МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Авиационно-технологический колледж

УТВЕРЖДАЮ
Директор колледжа
_____ В.А. Зибров

«___»_____2023г

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ДЛЯ ИЗУЧЕНИЯ ДИСЦИПЛИНЫ

ОП.08.ОСНОВЫ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

для студентов специальности

09.02.07 Информационные системы и программирование

Ростов-на-Дону

2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Базы данных играют огромную роль в современном мире, всё, с чем мы ежедневно сталкиваемся в жизни, скорее всего, зарегистрировано в той или иной базе данных. Специалисты в этой области никогда не окажутся безработными.

Герберт Шилдт.

Фундаментальной идеей современных информационных технологий является концепция баз данных : данные должны быть структурированы и адекватно отображать реальные объекты или конкретную предметную область для обеспечения информационных требований пользователя.

Для принятия обоснованных и эффективных решений в производственной деятельности, в управлении экономикой и в политике современный специалист должен уметь с помощью компьютера и средств связи получать, накапливать, хранить и обрабатывать данные, представляя результат в виде наглядных документов. В современном обществе информационные технологии развиваются очень стремительно, они проникают во все сферы человеческой деятельности.

Создание информационной системы - достаточно сложный и многоступенчатый процесс, который, обязательно содержит фазу информационного моделирования. Информационная система должна быть легко сопровождаемой и управляемой.

Главной частью любой информационной модели является база данных, в которой хранится вся необходимая информация.

Большинство баз данных имеют табличную структуру. Как мы знаем, в табличной структуре адрес данных определяется пересечением строк и столбцов. В базах данных столбцы называются полями, а строки – записями. Поля образуют структуру базы данных, а записи составляют информацию, которая в ней содержится. Для того чтобы легко усвоить понятие структуры базы данных, надо представить себе пустую базу, в которой пока еще нет никаких данных. Несмотря на то, что данных в базе нет, информация в ней все-таки есть. Это структура базы, т.е. набор полей. Они определяют, что будет записано в эту базу и в каком виде.

СУБД Access. Системы управления базами данных (СУБД) – это программные средства, с помощью которых можно создавать базы данных, наполнять их и работать с ними. В мире существует немало различных систем управления базами данных. Многие из них на самом деле являются не законченными продуктами, а специализированными языками программирования, с помощью которых каждый, освоивший данный язык, может сам создавать такие структуры, какие ему удобны, и вводить в них необходимые элементы управления. К подобным языкам относятся Clipper, Paradox, FoxPro и др. Необходимость программировать всегда сдерживала широкое внедрение баз данных в малом бизнесе. Крупные предприятия могли позволить себе сделать заказ на программирование специализированной системы «под себя». Малым предприятиям зачастую не по силам было не только решить, но даже и правильно сформулировать эту задачу. Положение изменилось с появлением в составе пакета Microsoft Office системы управления базами данных Access. С помощью Access обычные пользователи получили удобное средство для создания и эксплуатации достаточно мощных баз данных без необходимости что-либо программировать. В то же время работа с Access не исключает возможности программирования. При желании систему можно развивать и настраивать собственными силами. Для этого надо владеть основами программирования на языке Visual Basic. Еще одним дополнительным достоинством Access является интегрированность этой

программы с Excel, Word и другими программами пакета Office . Данные, созданные в разных приложениях, входящих в этот пакет, легко импортируются и экспортируются из одного приложения в другое. Мы будем рассматривать работу СУБД на примере Access XP, которая установлена в учебных лабораториях.

1 Основы разработки и проектирования базы данных

1.1 Основные понятия и определения

Базы данных – это особые структуры данных, причем информация, которая в них содержится, часто имеет общественную ценность. Нередко с одной и той же базой работают тысячи людей по всей стране. От информации, которая содержится в некоторых базах, может зависеть благополучие множества людей. Поэтому целостность содержимого базы не может и не должна зависеть ни от конкретных действий некоего пользователя, ни от перебоев в электросети.

Цель любой информационной системы – обработка данных об объектах реального мира. Основные идеи современной информационной технологии базируются на концепции ***баз данных*** (БД).

База данных (БД) - это поименованная совокупность структурированных данных, относящихся к определенной предметной области.

Согласно данной концепции основой информационной технологии являются ***данные,*** организованные в БД, адекватно отражающие реалии действительности в той или иной предметной области и обеспечивающие пользователя актуальной информацией в соответствующей предметной области.

Под ***предметной областью*** принято понимать часть реального мира, подлежащего изучению для организации управления и в конечном счёте автоматизации, например, предприятие, ВУЗ и т.д.

Первые БД появились уже на заре 1-го поколения ЭВМ 1970-х годах, представляя собой отдельные файлы данных.

Создавая базу данных, пользователь стремится упорядочить информацию по различным признакам и быстро извлекать выборку с произвольным сочетанием признаков. Сделать это возможно, только если данные структурированы.

Структурирование - это введение соглашений о способах представления данных.

Неструктурированными называют данные, записанные, например, в текстовом файле.

Пользователями базы данных могут быть различные прикладные программы, программные комплексы, а также специалисты предметной области, выступающие в роли потребителей или источников данных, называемые конечными пользователями.

Этап разработки и проектирования базы данных является наиболее ответственным и трудоемким, т.к. от оптимальности разработки структуры базы зависит вся дальнейшая работа с ней.

Примеры разрабатываемых баз данных –

Магазин –склад товаров- учет поступления и продажи,

Контроль аптек в департаменте,

Учет студентов, их успеваемость, зачисления, академические отпуска, задолженности,

Аукционы художественных произведений,

Учет и контроль прохождения плав_практик студентами морского колледжа,

Хранение и выдача изданий в библиотеке,

Перевозки грузов сухопутным и морским транспортом,

База данных по студентам для учета успеваемости и посещаемости

Учет и обработка результатов геологических изысканий. И т.д. ……

Банк данных (БнД) — это система баз данных , программных, технических и языковых средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных. Банки данных включают в свой состав следующие компоненты: базы данных, системы управления базами данных, словари данных, вычислительную систему и набор прикладных программ (приложений).

Предметная область — это часть реальной среды, которая описывается в базе данных и отражающая множество объектов среды и связей между ними.

Моделью представления данных называют логическую структуру хранимой в базе данных информации. К основным моделям представления данных (моделям данных) относятся следующие: иерархическая, сетевая, реляционная, постреляционная

Систе́ма управле́ния ба́зами да́нных (СУБД) — специализированная программа (чаще комплекс программ), предназначенная для организации и ведения базы данных.

Основные функции СУБД:

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

В отличие от файловых *систем* СУБД сохраняют не только сами данные, но и **структуры этих данных, и связи между ними, а также способы доступа к данным**.

Каждая база данных и СУБД строится на основе некоторой явной или неявной модели данных. Все СУБД, построенные на одной и той же модели данных, относят к одному типу.

Приложение представляет собой программу или комплекс программ, обеспечивающих автоматизацию обработки информации и удобный и дружественный интерфейс для пользователя.

При рассмотрении приложений, работающих с одной базой данных, предполагается, что они могут работать параллельно и независимо друг от друга, и именно СУБД призвана обеспечить работу множества приложений с единой базой данных таким образом, чтобы каждое из них выполнялось корректно, но учитывало все изменения в базе данных, вносимые другими приложениями.

Приложения могут создаваться с помощью системы программирования, использующей средства доступа к базам данных, к примеру Delphi или Си++ Builder или непосредственно в среде СУБД.

Словарь данных (СД) представляет собой подсистему БнД, предназначенную для централизованного хранения информации о структурах данных, взаимосвязях файлов БД друг с другом, типах данных и форматах их представления, принадлежности данных пользователям, кодах защиты и разграничения доступа.

Администратор базы данных**-** лицо или группа лиц, отвечающих за выработку требований к БД, ее проектирование, создание, эффективное использование и сопровождение. В процессе эксплуатации АДмин обычно следит за функционированием программных средств сети, управляет конфигурацией сети, обеспечивает восстановление программного обеспечения после сбоев и отказов оборудования, профилактические мероприятия и разграничения доступа пользователей к информации в БД .

1.2 Требования к базам данных

Итак, хорошо спроектированная база данных:

- Удовлетворяет всем требованиям пользователей к содержимому базы данных. Перед проектированием базы необходимо провести обширные исследования требований пользователей к функционированию базы данных.
- Гарантирует непротиворечивость и целостность данных. При проектировании таблиц нужно определить их атрибуты и некоторые правила, ограничивающие возможность ввода пользователем неверных значений.
- Обеспечивает удобное построение базы, что позволяет делать запросы из базы данных легкими для понимания;
- Обеспечивает защиту данных от несанкционированного доступа и повреждений.

Известно три основных вида моделей данных : ***иерархическая, сетевая и реляционная***;
Каждая из указанных моделей обладает характеристиками, делающими ее наиболее удобной для конкретных приложений.

2 Модели данных для концептуального уровня

Ядром любой базы данных является модель данных. Модель данных представляет собой множество структур данных, ограничений целостности и операций манипулирования данными. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

По способу установления связей между данными СУБД основывается на использовании трёх основных видов модели: ***иерархической, сетевой или реляционной***; на комбинации этих моделей или на некотором их подмножестве.

Однако, различия между этими моделями постепенно стираются, что обусловлено прежде всего интенсивными работами в области баз знаний (БЗ) и объектно-ориентированной инфотехнологией. Каждая из указанных моделей обладает характеристиками, делающими ее наиболее удобной для конкретных приложений. Одно из основных различий этих моделей состоит в том, что для иерархических и сетевых СУБД их структура часто не может быть изменена после ввода данных, тогда как для реляционных СУБД структура может изменяться в любое время. С другой стороны, для больших баз данных, структура которых остается длительное время неизменной, и постоянно работающих с ними приложений с интенсивными потоками запросов на БД-обслуживание именно иерархические и сетевые СУБД могут оказаться наиболее эффективными решениями, ибо они могут обеспечивать более быстрый доступ к информации БД, чем реляционные СУБД.

2.1 ИЕРАРХИЧЕСКАЯ МОДЕЛЬ ДАННЫХ

Иерархическая структура представляет совокупность элементов, связанных между собой по определенным правилам. Объекты, связанные иерархическими отношениями, образуют ориентированный граф (перевернутое дерево). Взаимосвязи между объектами отражаются по принципу иерархии типов объекта в виде связанного графа, вершины которого размещены на разных иерархических уровнях. Самая высокая вершина называется **корнем** (главный объект), а остальные, находящиеся на нижних уровнях иерархии. — подчиненными.

К основным понятиям иерархической структуры относятся: уровень, элемент (узел), связь.

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

дерево имеет только одну вершину (корень дерева), не подчиненную никакой другой вершине и находящуюся на самом верхнем (первом) уровне. Зависимые (подчиненные) узлы находятся на втором, третьем и т.д. уровнях.

Взаимосвязи между объектами отражаются по принципу иерархии типов объекта в виде связанного графа, вершины которого размещены на разных иерархических уровнях. Самая высокая вершина называется **корнем** (главный объект),
а остальные, находящиеся на нижних уровнях иерархии. — подчиненными. Корень (первый уровень) не подчиняется ни одной вершине. Все остальные вершины - объекты связаны с одной и только одной вершиной, которая размещена на более высоком уровне. Взаимосвязь между объектами отражена в след. Схеме.

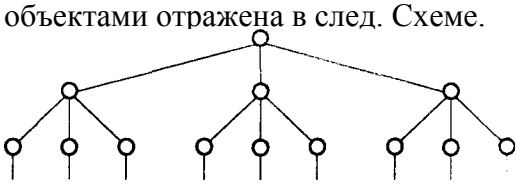


Рис. 2.3. Схема иерархической модели данных

Свойства иерархической модели данных:

- Несколько узлов низшего уровня связано только с одним узлом высшего уровня.
- Иерархическое дерево имеет только одну вершину (корень), не подчиненную никакой другой вершине.
- Каждый узел имеет свое имя (идентификатор).
- Существует только один путь от корневой записи к более частной записи данных.

Взаимосвязь между главными и подчиненными объектами устанавливается типом «один-ко-многим» (1 : M). Как видно из схемы иерархической модели данных, ее древовидная структура состоит из узлов .

Например, объект «директор» может иметь такие атрибуты: зам1, зам2, зам3, которые имеют в подчинении несколько других объектов.

Вершины графа, которые подчинены другой вершине ("имеют отца»), называются сыновьями. Любая вершина может иметь множество подчиненных ей вершин на более низком уровне. Каждая пара вершин соединена одной простой дугой.

В такой модели данных существует сильная зависимость между описанием структуры данных и способом их записи на внешние носители (диски). Для работы с такими базами данных разработаны СУБД- DataBase Focus, Data Edge и ИНЭС.

Каждому узлу структуры соответствует один сегмент**,** представляющий собой поименованный линейный кортеж полей данных**.** Каждому сегменту (кроме S1-корневого*)* соответствует один входной и несколько выходных сегментов. Каждый сегмент структуры лежит на единственном иерархическом пути, начинающемся от корневого сегмента.

К основным недостаткам иерархических моделей следует отнести: медленный доступ к сегментам данных нижних уровней иерархии, четкая ориентация на определенные типы запросов и др. В связи с этими недостатками ранее созданные иерархические СУБД подвергаются существенным модификациям, позволяющим поддерживать более сложные типы структур и, в первую очередь, ***сетевые*** и их модификации.

2.2 СЕТЕВАЯ МОДЕЛЬ ДАННЫХ

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

В сетевой модели понятие главного и подчиненных объектов иное, чем в иерархической модели: любой объект здесь может быть и главным и подчиненным, иметь любое количество связей с другими объектами.

Эта модель обычно используется при построении схемы процесса, например строительство или обучение. СУБД – DB Vista и Компас.

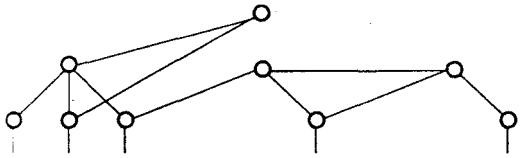


Рис. 2.4. Общая схема сетевой модели данных. ПРИМЕР!!!

