## Правила построения моделей

## Типы связи

|  |  |
| --- | --- |
| **один-к-одному** | один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). |
| **один-ко-многим** | один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). |
| **много-ко-многим** | каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. |

## Класс принадлежности

|  |  |
| --- | --- |
| Н | экземпляр одной сущности может быть связан с одним или несколькими экземплярами другой сущности, а может быть, и не связан ни с одним экземпляром |
| О | экземпляр одной сущности обязан быть связан не менее чем с одним экземпляром другой сущности |

## Проектирование концептуальной модели

состоит в построении реляционных отношений и их первичных ключей на основе анализа ER – диаграмм и применении следующих правил:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Степень связи | Принадлежность | Правило |
| 1 | 1:1 | О-О | Если степень связи 1:1 и класс принадлежности О –О то требуется 1 отношение, ключом которого может быть ключ любой сущности |
| 2 | 1:1 | О-Н | Если степень связи 1:1 и класс принадлежности О –Н, то требуется построить 2 отношения. Каждому отношению соответствует 1 сущность, ключ является ключом соответствующей сущности, ключ второй сущности добавляется в качестве атрибута в первое отношение (это внешний ключ) |
| 3 | 1:1 | Н –Н | Если степень связи 1:1 и класс принадлежности Н –Н, то требуется построить 3 отношения: по одному на каждую сущность с соответствующими ключами и связующего отношения, ключом которого будет комбинация ключей двух сущностей. |
| 4 | 1:N | \* –O | Если степень связи 1:N и класс принадлежности \* –O (Н-О/О-О), то достаточным является использование 2х отношений, соответствующих сущностям, при чем ключами будут ключи сущностей, а дополнительно ключ первой сущности должен быть атрибутом второго отношения. |
| 5 | 1:N | \* –Н | Если степень связи 1:N и класс принадлежности \* –Н (Н-Н/О-Н), то смотри правило 3) |
| 6 | M:N | \* | Если степень связи M:N и класс принадлежности не учитывай, то смотри правило 3) |

# 3ий вариант 1ая технология – Соколова

## БД «Телефонный справочник»

В БД следует хранить информацию о номерах телефонов

* физических лиц (ФИО, адрес, номер телефона), а также
* адреса различных организаций городов.
* Город имеет
  + телефонный код,
  + название и
  + находится на территории определенной страны.
* Организация характеризуется
  + названием,
  + адресом и
  + телефоном.
* Каждый телефон принадлежит определенной рубрике:
  + срочные услуги,
  + общеобразовательные учреждения,
  + ВУЗы,
  + авиакомпании,
  + аптеки,
  + магазины,
  + больницы,
  + поликлиники и т.д.
* Организации могут иметь несколько телефонов.

Необходимо предоставить возможность поиска информации в базе:

* для физических лиц – по номеру телефона, фамилии или адресу;
* для организаций – по названию или телефону.

Необходимо также выдавать следующие выходные документы:

* Телефонные коды городов, отсортированные по алфавиту.
* Телефоны физических лиц города, отсортированные по фамилии.
* Телефоны всех организаций городов, сгруппированные по городам и рубрикам.
* Телефоны организаций города определенной рубрики.

# ПОСТРОЕНИЕ ОБЪЕКТНОЙ МОДЕЛИ

## 07.08.17 Диаграмма прецедентов

Под ведением списка городов подразумевается уже ведение телефонного справочника кодов городов

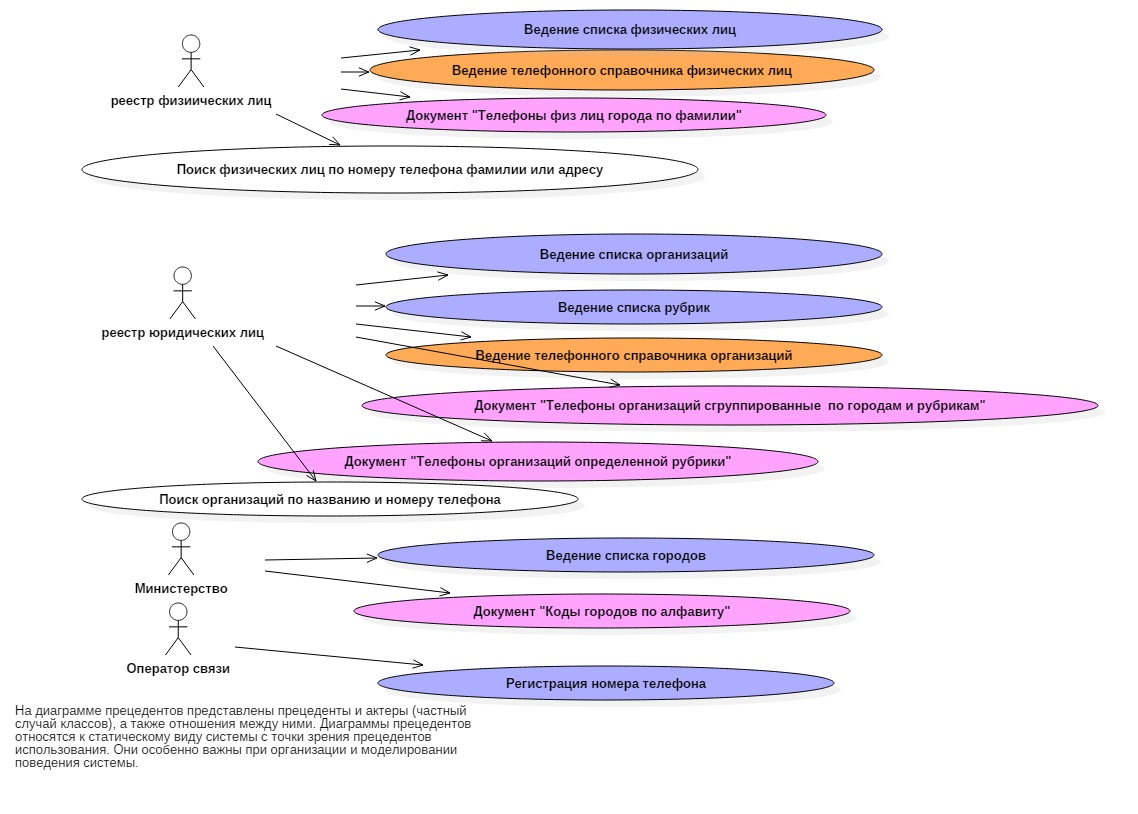


Диаграмма прецедентов

## 07.08.17 Диаграмма классов (пакет «Сущности БД»)

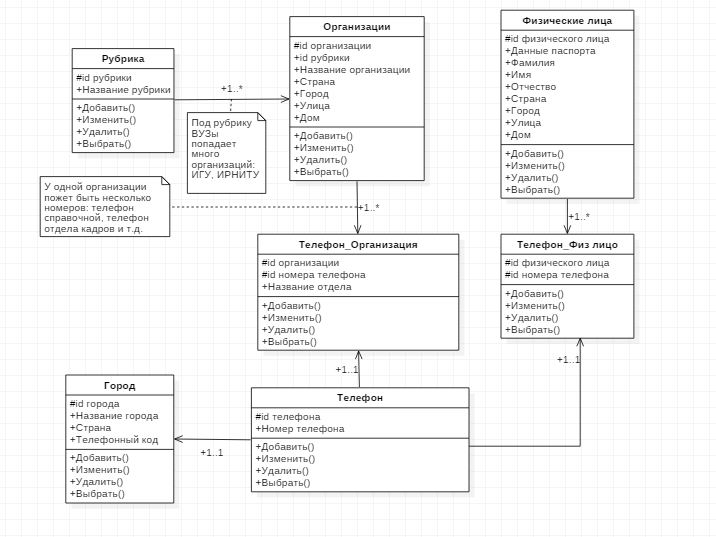


Диаграмма классов (пакет «Сущности БД»)

## Диаграмма классов (пакет «Интерфейс приложения»)

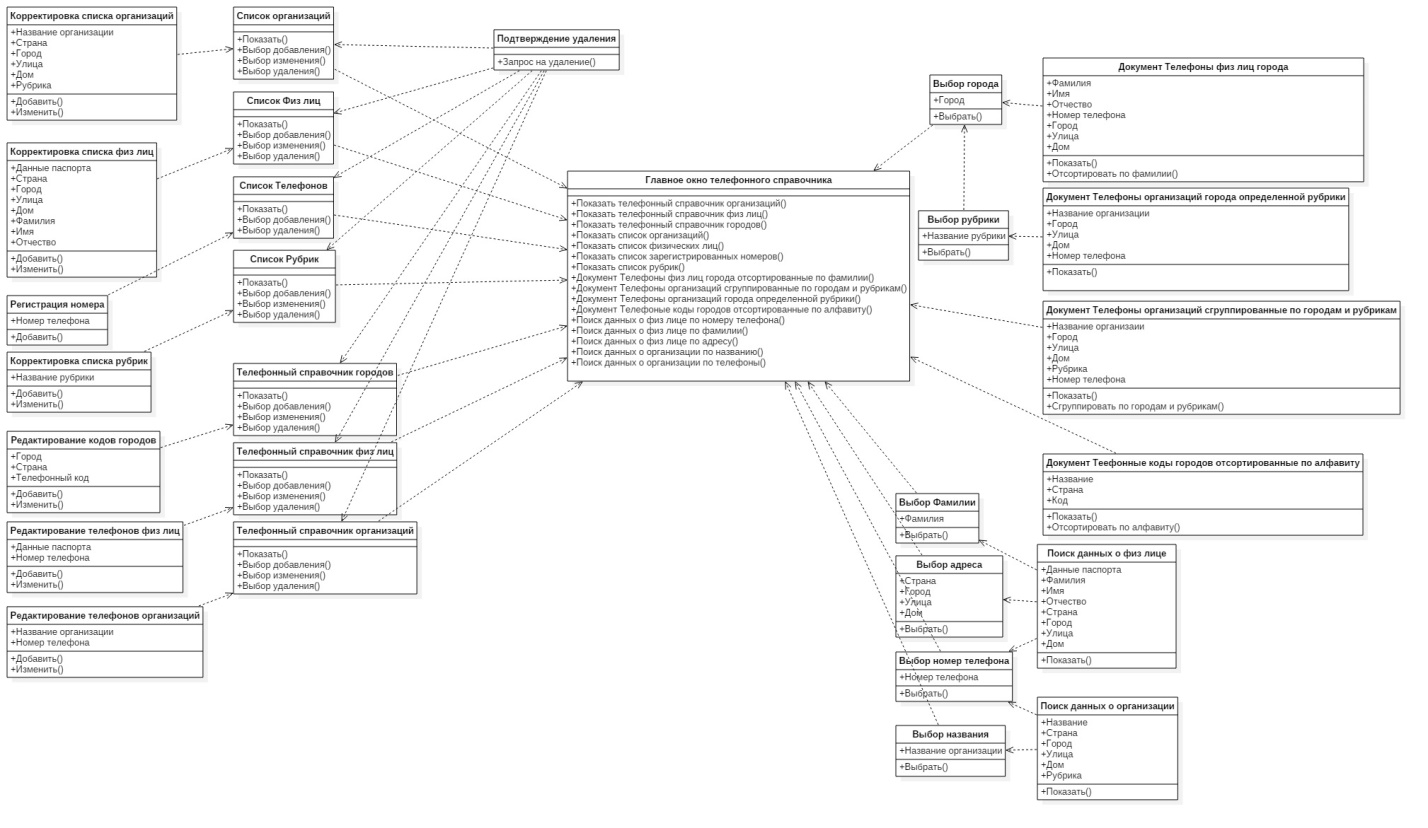
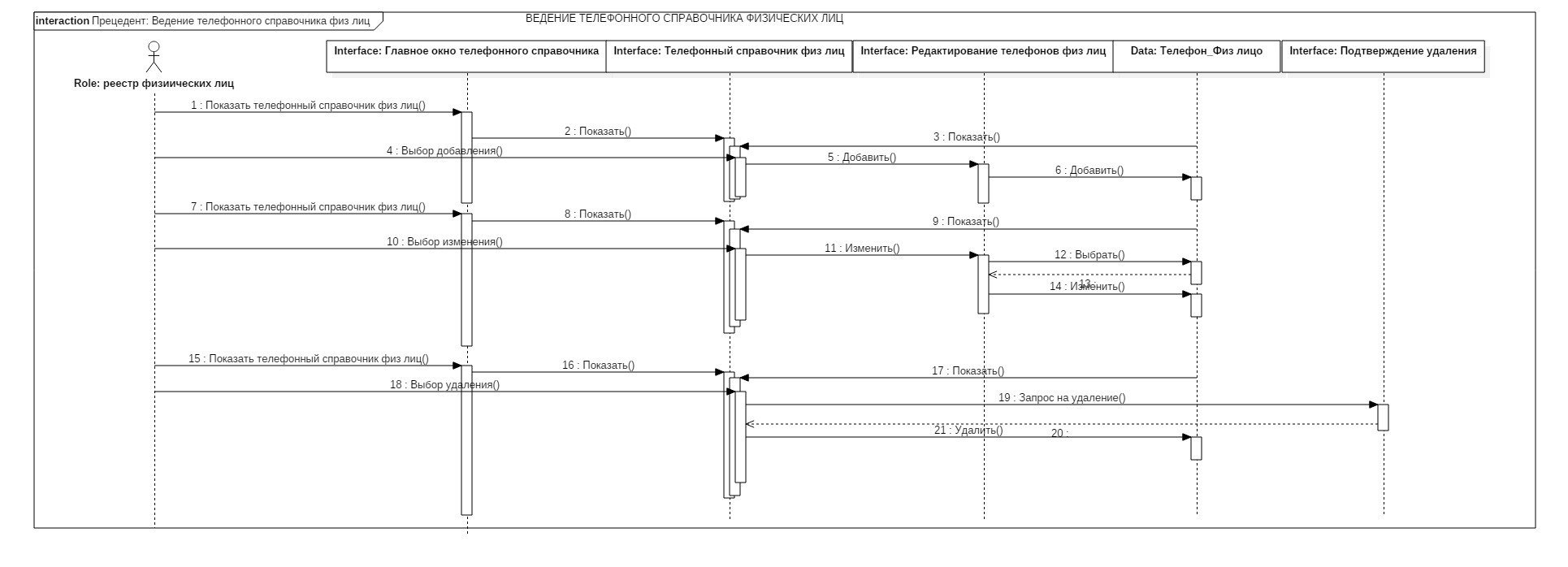


