ФЕДЕРАЛЬНОЕ МИНИСТЕРСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

КАФЕДРА ЭВМ и С

Семестровая работа

по курсу «Построение СУБД »

Тема: «Интернет-провайдер»

Выполнил:

студент группы

ИВТ-460

Юров Д.И.

Проверил:

Конченков В.И.

Волгоград, 2015

1. Задание

Разработать Систему управления базы данных «Интернет-провайдер» с графическим интерфейсом. СУБД должна содержать следующие таблицы:

Таблица 1 – Описание сущности пользователь

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Наименование** | **Тип** | **Длина** | **Диапазон значений** | **Вероятность существования** | **Коэффициент повторения** |
| Код пользователя | Числовой | - | 0-9 | 1 | 0 |
| Фамилия | Текст | 30 | А-Я | 1 | 1 |
| Имя | Текст | 15 | А-Я | 1 | 1 |
| Отчество | Текст | 30 | А-Я | 1 | 1 |
| Паспортные данные | Числовой | - | 0-9 | 1 | 0 |
| Адрес | Текст | 50 | А-Я, 0-9 | 1 | 0 |
| Контактный телефон | Числовой | - | 0-9 | 0 | 0 |

Таблица 2 – Описание атрибутов сущности договор

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Наименование** | **Тип** | **Длина** | **Диапазон значений** | **Вероятность существования** | **Коэффициент повторения** |
| Код договора | Числовой | - | 0-9 | 1 | 0 |
| Код пользователя | Числовой | - | 0-9 | 1 | 0 |
| Баланс | Числовой | - | 0-9 | 1 | 1 |
| Логин | Текст | 15 | А-Я, 0-9 | 1 | 0 |
| Пароль | Текст | 15 | А-Я, 0-9 | 1 | 1 |
| Код дома | Числовой | - | 0-9 | 1 | 1 |
| Номер квартиры | Числовой | - | 0-9 | 1 | 0 |
| Дата подключения | Дата | \* | \* | 1 | 1 |
| Код сотрудника | Числовой | - | 0-9 | 1 | 1 |

Таблица 3 – Описание атрибутов сущности тариф

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Наименование** | **Тип** | **Длина** | **Диапазон значений** | **Вероятность существования** | **Коэффициент повторения** |
| Код тарифа | Числовой | - | 0-9 | 1 | 0 |
| Наименование тарифа | Текст | 15 | А-Я, 0-9 | 1 | 0 |
| Описание | Текст | 50 | А-Я, 0-9 | 1 | 0 |
| Срок действия  (дата окончания действия) | Дата | \* | \* | 1 | 1 |
| Стоимость | Числовой | - | 0-9 | 1 | 1 |
| Примечания | Текст | 30 | А-Я, 0-9 | 1 | 0 |

Таблица 4 – Описание атрибутов сущности покрытие

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Наименование** | **Тип** | **Длина** | **Диапазон значений** | **Вероятность существования** | **Коэффициент повторения** |
| Код дома | Числовой | - | 0-9 | 1 | 0 |
| Район | Текст | 30 | А-Я, 0-9 | 1 | 1 |
| Улица | Текст | 30 | А-Я, 0-9 | 1 | 1 |
| Номер дома | Числовой | - | 0-9 | 1 | 1 |