Графическое изображение структуры связей сегментов такого типа моделей представляет собой сеть. Сегменты данных в сетевых базах данных могут иметь множественные связи с сегментами старшего уровня.

. В сетевой модели понятие главного и подчиненных объектов иное, чем в иерархической модели: любой объект здесь может быть и главным и подчиненным, иметь любое количество связей с другими объектами.

Эта модель обычно используется при построении схемы процесса, например - строительство или обучение. СУБД – DB Vista и Компас.

Таким образом, под сетевой СУБД понимается система, поддерживающая сетевую организацию: Возможно обращение ко всем записям в наборе, начиная с записи старшего уровня. Обращение к набору записей реализуется по указателям.

Сетевые СУБД поддерживают сложные соотношения между типами данных, что делает их пригодными во многих различных приложениях. Однако пользователи таких СУБД ограничены связями, определенными для них разработчиками БД-приложений.

Среди недостатков сетевых СУБД следует особо выделить проблему обеспечения сохранности информации в БД, решению которой уделяется повышенное внимание при проектировании сетевых БД.

2.3 РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Наиболее распространенная и часто используемая структура базы данных-реляционная, от Relation- таблица. Понятие реляционный (англ. relation — таблица) связано с разработками известного американского специалиста в области систем баз данных, сотрудника фирмы IBM д-ра Е. Кодда (Codd E.F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970), которым впервые был применен термин "реляционная модель данных".

В простейшем случае она представляет собой двухмерный массив или двухмерную таблицу. Каждая строка такой таблицы называется записью, а столбец – полем.

Табличная структура данных отражает отношения между реальными объектами и их характеристиками. Поиск и обработка записей не зависят от организации хранения данных в памяти компьютера. При этом эффективно используются математическая логика и алгебра.

Одно из важнейших достоинств реляционных баз данных состоит в том, что можно хранить логически сгруппированные данные в разных таблицах и задавать связи между ними, объединяя их в единую базу. Для задания связи таблицы должны иметь поля с одинаковыми именами или хотя бы с одинаковыми форматами данных. Связь между таблицами устанавливает отношения между совпадающими значениями в этих полях. Такая организация

данных позволяет уменьшить избыточность хранимых данных, упрощает их ввод и организацию запросов и отчетов.

Термин «реляционный» указывает прежде всего на то, что такая модель хранения данных построена на взаимоотношении составляющих ее частей.

Обычно - это двухмерная таблица, а при создании сложных информационных моделей составит совокупность взаимосвязанных таблиц. Каждая строка такой таблицы называется записью, а столбец – полем. Табличная структура данных отражает отношения между реальными объектами и их характеристиками. Поиск и обработка записей не зависят от организации хранения данных в памяти компьютера. При этом эффективно используются математическая логика и алгебра.

Основной принцип реляционных структур баз данных – это:

- получение из первичных таблиц БД необходимых данных, выбранных по заданным критериям,
- формирование новых таблиц соответствующей структуры на основе первичных таблиц при помощи логических операций.

Поскольку в локальных базах данных каждая таблица размещается в отдельном файле, то с точки зрения размещения данных для локальных баз данных отношение можно отождествлять с файлом. Кортеж представляет собой строку в таблице, или, что то же самое, запись. Атрибут же является столбцом таблицы, или - полем в записи. Домен же представляется неким обобщенным типом, который может быть источником для типов полей в записи.
Таким образом, следующие тройки терминов являются эквивалентными:

· отношение, таблица, файл (для локальных баз данных);
· кортеж, строка, запись ;
· атрибут, столбец, поле.

В реляционных моделях имеются следующие типы объектов: таблицы (отношения), атрибуты (столбцы), кортежи (строки) и домены (допустимые значения атрибутов). В этой модели объекты и взаимосвязи между ними представлены при помощи таблиц . Одна таблица представляет один объект и состоит из столбцов и строк. Каждая строка таблицы представляет собой одну запись, а каждый столбец — одно поле записи.

Таблица (relation)базы данных обладает следующими свойствами:

- все поля в таблице являются однородными, т.е. имеют одинаковый тип;
- каждое поле имеет уникальное имя;
- одинаковые записи в таблице отсутствуют;
- порядок записей в таблице может быть произвольным и может характеризоваться количеством полей, типом данных;
- при выполнении операций с таблицей ее строки и столбцы можно обрабатывать в любом порядке, независимо от их информационного содержания.

Классическая реляционная модель предполагает неделимость данных, хранящихся в полях записи таблиц. Это означает, что информация в таблице представляется по жестким правилам и требованиям. Существует ряд случаев, когда это ограничение мешает эффективной реализации приложений.

Постреляционная модель данных представляет собой расширенную реляционную модель, снимающую ограничения неделимости данных, хранящихся в записях таблицы. Постреляционная модель данных допускает многозначные поля – поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу. Постреляционная модель данных поддерживает также и ассоциированные многозначные поля (множественные группы данных).

Таким образом, в постреляционных моделях данных не накладывается ограничение на длину полей и количество полей в таблице, что означает, несомненно большую гибкость таких структур.

Работа с БД включает в себя такие действия, как:

- Ввод и хранение данных,
- Поиск необходимых сведений;
- Сортировка данных;
- Отбор данных;
- Вывод на печать;
- Изменение и дополнение данных.

Цикл взаимодействия пользователя с БД с помощью приложения можно разделить на следующие основные этапы:

1. Пользователь в процессе диалога с приложением формулирует запрос на некоторые данные из БД.

2. Приложение на программном уровне средствами языка манипулирования данными (SQL) формулирует запрос , с которым обращается к СУБД.
3.Используя свои системные управляющие блоки и таблицы, СУБД с помощью *словаря данных* определяет местоположение требуемых данных и обращается за ними к ОС.
4.Программы методов доступа файловой системы ОС считывают из внешней памяти искомые данные и помещают их в системные буферы СУБД.
5.Преобразуя полученные данные к требуемому формату, СУБД пересылает их в соответствующую область программы и сигнализирует о завершении операции каким-либо образом (например, кодом возврата).
6.Результаты выбора данных из базы приложение отображаются на компьютере пользователя .

СХЕМА!

Предложив реляционную модель данных, Э.Ф.Кодд создал и инструмент для удобной работы с отношениями – реляционную алгебру. Каждая операция этой алгебры использует одну или несколько таблиц в качестве ее операндов и продуцирует в результате новую таблицу, т.е. позволяет "разрезать" или "склеивать" таблицы.

3 Основные этапы проектирования базы данных.

Следующие пункты представляют основные шаги проектирования базы данных:

1. Определить информационные потребности базы данных.
2. Проанализировать объекты реального мира, которые необходимо смоделировать в базе данных. Сформировать из этих объектов сущности и характеристики этих сущностей (например, для сущности “деталь” характеристиками могут быть “название”, “цвет”, “вес” и т.п.) и сформировать их список.
4. Определить атрибуты, которые уникальным образом идентифицируют каждый объект.
5. Выработать правила, которые будут устанавливать и поддерживать целостность данных.ё
6. Установить связи между объектами (таблицами и столбцами), провести нормализацию таблиц.
7. Спланировать вопросы надежности данных и, при необходимости, сохранения секретности информации.

1 –ый этап проектирования БД – это разработка концептуальной модели. Этот этап является самым ответственным и важным, т.к. от правильности разработки таблиц напрямую зависит работы будущей БД.
Основные действия 1-го этапа:
1. Определить информационные потребности базы данных и пользователей.
2 Проанализировать объекты реального мира, которые необходимо смоделировать в базе данных. Сформировать из них объекты БД и характеристики этих объектов,
3. Поставить в соответствие объектам и характеристикам - таблицы и столбцы (поля) в выбранной СУБД (Paradox, dBase, FoxPro, Access, InterBase, Sybase, Informix, Oracle)
4.Определить поля, которые уникальным образом идентифицируют каждый объект.
5.Установить связи между объектами (таблицами и столбцами),
6.Спланировать вопросы надежности данных и, при необходимости, сохранения секретности информации.
7. Сформировать критерии и условия выборок данных из таблиц БД.
Итак, Первый шаг состоит в определении информационных потребностей базы данных. Он включает в себя опрос будущих пользователей для того, чтобы понять и структурировать их требования.
Следует выяснить следующие вопросы:

- кто будет вводить данные в базу и в какой форме; как часто будут изменяться данные;
- достаточно ли будет для Вашей предметной области одной базы или Вам потребуется несколько баз данных с различными структурами;

***2*этап-** построение Логической модели данных. Она отражает логические связи между элементами данных независимо от их содержания и места хранения.
3 этап –определение физической структуры, которая определяет размещение данных, методы доступа и технику индексирования. Физическую модель часто называют внутренней моделью системы. Физическая структура данных проектируется с некоторым избытком для обеспечения надежности управления данными.

Примеры построения концептуальной модели БД:

Построим концептуальную модель предметной области.
Объекты предметной области.
1 - **«Колледж»: объекты** …Студенты- КОД_СТ, ФИО, пол, возраст, адрес, телефон, № паспорта, отделение, группа, стипендия. Преподаватели,
Успеваемость+ Посещаемость- КОД_СТ, №_семестра, предмет, преподаватель, оценка
Справочник – отделения!
ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ: База данных предназначена для хранения данных об обучающихся студентах: ФИО, пол, возраст, адрес, телефон, № паспорта, отделение, группа. Кроме этих личных данных – успеваемость, а также пропуски . Подразумевается, что эта информация может изменятся в течении всего периода обучения и может быть затребована в любое время за период обучения студента или участвовать в формировании статистических данных о группе или курсе за любой временной промежуток. **Пользователями** базы данных являются работники деканатов, которые не имеют пользовательских навыков. Это накладывает определенные требования на разработку программного интерфейса.

4 КЛЮЧИ И ИНДЕКСЫ

4.1 Типы связей между таблицами.

Предметную область в реляционных моделях данных представляют в виде соответствующего числа таблиц. Классическое понятие «файл» в обработке данных можно ассоциировать с одной таблицей. Таблицы находятся между собой в соответствующих отношениях (связях).

Между информационными объектами существуют четыре типа ***отношений,*** для которых введены следующие обозначения:

«один-к-одному» (1 : 1); «один-ко-многим» (1 : М); «многие-к-одному» (М:1); «много-ко-многим»(М : М).

Рассмотрим эти типы отношений между парами связанных элементов данных.

Отношение (отражение) «один-к-одному» (1: 1*)* означает, что одной записи в одной таблице А соответствует не более чем одна запись в таблице **В, к** которой направлена связь, и наоборот. Связь поддерживается при помощи совпадающих полей, которые могут иметь разные имена. Связь между таблицами устанавливается не по совпадению имен полей, а на основе значений совпадающих полей. Связь уникальная как слева направо, так и справа налево.

Пример.

ФИО студента - № зачетки ФИО старосты - группа

1 : М

Отношение «один-ко-многим» означает,что однпа запись идентифицирует несколько записей другой. Совпадающие поля имеют одинаковые имена.

Пример. Номер группы - Фамилия студента;

Отношение «много-к-одному» (М : I) схоже с отношением «один-ко-многим". но является уникальной лишь связь слева направо.

Пример М: 1

Фамилия студента Номер группы

Отношение «многие -ко-многим» (М : М) возникает между двумя таблицами, когда:

а) одна запись с первой таблицы А (выходная связь) может быть связана больше чем с одной записью другой таблицы В (принимающая);

б) одна запись с другой таблицы может быть связана больше чем с одной записью первой таблицы. Идентификация не является уникальной в обоих направлениях.

Пример.

м:м

Преподаватели Студенты

Один преподаватель обучает множество студентов, один студент учится у многих преподавателей.

4.2 Реляционные ключи

В управлении базами данных КЛЮЧ - это идентификатор записей в файле данных, который уникально определяет запись об объекте. Чтобы связать две реляционные таблицы, необходимо ключ первой таблицы ввести в состав второй таблицы (возможно совпадение ключей). Например, в таблице «Студенты» КОД_СТ и есть **первичный ключ**, по которому связывается эта таблица и таблица «Успеваемость». А ключ во второй таблице называется ***внешним ключом, Т.е. на него приходит связь!***.

Первичный ключ (primary key)однозначно идентифицирует только одну запись, а вторичный или внешний (foreiqn key) – означает значение ключевого поля, которому соответствует несколько записей с одинаковыми значениями полей.

Информация о ключе хранится в отдельном файле с **расширением .РХ** и значения ключа располагаются в определенном порядке. Для каждого значения ключа имеется уникальная ссылка , указывающая на расположение записи в таблицах. Поэтому при поиске записи выполняется не последовательный просмотр всей таблицы, а прямой доступ к записи на основании упорядоченных значений ключа.

4.3 Индексы

Индекс базы данных предназначен для того, чтобы ускорить **поиск информации в**

таблице базы данных. Индекс строится по полям таблицы, он может допускать повторение составляющих его полей – в этом основное их отличие от ключа. Индекс предоставляет информацию о точном физическом расположении данных в таблице. При создании индекса в нем сохраняется информация о местонахождении записей, относящихся к индексируемому столбцу таблицы. Индексный файл содержит указания на соответствующие записи таблицы. При добавлении в таблицу новых записей или удалении существующих индекс также модифицируется.При выполнении запроса к базе данных, в условие поиска которого входит индексированный столбец, поиск значений производится в первую очередь в индексе. Если этот поиск оказывается успешным, то в индексе устанавливается точное месторасположение искомых данных в таблице базы данных. Ключевые поля обычно автоматически индексируются.**Использование индексов обеспечивает- сортировку записей по индексным полям**, что увеличивает скорость доступа и поиска данных. Для одной таблицы можно создать несколько индексов, по которым будет выполняться у порядочивание записей таблицы

Например, в таблице **Студенты** проиндексируем поле **Фио:**

1_Яковлев, 2_Абрамов, 3_Кожин, 4_Янин, 5_Волгин, 6_Алов. Индексный файл будет иметь след. **Структуру: 2,6,5,3,1,4. А можно проиндексировать по полю ГРУППА**

При необходимости вывода или поиска нужной ФИО система просматривает индексный файл и в соответствии с ним выводит данные столбца. Структура индексного файла - при начальном заполнении в каждом блоке остается свободная область - процент расширения .При добавлении новой записи в таблицу она записывается в конец основной области. В индексной же области выполняется занесение в конкретное место, не нарушая упорядоченности.

