***Министерство образования и науки***

***Республики Казахстан***

***Карагандинский государственный технический университет***

КУРСОВАЯ РАБОТА

***по дисциплине Разработка баз данных***

***Тема: Разработка программного обеспечения по управлению базой данных «График учета рабочего времени на шахте им. Т. Кузембаева***

***2009***

**Содержание**

Введение

Задание

1. Выбор СУБД

2. Концептуальная модель

3. Логическая модель данных

4. Нормализация отношений

5. Основная часть

6. SQL – запросы

7. Интерфейс программы

8. Отчеты

Заключение

Список используемых источников

**Введение**

Современная жизнь немыслима без эффективного управления. Важной категорией являются системы обработки информации, от которых во многом зависит эффективность работы любого предприятия ли учреждения. Такая система должна:

1. обеспечивать получение общих и/или детализированных отчетов по итогам работы;
2. позволять легко определять тенденции изменения важнейших показателей;
3. обеспечивать получение информации, критической по времени, без существенных задержек;
4. выполнять точный и полный анализ данных.

Современные СУБД в основном являются приложениями Windows, так как данная среда позволяет более полно использовать возможности персональной ЭВМ, нежели среда DOS. Снижение стоимости высокопроизводительных ПК обусловил не только широкий переход к среде Windows, где разработчик программного обеспечения может в меньше степени заботиться о распределении ресурсов, но также сделал программное обеспечение ПК в целом и СУБД в частности менее критичными к аппаратным ресурсам ЭВМ.

Среди наиболее ярких представителей систем управления базами данных можно отметить: Lotus Approach, Microsoft Access, Borland dBase, Borland Paradox, Microsoft Visual FoxPro, Microsoft Visual Basic, а также баз данных Microsoft SQL Server и Oracle, используемые в приложениях, построенных по технологии «клиент-сервер». Фактически, у любой современной СУБД существует аналог, выпускаемый другой компанией, имеющий аналогичную область применения и возможности, любое приложение способно работать со многими форматами представления данных, осуществлять экспорт и импорт данных благодаря наличию большого числа конвертеров. Общепринятыми, также, являются технологи, позволяющие использовать возможности других приложений, например, текстовых процессоров, пакетов построения графиков и т.п., и встроенные версии языков высокого уровня (чаще – диалекты SQL и/или VBA) и средства визуального программирования интерфейсов разрабатываемых приложений. Поэтому уже не имеет существенного значения, на каком языке и на основе какого пакета написано конкретное приложение, и какой формат данных в нем используется. Более того, стандартом «де-факто» стала «быстрая разработка приложений» или RAD (от английского Rapid Application Development), основанная на широко декларируемом в литературе «открытом подходе», то есть необходимость и возможность использования различных прикладных программ и технологий для разработки более гибких и мощных систем обработки данных. Поэтому в одном ряду с «классическими» СУБД все чаще упоминаются языки программирования Visual Basic 4.0 и Visual C++, которые позволяют быстро создавать необходимые компоненты приложений, критичные по скорости работы, которые трудно, а иногда невозможно разработать средствами «классических» СУБД. Современный подход к управлению базами данных подразумевает также широкое использование технологии «клиент-сервер».

Таким образом, на сегодняшний день разработчик не связан рамками какого-либо конкретного пакета, а в зависимости от поставленной задачи может использовать самые разные приложения. Поэтому, более важным представляется общее направление развития СУБД и других средств разработки приложений в настоящее время.

Актуальность задачи составления расписания обусловлена важностью практических приложений. Как и другие разделы прикладной математики, методы теории расписаний позволяют решать задачи, возникающие экономике и управлении, информатике и электротехнике, исследовании операций и организации производственных процессов. Задачи составления расписаний являются задачами оптимизации и носят самый общий характер. Они возникают там, где существует возможность выбора той или иной очередности выполнения работ на производстве, при составлении расписания движения поездов и самолетов, обслуживании клиентов в сервисных системах, составление расписаний занятий в учебных заведениях и т.д.

**Задание**

В рамках данной курсовой работы необходимо разработать программное обеспечение по управлению базой данных « График учета рабочего времени на шахте им. Т. Кузембаева в цехе «Автоматика»» с использованием СУБД. Программное обеспечение должно обеспечивать просмотр, редактирование, вставку записей базы данных, формирование запросов и отчетов. В качестве исходных данных начальником вышеуказанного цеха были переданы готовые графики нескольких прошедших месяцев. По этим графикам следует установить закономерности и правила их составления, для осуществления этой задачи программным путем. Приложение должно учитывать возможный отпуск у рабочего, а также государственные праздники. Приложение должно предусматривать возможность сохранения сгенерированных графиков в БД.

Программный продукт должен:

1. Хранить информацию о рабочих цеха.
2. Хранить информацию о государственных праздниках
3. Сохранять и загружать сгенерированные графики
4. Автоматически генерировать график учета рабочего времени для выбранного пользователем месяца.
5. Сообщать об ошибке, возникающей в случае недостаточного количества рабочих для выполнения определенных функций.
6. Отображать информацию о выходных и праздничных днях, отпусках, а также сменах каждого рабочего

Составление графика учета рабочего времени имеет следующие правила:

* Выходить на работу в выходной день в первую смену могут только рабочие, имеющие статус «Вентиляторщик, работающий в выходные»
* Дежурить (ходить не в первую смену) могут только рабочие, имеющие статус «Дежурный»
* Остальные рабочие выходят только в первую смену по будним и не праздничным дням
* Выделить визуально рабочих со статусом статус «Вентиляторщик, работающий в выходные» и «Вентиляторщик, не работающий в выходные»
* Каждый день в каждую смену должен выходить хотя бы один рабочий
* Желательное максимальное количество рабочих дней без выходных – 5
* После четвертой смены нельзя выходить сразу в первую
* Одному рабочему нежелательно дежурить только в третью и четвертую смены (т.к. выход в эти смены более высоко оплачивается). Необходимо распределить такие дежурства более равномерно
* Количество рабочих дней в месяц равно количеству дней в месяце без субботы, воскресенья, а также государственных праздников
* После четвертой смены желательно предусмотреть два выходных

**1. Выбор СУБД**

Разработка проекта проводится в Borland Delphi. В качестве СУБД был выбран MS Access как пакет MS Office, который в настоящее время сильно распространен на территории нашего государства. В связи с тем что приложение разрабатывается для нужд стороннего лица, то целесообразно применять MS Access в качестве СУБД, что несомненно облегчит понимание приложения пользователем. Данный выбор позволит избежать проблем, которые бы возникли при развертывании MS SQL сервера. MS SQL требует администрирования (нужно выполнять какие то административные действия, типа резервного копирования и т. п., когда для Access нужно просто скопировать файл), которое на предприятии не будет проводиться, и сервер оно на много тяжелее в использовании, для обычных пользователей.