Диаграмма классов (пакет «Интерфейс приложения»)

## 09.08.17 Одобрено Диаграмма последовательностей для прецедента «Ведение телефонного справочника физ лиц»

Диаграмма последовательностей для прецедента «Ведение телефонного справочника физ лиц»



## 090817 Одобрено Диаграмма последовательностей для прецедента «Ведение телефонного справочника организаций»

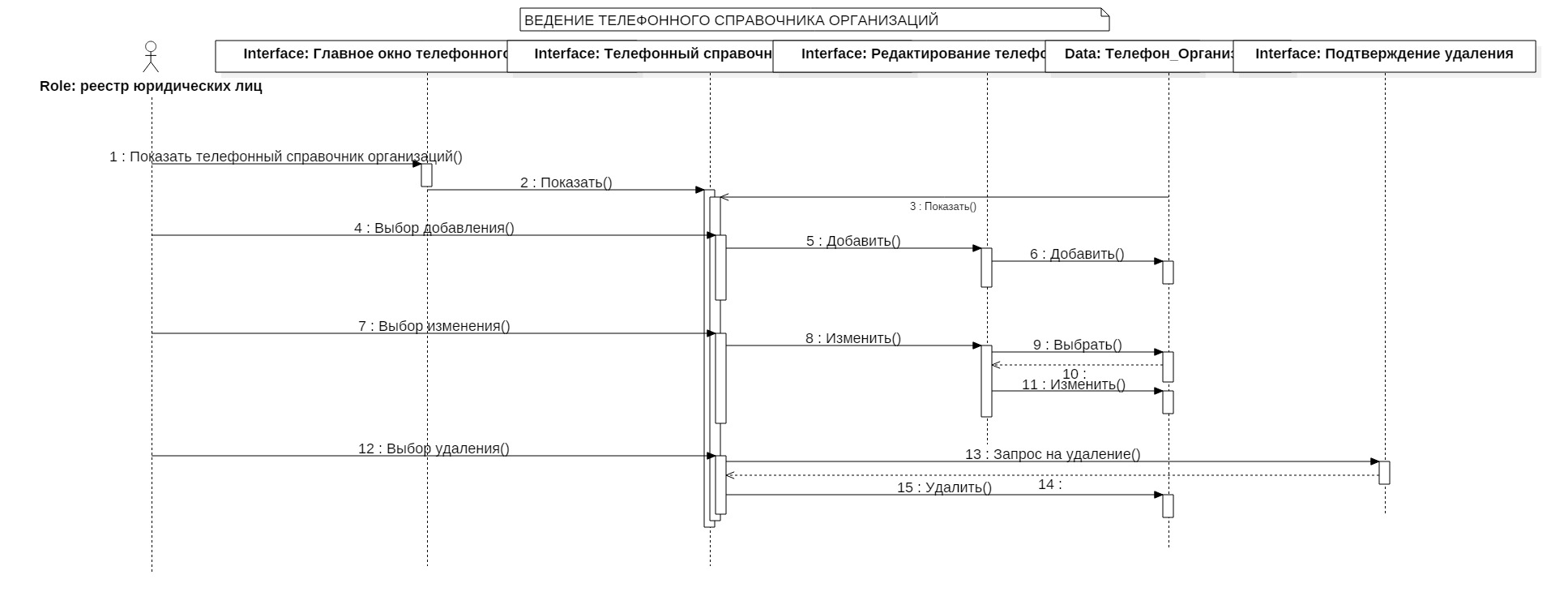


Диаграмма последовательностей для прецедента «Ведение телефонного справочника организаций»

## 090817 Одобрено Диаграмма последовательностей для прецедента «Ведение справочника кодов городов»

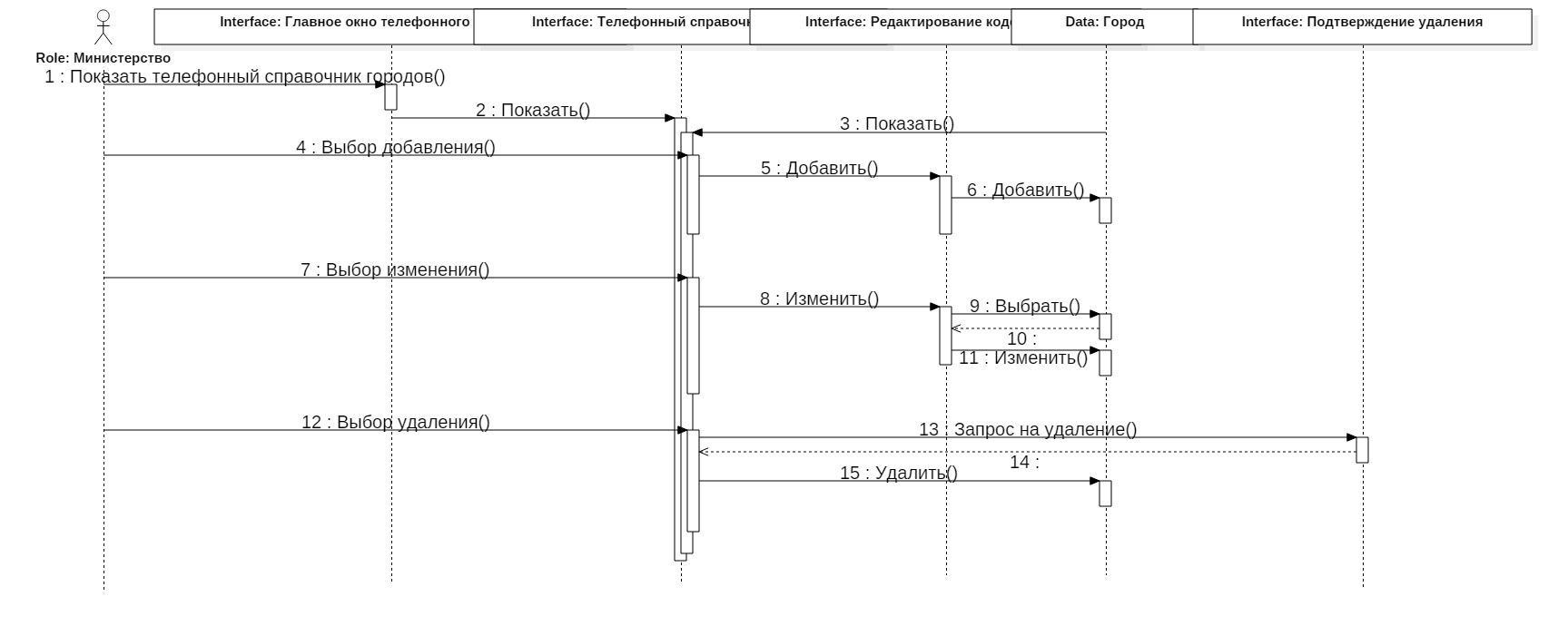


Диаграмма последовательностей для прецедента «Ведение справочника кодов городов»

## 090817 Одобрено Диаграмма последовательностей для прецедента «Документ : телефоны физ лиц города отсортированные по фамилии»

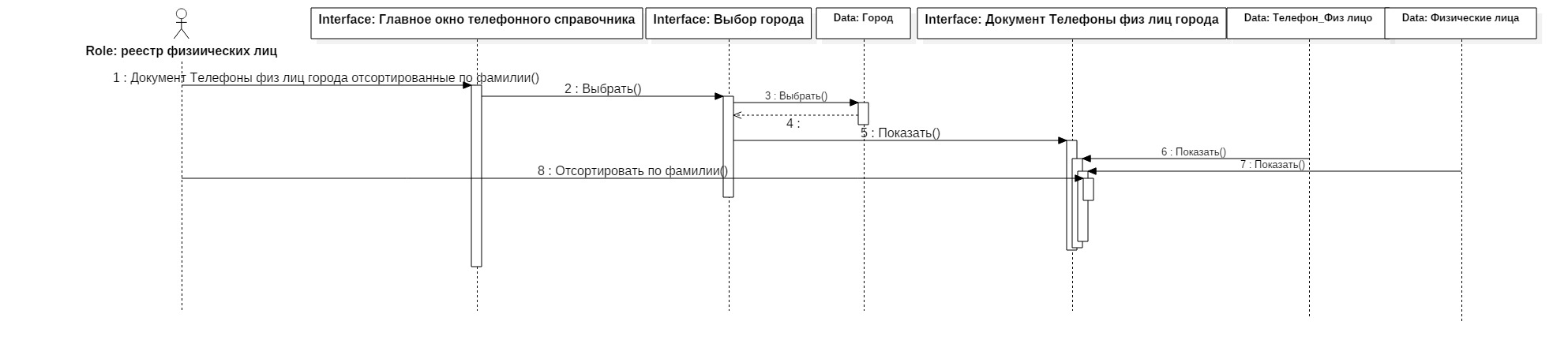


Диаграмма последовательностей для прецедента «Документ : телефоны физ лиц города отсортированные по фамилии»

## 090817 Одобрено Диаграмма последовательностей для прецедента «Документ : телефоны организаций сгруппированные по городам и рубрикам»

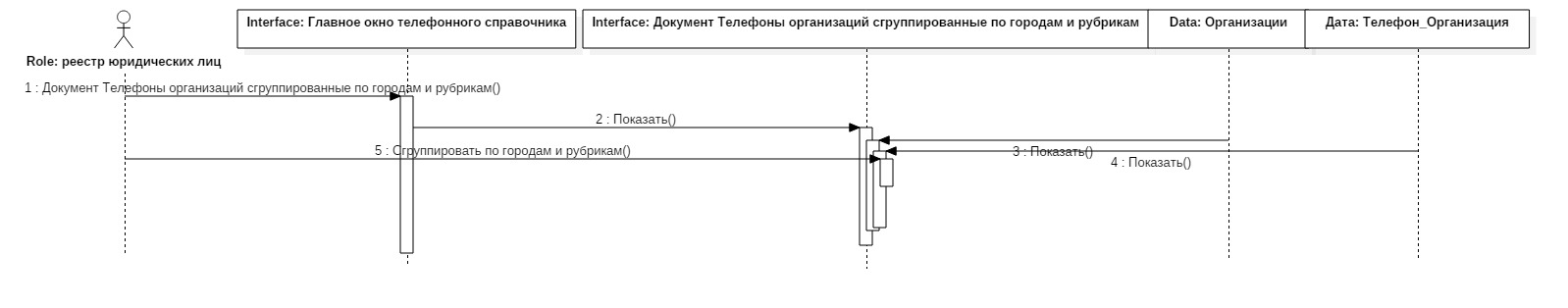


Диаграмма последовательностей для прецедента «Документ : телефоны организаций сгруппированные по городам и рубрикам»