Построим концептуальную модель предметной области !!!
Магазин»-поставки товаров, продажи …
«Строительная фирма»- Сотрудники, Клиенты, Объекты (…дата начала, окончания, оплата, качество), Виды работ.
Склад –Магазин – по продаже зап.частей.
«Аукционы»: Аукцион Предметы Покупатели Продавцы Сотрудники

4.4 Целостность данных

Целостность данных означает систему правил, используемых в СУБД для поддержания связей между записями в связанных таблицах, а также обеспечивающих защиту от случайного удаления или изменения связанных данных. Установить целостность данных можно, если выполнены следующие условия:
-Связанное поле главной таблицы является ключевым полем или имеет уникальный индекс.
-Связанные поля имеют один тип данных.
-Обе таблицы принадлежат одной базе данных. Если таблицы являются связанными, то они должны быть таблицами одного формата. Для таблиц, в которых проверяется целостность данных, пользователь имеет возможность указать, следует ли автоматически выполнять для связанных записей операции **каскадного обновления и каскадного удаления – т.е. при удалении записи из главной таблицы удаляются все связанные с ней записи в других таблицах.** Удалим студента – и удаляются все его стоки в таблице Успеваемость**.** Чтобы обеспечить целостность данных при изменении значения первичного ключа в главной таблице, автоматически вносятся необходимые изменения в связанные таблицы

5 Транзакции

Транзакцией называется последовательность операции, производимых над базой данных и переводящих базу данных из одного непротиворечивого -
состояния в другое непротиворечивое состояние.

Транзакция рассматривается как некоторое неделимое действие над базой дан-ных, осмысленное с точки зрения пользователя. В то же время это логическая единица работы системы. Именно разработчик определяет, какая последовательность операций составляет единое целое, то есть транзакцию. Разработчик приложений определяет это исходя из смысла обработки данных, он моделирует некоторую неразрывную цепочку действий и составляет транзакцию.

Возможны два варианта завершения транзакций. Если все операторы выполнены успешно и не было сбоев, тогда транзакция фиксируется. Это означает, что все изменения в базе данных , выполненные в рамках транзакции, записываются на диск и становятся видимыми другим транзакциям, работающих с уже новой БД. Пользователь с момента фиксации транзакции тоже видит новые данные. Все изменения, выполненные в транзакции, сохраняются.

Если в процессе выполнения Транзакции случился сбой, ТО происходит откат транзакции , который аннулирует все незавершенные изменения в БД и возвращает БД в исходное состояние. Группирование операторов в транзакцию в языке SQL сообщает системе, что эта группа должна быть выполнена как единое целое или не выполнена совсем. Реализация принципа сохранения всех промежуточных состояний БД в процессе выполнений транзакций, обеспечивается записями в журнале транзакций, в котором указывается номер, команда начала и окончания транзакции. Для большей надежности эти журналы дублируются системой во внешней памяти.

6 Разработка базы данных в СУБД MS Access

Пример разработки базы данных в СУБД MS Access

Прежде всего для успешной разработки базы данных необходимо изучить предметную область, т.е. ту сферу деятельности, аспекты которой необходимо автоматизировать и построить базу данных для удобства пользователя.

Разрабатывается концептуальная схема будущей базы данных – объекты и таблицы.

Для каждой таблицы выбирается ключевое поле, которое будет использоваться для связи между таблицами. На схеме обязательно устанавливаются эти связи.

Постановка задачи

Разработать реляционную базу данных для автоматизации учета и контроля работ на объектах фирмы «ИП Ткаченко», для получения оперативной и объективной информации по основным показателям деятельности «ИП Ткаченко С.Ю.».

Создаваемый программный комплекс должен обеспечивать выполнение следующих функций:

ввод, хранение и обработку информации по персоналу, их личных и профессиональных данных;

ввод, хранение и обработку данных по клиентам и их заказам;

выборки данных и обработку результатов деятельности фирмы;

вывод ведомости работ, выполненных указанным работником за текущий месяц с их сметной стоимостью;

вывод списка выполненных работ за указанный месяц или год всеми работниками (сгруппировать по каждому работнику);

вывод работы, не оплаченные полностью;

вывод суммы ремонтов по фирме, за введенный месяц, год, их количество;

формировать бланк - счет клиенту за выполненные работы;

вывод результатов анализа работы в виде отчетов и диаграмм.

Разработать удобный и дружественный интерфейс для работы с программой пользователей любого уровня подготовки. Предусмотреть защиту программного средства паролем.

База данных

Для обеспечения функционирования информационной системы создана база данных «ИП Ткаченко С.Ю.», состоящая из 7 таблиц. Перечень таблиц приведён на рисунке 1.

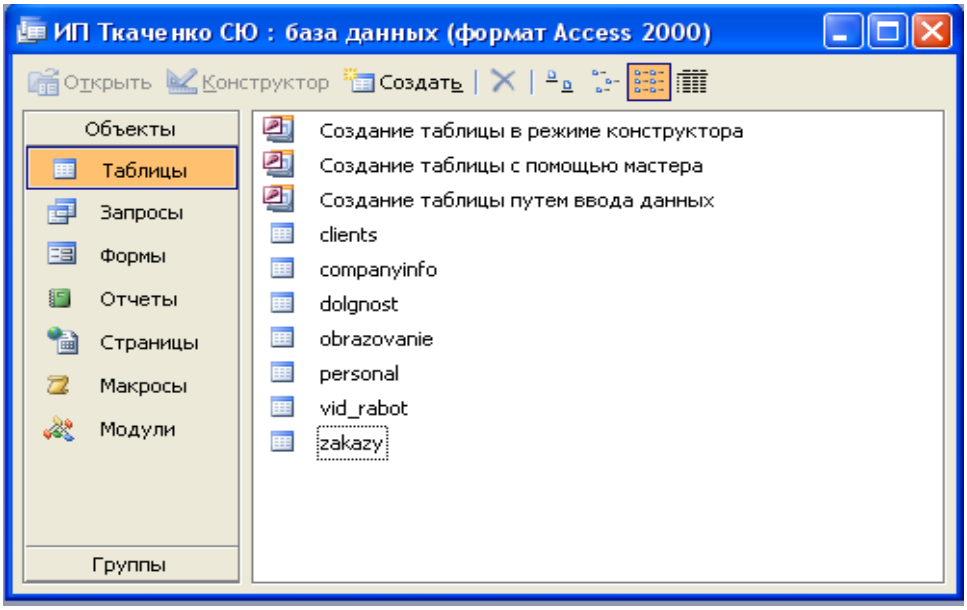


Рисунок 1 – Главное окно базы данных.

Структура каждой таблицы разрабатывалась в режиме конструктора.

Таблица 2 - «Клиенты»

Таблица 3 – «Заказы». Это главная таблица и из нее выполняются все выборки данных о работе и каждого работника, и всей фирмы в целом.

zakazy : таблица

Код заказа	Дата заказа	Клиент	Вид работ	Работник	Оплата (полностью)	Стоимость
1	13.04.2009	Иванов	Прочистка	Рогожин	☑	600,00р.
2	05.05.2009	Пономарева	Прочистка	Сучков	☑	400,00р.
3	05.06.2009	Петренко	Тестирование ОС	Сучков	☐	1 300,00р.
4	02.05.2009	Петренко	Замена элементов	Труев	☐	1 600,00р.
5	12.05.2009	Васильев	Тестирование ОС	Сучков	☑	1 300,00р.
6	15.05.2009	Иванов	Замена элементов	Труев	☐	1 600,00р.
7	12.06.2009	Нагорный	Прошивка BIOS	Сучков	☑	800,00р.
8	12.06.2009	Петренко	Пропайка	Сучков	☑	650,00р.
9	02.06.2009	Борисова	Тестирование ОС	Труев	☐	1 300,00р.
10	07.06.2009	Кузьменко	Замена элементов	Василенко	☐	1 600,00р.
11	09.06.2009	Шевченко	Прошивка BIOS	Рогожин	☑	800,00р.
12	09.06.2009	Васильев	Тестирование ОС	Василенко	☑	1 300,00р.
13	17.05.2009	Нагорный	Замена элементов	Шукин	☐	1 600,00р.
(Счетчик)					☐	0,00р.

Запись: 1 из 13

Далее представлены таблицы-справочники.

Таблица 4 – «Должности»

Таблица 5 – «Виды выполняемых работ»

Ниже приведена схема данных – таблиц и логических связей между ними.

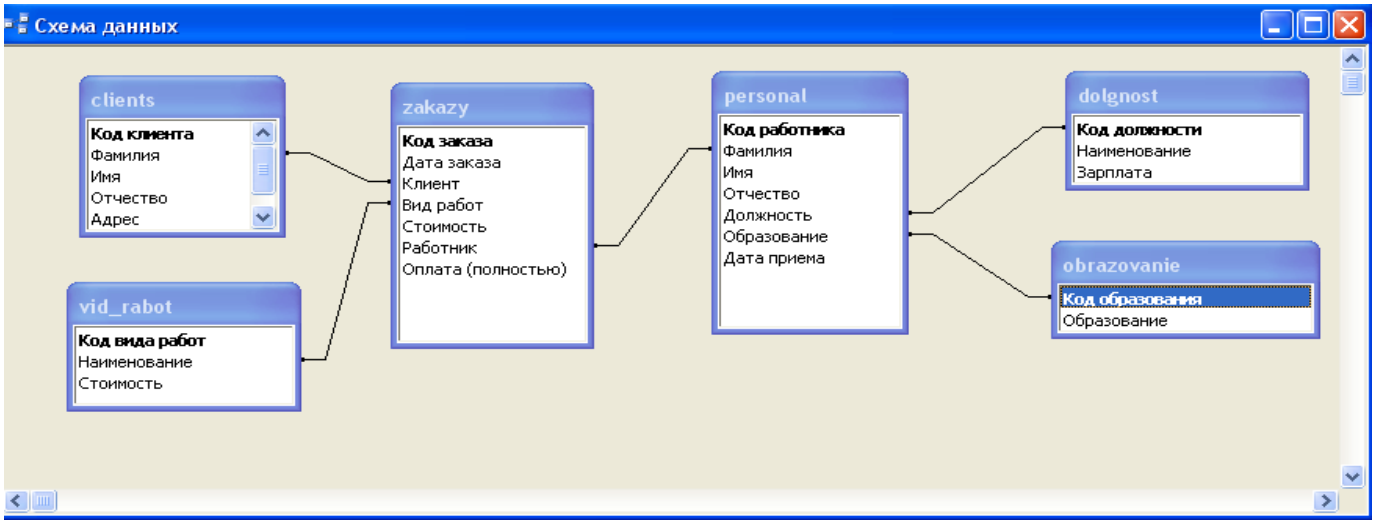


Рисунок 2 – Схема данных

Выходные данные- выборки из таблиц БД

База данных формирует следующие выходные документы:

- личные и профессиональные данные персонала фирмы по запросу;
- выбор работ, выполненных конкретным работником за отчетный период с подсчетом общих сумм;
- выбор данных по долгам клиентов, которые не полностью оплатили выполненные работы;
- диаграмма для анализа данных.

В программе предусмотрен режим не только просмотра всех выходных документов, но и печать их по желанию пользователя. Примеры выходных документов в виде отчетов представлены ниже. Выборки из таблиц выполнялись через запросы.

Запрос 1. Выбор информации о работнике. По введенной фамилии в диалоговое окно выводится вся необходимая информация о работнике.

Конструктор запроса:

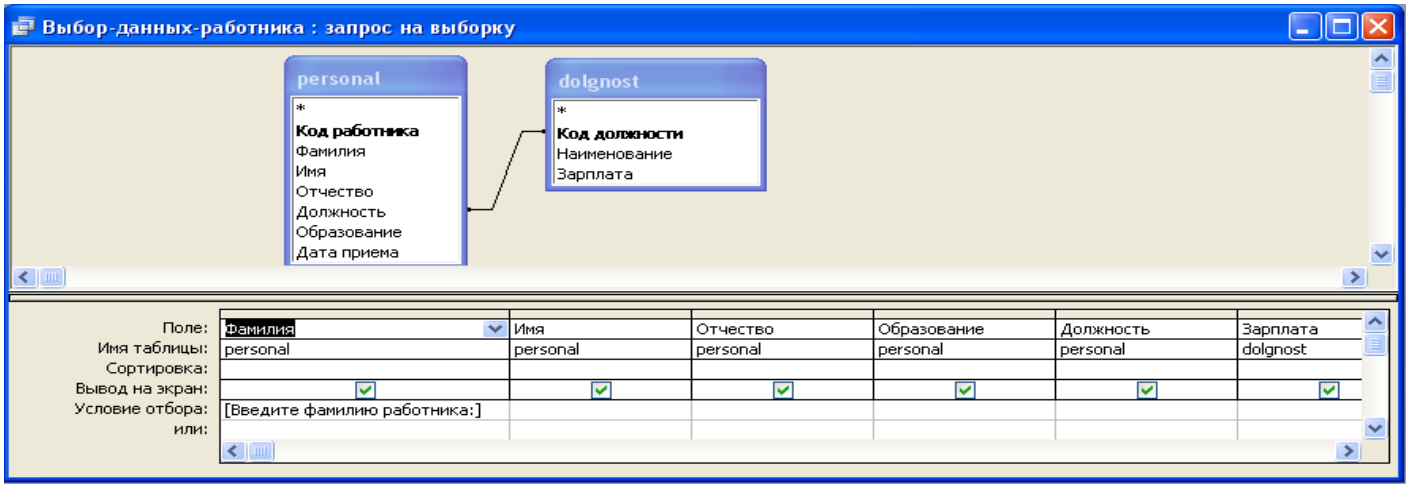


Рисунок 3 – конструктор запроса

При выполнении запроса выводится диалоговое окно для ввода данных

После этого на экран выводится следующая таблица

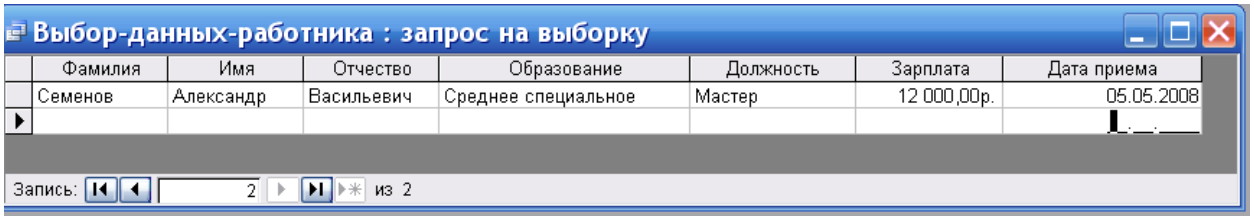


Рисунок 4 – Вывод данных по требованию пользователя

Запрос 2. Выбор работ за месяц. Конструктор запроса:

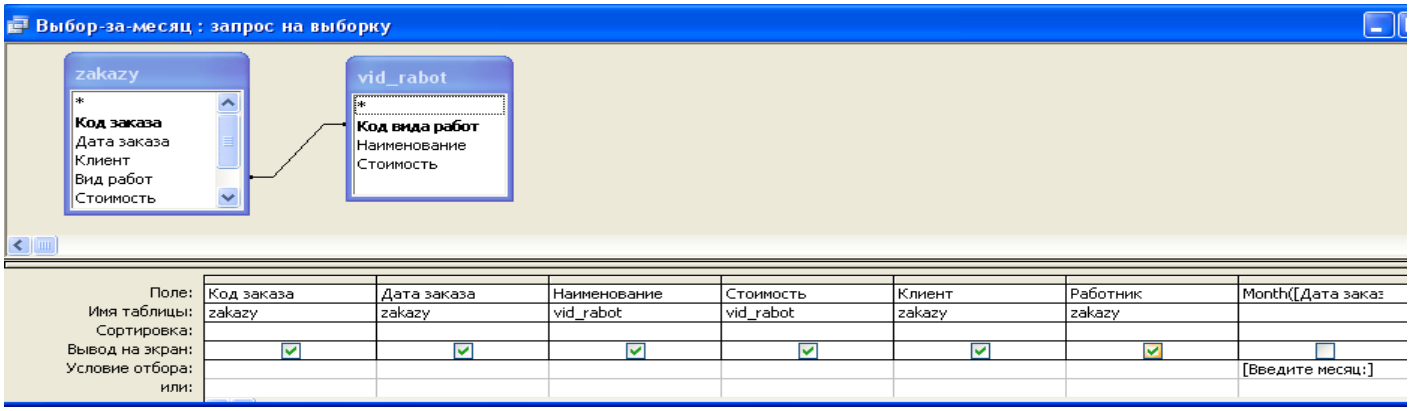
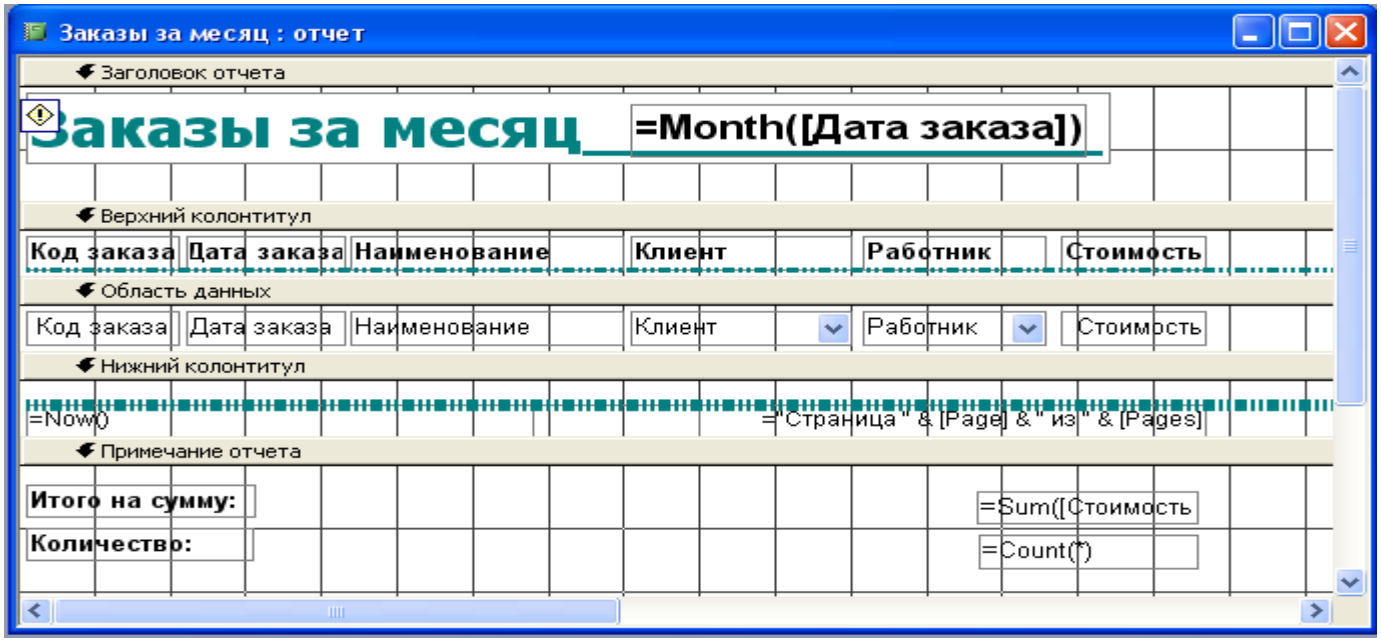


Рисунок 5 – таблица- результат запроса

В отчете, который построен по этому запросу, будут подведены итоги по суммам

Конструктор отчета с функциями подведения итогов



Результат отчета

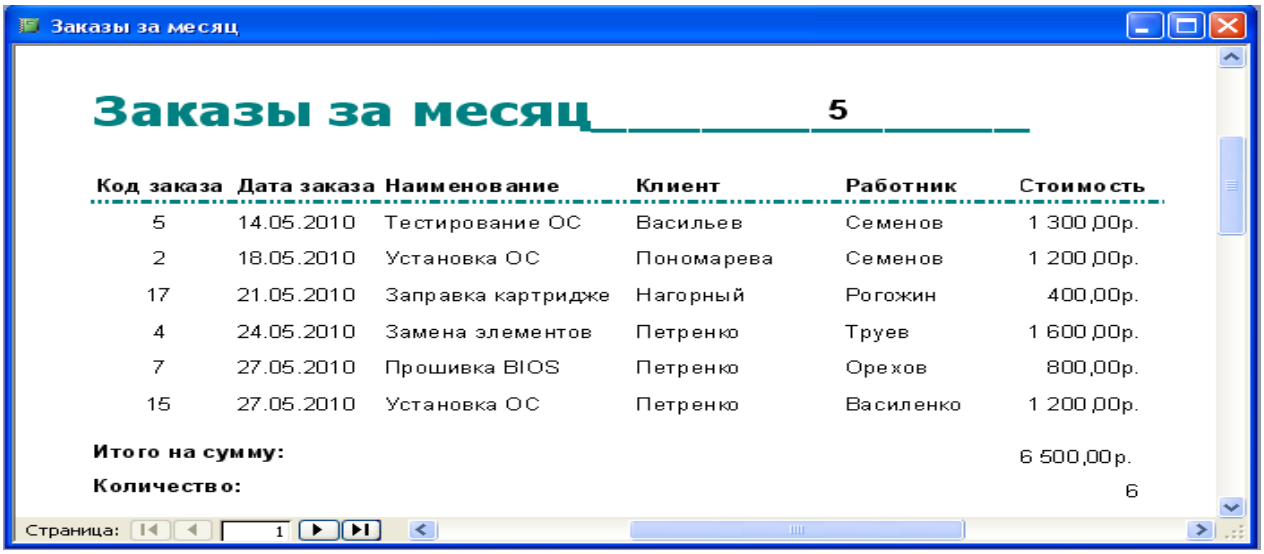
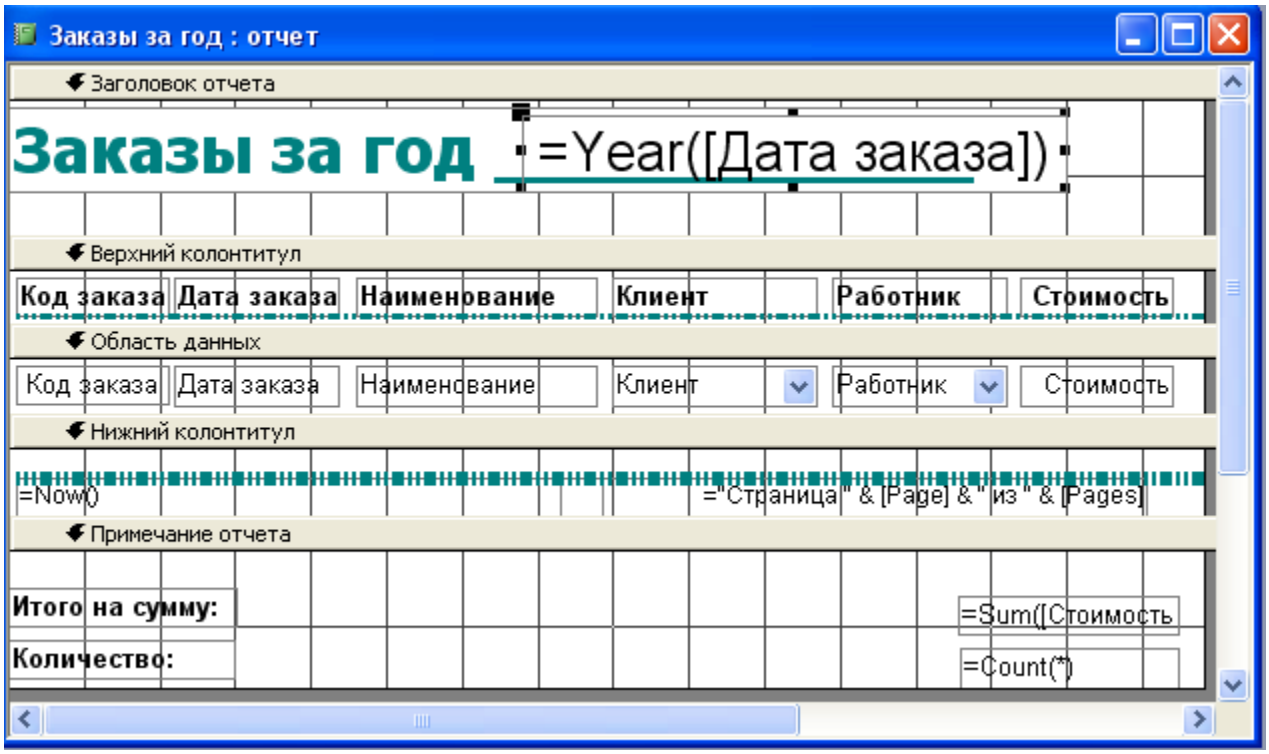


Рисунок 6 – отчет о работе за указанный месяц.

Результат выборки данных

В отчете также подведены итоговые суммы.



Заказы за год 2010

Код заказа	Дата заказа	Наименование	Клиент	Работник	Стоимость
5	14.05.2010	Тестирование ОС	Васильев	Семенов	1 300,00р.
2	18.05.2010	Установка ОС	Пономарева	Семенов	1 200,00р.
4	24.05.2010	Замена элементов	Петренко	Труев	1 600,00р.
15	27.05.2010	Установка ОС	Петренко	Василенко	1 200,00р.
8	16.06.2010	Заправка картрид	Шевченко	Семенов	400,00р.
6	21.06.2010	Замена элементов	Иванов	Орехов	1 600,00р.
12	21.06.2010	Тестирование ОС	Васильев	Орехов	1 300,00р.
11	24.06.2010	Прошивка BIOS	Шевченко	Рогожин	800,00р.

Итого на сумму: 9 400,00р.

Количество: 8

Рисунок 7 –Отчет о заказах фирмы за год.

Прежде, чем оформлять бланк-счет клиенту за выполненную работу, необходимо убедиться, что он полностью оплатил работу специалиста/ Для этого создан запрос – Оплаченные заказы.

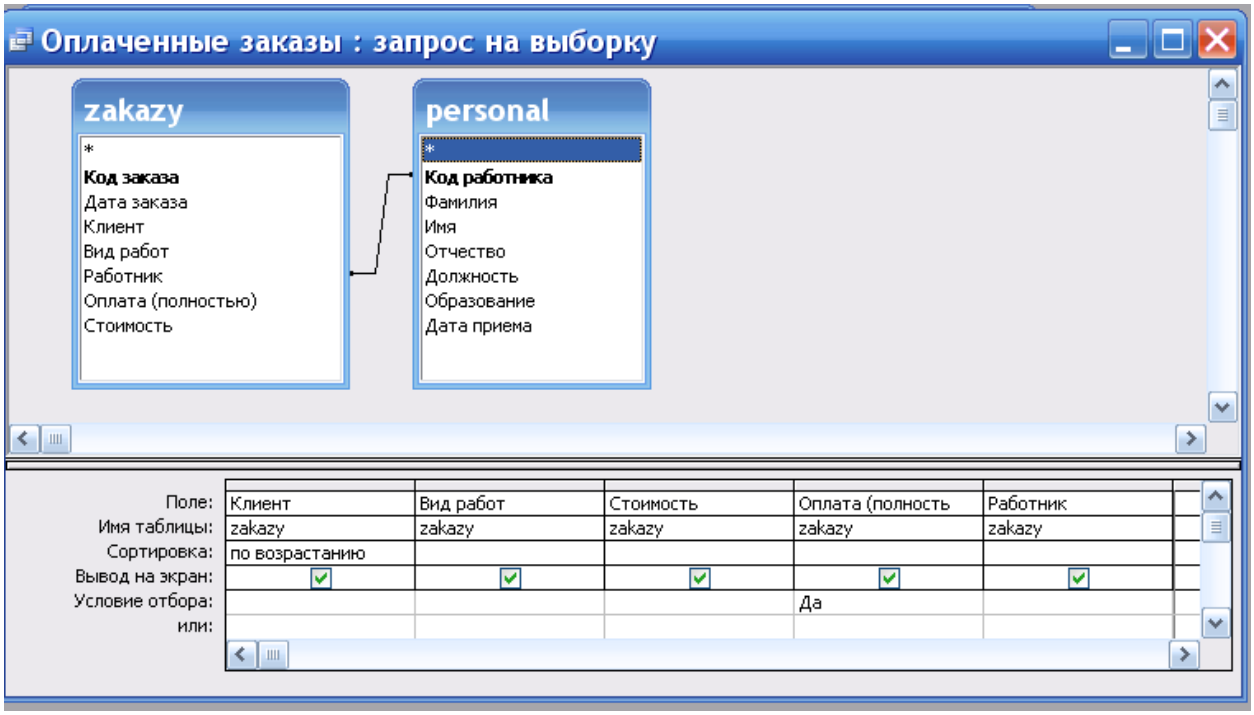


Рисунок 10- Запрос по оплате заказов.