Для работы с базами данных в Delphi есть несколько наборов компонентов. Так как Microsoft встроила в свои продукты разработки технологию доступа к данным ADO (Active Data Objects), для реализации настоящего приложения используются компоненты с закладки ADO. Работа с базами данных Access идет через специальную надстройку DAO, которая устанавливается на компьютер вместе с Office, либо как отдельная установка. Для доступа к базам данных MS Access используется ADO драйвер «Microsoft Jet 4.0 OLE DB Provider».

Microsoft Access обладает всеми чертами классической системы управления базами данных (СУБД). Access – это не только мощная, гибкая и простая в использовании СУБД, но и система для разработки приложений баз данных. К числу наиболее мощных средств Access относятся средства разработки объектов – мастера, которые можно использовать для создания таблиц, запросов, различных типов форм и отчетов. В Microsoft Access включены мастера, помогающие производить анализ структуры данных, импортировать электронные таблицы и текстовые данные, повышать быстродействие приложения, создавать и настраивать одно из более, чем двадцати типов приложений с использованием встроенных шаблонов. Чтобы полностью автоматизировать работу приложения, можно использовать макросы для связывания данных с формами и отчетами. Большинство приложений можно создать, не написав ни единой строки программного кода. Однако при необходимости построения действительно сложного приложения можно использовать язык программирования – Visual Basic для приложений.

К областям применения Microsoft Access можно отнести следующие:

* в малом бизнесе (бухгалтерский учет, ввод заказов, ведение информации о клиентах, ведение информации о деловых контактах);
* в работе по контракту (разработка внутриотраслевых приложений, разработка межотраслевых приложений);
* в крупных корпорациях (приложения для рабочих групп, системы обработки информации);
* в качестве персональной СУБД (справочник по адресам, ведение инвестиционного портфеля, поваренная книга, каталоги книг, пластинок, видеофильмов и т. п.).

Рассмотрим основные определения, связанные с разработкой баз данных. Краткий словарь терминов приведен в приложении 1.

База данных (БД, data base, DB) – совокупность взаимосвязанных данных, используемых под управлением СУБД.

В самом общем смысле база данных – это набор записей и файлов, организованных специальным образом.

Система управления базой данных (СУБД, DBMS) – программная система, обеспечивающая определение физической и логической структуры базы данных, ввод информации и доступ к ней.

Возможности СУБД: система управления базами данных предоставляет возможность контролировать задание структуры и описание данных, работу с ними и организацию коллективного пользования информацией. СУБД также существенно увеличивает возможности и облегчает каталогизацию и ведение больших объемов хранящейся в многочисленных таблицах информации. СУБД включает в себя три основных типа функций: определение (задание структуры и описание) данных, обработки данных и управление данными.

Основные функции СУБД следующие:

Определение данных. Определяется, какая именно информация будет храниться в базе данных, задается структура данных и их тип (например, количество цифр или символов), а также указывается то, как данные будут связаны между собой. Задаются форматы и критерии проверки данных.

Обработка данных. Данные можно обрабатывать самыми различными способами. Можно выбирать любые поля, фильтровать и сортировать данные. Можно объединять данные с другой связанной информацией и вычислять итоговые значения.

Управление данными. Указываются правила доступа к данным, их корректировки и добавления новой информации. Можно также определить правила коллективного пользования данными.

**2. Концептуальная модель**

В приложении представлена концептуальная модель настоящей базы данных.

Концептуальная модель (англ. conceptual model) — это определённое множество понятий и связей между ними, являющихся смысловой структурой рассматриваемой предметной области.

Концептуальная модель - модель предметной области, состоящей из перечня взаимосвязанных понятий, используемых для описания этой области, вместе со свойствами и характеристиками, классификацией этих понятий, по типам, ситуациям, признакам в данной области и законов протекания процессов в ней. (Толковый словарь по искусственному интеллекту).

**3. Логическая модель**

Логическая модель данных отражает логические связи между сущностями базы данных. Ниже представлена логическая модель базы данных «График учета рабочего времени на шахте им. Т. Кузембаева в цехе «Автоматика» ».

Информацию о рабочих цеха содержит таблица «Рабочие», содержащая следующие атрибуты:

1. «ID» - уникальный идентификатор рабочего, выполняющий роль первичного ключа (счетчик);
2. «WokerName» - ФИО рабочего (текстовый);
3. «TabNumber» - табельный номар рабочего (числовой)
4. «LinkProfession» - код профессии рабочего (числовой), является вторичным ключом поля «ID» таблицы «Профессия»;
5. «LinkStatus» - код статуса рабочего (числовой), является вторичным ключом поля «ID» таблицы «Статус»;

Информацию о профессии рабочего хранится в таблице «Профессия», содержащая следующие атрибуты:

1. «ID» - уникальный идентификатор профессии, выполняющий роль первичного ключа (счетчик);
2. «NameProfession» - название профессии (текстовый);

Информацию о статусе рабочего хранится в таблице «Статус», содержащая следующие атрибуты:

1. «ID» - уникальный идентификатор статуса, выполняющий роль первичного ключа (счетчик);
2. «NameStatus» - название статуса (текстовый);

Информацию об отпусках рабочих содержит таблица «Отпуски», содержащая следующие атрибуты:

1. «ID» - уникальный идентификатор рабочего, выполняющий роль первичного ключа (счетчик);
2. «LinkWoker» - код ФИО рабочего (числовой), является вторичным ключом поля «ID» таблицы «Рабочие»;
3. «DateBegin» - дата начала отпуска (дата/время)
4. «Length» - продолжительность отпуска (числовой);