## 090817 Диаграмма последовательностей для прецедента «Документ : телефоны организаций города определенной рубрики»

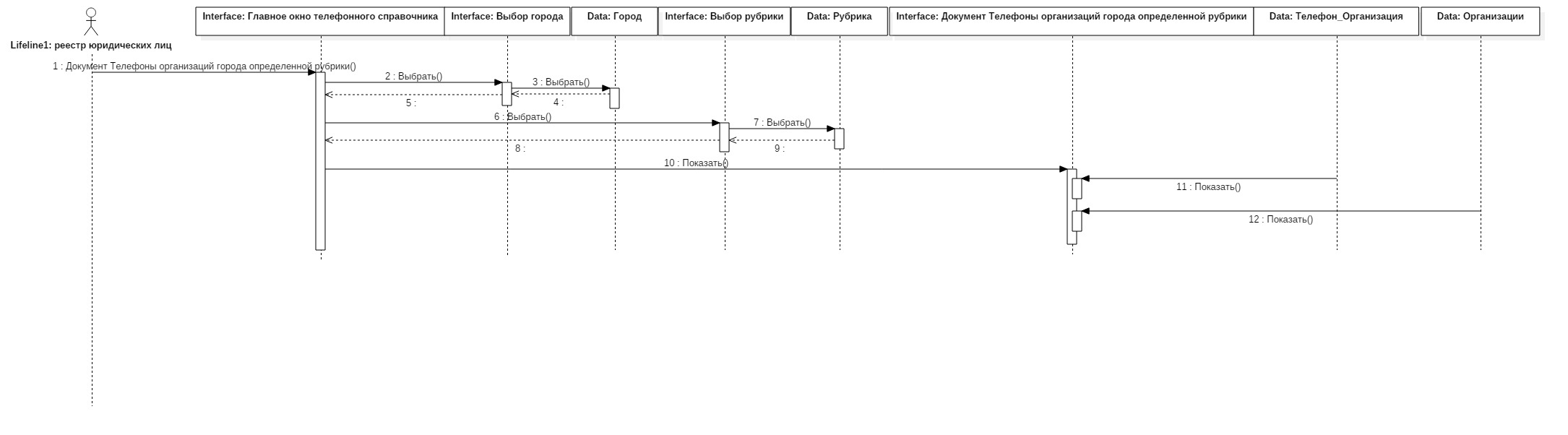


Диаграмма последовательностей для прецедента «Документ : телефоны организаций города определенной рубрики

## 090817 Диаграмма последовательностей для прецедента «Документ : Коды городов отсортированных по алфавиту»

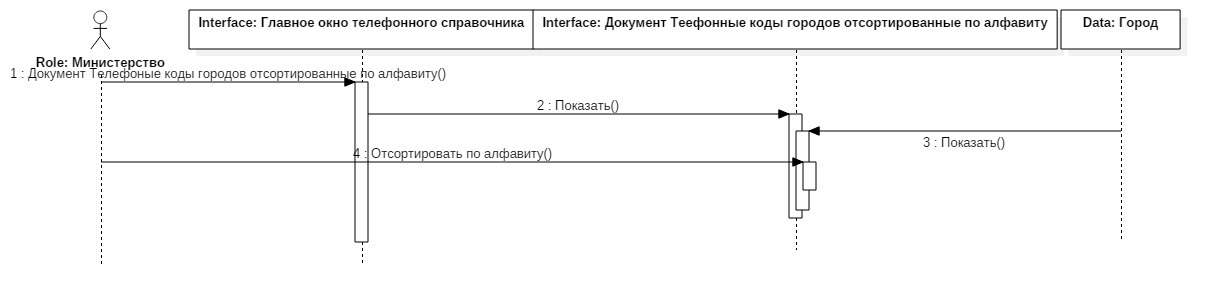


Диаграмма последовательностей для прецедента «Документ : Коды городов отсортированных по алфавиту»

## 100817 Диаграмма последовательностей для прецедента «Поиск: организаций по названию и номеру телефона»

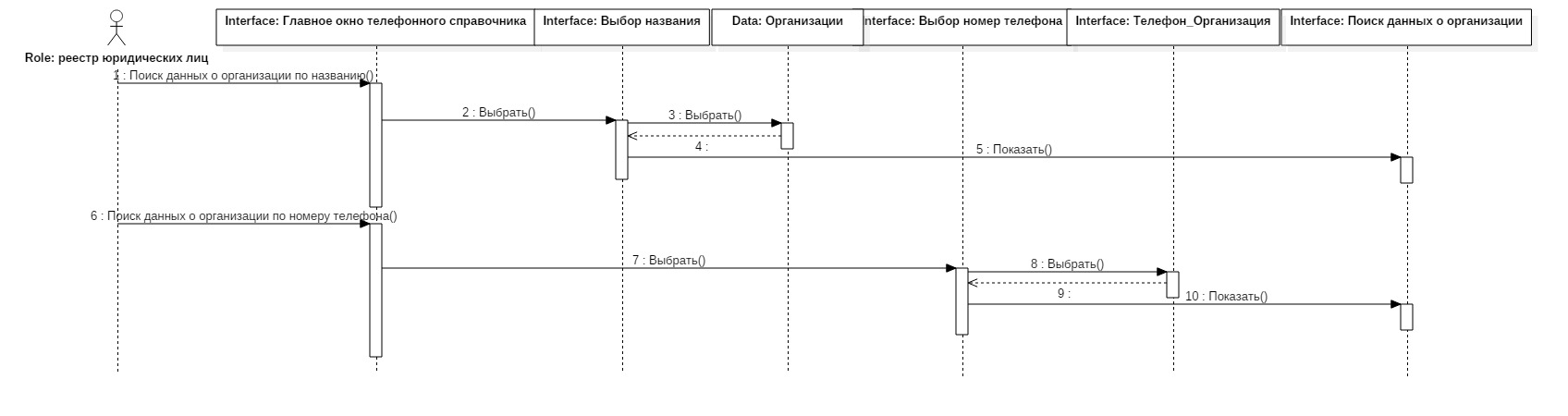


Диаграмма последовательностей для прецедента «Поиск: организаций по названию и номеру телефона»

## 100817 Диаграмма последовательностей для прецедента «Поиск: физических лиц по фамилии, номеру телефона или адресу»

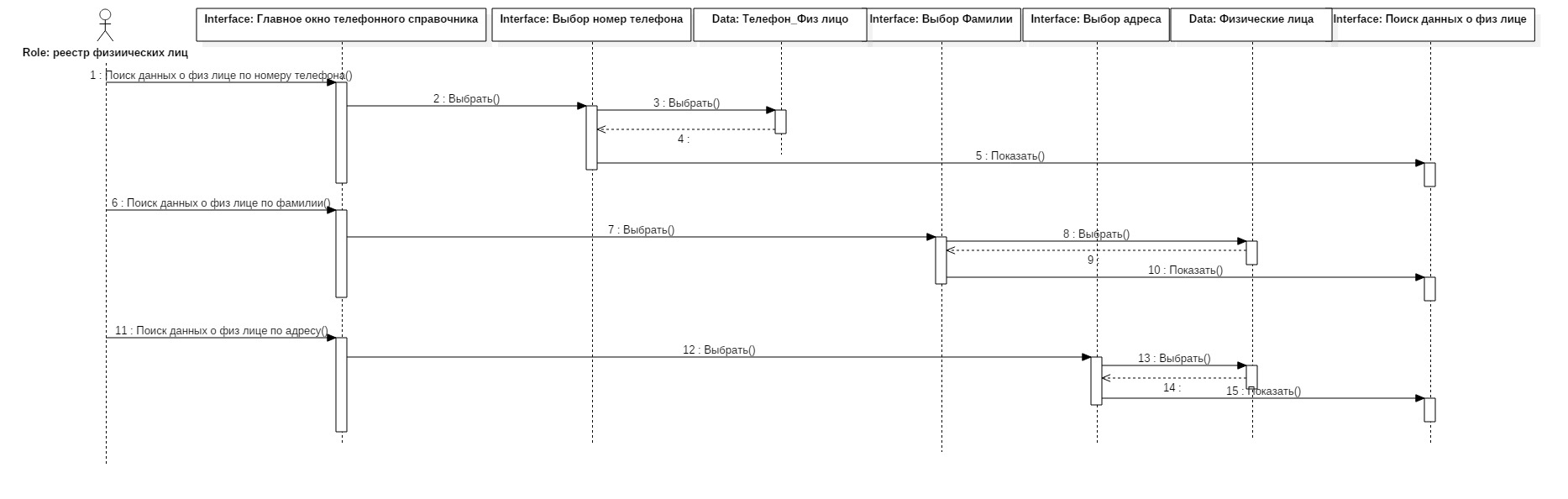


Диаграмма последовательностей для прецедента «Поиск: физических лиц по фамилии, номеру телефона или адресу»

## 100817 Диаграмма последовательностей для прецедента «Ведение списка физ лиц»

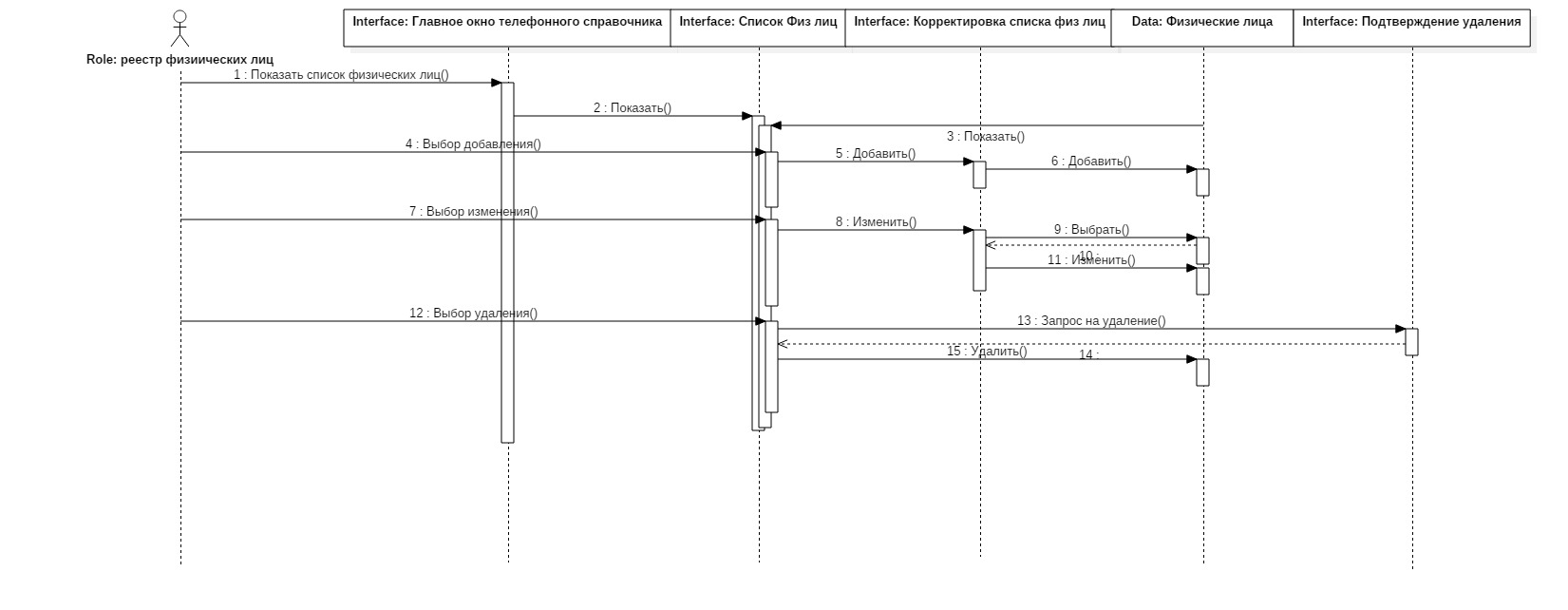


Диаграмма последовательностей для прецедента «Ведение списка физических лиц»