Отчет по этому запросу приведен ниже.

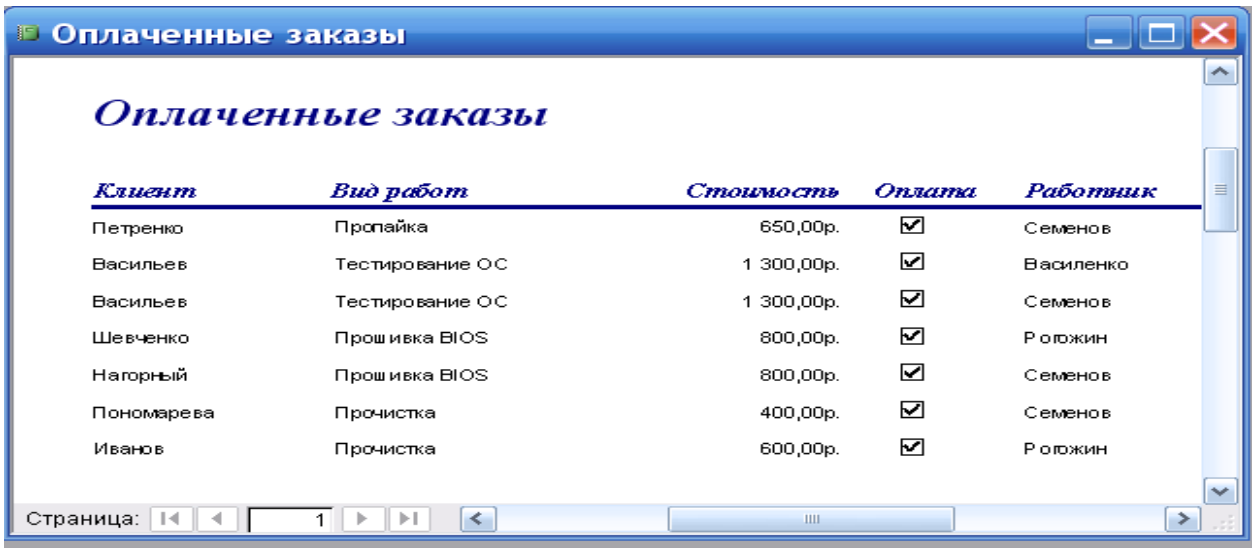


Рисунок 11 – Отчет по оплате заказов

На рисунке 12 отображено взаимодействие входной информации, созданных баз данных (файлов) и выходной информации.

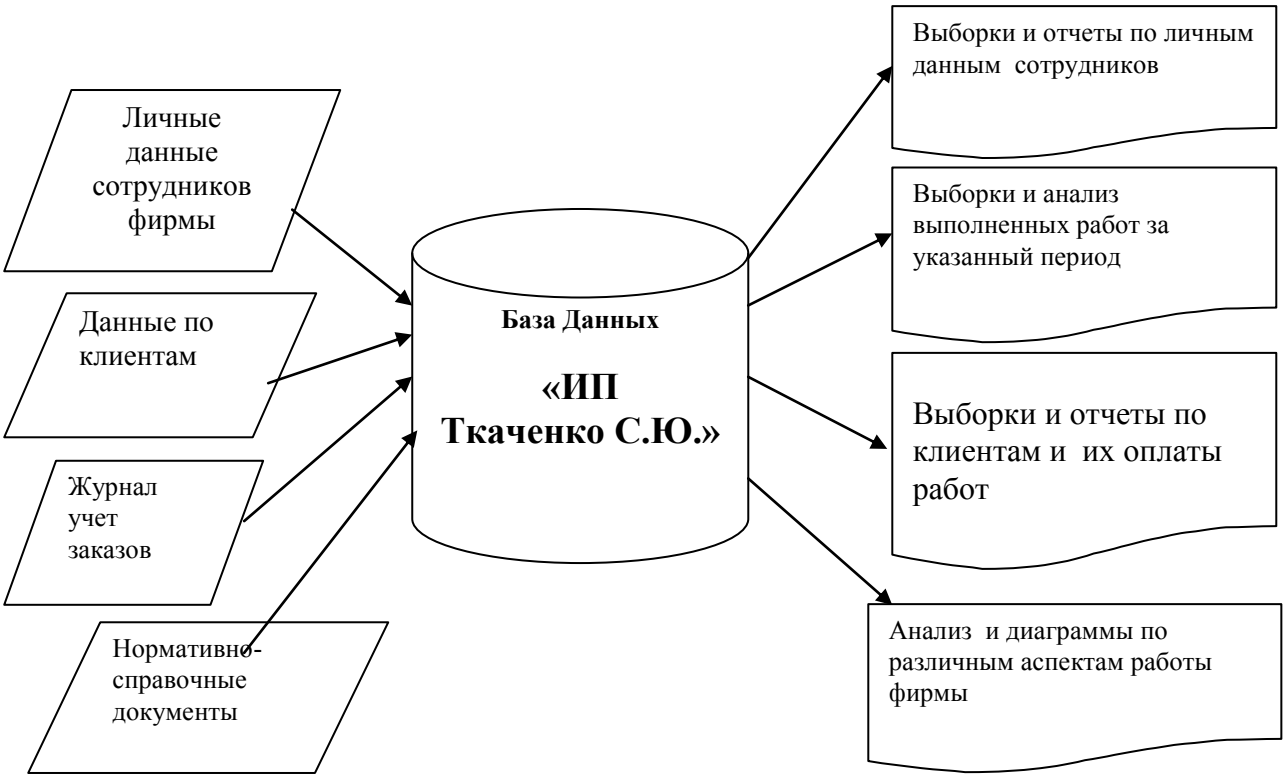


Рисунок 12 – Информационная модель задачи.

Описание программного средства

С точки зрения пользователя разработанное программное средство представляет собой совокупность форм, каждая из которых появляется на экране в результате какого-либо события, связанного с действием пользователя. Ниже представлена структура интерфейса данного программного средства. Взаимодействие пользователя с системой начинается с запуска приложения «БД УЧЕТ.mdb», после чего на экране появляется форма, предлагающая пользователю ввести пароль базы данных, как это показано на рисунке 14.

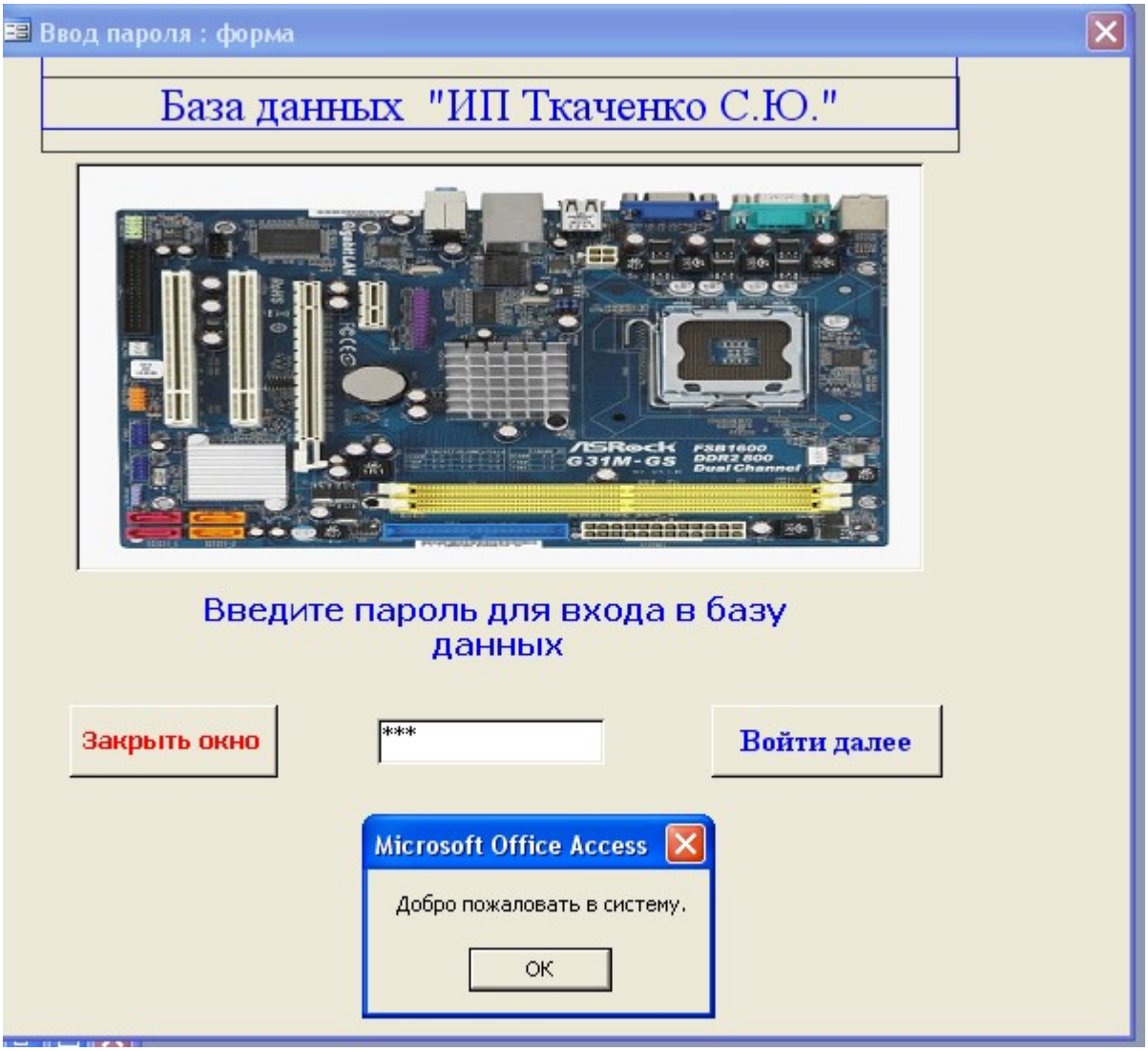


Рисунок 14 – Парольная форма

Если введен неверный пароль, то выводится сообщение.

При правильном вводе пароля открывается новая форма «главная форма», на которой выведены кнопки для просмотра и корректировки базы данных.

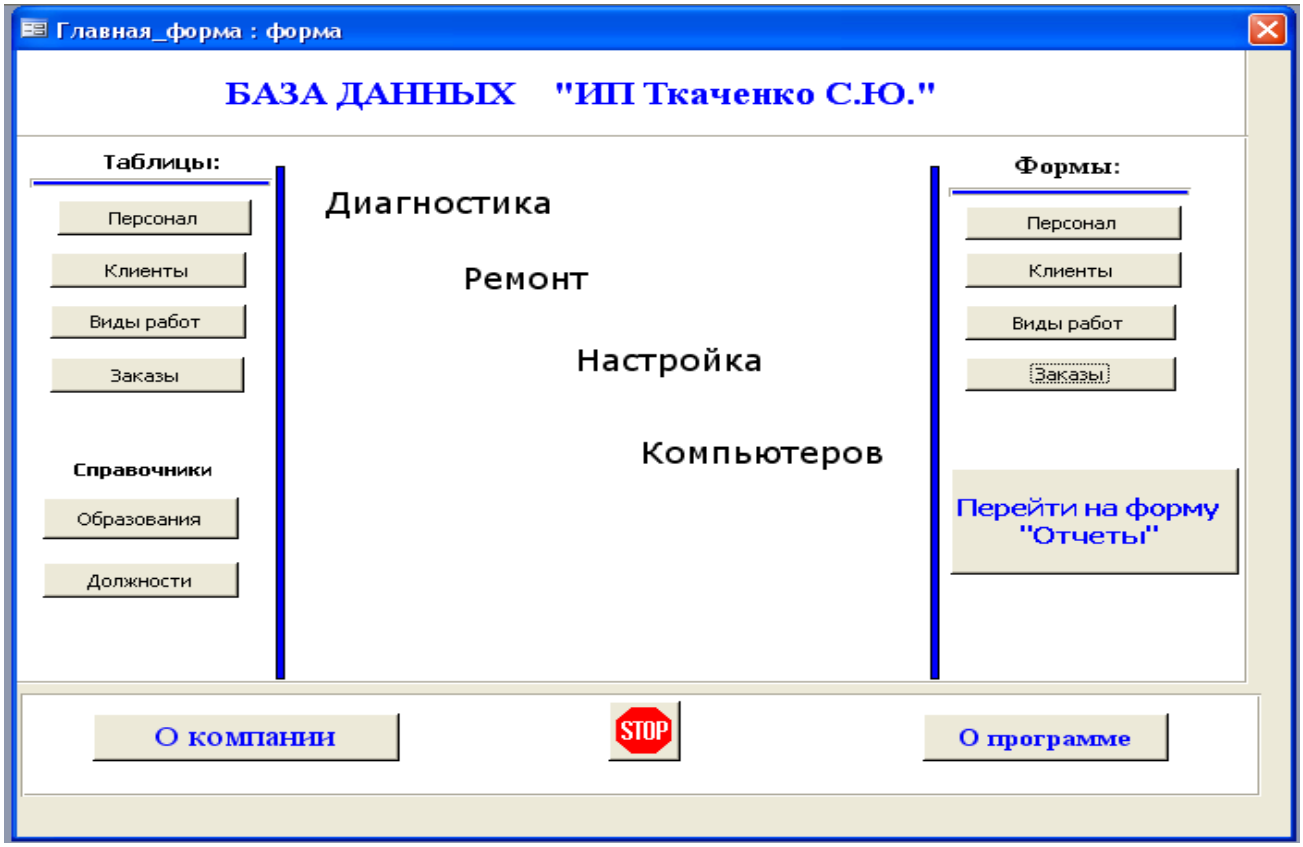


Рисунок 15 – Главная форма с кнопками для работы с объектами базы данных.