Информацию о сохраненных графиках содержит таблица «Сохраненные\_ графики», содержащая следующие атрибуты:

1. «ID» - уникальный идентификатор сохранения, выполняющий роль первичного ключа (счетчик);
2. «NameGraphic» - название сохраненного графика (текстовый);
3. «DateOfSave» - дата сохранения графика (дата/время);
4. «LinkMonthNum» - код месяца сохраненного графика (числовой), является вторичным ключом поля «ID» таблицы «Месяцы»;
5. «YearNum» - год сохраненного графика (числовой);

Информацию о графиках содержит таблица «Графики», содержащая следующие атрибуты:

1. «ID» - уникальный идентификатор графика, выполняющий роль первичного ключа (счетчик);
2. «LinkSaveID» - код номера сохранения графика (числовой), является вторичным ключом поля «ID» таблицы «Сохраненные графики»;
3. «DateOfCell» - дата ячейки (дата/время);
4. «LinkWoker» - код ФИО рабочего (числовой), является вторичным ключом поля «ID» таблицы «Рабочие»;
5. «LinkTypeOfDay» - код типа дня (числовой), является вторичным ключом поля «ID» таблицы «Типы\_дней»;

Информацию о месяце сохраненного графика хранится в таблице «Месяцы», содержащая следующие атрибуты:

1. «ID» - уникальный идентификатор месяца, выполняющий роль первичного ключа (счетчик);
2. «MonthName» - название месяца (текстовый);

Информацию выходных днях в связи с государственными праздниками хранится в таблице «Выходные», содержащая следующие атрибуты:

1. «ID» - уникальный идентификатор выходного дня, выполняющий роль первичного ключа (счетчик);
2. «DateCelebrate» - дата выходного дня (дата/время);
3. «LinkName» - код названия выходного дня в связи с праздником (числовой), является вторичным ключом поля «ID» таблицы «Праздники»;

Информацию названии государственных праздниках хранится в таблице «Праздники», содержащая следующие атрибуты:

1. «ID» - уникальный идентификатор праздника, выполняющий роль первичного ключа (счетчик);
2. «Title» - название государственного праздника (текстовый);

Наглядно связь между таблицами изображена на Рисунке 3.1

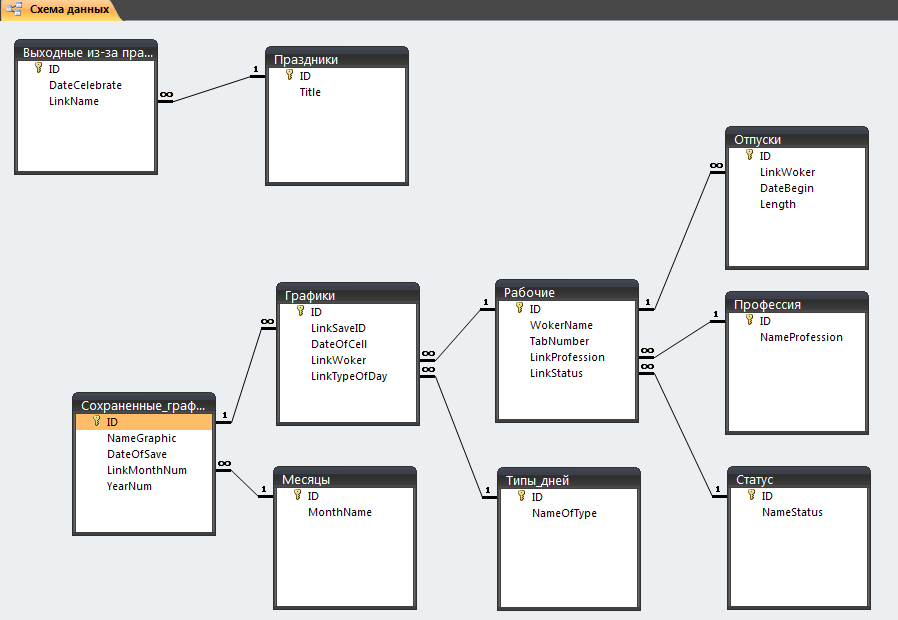


Рисунок 3.1 – Связи между таблицами БД «График учета рабочего времени на шахте им. Т. Кузембаева в цехе «Автоматика» »

**4. Нормализация отношений**

Нормальная форма — требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц). Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией.

Каждая нормальная форма представляет собой определённое условие, которому должна соответствовать таблица базы данных. Если таблица не соответствует нормальной форме, она может быть приведена к ней (нормализована) за счёт декомпозиции, то есть разбиения на несколько таблиц, связанных между собой. Теоретически, в результате нормализации объём БД должен уменьшиться. Принципиальным здесь является то, что нормализация — обратимый процесс, из группы таблиц, получившихся при декомпозиции, всегда можно получить в точности исходную таблицу. Таким образом, нормализация не сокращает объём информации, хранимой в БД, а лишь устраняет информацию, которая может быть вычислена.

Типы нормальных форм. Нормализация может применяться к таблице, первоначально отвечающей следующим требованиям:

* Таблица содержит нуль или более записей.
* Все записи таблицы имеют одно и то же множество полей, причём одноимённые поля относятся к одинаковым типам данных.
* Таблица не может содержать двух полностью идентичных записей.

Обычно выделяют шесть нормальных форм:

Первая нормальная форма (1NF). Таблица находится в первой нормальной форме, если каждый её атрибут атомарен и все строки различны. Под выражением "атрибут атомарен" понимается, что атрибут может содержать только одно значение. Таким образом, не соответствуют 1NF таблицы, в полях которых могут храниться списки значений. Для приведения таблицы к 1NF обычно требуется разбить таблицу на несколько отдельных таблиц.

Вторая нормальная форма (2NF). Таблица находится во второй нормальной форме, если она находится в первой нормальной форме, и при этом любой её атрибут, не входящий в состав первичного ключа, функционально полно зависит от первичного ключа. Функционально полная зависимость означает, что атрибут функционально зависит от всего первичного ключа, но при этом не находится в функциональной зависимости от какой-либо его части.