## 100817 Диаграмма последовательностей для прецедента «Ведение списка организаций»

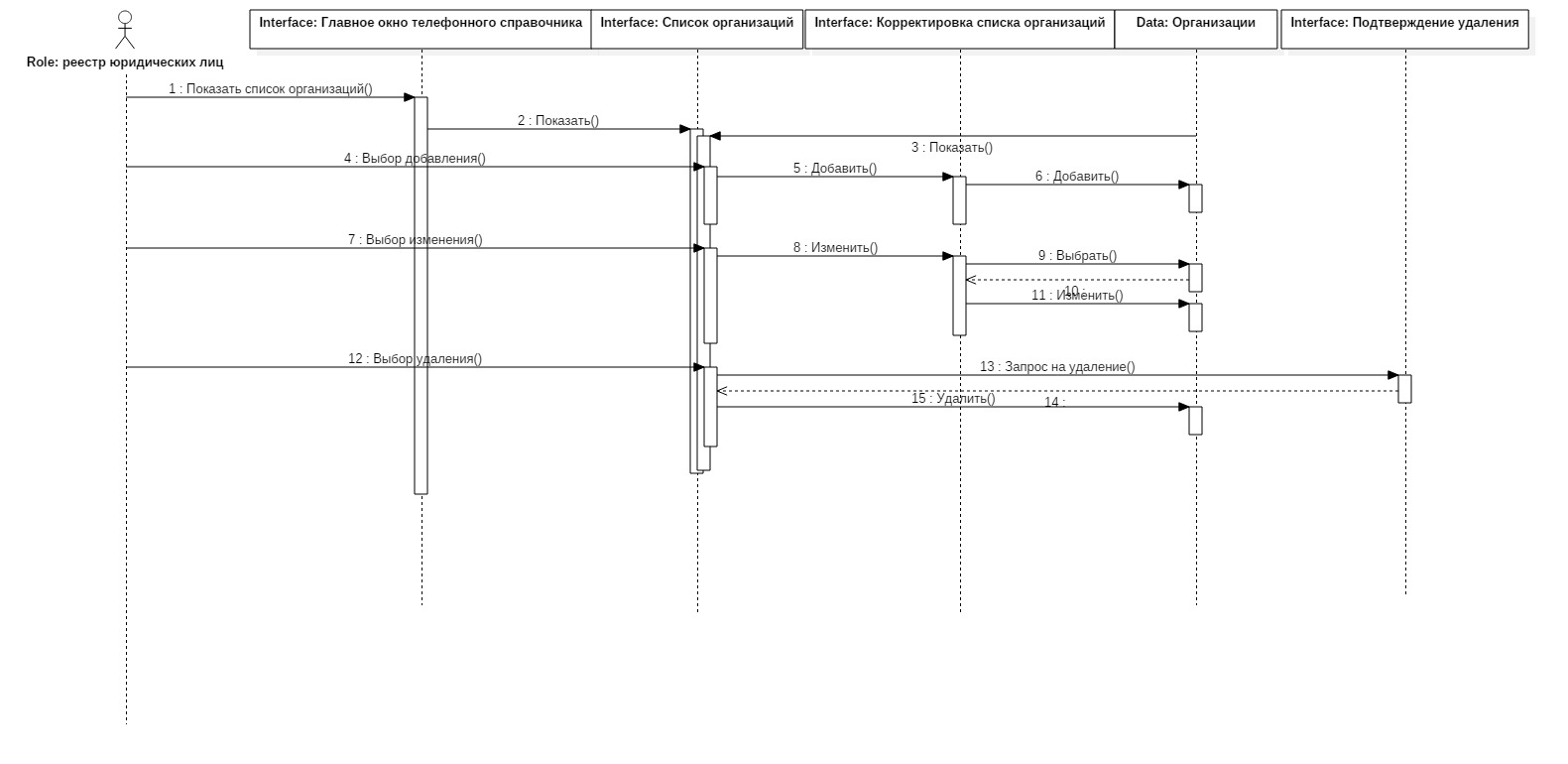


Диаграмма последовательностей для прецедента «Ведение списка организаций»

## 100817 Диаграмма последовательностей для прецедента «Регистрация номера телефона в банке номеров»

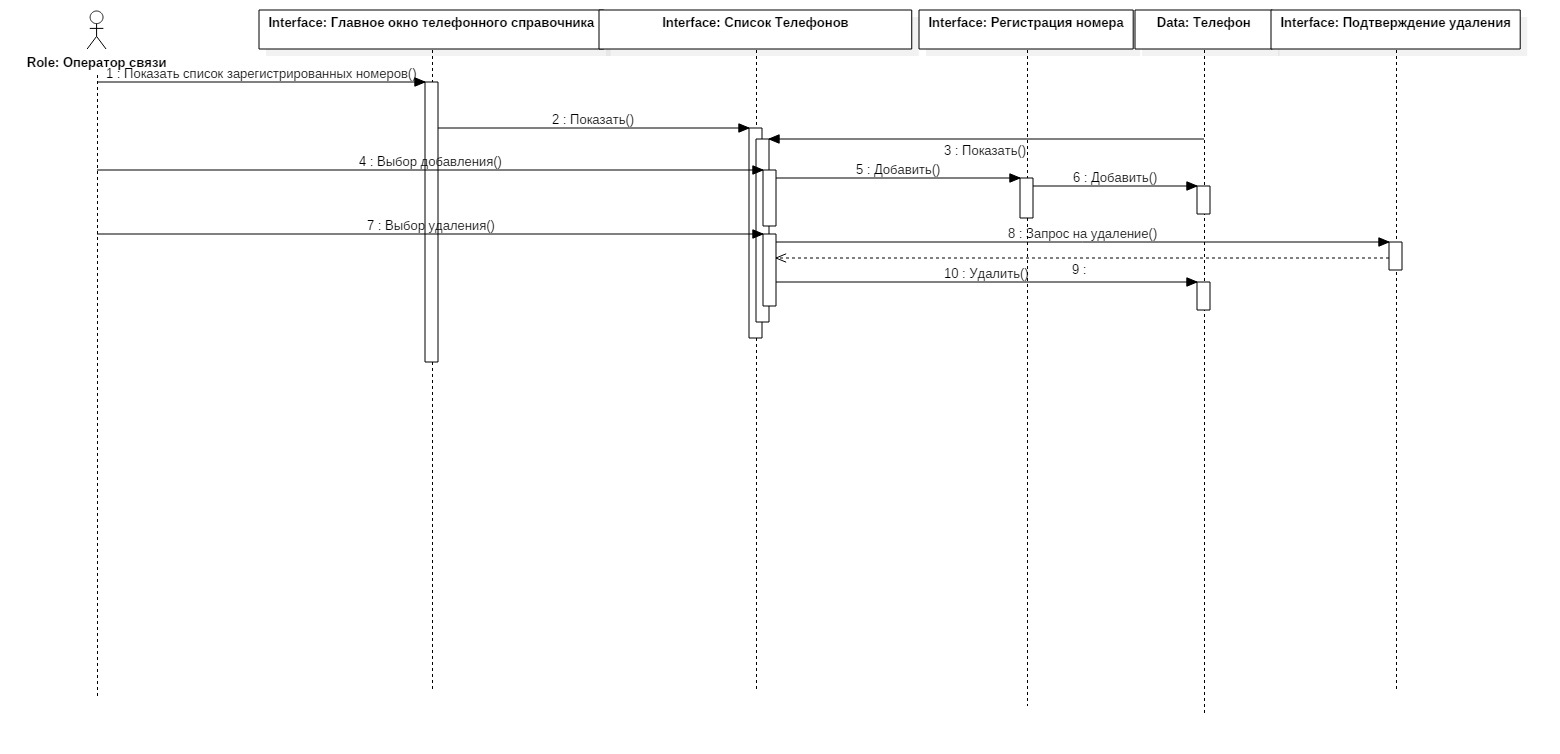


Диаграмма последовательностей для прецедента «Регистрация номера телефона»

## 100817 Диаграмма последовательностей для прецедента «Ведение списка рубрик»

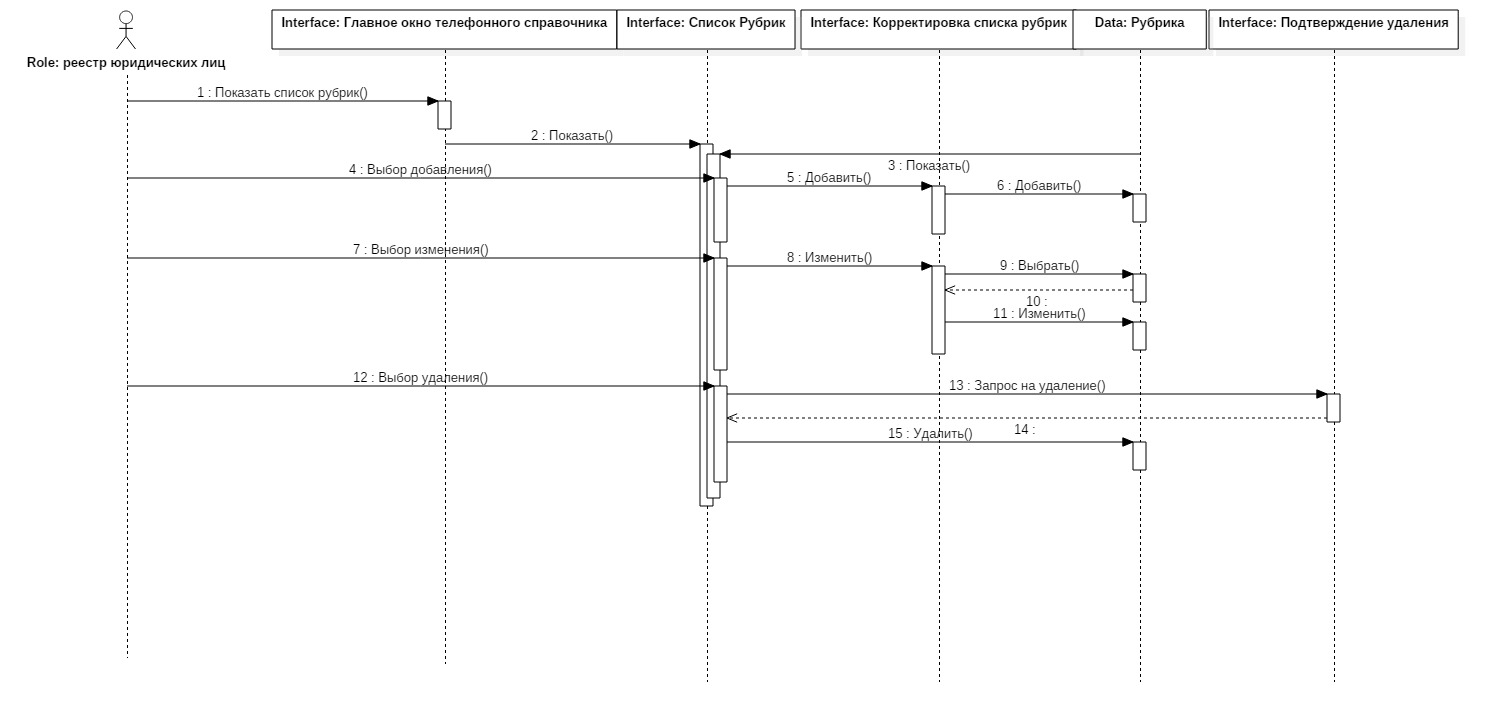


Диаграмма последовательностей для прецедента «Ведение списка рубрик»

# 100817 АНАЛИЗ ER ДИАГММЫ – ПОСТРОЕНИЕ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ (инфологическая модель)





**Правило 3:** Требуется построить 3 отношения: по одному на каждую сущность с соответствующими ключами и связующего отношения, ключом которого будет комбинация ключей двух сущностей.

* **Физическое лицо** (Паспортные данные(pk))
* **Телефон** (Номер телефона (pk))
* **Телефоны физ лиц** (Паспортные данные(fk,pk), Номер телефона (fk,pk))



**Правило 2:** Требуется построить 2 отношения. Каждому отношению соответствует 1 сущность, ключ является ключом соответствующей сущности, ключ второй сущности добавляется в качестве атрибута в первое отношение (это внешний ключ)

* **Город** (название (pk), страна (pk), код города (fk))
* **Телефон** (номер телефона (pk))



**Правило 3:** Требуется построить 3 отношения: по одному на каждую сущность с соответствующими ключами и связующего отношения, ключом которого будет комбинация ключей двух сущностей.