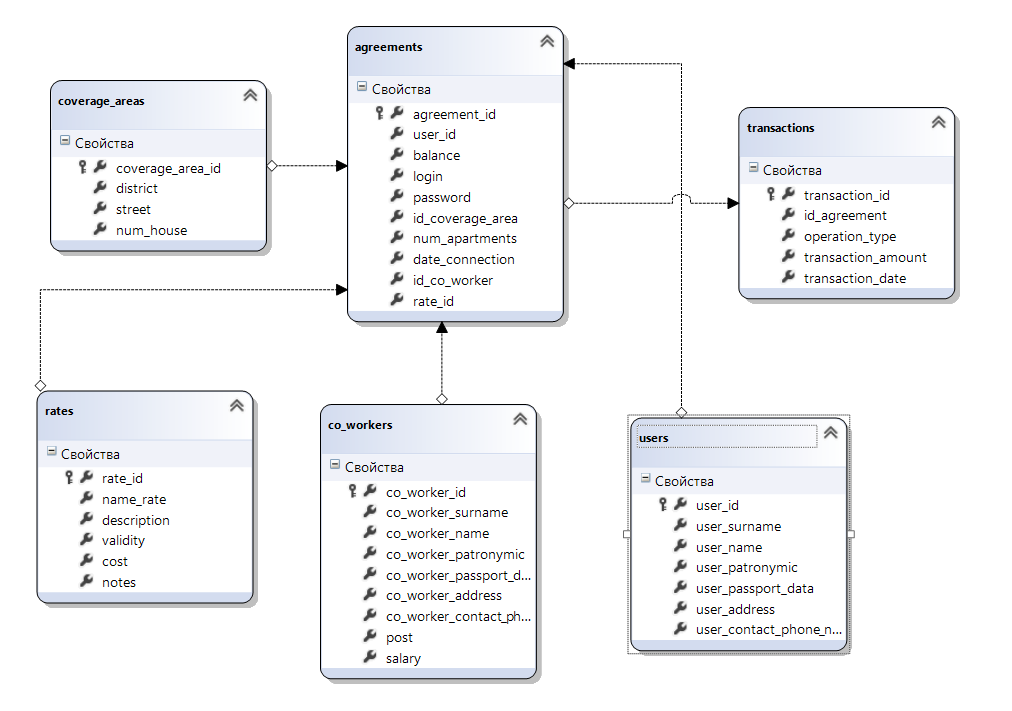
Таблица 5 – Описание атрибутов сущности денежная операция

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Наименование** | **Тип** | **Длина** | **Диапазон значений** | **Вероятность существования** | **Коэффициент повторения** |
| Код денежной операции | Числовой | - | 0-9 | 1 | 0 |
| Код договора | Числовой | - | 0-9 | 1 | 1 |
| Вид операции | Текст | 30 | А-Я, 0-9 | 1 | 1 |
| Сумма операции | Числовой | - | 0-9 | 1 | 1 |
| Дата транзакции | Дата | \* | \* | 1 | 1 |

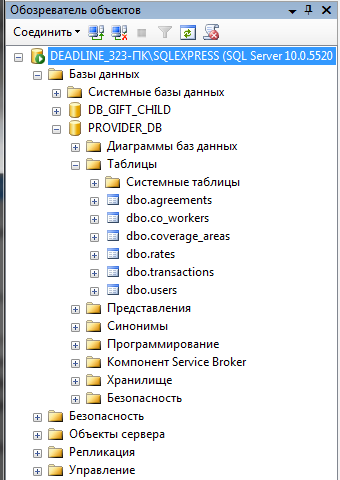
Таблица 6 – Описание атрибутов сотрудник

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Наименование** | **Тип** | **Длина** | **Диапазон значений** | **Вероятность существования** | **Коэффициент повторения** |
| Код сотрудника | Числовой | - | 0-9 | 1 | 0 |
| Фамилия | Текст | 30 | А-Я | 1 | 1 |
| Имя | Текст | 15 | А-Я | 1 | 1 |
| Отчество | Текст | 30 | А-Я | 1 | 1 |
| Паспортные данные | Числовой | - | 0-9 | 1 | 0 |
| Адрес | Текст | 50 | А-Я, 0-9 | 1 | 0 |
| Контактный телефон | Числовой | - | 0-9 | 1 | 0 |
| Должность | Текст | 50 | А-Я, 0-9 | 1 | 1 |
| Зарплата | Числовой | - | 0-9 | 1 | 1 |

1. Проектирование диаграмм



1. Создание и подключение Файла базы данных



1. Настройки подключения

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<configSections>

</configSections>

<connectionStrings>

<add name="Provider\_DB.Properties.Settings.PROVIDER\_DBConnectionString"

connectionString="Data Source=DEADLINE\_323-ПК\SQLEXPRESS;Initial Catalog=PROVIDER\_DB;Integrated Security=True"

providerName="System.Data.SqlClient" />

</connectionStrings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />

</startup>

</configuration>

1. Пользовательский интерфейс

Интерфейс программы реализован так, что пользователь может работать непосредственно со всеми таблицами базы данных, а именно добавлять, изменять и удалять любые записи в таблицах. Помимо этого программа позволяет выполнить шесть самых популярных запросов к базе данных, среди которых:

* Просмотр договоров заключённых выбранным сотрудником,
* Проверка пользователя на нахождение его в зоне покрытия по его имени,
* Вывод списка всех денежных операций пользователя,
* Вывод списка денежных операций пользователя за последний месяц,
* Вывод списка договоров заключённых данным сотрудником,
* Вывод списка пользователей подключённых к данному тарифному плану

1. **Результат работы**

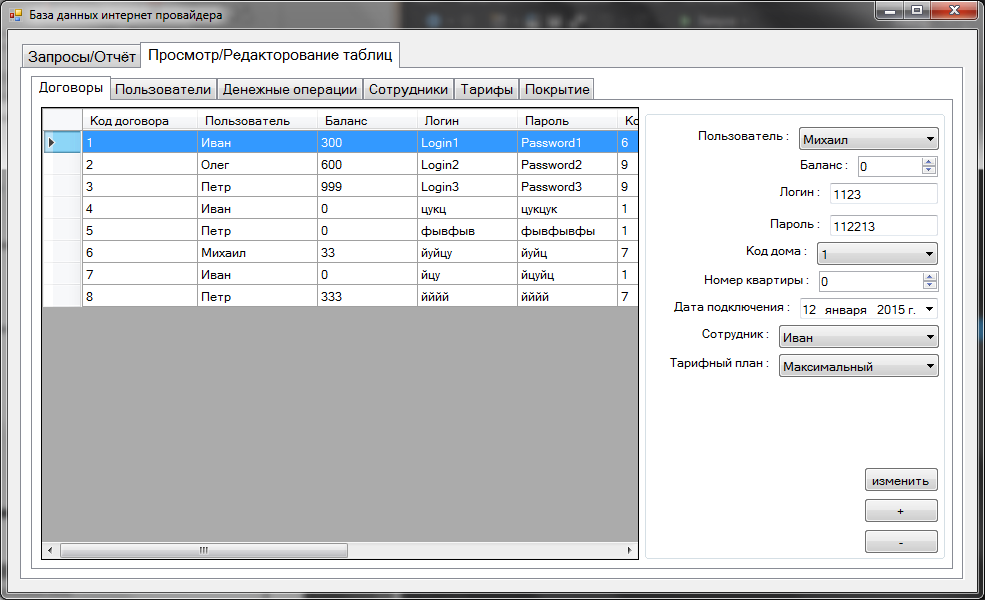


Рисунок 1. Редактирование данных в таблицах

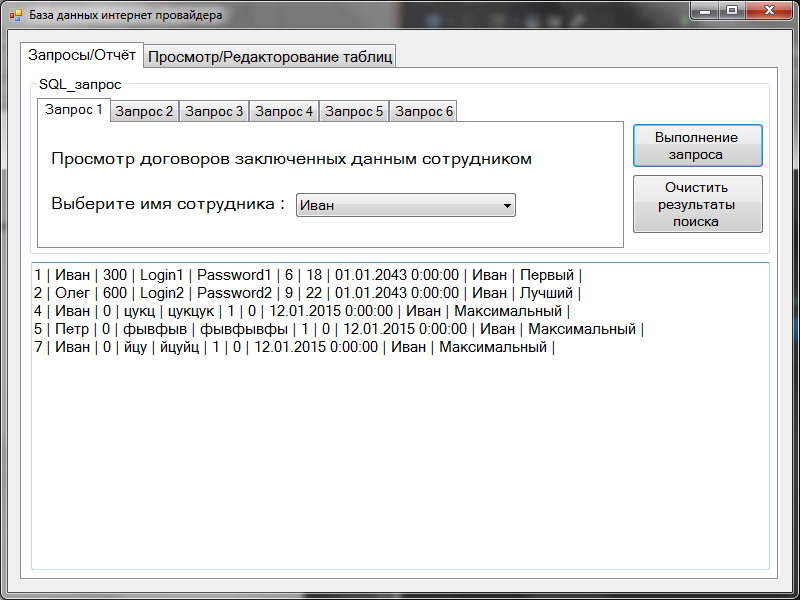


Рисунок 2. Выполнение одного из запросов

1. **Текст запросов на SQL**

|  |  |
| --- | --- |
| Цель запроса | Запрос |
| Просмотр договоров заключённых выбранным сотрудником | *SELECT agreement.agreement\_id, agreement.user\_id, agreement.date\_connection*  *FROM agreement*  *JOIN co\_worker ON co\_worker.co\_worker.id=*  *agreement.* *id\_co\_worker*  *WHERE co\_worker.co\_worker\_name=<имя сотрудника>* |
| Проверка пользователя на нахождение его в зоне покрытия по его имени | *SELECT coverage\_area.coverage\_area\_id, coverage\_area.district,*  *coverage\_area.street, coverage\_area.num\_house*  *FROM coverage\_area*  *JOIN user ON user.user\_address = CONCAT(coverage\_area.district,*  *coverage\_area.street,*  *coverage\_area.num\_house)*  *WHERE user.user\_name =<имя пользователя>* |
| Вывод списка всех денежных операций пользователя | *SELECT transaction.transaction\_id, transaction.id\_agreement, transaction.operation\_type, transaction.transaction\_amount,*  *transaction.transaction\_date*  *FROM transaction*  *JOIN agreement ON agreement.agreement\_id = transaction.id\_agreement*  *JOIN user ON user.user\_id = agreement.user\_id*  *WHERE user.user\_name=<имя пользователя>* |