Третья нормальная форма (3NF). Таблица находится в третьей нормальной форме, если она находится во второй нормальной форме, и при этом любой её неключевой атрибут функционально зависит только от первичного ключа.

Нормальная форма Бойса-Кодда (BCNF)Таблица находится в BCNF, если она находится в третьей нормальной форме, и при этом отсутствуют функциональные зависимости атрибутов первичного ключа от неключевых атрибутов. Данная нормальная форма — это модификация третьей нормальной формы. Таблица может находиться в 3NF, но не в BCNF, только в одном случае: если она имеет, помимо первичного ключа, ещё по крайней мере один составной возможный ключ, и по крайней мере один из атрибутов таблицы входит и в первичный, и в возможный ключи. Такое бывает достаточно редко, в остальном 3NF и BCNF эквивалентны.

Четвёртая нормальная форма (4NF).Таблица находится в 4NF, если она находится в BCNF и не содержит нетривиальных многозначных зависимостей. Многозначная зависимость не является функциональной, она существует в том случае, когда из факта, что в таблице содержится некоторая строка X, следует, что в таблице обязательно существует некоторая определённая строка Y.

Пятая нормальная форма (5NF).Таблица находится в 5NF, если она находится в 4NF и любая многозначная зависимость соединения в ней является тривиальной.

Основываясь на концептуальной модели, создаем следующие сущности: Рабочие, Праздники, Отпуска, Статус, Выходные, Профессия, Сохраненные графики, Месяцы, Графики, Типы дней.

Сущность «Сохраненные\_графики» имеет следующие атрибуты:

ID, NameGraphic, DateOfSave, LinkMonthNum, YearNum.

Сущность «Рабочие» имеет следующие атрибуты: ID, WokerName, TabNumber, LinkProfession, LinkStatus.

Сущность «Выходные» имеет атрибуты: ID, DateCelebrate, LinkName.

Таким образом в данной базе данных отношения находятся в третьей нормальной форме, т.к. все записи атомарные, значения одного атрибута одного и того же типа, порядок следования атрибутов в таблице не существенен, во всех отношениях первичный ключ состоит из одного атрибута, в отношениях нет транзитивных зависимостей.

**5. Основная часть**



На данной блок схеме представлен общий алгоритм составления графика учета рабочего времени. Для более детального изучения алгоритма можно посмотреть приложение Б в котором помещен исходный код с подробными комментариями. При запуске приложение проверяет, существует ли файл базы данных, если он не найден, происходит динамическое создание структуры базы данных с заполнением первоначальных значений в поля.

**6.** **SQL запросы**

Для базы данных «График учета рабочего времени на шахте им. Т. Кузембаева в цехе «Автоматика» » выполнены следующие запросы:

* Подсчет количества записей в таблице

Function SQL\_CountRows(table:string;Connection:TADOConnection):integer;

var

str:\_Recordset;

request:string;

begin

request := 'SELECT COUNT(ID) FROM ' + table;

STR := Connection.Execute(request);

result := integer(str.Fields.Item[0].Value);

end;

где table – передаваемый в функцию параметр – название таблицы;

* Получение значений атрибутов из таблицы «Отпуски», для заполнения массива, хранящего эти значения для последующих вычислений.

request := 'SELECT LinkWoker, DateBegin, Length FROM ОТПУСКИ';

STR := ADOConnection1.Execute(request);

* Получение значений атрибутов из таблицы «Рабочие», для заполнения массива, хранящего эти значения для последующих вычислений

request := 'SELECT WokerName, LinkStatus, TabNumber FROM РАБОЧИЕ';

STR := ADOConnection1.Execute(request);

* Пометка выходных дней в массиве дней месяца из таблицы, в которой хранятся данные о выходных днях «Выходные».

with sql\_CreateQuery(ADOConnection1) do begin

SQL.Text := 'SELECT DateCelebrate FROM Выходные WHERE DateCelebrate BETWEEN :Date1 and :Date2';

with Parameters do begin

ParseSQL(SQL.Text, true);

ParamValues['Date1'] := Date1;

ParamValues['Date2'] := Date2;

end;

Open;

while not(EOF) do begin

MonthInfo[DayOfTheMonth(TDateTime(FieldValues['DateCelebrate']))] := DayCelebrate;

next;

end;

Close;

Free;

end;

* Сохранение названия сохраненного графика, даты сохранения в таблицу «Сохраненные\_графики».

procedure SaveToSavedGraphic(NameBD:string; DateOfSave:TDateTime; MonthNum, YearNum:integer);

begin

with sql\_CreateQuery(ADOConnection1) do begin

SQL.Text := 'INSERT INTO Сохраненные\_графики (NameGraphic, DateOfSave, LinkMonthNum, YearNum) VALUES(:p1, :p2, :p3, :p4)';

with parameters do begin

ParseSQL(SQL.Text, true);

ParamValues['p1'] := NameBD;

ParamValues['p2'] := DateOfSave;

ParamValues['p3'] := MonthNum;

ParamValues['p4'] := YearNum;

end;

ExecSQL();

Close;

Free;

end;

end;

* Процедура сохранения самого графика в таблицу «Графики», с указанием ID сохранения в таблице «Сохраненные\_графики».

procedure SaveToGraphic(IDNum, LinkWoker, LinkTypeOfDay:integer; DateOfCell:TDateTime);

begin

with sql\_CreateQuery(ADOConnection1) do begin

SQL.Text := 'INSERT INTO Графики (LinkSaveId, LinkWoker, LinkTypeOfDay, DateOfCell) VALUES(:p1, :p2, :p3, :p4)';

with parameters do begin

ParseSQL(SQL.Text, true);

ParamValues['p1'] := IDNum;

ParamValues['p2'] := LinkWoker + 1;

ParamValues['p3'] := LinkTypeOfDay;

ParamValues['p4'] := DateOfCell;

end;

ExecSQL();

Close;

Free;

end;

* Получение ID последней добавленной в таблицу строки.

function GetLastID(conn:TADOConnection):integer;

begin

result := 0;

with sql\_CreateQuery(conn) do begin

SQL.Text := 'SELECT @@IDENTITY';