* **Организация** (Название организации (pk))
* **Телефон**(Номер телефона(pk))
* **Телефоны организаций** (Название организации (pk,fk), Номер телефона (pk,fk))



**Правило 4:** достаточным является использование 2х отношений, соответствующих сущностям, при чем ключами будут ключи сущностей, а дополнительно ключ первой сущности должен быть атрибутом второго отношения.

* **Организация** (Название (pk),Рубрика (fk))
* **Рубрика** (Название рубрики (pk))



**Правило 4:** достаточным является использование 2х отношений, соответствующих сущностям, при чем ключами будут ключи сущностей, а дополнительно ключ первой сущности должен быть атрибутом второго отношения.

* **Организация** (Название организации(pk), Город(fk), Страна(fk))
* **Город** (Название города(pk), страна (pk))



**Правило 4:** достаточным является использование 2х отношений, соответствующих сущностям, при чем ключами будут ключи сущностей, а дополнительно ключ первой сущности должен быть атрибутом второго отношения.

* **Город** (Название (pk), страна(pk))
* **Физические лицо (**Серия и номер паспорта (pk), Город (fk), Страна(fk))

# Нормализация и построение инфологической модели

## НФ1

Требует, чтобы каждое поле таблицы было атомарным (неделимым) и отсутствовали повторяющиеся группы атрибутов.

* Так как нам придется телефоны физ лиц сортировать по фамилиям, то нам требуется разбить поле ФИО если таковое есть на Фамилию, Имя, Отчество
* Так как нам придется телефоны физ лиц группировать по городам, то адрес придется разбить на Страна, Город, Улица, Дом, Квартира
* Так как нам придется телефоны организаций группировать тоже по городам, то нужно адрес «Организация» тоже разбить на составляющие.



## НФ2

Отношение находится во 2ой НФ, если оно удовлетворяет НФ1, pk не является избыточным ( там нет лишних атрибутов) и все не ключевые атрибуты функционально полно зависят ПФЗ от составного pk.

Телефонный справочник городов:

Телефонный справочник организаций:

Телефонный справочник физических лиц

Под назначением я имею в виду: рабочий телефон или домашний или сотовый

## НФ3

Отношение находится в НФ3, если

* оно соответствует НФ2 и
* все не ключевые атрибуты взаимно независимы, то есть не существует транзитивных зависимостей между не ключевыми атрибутами:

## БКНФ

Требует независимости ключевых атрибутов от не ключевых

## НФ4

Требует отсутствия многозначных зависимостей между атрибутами



Логическая модель БД

Для номера дома выбран тип данных VARCHAR (строка) потому что иногда пишут «дом 183а»

Подразумевается что организация расположена в одном городе и филиалов в других городах у нее нет. А телефонов у одной организации много, так как много различных отделов организации (отдел работы с кадрами, отдел бухгалтерии и т.п.).



Физическая модель БД

# 1.2 Построение физической модели

В логической модели рассматриваются отношения и атрибуты, а

в физической – таблицы и столбцы.

перейти к физической модели

следует выбрать сервер БД:

Menu->**Server, Target Server** (в **ERwin 4.0** сначала следует перейти в физическую модель **Physical**, затем выбрать пункт меню **Database, Choose database**) При этом появится окно, в котором следует выбрать сервер **InterBase** (рис. 5):

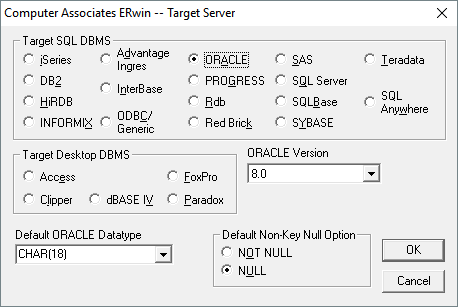
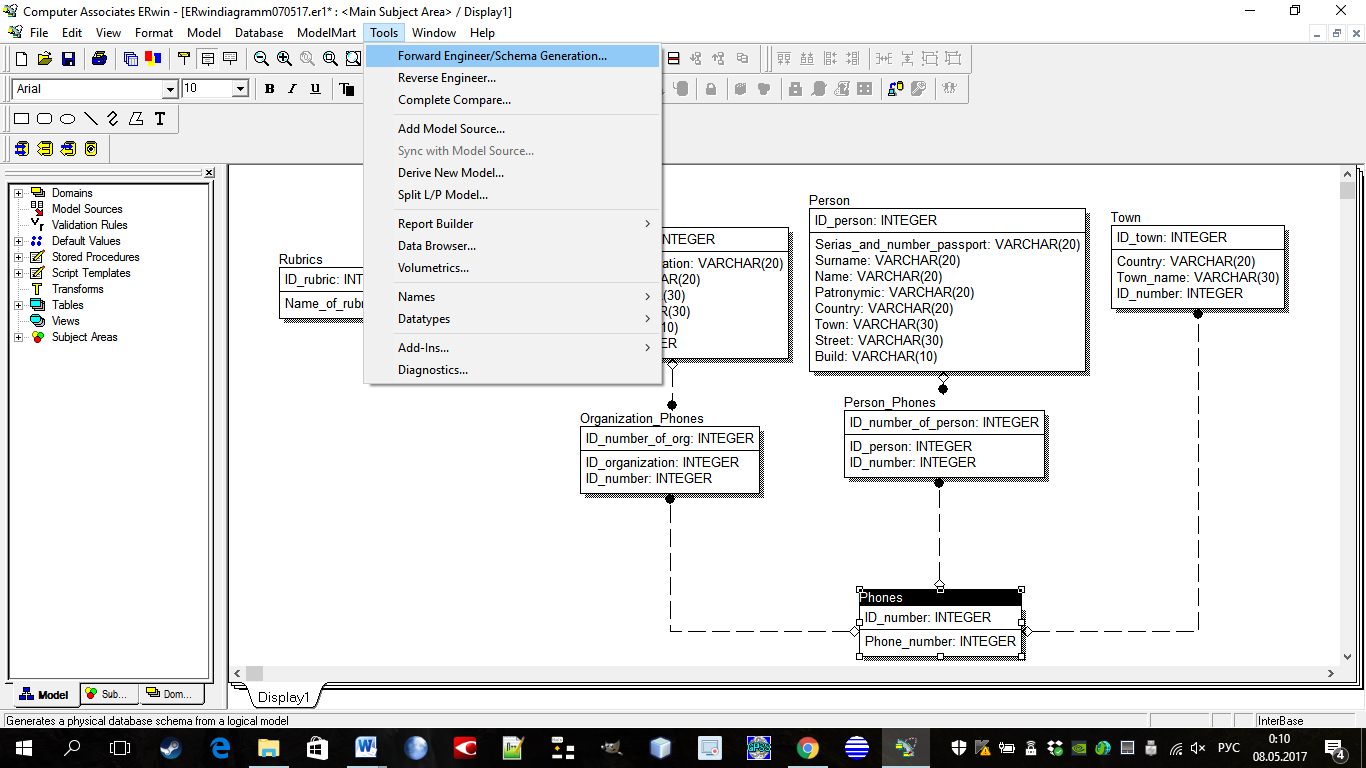


Рисунок физической модели выше

# 1.3 Генерация и корректировка текста скрипта БД (DDL-сценария)

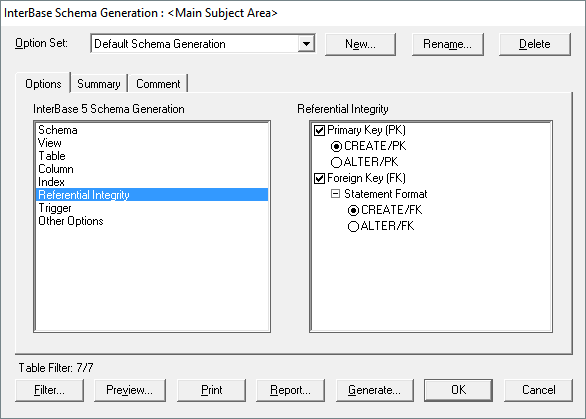
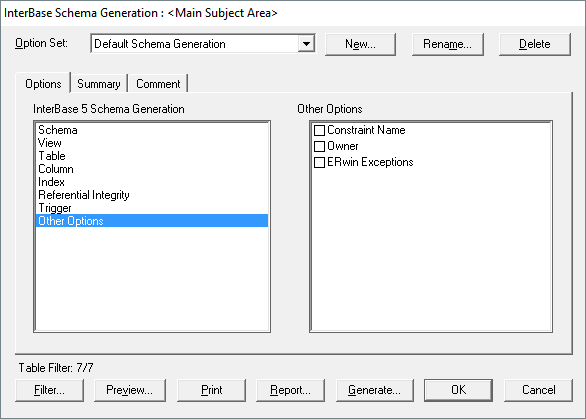
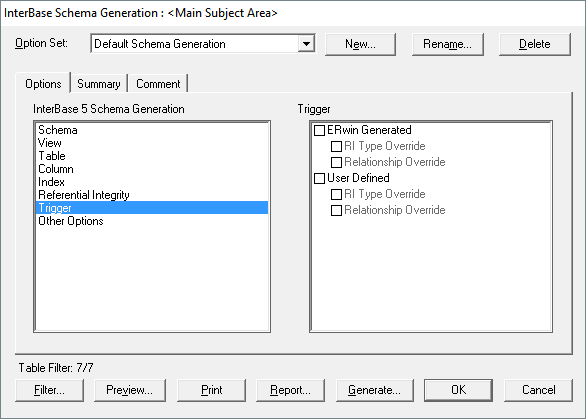
После корректировки типов и размера полей физической модели следует выбрать пункт основного меню

Tasks, Forward Engineer/Schema Generation (в ERwin 4.0 пункт Task соответствует Tools).



Появитсяокно, в котором задаются режимы генерации операторов языка *SQL*. Рекомендуется

* отменить генерацию триггеров (отключить все опции в пункте **Trigger**) и
* отменить генерацию исключений (ERWin Exception в **Other Options**).
* Указать, что первичные и внешние ключи описываются при создании таблицы (на уровне таблицы), а не при ее обновлении (ALTER), что сократит текст файла и повысит наглядность (рис. 8).



Нажали кнопку Report и сгенерировал файл с расширением sql.

указать путь для сохранения скрипта и имя файла.

Получили скрипт sql:

## Исходный SQL скрипт

/\*Создание таблицы Рубрика\*/

CREATE TABLE Rubrics (

ID\_rubric INTEGER NOT NULL,

Name\_of\_rubric VARCHAR(30) NOT NULL,

PRIMARY KEY (ID\_rubric) /\* установка первичного ключа \*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Телефон\*/

CREATE TABLE Phones (

ID\_number INTEGER NOT NULL,

Phone\_number INTEGER NOT NULL,

Minutes INTEGER NOT NULL,

Tariff NUMERIC(4,2) NOT NULL,

Pay INTEGER NOT NULL,

PRIMARY KEY (ID\_number) /\* установка первичного ключа \*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Город\*/

CREATE TABLE Town (

ID\_number INTEGER NOT NULL,

ID\_town INTEGER NOT NULL,

Town\_name VARCHAR(30) NOT NULL,

Country VARCHAR(20),

PRIMARY KEY (ID\_town), /\* установка первичного ключа \*/

FOREIGN KEY (ID\_number) REFERENCES Phones /\* установка внешнего ключа для связи с таблицей Телефоны \*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Физические лица\*/

CREATE TABLE Person (

ID\_person INTEGER NOT NULL,

Serias\_and\_number\_passport VARCHAR(20) NOT NULL,

Surname VARCHAR(20) NOT NULL,

Name VARCHAR(20) NOT NULL,

Patronymic VARCHAR(20) NOT NULL,

Street VARCHAR(30) NOT NULL,

House VARCHAR(10) NOT NULL,

ID\_town INTEGER NOT NULL,

PRIMARY KEY (ID\_person), /\* установка первичного ключа \*/

FOREIGN KEY (ID\_town) REFERENCES Town

/\* установка внешнего ключа для связи с таблицей Город \*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Телефоны\_Физ\_лиц\*/

CREATE TABLE Person\_Phones (

ID\_number\_of\_person INTEGER NOT NULL,

ID\_person INTEGER NOT NULL,

ID\_number INTEGER NOT NULL,

Phone\_assignment VARCHAR(20),

PRIMARY KEY (ID\_number\_of\_person),

/\* установка первичного ключа \*/

FOREIGN KEY (ID\_person) REFERENCES Person, /\* установка внешнего ключа для связи с таблицей «Физ лица»\*/

FOREIGN KEY (ID\_number) REFERENCES Phones

/\* установка внешнего ключа для связи с таблицей «Телефон»\*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Организации\*/

