какой-либо дисциплины меняется преподаватель, то необходимо изменить информацию во всех строках, относящихся к этой дисциплине. Иначе дело будет выглядеть так, что один предмет ведут два преподавателя.

– **Аномалии включения**. В базу данных нельзя включить студента, если он не изучал ни одну из дисциплин (например, при переводе из другого ВУЗа). Однако часто бывает, что необходимо учесть все данные, которые могут потенциально использоваться.

– **Аномалии удаления**. При удалении дисциплины будет потеряна информация о преподавателе и студентах. Однако эти данные могут понадобиться в будущем, из-за чего потребуется их повторный ввод.

Большая часть проблем исчезнет, если данные из универсального отношения разнести в несколько мелких таблиц. Именно эту задачу и решает нормализация. Процесс нормализации разбивается на несколько этапов. На каждом из этапов структура данных должна удовлетворять определенным требованиям. Таблица считается *нормализованной на определенном уровне*, если она удовлетворяет требованиям, выдвигаемым соответствующей *формой нормализации* (*нормальной формы*).

При описании нормальных форм существуют несколько понятий:

– *Функциональной зависимостью* между полями X и Y называется зависимость, при которой каждому значению X в любой момент времени соответствует *единственное* значение Y из всех

возможных. Примером функциональной зависимости может служить связь города и страны, т.к. любой город находится в единственной стране и с течением времени эта связь не меняется.

– *Полной функциональной зависимостью* между составным полем X и полем Y называется зависимость, при которой поле Y зависит функционально от поля X и не зависит функционально от любого подмножества поля X.

– *Многозначная функциональная зависимость*. Поле X однозначно определяет поле Y, если для каждого значения поля X существует хорошо определенное множество соответствующих значений поля Y. Например, если рассматривать таблицу с описанием сотрудников организации и поле "Образование", то это поле имеет хорошо определенное множество допустимых значений (высшее, среднее, незаконченное среднее и т.д.).

– *Транзитивная функциональная зависимость* между полями X и Z наблюдается в том случае, если поле Y функционально зависит от поля X и поле Z функционально зависит от поля Y. В то же время не существует функциональной зависимости поля X от поля Y.

– Несколько полей *взаимно независимы*, если ни одно из них не является функционально зависимым от другого поля.

– *Неключевым* полем таблицы называется каждое поле, не входящее в состав первичного ключа.

Нормализация представляет собой последовательное изменение структуры данных и таблиц в соответствии с требованиями нормальных форм. Всего существуют шесть нормальных форм, однако на практике чаще всего применяются только три первых формы.

Первая нормальная форма. Считается, что таблица находится в первой нормальной форме, если ни одно поле строки не содержит более одного значения и любое ключевое поле не пусто. То есть ни один элемент таблицы не является, в свою очередь, таблицей и не содержит сложных данных. Не трудно заметить, что это требование перекликается с определением отношения, в котором говорится, что любой столбец состоит из атомарных значений, которые не могут быть разложены на более мелкие составляющие. То есть любая таблица в

реляционной базе данных автоматически находится в первой нормальной форме.

Вторая нормальная форма. Таблица находится во второй нормальной форме тогда и только тогда, когда она удовлетворяет требованиям первой нормальной формы и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом. То есть необходимо, чтобы только первичный ключ однозначно идентифицировал значения в любом столбце, и в то же время значения в столбцах не зависели ни от какой части составного ключа. Если первичный ключ состоит из одного столбца, то это требование удовлетворяется автоматически. Если же первичный ключ состоит из двух и более столбцов, то таблица необязательно будет находиться во второй нормальной форме. В этом случае таблица должна быть разбита на две или более таблиц таким образом, чтобы первичный ключ однозначно идентифицировал значение в любом столбце. Бывают ситуации, когда в таблице имеется поле, не зависящее от первичного ключа. В этом случае необходимо добавить в первичный ключ дополнительный столбец.

Третья нормальная форма. Таблица находится в третьей нормальной форме тогда и только тогда, когда она удовлетворяет требованиям второй нормальной формы и ни одно из ее неключевых полей не зависит функционально от любого неключевого поля. Любое неключевое поле должно зависеть только от значения первичного ключа и не зависеть от любого другого неключевого атрибута. То есть каждое неключевое поле нетранзитивно зависит от первичного ключа. Для устранения транзитивной зависимости между неключевыми полями выполняется расщепление исходной таблицы. В результате часть полей удаляется из исходной таблицы и включается в состав других (возможно вновь созданных) таблиц.

Структура данных и таблиц после проведения нормализации универсального отношения "ОценкиСтудентов", приведенного в таблице 4, показана на рисунке 1.

Рисунок 1 – Нормализованное универсальное отношение

1.5 Связывание таблиц

После нормализации таблиц может быть получен набор из множества таблиц. Данные, принадлежащие единственной *логической записи*, могут находиться в нескольких таблицах. Когда данные хранились в универсальном отношении, то можно было легко сразу получить всю необходимую информацию. После проведения нормализации доступ к данным несколько усложнился. Чтобы выбрать ту ли иную информацию, необходимо просмотреть данные в нескольких таблицах. При этом в качестве механизма, обеспечивающего связывание данных в разных таблицах, выступают ключи.

1.5.1 Первичный и внешний ключи

Связывание строк таблиц реляционной базы данных выполняется с помощью *первичного* (primary) и *внешнего* (foreign) ключей. Разработчик базы данных должен определить правила связывания

данных в разных таблицах, выделив в них одну или более колонок в качестве первичного или внешнего ключа. Напомним, что первичный ключ позволяет однозначно идентифицировать любую строку таблицы. При выборе столбцов, которые будут входить в состав первичного ключа, необходимо следовать требованиям *уникальности* и *минимальности*, описанным ранее. Также необходимо учитывать, что в первичный ключ не могут входить столбцы, для которых разрешено хранение пустых значений (Null).

В качестве первичного ключа могут быть использованы как уже имеющиеся столбцы таблицы, так и новые столбцы, специально созданные для этих целей. Хотя в качестве первичного ключа можно выбрать один или более атрибутов описываемого в таблице объекта, все же на практике часто создают специально выделенный новый столбец. Использование существующих атрибутов связано с определенными сложностями. Например, на первый взгляд кажется, что номер паспорта – хороший пример первичного ключа. Однако при связывании строк в разных таблицах необходимо будет каждый раз указывать этот номер. При этом длина ключа может быть сравнительно большой. Кроме того, человек может поменять паспорт, и тогда необходимо будет изменить данные во всех связанных таблицах. Использование специальных столбцов снимает эти проблемы.

Создание первичного ключа – лишь половина дела. Следующий шаг в связывании таблиц – определение *внешнего ключа*. Внешний ключ создается в таблице, поля которой ссылаются на строки главной таблицы. Для каждой строки зависимой таблицы необходимо, чтобы значению внешнего ключа было сопоставлено значение первичного ключа. То есть нельзя вставлять в зависимую таблицу строки со значением внешнего ключа, не определенного в главной таблице. Однако допускается, что значение внешнего ключа в зависимой таблице будет не определено, т.е. внешний ключ будет хранить значение Null. Впоследствии это значение может быть изменено на корректное значение, соответствующее значению внешнего ключа в главной таблице.

В отличие от первичного ключа, внешний ключ не должен быть уникальным. То есть в зависимой таблице может существовать

множество строк, имеющих одинаковые значения для полей, сконфигурированных в качестве внешнего ключа.

После того, как первичный и внешний ключи будут связаны, на данные в зависимой таблице будут наложены ограничения на значения полей, определенных в качестве внешнего ключа. При этом возникнет необходимость как-то согласовывать изменения ключевых полей, осуществляемые в главной таблице, со значениями в зависимой таблице. Если не выполнять никаких дополнительных действий, то возможно нарушение целостности данных. Например, если в качестве первичного ключа был выбран номер паспорта, и с ним было связано несколько внешних ключей разных таблиц, то при изменении у человека номера паспорта только в главной таблице связь станет нарушенной. Строки в зависимых таблицах окажутся *потерянными*, т.е. для них не будет сопоставлена ни одна строка главной таблицы. Это и есть нарушение целостности.

Во избежание подобных проблем в СУБД реализованы специальные механизмы, обеспечивающие автоматическую поддержку целостности данных. При попытке изменения (командой UPDATE) значения первичного ключа в главной таблице СУБД может вести себя следующим образом:

– **Установление (Relation)**. Когда значение первичного ключа главной таблицы изменяется, то автоматически устанавливаются значения внешних ключей во всех связанных строках в неопределенное значение (Null). При этом теряется информация о том, с какой строкой главной таблицы были связаны строки зависимой таблицы. При изменении в главной таблице более одной строки в зависимых таблицах может образоваться несколько наборов строк с неопределенным значением внешнего ключа. Определить, какая строка зависимой таблицы с какой строкой главной таблицы была связана, станет невозможно.

– **Ограничение (Restrict)**. В этом режиме будут отвергаться изменения значения первичного ключа, если в зависимой таблице имеется хоть одна строка, связанная с изменяемой строкой главной таблицы. Изменение разрешается только в том случае, если ни в одной

зависимой таблице не имеется ни одной строки, значение внешнего ключа которой совпадает со значением изменяемого первичного ключа. В общем случае, чтобы изменить значение первичного ключа, пользователь должен сам позаботиться о предварительном изменении значений связанных внешних ключей.

– **Каскадирование (Cascading)**. Это самый удобный и гибкий режим, обеспечивающий автоматическое соблюдение целостности данных. При изменении значения первичного ключа в главной таблице СУБД будет автоматически изменять значения всех связанных внешних ключей во всех строках зависимых таблиц.

Рассмотрим также возможность удаления строк главной таблицы. Если в зависимых таблицах с первичным ключом удаляемой строки не была связана ни одна строка, то проблем нет. Если же такие строки в зависимых таблицах существуют, то необходимо выполнить удаление таким образом, чтобы обеспечить целостность данных. Возможны следующие варианты поведения СУБД при удалении строк из главной таблицы:

– **Установление (Relation)**. При удалении первичного ключа для всех связанных внешних ключей будет автоматически устанавливаться неопределенное значение. Впоследствии такие строки могут быть удалены вручную или связаны с другим первичным ключом.

– **Ограничение (Restrict)**. Перед тем, как станет возможным удаление строки в главной таблице, ни в одной зависимой таблице не должно быть строки, имеющей то же значение внешнего ключа, что и первичный ключ удаляемой строки. Пользователь обязан либо удалить такие строки из зависимой таблицы, либо установить для них значение внешнего ключа в неопределенное значение, либо связать его с любым другим первичным ключом главной таблицы.

– **Каскадирование (Cascading)**. В этом режиме система станет автоматически удалять все связанные строки из зависимых таблиц.

Необходимо заметить, что режим Cascading следует использовать осторожно, т.е. случайное удаление строки главной таблицы может повлечь за собой удаление цепочки строк в зависимых таблицах. Более безопасным методом удаления является использование режима Restrict.

1.5.2 Типы связей между таблицами

При связывании строк главной и зависимой таблицы возможны самые разные сочетания. Тип связи определяет количественные правила сопоставления строк главной и зависимой таблицы. Существуют следующие виды связей.

Один-к-одному (One-to-one). При установке отношения "один-к-одному" (1:1) каждой строке главной таблицы соответствует единственная (или ни одной) строка зависимой таблицы. С другой стороны, каждая строка зависимой таблицы должна быть связана только с одной строкой главной таблицы. Этот тип связи используют не очень часто, поскольку такие данные могут быть помещены в одну таблицу. Связь с отношением "один-к-одному" используют для разделения очень широких таблиц, для отделения части таблицы по соображениям защиты, а также для сохранения сведений, относящихся к подмножеству записей в главной таблице. Например, такой тип связей между таблицами подходит для сохранении сведений об участии студентов в спортивных мероприятиях (рисунок 2).

Один-ко-многим (One-to-many). При использовании связи "один-ко-многим" (1:n) каждой строке главной таблицы соответствует ноль, одна или более строк зависимой таблицы. С другой стороны, каждая строка зависимой таблицы должна быть связана только с одной строкой главной таблицы (см. рисунки 3, 4). Примером такой связи может являться связь человека с его детьми. Частным случаем связи "один-ко-многим" является связь "один-к-одному".

Студенты

КодСтудента	Фамилия	Имя
1	Белов	Иван
2	Новиков	Павел
3	Бабкина	Ольга
4	Воронова	Дарья
5	Кротов	Андрей

Каждый футболист имеет по одной соответствующей записи в таблице “Студенты”

Футболисты

КодСтудента	ТипИгрока	Уровень
1	Защитник	2
2	Голкипер	1
5	Нападающий	2

Этот набор значений является поднабором поля “КодСтудента” таблицы “Студенты”

Рисунок 2 – Связь с отношением "один-к-одному"

Какой-либо национальности...

Национальности

КодНациональности	Национальность
1	Русский
2	Таджик
3	Татарин

...может принадлежать несколько студентов.

Студенты

КодСтудента	Фамилия	Имя	КодНациональности
1	Белов	Иван	1
2	Новиков	Павел	1
3	Бабкина	Ольга	1
4	Воронова	Дарья	3
5	Кротов	Андрей	2

Но каждый студент имеет лишь одну национальность.

Рисунок 3 – Связь с отношением "один-ко-многим"

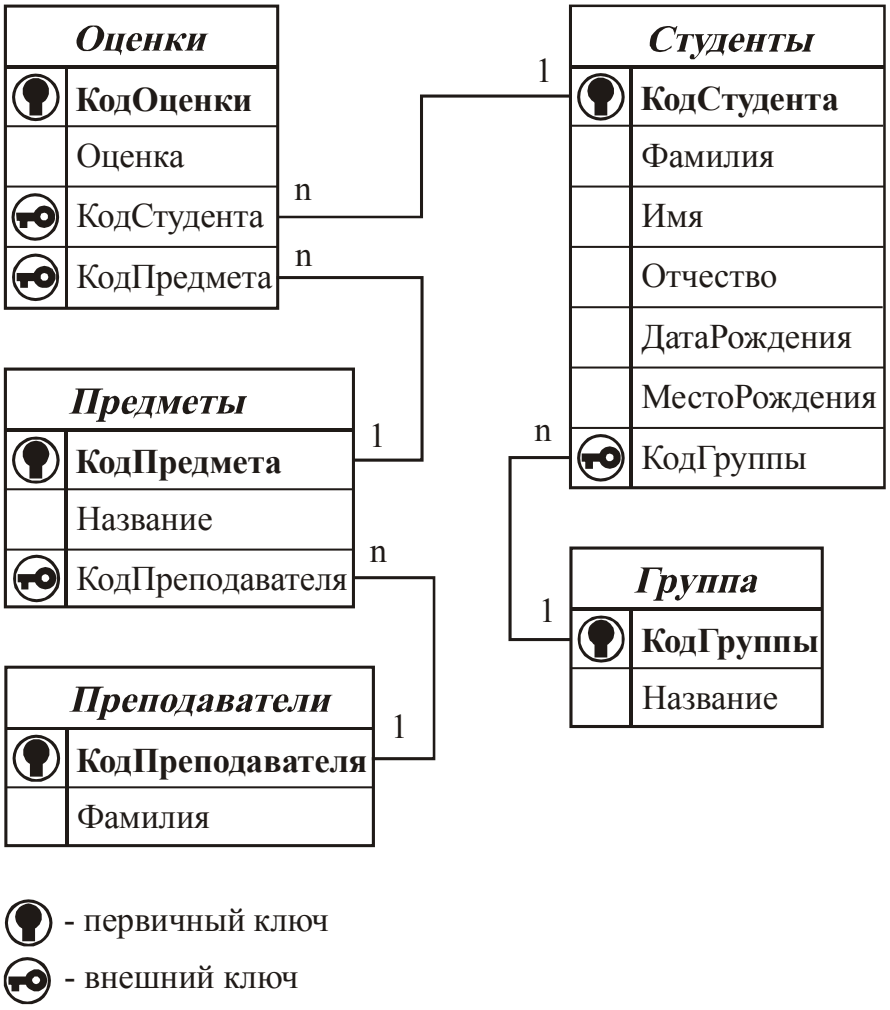


Рисунок 4 – Связи между таблицами для отношения "ОценкиСтудентов"

Многие-ко-многим (Many-to-Many). Этот тип связи предполагает, что любой строке главной таблицы может соответствовать ноль, одна или множество строк зависимой таблицы. При этом каждая строка зависимой таблицы может быть связана с одной или более строк главной таблицы. Примером такой связи может являться использование файлов сотрудниками.

При отношении "многие-ко-многим" одной записи в таблице A могут соответствовать несколько записей в таблице B, а одной записи в таблице B несколько записей в таблице A. Такая схема реализуется

только с помощью третьей (связующей) таблицы, ключ которой состоит из по крайней мере двух полей, которые являются полями внешнего ключа в таблицах A и B. Например, между таблицами "Предметы" и "Аудитории" имеется отношение "многие-ко-многим", которое определяется путем создания двух связей с отношением "один-ко-многим" для таблицы "Расписание" (рисунок 5).

Предметы

КодПредмета	Наименование
1	БЖД
2	Экология
3	ГИС

Изучение дисциплины может осуществляться в нескольких аудиториях...

Расписание

КодПредмета	КодАудитории	Время	День	Вид
2	3	8:30	пн.	л/р
2	1	13:30	пн.	лекция
2	3	8:30	вт.	п/р
1	3	12:00	ср.	лекция
3	5	15:00	ср.	л/р

...и каждая аудитория может служить для преподавания многих дисциплин.

Аудитории

КодАудитории	Номер	Принадлежность
1	А-27	каф. “ОТиОС”
2	А-37	каф. “ОТиОС”
3	А-38	каф. “ОТиОС”
4	А-35	общая
5	А-45	каф. “ОТиОС”

Рисунок 5 – Связь с отношением "многие-ко-многим"

1.5.3 Связывание таблиц в СУБД MS ACCESS

Для определения связей между таблицами необходимо выполнить следующие действия:

1. Закройте все открытые таблицы. Создавать или изменять связи между открытыми таблицами нельзя.
2. Переключитесь в окно базы данных.
3. Нажмите кнопку **Схема данных** на панели инструментов.
4. Если в базе данных не определено никаких связей, то на экран автоматически будет выведено окно **Добавление таблицы**. Если необходимо добавить таблицы до определения связей, а диалогового окна **Добавление таблицы** на экране нет, нажмите кнопку **Добавить таблицу** на панели инструментов. Если же таблицы, которые необходимо связать, отображены на экране, перейдите к шагу 6.
5. Дважды щелкните на именах таблиц, для которых требуется определить связи. Затем закройте диалоговое окно **Добавление таблицы**.
6. Для связывания полей выберите поле в одной таблице и перетащите его на соответствующее поле во второй таблице (для связывания сразу нескольких полей переместите их при нажатой клавише CTRL).

В большинстве случаев связывают поле первичного ключа (представленное в списке полей полужирным шрифтом) одной таблицы с соответствующим ему полем (часто имеющим то же имя) внешнего ключа во второй таблице (рисунок 6). Связанные поля не обязательно должны иметь одинаковые имена, но они должны иметь одинаковые типы данных и иметь содержимое одного типа. Кроме того, связываемые поля числового типа должны иметь одинаковые значения свойства **Размер поля** (**FieldSize**). Необходимо заметить, что поле счетчика можно связывать с числовым полем, если в последнем в свойстве **Размер поля** задано значение "Длинное целое".

7. В диалоговом окне **Схема данных**, отображенном на экране, проверьте имена полей, представленные в двух колонках. Если необходимо, установите параметры связи (рисунок 7).
8. Для создания связи нажмите кнопку **Создать**.

9. Для каждой пары таблиц, которые необходимо связать, выполните шаги 5-8.

При закрытии окна схемы данных на экран будет выведено сообщение, нужно ли сохранять макет. Не зависимо от того, будет он сохранен или нет, связи, созданные в базе данных, будут сохранены.

Примечание. Для связывания таблицы самой с собой или для связывания поля таблицы с другим полем той же таблицы следует дважды добавить таблицу. Такая ситуация возникает при определении поля с подстановкой значений из той же таблицы.