Open();

if Not (eof) then begin

result := Fields.Fields[0].Value;

end;

Close();

Free;

end;

end;

request := 'CREATE TABLE Профессия ' + '( ' +

ID IDENTITY (1, 1) CONSTRAINT PK\_Profession\_ID PRIMARY KEY, '+

nameProfession CHAR(255) NOT NULL UNIQUE '+ ')';

Conn.Execute(request);

request := 'CREATE TABLE Статус ' + '( ' +

' ID IDENTITY (1, 1) CONSTRAINT PK\_Profession\_ID PRIMARY KEY, ' +

nameStatus CHAR(255) NOT NULL UNIQUE '+ ')';

Conn.Execute(request);

request := 'CREATE TABLE Рабочие ' +'( ' +

' ID IDENTITY (1, 1) CONSTRAINT PK\_Woker\_ID PRIMARY KEY, ' +

' WokerName CHAR(128) NOT NULL , ' +

' TabNumber INTEGER , ' +

' LinkProfession INTEGER not null CONSTRAINT LINK\_Profession\_ID ' +

' REFERENCES Профессия(ID) , ' +

' LinkStatus INTEGER not null CONSTRAINT LINK\_Status\_ID ' +

' REFERENCES Статус(ID)' + ')';

Conn.Execute(request);

request := 'CREATE TABLE Отпуски ' + '( ' +

' ID IDENTITY (1, 1) CONSTRAINT PK\_Holidays\_ID PRIMARY KEY , ' +

' LinkWoker INTEGER not null CONSTRAINT LINK\_Woker\_ID ' +

' REFERENCES Рабочие(ID) , ' +

' DateBegin Date , ' +

' Length INTEGER ' + ')';

Conn.Execute(request);

request := 'CREATE TABLE Типы\_дней ' + '( ' +

' ID INTEGER UNIQUE, ' +

' NameOfType CHAR(128) ' + ')';

Conn.Execute(request);

request := 'CREATE TABLE Месяцы ' + '( ' +

' ID INTEGER UNIQUE, ' +

' MonthName CHAR(128) ' + ')';

Conn.Execute(request);

request := 'CREATE TABLE Сохраненные\_графики ' + '( ' +

' ID IDENTITY (1, 1) CONSTRAINT PK\_SavedGraphic\_ID PRIMARY KEY , ' +

' NameGraphic CHAR(128) , ' +

' DateOfSave DATE , ' +

' LinkMonthNum INTEGER not null CONSTRAINT LINK\_Month\_ID ' +

' REFERENCES Месяцы(ID) , ' +

' YearNum CHAR(128) ' + ')';

Conn.Execute(request);

request := 'CREATE TABLE Графики ' + '( ' +

' ID IDENTITY (1, 1) CONSTRAINT PK\_Graphic\_ID PRIMARY KEY, ' +

' LinkSaveID INTEGER not null CONSTRAINT LINK\_SavedGraphics\_ID ' +

' REFERENCES Сохраненные\_графики(ID) , ' +

' DateOfCell DATE , ' +

' LinkWoker INTEGER not null CONSTRAINT LINK\_Woker\_IDN ' +

' REFERENCES Рабочие(ID) , ' +

' LinkTypeOfDay INTEGER not null CONSTRAINT LINK\_TypeOfDay\_ID ' +

' REFERENCES Типы\_дней(ID) ' + ')';

Conn.Execute(request);

request := 'CREATE TABLE Праздники ' + '( ' +

' ID IDENTITY (1, 1) CONSTRAINT PK\_Celebrate\_ID PRIMARY KEY , ' +

' Title CHAR(128) ' + ')';

Conn.Execute(request);

request := 'CREATE TABLE Выходные ' + '( ' +

' ID IDENTITY (1, 1) CONSTRAINT PK\_DateCelebrate\_ID PRIMARY KEY , ' +

' DateCelebrate DATE , ' +

' LinkName INTEGER not null CONSTRAINT LINK\_HOL\_ID' +

' REFERENCES Праздники(ID) ' + ')';

Conn.Execute(request);

//---------------------------------Заполнение полей значениями------------------

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (0, ''Январь '')';

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (1, ''Февраль '')';

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (2, ''Март '')' ;

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (3, ''Апрель '')';

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (4, ''Май '')';

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (5, ''Июнь '')';

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (6, ''Июль '')';

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (7, ''Август '')';

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (8, ''Сентябрь '')';

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (9, ''Октябрь '')';

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (10, ''Ноябрь '')';

Conn.Execute(request);

request := 'INSERT INTO Месяцы(ID, MonthName)'+

' VALUES (11, ''Декабрь '')';

Conn.Execute(request);

//------------------------------------------------------------------------------------/

request := 'INSERT INTO Типы\_дней(ID, NameOfType)'+

' VALUES (0, ''Смена1 '')';

Conn.Execute(request);

request := 'INSERT INTO Типы\_дней(ID, NameOfType)'+

' VALUES (1, ''Смена2 '')';

Conn.Execute(request);

request := 'INSERT INTO Типы\_дней(ID, NameOfType)'+

' VALUES (2, ''Смена3 '')';

Conn.Execute(request);

request := 'INSERT INTO Типы\_дней(ID, NameOfType)'+

' VALUES (3, ''Смена4 '')';

Conn.Execute(request);

request := 'INSERT INTO Типы\_дней(ID, NameOfType)'+

' VALUES (4, ''Нерабочий день '')';

Conn.Execute(request);

request := 'INSERT INTO Типы\_дней(ID, NameOfType)'+

' VALUES (5, ''Отпуск '')';

Conn.Execute(request);

//------------------------------------------------------------------------------------/

request := 'INSERT INTO Профессия(ID, NameProfession)'+

' VALUES (1, ''ПЭС '')';

Conn.Execute(request);

request := 'INSERT INTO Профессия(ID, NameProfession)'+

' VALUES (2, ''ДЭС '')';

Conn.Execute(request);

//------------------------------------------------------------------------------------/

request := 'INSERT INTO Статус(ID, NameStatus)'+

' VALUES (1, ''Вентиляторщик, работающий в выходные '')';