CREATE TABLE Organization (

ID\_organization INTEGER NOT NULL,

Name\_of\_organization VARCHAR(20) NOT NULL,

Street VARCHAR(30) NOT NULL,

House VARCHAR(10) NOT NULL,

ID\_rubric INTEGER NOT NULL,

ID\_town INTEGER NOT NULL,

PRIMARY KEY (ID\_organization),

/\* установка первичного ключа \*/

FOREIGN KEY (ID\_town) REFERENCES Town,

/\* установка внешнего ключа для связи с таблицей «Город» \*/

FOREIGN KEY (ID\_rubric) REFERENCES Rubrics /\* установка внешнего ключа для связи с таблицей «Рубрика»\*/

);

/\*---------------------------------------\*/

/\*Создание таблицы Телефоны Организации\*/

CREATE TABLE Organization\_Phones (

ID\_number\_of\_org INTEGER NOT NULL,

ID\_organization INTEGER NOT NULL,

ID\_number INTEGER NOT NULL,

Department\_name VARCHAR(20),

PRIMARY KEY (ID\_number\_of\_org), /\* установка первичного ключа \*/

FOREIGN KEY (ID\_number) REFERENCES Phones, /\* установка внешнего ключа для связи с таблице «Телефон»\*/

FOREIGN KEY (ID\_organization) REFERENCES Organization /\* установка внешнего ключа для связи с таблице «Телефон»\*/

);

**Ключевые поля num\_ИмяТаблицы должны идти первыми в описании таблицы (если это не так – необходимо переставить их на первое место).**

|  |  |  |
| --- | --- | --- |
| CHECK | Ограничения, накладываемые на столбцы таблицы | Пример 1: CHECK (nagruzka>=0 and nagruzka<=22) или  Пример 2: CHECK (nagruzka between 0 and 22) |
|  |  |  |
| **DEFAULT** | по умолчанию (вместо **not null**) | Пример 1:  CREATE TABLE Teach\_pred (  num\_teacher INTEGER NOT NULL,  num\_predmet INTEGER NOT NULL,  nagruzka INTEGER **DEFAULT 0**,  PRIMARY KEY (num\_teach\_pred),  FOREIGN KEY (num\_predmet)  REFERENCES Predmet,  FOREIGN KEY (num\_teacher)  REFERENCES Teacher,  **CHECK (nagruzka between 0 and 22)**  );  Пример 2:  Pol CHAR(1) DEFAULT ‘м’ CHECK (Pol = ‘м’ OR Pol = ‘ж’) |
| **COMPUTED BY (выражение)** | Столбцы вычисляемых значений | CREATE TOVAR (  Num\_Tov INTEGER,  Price INTEGER NOT NULL,  Kol\_vo INTEGER NOT NULL,  **Stoim COMPUTED BY (Price\*Kol\_vo),**  PRIMARY KEY (Num\_Tov)); |
|  | описание ссылочной целостности таблиц | FOREIGN KEY (список столбцов внешнего ключа)  REFERENCES имя\_родительской\_таблицы  [список столбцов родительской таблицы]  [ON DELETE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]  [ON UPDATE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]  Список столбцов внешнего ключа определяет столбцы дочерней таблицы, по которым строится внешний ключ. Список столбцов родительской таблицы необязателен при ссылке на первичный ключ родительской таблицы. Параметры **ON** **DELETE**, **ON** **UPDATE** определяют способы изменения подчиненных записей дочерней таблицы при удалении или изменении поля связи в записи родительской таблицы. Перечислим эти способы:   * *NO ACTION* – запрет удаления/изменения родительской записи при наличии подчиненных записей в дочерней таблице; * *CASCADE* – для оператора ON DELETE: при удалении записи родительской таблицы происходит удаление подчиненных записей в дочерней таблице; для ON UPDATE: при изменении поля связи в записи родительской таблицы происходит изменение на то же значение поля внешнего ключа у всех подчиненных записей в дочерней таблице; * *SET DEFAULT* – в поле внешнего ключа у записей дочерней таблицы заносится значение этого поля по умолчанию, указанное при определении поля (параметр DEFAULT); * *SET NULL* – в поле внешнего ключа у записей дочерней таблицы заносится значение NULL. |

Получили новый скрипт, сравним со старым, какие строки претерпели изменения:

## Измененный SQL скрипт

/\*---------------------------------------\*/

### /\*Создание таблицы Рубрика\*/

CREATE TABLE Rubrics (

ID\_rubric INTEGER NOT NULL,

Name\_of\_rubric VARCHAR(30) DEFAULT 'Без названия',

PRIMARY KEY (ID\_rubric) /\* установка первичного ключа \*/

);

/\*Создание индекса по ключу для таблицы Рубрика\*/

CREATE UNIQUE INDEX XPKRubrics ON Rubrics (

ID\_rubric

);

/\*---------------------------------------\*/

### /\*Создание таблицы Организация\*/

CREATE TABLE Organization (

ID\_organization INTEGER NOT NULL,

Name\_of\_organization VARCHAR(20) DEFAULT 'Без названия',

Street VARCHAR(30) DEFAULT 'Улица',

House VARCHAR(10) DEFAULT 'Дом',

ID\_rubric INTEGER NOT NULL,

ID\_town INTEGER NOT NULL,

PRIMARY KEY (ID\_organization),

/\* установка первичного ключа \*/

FOREIGN KEY (ID\_town) REFERENCES Town

/\* установка внешнего ключа для связи с таблицей «Города» \*/

/\* нельзя удалить город с организациями\*/

ON DELETE NO ACTION

/\*каскадное изменение\*/

ON UPDATE CASCADE,

FOREIGN KEY (ID\_rubric) REFERENCES Rubrics

/\* установка внешнего ключа для связи с таблицей «Рубрика» \*/

/\* нельзя удалить рубрику с организациями\*/

ON DELETE NO ACTION

/\*каскадное изменение\*/

ON UPDATE CASCADE

);

/\*Создание индекса по ключу для таблицы Организация\*/

CREATE UNIQUE INDEX XPKOrganization ON Organization (

ID\_organization

);

/\*Создание индекса по внешнему ключу id\_rubric для таблицы Организация\*/

CREATE UNIQUE INDEX XIF1Organization ON Organization (

ID\_rubric

);

/\*Создание индекса по внешнему ключу id\_town для таблицы Организация\*/

CREATE UNIQUE INDEX XIF2Organization ON Organization (

ID\_town

);

/\*---------------------------------------\*/

### /\*Создание таблицы Физическое лицо\*/

CREATE TABLE Person (

ID\_person INTEGER NOT NULL,

Serias\_and\_number\_passport VARCHAR(20) NOT NULL,

Surname VARCHAR(20) DEFAULT 'Иванов',

Name VARCHAR(20) DEFAULT 'Иван',

Patronymic VARCHAR(20) DEFAULT 'Иванович',

Street VARCHAR(30) DEFAULT 'Улица',

House VARCHAR(10) DEFAULT 'Дом',

ID\_town INTEGER NOT NULL,

PRIMARY KEY (ID\_person),

FOREIGN KEY (ID\_town)

REFERENCES Town

/\* нельзя удалить город с физическими лицами\*/

ON DELETE NO ACTION

ON UPDATE CASCADE

);

CREATE UNIQUE INDEX XPKPerson ON Person (

ID\_person

);

CREATE UNIQUE INDEX XIF3Person ON Person (

ID\_town

);

/\*---------------------------------------\*/

### /\*Создание таблицы телефоны\*/

CREATE TABLE Phones (

ID\_number INTEGER NOT NULL,

Phone\_number INTEGER NOT NULL,

Minutes INTEGER DEFAULT 0,

Tariff NUMERIC(4,2) DEFAULT 0,

Pay COMPUTED BY (Minutes \* Tarrif),

PRIMARY KEY (ID\_number)

);

CREATE UNIQUE INDEX XPKPhones ON Phones (

ID\_number

);

/\*---------------------------------------\*/

### /\*Создание таблицы Физическое Телефоны организаций\*/

CREATE TABLE Organization\_Phones (

ID\_number\_of\_org INTEGER NOT NULL,

ID\_organization INTEGER NOT NULL,

ID\_number INTEGER NOT NULL,

Department\_name VARCHAR(20) DEFAULT 'Без названия',

PRIMARY KEY (ID\_number\_of\_org),

FOREIGN KEY (ID\_number)

REFERENCES Phones

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (ID\_organization)

REFERENCES Organization

ON DELETE CASCADE

ON UPDATE CASCADE

);

CREATE UNIQUE INDEX XPKOrganization\_Phones ON Organization\_Phones (

ID\_number\_of\_org

);

CREATE UNIQUE INDEX XIF4Organization\_Phones ON Organization\_Phones (

ID\_organization

);

CREATE UNIQUE INDEX XIF5Organization\_Phones ON Organization\_Phones (

ID\_number\_of\_org

);

CREATE TABLE Person\_Phones (

ID\_number\_of\_person INTEGER NOT NULL,

ID\_person INTEGER NOT NULL,

ID\_number INTEGER NOT NULL,

Phone\_assignment VARCHAR(20) DEFAULT 'Рабочий',

PRIMARY KEY (ID\_number\_of\_person),

FOREIGN KEY (ID\_person)

REFERENCES Person

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (ID\_number)

REFERENCES Phones

ON DELETE CASCADE

ON UPDATE CASCADE

);

CREATE UNIQUE INDEX XPKPerson\_Phones ON Person\_Phones (

ID\_number\_of\_person

);

CREATE UNIQUE INDEX XIF6Person\_Phones ON Person\_Phones (

ID\_person

);

CREATE UNIQUE INDEX XIF7Person\_Phones ON Person\_Phones (

ID\_number

);

CREATE TABLE Town (

ID\_town INTEGER NOT NULL,

Town\_name VARCHAR(30) DEFAULT 'Город без названия',

ID\_number INTEGER NOT NULL,

Country VARCHAR(20) DEFAULT 'Страна без названия',

PRIMARY KEY (ID\_town),

FOREIGN KEY (ID\_number)

REFERENCES Phones

ON DELETE NO ACTION

ON UPDATE CASCADE

);

CREATE UNIQUE INDEX XPKTown ON Town (

ID\_town

);

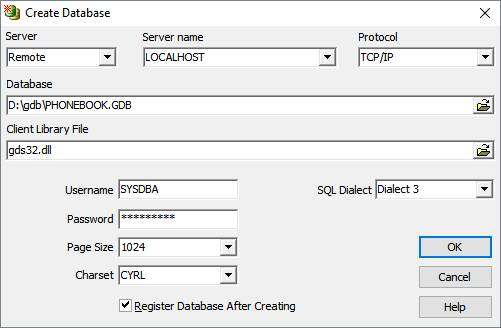
CREATE UNIQUE INDEX XIF8Town ON Town (

ID\_number

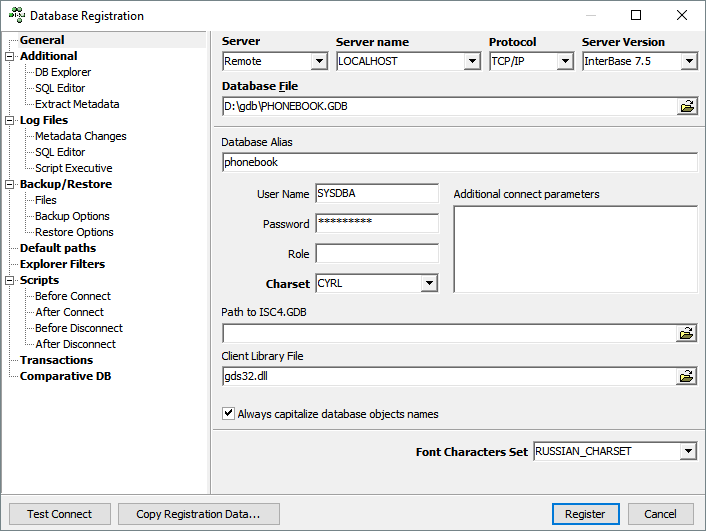
);

# 1.4 Создание БД и выполнение сгенерированного sql-скрипта

Database->Create Database

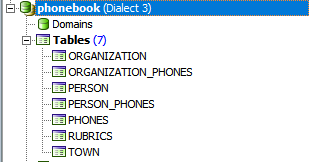


Password: masterkey



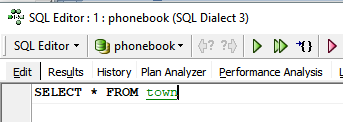
Database->Connect to database

Tools->Script Executive



Меню-> Tools->New SQL Editor

Запрос выдачи всех данных из таблицы «Города»