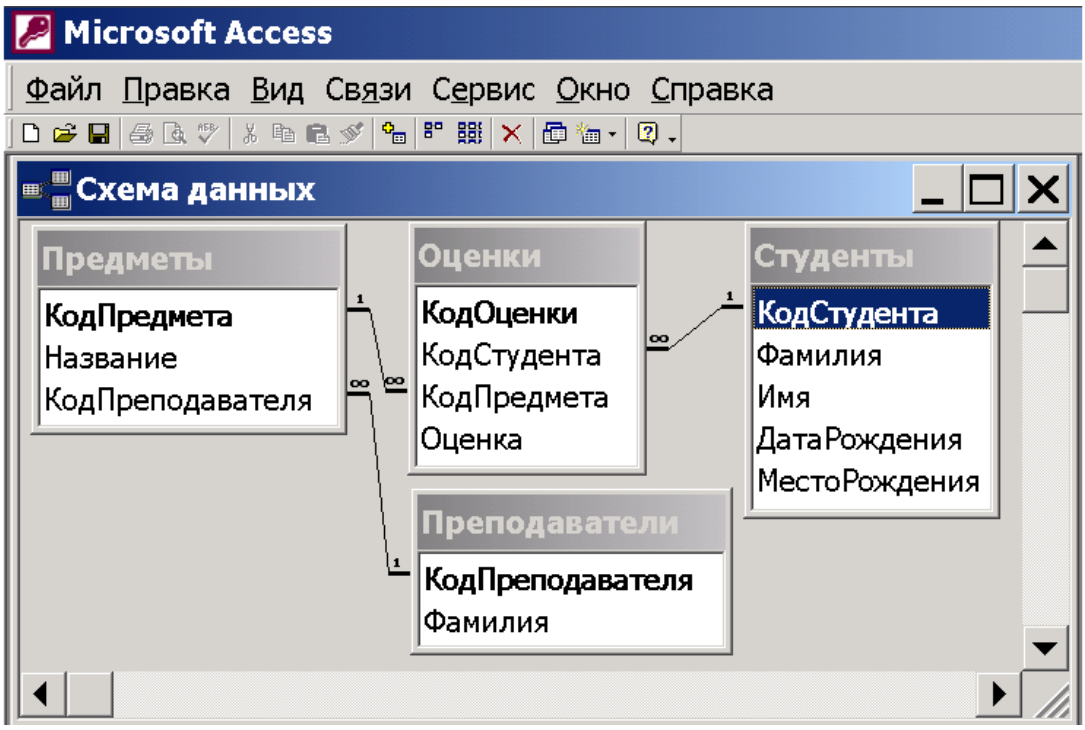


Рисунок 6 – Схема данных

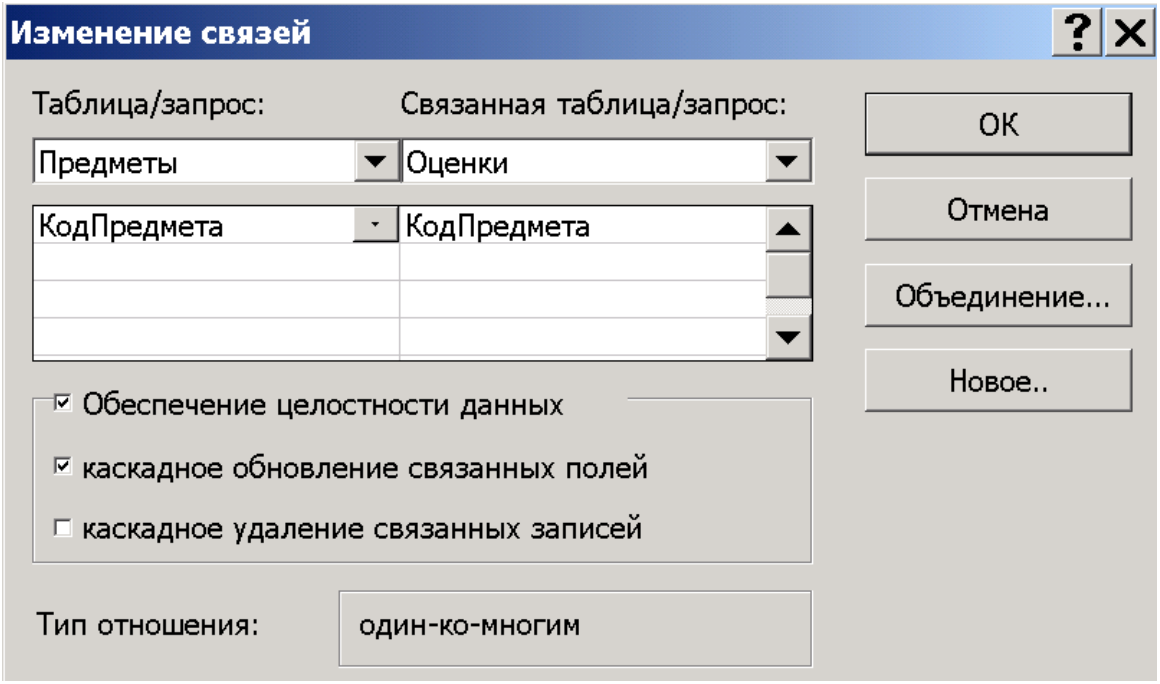


Рисунок 7 – Установка параметров связи

2 Порядок выполнения работы

Согласно выданному преподавателем варианту:

1. Определить предварительную структуру базы данных, провести ее нормализацию до третьей нормальной формы и составить макеты таблиц.
2. Определить правила связывания данных в разных таблицах, для чего выделить поля, которые будут выступать в качестве первичных и внешних ключей. Задать типы связей между таблицами.

Проектирование структуры базы данных осуществляется в соответствие с приведенной ниже информацией.

Вариант №1 – база данных по учету товаров в продуктовом магазине. Каждый товар имеет название и относится к определенному типу (кондитерские, мясные, рыбные и т.п. изделия). Товары одного названия могут присутствовать на складе по разной цене и в любом количестве. Товары поставляются несколькими поставщиками из разных городов. На склад товары поступают группой, при этом

учитывается день поставки. Товары, входящие в группу, могут отличаться по типу, наименованию, цене и количеству.

Вариант №2 – база данных по учету результатов игр в КВН. В базе должна учитываться информация о типе игры (финал, полуфинал и т.д.), названии конкурсов, а также храниться фамилии судей, названия команд и город, откуда они приехали. При проведении игры в базу заносится день, в который она прошла, имена участвовавших команд, названия конкурсов, в которых они участвовали, полученные за конкурс оценки и фамилии поставивших их судей.

Вариант №3 – база данных по учету вредных выбросов промышленных предприятий. В базе хранятся названия предприятий, названия городских округов, список вредных веществ и величины соответствующих ПДК, наименования систем человека (нервная, сердечно-сосудистая и др.). Учесть, что на какую-либо систему отрицательное воздействие могут оказывать несколько вредных веществ, и, в то же время, одно вещество может влиять на несколько систем человека. Данные о выбросах для каждого предприятия заносятся в базу ежемесячно, при этом указывается наименование вещества и величина соответствующего выброса.

Вариант №4 – база данных по учету книг в библиотеке. В базе хранится информация по тематике книг (математика, физика, история, поэзия и т.д.), виду издания (монография, учебник, справочник), о фамилиях авторов. Принять, что автор у книги единственный, однако один автор может написать несколько различных книг. Каждому названию книги соответствует некоторый автор, тематика и вид издания, год выпуска, общее количество экземпляров, количество экземпляров, находящихся в фонде (остальные "на руках"), число читавших данную книгу.

Вариант №5 – база данных по учету больных в поликлинике. В базе хранится информация о поле посетителей ("мужской", "женский"), обслуживаемых поликлиникой улицах, данные о человеке. Данные о человеке слагаются из фамилии, имени, года рождения, пола, названия улицы и номера дома. Должен присутствовать список специалистов (окулист, хирург, терапевт и т.п.). При учете фиксируются данные о

больном, месяц и год посещения, к какому специалисту было обращение.

Содержание отчета:

1. Наименование работы.
2. Цель работы.
3. Структура разработанной базы данных, представленная посредством схемы базы данных и макетов таблиц.

Контрольные вопросы:

1. Дайте определение понятий "сущность", "атрибут".
2. Перечислите требования, которые должны быть соблюдены при проектировании базы данных.
3. Из каких соображений выбирается ключевое поле таблицы?
4. С какой целью проводится процесс нормализации?
5. В чем заключается сущность нормальных форм?
6. Какими механизмами поддерживается целостность данных?
7. Какие существуют типы связей между таблицами?

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра охраны труда и окружающей среды

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«14» 02 2018 г.

РЕАЛИЗАЦИЯ СТРУКТУР БАЗ ДАННЫХ В СИСТЕМАХ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Методические указания к проведению
лабораторных и практических работ
для студентов направлений подготовки 20.03.01, 20.04.01
«Техносферная безопасность»

Курск 2018

УДК 699.85
Составители: И.О. Кирильчук, А.В. Гнездилова

Рецензент
Кандидат технических наук, доцент *Г.П. Тимофеев.*

Реализация структур баз данных в системах управления базами данных: методические указания к проведению лабораторных и практических работ / Юго-Зап. гос. ун-т; сост.: И.О. Кирильчук, А.В. Гнездилова. Курск, 2018. 19 с.

Излагается информация по созданию баз данных в геоинформационной системе MapInfo и системе управления базами данных Microsoft Access.

Методические указания предназначены для студентов направлений подготовки 20.03.01, 20.04.01 Техносферная безопасность.

Текст печатается в авторской редакции

Подписано в печать 14.02.18. Формат 60х84 1/16.
Усл. печ. л. 1,1. Уч.-изд.л. 1,0. Тираж 30 экз. Заказ 1068. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Цель работы:

– получение навыков реализации структур баз данных в системе управления базами данных Microsoft Access и геоинформационной системе MapInfo.

Общие положения

1. Создание нового файла базы данных

1.1. Microsoft Access

Для создания нового файла базы данных необходимо запустить программу Microsoft Access и в разделе **Создание базы данных** (Create a new database using) диалогового окна Microsoft Access выбрать переключатель **Новая база данных** (Blank Access database).

В диалоговом окне **Файл новой базы данных** (File new database) укажите папку, в которой будет храниться создаваемая база данных, имя файла и нажмите кнопку **Создать** (Create). После этого в главном окне Microsoft Access появится окно новой базы данных.

1.2. MapInfo

В отличие от Microsoft Access, в котором все объекты (таблицы, запросы, индексы и т.д.) хранятся в единственном файле сложной структуры, в ГИС MapInfo хранение данных организовано по-другому.

В MapInfo картографическая информация выводится посредством наложения друг на друга отдельных слоев, каждый из которых описывает положение в пространстве и характеристики отдельного класса географических объектов. Например, итоговая карта города может слагаться из слоев, описывающих положение улиц, водных объектов, лесных массивов, железнодорожных путей, промышленных предприятий, школ и т.д. Каждому слою *логически* соответствует таблица с определенным именем, которая *физически* хранится в виде набора файлов с одинаковыми именами (совпадающими с именем логической таблицы), но разными расширениями. Например, слою, описывающему положение улиц города, может соответствовать таблица "улицы", хранящаяся на жестком диске компьютера

посредством файлов "улицы.tab", "улицы.id", "улицы.ind", "улицы.dat" и "улицы.map"). Главным из этих файлов является файл с расширением ".tab", который хранит настройки слоя и связывает в единую структуру все остальные файлы. Необходимо заметить, что как раз список файлов с расширением ".tab" появляется при открытии таблицы через меню **Файл/Открыть таблицу**.

Слои, определяющие вид итоговой карты, образуют *Рабочий Набор*. В *Рабочем Наборе* хранятся настройки, характеризующие множество слоев в целом. Таким образом, приближенно *Рабочий Набор* представляет собой аналог файла базы данных Microsoft Access. Создать прямо *Рабочий Набор* нельзя, необходимо сначала открыть (или создать новые) таблицы слоев карты, а затем через меню **Файл/Сохранить Рабочий Набор** записать настройки карты в файл с расширением ".wor".

Для удобства работы с отдельными картами, принято помещать *Рабочий Набор* карты и все файлы, относящиеся к ней, в отдельную папку.

2. Создание таблиц

2.1. Microsoft Access

После создания нового файла базы данных выберите группу **Таблицы** (Tables). В появившемся списке присутствуют три ярлыка, соответствующие трем разным способам создания таблиц: путем ввода данных, с помощью *Конструктора таблиц* и с помощью *Мастера создания таблиц*. Рассмотрим второй способ.

Для открытия окна *Конструктора таблиц* необходимо дважды щелкнуть левой кнопкой мыши на ярлыке **Создание таблицы с помощью конструктора** (Create table in Design View).

В режиме *Конструктора таблицы* создаются путем задания имен полей, их типов и свойств посредством следующих действий (рисунок 1):

1. В столбец **Имя поля** (Field Name) вводятся имена полей создаваемой таблицы.

2. В столбце **Тип данных** (Data Type) для каждого поля выбирается из раскрывающегося списка тип данных, которые будут содержаться в поле.
3. В столбце **Описание** (Description) можно ввести описание данного поля (необязательно).
4. В нижней части *Конструктора таблиц* на вкладках **Общие** (General) и **Подстановка** (Lookup) вводятся свойства каждого поля или оставляются значения свойств по умолчанию.
5. Указываются ключевые поля, для чего выделяется соответствующая строка таблицы и нажимается кнопка **Ключевое поле** (Primary Key) на панели инструментов **Конструктор таблиц** (Table Design). Панель инструментов обычно расположена под основным меню.
6. После описания всех полей таблицы нажимается кнопка **Закрыть** (Close) и в диалоговом окне **Сохранить как** (Save As) вводится имя создаваемой таблицы.

Для описания таблиц можно использовать как символы английского, так и русского алфавитов. При этом следует придерживаться правил именования пользовательских объектов Access:

- Имена полей в таблице не должны повторяться, т.е. должны быть уникальными.
- Имена могут содержать не более 64 символов, включая пробелы.
- Желательно избегать употребления имен, совпадающих с именами встроенных функций или свойств Microsoft Access (например, Name).
- Имя поля не должно начинаться с пробела или управляющего символа (коды ASCII 00-31).
- Имена полей могут содержать любые символы, включая буквы, цифры, пробелы, специальные символы, за исключением точки (.), восклицательного знака (!), надстрочного символа (`) и прямых скобок ([ ]).

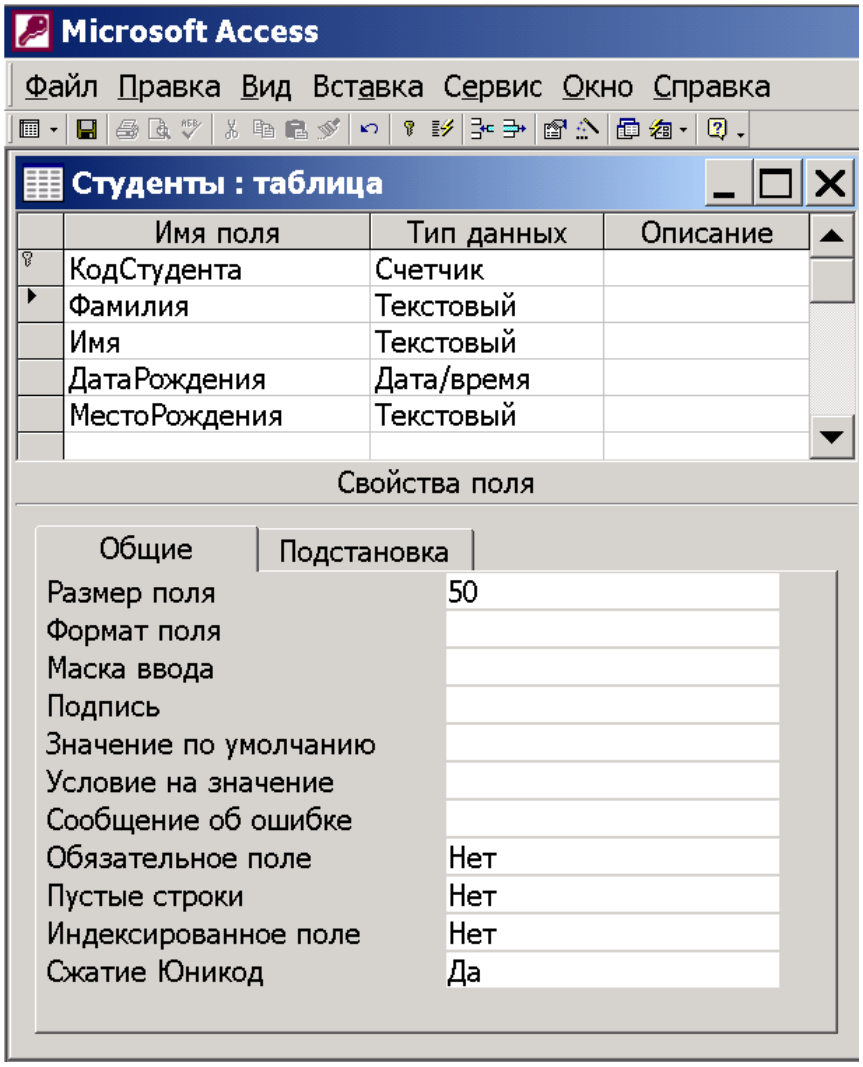


Рисунок 1 – Окно конструктора таблицы

Хотя при именовании полей допустимо использование пробелов и отличных от букв и цифр символов, рекомендуется этого избегать. Это обусловлено тем, что если хранящиеся в базе данные будут в дальнейшем обрабатываться в СУБД, отличной от Microsoft Access (например, MapInfo), то велика вероятность возникновения ошибок при подключении базы данных. Поэтому на практике часто используется следующий стиль написания имен полей: все слова пишутся слитно, начиная каждое с заглавной буквы (например, КодСтудента, ГодРождения, СемейноеПоложение).

2.2. MapInfo

Для создания новой таблицы необходимо щелкнуть левой кнопкой мыши на пункте меню **Файл**/**Новая таблица**, что вызовет появление на экране диалога **Новая таблица** (рисунок 2). Это диалог позволяет выбрать, в каком окне будет показана новая таблица после ее создания: в окне *Списка*, в новом окне *Карты* или в виде нового слоя в уже открытом окне *Карты*:

- **Показать списком**. Если флажок выбран, то MapInfo откроет пустое окно *Списка*, когда формирование новой таблицы и задание для нее имени будет закончено.
- **Показать Картой**. Если флажок выбран, то MapInfo откроет окно Карты, когда формирование новой таблицы и задание для нее имени будет закончено (необходимо задать проекцию таблицы в диалоге "Создать структуру таблицы").
- **Добавить к Карте**. Если флажок выбран, то MapInfo добавляет слой в активное окно *Карты*. Таблица должна иметь проекцию карты.

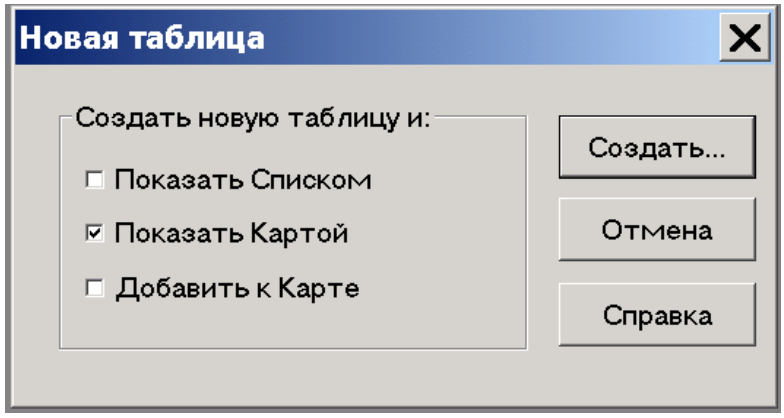


Рисунок 2 – Создание новой таблицы

После отметки нужных опций нажимается кнопка **Создать**, вызывающая появление диалога **Создать структуру таблицы** (рисунок 3).

В этом диалоге создается структура таблицы, позволяющая иметь графические объекты. Такая структура позволяет отображать таблицу в виде карты или плана. Каждому объекту в таблице соответствует запись с табличными (неграфическими данными). Данные таблицы,

имеющей структуру без графики, могут быть показаны только в окне *Списка*.

Замечание: Для использования команды **Запрос/Найти** таблица должна иметь индексированные поля и возможность иметь графические объекты.

Вверху диалога расположено окошко со списком, верхняя строчка которого соответствует первому полю таблицы, вторая строчка - второму полю и так далее. Для изменения поля, необходимо выбрать соответствующую строчку в списке.

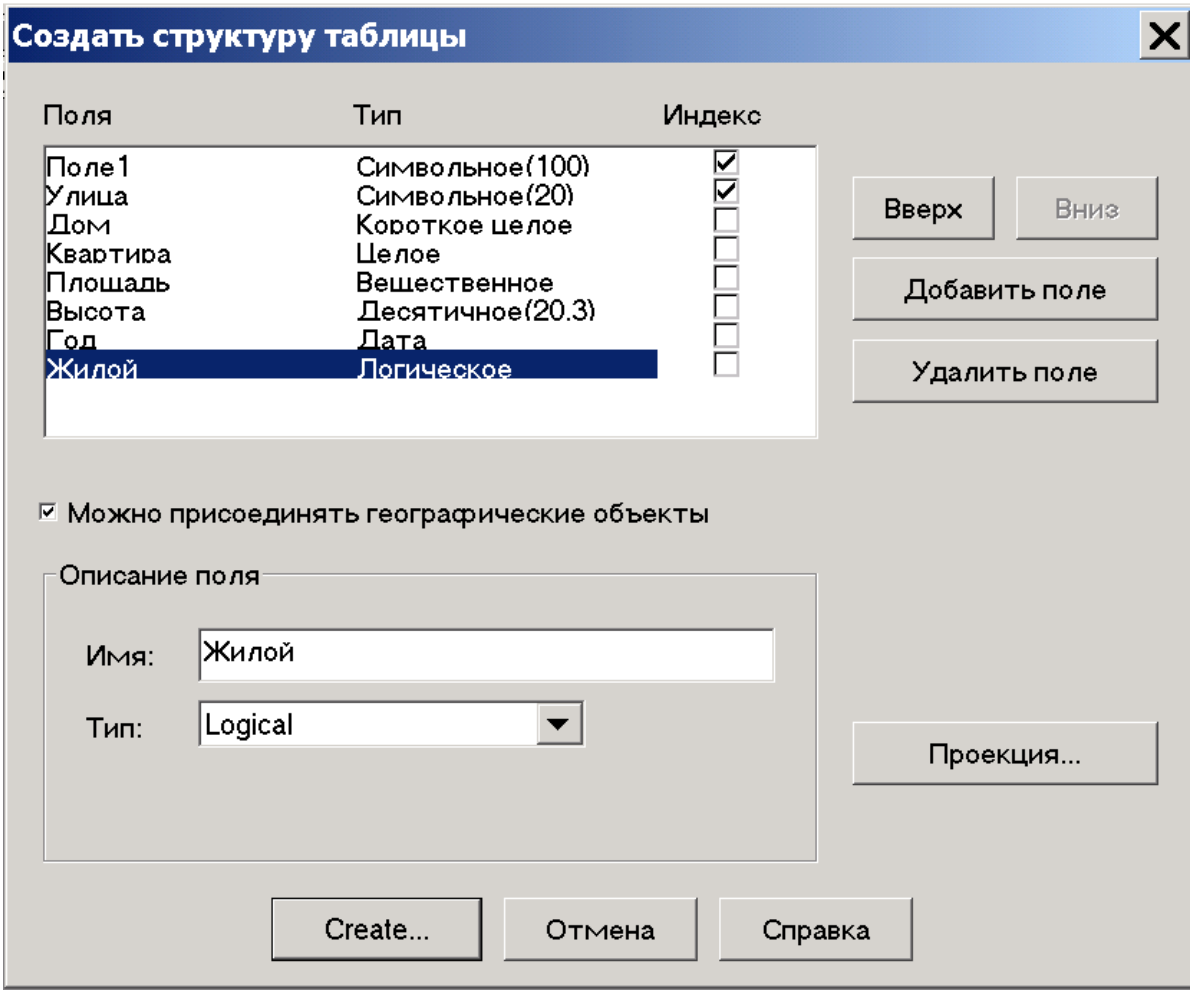


Рисунок 3 – Определение структуры таблицы

Таблица должна иметь хотя бы одно поле. Если при создании таблицы заранее неизвестно, какие поля она должна иметь, то создается одно поле, а следующие добавляются уже потом.