Conn.Execute(request);

request := 'INSERT INTO Статус(ID, NameStatus)'+

' VALUES (2, ''Вентиляторщик, не работающий в выходные '')';

Conn.Execute(request);

request := 'INSERT INTO Статус(ID, NameStatus)'+

' VALUES (3, ''Дежурный '')';

Conn.Execute(request);

request := 'INSERT INTO Статус(ID, NameStatus)'+

' VALUES (4, ''Рабочий'')';

Conn.Execute(request);

//------------------------------------------------------------------------------------/

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (1, ''Забудский С. Ю.'',78704, 1, 1)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (2, ''Ивановский А. В.'',79856, 1, 1)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (3, ''Палагушкин Б. А.'',79485, 1, 1)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (4, ''Девяткин А. М.'',55288, 1, 1)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (5, ''Филонов М. Л.'',107083, 2, 2)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (6, ''Васекин С. А.'',79446, 1, 3)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (7, ''Статилко В. В.'',78715, 1, 3)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (8, ''Васильев А. Д.'',55337, 1, 3)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (9, ''Матейко А. В.'',55681, 1, 3)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (10, ''Чирок А. Н.'',55844, 1, 3)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (11, ''Никулин А. Н.'',67128, 1, 4)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (12, ''Букин Б. И.'',52696, 1, 2)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (13, ''Сибгатулин Р. И.'',104624, 1, 3)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (14, ''Сибгатулин Р. И.'',71763, 1, 3)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (15, ''Денисенко В. В.'',80087, 1, 3)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (16, ''Аборонов В. И.'',55780, 1, 3)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (17, ''Симулин С. Г.'',86149, 1, 4)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (18, ''Майер В. В.'',109859, 1, 4)';

Conn.Execute(request);

request := 'INSERT INTO Рабочие(ID, WokerName, TabNumber, LinkProfession, LinkStatus)'+

' VALUES (19, ''Гетц А. В.'',109525, 1, 1)';

Conn.Execute(request);

//------------------------------------------------------------------------------------/

request := 'INSERT INTO Праздники(ID, Title)'+

' VALUES (1, ''Новый год'')';

Conn.Execute(request);

request := 'INSERT INTO Праздники(ID, Title)'+

' VALUES (2, ''Международный женский день'')';

Conn.Execute(request);

request := 'INSERT INTO Праздники(ID, Title)'+

' VALUES (3, ''Наурыз мейрамы'')';

Conn.Execute(request);

request := 'INSERT INTO Праздники(ID, Title)'+

' VALUES (4, ''Праздник единства народа Казахстана'')';

Conn.Execute(request);

request := 'INSERT INTO Праздники(ID, Title)'+

' VALUES (5, ''День Победы'')';

Conn.Execute(request);

request := 'INSERT INTO Праздники(ID, Title)'+

' VALUES (6, ''День Конституции Республики Казахстан'')';

Conn.Execute(request);

request := 'INSERT INTO Праздники(ID, Title)'+

' VALUES (7, ''День республики'')';

Conn.Execute(request);

request := 'INSERT INTO Праздники(ID, Title)'+

' VALUES (8, ''День Независимости'')';

Conn.Execute(request);

//------------------------------------------------------------------------------------/

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (1, ''01.01.2008'', 1)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (2, ''02.01.2008'', 1)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (3, ''08.03.2008'', 2)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (4, ''22.03.2008'', 3)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (5, ''01.05.2008'', 4)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (6, ''09.05.2008'', 5)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (7, ''30.08.2008'', 6)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (8, ''25.10.2008'', 7)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (9, ''16.12.2008'', 8)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (10, ''17.12.2008'', 8)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (11, ''15.12.2007'', 8)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (12, ''16.12.2007'', 8)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (13, ''17.12.2007'', 8)';

Conn.Execute(request);

request := 'INSERT INTO Выходные(ID, DateCelebrate, LinkName)'+

' VALUES (14, ''08.31.2007'', 2)';

Conn.Execute(request);

//------------------------------------------------------------------------------------/

request := 'INSERT INTO Отпуски(ID, LinkWoker, DateBegin, Length)'+

' VALUES (1, 14, ''08.11.2007'', 26)';

Conn.Execute(request);

request := 'INSERT INTO Отпуски(ID, LinkWoker, DateBegin, Length)'+

' VALUES (2, 16, ''18.12.2007'', 26)';

Conn.Execute(request);

request := 'INSERT INTO Отпуски(ID, LinkWoker, DateBegin, Length)'+

' VALUES (3, 2, ''24.12.2007'', 20)';

Conn.Execute(request);

request := 'INSERT INTO Отпуски(ID, LinkWoker, DateBegin, Length)'+

' VALUES (4, 9, ''01.11.2007'', 10)';

Conn.Execute(request);

request := 'INSERT INTO Отпуски(ID, LinkWoker, DateBegin, Length)'+

' VALUES (5, 6, ''01.11.2007'', 18)';

Conn.Execute(request);

**7.** **Интерфейс программы**

Интерфейс программы состоит из 3 окон. При запуске программы открывается главное окно, представленное на рисунке 7.1