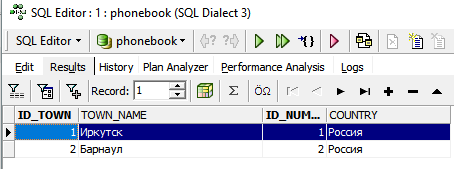
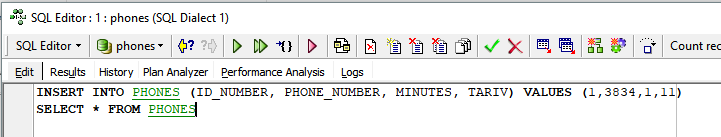
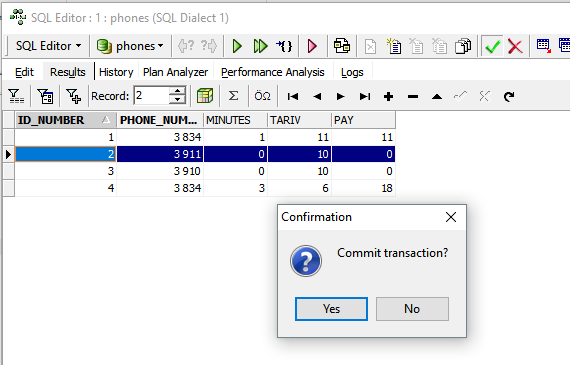


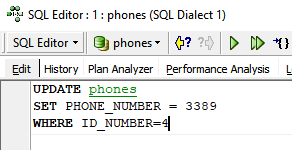
Рисунок – Запрос выдачи данных из таблицы «Города»

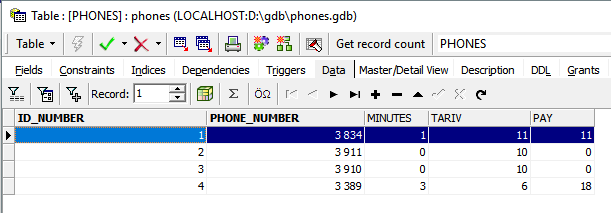
Пример добавления данных в таблицу «Телефоны»



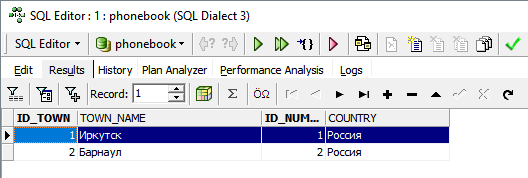


Пусть требуется обновить данные





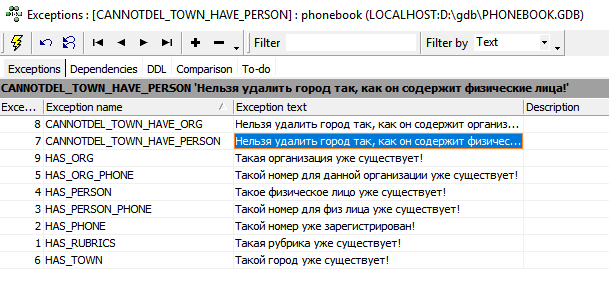
Теперь заполним таблицу «Города»: для это сначала требуется нажать на пиктограмму , затем занести данные, после чего нажать на «Commit» .



# 1.5 Создание исключений

Исключение – именованное сообщение, которое вызывается из хранимой процедуры или триггера при возникновении какой-либо ситуации.

Database->New exception

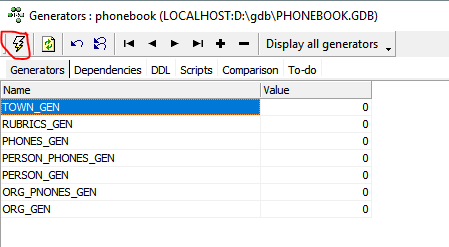


После создания исключения его необходимо откомпилировать и подтвердить транзакцию (Commit).

# 1.6 Создание генераторов

Генератор – механизм для создания уникальных значений первичных ключей при добавлении строк к таблице.

Database->New generator



После создания генератора его необходимо откомпилировать и подтвердить транзакцию (Commit).

# 1.7 Создание хранимых процедур

Хранимая процедура – это модуль написанный на процедурном языке сервера БД ( в нашем случае Interbase) и хранящийся в БД как метаданне (данные о данных).

Сущестувет 2 разновидности хранимых процедур:

* Процедуры выбора – возвращают более одного значения
* Процедура действия – вообще может не возвращать данных и используется для реализации каких-либо действий

## Процедура добавления записи в таблицу «Организации» INS\_ORG

Процедура проверяет, нет ли в таблице организации с таким ИНН и с таким же названием организации, если нет, то в таблицу добавляется запись об организации, если подобная запись есть, то вызывается исключение has\_org, которое выводит на экран сообщение: «Такая организация уже существует!». То есть в данном случает подразумевается, что может быть так, что у двух разных по названию организаций будет одинаковое ИНН, что будет означает, что одна из данных организаций является дочерней.

Согласно п. 7 ст. 84 НК РФ идентификационный номер налогоплательщика (ИНН) присваивается именно налогоплательщику, т.е. самой организации (юридическому лицу). Приложением N 1 к приказу МНС РФ от 03.03.2004 N БГ-3-09/178 утвержден Порядок и условия присвоения, применения, а также изменения ИНН и форм документов, используемых при постановке на учет, снятии с учета юридических и физических лиц (далее - Порядок).   
Согласно п. 2.1.4 Порядка код причины постановки на учет (КПП) присваивается организации в том числе при постановке на учет по месту нахождения обособленных подразделений на основании:   
- заявления о постановке на учет;   
- одновременно представленных в одном экземпляре заверенных в установленном порядке копий свидетельства о постановке на учет в налоговом органе организации по месту ее нахождения;   
- документов, подтверждающих создание обособленного подразделения (при их наличии).   
Из вышеизложенного следует, что ИНН присваивается организации в целом, т.е. ИНН филиала и головной компании соответствуют друг другу. При этом КПП филиала может отличаться от КПП самой головной организации (юридического лица).

CREATE PROCEDURE INS\_ORG (

param\_org\_name varchar(20),

param\_street varchar(30),

param\_house varchar(10),

param\_id\_rubric integer,

param\_id\_town integer,

param\_inn integer)

as

declare variable k integer; /\* count of org where eaqul inn \*/

begin

select count(\*) from organization where INN=:param\_inn and NAME\_OF\_ORGANIZATION=:param\_org\_name INTO K;

/\*or with such inn have not in table\*/

if (:K=0) then

insert into organization (ID\_ORGANIZATION, NAME\_OF\_ORGANIZATION, STREET, HOUSE, ID\_RUBRIC, ID\_TOWN, INN)

values (gen\_id(ORG\_GEN,1), :param\_org\_name, :param\_street, :param\_house, :param\_id\_rubric, :param\_id\_town, :param\_inn);

else

exception has\_org;

end

## Процедура добавления записи в таблицу «Физические лица» INS\_PERSON

Процедура проверяет, нет ли в таблице физического лица с такими же паспортными данными, если нет, то в таблицу добавляется запись об физическом лице, если подобная запись есть, то вызывается исключение has\_person, которое выводит на экран сообщение: «Такое физическое лицо уже существует!».

CREATE PROCEDURE INS\_PERSON (

param\_passport\_number varchar(20),

param\_surname varchar(20),

param\_name varchar(20),

param\_patronymic varchar(20),

param\_street varchar(30),

param\_house varchar(10),

param\_id\_town integer)

as

declare variable k integer; /\* count of person where eaqul passpot number \*/

begin

select count(\*) from PERSON where SERIAS\_AND\_NUMBER\_PASSPORT=:param\_passport\_number INTO K;

/\*or with such inn have not in table\*/

if (:K=0) then

insert into PERSON (ID\_PERSON, SERIAS\_AND\_NUMBER\_PASSPORT, SURNAME, NAME, PATRONYMIC, STREET, HOUSE, ID\_TOWN)

values (gen\_id(PERSON\_GEN,1), :param\_passport\_number,:param\_surname, :param\_name, :param\_patronymic, :param\_street, :param\_house, :param\_id\_town);

else

exception has\_person;

end

## Процедура удаления записи из таблицы «Город» DEL\_TOWN

Здесь вызывается исключение CANNOTDEL\_TOWN\_HAVE\_PERSON, если в данном городе содержатся физические лица, а также вызывается исключение CANNOTDEL\_TOWN\_HAVE\_ORG, если в данном городе содержатся юридические лица (организации).

CREATE PROCEDURE DEL\_TOWN (

param\_id\_town integer)

as

declare variable k integer;

declare variable l integer;

begin

select count(\*) from person where ID\_TOWN=:param\_id\_town into :K;

if (:K=0) then

delete from TOWN where ID\_TOWN=:param\_id\_town;

else exception CANNOTDEL\_TOWN\_HAVE\_PERSON;

/\*-------------------------------------------------------------------\*/

select count(\*) from organization where ID\_TOWN=:param\_id\_town into :L;

if (:L=0) then

delete from TOWN where ID\_TOWN=:param\_id\_town;

else exception CANNOTDEL\_TOWN\_HAVE\_ORG;

end

## Процедура добавления номера телефона в справочник INS\_PHONE

Здесь выдается количество записей из таблицы «Телефонные номера», где номер телефона совпадает с тем номером, который хочет зарегистрировать пользователь. Если такой номер уже существует в таблице, то вызывается исключение HAS\_PHONE, иначе запись заносится в таблицу.

CREATE PROCEDURE INS\_PHONE (

param\_phone\_number integer,

param\_minutes integer,

param\_tariv integer)

as

declare variable k integer; /\* count of phones where eaqul phone\_number \*/

begin

select count(\*) from phones where phone\_number=:param\_phone\_number INTO K;

/\*phone with such number have not in table\*/

if (:K=0) then

insert into phones (id\_number, phone\_number, minutes, tariv)

values (gen\_id(PHONES\_GEN,1), :param\_phone\_number, :param\_minutes, :param\_tariv);

else

exception HAS\_PHONE;

end

## Процедура добавления рубрики INS\_RUBRIC

Здесь выдается количество записей из таблицы «Рубрики», где название рубрики совпадает с тем названием новой рубрики, которое хочет зарегистрировать пользователь. Если такая рубрика уже существует в таблице, то вызывается исключение HAS\_RUBRICS, иначе запись заносится в таблицу.

CREATE PROCEDURE INS\_RUBRICS (

param\_name\_of\_rubric varchar(30))

as

declare variable k integer; /\* count of rubrics where eaqul name\_of\_rubrics \*/

begin

select count(\*) from rubrics where name\_of\_rubric=:param\_name\_of\_rubric INTO K;

/\*rubrics with such name have not in table\*/

if (:K=0) then

insert into rubrics (id\_rubric, name\_of\_rubric)

values (gen\_id(RUBRICS\_GEN,1), :param\_name\_of\_rubric);

else

exception HAS\_RUBRICS;

end

## Процедура добавления записи в таблицу «Город» INS\_TOWN

Здесь выдается количество записей из таблицы «Город», где название города и страна совпадает с тем городом и страной, которые хочет зарегистрировать пользователь. Если такой город в такой стране уже существует в таблице, то вызывается исключение HAS\_TOWN, иначе запись заносится в таблицу. То есть здесь предусматривается, что два города с одинаковыми названиями могут находиться в разных странах.

CREATE PROCEDURE INS\_TOWN (

param\_town\_name varchar(30),

param\_id\_number integer,

param\_country varchar(20))

as

declare variable k integer; /\* count of town where eaqul town\_name and country \*/

begin

select count(\*) from town where town\_name=:param\_town\_name and country=:param\_country INTO K;

/\*town with such name have not in table\*/

if (:K=0) then

insert into town (id\_town, town\_name, id\_number, country)

values (gen\_id(TOWN\_GEN,1), :param\_town\_name, :param\_id\_number, :param\_country);

/\*values ('3', :param\_town\_name, :param\_id\_number, :param\_country); \*/

else

exception HAS\_TOWN;

end

## Процедура добавления записи в таблицу «Телефоны контрагентов» INS\_ORG\_PHONES

Если в таблице уже существует телефон для данной организации, то вызывается исключение HAS\_ORG\_PHONES, иначе запись добавляется в таблицу.