Диалог **Создать структуру таблицы** состоит из следующих элементов:

- **Поля**. Слово расположено над окошком со списком полей и является заголовком колонки с именами полей (колонок) таблицы.
- **Тип**. Слово расположено над окошком со списком полей и является заголовком колонки с типами, назначенными полям таблицы. Некоторые типы включают в скобках длину поля (если поле фиксированной длины).
- **Индекс**. Слово расположено над окошком со списком полей и является заголовком колонки, в которой в строке индексированного поля ставится галочка. Если поле не индексировано, то в этой колонке будет пустое место. Индексировать можно любое количество столько полей. Индексирование не изменяет порядок записей в таблице. Индексирование необходимо для:
  - выполнения команды **Запрос**/**Найти** (поиск осуществляется только по индексированным колонкам);
  - ускорения выполнения запросов с использованием Числового или алфавитного сравнения;
  - ускорение процесса объединения.
- **Вверх**, **Вниз**. Изменяют порядок полей в таблице. Выберите поле в списке и нажмите на кнопку **Вверх**, если хотите уменьшить его порядковый номер, или на кнопку **Вниз**, если хотите увеличить. Соответственно меняется структура таблицы. От порядка расположения полей таблицы напрямую зависит расположение колонок в открываемом окне *Списка*. Первое поле соответствует самой левой колонке *Списка*, второе поле - второй колонке от левого края и т. д.
- **Добавить поле**. Добавляет в конец списка новое поле. Новому полю присваивается стандартное имя Поле1, Поле2, Поле3 и т.

д. Номер зависит от порядкового номера, который присваивается новому полю при создании.

- **Удалить поле**. Удаляет поле из таблицы.
- **Сопоставить таблице графические объекты**. Если флажок установить, то структура таблицы разрешает создание в таблице графических объектов.
- **Проекция**. Вызывает диалог **Выбор проекции** для изменения проекции таблицы.
- **Описание поля**. В этой части диалога производится настройка поля, выбранного в верхнем списке. Изменение в описанных ниже окошках отражается в колонках списка полей.
- **Имя**. Введите имя поля. При создании поля ему присваивается стандартное имя Поле1, Поле2, Поле3 и т. д. Вы можете изменить имя поля на любое не более 31 символа длиной. Для имени могут использоваться буквы, цифры и символ подчеркивания. Пробелы не используются, в место них рекомендуем между словами использовать символ подчеркивания (_). В именах можно использовать заглавные и строчные буквы, но следует помнить, что MapInfo их не различает.
- **Тип**. Список содержит все типы данных, которые может содержать поле таблицы. Выберите один их них.
- **Знаков**. Для символьного или десятичного типа. Максимальное значение для символьного типа – 250 знаков, а для десятичного – 19.
- **Знаков после запятой**. Задайте число знаков для десятичного типа.
- **Индекс**. Установите флажок для создания индекса для выбранного поля.
- После нажатия кнопки **Создать** (Create) создается таблица с заданной структурой. Для задания имени файлов таблицы, а также диска и каталога будет открыт диалог **Создать новую таблицу** (рисунок 4). Новая таблица показывается или не

показывается в одном из окон MapInfo в соответствии с установкой в диалоге **Новая таблица**.

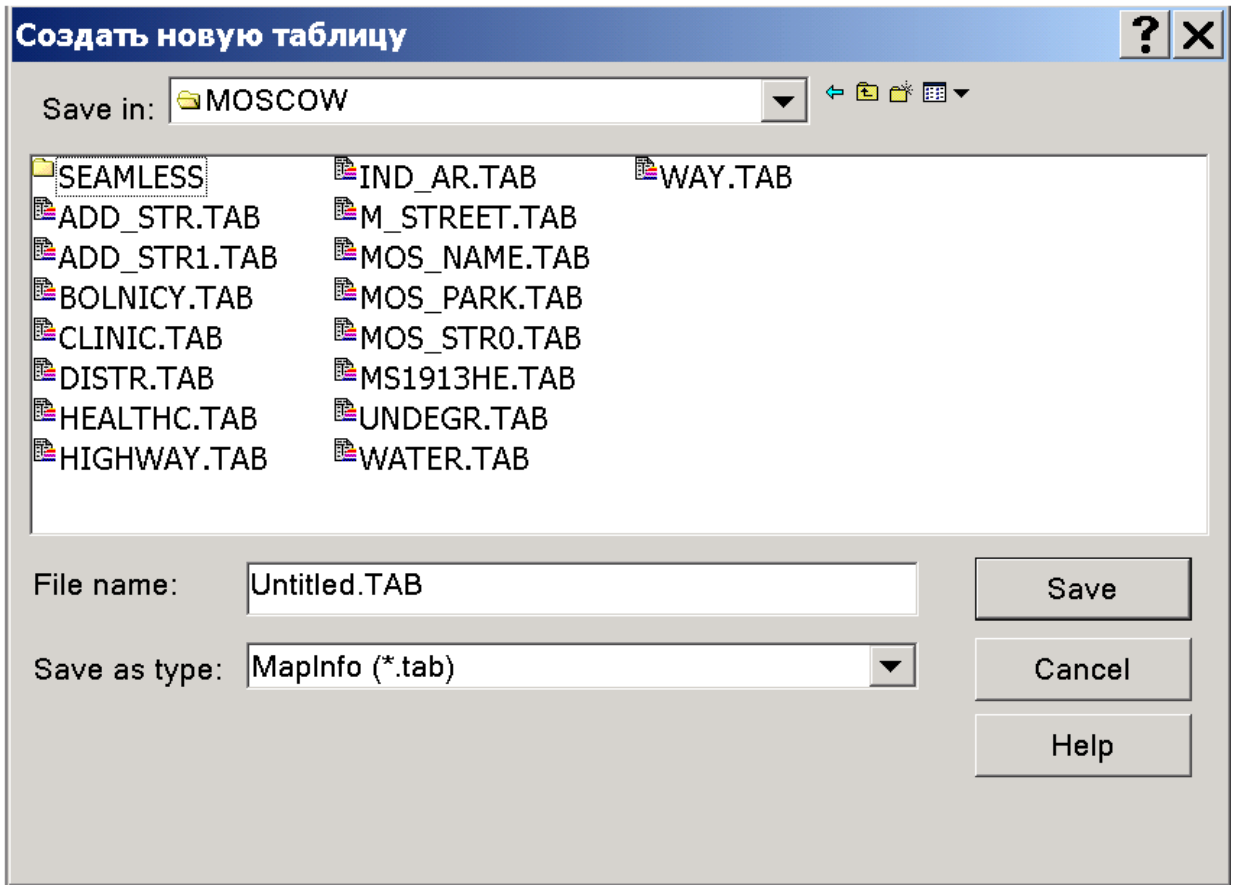


Рисунок 4 – Диалог сохранения таблицы

3. Задание типа данных

3.1. Microsoft Access

Тип данных определяет вид информации, которая будет храниться в поле, и выбирается в столбце **Тип данных** (Data Type) *Конструктора таблиц* (рисунок 5). Описание типов данных приведено в таблице 1.

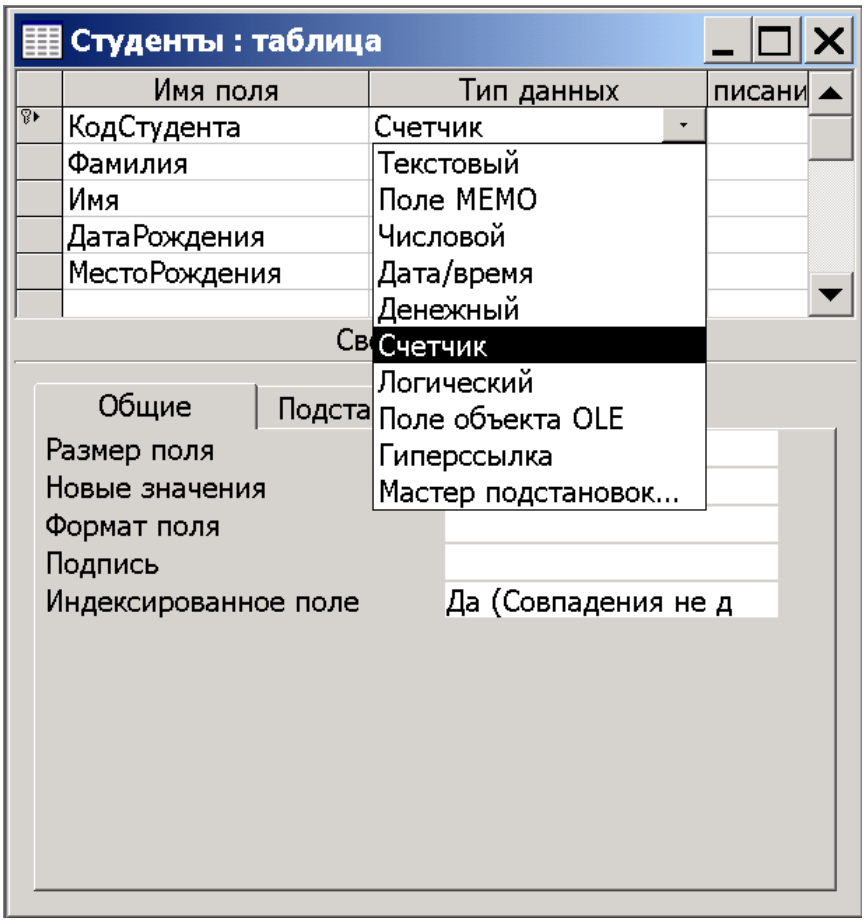


Рисунок 5 – Выбор типа данных поля таблицы

После определения типа данных, которые будут храниться в поле таблицы, в нижней части *Конструктора таблиц* на вкладке **Общие** (General) вводятся свойства каждого поля или оставляются значения свойств по умолчанию. Набор свойств поля зависит от выбранного типа данных. В таблице 2 дано краткое описание свойств полей.

Таблица 1

Типы данных Microsoft Access

Тип данных	**Описание**
Текстовый (Text)	Символьные или числовые данные, не требующие вычислений. Поле данного типа может содержать до 255 (2^8-1) символов. Размер текстового поля задается с

Тип данных	Описание
	помощью свойства **Размер Поля** (FieldSize). В нем указывается максимальное количество символов, которые могут быть введены в данное поле.
Поле MEMO (Memo)	Поле MEMO предназначено для ввода текстовой информации, по объему превышающей 255 символов. Это поле может содержать до 65535 (2^{16}-1) символов. Этот тип данных отличается от типа **Текстовый** (Text) тем, что в таблице хранятся не сами данные, а ссылки на блоки данных, которые хранятся отдельно. За счет этого ускоряется обработка таблиц (сортировка, поиск и т.п.). Поле типа MEMO не может быть ключевым или проиндексированным.
Числовой (Number)	Числовой тип используется для хранения числовых данных, используемых в математических расчетах. Имеет много подтипов. От выбора подтипа (размера) данных числового типа зависит точность вычислений. Для установки подтипа числовых данных служит свойство **Размер Поля** (FieldSize). Обычно по умолчанию используется подтип **Длинное целое** (Long Integer), которое занимает 4 байта и представляет собой число в пределах от -2147483648 до 2147483647 ($-2^{31}..2^{31}$-1).
Дата/Время (Data/Time)	Тип для представления даты и времени. Позволяет вводить даты с 100 по 9999 год. Размер поля 8 байт. Даты и время хранятся в специальном фиксированном числовом формате. Дата является целой частью значения поля типа **Дата/Время**, а время его дробной частью.
Денежный (Currency)	Тип данных, предназначенный для хранения данных, точность представления которых колеблется от 1 до 4 знаков после десятичной точки. Целая часть данного типа может содержать до 15 десятичных знаков.
Счетчик	Поле содержит 4-байтный уникальный номер,

Тип данных	Описание
(AutoNumber)	определяемый Microsoft Access для каждой новой записи автоматически путем увеличения предыдущего значения на 1 или случайным образом. Значения полей типа счетчика обновлять (вносить изменения) нельзя. Максимальное число записей в таблице с полем счетчика не должно превышать 2 миллиарда (2^{31}).
Логический (Yes/No)	Логическое поле, которое может содержать только два значения, интерпретируемых как **Да**/**Нет**, **Истина**/**Ложь**, **Включено**/**Выключено**. Access использует величину –1 для представления значения **Да** и величину 0 для **Нет**. Поля логического типа не могут быть ключевыми, но их можно индексировать.
Поле объекта OLE (OLE object)	Содержит ссылку на OLE-объект (лист Microsoft Excel, документы Microsoft Word, MathCAD, звук, изображение и т.п.). Объем объекта ограничивается имеющимся в наличии дисковым пространством.

Таблица 2

Описание свойств полей таблицы

Свойство	Описание
Размер поля (FieldSize)	Определяет максимальную длину текстового или числового поля (учтите, что при создании полей слишком большого размера неэкономно расходуется память и замедляется работа СУБД, а полей маленького размера – искажается содержимое поля)
Формат поля (Format)	Устанавливает формат отображения данных в форме и запросе
Число десятичных знаков (DecimalPlaces)	Определяет количество разрядов в дробной части десятичного числа
Маска ввода	Определяет маску данных при вводе

Свойство	Описание
(InputMask)	
Подпись (Caption)	Содержит надпись, которая выводится рядом с полем в форме или отчете
Значение по умолчанию (DefaultValue)	Содержит значение, устанавливаемое в соответствующем поле таблицы по умолчанию при создании новой записи (например, для таблицы на рис.5 значение по умолчанию для поля *МестоРождения* можно задать как "Курск")
Условие на значение (ValidationRule)	Определяет множество значений, которые пользователь может вводить в это поле при заполнении таблицы (для поля *ДатаРождения* на рис.5 вполне можно задать условие: >1.1.1970)
Сообщение об ошибке (ValidationText)	Определяет сообщение, которое появляется на экране при вводе недопустимого значения
Обязательное поле (Required)	Данная опция определяет, обязательно ли нужно при добавлении новой записи сразу же заносить в поле какое-либо значение, или можно ли это сделать позже
Пустые строки (AllowZeroLength)	Данная опция определяет, допускается ли ввод в данное поле пустых строк ("")
Индексированное поле (Indexed)	Определяет простые индексы для ускорения поиска записей (поле первичного ключа индексируется автоматически)
Сжатие Юникод (Unicode Compression)	Указывает, используется ли сжатие символов в кодировке Unicode

Размер поля устанавливается с помощью параметра **Размер поля** и зависит от значений, которые предполагается вводить в данное поле. В таблице 3 и на рисунке 6 приведены значения, которые может иметь параметр **Размер поля** для числовых полей.

Таблица 3

Характеристики подтипов для типа данных **Числовой**

Размер поля	Описание
Байт (Byte)	Хранятся целые числа из диапазона от 0 до 255 (2^8-1). Поле занимает 1 байт.
Целое (Integer)	Хранятся целые числа из диапазона от –32768 до 32767 (-2^{15}...2^{15}-1). Поле занимает 2 байта.
Длинное целое (Long Integer)	Хранятся целые числа из диапазона от -2147483648 до 2147483647 (-2^{31}..2^{31}-1). Поле занимает 4 байта. Значение **Длинное целое** устанавливается по умолчанию.
Одинарное с плавающей точкой (Single)	Хранятся дробные числа с шестью знаками после запятой из диапазона от –3.402823 ·10^{38} до 3.402823 ·10^{38} . Поле занимает 4 байта.
Двойное с плавающей точкой (Double)	Хранятся дробные числа с пятнадцатью знаками после запятой из диапазона от –1.79769313486232·10^{308} до 1.79769313486232·10^{308}. Поле занимает 8 байт.
Действительное (Decimal)	Хранятся любые числа от –10^{28}–1 до 10^{28}–1, записанные максимум 28 цифрами. Поле занимает 12 байт.

Для хранения информации полей типа **Текстовый**, **МЕМО** в Access используется стандарт кодировки текста Unicode. В Unicode каждый символ представляется двумя байтами, что дает возможность поддерживать до 65536 символов (при однобайтовой кодировке обеспечивается доступ только к одной кодовой странице, состоящей из 256 символов). Следовательно, поля с кодировкой Unicode требуют больше места для хранения данных. Установленное в значение **Да** свойство поля **Сжатие Юникод** приводит к тому, что все символы, первый байт которых равен 0, будут сжиматься при сохранении и восстанавливаться при выборке.

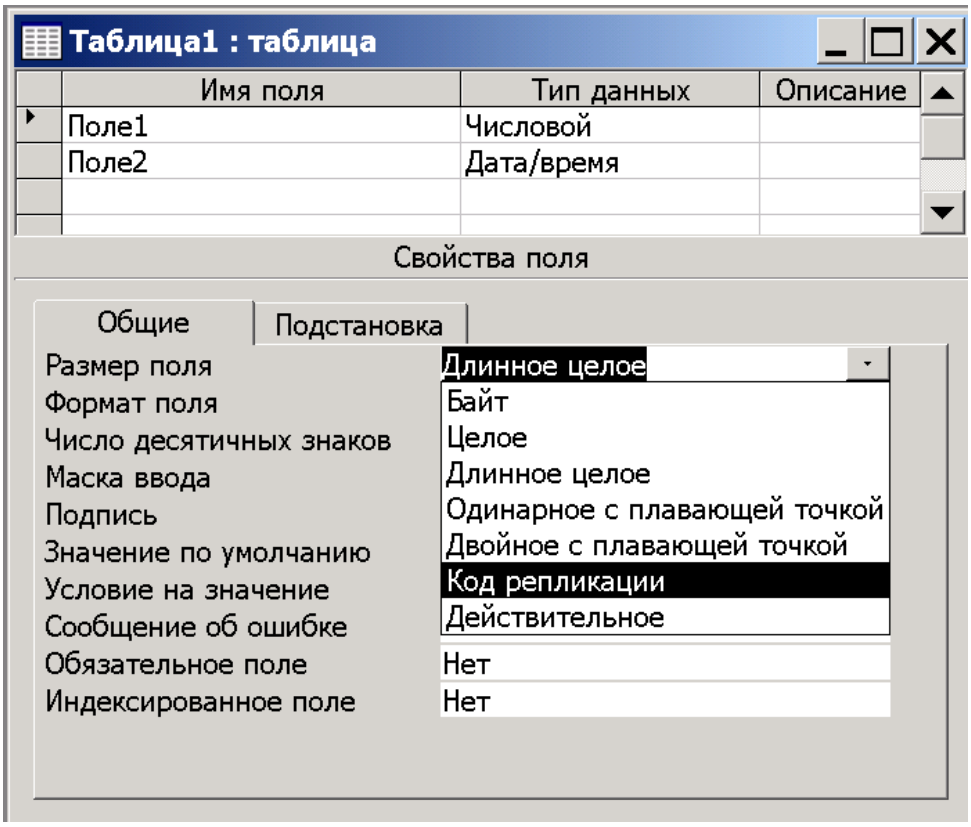


Рисунок 6 – Выбор размера поля таблицы

3.2. MapInfo

В MapInfo поле таблицы может содержать данные следующих типов:

- **Символьное** (Character). Строки до 254 символов длиной. Данный тип аналогичен типу **Текстовый** Microsoft Access.
- **Целое** (Integer). Целое число в пределах от –2147483648 до 2147483647 ($-2^{31}..2^{31}-1$). Данный тип аналогичен подтипу **Длинное целое** (Long Integer) Microsoft Access.
- **Короткое целое** (Small Integer). Целое число от -32768 до 32767 ($-2^{15}..2^{15}-1$). Данному типу соответствует подтип **Целое** (Integer) Microsoft Access.
- **Вещественное** (Float). Десятичные числа с плавающей точкой, соответствуют подтипу **Двойное с плавающей точкой** Microsoft Access.