| Вывод списка денежных операций пользователя за последний месяц | *SELECT transaction.transaction\_id, transaction.id\_agreement, transaction.operation\_type, transaction.transaction\_amount,*  *transaction.transaction\_date*  *FROM transaction*  *JOIN agreement ON agreement.agreement\_id = transaction.id\_agreement*  *JOIN user ON user.user\_id = agreement.user\_id*  *WHERE (transaction.transaction\_date >= <Начало текущего месяца(2000.01.01)> AND user.user\_name=<имя пользователя>)* |
| Вывод списка договоров заключённых данным сотрудником | *SELECT agreement.agreement\_id, agreement.user\_id, agreement.login, agreement.date\_connection*  *FROM agreement*  *JOIN co\_worker ON co\_worker.co\_worker\_id = agreement.id\_co\_worker*  *WHERE co\_worker.co\_worker\_name=<имя сотрудника>* |
| Вывод списка пользователей подключённых к данному тарифному плану | *SELECT user.user\_id, user.user\_surname, user.user\_name,*  *FROM user*  *JOIN agreement ON user.user\_id = agreement.user\_id*  *JOIN rate ON agreement.id\_rate = rate.rate\_id*  *WHERE rate.name\_rate =<название тарифного плана>* |

1. Код основных методов на C#

private void tabMainChanged()

{

switch (tabMain.SelectedIndex)

{

case 0:

tabQueryChanged();

break;

case 1:

tabTablesChanged();

break;

}

}

private void fillAgreements()

{

// Перезаполним ComboBox-ы

id\_user.Items.Clear();

id\_coverage\_area.Items.Clear();

id\_co\_worker.Items.Clear();

id\_rate.Items.Clear();

var query1 = dc.users;

foreach (var item in query1) id\_user.Items.Add(item.user\_name.ToString());

var query2 = dc.coverage\_areas;

foreach (var item in query2) id\_coverage\_area.Items.Add(item.coverage\_area\_id.ToString());

var query3 = dc.co\_workers;

foreach (var item in query3) id\_co\_worker.Items.Add(item.co\_worker\_name.ToString());

var query4 = dc.rates;

foreach (var item in query4) id\_rate.Items.Add(item.name\_rate.ToString());

if (id\_user.Items.Count > 0) id\_user.SelectedIndex = 0;

if (id\_coverage\_area.Items.Count > 0) id\_coverage\_area.SelectedIndex = 0;

if (id\_co\_worker.Items.Count > 0) id\_co\_worker.SelectedIndex = 0;

if (id\_rate.Items.Count > 0) id\_rate.SelectedIndex = 0;