CREATE PROCEDURE INS\_ORG\_PHONES (

param\_id\_organization integer,

param\_id\_number integer,

param\_department\_name varchar(20))

as

declare variable k integer; /\* count of org\_phons where eaqul id\_org and id\_number \*/

begin

select count(\*) from organization\_phones

where id\_organization=:param\_id\_organization and id\_number=:param\_id\_number INTO K;

/\*org\_phones with such id\_org and id\_number have not in table\*/

if (:K=0) then

insert into organization\_phones (id\_number\_of\_org, id\_organization, id\_number, department\_name)

values (gen\_id(ORG\_PNONES\_GEN,1), :param\_id\_organization, :param\_id\_number, :param\_department\_name);

else

exception HAS\_ORG\_PHONE;

end

## Процедура добавления записи в таблицу «Телефоны физических лиц» INS\_PERSON\_PHONES

Если в таблице уже существует телефон физического лица, то вызывается исключение HAS\_PERSON\_PHONES, иначе запись добавляется в таблицу.

CREATE PROCEDURE INS\_PERSON\_PHONES (

param\_id\_person integer,

param\_id\_number integer,

param\_phone\_assignment varchar(20))

as

declare variable k integer; /\* count of person\_phones where eaqul id\_person and id\_number \*/

begin

select count(\*) from person\_phones

where id\_person=:param\_id\_person and id\_number=:param\_id\_number INTO K;

/\*person\_phones with such id\_person and id\_number have not in table\*/

if (:K=0) then

insert into person\_phones (id\_number\_of\_person, id\_person, id\_number, phone\_assignment)

values (gen\_id(PERSON\_PHONES\_GEN,1), :param\_id\_person, :param\_id\_number, :param\_phone\_assignment);

else

exception HAS\_PERSON\_PHONE;

end

## Процедура UPD\_TOWN

CREATE PROCEDURE UPD\_TOWN (

p\_id\_town integer,

p\_town\_name varchar(30),

p\_id\_number integer,

p\_country varchar(20))

as

declare variable k integer;

begin

select count(\*) from TOWN

Where town\_name=:p\_town\_name and country=:p\_country into :k;

if (:k=0) then

UPDATE TOWN

SET town\_name=:p\_town\_name, id\_number=:p\_id\_number, country=:p\_country

WHERE id\_town=:p\_id\_town;

else

exception HAS\_TOWN;

suspend;

end

## Процедура UPD\_RUBRICS

CREATE PROCEDURE UPD\_RUBRICS (

pid\_rubric integer,

prubric\_name varchar(30))

as

declare variable k integer;

begin

select count(\*) from RUBRICS

Where name\_of\_rubric=:prubric\_name into :k;

if (:k=0) then

UPDATE RUBRICS

SET name\_of\_rubric=:prubric\_name

WHERE id\_rubric=:pid\_rubric;

else

exception HAS\_RUBRICS;

suspend;

end

## Процедура UPD\_PHONE

CREATE PROCEDURE UPD\_PHONE (

p\_id\_number integer,

p\_phone\_number integer,

p\_minutes integer,

p\_tariv integer)

as

declare variable k integer;

begin

select count(\*) from PHONES

Where phone\_number=:p\_phone\_number into :k;

if (:k=0) then

UPDATE PHONES

SET phone\_number=:p\_phone\_number,minutes=:p\_minutes, tariv=:p\_tariv

WHERE id\_number=:p\_id\_number;

else

exception HAS\_PHONE;

suspend;

end

## Процедура UPD\_PERSON\_PHONE

CREATE PROCEDURE UPD\_PERSON\_PHONES (

p\_id\_number\_of\_person integer,

p\_id\_person integer,

p\_id\_number integer,

p\_phone\_assignment varchar(20))

as

declare variable k integer;

begin

select count(\*) from PERSON\_PHONES

Where id\_person=:p\_id\_person and id\_number=:p\_id\_number into :k;

if (:k=0) then

UPDATE PERSON\_PHONES

SET id\_person=:p\_id\_person, id\_number=:p\_id\_number, phone\_assignment=:p\_phone\_assignment

WHERE id\_number\_of\_person=:p\_id\_number\_of\_person;

else

exception HAS\_PERSON\_PHONE;

suspend;

end

## Процедура UPD\_PERSON

CREATE PROCEDURE UPD\_PERSON (

p\_id\_person integer,

p\_serias\_and\_number\_passport varchar(20),

p\_surname varchar(20),

p\_name varchar(20),

p\_patronymic varchar(20),

p\_street varchar(30),

p\_house varchar(10),

p\_id\_town integer)

as

declare variable k integer;

begin

select count(\*) from PERSON

Where serias\_and\_number\_passport=:p\_serias\_and\_number\_passport into :k;

if (:k=0) then

UPDATE PERSON

SET serias\_and\_number\_passport=:p\_serias\_and\_number\_passport,

surname=:p\_surname,

name=:p\_name,

patronymic=:p\_patronymic,

street=:p\_street,

house=:p\_house,

id\_town=:p\_id\_town

WHERE id\_person=:p\_id\_person;

else

exception HAS\_PERSON;

suspend;

end

## Процедура UPD\_ORGANIZATION\_PHONES

CREATE PROCEDURE UPD\_ORGANIZATION\_PHONES (

p\_id\_number\_of\_org integer,

p\_id\_organization integer,

p\_id\_number integer,

p\_department\_name varchar(20))

as

declare variable k integer;

begin

select count(\*) from ORGANIZATION\_PHONES

Where id\_organization=:p\_id\_organization and id\_number=:p\_id\_number into :k;

if (:k=0) then

UPDATE ORGANIZATION\_PHONES

SET id\_organization=:p\_id\_organization,

id\_number=:p\_id\_number,

department\_name=:p\_department\_name

WHERE id\_number\_of\_org=:p\_id\_number\_of\_org;

else

exception HAS\_ORG\_PHONE;

suspend;

end

## Процедура UPD\_ORGANIZATION

CREATE PROCEDURE UPD\_ORGANIZATION (

p\_id\_organization integer,

p\_name\_of\_organization varchar(20),

p\_street varchar(30),

p\_house varchar(10),

p\_id\_rubric integer,

p\_id\_town integer,

p\_inn integer)

as

declare variable k integer;

begin

select count(\*) from ORGANIZATION

Where name\_of\_organization=:p\_name\_of\_organization and inn=:p\_inn into :k;

if (:k=0) then

UPDATE ORGANIZATION

SET name\_of\_organization=:p\_name\_of\_organization,

street=:p\_street,

house=:p\_house,

id\_rubric=:p\_id\_rubric,

id\_town=:p\_id\_town,

inn=:p\_inn

WHERE id\_organization=:p\_id\_organization;

else

exception HAS\_ORG;

suspend;

end

## Процедура выходного документа: Телефонные коды городов, отсортированные по алфавиту.

CREATE PROCEDURE TOWN\_ORDER\_BY\_ALFAVIT

returns (

p\_town\_name varchar(30),

p\_country varchar(20),

p\_phone\_number integer)

as

begin

For select t.town\_name, t.country, p.phone\_number

from TOWN t, PHONES p

where t.id\_number=p.id\_number

order by t.town\_name into :p\_town\_name, :p\_country, :p\_phone\_number

do

suspend;

end

## Процедура выходного документа: Телефоны физических лиц города, отсортированные по фамилии.

CREATE PROCEDURE PP\_ORDER\_BY\_ALF (

p\_id\_town integer)

returns (

p\_surname varchar(20),

p\_name varchar(20),

p\_patronymic varchar(20),

l\_phone\_assignment varchar(20),

h\_phone\_number integer,

t\_town\_name varchar(30),

t\_country varchar(20))

as

begin

for select p.surname, p.name, p.patronymic,

l.phone\_assignment,

h.phone\_number,

t.town\_name, t.country

from PERSON p, PERSON\_PHONES l, PHONES h, TOWN t

where p.id\_town=t.id\_town and p.id\_town=:p\_id\_town and p.id\_person=l.id\_person and l.id\_number=h.id\_number

order by p.name into :p\_surname, :p\_name , :p\_patronymic,

:l\_phone\_assignment,

:h\_phone\_number,

:t\_town\_name,:t\_country

do

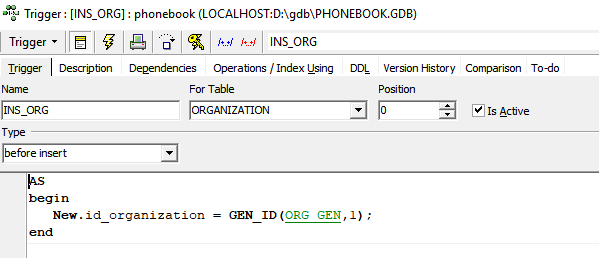
suspend;

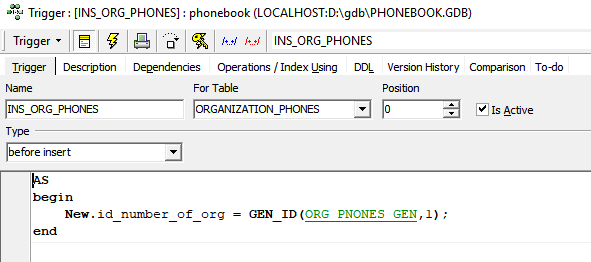
end

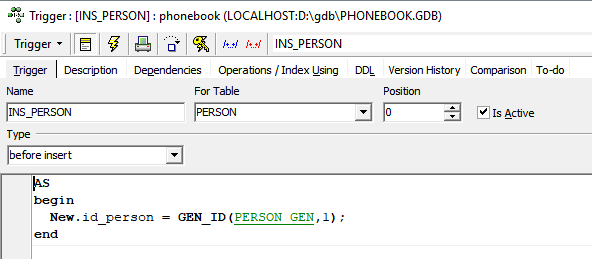
# 1.8 Создание триггеров для поддержания ссылочной целостности

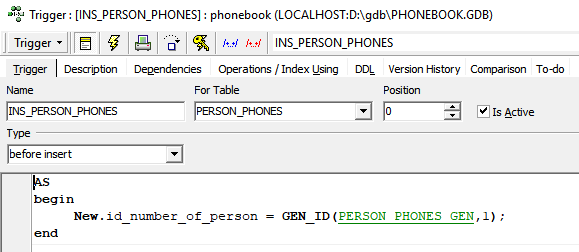
Триггер – это процедура БД, автоматически вызываемая SQL сервером при обновлении, удалении или добавлении записи в таблицах БД.

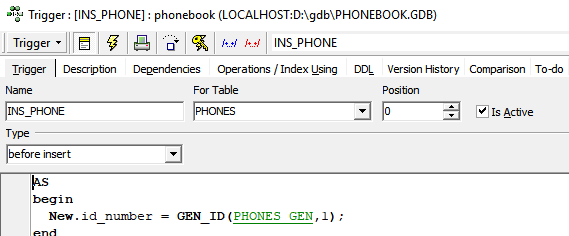
При добавлении новой записи в таблицу, генерируется новый id этой записи.

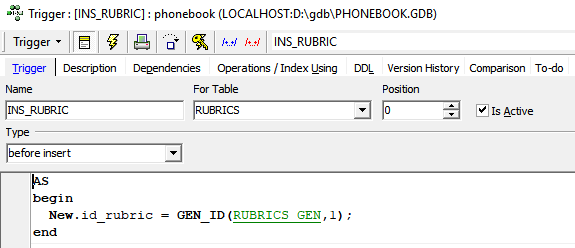


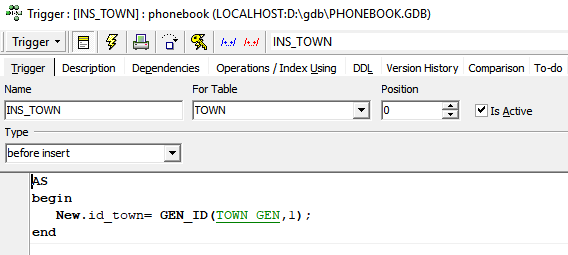












Значение Old.<имя столбца> позволяет обратиться к состоянию столбца до внесения изменений

Значение New.<имя столбца> позволяет обратиться к состоянию столбца после внесения изменений

# 1.9 Создание просмотров

Просмотр (view) – это виртуальная таблица, которая не сохраняется в БД физически, но выглядит как реальная таблица. Просмотр может содердать выборочные данные из одной или нескольких просмотров и используется для сохранения наиболее часто используемых запросов к БД.

DataBase->NewView