На этой же форме можно открыть и прочитать информацию о компании и о программе.

Формы со справочной информацией.

По кнопке форма-Персонал открывается форма для ввода нового работника или корректировки данных в таблице.

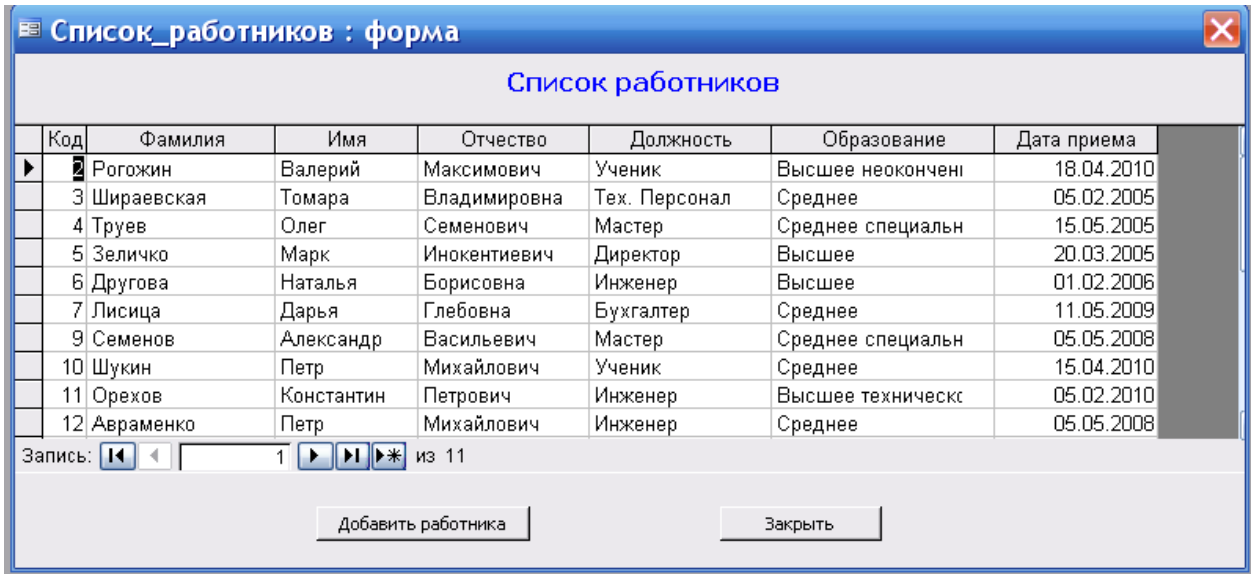
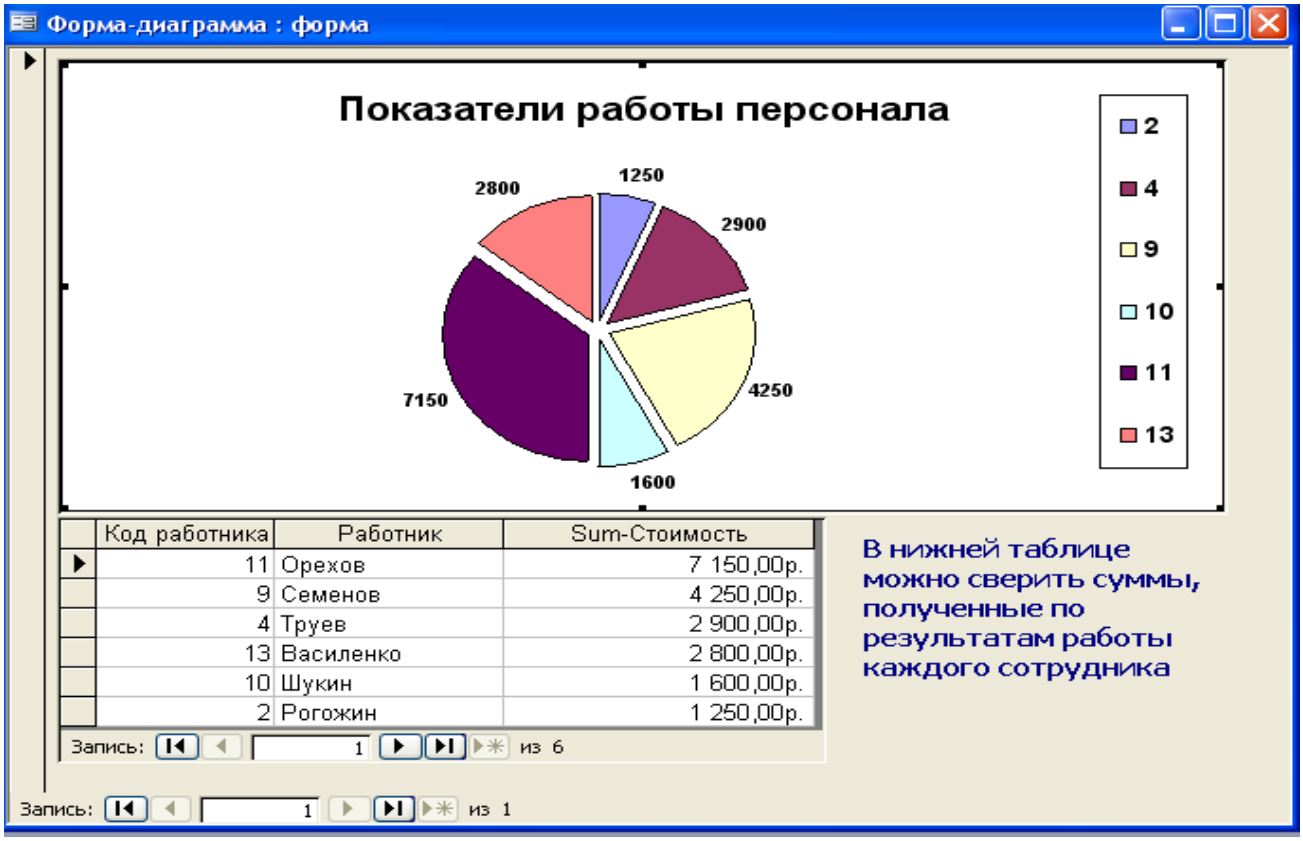


Рисунок 16 – форма для работы с таблицей «Персонал»

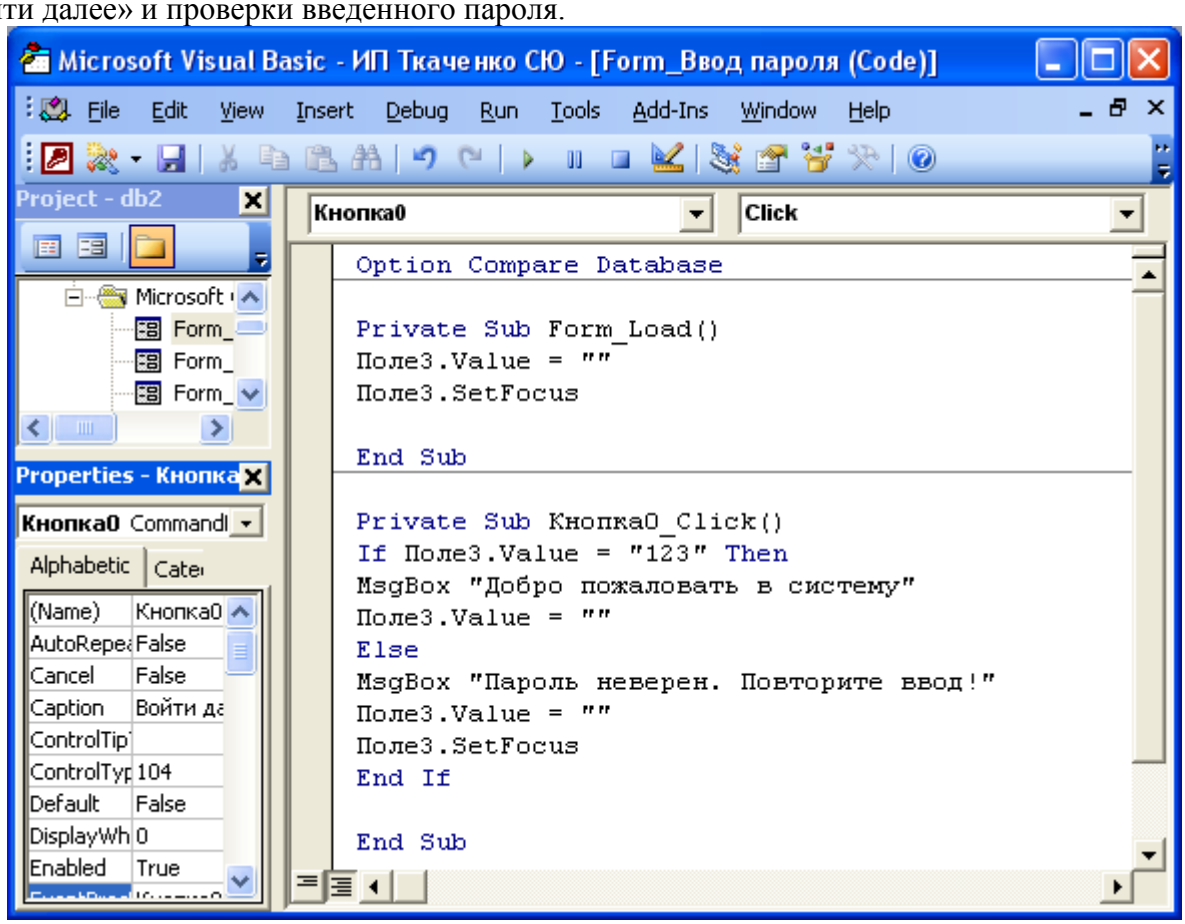
По кнопке Отчеты пользователь переходит на форму «Отчеты», на которой расположены кнопки для вызова запросов и отчетов.

Рисунок 17 – Форма для открытия запросов, отчетов и диаграммы

На кнопке «Диаграммы» динамически формируется диаграмма «Показатели работы персонала», на которой отображается выработка каждого сотрудника (в денежном выражении) за текущий год. Эта диаграмма позволяет сделать сравнительный анализ по эффективности работы каждого сотрудника.



При создании парольной формы используется среда программирования VBA- Visual Basic For Application. Процедура обработки события- клик по кнопке «Войти далее» и проверки введенного пароля.



В этой процедуре проверяется введенный в поле пароль, выводится соответствующее

сообщение и открывается Главная форма базы данных.

Таким образом, нами выполнены все поставленные задачи – разработаны таблицы, созданы формы для ввода данных, сформированы запросы и отчеты, построены диаграммы для визуализации выбранных данных. Информация в базе данных защищена паролем.

7 Основы языка SQL

Рассматривая вопросы, связанные с БД и СУБД, было бы нелогично оставить в стороне язык баз данных – SQL. Язык SQL в настоящее время является промышленным стандартом. Язык SQL предназначен для манипулирования данными в реляционных базах данных, определения структуры баз данных и для управления правами доступа к данным в многопользовательской среде. Встроенный SQL используется в прикладных программах, позволяя им посылать запросы к серверу и обрабатывать полученные результаты.

В начале 70-х годов в компании IBM была разработана экспериментальная СУБД SystemR. Целью разработки было создание простого непроцедурного языка, которым мог воспользоваться любой пользователь, даже не имеющий навыков программирования. В 1981 году IBM объявила о своем первом, основанном на SQL программном продукте – SQL/DS. Чуть позже к ней присоединились Oracle и другие производители. Первый стандарт языка SQL был принят Американским национальным институтом стандартизации (ANSI) в 1987 и несколько уточнен в 1989 году
(SQL level 2). Дальнейшее развитие языка поставщиками СУБД потребовало принятия в 1992 нового расширенного стандарта. В настоящее время ведется работа по подготовке третьего стандарта SQL, который должен включать элементы объектно-ориентрованного доступа к данным.

В SQL определены три подмножества языка:

- язык манипулирования данными (Data Manipulation Language, DML)
- язык определения данных (Data Definition Language, DDL)
- язык управления доступом (Data Control Language, DCL).