- **Десятичное** (Decimal). Десятичные числа с фиксированной точкой. Данный тип аналогичен подтипу **Действительное** (Decimal) Microsoft Access.
- **Дата** (Date). Календарная дата. Данному типу соответствует тип **Дата/Время** Microsoft Access.
- **Логическое** (Logical). Значения, интерпретируемые как **Истина**/**Ложь**. В поле такого типа появляется либо символ "T" (TRUE) в случае значения **Истина**, либо литера "F" (FALSE) в случае значения **Ложь**. Данному типу соответствует тип **Логический** Microsoft Access.

Порядок выполнения работы:

Согласно выданному преподавателем варианту реализовать в Microsoft Access структуру базы данных, разработанную в ходе проведения лабораторной работы "Проектирование структур баз данных информационных систем". Для достижения этой цели необходимо:

1. В режиме *Конструктора таблиц* Microsoft Access создать набор таблиц.
2. В режиме *Схемы данных* осуществить связывание данных в разных таблицах. В качестве ограничений целостности данных использовать каскадное обновление связанных полей.
3. Ввести данные в базу.

Содержание отчета:

1. Наименование работы.
2. Цель работы.
3. Структура разработанной базы данных, представленная в виде схемы базы данных Microsoft Access и описания полей таблиц с указанием их типов.

Контрольные вопросы:

1. В чем состоит отличие баз данных ГИС MapInfo и Microsoft Access?
2. Какова процедура создания таблиц в Microsoft Access?
3. Перечислите общие и отличительные черты процессов создания таблиц в Microsoft Access и MapInfo.
4. Какие типы данных могут храниться в таблицах?

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра охраны труда и окружающей среды

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«14» 02 2018 г.

ЯЗЫК СТРУКТУРИРОВАННЫХ ЗАПРОСОВ К ДАННЫМ ИНФОРМАЦИОННЫХ СИСТЕМ

Методические указания к проведению
лабораторных и практических работ
для студентов направлений подготовки 20.03.01, 20.04.01
«Техносферная безопасность»

Курск 2018

УДК 699.85
Составители: И.О. Кирильчук, А.В. Гнездилова

Рецензент
Кандидат технических наук, доцент *Г.П. Тимофеев.*

Язык структурированных запросов к данным информационных систем: методические указания к проведению лабораторных и практических работ / Юго-Зап. гос. ун-т; сост.: И.О. Кирильчук, А.В. Гнездилова. Курск, 2018. 28 с.

Описан синтаксис языка структурированных запросов (язык SQL), обеспечивающего выборку и анализ данных в системах обработки информации; даются навыки практического применения языка SQL для построения оптимальных запросов к данным информационных систем.

Методические указания предназначены для студентов направлений подготовки 20.03.01, 20.04.01 Техносферная безопасность.

Текст печатается в авторской редакции

Подписано в печать 14.02.18. Формат 60х84 1/16.
Усл. печ. л. 1,62. Уч.-изд.л. 1,47. Тираж 30 экз. Заказ 1063. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Цель работы:

изучение синтаксиса языка структурированных запросов; получение навыков построения оптимальных запросов для выборки данных из реляционных баз данных информационных систем.

1 Общие положения

Целью любой системы управления базами данных является предоставление пользователю простых и эффективных механизмов манипулирования данными. Для этого можно использовать самые различные методы управления данными. Одним из таких методов является язык структурированных запросов (SQL, Structured Query Language). Разработанный в 1970 г. фирмой IBM, язык SQL стал к настоящему моменту общепринятым стандартом доступа к данным.

Для выборки данных в языке SQL существует команда SELECT, которая позволяет делать как простую выборку всех данных из одной таблицы текущей базы данных, так и выполнять сложные запросы одновременно к множеству таблиц различных баз данных. В самом простейшем случае выборка данных производится с помощью команды:

SELECT * FROM *table_name*

Эта команда выводит данные из всех столбцов для всех строк таблицы, т.е. возвращается вся информация, содержащаяся в таблице. Однако в большинстве случаев применяются более сложные конструкции, использующие группировку, подзапросы, условия и другие дополнительные механизмы управления запросом.

Необходимо отметить, что выборка данных может осуществляться как из статических таблиц базы данных, так и из виртуальных таблиц, содержимое которых генерируется динамически на основе результата выполнения запроса. Такие виртуальные таблицы носят название представлений (views) и являются, по сути, поименованными запросами SELECT. Они используются в основном для скрытия от пользователей столбцов с конфиденциальными данными. Для пользователя работа с представлениями мало отличается от работы с таблицами, однако имеют место некоторые ограничения. Так, например, для представления

нельзя определить ограничения целостности. Заметим также, что в СУБД Microsoft Access в отличие от Microsoft SQL Server представления выделены не в отдельную группу, а объединены с обычными запросами.

Команда SELECT разбита на отдельные разделы, каждый из которых выполняет узкоспециализированную функцию и является практически независимым от других разделов. Хотя количество разделов довольно велико, на практике обычно применяется "облегченный" вариант команды SELECT:

SELECT *select_list*
FROM *table_sourse*
[WHERE *search_condition*]
[GROUP BY *group_by_expression*]
[HAVING *search_condition*]
[ORDER BY *order_expression* [ASC | DESC]]

Далее подробно рассматривается структура и примеры использования разделов команды SELECT. Приводимые примеры иллюстрируют работу с базой данных, структура которой изображена на рисунке 1.

1.1 Раздел SELECT

С помощью этого раздела указывается список столбцов, которые будут включены в результат выборки. Кроме того, в этом разделе можно управлять количеством и качеством строк, входящих в результат выборки. Структура раздела SELECT такова:

SELECT [ALL | DISTINCT] [TOP *n* [PERCENT]] <*select_list*>

Рассмотрим использование параметров раздела.

ALL

Это ключевое слово указывает, что в результат выборки должны быть включены все строки, возвращаемые запросом. То есть результат выборки может содержать повторяющиеся строки. Параметр ALL используется по умолчанию и его указание необязательно.

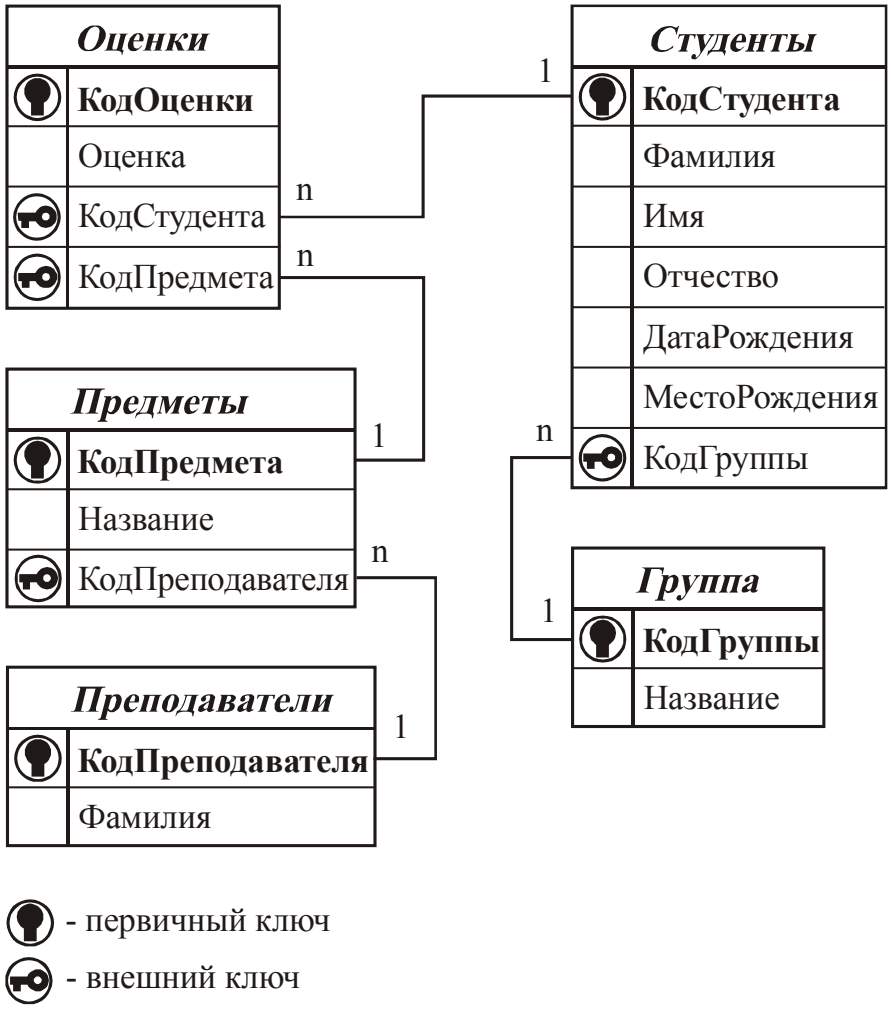


Рисунок 1 – Схема базы данных по учету успеваемости студентов

DISTINCT

Применение этого параметра позволяет исключить из возвращаемого результата повторяющиеся строки. Тем самым можно обеспечить уникальность каждой строки, возвращаемой запросом. Значения `Null` считаются эквивалентными и включаются в выборку. Если DISTINCT не указывается, то будет использоваться параметр ALL.

TOP *n* [PERCENT]

С помощью этой конструкции можно ограничить количество

строк, которые будут включены в результат выборки. После ключевого слова TOP с помощью параметра *n* задается максимальное количество строк, которое может содержать результат. Если указывается ключевое слово PERCENT, то параметр *n* означает количество строк в процентах от общего числа строк, возвращаемым запросом.

<select_list>

С помощью этой конструкции формируется собственно список столбцов, которые будут включены в результат выборки, а также значения для этих столбцов. Структура этой конструкции такова:

<select_list> ::= { *
| { *table_name* | *view_name* | *table_alias* }.*
| { *column_name* | *expression* } [AS *column_alias*] } [,...n]

Рассмотрим подробно назначение каждого из параметров:

– *. Указание этого символа повлечет включение в результат выборки всех столбцов всех таблиц и представлений, участвующих в запросе и указанных в разделе FROM. Однако следует быть внимательным при выборке одноименных столбцов из разных таблиц. Обращение к таким столбцам будет весьма затруднено при дальнейшем использовании результатов выборки. Порядок перечисления столбцов в результате выборки соответствует физическому порядку столбцов в таблице. Кроме того, сначала перечисляются все столбцы первой таблицы, указанной в разделе FROM, затем второй таблицы и т.д., пока не будут выведены столбцы всех таблиц. Если не используется конструкция WHERE, то для каждой строки одной таблицы будет выводиться полный набор комбинаций строк других таблиц. Например, если выборка производится из двух таблиц с количеством строк 23 и 13, то общее количество возвращаемых строк будет 299 (23*13).

– { *table_name* | *view_name* | *table_alias* }.*. Позволяет ограничить количество столбцов включением только *всех* столбцов одной таблицы или представления. Как видно из синтаксиса, сначала указывается имя объекта, из которого будет производиться выборка, потом точка, и в конце символ *. Параметры *table_name* и *view_name* позволяют ссы-

латься на конкретную таблицу или представление, параметр *table_alias* – на данные через псевдоним таблицы. При этом под псевдонимом можно обратиться как к реальной физической таблице, так и к результату выборки, возвращаемому подзапросом (динамической таблице). В любом случае, имя объекта должно быть упомянуто в разделе FROM.

– *column_name*. Подразумевает указание имени столбца, который должен быть включен в результат выборки. Столбец должен принадлежать таблице или представлению, указанному в разделе FROM. Если в таблицах и представлениях, используемых в запросе, содержится более одного столбца с одинаковым именем и этот столбец должен быть включен в результат выборки, то помимо имени самого столбца следует указать имя таблицы, к которой принадлежит столбец. То есть необходимо задать полное имя столбца в формате *table_name. column_name*.

– *expression*. Этот параметр подразумевает указание выражения, на основе которого будет формироваться содержимое столбца. Имя столбца является частным случаем выражения. В выражении допускается использование констант, функций, имен столбцов, а также любых разрешенных операций над ними. Следует учитывать, что по умолчанию столбец, содержимое которого формируется на основе вычисления выражения, не имеет никакого имени. Если требуется присвоить столбцу конкретное имя, то необходимо указать псевдоним столбца.

– AS *column_alias*. С помощью этого параметра можно определять (или изменять по сравнению с первоначальными именами) псевдонимы (alias) для столбцов. Обычно псевдонимы служат для формирования в результате выборки набора столбцов с разными именами, для присваивания столбцам более понятных названий на национальном языке, а также используются при формировании столбцов на основе вычисления выражений. Применение псевдонимов неизбежно при работе с подзапросами, т.к. для ссылки на столбцы подзапроса каждый из столбцов должен иметь конкретное уникальное имя.

Использование параметров раздела SELECT рассмотрим на следующих примерах.

Применим параметр DISTINCT для вывода имен студентов:

```
SELECT DISTINCT Имя FROM Студенты
```

Имя
Александр
Светлана
Юлия
Елена
Ольга

Если добавить ограничение на количество выводимых строк (40 %), то получим следующий результат:

```
SELECT DISTINCT TOP 40 PERCENT Имя FROM Студенты
```

Имя
Александр
Светлана

Применение псевдонимов и выражений иллюстрирует следующий запрос:

```
SELECT TOP 4 (Фамилия+" "+Имя+" "+Отчество) AS ФИО,
МестоРождения AS Город FROM Студенты
```

ФИО	Город
Багута Елена Петровна	Курск
Шумакова Светлана Васильевна	Железногорск
Тарасова Юлия Сергеевна	Курск
Семыкина Ольга Алексеевна	Железногорск

1.2 Раздел FROM

С помощью этого раздела указываются таблицы и представления, из которых будет производиться выборка данных. При этом все таблицы и представления, участвующие в выборке данных, должны быть указаны в разделе FROM. Кроме того, даже если ни один из столбцов

таблицы не включен в результат выборки, но используется в разделе WHERE, ORDER BY, GROUP BY или других, то имя этой таблицы также должно быть указано в разделе FROM. Так как выборка данных из таблиц и представлений не имеет принципиальных различий, далее будет рассматриваться только работа с таблицами.

Раздел FROM имеет следующий синтаксис:

FROM { <*table_sourse*> } [,...n]
<*table_sourse*> ::= *table_name* [AS *table_alias*]
| *derived_table* AS *table_alias* | <*joined_table*>

Рассмотрим подробно использование каждого из параметров:

table_name

Имя таблицы, из которой будет производиться выборка.

AS *table_alias*

С помощью этого параметра можно присвоить таблице псевдоним, под которым на нее можно будет ссылаться в запросе. Часто псевдонимы служат для упрощения вида запроса при работе с длинными именами таблиц. Допустим, если в запросе часто упоминается таблица `ДанныеОСотрудниках`, то ей можно присвоить псевдоним, например, `ДС`. Это позволит не писать каждый раз полное имя таблицы, а приводить короткое имя. Кроме того, псевдонимы таблиц активно используются при работе с подзапросами. Если в подзапросе будет выполняться обращение к той же таблице, что и в основном запросе, возможна ситуация, что подзапрос обращается к столбцу, а СУБД не может определить, какое значение нужно вернуть – текущее значение основного запроса или значение, обрабатываемое в подзапросе. Применение псевдонимов позволяет однозначно идентифицировать нужные данные.

derived_table* AS *table_alias

Эта конструкция позволяет использовать в запросе динамические таблицы. Динамические таблицы представляют собой набор данных, формируемых в момент выполнения запроса на основе информации,

возвращаемой подзапросом. Динамические таблицы не существуют физически в базе данных, и поэтому у них нет постоянного имени. Однако чтобы иметь возможность ссылаться на данные динамической таблицы, ей должно быть присвоено какое-то имя. Для этого используются псевдонимы, которые указываются с помощью параметра *table_alias*.

<joined_table>

Эта конструкция предназначена для выборки данных из таблиц, связанных посредством внешних ключей. Помимо этого, конструкция *<joined_table>* используется и при связывании таблиц, между которыми не установлены ограничения целостности. Так, например, часто бывает необходимо в создаваемом запросе связать таблицы каким-то особым образом, отличным от общей структуры взаимосвязей между таблицами базы данных. Синтаксис конструкции таков:

<joined_table> ::=
<table_sourse> *<join_type>* *<table_sourse>* ON *<search_condition>*
| *<joined_table>*

Рассмотрим назначение и использование каждого из аргументов:

– *<table_sourse>*. С помощью этого параметра указывается источник, из которого будут браться данные. В качестве источника может выступать стандартная статическая таблица, представление или динамическая таблица. Поскольку конструкция *<joined_table>* является частью конструкции *<table_sourse>*, то имеет место рекурсивная зависимость конструкций *<table_sourse>* и *<joined_table>* друг от друга, что позволяет создавать многоуровневые зависимости между таблицами.

– *<join_type>*. Это параметр определяет метод связывания данных двух таблиц между собой. От выбранного метода связывания зависит, какие конкретно строки каждой из связываемых таблиц будут рассматриваться при выполнении запроса. Структура конструкции *<join_type>* следующая:

<join_type> ::= { INNER | LEFT | RIGHT } JOIN

Рассмотрим назначение параметров и их использование:

– INNER. При задании этого типа связывания каждая из двух

участвующих в связывании таблиц будет включать только те строки, для которых есть соответствие во второй таблице. Соответствие определяется условием связывания, которое задается с помощью параметра <*search_condition*>. Например, если в качестве условия связывания таблиц `Студенты` и `Оценки` рассматривать столбец `КодСтудента`, то оно будет выглядеть как `Студенты.КодСтудента=Оценки.КодСтудента`. При использовании связывания INNER в каждую таблицу будут включены только те строки, для которых значения `КодСтудента` имеются как в левой (`Студенты`), так и в правой (`Оценки`) таблицах.

– LEFT. При использовании этого типа связывания в левой таблице будут оставлены все строки независимо от того, есть ли для них соответствие в правой таблице. Применительно к предыдущему примеру это означает, что если в таблице Студенты описаны учащиеся вуза, у которых отсутствуют отметки в таблице Оценки, то строки, соответствующие этим студентам, все равно окажутся доступными. В правой же таблице будут доступны только те строки, для которых имеется соответствие в левой таблице. В столбцах, соответствующих отсутствующим данным правой таблицы, будут выведены значения `Null`.

– RIGHT. Действие этого параметра обратно предыдущему. При использовании указанного типа связывания в правой таблице будут оставлены все строки независимо от того, есть ли для них соответствие в левой таблице. В результате выборки столбцы, отображающие данные левой таблицы, будут содержать значения Null во всех строках, для которых нет соответствия.

Замечание. Тип связывания INNER можно легко заменить использованием оператора = в разделе WHERE.

Использование параметров раздела FROM рассмотрим на следующих примерах.