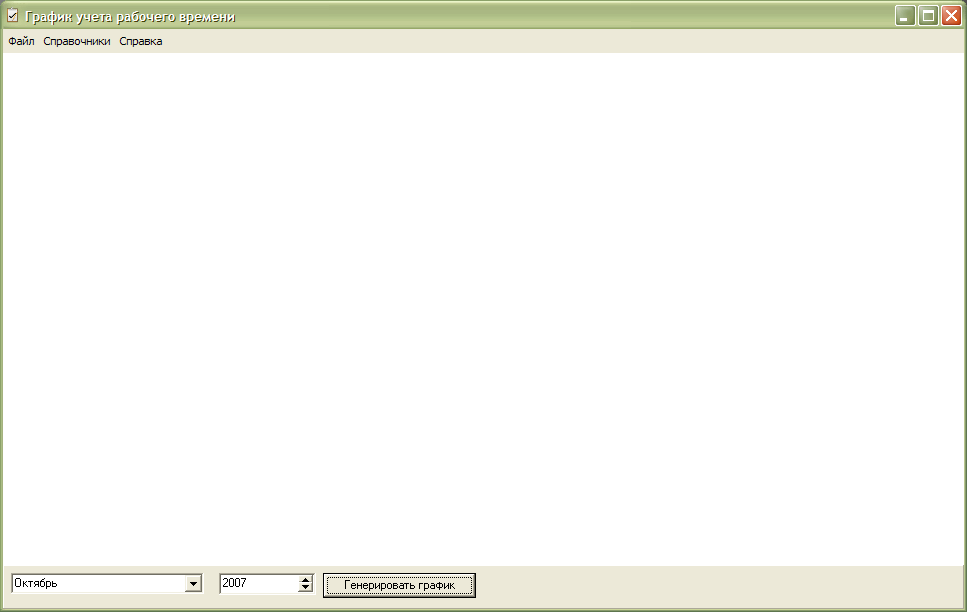


Рис 7.1 – Главное окно

На главной форме приложения находится главное меню, содержащее пункты Файл, Справочник, Справка. Пункты меню распадаются на подменю

Файл -> Открыть, Сохранить, Предварительный просмотр, Печать; Справочники -> Справочник рабочих, Справочник отпусков, Справочник праздников, Справочник названий праздников;

Справка -> О программе.

Команда «Открыть» вызывает окно выбора для загрузки названия и дата ранее сохраненных графиков, представленное на рисунке 7.2

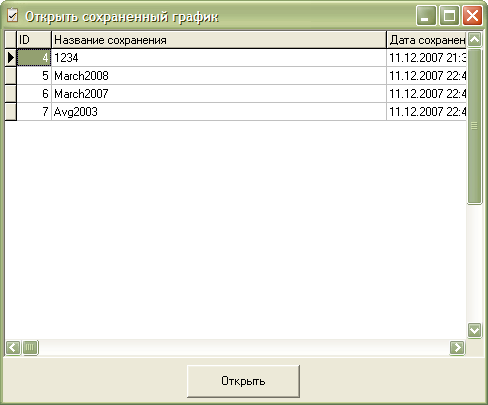


Рисунок 7.2 – окно открытия ранее сохраненных графиков

Команда «Сохранить» вызывается окно сохранения графика, в котором нужно указать ему название. Окно представлено на Рисунке 7.3

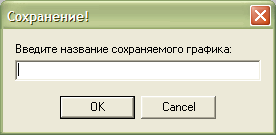


Рисунок 7.3 – Окно сохранения графика

Команда «Предварительный просмотр» рассматривается в разделе «Отчеты».

Команда «Справочник рабочих», вызывает окно содержащее сведения о рабочих, предоставляющее возможность добавления или изменения данных о рабочих. Окно представлено на Рисунке 7.4

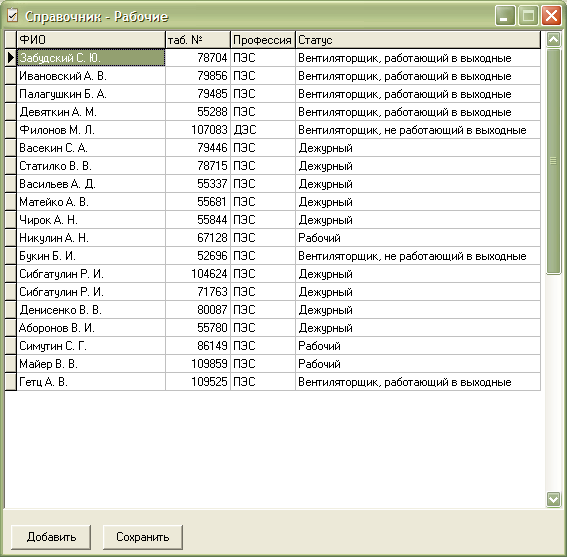


Рисунок 7.4 – Окно справочника рабочих

Команда «Справочник отпусков», вызывает окно содержащее сведения об отпусках рабочих, предоставляющее возможность добавления или изменения данных об отпусках. Окно представлено на Рисунке 7.5



Рисунок 7.5 - Окно справочника отпусков

Команда «Справочник Праздников», вызывает окно содержащее сведения о государственных праздниках, предоставляющее возможность добавления или изменения данных о дате празднования. Окно представлено на Рисунке 7.6

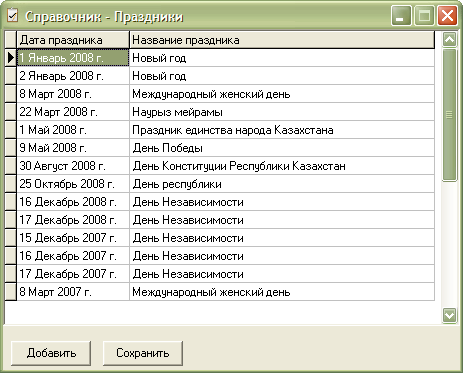


Рисунок 7.7 – Окно справочника праздников

Команда «Справочник названий праздников», вызывает окно содержащее сведения о названия государственных праздников, предоставляющее возможность добавления или изменения данных о названии праздника. Окно представлено на Рисунке 7.8

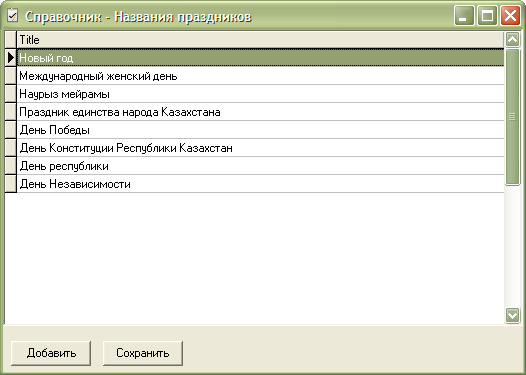


Рисунок 7.8 – Окно Справочника названий праздников

Команда «О программе» вызывает окно, содержащее сведения о разработчике и краткое описание созданного приложения. Окно представлено на Рисунке 7.9

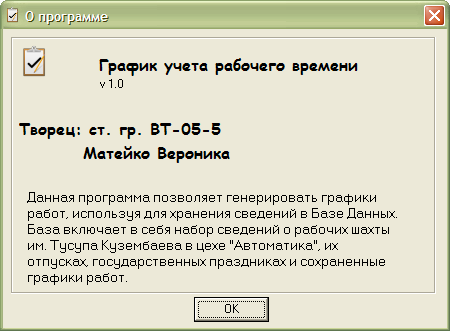


Рисунок 7.9 – Окно «О программе»

Внизу формы расположен выпадающий список для выбора месяца генерируемого графика, а также поле выбора года.