- **SQL-DDL** (Data Definition Language) – язык определения структур и ограничений целостности баз данных. Сюда относятся команды создания и удаления баз данных; создания, изменения и удаления таблиц; управления пользователями и т.д.
- **SQL-DML** (Data Manipulation Language) – язык манипулирования данными: добавление, изменение, удаление и извлечение данных, управления транзакциями

Как и большинство языков программирования, SQL реализует стандартный набор типов данных дополненный парой специфических для баз данных типов значений. Перечислим основные типы данных языка:

- Символьные типы данных – содержат буквы, цифры и специальные символы: *CHAR* или *CHAR(n),VARCHAR(n)*
- Целые типы данных – поддерживают только целые числа (дробные части и десятичные точки не допускаются): *INTEGER* или *INT,SMALLINT*
- Вещественные типы данных – описывают числа с дробной частью: *FLOAT* и *SMALLFLOAT ,DECIMAL(p), DECIMAL(p,n)*
- Денежные типы данных – описывают, естественно, денежные величины: *MONEY(p,n).*
- Дата и время – используются для хранения даты, времени и их комбинаций: *DATE,TIME,INTERVAL,DATETIME*

Перечислим основные операторы DDL

Таблица 3 – Операторы базы данных:

Команда	Описание
Создание базы данных.	**CREATE DATABASE <имя_базы_данных>**
Удаление базы данных.	**DROP DATABASE <имя_базы_данных>**

Таблица 4 – Операторы модификации таблиц:

Команда	Описание
Добавить столбцы	**ALTER TABLE <имя_таблицы> ADD (<имя_столбца> <тип_столбца> [NOT NULL] [UNIQUE | PRIMARY KEY] [REFERENCES <имя_мастер_таблицы> [<имя_столбца>]] ,...)**
Удалить столбцы	**ALTER TABLE <имя_таблицы> DROP (<имя_столбца>,...)**
Модификация типа столбцов	**ALTER TABLE <имя_таблицы> MODIFY (<имя_столбца> <тип_столбца> [NOT NULL] [UNIQUE | PRIMARY KEY] [REFERENCES <имя_мастер_таблицы> <имя_столбца>]] ,...)**

Таблица 5 – Операторы работы с индексами

Команда	Описание
Создание индекса	**CREATE [UNIQUE] INDEX <имя_индекса> ON <имя_таблицы> (<имя_столбца>,...) [REFERENCES <имя_мастер_таблицы> [<имя_столбца>]],...)**
Удаление индекса	**DROP INDEX <имя_индекса>**

Таблица 6 – Операторы управления правами доступа

Команда	Описание
Операция передачи прав на таблицу	**GRANT <тип_права_на_таблицу> ON <имя_таблицы> [<список_столбцов>] TO <имя_пользователя>** **Типы прав:** • SELECT - получение информации из таблицы • UPDATE - изменение информации в таблице • INSERT - добавление записей в таблицу • DELETE - удаление записей из таблицы • INDEX - индексирование таблицы • ALTER - изменение схемы определения таблицы • ALL - все права
Отмена прав	**REVOKE <тип_права_на_таблицу> ON <имя_таблицы> [<список_столбцов>] FROM <имя_пользователя>**

Основные операторы манипулирования данными приведены в табл.7

Таблица 7 – Команды модификации данных

Добавление записи в таблицу	**INSERT INTO <имя_таблицы> [(<имя_столбца>,<имя_столбца>,...)] VALUES (<значение>,<значение>,..)**
Модификация записи	**UPDATE <имя_таблицы> SET <имя_столбца>=<значение>,... [WHERE <условие>]**
Удаление записи	**DELETE FROM <имя_таблицы> [WHERE <условие>]**

Поэтому, в язык SQL в качестве составных частей входят:
- язык манипулирования данными (Data Manipulation Language, DML)
- язык определения данных (Data Definition Language, DDL)
- язык управления данными (Data Control Language, DCL).

Язык определения данных используется для создания и изменения структуры базы данных и ее составных частей - таблиц, индексов, представлений (виртуальных таблиц), а также триггеров и сохраненных процедур.

Основными его командами являются:
CREATE DATABASE (создать базу данных)
CREATE TABLE (создать таблицу)
CREATE INDEX (создать индекс)
CREATE TRIGGER (создать триггер)
CREATE PROCEDURE (создать хранимую процедуру)
ALTER DATABASE (модифицировать базу данных)
ALTER TABLE (модифицировать таблицу)
ALTER INDEX (модифицировать индекс)
ALTER TRIGGER (модифицировать триггер)
ALTER PROCEDURE (модифицировать сохраненную процедуру)
DROP DATABASE (удалить базу данных)
DROP TABLE (удалить таблицу)
DROP INDEX (удалить индекс)
DROP TRIGGER (удалить триггер)
DROP PROCEDURE (удалить сохраненную процедуру).

Язык манипулирования данными используется, как это следует из его названия, для манипулирования данными в таблицах баз данных. Он состоит из 4 основных команд:

SELECT (выбрать)
INSERT (вставить)
UPDATE (обновить)
DELETE (удалить).

Язык управления данными используется для управления правами доступа к данным и выполнением процедур в многопользовательской среде. Более точно его можно назвать “язык управления доступом”. Он состоит из двух основных команд:
GRANT (дать права)
REVOKE (забрать права).

Операторы (инструкции) языка SQL

Операторы (инструкции) языка SQL в СУБД MS Access используются при разработке форм, отчётов, а также написании макрокоманд и программ. Программы в Access хранятся в

модулях, написанных на языке программирования Access Basic (в ранних версиях MS Access) либо Access Basic для приложений, которые поглощают язык Access Basic.

К числу основных операторов языка SQL, реализованного в Access, относятся следующие: ALTER TABLE, CREATE INDEX, CREATE TABLE, DELETE, DROP, INSRT INTO, SELECT, TRANSFORM и UPDATE.

Приведём описание инструкций, которое составлено по сведениям справочной системы Access.

1. Инструкции ALTER TABLE

Назначение: изменение структуры таблицы, созданной с помощью инструкции CREATE TABLE.
Синтаксис:

```
ALTER TABLE <таблица>
{ADD {COLUMN<поле><тип>[(<размер>)][CONSTRAINT<индекс>] |
CONSTRAINT <составной _ индекс>} |
DROP {COLUMN <поле>| CONSTRAINT<имя _ индекса>}
```

Пример:

Удалить поле «Оклад» из таблицы «Сотрудники», которая создана ранее с помощью инструкции CREATE TABLE.

```
ALTER TABLE Сотрудники DROP COLUMN Оклад;
```

2. Инструкция CREATE INDEX

Назначение: создание нового индекса для существующей таблицы.
Синтаксис:

```
CREATE [UNIQUE] INDEX<индекс>
ON <таблица> (<поле> [ASC|DESC][, <поле> [ASC|DESC],…])
WITH {PRIMARI | DISALLOW NULL | IGNORE NULL}]
```

Пример:

Создать в таблице «Клиенты» индекс по полю «Код Клиента». В этом поле исключить повторяющиеся и пустые значения.

```
CREATE UNIQUE INDEX Индекс Клиента ON Клиенты (Код Клиента)
WITH DISALLOW NULL;
```

3. Инструкция CREATE TABLE

Назначение: создание новой таблицы.
Синтаксис:

```
CREATE TABLE <таблица> (
<поле 1> <тип> [(<размер>)] [<индекс 1>]
[, <поле 2> <тип> [(<размер>)] [<индекс 2>] [,…]]
[, <составной _ индекс> [, ... ]])
```

Пример:

Создать новую таблицу «Таблица 1» с двумя текстовыми полями и одним целочисленным полем. Поле «Страховка» сделать ключевым.

CREATE TABLE Таблица 1 (Имя TEXT, Фамилия TEXT, Страховка INTEGER CONSTRAINT Индекс 1 PRIMARY KEY);

Инструкция DELETE

Назначение: создание запроса на удаление записей из одной или нескольких таблиц, перечисленных в предложении FROM, которые удовлетворяют предложению WHERE.

Синтаксис:

DELETE [<таблица>.*]
FROM <таблица>
WHERE <условие _ отбора>

Пример:

Удалить записи о всех сотрудниках, которые занимают должность «Стажёр» и имеют записи в таблице «Оплата». Между таблицами «Сотрудники» и «Оплата» установлена связь 1:1.

DELETE Сотрудники. * FROM Сотрудники INNER JOIN Оплата
ON Сотрудники. Код Сотрудника = Оплата. Код Сотрудника
WHERE Сотрудники. Должность = 'Стажёр';

Инструкция DROP

Назначение: удаление таблицы из базы данных или индекса из таблицы.

Синтаксис:

DROP {TABLE < таблица > |INDEX <индекс> ON < таблица >}

Аргументы: <таблица> - имя таблицы, которую следует удалить или из которой следует удалить индекс;
<индекс> - имя индекса, удаляемого из таблицы;

Пример: Удалить «Индекс 1» из таблицы «Стажеры».

DROP INDEX Индекс 1 ON Стажеры;

4. Инструкция INSERT INTO

Назначение: добавить запись или записи в таблицу. Эта инструкция образует запрос на добавление.

Синтаксис:

Запрос на добавление нескольких записей:

INSERT INTO <назначение>
[IN<внешняя _ база данных>] [(<поле 1> [,<поле 2> [,…]])]
SELECT [<источник>] < поле 1> [, < поле 2[,…]
FROM <выражение>

Запрос на добавление одной записи
INSERT INTO <назначение> [(<поле 1> [, <поле 2> [,…]])]
VALUES (<значение 1> [,<значение 2> [,…])

Пример: Отобрать все записи из таблицы «Стажёры» для стажёров, принятых на работу более 30 дней назад, и добавить их в таблицу «Сотрудники».

INSERT INTO Сотрудники SELECT Стажеры.* FROM Стажеры
WHERE Дата Найма <Now ()-30;

1. Инструкция SELECT

Назначение: представить данные из база данных в виде набора записей.

Синтаксис: SELECT [<предикат>] {* |<таблица>.*| [<таблица.>] <поле 1>
[AS<псевдоним 1>] [,[<таблица>.] <поле 2>[AS <псевдоним 2>][,…]}
FROM <выражение> [,…] [IN < внешняя _ база данных >]
[WHERE…]
[GROUP BY…]
[HAVING…]
[ORDER BY…]
[WITH OWNERACCESS OPTION]

Примеры: Отобрать все поля из таблицы «Сотрудники».

SELECT Сотрудники.* FROM Сотрудники;

Посчитать число записей, которые содержат не пустое значение в поле «Индекс», и присвоить заголовок «Итого» полю, в которое возвращается результат.

SELECT Count (Индекс) AS Итого FROM Клиенты;

Вывести число сотрудников и их среднюю и максимальную зарплату.

SELECT Count (*) AS Число Сотрудников, Avg (Оклад) AS Средний оклад, Max(Оклад) AS Максимальный оклад FROM Сотрудники;

Инструкция UPDATE

Назначение: создание запроса на обновление записей, который изменяет значение полей указанной таблицы на основе заданного условия отбора.

Синтаксис: UPDATE <таблица>
SET <новое _ значение>
WHERE <условие _ отбора>;

Пример: Увеличить на 10 процентов цену на все товары поставщика, имеющего код 8, поставки которых ещё не запрещены.

UPDATE Товары SET Цена = Цена * 1.1 WHERE Код Поставщика = 8
AND Поставки прекращены = No;

2. Предложение CONSRTAINT

Назначение: создание или удаление индексов в инструкциях ALTER, TABLE CREATE и TABLE. С его помощью можно создавать или удалять простой

индекс (по одному полю) или составной индекс (по нескольким полям).
Синтаксис: предложения CONSRAINT для создания простого индекса:

CONSRAINT <имя>
PRIMARY KEY (<ключевое 1> [, <ключевое 2> [,…]]) |
UNIQUE (<уникальное 1> [, <уникальное 2> [,…]])] |
FOREIGN KEY (<ссылка 1> [, <ссылка 2> [,..]])
REFERENCES <внешняя _ таблица>[(<внешнее _ поле 1>
[,<внешнее _ поле 2> [,…]])]}

Выборка данных. Для извлечения записей из таблиц в SQL определен оператор *SELECT*. С помощью этой команды осуществляется не только операция реляционной алгебры "выборка" (горизонтальное подмножество), но и предварительное соединение (join) двух и более таблиц. Это наиболее сложное и мощное средство SQL, полный синтаксис оператора *SELECT* имеет вид:

SELECT [ALL | DISTINCT] <список_выбора>
FROM <имя_таблицы>, ...
[WHERE <условие>]
[GROUP BY <имя_столбца>,...]
[HAVING <условие>]
[ORDER BY <имя_столбца> [ASC | DESC],...]

Порядок предложений в операторе *SELECT* должен строго соблюдаться (например, *GROUP BY* должно всегда предшествовать *ORDER BY*), иначе это приведет к появлению ошибок.

Как видно, SQL предоставляет все необходимы средства не только для работы со структурой модели данных и отдельных отношений, но для манипулирования хранимой в отношениях информацией. Однако главное назначение языка кроется в другом. С его помощью организуется процесс общения (обмена информацией) между серверной частью (собственно сама СУБД) и клиентской частью (пользовательское приложение). Некоторые вопросы данного взаимодействия будут освещены далее.

SELECT [DISTINCT] список_выбираемых_элементов (полей)

FROM список_таблиц (или представлений)

[WHERE -условия выборки]

[GROUP BY поле (или поля) [HAVING предикат]]

[UNION другое_выражение_Select]

[ORDER BY поле (или поля) или номер (номера)];

Любой SQL-запрос должен заканчиваться символом «;»

Пример, SELECT NAME, SURNAME
FROM STUDENT;

Операция выборки позволяет получить все строки (записи) либо часть строк одной таблицы.
SELECT * FROM country - Получить все строки таблицы Country
Выделить подмножество **столбцов** таблицы. Например:

SELECT currency FROM country Получить список денежных единиц

Например,

запрос «Получить список названий городов, где проживают студенты, в таблице STUDENT», можно записать в следующем виде.

SELECT CITY
FROM STUDENT; Его результатом будет таблица: CITY

Предикат **WHERE**

предложения **WHERE**, содержит условия отбора

Типы предикатов, используемых в предложении WHERE:

- **сравнение с использованием реляционных операторов**

= равно <> не равно != не равно > больше < меньше >= больше или равно <= меньше или равно

- **BETWEEN**
- **IN**
- **LIKE**
- **CONTAINING**
- **IS NULL**
- **EXIST**
- **ANY**
- **ALL**

Например, запрос для получения имен и фам. студентов, обучающихся на третьем курсе и получающих стипендию (размер стипендии больше нуля), будет выглядеть таким образом:

```
SELECT NAME,SURNAME
FROM STUDENT
 WHERE KURS = 3 AND STIP > 0;
```

SELECT first_name, last_name, dept_no, job_country

FROM employee

WHERE job_country <> "USA" получить список сотрудников (а также номера их отделов и страну), работающих вне США

Операторы IN, BETWEEN, LIKE, IS NOLL

При задании логического условия Построенный с использованием IN предикат (условие) считается истинным, если значение поля, имя которого указано слева от IN, соблд&re/и (подразумевается точное совпадение) с одним из значений, перечисленных в списке, указанном в скобках справа от IN.