Для начала приведем пример применения в запросе двух таблиц с указанием псевдонимов:

```
SELECT TOP 5 П.Фамилия AS Преподаватель,
Д.Название AS Дисциплина, Д.Семестр AS №
FROM Преподаватели AS П, Предметы AS Д
WHERE П.КодПреподавателя = Д.КодПреподавателя
```

Преподаватель	Дисциплина	
Кирильчук	Прикладная информатика	
Кирильчук	ГИС	
Кирильчук	Прогнозирование ЧС	
Томаков	Надежность систем	
Тимофеев	БЖД	

В следующем примере в разделе FROM формируется динамическая таблица, которая будет доступна под псевдонимом Итог:

```
SELECT Итог+" по '"+Название+"'" AS Итоги
FROM Предметы,
  (SELECT О.КодПредмета, Фамилия+" "+Left(Имя,1)+"."+Left(Отчество, 1)
  +". имеет "+Str(Оценка) AS Итог
   FROM Студенты AS С, Оценки AS О
   WHERE С.КодСтудента = О.КодСтудента) AS ТИ
WHERE Предметы.КодПредмета = ТИ.КодПредмета
```

Итоги
Багута Е.П. имеет 4 по 'Надежность систем'
Шумакова С.В. имеет 3 по 'Надежность систем'
Багута Е.П. имеет 4 по 'Прикладная информатика'
Багута Е.П. имеет 4 по 'Промышленная экология'
Шумакова С.В. имеет 4 по 'Промышленная экология'
Тарасова Ю.С. имеет 3 по 'Промышленная экология'
Семыкина О.А. имеет 5 по 'Промышленная экология'

Следующий пример иллюстрирует использование связывания INNER для достижения того же результата, что и предыдущий пример:

```
SELECT Фамилия+" "+ Left(Имя, 1)+"."+Left(Отчество, 1)
+". имеет "+Str(Оценка)+" по '"+Название+"'" AS Итоги
FROM Студенты AS С INNER JOIN (Предметы AS П INNER
JOIN Оценки AS О ON
П.КодПредмета = О.КодПредмета) ON С.КодСтудента =
О.КодСтудента
```

В рассмотренном примере используется двойное связывание. Сначала выполняется связывание таблиц Предметы и Оценки по столбцу КодПредмета, а затем – связывание таблиц Студенты и Оценки по столбцу КодСтудента. В итоге таблицы Студенты и Предметы также оказываются связанными. Для полноты картины можно привести еще один пример, приводящий к достижению того же результата, но выполняющий связывание таблиц при помощи раздела WHERE:

```
SELECT Фамилия+" "+Left(Имя,1)+"."+Left(Отчество,1)
+". имеет "+Str(Оценка)+" по '"+Название+"'" AS Итоги
FROM Студенты AS С, Оценки AS О, Предметы AS П
WHERE С.КодСтудента = О.КодСтудента AND
П.КодПредмета = О.КодПредмета
```

1.3 Раздел WHERE

Этот раздел предназначен для ограничения количества строк, включаемых в результат выборки. Будут включены только те строки, которые удовлетворяют указанному логическому условию. Синтаксис раздела следующий:

WHERE <*search_condition*>

Рассмотрим назначение и использование параметров:

<search_condition>

С помощью этой конструкции можно задать любое произвольное условие для выборки данных. Конструкция <*search_condition*> является

выражением, которое должно возвращать булево значение – TRUE (истина) FALSE (ложь). Указанное выражение вычисляется для каждой строки таблицы и, только если возвращается значение TRUE, то строка включается в результат выборки. В противном случае строка игнорируется. Обычно условие включает имена столбцов таблицы, вместо которых при сканировании для каждой строки подставляется конкретное значение. Однако в условии имена столбцов могут и не участвовать. Структура конструкции <*search_condition*> такова:

<*search_condition*> ::=
{
{ [NOT] <*predicate*> | (<*search_condition*>) }
[{ AND | OR } [NOT] { <*predicate*> | (<*search_condition*>) }]
} [,...n]

Основное назначение самой конструкции <*search_condition*> состоит в объединении множества логических условий, каждое из которых возвращает булево значение. Объединение выполняется с помощью операторов AND, OR и NOT. Кроме того, из синтаксиса конструкции видно, что можно создавать сложные вложенные условия. Само же условие определяется с помощью конструкции <*predicate*>, имеющей следующий синтаксис:

<*predicate*> ::=
{
expression { = | <> | > | >= | < | <= } *expression*
| *string_expression* [NOT] LIKE *string_expression*
| *expression* [NOT] BETWEEN *expression* AND *expressio*n
| *expression* IS [NOT] NULL
| *expression* [NOT] IN (*subquery* | *expression* [,...n])
| *expression* { = | <> | > | >= | < | <= } { ALL | SOME | ANY } (*subquery*)
| EXISTS (*subquery*)
}

Работа с частью операторов не вызывает особых затруднений. К таким операторам относятся **=**, **<**, **>** и т.д. Тем не менее, часть операторов и аргументов требует дополнительных комментариев:

– *expression*. Этот параметр означает использование любого выражения. В большинстве случаев выражение возвращает некоторый результат, который сравнивается с результатом вычисления другого выражения. Если в результате сравнения возвращается `TRUE`, то строка включается в результат выборки.

– *string_expression*. Этот аргумент подразумевает наличие выражения, возвращающего символьное значение. Для выполнения некоторых, специфических только для строк, операций применяется оператор LIKE, с помощью которого можно проверить соответствие строкового выражения шаблону. В шаблоне могут присутствовать следующие специальные символы (в скобках дан вариант, используемый в Microsoft SQL Server и на уровне встроенных в Windows средств доступа к данным):

- **Символ** * (%), которому соответствует любое количество любых символов (в т.ч. и ни одного) в сравниваемом выражении.
- **Символ** ? (_), вместо которого может быть подставлен один произвольный символ сравниваемого выражения.
- **Символы** [ ]. Вместо скобок может подставляться один символ из указанного набора. Набор разрешается задавать как перечислением всех возможных значений (например, [АБФЯ]), так и указанием диапазона ([д-о]). Допускается комбинирование обоих типов перечисления значений.
- **Символы** [!] ([^]). В этом случае в сравниваемом выражении допускается наличие любого символа, *не входящего* в указанный набор.

– *expression* [NOT] BETWEEN *expression* AND *expression*. С помощью этой конструкции можно выполнить проверку на принадлежность значения, возвращаемого при вычислении выражения, определенному диапазону. Границы диапазона также задаются как результат вычисления выражения. При использовании следует BETWEEN учитывать, что если проверяемое выражение совпадает с границами диапа-

зона, то будет возвращено TRUE.

– *expression* IS [NOT] NULL. Эта конструкция используется для проверки выражения на равенство значению NULL. Если результатом вычисления выражения является значение NULL, то выражение *expression* IS NULL вернет TRUE, результатом чего будет включение соответствующей строки в результат выборки. Заметим, что применять операторы = или <> для сравнения выражения и значения NULL нельзя, т.к. любое выражение с участием NULL возвратит NULL, а не TRUE или FALSE.

– *expression* [NOT] IN (*subquery* | *expression* [,...n]). С помощью оператора IN можно проверить вхождение выражения в набор данных, который может задаваться с помощью подзапроса или простого перечисления через запятую всех возможных значений. Оператор IN является своего рода аналогом оператора =ANY, однако спектр его применения более широк.

– *expression* { = | <> | > | >= | < | <= } { ALL | SOME | ANY } (*subquery*). С помощью этой конструкции производится проверка удовлетворения значения, возвращаемого при вычислении выражения, логическому условию для всех или хотя бы одного значения в подзапросе. Логические операторы ALL, SOME и ANY выполняют следующие действия:

- ALL. Сравнивает скалярное значение со всеми значениями в наборе. Если условие выполняется для всех значений набора, то возвращается TRUE. В противном случае возвращается FALSE.
- SOME и ANY. Эти два оператора идентичны. Оператор сравнивает указанное скалярное выражение со всеми значениями в наборе и возвращает TRUE, если логическое условие выполняется хотя бы для одного значения набора. Если же условие не выполняется ни для одного значения, то возвращается FALSE.

– EXISTS (*subquery*). Оператор EXISTS возвращает значение TRUE, если подзапрос возвращает хоть одну строку. В противном случае возвращается FALSE.

Рассмотрим несколько примеров по формированию условий отбора строк.

Например, запрос (функция Month() возвращает номер месяца из даты) SELECT Фамилия FROM Студенты WHERE Month(ДатаРождения) BETWEEN 6 AND 8 позволяет вывести список студентов, родившихся в летний период.

Разберем применение оператора LIKE. Так, запрос SELECT Фамилия FROM Студенты WHERE Фамилия Like "*кова" возвратит список фамилий, оканчивающихся на "кова".

Выполняя запрос SELECT Фамилия FROM Студенты WHERE Фамилия Like "?a*", получим список фамилий, второй буквой которых является буква "а".

Запрос SELECT Фамилия FROM Студенты WHERE Фамилия Like "[А-МС]*" выдаст список фамилий, начинающихся на буквы от "А" до "М", а также "С".

Запрос SELECT Фамилия FROM Студенты WHERE Фамилия Like "[!А-ЛЯ]*[о]" возвратит список фамилий, начинающихся на буквы от "М" до "Ю" и заканчивающихся на букву "о".

Использование оператора IN иллюстрирует следующий запрос:

SELECT Фамилия, МестоРождения AS Город FROM Студенты, [SELECT МестоРождения AS М FROM Студенты WHERE Фамилия IN ("Багута", "Кательникова")]. AS Г WHERE МестоРождения IN (Г.М)

Результатом запроса будет список студентов, родившихся в том же городе, что и студенты с фамилиями "Багута" и "Кательникова":

Фамилия	Город
Багута	Курск
Тарасова	Курск
Рагулина	Курчатов
Мазурова	Курск
Кательникова	Курчатов

Применение оператора ALL иллюстрирует запрос

SELECT Фамилия, МестоРождения FROM Студенты WHERE КодСтудента<> All (SELECT КодСтудента FROM Студенты WHERE МестоРождения="Курск")

Он формирует список студентов, родившихся вне г. Курска:

Фамилия	МестоРождения
Шумакова	Железногорск
Семыкина	Железногорск
Рагулина	Курчатов
Кательникова	Курчатов
Феоктистов	Железногорск
Горемыкина	Курчатов

1.4 Раздел GROUP BY

С помощью этого раздела можно осуществить группировку данных. Данные группируются по одному или более столбцам таким образом, что для всех строк с одинаковыми значениями в столбце, по которому выполняется группировка, в результате выборки будет возвращена всего одна строка. При этом в результат выборки разрешается включение только столбцов, по которым осуществляется группировка, а также столбцов, использующих функции агрегирования.

Синтаксис раздела таков:

GROUP BY *group_by_expression* [,...n]

С помощью параметра *group_by_expression* указывается выражение, по которому будет выполняться группировка. В качестве такого выражения может выступать как имя отдельного столбца, так и сложное выражение, в котором используются ссылки на несколько столбцов и различные операторы. Выражение, по которому производится группировка, может быть включено в раздел SELECT.

Необходимо отметить, что в раздел SELECT нельзя включать непосредственно имена столбцов, не указанных в разделе GROUP BY.

Также нельзя использовать имена таких столбцов в любых выражениях. В непосредственном виде допускается применение только имен столбцов, по которым выполняется агрегирование.

Агрегирование осуществляется специальными функциями, возвращающими для всех строк группы единственное значение, которое и включается в результат выборки:

– COUNT (*column_name*). Возвращает количество строк в группе с непустым значением в указанном столбце.

– COUNT (*). Возвращает общее количество строк в группе, включая строки с неопределенным значением.

– MAX (*column_name*). Возвращает максимальное значение в указанном столбце в пределах группы.

– MIN (*column_name*). Возвращает минимальное значение в указанном столбце в пределах группы.

– SUM (*column_name*). Возвращает сумму всех значений в пределах группы в указанном столбце. Эта функция может применяться только к столбцам с числовым типом данным.

– AVG (*column_name*). Возвращает среднее арифметическое для указанного столбца в пределах строк, принадлежащих группе. Эта функция может применяться только к столбцам с числовым типом данным.

Помимо указания в разделе SELECT непосредственно функций агрегирования, также допускается использование различных выражений, построенных на основе этих функций и столбцов, по которым осуществляется группировка. Кроме того, функции агрегирования могут применяться не только непосредственно к определенному столбцу, но и к различным выражениям, построенным на основе этих столбцов.

Приведем несколько примеров, демонстрирующих сказанное.

Запрос

SELECT Фамилия, Day(ДатаРождения)*Sum(Оценка)/31 AS К, Avg(Оценка) AS [Ср оценка], Min(Оценка) AS [Мин оценка] FROM Студенты INNER JOIN Оценки ON Студенты.КодСтудента = Оценки.КодСтудента GROUP BY Фамилия, ДатаРождения выполняет группировку данных по фамилиям сту-

дентов. Помимо фамилий в результат включаются три столбца. В первом вычисляется выражение на основе суммарного полученного балла и дня рождения, во втором и третьем – средняя и минимальная оценки в пределах группы. Будет возвращен результат:

Фамилия	К	Ср оценка	Мин оценка
Багута	0,806451612903226	3,125	2
Кательникова	21,2903225806452	3,75	1
Мазурова	1,67741935483871	3,25	1
Рагулина	1,74193548387097	3,375	2
Семыкина	1,93548387096774	3,75	2
Стыценко	1,87096774193548	3,625	2
Тарасова	2,12903225806452	4,125	3
Шумакова	1,74193548387097	3,375	2

Следующий пример демонстрирует использование функций агрегирования не к отдельным столбцам, а к различным выражениям, построенным на основе этих столбцов:

```
SELECT Фамилия, Max(Оценка*Sin(КодОценки) + Cos(О.КодСтудента)) AS К FROM Студенты AS С INNER JOIN Оценки AS О ON С.КодСтудента = О.КодСтудента GROUP BY Фамилия
```

Будет возвращен результат:

Фамилия	К
Багута	4,51179289820629
Кательникова	4,06300369490735
Стыценко	3,74062580290372
Тарасова	3,92794623057128
Шумакова	2,31690002239734

Приведем пример группировки строк таблицы Студенты по

столбцу Имя с вычислением количества студентов с одинаковым именем:

SELECT Имя, Count(*) AS [Кол-во] FROM Студенты GROUP BY Имя

Имя	Кол-во
Александр	2
Вера	1
Екатерина	1
Елена	2
Олеся	1
Ольга	2
Светлана	1
Татьяна	2
Юлия	1

Если же провести группировку по первой букве имени:
SELECT Left(Имя,1) AS Б, Count(Имя) AS [Кол-во]
FROM Студенты GROUP BY Left(Имя,1)
то получим такой результат:

Б	Кол-во
А	2
В	1
Е	3
О	3
С	1
Т	2
Ю	1

Рассмотрим пример подсчета количества студентов, сгруппировав

их по критерию успеваемости (функция Int() округляет значение параметра до ближайшего целого в меньшую сторону):

```
SELECT Инфо.СрОценка AS [Учатся на],
Count(Инфо.КодСтудента) AS [Кол-во]
FROM [SELECT КодСтудента, Int(Avg(Оценка)) AS СрОценка FROM Оценки GROUP BY КодСтудента]. AS Инфо GROUP BY Инфо.СрОценка
```

Учатся на	Кол-во
2	1
3	3
4	3
5	2

1.5 Раздел HAVING

Этот раздел обычно используется совместно с разделом GROUP BY, дополняя его. Назначением раздела HAVING является ограничение набора строк, подвергаемых группировке. По своим функциям раздел HAVING очень близок к разделу WHERE. Синтаксис раздела HAVING таков:

```
HAVING <search_condition>
```

Применение конструкции <*search_condition*> при работе с разделом HAVING ничем не отличается от указания этой же конструкции при работе с разделом WHERE. В одном запросе допускается использование как раздела WHERE, так и раздела HAVING. При этом каждый из них будет вести себя по-своему. Например, при выполнении запроса

```
SELECT Фамилия, Avg(Оценка) AS СредОценка FROM Студенты INNER JOIN Оценки ON Студенты.КодСтудента = Оценки.КодСтудента WHERE Фамилия Like "*а" GROUP BY Фамилия HAVING Avg(Оценка)>2.5
```

будет возвращен такой результат:

Фамилия	СредОценка
Кательникова	3,25
Мазурова	2,75
Тарасова	3
Шумакова	2,75

В следующем примере приводится запрос, позволяющий узнать, какие студенты могут претендовать на получение диплома с отличием (критерий отбора – 60 % пятерок и отсутствие троек):

```
SELECT Фамилия, Avg(Оценка) AS [Средний балл]
FROM Студенты INNER JOIN Оценки ON
Студенты.КодСтудента = Оценки.КодСтудента
GROUP BY Фамилия
HAVING (((Avg(Оценка)-4)*100)>=60) AND (Min(Оценка)>=4)
```

Фамилия	Средний балл
Кательникова	5
Мазурова	4,75
Семыкина	4,875
Стыценко	4,625

6. РАЗДЕЛ ORDER BY

Часто возникает необходимость упорядочить данные по различным критериям, причем иногда по нескольким сразу. Для выполнения сортировки данных, возвращаемым запросом, в распоряжении пользователей имеется раздел ORDER BY, специально предназначенный для определения параметров порядка вывода строк. Синтаксис этого раздела следующий:

ORDER BY { *order_by_expression* [ASC | DESC] } [,...n]

Рассмотрим назначение аргументов раздела:

order_by_expression

Этот параметр определяет выражение, в соответствии с которым будет выполняться сортировка данных. В качестве выражения сортировки может указываться имя, псевдоним или любое выражение. Также допускается указание номера столбца результата выборки, по которому следует осуществлять сортировку. Следует отдельно отметить, что сортировка не обязательно должна производиться по выражению, включенному в результат выборки.

ASC | DESC

Эти параметры указываются дополнительно к параметру *order_by_expression* и определяют порядок сортировки. При использовании ASC данные располагаются по возрастанию, тогда как параметр DESC определяет порядок сортировки по убыванию. Если порядок сортировки не указан явно, то по умолчанию данные будут располагаться по возрастанию.

[,...n]

Этот параметр говорит о том, что в одном запросе сортировка может выполняться по нескольким критериям. Для этого необходимо через запятую указать выражения, по которым будет выполняться сортировка, а также дополнительно и порядок сортировки. Порядок указания выражений сортировки определяет приоритет того или иного выражения.

Рассмотрим несколько примеров. Запрос

```
SELECT Фамилия, Имя, Отчество
FROM Студенты ORDER BY Фамилия, 2, Отчество DESC
```

отсортирует строки следующим образом:

Фамилия	Имя	Отчество
Багута	Елена	Петровна
Воротникова	Елена	Григорьевна
Горемыкина	Татьяна	Алексеевна
Кательникова	Вера	Сергеевна

Выполнение сортировки по выражению, не включенному в ре-

зультат выборки, можно проиллюстрировать таким запросом:

```
SELECT Фамилия, Month(ДатаРождения) AS Месяц
FROM Студенты INNER JOIN Оценки ON
Студенты.КодСтудента = Оценки.КодСтудента
GROUP BY Фамилия, Month(ДатаРождения)
ORDER BY Avg(Оценка)*100/5+Count(Оценка)
```

Будет возвращен следующий результат:

Фамилия	Месяц
Багута	1
Стыценко	2
Рагулина	2
Семыкина	2
Тарасова	2
Кательникова	12

2 Порядок выполнения работы

Согласно выданному при проведении лабораторной работы "Проектирование структур баз данных информационных систем" варианту составить запросы к базе данных, созданной в работе "Реализация структур баз данных в системах управления базами данных".

Созданные запросы должны решать следующие задачи.

Вариант №1 – база данных по учету товаров в продуктовом магазине.

1. Сколько наименований товаров каждого типа хранится на складе?
2. Сколько различных товаров каждого типа хранится на складе?
3. Сколько различных товаров имеется от каждого поставщика?
4. От какого поставщика получено наибольшее количество товаров?
5. Какова стоимость товаров, полученных от каждого поставщика?
6. Какой поставщик поставил товаров на наибольшую сумму?
7. Сколько поставщиков имеется по каждому наименованию товара?
8. На каком типе товара специализируется каждый поставщик?

9. Какова величина товарооборота с каждым городом?
10. Какие товары были получены до определенной даты?
11. Цена каких наименований товаров лежит в заданном диапазоне?

Вариант №2 – база данных по учету результатов игр в КВН.

1. В каких играх участвовала каждая команда?
2. Сколько игр было у каждой команды?
3. С какими командами играла заданная команда?
4. Какие средние баллы получили команды по каждому конкурсу в определенной игре?
5. Какие итоговые баллы получили команды по всем играм?
6. Какие итоговые баллы получили команды в определенной игре?
7. Какие команды были финалистами?
8. Какая команда в последний раз стала чемпионом?
9. Сколько сидел в жюри каждый из судей?
10. Какие оценки предпочитает ставить каждый из судей?
11. Какой команде отдает предпочтение заданный судья?

Вариант №3 – база данных по учету вредных выбросов промышленных предприятий.

1. Какими веществами загрязняют окружающую среду предприятия города, и какова их средняя концентрация?
2. На каких предприятиях выбросы превышают ПДК, и когда это происходило?
3. Когда происходило превышение ПДК для заданного предприятия?
4. Какие вредные вещества выбрасываются по округам?
5. Выброс какого вещества преобладает в заданном округе?
6. В каком округе наибольшая концентрация предприятий?
7. Какие вещества оказывают наибольшее влияние на организм человека?
8. Заболевание каких органов может вызвать работа на данном предприятии?
9. Болезнь каких органов наиболее вероятна при работе на данном предприятии?
10. Узнайте для данного предприятия, в каком месяце был зафиксирован наибольший выброс определенного вещества, и недомогания каких органов это могло вызвать.

Вариант №4 – база данных по учету книг в библиотеке.

1. Сколько книг имеется в библиотеке по каждой теме?
2. Сколько учебников, справочников и т.п. имеется в библиотеке?
3. Сколько книг имеется в библиотеке?
4. Сколько книг находится на руках?
5. Выведите десятку наиболее читаемых книг.
6. Выведите тройку наиболее читаемых авторов.
7. Какая тематика наименее пользуется спросом?
8. Какие книги являются наиболее старыми?
9. Выведите список учебников, изданных за последние десять лет.
10. Какие авторы издали наибольшее количество различных книг?
11. Выведите список авторов, пишущих по одной тематике.

Вариант №5 – база данных по учету больных в поликлинике.

1. Какими болезнями люди чаще всего страдают?
2. По какому поводу люди обращаются чаще всего весной (осенью)?
3. Чем чаще болеет мужское население (женское)?
4. Кто чаще болеет – мужчины или женщины?
5. Чем чаще болеют мальчики (девочки)?
6. Какой человек чаще всего болел?
7. Какая семья чаще всего болела?
8. Выведите годовую динамику заболевших, родившихся в годы Второй мировой войны (в годы перестройки).
9. Были ли случаи групповых заболеваний в какой-либо из семей (обращение по одному поводу 2 и более человек, живущих в одном доме)?

Содержание отчета:

1. Наименование работы.
2. Цель работы.
3. Перечень вопросов, ставящихся к базе данных. Каждому вопросу должен соответствовать запрос, представленный на языке SQL, и отображаемый в табличной форме результат выполнения запроса.

Контрольные вопросы:

1. Перечислите разделы команды SELECT.

2. С помощью каких конструкций можно ограничить количество строк, которые будут включены в результат выборки?
3. Какими путями в команде SELECT можно задать список выводимых столбцов?
4. С какой целью в языке SQL применяются псевдонимы?
5. Каким образом в языке SQL обеспечивается связывание таблиц?
6. Поясните суть группировки данных.
7. В чем состоит разница между разделом WHERE и HAVING?
8. Как в команде SELECT задается порядок выводимых строк?