// Перезаполним таблицу

int countRow = agreements\_list.Rows.Count;

for (int i = 0; i < countRow; i++)

{

agreements\_list.Rows.RemoveAt(0);

}

var query5 = dc.agreements;

int numRow = 0;

foreach (var item in query5)

{

agreements\_list.Rows.Add();

agreements\_list.Rows[numRow].Cells[0].Value = item.agreement\_id.ToString();

agreements\_list.Rows[numRow].Cells[1].Value = item.users.user\_name.ToString();

agreements\_list.Rows[numRow].Cells[2].Value = item.balance.ToString();

agreements\_list.Rows[numRow].Cells[3].Value = item.login.ToString();

agreements\_list.Rows[numRow].Cells[4].Value = item.password.ToString();

agreements\_list.Rows[numRow].Cells[5].Value = item.id\_coverage\_area.ToString();

agreements\_list.Rows[numRow].Cells[6].Value = item.num\_apartment.ToString();

agreements\_list.Rows[numRow].Cells[7].Value = item.date\_connection.ToString();

agreements\_list.Rows[numRow].Cells[8].Value = item.co\_workers.co\_worker\_name.ToString() + " " +

item.co\_workers.co\_worker\_surname.ToString();

agreements\_list.Rows[numRow].Cells[9].Value = item.rates.name\_rate.ToString();

numRow++;

}

}

private void btn\_agreement\_add\_Click(object sender, EventArgs e)

{

var Agr = new agreements()

{

id\_user = dc.users.FirstOrDefault(u => u.user\_name == id\_user.SelectedItem.ToString()).user\_id,

balance = Int32.Parse(balance.Value.ToString()),

login = login.Text,

password = password.Text,

id\_coverage\_area = Int32.Parse(id\_coverage\_area.SelectedItem.ToString()),

num\_apartment = Int32.Parse(num\_house.Value.ToString()),

date\_connection = Convert.ToDateTime(date\_connection.Text),

id\_co\_worker = dc.co\_workers.FirstOrDefault(c => c.co\_worker\_name == id\_co\_worker.SelectedItem.ToString()).co\_worker\_id,

id\_rate = dc.rates.FirstOrDefault(r => r.name\_rate == id\_rate.SelectedItem.ToString()).rate\_id

};

dc.agreements.InsertOnSubmit(Agr);

dc.SubmitChanges();

fillAgreements();

}

private void btn\_query\_exec\_Click(object sender, EventArgs e)

{

ResultsList.Text = "";

switch (tabQuery.SelectedIndex)

{

case 0:

var query = from a in dc.agreements join c in dc.co\_workers

on a.id\_co\_worker equals c.co\_worker\_id

where c.co\_worker\_name == (comboBox\_Query1.SelectedItem.ToString())

select a;

string str = "";

foreach (var item in query)

{

str += item.agreement\_id.ToString() + " | " +

item.users.user\_name.ToString() + " | " +

item.balance.ToString() + " | " +

item.login.ToString() + " | " +

item.password.ToString() + " | " +

item.id\_coverage\_area.ToString() + " | " +

item.num\_apartment.ToString() + " | " +

item.date\_connection.ToString() + " | " +

item.co\_workers.co\_worker\_name.ToString() + " | " +

item.rates.name\_rate.ToString() + " | ";

str += "\r\n";

}

ResultsList.Text = str;

break;

case 1:

break;

}

}

// происходит когда элемент стает активным

private void user\_address\_Enter(object sender, EventArgs e)

{

if (user\_address.Text == "РайонУлица№дома << формат")

{

user\_address.Text = null;

user\_address.ForeColor = Color.Black;

}

}

// происходит когда элемент стает НЕактивным

private void user\_address\_Leave(object sender, EventArgs e)

{

if (user\_address.Text == "")

{

user\_address.Text = "РайонУлица№дома << формат";

user\_address.ForeColor = Color.Gray;

}

}