Предикат, построенный с использованием NOT IN, считается истинным, если значение поля, имя которого указано слева от NOT IN, Йе cqp/M&zem ни с одним из значений, перечисленных в списке, указанном в скобках справа от NOT IN. Примеры

Получить из таблицы EXAM_MARKS сведения о студентах, экзаменационные оценки только 4 и 5.

```
SELECT *
FROM EXAM_MARKS
    WHERE MARK IN (4, 5);
```

Получить сведения о студентах, не имеющих ни одной экзаменационной оценки, равной 4 и 5.

```
SELECT *
   FROM EXAM_MARKS
   WHERE MARK NOT IN (4, 5);
```

Оператор BETWEEN используется для проверки условия вхождения значения поля в заданный интервал, то есть вместо списка значений атрибута этот оператор задает границы его изменения.

Например, **запрос на вывод записей о предметах, на изучение которых отводится количество часов, находящееся в пределах между 30 и 40, имеет вид:**

```
SELECT *
    FROM SUBJECT
    WHERE HOUR BETWEEN 30 AND 40;
```

Граничные значения, в данном случае значения 30 и 40, %/я во множество значений, с которыми производится сравнение. Оператор BETWEEN может использоваться как для числовых, так и для символьных типов полей. Оператор LIKE применим только к символьным полям типа CHAR или VARCHAR (см. раздел 1.5 «Типы данных SQL»).

Предикат BETWEEN с отрицанием NOT (NOT BETWEEN) позволяет получить выборку записей, указанные поля которых имеют значения меньше нижней границы и больше верхней границы.

SELECT first_name, last_name, hire_date FROM employee WHERE hire_date NOT BETWEEN "1-JAN-1989" AND "31-DEC-1993" - получить список самых “старых” и самых “молодых” (по времени поступления на работу) сотрудников

А вот пример запроса, использующего предикат NOT IN:

SELECT first_name, last_name, job_country
FROM employee
WHERE job_country NOT IN ("USA", "Japan", "England") - получить список сотрудников, работающих не в США, не в Японии и не в Великобритании

LIKE

Предикат LIKE используется только с символьными данными. Он проверяет, соответствует ли данное символьное значение строке с указанной маской. В качестве маски используются все разрешенные символы (с учетом верхнего и нижнего регистров), а также специальные символы:

% - замещает любое количество символов (в том числе и 0),

_ - замещает только один символ.

Разрешено также использовать конструкцию NOT LIKE.

SELECT first_name, last_name FROM employee

WHERE last_name LIKE "F%" - получить список сотрудников, фамилии которых начинаются с буквы “F”

IS NULL

В SQL-запросах **NULL** означает, что значение столбца неизвестно. Поисковые условия, в которых значение столбца сравнивается с NULL, всегда принимают значение unknown (и, соответственно, приводят к ошибке), в противоположность true или false.

Предикат **IS NULL** принимает значение true только тогда, когда выражение слева от ключевых слов “IS NULL” имеет значение null (пусто, не определено). Разрешено также использовать конструкцию IS NOT NULL, которая означает “не пусто”, “имеет какое-либо значение”.

SELECT department, mngr_no

FROM department WHERE mngr_no IS NULL- получить список отделов, в которых еще не назначены начальники

Преобразования выводимых данных

Список выбираемых элементов может содержать следующее:

- имена полей
- *
- вычисления
- литералы
- функции преобразования данных
- агрегирующие функции

Преобразование вывода и встроенные функции

В SQL реализованы операторы преобразования данных и встроенные функции, предназначенные для работы со значениями столбцов и/или константами в выражениях. Использование этих операторов допустимо в запросах везде, где допустимы выражения.
2.3.1. Числовые, символьные и строковые константы
Для придания большей наглядности получаемому результату можно использовать литералы. Литералы - это строковые константы, которые применяются наряду с наименованиями столбцов и, таким образом, выступают в роли “псевдостолбцов”. Строка символов, представляющая собой литерал, должна быть заключена в одинарные или двойные скобки Они помещаются в списке столбцов и могут сопровождаться псевдонимами.

Например, результатом выполнения запроса
SELECT 'Фамилия', SURNAME, ‘Имя’, NAME,
FROM STUDENT; Результат является таблица следующего вида:

Фамилия Иванов Имя Иван
Фамилия Петров Имя Петр

SELECT first_name, "получает", salary, "долларов в год"

FROM employee - получить список сотрудников и их зарплату

========== ======== ========== ==============

Robert получает 105900.00 долларов в год

Операция конкатенации строк

Операция конкатенации «||» позволяет соединять (склеивать)значения двух или более столбцов символьного типа или констант в одну строку. Операция имеет синтаксис

SELECT SURNAME |' '| NAME, STIPEND
FROM STUDENT
WHERE KURS = 4 AND STIPEND > 0; Результат запроса будет

	STIPEND
Сидоров_Вадим	150
Петров_Антон	200

Функции преобразования символов в строке

* LOWER(<строка>) — перевод встречные символы (нижниЩ регистр)
* UPPER(<строка>) — перевод в прописные символы (верхний регистр)
* INITCAP (<строка>) - перевод первой буквы каждого слова строки в прописную (заглавную)

Например:

SELECT INITCAP (SURNAME), UPPER(NAME) FROM STUDENT
WHERE KURS = 4 AND STIPEND > 0;

Результат запроса будет выглядеть следующим образом:

SURNAME	NAME
Сидоров	ВАДИМ

Строковые функции

LTRIM(<строка>[,<подстрока>]) — удаление левых граничных символов
Из строки> удаляются слева символы, указанные в <подстроке>;
* если <подстрока> не указана, по умолчанию удаляются пробелы;

RTRIM(<строка>[,<подстрока>]) — удаление правых граничных символов
* из <строки> удаляются справа символы, указанные в <подстроке>;
* если <подстрока> не указана, по умолчанию удаляются пробелы;

Функции LTRIM и RTRIM рекомендуется использовать при написании условных выражений, в которых сравниваются текстовые строки. Дело в том, что наличие начальных или конечных пробелов в сравниваемых операндах может исказить результат сравнения.
константы ' AAA' и 'AAA ' не равны друг другу.

SUBSTR (<строка>,<начало>[,<количество>]) - выделение подстроки
и з <строки> выбирается заданное <количество> символов, начиная с указанной параметром <начало> позиции и строке;

*** INSTR — ПОИСК ПОДСТРОКИ**
INSTR (<строка>,<подстрока>[,<начало поиска> [,<номер вхождения>]])

- <начало поиска> задает начальную позицию в строке для поиска <подстроки
- <номер вхождения> задает порядковый номер искомой подстроки. Если не задан, то по умолчанию принимается значение 1;
- функция возвращает позицию найденной подстроки.

LENGTH(<строка>) — определение длины строки тип возвращаемого значения — INT;
* функция возвращает NULL, если <строка> имеет NULL-зна чение.

```
SELECT  SUBSTR (NAME, 1, 1) ||'.'|| SURNAME, CITY, LENGTH (CITY)
 FROM STUDENT
WHERE KURS IN(2, 3, 4)AND STIPEND > 0;
```

результат:

	CITY	
П. Петров	Курск	5
С. Сидоров	Москва	6
О. Зайцева	Азов	4

Работа с датами

Дата может быть представлена строками различных форматов, например:

- “январь 27, 2005”
- “27-янв-2005”
- “10-27-95”
- “10/27/95”

SELECT first_name, last_name, hire_date
FROM employee
WHERE hire_date > '1-1-94' - получить список сотрудников, принятых на работу после 1 января 1994 года

Значения дат можно сравнивать друг с другом, сравнивать с относительными датами, вычитать одну дату из другой.

SELECT first_name, last_name, hire_date

FROM employee WHERE 'today' - hire_date > 365 * 7 + 1 -получить список служащих, проработавших на предприятии к настоящему времени более 7 лет

Над значениями типа DATE разрешены следующие операции:

* операция сложения;
* операция вычитания.

В операциях один из операндов должен иметь значение отдельного элемента даты: только год, или только месяц, или только день. Например:
* при добавлении к дате '22.05.1998' пяти лет получится дата "22.05.2003

- при добавлении к этой же дате девяти месяцев получится '22.02.1998
• при добавлении 10 дней получим '01.06.1998

При сложении двух полных дат результат непредсказуем.
Пример запроса:

```
SELECT SURNAME, NAME, BIRTHDAY, DATAPriema
TO_CHAR(BIRTHDAY, 'DD-MON-YYYY*),
TO_CHAR (DATAPriema, 'DD.MM.YY')
FROM STUDENT;
```

результат:

SURNAME	NAME	BIRTHDAY		
aaaaa	Иван	3/12/1982	3-дек-1982	3.12.82
bbbbbb	Петр	1/12/1980	1-дек-1980	1.12.80

Агрегирование и групповые функции

Агрегирующие функции позволяют получать из таблицы сводную (агрегированную) информацию, выполняя операции над группой строк таблицы. Для задания в SELECT-запросе агрегирующих операций используются следующие ключевые слова:
*** COUNT** определяет количество строк или значений поля, выбранных посредством запроса и не являющихся NOLL-значениями;
*** SuM** вычисляет арифметическую сумму всех выбранных значений данного поля;
*** AVG** вычисляет среднее значение для всех выбранных значений данного поля;
*** MAX** вычисляет наибольшее из всех выбранных значений данного поля;
*** MIN** вычисляет наименьшее из всех выбранных значений данного поля.

В SELECT-запросе агрегирующие функции используются аналогично именам полей, при этом последние (имена полей) используются в качестве аргументов этих функций.
Функция AVG предназначена для подсчета среднего значения поля на множестве записей таблицы.
Например, для определения среднего значения поля MARK (оценки) по всем записям таблицы EXAM_MARKS можно использовать запрос с функцией AVG следующего вида:

```
SELECT AVG (MARK)
FROM EXAM_MARKS;
```

Для подсчета общего количества строк в таблице следует использовать функцию COUNT со звездочкой.

```
SELECT COUNT(*)
FROM EXAM_MARKS;
```

Аргументы DISTINCT и ALL позволяют, соответственно, исключать и включать дубликаты обрабатываемых функцией COUNT значений, при этом необходимо учитывать, что при использовании опции ALL значения NOLL все равно не войдут в число подсчитываемых значений.

```
SELECT COUNT(DISTINCT SUBJ_ID)
FROM SUBJECT;
```

Права доступа

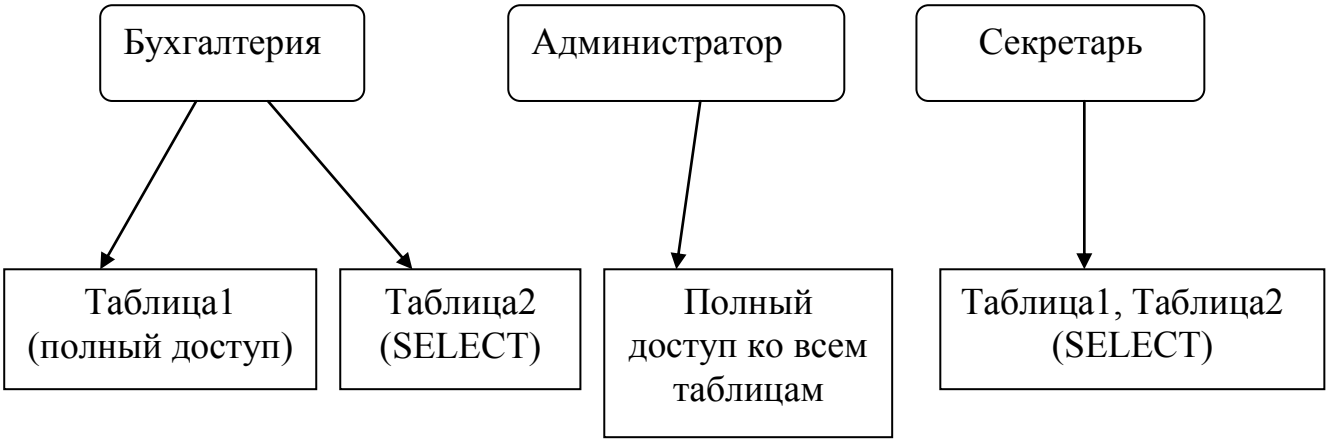
В режиме постоянного доступа к данным многими пользователями необходимо четко различать их действия с данными и обеспечивать защиту информации от несанкционированного доступа.
Работа с пользователями включает три этапа:

- Идентификация пользователя – т.е. он подтверждает, что он именно тот, кем представляется (пароль),

- П. имеет право оперировать только разрешенными объектами БД,
- П. совершает над объектами только разрешенные действия.

Система защиты включает три основных понятия:

- Пользователи,
- Объекты БД – таблицы, формы, и тд
- Привилегии – права П. на работу с таблицами БД.



Пользователи получают от АДМИНА свой идентификатор или пароль.
П., имеющий право на создание таблицы, является ее владельцем и он может назначить или передать права на действия с этой таблицей другому П.
Например,

Назначение полномочий

Команда **GRANT** <действия с таблицей> **ON** <имя таблицы> **TO** <пользователи>;

GRANT Select, Insert **ON** Student **TO** 12345;

GRANT Update(Stip) **ON** Student **TO** Olga;

Параметры **PUBLIC и ALL** **. ALL** используется для передачи **всех привилегий** на таблицу, **PUBLIC – (**общедоступный) относится ко **всем пользователям** системы.

GRANT ALL **ON** Student **TO** Иван;
GRANT Select **ON** Student **TO PUBLIC** ;
Пользователь – создатель таблицы обязательно ставит свое имя перед именем таблицы, если он передает привилегии кому-то еще.

GRANT Select **ON** PETRO.Student **TO** Olga With Grant Option;

Кстати, никто не может удалить таблицу, кроме создателя.

Снять полномочия.

Команда REVOKE <действия> ON <имя таблицы> FROM <Пользователь>;

REVOKE Insert **ON** Student **FROM** 12345;

Литература

1. Одиночкина С.В. Разработка базы данных в MS Access 2010: Учебное пособие СПб.: НИУ ИТМО 2013.
2. Тарасов В.Л. Работа с базами данных в MS Access 2010. Нижний Новгород. НГГУ, 2013.
3. Office. Microsoft.com