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра охраны труда и окружающей среды

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«14» 02 2018 г.

ОБРАБОТКА И АНАЛИЗ ПРОСТРАНСТВЕННО РАСПРЕДЕЛЕННОЙ ИНФОРМАЦИИ В ГИС MAPINFO

Методические указания к проведению
лабораторных и практических работ
для студентов направлений подготовки 20.03.01, 20.04.01
«Техносферная безопасность»

Курск 2018

УДК 371.64/.69:004
Составители: И.О. Кирильчук, А.В. Гнездилова

Рецензент
Кандидат технических наук, доцент *Г.П. Тимофеев.*

Обработка и анализ пространственно распределенной информации в ГИС MapInfo: методические указания к проведению лабораторных и практических работ / Юго-Зап. гос. ун-т; сост.: И.О. Кирильчук, А.В. Гнездилова. Курск, 2018. 25 с.

Описывается методика применения языка структурированных запросов, реализованного в ГИС MapInfo, для решения задачи получения и анализа пространственно распределенной информации.

Методические указания предназначены для студентов направлений подготовки 20.03.01, 20.04.01 Техносферная безопасность.

Текст печатается в авторской редакции

Подписано в печать 14.02.18. Формат 60х84 1/16.
Усл. печ. л. 1,45. Уч.-изд.л. 1,31. Тираж 30 экз. Заказ 1062. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Цель работы:

- изучение синтаксиса языка структурированных запросов, реализованного в ГИС MapInfo;
- получение навыков формирования запросов для обработки и анализа пространственно распределенной информации.

1 Общие положения

Программа MapInfo располагает двумя командами, которые выбирают объекты, формируя таблицу запроса: ***Выбрать*** и ***SQL-запрос***. Команда ***Выбрать*** составляет более простые запросы, чем команда ***SQL-запрос***.

Вы можете применить команду ***Выбрать*** к объектам, составляющим выборку, или ко всей таблице. Этой командой можно выделять записи, соответствующие объектам в окне ***Карт*** или строкам в окне ***Списка***, удовлетворяющие заданному критерию. Кроме того, создается таблица результатов запроса SELECTION (таблицу запроса), которую затем можно просматривать, отображать в окнах ***Карт*** и ***Графиков*** так же, как любую другую таблицу MapInfo.

В окне ***Списка*** объекты, удовлетворяющие критерию запроса, выделяются. В окне ***Карты*** графические объекты, соответствующие выбранным записям, выделяются с помощью линий и штриховки, выбранных в окне "Выделение объектов" диалога "Режимы окна Карты". При работе с окнами обоих типов объекты выделяются в обоих окнах. В любом случае MapInfo автоматически создает рабочую таблицу с названием "Selection", содержащую выборку по запросу. Вы можете просматривать эту таблицу, отображать ее в окнах ***Карт*** и ***Графиков*** так же, как и все остальные таблицы. Эту таблицу можно также записать, выполнив команду ***Файл→Сохранить копию***.

1.1 Выборка данных из одной таблицы

Диалог позволяет совершить выбор записей таблицы по заданному критерию (рисунок 1). Он содержит следующие элементы управления:

Выбрать записи из таблицы

Список содержит имена всех открытых таблиц. Откройте список, нажав на кнопку со стрелкой, справа от окошка, и выберите имя таблицы, из которой будут выбираться записи.

Согласно условию

Введите запрос, организованный в виде логического выражения. Для составления выражения можно использовать диалог "Выражение", вызываемый кнопкой "Составить".

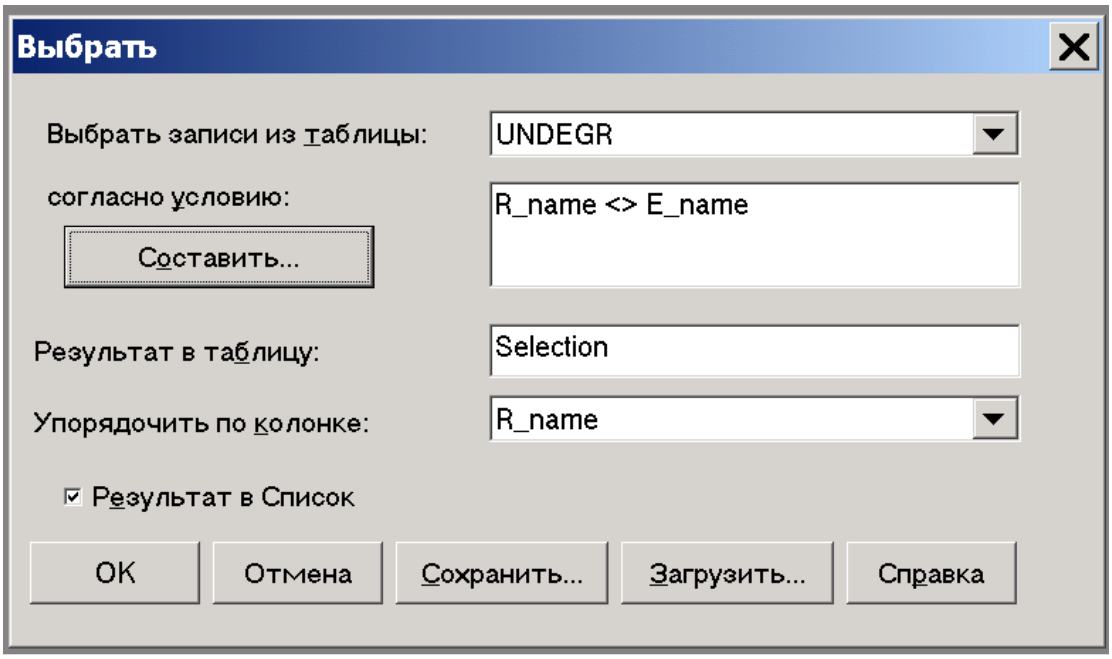


Рисунок 1 – Диалог выбора данных

Поместить результат в

В этом окне можно ввести название временной таблицы, содержащей результаты поиска. Стандартное название временной таблицы – "Selection" ("Выборка"). Если оставить стандартное название в этом окне, то MapInfo будет помещать результаты запроса в временные таблицы с названиями: "Запрос1", "Запрос2" и так далее.

Упорядочить по колонке

Этот режим позволяет выбрать колонку, по которой следует сортировать результаты поиска. Если этот режим не устанавливать, результаты сортироваться не будут. Вы можете сортировать таблицу по одной из колонок, выполнив команду ***Выбрать*** без задания выражения. Тогда MapInfo выберет все записи таблицы и отсортирует их по значениям в заданной колонке.

Составить

Вызывает диалог "Выражение".

Результат в список

При открытии диалога этот флажок установлен. Не сбрасывайте его, если хотите просматривать результаты поиска. Иначе сбросьте флажок. Окна ***Списков*** получат заголовки согласно значениям, выбранным в окне "Поместить результат в".

OK

Выполняет запрос.

1.2 Общая процедура создания SQL-запроса

SQL-запрос – это мощный инструмент извлечения информации. В одних случаях SQL-запрос помогает эффективно фильтровать данные, в других – сортировать и группировать, вычислять промежуточные суммы и т.д..

SQL расшифровывается как Structured Query Language – Структурированный Язык Запросов. Многие программные пакеты, работающие с базами данных, в том числе и MapInfo, поддерживают синтаксис команд SQL. Изучив версию SQL MapInfo, можно потом применить эти знания в других пакетах.

Основная процедура использования команды SQL-запрос следующая:

1. Откройте таблицу с данными на основе которых будет осуществляться запрос. Эту таблицу будем называть исходной.
2. Выберите команду ***Запрос/SQL-запрос***. Будет открыт диалог "SQL-запрос". Заполните окошки для определения запроса и нажмите на кнопку "OK". MapInfo выполнит запрос.

На основе данных исходной таблицы MapInfo строит специальную временную таблицу, которую в дальнейшем будем называть результирующей. Результирующая таблица состоит только из тех строк и колонок, которые отвечают критериям выполненного SQL-запроса. Стандартное имя для результирующей таблицы – Selection (если вы не изменили это имя в окне "И поместить в таблицу" в диалоге "SQL-запрос").

3. Откройте окно ***Карты*** или ***Списка*** с результирующей таблицей, если хотите просмотреть результаты запроса. Если в диалоге

был установлен флажок "Результат в список", то окно ***Списка*** с результатами запроса откроется автоматически после выполнения запроса.

Если оставить стандартное название результирующей таблицы Selection, то MapInfo именовать временные таблицы так: ЗАПРОС1, ЗАПРОС2 и так далее. Это происходит для того, чтобы зафиксировать результат запроса, так как таблица Selection постоянно меняется в зависимости от изменения выбора в окнах MapInfo.

Если было задано свое имя для результирующей таблицы в диалоге "SQL-запрос", то MapInfo не будет переименовывать результирующую таблицу в ЗАПРОСN.

4. MapInfo автоматически выбирает все строки в результирующей таблице после выполнения запроса. Таким образом, после выполнения SQL-запроса пользователь сразу может копировать эти строки.

Обычно изменения, сделанные в таблице запроса, автоматически дублируются в исходной (базовой) таблице. Например, пусть к таблице ORDERS, был применен SQL запрос и получена результирующая таблица. Тогда, если удалить из нее несколько строк, то в базовой таблице ORDERS также несколько строк будут удалены. Однако если запрос рассчитывает промежуточные суммы, то изменения результирующей таблицы не дублируются в оригинальной.

5. Используйте команду ***Файл/Сохранить копию***, если хотите иметь таблицу запроса как постоянную таблицу. Таблица, полученная в результате SQL-запроса, является временной и удаляется после завершения работы в MapInfo.

1.3. Диалог "SQL-запрос"

Диалог "SQL-запрос" может выполнять комплекс задач при построении запроса. Несмотря на обилие окон и режимов в этом диалоге, при составлении конкретного запроса обычно используется только часть из них (рисунок 2).

Выбрать колонки

Список колонок, которые будет содержать таблица запроса. Если Вы хотите использовать все колонки, поставьте в этом окне звездочку. При перечислении названия колонок разделяются запятыми

Выбрать колонки: *Область*, *Население*, *Площадь*

из таблиц

Перечислите таблицы, данные из которых будут использоваться. При многошаговом объединении должны быть указаны базовые таблицы, т.е. те, которые были сохранены на диске. Нельзя указывать таблицы запросов (промежуточные при многошаговом объединении) в сложном SQL-запросе.

Если вводится несколько таблиц, то надо задать выражение для объединения таблиц в окне "с условием". Если введено две таблицы, то MapInfo автоматически попытается задать объединение. Для трех или более таблиц объединение нужно уже задавать вручную.

с условием

Укажите, какие записи (строки) из исходных таблиц нужно вносить в таблицу запроса. Условие задается обычным образом с помощью переменных (колонок) и отношений между ними (операторов). Например,

с условием: *Население > 5000000.*

Вы можете использовать любую колонку любой исходной таблицы, независимо от того, указана ли она в окне "Выбрать колонки". Можно ввести название колонки или ее номер в окне "Выбрать колонки": "col1", "col2" и "col6" указывают соответственно на первую, вторую и шестую колонки в этом окне. Перед номером должны стоять буквы "col".

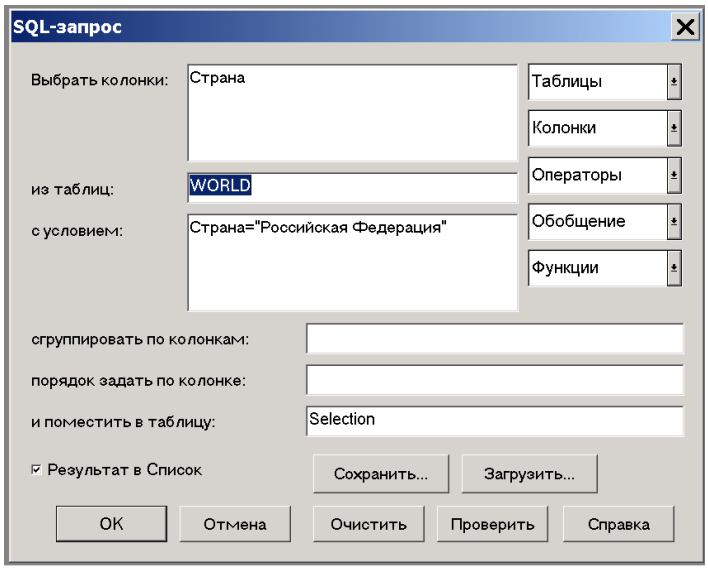


Рисунок 2 – Диалог построения SQL-запроса

Окно "с условием" должно быть заполнено при выборе по нескольким таблицам: оно будет задавать для MapInfo условие объединения таблиц. В окне "с условием" может быть просто указано, значения каких полей в двух таблицах должны совпадать: *ТАБЛИЦА_А.НОМЕР = ТАБЛИЦА_Б.ЗАКАЗ*. Так как допустимо использование нескольких таблиц при поиске, перед названием колонки должно быть указано название таблицы.

В случае поиска по нескольким таблицам при задании критерия в окне "с условием" необходимо следить за порядком задания названий колонок.

В окне "с условием" нельзя использовать функции обобщения.

сгруппировать по колонкам

В этом режиме строки таблицы запроса группируются так, что все строки с одинаковыми значениями в заданной колонке будут объединены. При использовании функций обобщения (**Count**, **Sum**, **Avg**, **Min**, **Max**) строки с одинаковыми значениями во всех группируемых колонках рассматриваются как группа, то есть повторение строк игнорируется, а на их место помещается некое обобщенное значение – количество, сумма, среднее, минимум или максимум.

Не обязательно использовать окно "Сгруппировать по колонкам". Если в этом окне не было ничего задано, то MapInfo не будет вычислять промежуточные суммы.

порядок определить по колонке

Задайте колонку, в соответствии с которой MapInfo расположит записи в таблице запроса. Стандартный порядок сортировки записей в MapInfo – по возрастанию (алфавитный порядок для символов). Если указать больше одной колонки в данном окне, то MapInfo начнет сортировку по первой из указанных колонок, затем для записей с совпавшими значениями будет проведена сортировка по второй указанной колонке и так далее (количество задаваемых колонок не ограничено).

Если необходимо упорядочить записи в порядке убывания, поставьте ключевое слово "desc" после названия колонки через пробел:

порядок определить по колонке: *Население desc*

Не обязательно использовать окно "порядок определить по колонке". Если в этом окне ничего не задано, то MapInfo не будет сортировать строки.

и поместить в таблицу

Задайте имя таблицы запроса. Если в окне оставлено стандартное имя SELECTION, MapInfo именует таблицы запросов "Запрос1", "Запрос2" и так далее. Также можно задать собственное название, заменив им слово "Selection" в окне "поместить в таблицу":

и поместить в таблицу: *Результат*

Результат в Список

После выполнения запроса при установленном режиме будет открыто окно ***Списка*** с таблицей запроса. Все записи в таблице запроса будут выбраны.

Пять списков в верхнем левом углу диалога

Используются для вставки имен таблиц, колонок и составления выражений в окнах диалога. Вставка осуществляется в окно, где в данный момент находится текстовый курсор.

Например, для заполнения окна "из таблиц":

1. Укажите не окно и в нем появится мерцающая вертикальная черточка - текстовый курсор.
2. Нажмите на кнопку со стрелкой справа от меню "Таблицы". Откроется список с именами открытых таблиц.
3. Выберите одно из них в списке.

MapInfo скопирует его в окно "из таблиц". Если дополнительно выбрать имя другой таблицы, то оно будет помещено за первым через запятую.

1.4 Предложение SELECT COLUMNS

1.4.1 Выбор колонок для таблицы запроса

Сколько колонок было задано в окне "Выбрать колонки", из стольких колонок будет состоять таблица запроса. Это бывает полезно, если работа осуществляется с таблицей, имеющей большое число колонок, но используется только несколько колонок.

Для определения колонок таблицы запроса:

1. Введите имя таблицы в окно "из колонки". Ввести имя таблицы (или имена таблиц через запятую) можно вручную или мож-

но воспользоваться меню "Таблицы". Поместите курсор в окно "из колонки" и просто выберите имя таблицы в меню. Оно появится автоматически в окне.

2. Поместите курсор в окно "Выбрать колонки".

3. Удалите звездочку, если не хотите создать таблицу запроса из всех колонок заданных таблиц в окне "из колонки". Колонки будущей таблицы запроса могут быть заданы либо звездочкой, либо списком колонок и/или выражений.

4. Для выбора колонки используйте меню "Колонки" или вводите их имена вручную.

5. Если вставляется второе и следующие имена колонок посредством выбора их в списке меню "Колонки", MapInfo автоматически будет вставлять запятые между именами колонок.

1.4.2 Использование окошка "Выбрать колонки" в SQL-запросе

В окне "Выбрать колонки" задаются колонки, которые будут составлять таблицу запроса.

Если необходимо, чтобы таблица запроса состояла из тех же колонок, что и исходная таблица, то в окне "Выбрать колонки" должна стоять только звездочка (*).

Если же задается набор колонок, отличный от исходной таблицы, то звездочка не используется. Колонки задаются списком через запятую. Можно указывать два типа колонок:

- колонки одной из исходных таблиц;
- вычисляемые колонки.

Если используется несколько исходных таблиц, перед названием колонки должно указываться название таблицы и точка. Название "RUSSIA.Область" обозначает колонку "Область" в таблице областей, а "RUSSIA.Население" - колонку "Население" в таблице областей. Если выбрать название колонки из списка, MapInfo автоматически укажет для него название таблицы.

1.4.3 Создание вычисляемой колонки

Команда SQL-запрос может создавать вычисляемые колонки и помещать их в таблицы запроса. Вычисляемая колонка является спе-

циальной временной колонкой, значения которой MapInfo динамически вычисляет, используя значения из других колонок исходной таблицы.

Например, таблица содержит поля населения ПОКУПКИ92 и ПОКУПКИ93. Чтобы динамически сложить значения из обоих колонок и поместить их в результирующую колонку, надо записать в окно "Выбрать колонки" выражение:

ПОКУПКИ92 +ПОКУПКИ93.

Аналогично, можно составить колонку полных имен клиентов, имея колонку имен и фамилий и задав выражение в форме:

ИМЯ + " " + ФАМИЛИЯ.

Для того, чтобы колонку можно было использовать при построении вычисляемой колонки, соответствующая таблица должна быть упомянута в окне "из таблиц".

Чтобы задать выражение для вычисляемой колонки:

1. Перейдите в окно "Выбрать колонки".
2. Удалите из него звездочку.
3. Введите выражение для колонки. Оно должно состоять из имен существующих колонок и арифметических знаков (+, - и т.д.). Вы также можете применять функции из списка "Функции".
4. Если хотите, задайте псевдоним для результирующей колонки – строку в кавычках.

Если псевдоним не задан, то MapInfo создаст его автоматически из выражения (например, ИМЯ_ФАМИЛИЯ).

5. В окне "Выбрать колонки" можно создавать несколько вычисляемых колонок сразу. Определения должны быть разделены запятыми.

В следующем примере вычисляемая колонка создается из значений двух других (подразумевается, что они числовые):

Выбрать колонки: *ПОКУПКИ92 + ПОКУПКИ92*

В следующем примере та же вычисляемая колонка приобретает имя ("Сумма_Покупок"):

Выбрать колонки: *ПОКУПКИ92 + ПОКУПКИ92 "Сумма_Покупок"*

В следующем примере вычисляется, сколько миллионов человек составляет Население:

Выбрать колонки: *Население / 1000000 "Миллионов"*

В следующем примере две строки сливаются в одну и результат помещается в вычисляемую колонку с заданным именем:

Выбрать колонки: *ИМЯ + " " + ФАМИЛИЯ "Полное_Имя"*

Знак "+" для строчных переменных работает как оператор слияния (конкатенации).

В следующем примере в выражении для вычисляемой колонки применяется функция **Proper$**, которая делает первую букву в слове прописной.

Выбрать колонки: *Proper$(Имя + " " + Фамилия) "Full_Name"*

В следующем примере функция **Format$** используется, чтобы расставить запятые (разделители тысяч) в больших числах. Подразумевается, что колонка ПОКУПКИ93 - числовая:

Выбрать колонки: *Format$(ПОКУПКИ93, "$,#") "Покупки_1993"*

Инструкция "$,#" в функции **Format$** показывает знак доллара в начале результирующего значения и добавляет запятую как разделитель тысяч.

В следующем примере вычисляются площади объектов, присоединенных к записям базовой таблицы:

Выбрать колонки: *Area(Obj, "sq km") "Net_Area"*

Obj – это специальное имя, представляющее географический объект, присоединенный к каждой записи таблицы.

1.5 Предложение WHERE

1.5.1 Географическое объединение таблиц

Если две таблицы имеют графические объекты, то MapInfo может объединить эти таблицы на основе пространственных отношений между объектами этих таблиц. Поэтому если таблицы не содержат общей колонки, то можно объединить их географически.

В MapInfo имеется несколько географических операторов. Они используются для выбора объектов на основании их взаимного расположения в пространстве. С географическими операторами в MapInfo используется специальное ключевое слово: "obj" или "object". Оно определяет, что MapInfo должно вычислить значение на

основании графических объектов, а не соответствующих им в таблице числовых полей.

Имя географического оператора указывается между географическими объектами; выбрать его можно в списке "Операторы" в диалоге "SQL-запрос":

Contains "Содержит". Объект А содержит объект Б, если центроид Б лежит в границах А.