Генерация графика осуществляется по нажатии кнопки «Генерировать график». Результат генерации представлен на Рисунке 7.10

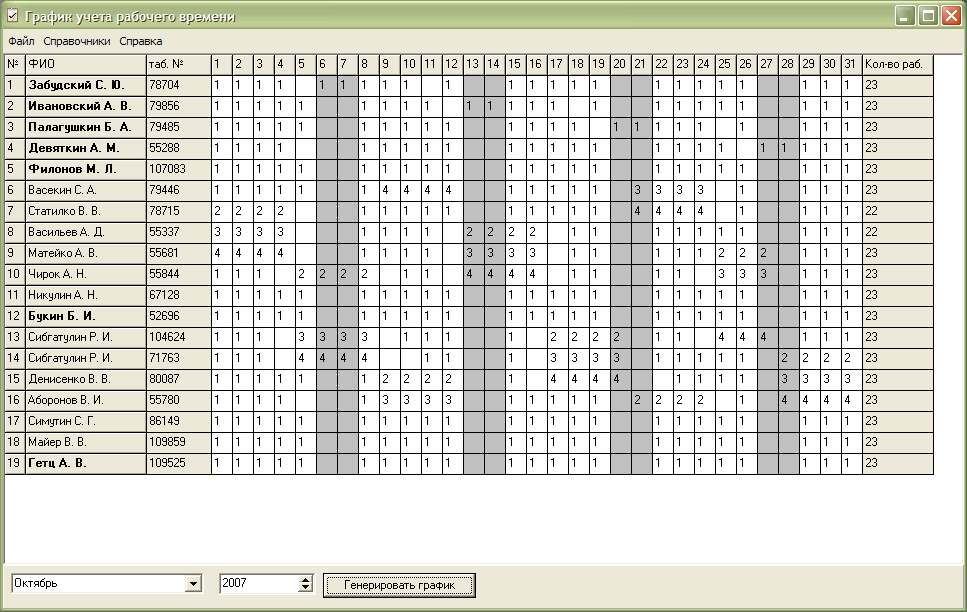


Рисунок 7.10 – результат генерации графика

**8. Отчеты**

В данной программе можно просмотреть и распечатать отчет, содержащий информацию о графике учета рабочего времени на заданный месяц.

Команда «Предварительный просмотр» вызывает окно, отображающее как будет выглядеть график на печати. Окно представлено на Рисунке 8.1

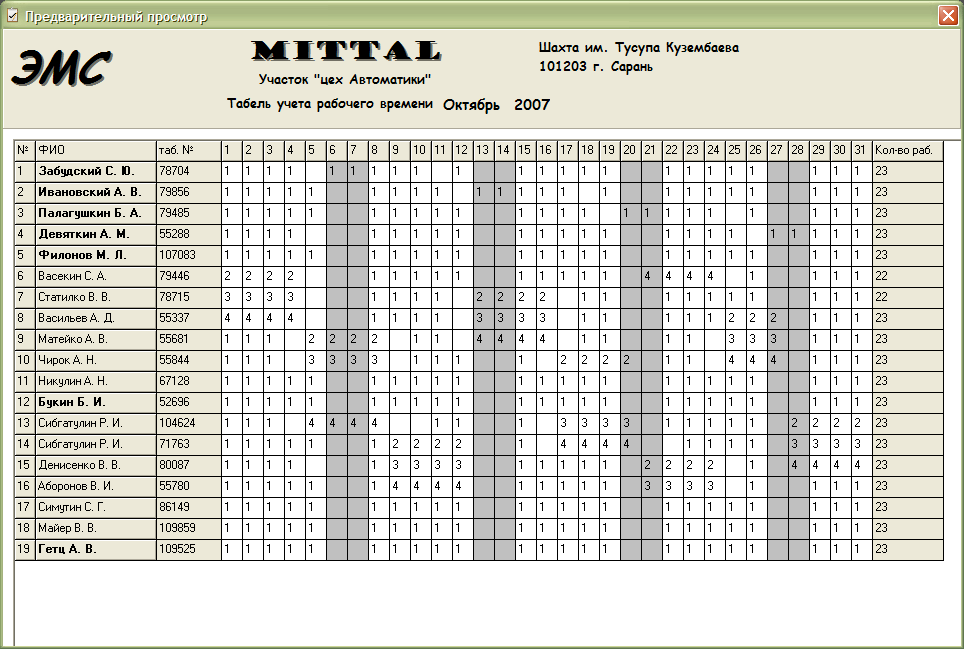


Рисунок 8.1 – Окно предпросмотра печати, отчет программы

Команда «Печать» выводит график указанного месяца и года на принтер.

**Заключение**

В результате выполнения данного курсового проекта было разработано приложение удобное в обращении, позволяющее хранить необходимые сведения в единой Базе Данных, а также генерировать автоматически график работ, что в настоящее время, необходимо для быстрого и точного принятия решений, относительно расписания работ. Вообще, задача составления расписания является достаточно сложной и актуальной. Необходимость в автоматической генерации расписания возникает там, где существует возможность выбора той или иной очередности выполнения работ на производстве, при составлении расписания движения поездов и самолетов, обслуживании клиентов в сервисных системах, составление расписаний занятий в учебных заведениях и т.д.

После апробирования данного приложения на предприятии можно судить о том, что в рамках курсовой работы мне удалось создать удовлетворяющую требованиям предприятия автоматическую генерацию графика учета рабочего времени, с возможностью хранения посредством Базы Данных

**Список использованных источников**

1. Терлецкая А. М. – лекции по дисциплине «Технология и разработка баз данных»;
2. П. Дарахвелидзе, Е. Марков. Разработка web-служб средствами Delphi – СПб: БХВ-Петербург, 2003
3. http://club.shelek.com/viewart.php?id=177