Contains Entire "Полностью содержит". Объект А полностью содержит объект Б, если граница Б полностью лежит внутри границ А.

Within "Внутри". Объект А лежит внутри объекта Б, если его центроид лежит в границах Б.

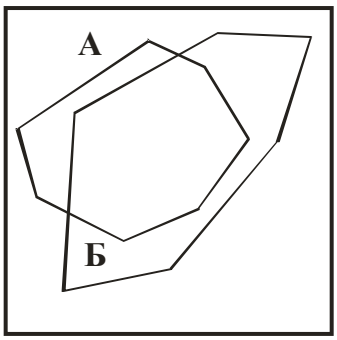
Entirely Within "Полностью внутри". Объект А лежит полностью внутри объекта Б, если его граница полностью лежит внутри границ Б.

Intersects "Пересекает". Объект А пересекается с объектом Б, если они имеют хотя бы одну общую точку.

Различие между **Contains** и **Within**, с одной стороны, и **Contains Entire** и **Entirely Within**, с другой, состоит в том, что **Contains** и **Within** основаны на анализе центроида объекта, а **Contains Entirely** и **Entirely Within** - на анализе всего объекта. Рис.3 объясняет это различие.

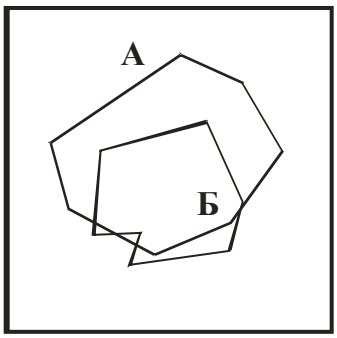
В обоих случаях объект А содержит объект Б, так как центроид объекта Б лежит внутри границы объекта А. Однако, на рисунке слева часть объекта Б лежит вне границ объекта А. А на рисунке справа весь объект Б лежит внутри объекта А. Только во втором случае мы говорим, что "объект А полностью содержит объект Б" или что "объект Б лежит полностью внутри объекта А". Далее, из того, что А полностью содержит Б, следует, что А содержит Б; а из того, что А полностью лежит внутри Б, следует, что А лежит внутри Б.

При этом следует помнить, что MapInfo выполняет простые операции "Содержит" (**Contains**) и "Внутри" (**Within**) гораздо быстрее, чем "Содержит полностью" (**Contains Entire**) и "Полностью внутри" (**Entirely Within**). Поэтому, если Вам не обязательно точно знать, полностью ли один объект содержит другой, используйте **Contains** и **Within** вместо **Contains Entire** и **Entirely Within**.



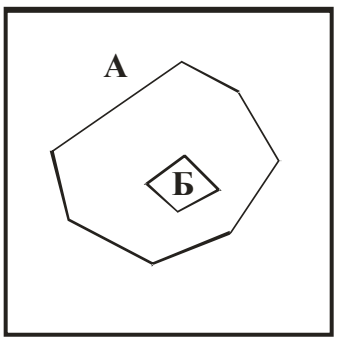
Объект А Содержит (**Contains**) Объект Б

Объект Б Внутри (**Within**) Объекта А



Объект А Содержит (**Contains**) Объект Б

Объект Б Внутри (**Within**) Объекта А



Объект А Полностью Содержит (**Contains Entire**) Объект Б

Объект Б Полностью Внутри (**Entirely Within**) Объекта А

Объект А Пересекает (**Intersects**) Объект Б

Объект Б Пересекает Объект А

Рисунок 3 – Различие графических операторов

Географические операторы удобно использовать при работе с несколькими таблицами. Если в таблицах нет колонки, которая определяла бы порядок объединения, можно задать объединение с помощью географических операторов (в окне "с условием"). При работе с таблицами городов и штатов можно использовать один из следующих вариантов:

CITIES.obj within STATES.obj

STATES.obj contains CITIES.obj

В обоих случаях MapInfo ищет внутри каждого штата города и ставит в соответствие строке города строку этого штата. С помощью функций обобщения можно сосчитать число городов в штате или

найти какие-нибудь средние характеристики для городов каждого штата.

Если имеется таблица графств и таблица покупателей, причем графства изображены многоугольниками, а покупатели - точками, Вы возможно задать географическое объединение вида:

CUSTOMER.obj within COUNTY.obj
COUNTY.obj contains CUSTOMER.obj

Географические операторы в частности используются в комбинации с подзапросами.

1.5.2 Задание условия для фильтрации данных

Фильтрация – это формирование запроса с условием, заданным логическим выражением, которое обычно сравнивает значение колонки с другой величиной. Например, в SQL-запросе будут рассматриваться только те строки, которые имеют в колонке "Количество" значение больше ста:

с условием: *Количество > 100*

Если при создании запроса используется окно "с условием", то в результирующей таблице будут только те строки, которые удовлетворяют заданному фильтру.

Фильтр может задаваться двумя или более логическими выражениями, если они разделены операционными словами **And** или **Or**. Если два выражения разделены словом **And**, то MapInfo извлечет строки, удовлетворяющие сразу двум условиям. Если два выражения разделены словом **Or**, то строки в таблице запроса будут удовлетворять одному из условий.

В окне "с условием" могут быть использованы любые колонки из исходных таблиц, включая те которые указаны в окне "Выбрать колонки".

Колонки в условии могут задаваться именами и номером в списке из окошка "Выбрать колонки". Номер колонки пишется после букв "col". Например, "col1" или "col6" - первая или шестая колонка таблицы запроса.

Окно "с условием" может быть использовано для фильтрации таблицы (извлечения из таблицы строк, удовлетворяющих определенному критерию) и для задания правила объединения данных в

таблицах, если для построения запроса используется несколько таблиц.

Замечание: Не допускается использование функций обобщения в окне "с условием".

1.5.3 Объединение двух или более таблиц

Обычно необходимая информация хранится в нескольких базах данных. SQL-запрос позволяет задавать отношения между различными базами, что дает возможность отображения на карте данных из многих баз одновременно.

Предположим, имеется таблица областей с демографическими данными – численностью жителей разных возрастов, этнических групп и профессий. Кроме того, имеется база данных о заказах из разных областей. Вы можете сравнить данные из этих двух таблиц, чтобы просмотреть демографические данные тех областей, откуда сделаны заказы, или сделать выборку по заказам и данным об областях.

Для этого надо уметь объединять таблицы между собой. Объединить две таблицы можно с помощью сравнения колонок с одинаковой информацией. В нашем примере, как таблица областей, так и таблица заказов должна содержать колонку с названием области. С помощью такой колонки MapInfo может сравнивать объекты в двух таблицах.

Таблица областей			Таблица заказов		
Область	Нас_1980	Нас_1990	Заказ	Покупатель	Область
Калмыкия	23,789	27,135	478001	Иванов	Калмыкия
Якутия	35,456	34,846	478002	Петров	Якутия
Бурятия	147,101	151,201	478003	Сидоров	Бурятия

При выполнении команды осуществляется сравнение данных в колонке "Область" таблицы областей и колонке "Область" таблицы заказов. Это позволяет MapInfo объединить данные о заказах с демографическими данными об областях.

Выбрать колонки: *

из таблиц: *РОССИЯ, ЗАКАЗ*

с условием: *РОССИЯ.Область = ЗАКАЗ.Область*

Также важно следить за тем, чтобы порядок колонок в окне "с условием" совпадал с порядком таблиц в окне "из таблиц".

Также рекомендуется при задании условия объединения помещать условие объединения на первое место в окне "с условием".

Количество строк в результате запроса зависит от того, насколько таблицы соответствуют друг другу. Например, в таблице ЗАКАЗЫ 10000 записей, и она связывается с таблицей ОБЛАСТИ, в которой 50 строк. В результирующей таблице может оказаться 10000 строк. Однако, если для записи из таблицы ЗАКАЗЫ не найдется подходящей в таблице ОБЛАСТИ, результат будет состоять менее чем из 10000 строк.

Итак, если вы хотите обновить одну колонку таблицы значениями из другой таблицы:

1. Объедините таблицы командой ***SQL-запрос***.
2. Примените команду ***Обновить колонку*** к выборке (Selection). Соответствующая базовая таблица будет автоматически обновлена.

1.5.4 Объединение двух таблиц по порядку строк

Если две таблицы имеют разные колонки, но одинаковое число строк, то их колонки можно объединить в одну таблицу по порядковым номерам строк. Если вы знаете, что первая таблица содержит N строк и вторая таблица также содержит столько же N строк, то объединение можно сделать, используя ссылку на колонку **RowID**.

Специальная невидимая колонка **RowID** содержит целочисленные номера строк таблицы. Так первая запись таблицы имеет значение в этой колонке 1, вторая - 2 и т. д.

Для объединения двух таблиц надо задать условие следующего вида:

с условием: *TABLE_1.RowID = TABLE_2.RowID*

1.5.5 Подзапросы

MapInfo допускает использование подзапросов в SQL-запросе. Подзапрос – это выбор, задаваемый в окне "с условием" диалога

"SQL-запрос". MapInfo сначала выполняет подзапрос, а затем уже использует его результаты в основном SQL-запросе.

Например, необходимо выбрать все области, население в которых больше среднего. Для этого надо указать в окне "с условием":

Население>среднее

Допустим, среднее значение неизвестно, однако MapInfo может вычислить его с помощью выражения:

Avg(Население)

Тогда вы можете составить частичный запрос или подзапрос с участием функции обобщения для вычисления средней численности населения для областей. Диалог SQL-запрос будет иметь вид:

Выбрать колонки: *

из таблиц: *RUSSIA*

с условием: *Население>(select Avg(Население) from RUSSIA)*

Подзапрос находится в окне "с условием" после оператора больше (>). Заметьте, что такие слова как "select" и "from" в окно "с условием" следует вводить, т.к. их нет в списках. Кроме того, подзапрос должен быть заключен в скобки.

Чаще всего используются подзапросы с предложениями **Select**, **From** и **Where**, то есть вида:

Select *колонки* **From** *таблица* **Where** *условие*

Рассмотрим пример SQL-запроса, выбирающий все города в областях с населением более 4,000,000 человек:

Выбрать колонки: *

из таблиц: *BIGTOWNS*

с условием: *obj within any (select obj from RUSSIA where Население > 4000000)*

Подзапрос выдает графические объекты для всех штатов с населением более 4,000,000. Основная процедура запроса выдает затем все города, находящиеся в областях, которые были выбраны подзапросом. Заметьте, что основная процедура запроса использует географический оператор (**Within**).

Заметьте также, что в окне "Из таблиц" указана только таблица BIGTOWNS, хотя работаем не только с ней, но и с таблицей RUSSIA.

Это возможно, так как не применяется объединение. RUSSIA просто использована в подзапросе.

В следующем примере выбираются все области, которые пересекаются (то есть граничат) с Московской областью.

Выбрать колонки: *

из таблиц: *RUSSIA*

с условием: *obj intersects (select obj from RUSSIA where Область = "Московская")*

Предложение в окне "с условием" имеет вид: "obj intersects obj". Второй объект, в свою очередь, представлен подзапросом:

select obj from RUSSIA where Область="Московская".

В подзапросе MapInfo находит графический объект, соответствующий Московской области, а затем в главном запросе находит объекты, пересекающиеся с ним. Похожий запрос можно использовать для нахождения всех участков улиц, пересекающих заданную улицу.

Теперь рассмотрим такой пример:

Выбрать колонки: *

из таблиц: *RUSSIA*

с условием: *RUSSIA.obj contains any (select obj from DEALERS)*

Этот запрос находит все области, в которых действует хотя бы один дилер. Основной поиск по полю "с условием" имеет вид: obj contains obj. Второй объект выбирается с помощью подзапроса:

select obj from DEALERS.

MapInfo находит строки для каждой области, содержащей объект, обозначающий дилера.

В заключение сделаем ряд замечаний о подзапросах:

- В подзапросах можно использовать таблицы, не упомянутые в окне "из таблиц". Но такие таблицы, конечно, должны быть перечислены в предложении From подзапроса.
- Если с подзапросом используется ключевое слово "any" или "all", он должен выдавать одну и только одну колонку. Следующее предложение некорректно:

Any(select Область, Население from RUSSIA)

– Если к подзапросу не относятся слова "any", "all" или "in", он должен возвращать единственное значение. Не допускается:

obj within (select obj from RUSSIA where Население > 4000000)

Если к подзапросу не относятся слова "any", "all" или "in", не допускается использование предложения "Сгруппировать по колонкам" в подзапросе.

– Нельзя вкладывать подзапросы друг в друга, т.е. в операторе **Select** может быть не более одного подзапроса.

1.6 Предложение ORDER BY

1.6.1 Использование окошка "порядок определить по колонке" в SQL-запросе

В окне "Порядок определить по колонке" можно задать порядок, в котором MapInfo расположит записи в таблице запроса. Стандартный порядок сортировки записей в MapInfo – по возрастанию (алфавитный порядок для символов). Если указать больше одной колонки в данном окне, то MapInfo начнет сортировку по первой из указанных колонок, затем для записей с совпавшими значениями будет проведена сортировка по второй указанной колонке и так далее (количество задаваемых колонок не ограничено).

В окне "Порядок определить по колонке" необходимо указывать названия или номера колонок. Если не производится объединение, можно использовать названия колонок. Если же группируются вычисляемые колонки по значениям или используются объединенные таблицы, надо указывать номер колонки (здесь буквы "col" перед номером не ставятся). Отметим, что можно сортировать записи не только по колонкам, указанным в окне "Выбрать колонки".

1.6.2 Сортировка строк в таблице запроса

По умолчанию, MapInfo сортирует строки в восходящем порядке. Так, при сортировке по значениям символьного поля в восходящем порядке после строки со значением, начинающемся на букву А, идет строка со значением, начинающаяся на букву Б. Если же имеет место сортировка по значениям численного поля, первой строкой будет та, которая имеет наименьшее значение.

Для сортировки в обратном, убывающем порядке используется слово **desc** после имени колонки в окне "порядок определить по колонке". Например, если формируется запрос из записей таблицы РОССИЯ, то возможен следующий критерий:

порядок определить по колонке: *Население desc*

Он сортирует строки в таблице запроса в убывающем порядке значений из колонки "Население".

Вы можете задать многоуровневую сортировку, где каждый уровень может быть задан как восходящем порядке, так и в обратном порядке. Например, сортировка строк из таблицы ГОРОДА по колонке "Область" в алфавитном порядке и в пределах одной области в убывающем порядке значений из колонки "Население":

Выбрать колонки: *

из таблиц: *Города*

порядок определить по колонке: *Область, Население desc*

Сортировка по колонке в окне "порядок определить по колонке" может быть задана двумя способами:

- Введите имя колонки (так как в примерах выше).
- Введите номер колонки из окошка "Выбрать колонки". 1 соответствует первой колонке в будущей таблице запроса.

Если проводится сортировка по значениям вычисляемой колонки и при этом вычисляемой колонке задан синоним, то можно использовать его как имя колонки в окне "порядок определить по колонке". Если синонима не задано, то подходит второй способ, т. е. по номеру колонки в таблице запроса.

1.7 Предложение GROUP BY COLUMNS

1.7.1 Использование окошка "сгруппировать по колонкам" в SQL-запросе

Окно "сгруппировать по колонкам" в диалоге "SQL-запрос" не является обязательным. Если задать в этом окне одну или более колонок, то таблица запроса будет содержать обобщающую информацию исходной таблицы.

В этом режиме строки таблицы запроса группируются так, чтобы все строки с одинаковыми значениями в заданной колонке будут объединены. При использовании функций обобщения (**Count**, **Sum**,

Avg, **Min**, **Max**) строки с одинаковыми значениями во всех группируемых колонках рассматриваются как группа, то есть повторение строк игнорируется, а на их место помещается некое обобщенное значение - количество, сумма, среднее, минимум или максимум.

Для задания критерия подмножества:

1. В окно "сгруппировать по колонкам" введите имя или номер колонки.
2. В окно "Выбрать колонки" введите имя колонки исходной таблицы.
3. В окно "Выбрать колонки" введите одну или более обобщающих функций (**Sum**, **Count**, **Avg**, **Min** или **Max**), не забывая разделять их запятыми.

В окне "Сгруппировать по колонкам" должны быть перечислены все колонки, введенные в окно "Выбрать колонки" и не использующие функции обобщения. По таким колонкам MapInfo проводит группировки. Каждому набору совпадающих данных в этих колонках в таблице запроса соответствует единственная строка. Вычисляемые колонки задаются номерами, которые обозначают их относительные позиции: 1, 2, и 5 будут обозначать первую, вторую и пятую колонки. Например:

Выбрать колонки: *Month(День_болезни), Count(*)*
из таблицы: *BOLESNI*
сгруппировать по колонкам: *1*

По этому запросу MapInfo сосчитает все записи, относящиеся к определенным датам, и выдаст таблицу запроса, в которой записи будут сгруппированы по этим датам. На каждый день будет выделена строка, и эта строка будет содержать число больных в этот день.

В окне "Сгруппировать по колонкам" колонку можно указывать по названию или по порядковому номеру. Если не производится объединение, можно использовать названия колонок. Если же группируются вычисляемые колонки по значениям или используются объединенные таблицы, то необходимо указывать номер колонки (здесь буквы "col" перед номером не ставятся).

Можно указать более одной колонки. При этом MapInfo сгруппирует записи по первой из указанных колонок. Внутри найденных групп записи будут сгруппированы по второй колонке и так далее.

Для каждой итоговой строки таблица запроса содержит обобщенные значения для всех колонок.

Замечание: Колонки в окне "Выбрать колонки", основанные на функциях обобщения, не могут быть указаны в окне "Сгруппировать по колонкам".

1.7.2 Примеры запросов с группировкой по колонкам

Предположим, что имеется таблица заказов. Каждому заказу соответствует отдельная строка в таблице. Одна колонка таблицы содержит фамилию торгового представителя, принявшего заказ, вторая - фамилию заказчика и третья - сумму заказа.

Вы хотите просмотреть:

- сколько заказов принял каждый торговый представитель;
- среднюю сумму заказа;
- общую сумму заказов.

Эту информацию можно получить, заполнив так следующие поля диалога "SQL-запрос":

Выбрать колонки: *Торг_пред, count(*), average(Объем), sum(Объем)*

из колонки: *ZAKAZ*

сгруппировать по колонкам: *Торг_пред*

Обратите внимание на функции обобщения в окне "Выбрать колонки" и окно "Сгруппировать по колонкам". MapInfo выполнит следующие действия:

1. Найдет все строки для каждого торгового представителя.
2. Посчитает число таких строк: **Count**(*).
3. Вычислит среднюю сумму заказа для данного торгового представителя: **Avg**(Объем).
4. Вычислит общую сумму заказов для данного торгового представителя: **Sum**(Объем).

MapInfo выполнит эти операции для каждого торгового представителя и создаст итоговую таблицу запроса, в которой каждому торговому представителю будет соответствовать одна строка. Функции обобщения обработают значения для всех строк, имеющих одно значение в поле "Торг_пред".

Рассмотрим следующий запрос SQL-запрос:

Выбрать колонки: *Покупатель, count(*), average(Объем), sum(Объем)*

из колонки: *ZAKAZ*

сгруппировать по колонкам: *Покупатель*

Это практически такой же запрос, как предыдущий, только группировка будет вестись по полю "Покупатель", а не "Торг_пред". Такой SQL-запрос найдет число, среднее значение и сумму для заказов, относящихся к одинаковым покупателям, а не торговым представителям.

Рассмотрим еще один пример SQL-запроса:

Выбрать колонки: *Торг_пред, Покупатель, count(*), average(Объем), sum(Объем)*

из колонки: *ZAKAZ*

сгруппировать по колонкам: *Торг_пред, Покупатель*

В окне "Сгруппировать по колонкам" заданы две колонки. В данном случае MapInfo будет группировать строки сначала по торговым представителям, а затем по фамилиям покупателей. В итоговой таблице для этого запроса отдельная строка будет соответствовать каждой новой паре “торговый представитель/покупатель”. Если заказчик делал заказы через разных торговых представителей, в таблице будет заведена строка, суммирующая его заказы через каждого из представителей. Строки будут сначала сгруппированы по торговым представителям, а затем - для каждого из них - по фамилиям заказчиков.

1.8 Функции обобщения MAPINFO

MapInfo располагает следующими функциями обобщения данных:

Count (*): Вычисляет количество записей в группе. В качестве аргумента указывается * потому, что функция применяется к целым записям, а не отдельным полям.

Sum (*выражение*): Вычисляет сумму значений в <выражении> для всех записей группы.

Avg (*выражение*): Вычисляет среднее значение в <выражении> для всех записей группы.

Max (*выражение*): Находит наибольшее значение в <выражении> для всех записей группы.

Min (*выражение*): Находит наименьшее значение в <выражении> для всех записей группы.

2 Порядок выполнения работы

1. Изучить состав и назначение элементов управления в окнах диалогов, соответствующих командам MapInfo ***Выбрать*** и ***SQL-запрос***.

2. Используя в качестве основы электронные карты г. Курска в масштабе 1:10000, составить запросы к данным различного вида. Исследовать влияние на результаты выборки возможности группировки и сортировки данных, применения условий, графического объединения таблиц, подзапросов. Изучить возможности, даваемые использованием функций обобщения и вычисляемых колонок.

3. Установить соответствие между видом запроса в окнах диалогов команд MapInfo ***Выбрать*** и ***SQL-запрос*** и конструкцией запроса на SQL-языке.

4. Провести анализ результатов выборки, используя окна ***Карты*** и ***Списка***.

Содержание отчета:

1. Наименование работы.
2. Цель работы.
3. Перечень запросов на языке SQL и соответствующие им результаты выборки данных. Результаты выборки представить в табличной форме на основе информации, выводимой в окнах ***Списков***.

Контрольные вопросы:

1. Какие команды ГИС MapInfo позволяют осуществить выборку данных?
2. Какие задачи позволяет решить диалог "SQL-запрос"?
3. Поясните назначение группировки данных.
4. Какова процедура создания вычисляемой колонки?
5. В каких случаях необходимо применение подзапросов?
6. Перечислите функции обобщения MapInfo.

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра охраны труда и окружающей среды

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«14» 02 2018 г.

ПРИМЕНЕНИЕ ВЫРАЖЕНИЙ В ВЫБОРКЕ ПРОСТРАНСТВЕННО РАСПРЕДЕЛЕННОЙ ИНФОРМАЦИИ В ГИС MAPINFO

Методические указания к проведению
лабораторных и практических работ
для студентов направлений подготовки 20.03.01, 20.04.01
«Техносферная безопасность»

Курск 2018

УДК 371.64/.69:004
Составители: И.О. Кирильчук, А.В. Гнездилова

Рецензент
Кандидат технических наук, доцент *Г.П. Тимофеев.*

Применение выражений в выборке пространственно распределенной информации в ГИС MapInfo: методические указания к проведению лабораторных и практических работ / Юго-Зап. гос. ун-т; сост.: И.О. Кирильчук, А.В. Гнездилова. Курск, 2018. 14 с.

Рассмотрен синтаксис выражений, применяемых в геоинформационной системе MapInfo для обработки и анализа пространственно распределенной информации.

Методические указания предназначены для студентов направлений подготовки 20.03.01, 20.04.01 Техносферная безопасность.

Текст печатается в авторской редакции

Подписано в печать 14.02.18. Формат 60х84 1/16.
Усл. печ. л. 0,81. Уч.-изд.л.0,73. Тираж 30 экз. Заказ 1069 . Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Цель работы: изучение синтаксиса выражений, применяемых в геоинформационной системе MapInfo; получение навыков формирования выражений при обработке и анализе пространственно распределенной информации.

1 Общие положения

1.1 Использование выражений

В некоторых диалогах MapInfo представлена возможность вызвать диалог "Выражение" для составления математических и символьных выражений, вычисления значений из колонок.

В разных командах выражения служат разным целям. Например:

- В выборе выражения участвуют в определении критерия выбора.
- В команде ***Обновить колонку*** выражение участвует для вычисления значений колонки.
- В тематической ***Карте*** выражения используются для вычисления величины, которая отображается на карте.
- В настройке слоя выражения используются для построения подписей.

Выражения можно разделить на две группы:

- Выражения, в результате вычисления которых получается логическая величина (TRUE или FALSE).
- Выражения, в результате вычисления которых получается численная или строковая величина.

Выражения первой группы обычно состоят из нескольких подвыражений и операторов сравнения между ними. Логические выражения могут участвовать в выборе объектов.

Выражения второй группы не используют операторы сравнения и не имеют подвыражений. Эти выражения используются в тематическом выделении объектов, обновлении колонок и построении временной колонки, задании вида подписи.

1.2 Диалог "ВЫРАЖЕНИЕ"

Выбрать запись из таблицы

В окошке составляется выражение. Текст вводится с клавиатуры или используются управляющие элементы окна диалога, описанные ниже. При выборе элемента из какого-либо списка тот автоматически добавляется в окошко на место курсора (рисунок 1).

Колонки

Список содержит имена колонок (то есть имена полей).

Операторы

Список содержит операторы, которые могут участвовать в выражении ("+", "-", "and", "or" и т. п.).

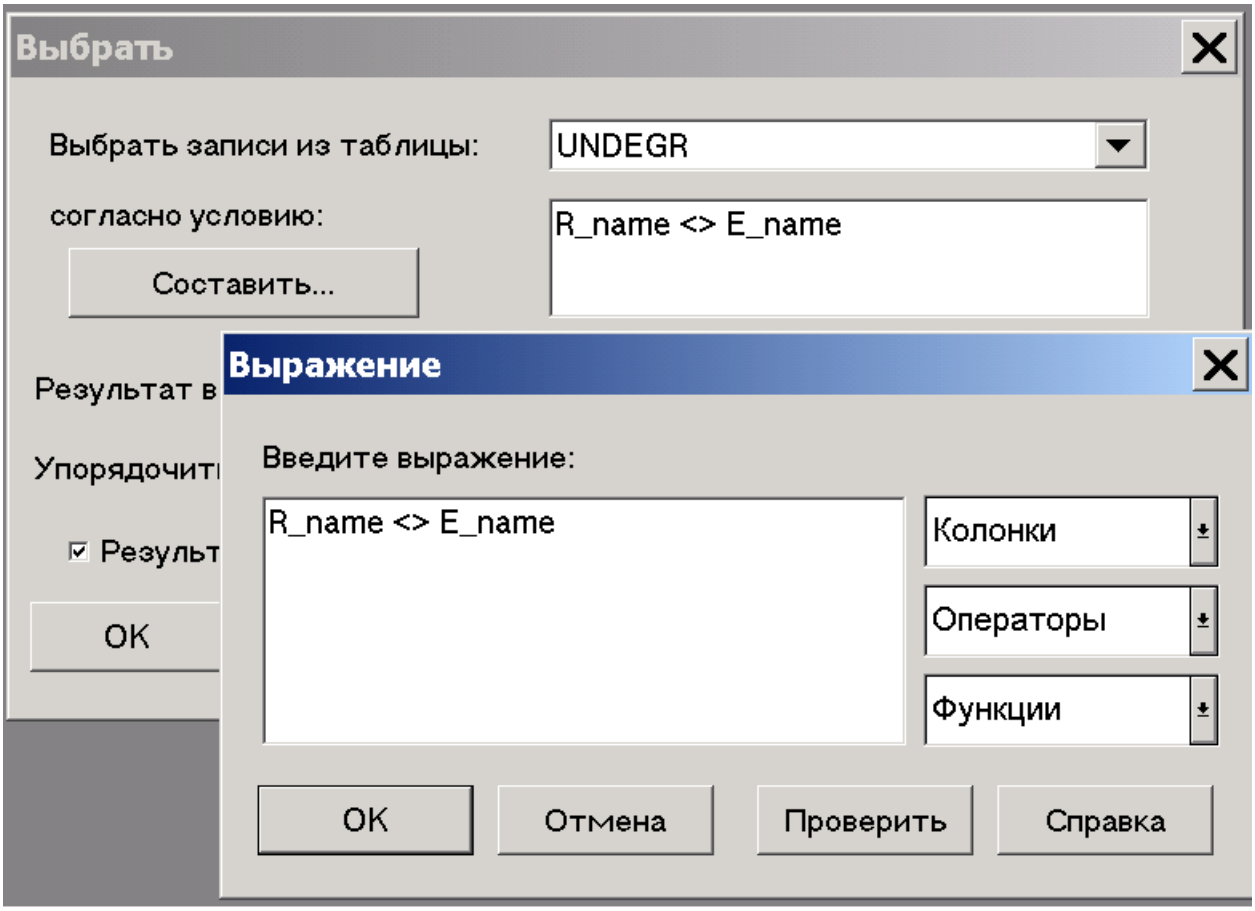


Рисунок 1 – Окно диалога "Выражение"

Функции

Список содержит функции (**Area**, **Sin**, **Year** и т. п.). Функции могут иметь параметры или не иметь их и возвращают значения.

Значения функций могут быть использованы как в выражении, так и в качестве аргумента другой функции.

Проверить

Запускает проверку синтаксиса выражения из окошка.

Отмена

Отменяет составленное выражение и закрывает диалог.

Кнопка "OK"

Составленное выражение переносит с диалог выбора и закрывает диалог.

1.3 Использование констант в выражениях

Для составления выражений во многих диалогах MapInfo используется диалог "Выражение".

Существуют определенные правила и соглашения использования в выражениях строковых констант, чисел и дат.

Символьные строки

Символьные константы в выражении заключаются в кавычки. Если строка не заключена в кавычки, то она понимается как имя поля (колонки). Например, строки 1 и 2 - константы, а строки 3 и 4 - нет.

1. "Фрукты"
2. "Саратов"
3. Фрукты
4. Саратов

Числа

Для задания чисел в выражениях не используйте запятую, знак доллара и другие знаки, не являющиеся цифрами, точкой и знаком минус для отрицательных чисел. Для задания числовых величин в экспоненциальном виде может также использоваться символ E.

Даты

Дата состоит из дня, месяца и года (последнее необязательно). Значения дня, месяца и года пишутся через точку. Год может задаваться двумя или четырьмя цифрами (если двумя, то подразумевается текущее столетие). Если год в дате не дан, то понимается, что эта дата текущего года. Сама дата заключается в

кавычки. Ниже показаны примеры записи даты для 1 февраля 2001 года:

1. "1.2.01"
2. "01.02.2001"
3. "1.2"

1.4 Ключевые слова в выражениях

В MapInfo используются ключевые слова "any", "all", "in" и "between". При задании выражений эти ключевые слова надо набирать с клавиатуры.

Слово "any" обозначает выбор любого из элементов множества.

ABBR = any("AL", "MN", "TX")

"Any" будет истинным для каждой записи, где штат есть Алабама, Минисота или Техас.

Чтобы понять значение "all", рассмотрим пример:

ABBR <> all("AL", "MN", "TX")

Это сообщение означает: выбрать все записи, для которых штат не равен Алабама, Минисота или Техас. Будут выбраны все записи, кроме записей для Alabama, Minnesota или Texas. Сравните с тем, что означает:

ABBR <> any("AL", "MN", "TX")

Следующий пример демонстрирует использование "in":

ABBR in("AL", "MN", "TX")

В данном случае смысл у "in" тот же, что у "=any", а у "not in" - тот же, что у "<>all".

Примеры использования ключевого слова "between":

ЦЕНА between 50000 and 100000

(ЦЕНА between 50000 and 100000) or (ЦЕНА between 150000 and 200000)

1.5 Логические и географические операторы

Логические операторы

"And" (И), "or" (ИЛИ) и "not" (НЕ) - это логические операторы. Они используются при составлении выражений в диалоге команды

Выбрать и в окошке "С условием" диалога команды ***SQL-запрос***. MapInfo использует такие выражения как проверку, которая производится над каждой записью таблицы. Результатом каждой проверки является ответ "да" или "нет" (т.е. "истина" или "ложь"). Комбинируя результаты проверки каждого условия с помощью логических операторов, MapInfo выдает общий ответ: удовлетворяет ли данная запись условию выбора?

- **And** – принимает значение "истина" только в том случае, если оба ее аргумента (логические выражения) истинны. То есть запись должны удовлетворять обоим условиям, чтобы попасть в выборку.
- **Or** – принимает значение "истина", если хотя бы один из аргументов имеет значение "истина". То есть запись должны удовлетворять хотя бы одному из условий, чтобы попасть в выборку.
- **Not** – принимает значение "истина", если аргумент имеет значение "ложь". То есть запись не должны удовлетворять условию, чтобы попасть в выборку.

Географические операторы

В MapInfo имеется несколько географических операторов. Они используются для выбора объектов на основании их взаимного расположения в пространстве. С географическими операторами в MapInfo используется специальное ключевое слово: "obj" или "object". Оно определяет, что MapInfo должно вычислить значение на основании графических объектов, а не соответствующих им в таблице числовых полей.

Имя географического оператора указывается между географическими объектами; выбрать его можно в списке "Операторы" в диалоге "SQL-запрос":

- **Contains** "Содержит". Объект А содержит объект Б, если центроид Б лежит в границах А.
- **Contains Entire** "Полностью содержит". Объект А полностью содержит объект Б, если граница Б полностью лежит внутри границ А.

- **Within** "Внутри". Объект А лежит внутри объекта Б, если его центроид лежит в границах Б.
- **Entirely Within** "Полностью внутри". Объект А лежит полностью внутри объекта Б, если его граница полностью лежит внутри границ Б.
- **Intersects** "Пересекает". Объект А пересекается с объектом Б, если они имеют хотя бы одну общую точку.

1.6 Математические операторы

Математические операторы наиболее часто используются в выражениях. Следующая таблица представляет символы операторов и формулы преобразования типов значений.

Оператор	Синтаксис	Примеры
+ (плюс)	A + B	Дата + Число : Дата любое число + любое число: Вещественное число + Дата: Дата любое целое + любое целое: целое
- (минус)	A – B (вычитание) -A (отрицательное число)	Дата - Число: Дата Число - Дата: Дата любое целое - любое целое: целое любое число - любое число: Вещественное число
* (умножить)	A * B	любое целое * любое целое: целое любое число * любое число: Вещественное число
/ (разделить)	A / B	любое число / любое число: Вещественное число
^ (возвести в степень)	A ^ B	любое число ^ любое число: Вещественное число

Разрешаются следующие виды вычислений:

- Прибавление чисел к датам с получением новой даты;
- Вычитание чисел из дат с получением новой даты;

– Вычитание даты из даты с получением числа.

При прибавлении чисел к датам или вычитании чисел из дат, MapInfo считает числа номером дня в месяце. Так, для вычитания или прибавления недели надо использовать число 7, а для вычитания или прибавления месяца - числа 30 или 31. При вычитании даты из даты результат содержит количество дней.

1.7 Строковые операторы в выражении

В MapInfo возможно соединение строк или строковых выражений при помощи оператора "+" (конкатенация).

Строки должны быть заключены в двойные кавычки. Рассмотрим, например:

"Господин " + Last_Name

При вычислении значения этого выражения MapInfo поставит "Господин " перед каждым значением Last_Name. Строковая константа ("Господин ") взята в двойные кавычки. Аналогично,

"Здравствуй, " + "мир." дает "Здравствуй, мир.".

1.8 Операторы сравнения

В выражениях часто используются следующие операторы сравнения:

- = – равенство;
- <> – неравенство;
- > – более чем;
- < – менее чем;
- >= – больше или равно;
- _ – сравнение (один символ подчеркивания равен одному символу);
- % – сравнение (один символ процента равен нескольким символам).

1.9 Приоритет операторов

Когда MapInfo вычисляет выражение, то некоторые операторы выполняются первыми не зависимо от порядка в выражении. Это

называется приоритетом. Существует несколько уровней приоритета операторов в выражении. Самыми первыми вычисляются операторы наивысшего приоритета, следующими вычисляются операторы более низкого приоритета и т. д. Операторы одного уровня приоритета вычисляются слева на право.

Следующая таблица представляет приоритет операторов в убывающем порядке. Первая строка соответствует самому высокому приоритету и далее ниже.

Наивысший	**Операторы**
↓	Скобки
	Возведение в степень (^)
	Отрицательный знак (-)
	Умножение, деление (* /)
	Сложение, вычитание (+ -)
	Географические операторы, операторы сравнения
	Not (логическое НЕ)
	And (логическое И)
	Or (логическое ИЛИ)
Наинизший	

1.10 Функции

В выражениях могут быть использованы функции. При описании функций будут использоваться следующие обозначения:

num – численное выражение;

str – строковое выражение;

date – выражение с датой;

obj – объектное выражение.

Например, "*Улицы.obj*" представляет объектную колонку таблицы *Улицы*.

Математические функции

- **Abs**(*num*) – Возвращает абсолютное значение числа (модуль).
- **Cos**(*num*) – Возвращает косинус числа num в радианах.
- **Int**(*num*) – Возвращает целую часть числа.

- **Maximum**(*num1*,*num2*) – Возвращает наибольшее из двух чисел.
- **Minimum**(*num1*,*num2*) – Возвращает наименьшее из двух чисел.
- **Round**(*num1*, *num2*) – Возвращает число *num1*, округленное до ближайшего кратного *num2*.
- **Sin**(*num*) – Возвращает синус числа *num* в радианах.
- **Tan**(*num*) – Возвращает тангенс числа *num* в радианах.

Функции даты и времени

- **CurDate**( ) – Возвращает текущую дату.
- **Day**(*date*) – Возвращает день даты (от 1 до 31).
- **Month**(*date*) – Возвращает месяц даты (от 1 до 12).
- **Weekday**(*date*) – Возвращает день недели (от 1 до 7), 1 соответствует воскресенью.
- **Year**(*date*) – Возвращает год-компоненту даты.

Строковые функции

- **Chr$**(*num*) – Возвращает символ, заданный кодом (например, Chr$(65) равно "A").
- **DeformatNumber$**(*str*) – Обладает обратным действием к функции **FormatNumber$**; удаляет разделители тысяч из строки.
- **Format$**(*num*, *str*) – Возвращает строковое представление числа. Например:**Format$**(12345.678, "$,#.##") возвращает "$12,345.68".
- **FormatNumber$**(*num*) – Возвращает строку, представляющую форматированную строку. Эта функция проще, чем **Format$**, но менее гибкая (например, всегда вставляет разделитель тысяч).
- **InStr**(*num*, *str1*, *str2*) – Возвращает позицию первого символа подстроки в строке.
- **LCase$**(*str*) – Возвращает строку, написанную в нижнем регистре.
- **Left$**(*str*, *num*) – Возвращает первые num символов строки *str*.
- **Len**(*str*) – Возвращает число символов строки.
- **LTrim$**(*str*) – Удаляет все пробелы из начала строки.

- **Mid$**(*str*, *num1*, *num2*) – Возвращает *num2* символов из *str*, начиная с символа, номер которого определен параметром *num1*.
- **Proper$**(*str*) – Возвращает строку, написанную в смешанном регистре (первый символ каждого слова заглавный).
- **Right$**(*str*, *num*) – Возвращает последние *num* символов строки *str*.
- **RTrim$**(*str*) – Удаляет все пробелы из конца строки.
- **Str$**(*expr*) – Возвращает строковое представление выражения.
- **UCase$**(*str*) – Возвращает строку, написанную в верхнем регистре.
- **Val**(*str*) – Возвращает число из строки, например, **Val**("18") равно 18.

Географические функции

- **Area**(*obj*, *str*) – Возвращает площадь объекта. Параметр *str* задает единицы измерения, такие как "sq mi" и "sq km".
- **CentroidX**(*obj*) – Возвращает X-координату центроида.
- **CentroidY**(*obj*) – Возвращает Y-координату центроида.
- **Distance**(*num_x*, *num_y*, *num_x2*, *num_y2*, *str*) – Возвращает расстояние между двумя точками, заданными координатами. Параметр *str* задает единицы измерения, такие как "mi" и "km".
- **ObjectLen**(*obj*, *str*) – Возвращает длину объекта. Параметр *str* задает единицы измерения, такие как "mi" и "km". Только объекты типа "дуга, "линия" и "полилиния" имеют ненулевую длину.
- **Perimeter**(*obj*, *str*) – Возвращает периметр объекта. Параметр *str* задает единицы измерения, такие как "mi" или "km". Только объекты типа "эллипс, "область" и "прямоугольник" имеют ненулевую длину.

Функции, возвращающие объекты

- **Buffer**(*obj*, *num_res*, *num_width*, *str*) – Возвращает буферную зону. Параметр *num_res* задает разрешение; *num_width* - радиус буфера; *str* - имя единицы измерения.

- **Centroid**(*obj*) – Возвращает точечный объект на месте центроида объекта *obj*.
- **CreateCircle**(*num_x*, *num_y*, *num_radius*) – Возвращает объект типа "эллипс", окружность. Параметр *num_radius* задает радиус в милях.
- **CreateLine**(*num_x*,*num_y*,*num_x2*,*num_y2*) – Возвращает линию.
- **CreatePoint**(*num_x*, *num_y*) – Возвращает точечный объект.

Каждая из этих функций возвращает географический объект. Если Вы введете команду **Update** в окно MapBasic, то с ее помощью можно создавать объекты для отдельных строк в таблице. Например, если таблица содержит колонки x1, y1, x2 и y2, то следующий оператор создает для каждой записи линию:

Update tablename Set Obj = CreateLine(x1, y1, x2, y2)

2 Порядок выполнения работы

1. Изучить состав и назначение элементов управления в окне диалога "Выражение".

2. Используя в качестве основы электронные карты г. Курска в масштабе 1:10000, составить запросы к данным, применяя в выражениях различные операторы.

3. Провести анализ результатов выборки, используя окна ***Карты*** и ***Списка***.

4. Изучить функции MapInfo, применяя для этого встроенный в ГИС MapInfo язык программирования MapBasic. Для вывода возвращаемых функциями результатов можно воспользоваться следующими операторами:

Note *message* – показывает сообщение в простом диалоговом окне;

Print *message* – печатает пояснительный текст или текст сообщения программы в окне "Сообщение".

Ввод функций, их параметров и операторов вывода осуществляется в окне ***MapBasic***, вызываемом командой меню ***Настройки/Показать окно MapBasic***.

Содержание отчета:

1. Наименование работы.
2. Цель работы.
3. Список функций, встроенных в ГИС MapInfo. Для каждой функции указывается: ее краткое описание; значения параметров при вызове функции; возвращаемый результат.

Контрольные вопросы:

1. Для каких целей в MapInfo применяются выражения?
2. На какие группы делятся выражения?
3. Перечислите правила и соглашения при использовании в выражениях строковых констант, чисел и дат.
4. Какие ключевые слова используются в выражениях?
5. Объясните сущность географических операторов.
6. Что подразумевается под приоритетом операторов и каков их порядок?