**E6DAB**

**Efterår 2018**

**DAB Hand In 2**

**Gruppe 4**

**Midtvejsevaluering**

**Deltagere:**

|  |
| --- |
| ***#1***  Stud.nr.: 201509807 Navn: Ali Saleh |
| ***#2***  Stud.nr.: 201600057 Navn: Jonas Jul Yde |
| ***Dato: 30/09-2018*** |

# Preface

This paper has been made by Jonas Yde & Ali Saleh. It has been built upon Hand In 1 – Endelig Aflevering and improved to solve the issues given in Hand In 2.

Table of Contents

[Preface 1](#_Toc525995143)

[Domain Model with Entity Relationship Diagram (ERD) 2](#_Toc525995144)

[Logical Schema from DDS Lite 3](#_Toc525995145)

[Domain Model with Domain Driven Design (DDD) 6](#_Toc525995146)

[JSON Structure of Domain Model 7](#_Toc525995147)

[XML Model of Domain Model 8](#_Toc525995148)

[Application Software Architecture 9](#_Toc525995149)

[Scaffolding model 10](#_Toc525995150)

[Graph-tree model 10](#_Toc525995151)

# Domain Model with Entity Relationship Diagram (ERD)

This is the Entity Relationship Diagram. The figure explains the participation, connectivity, cardinality, degrees and navigation.

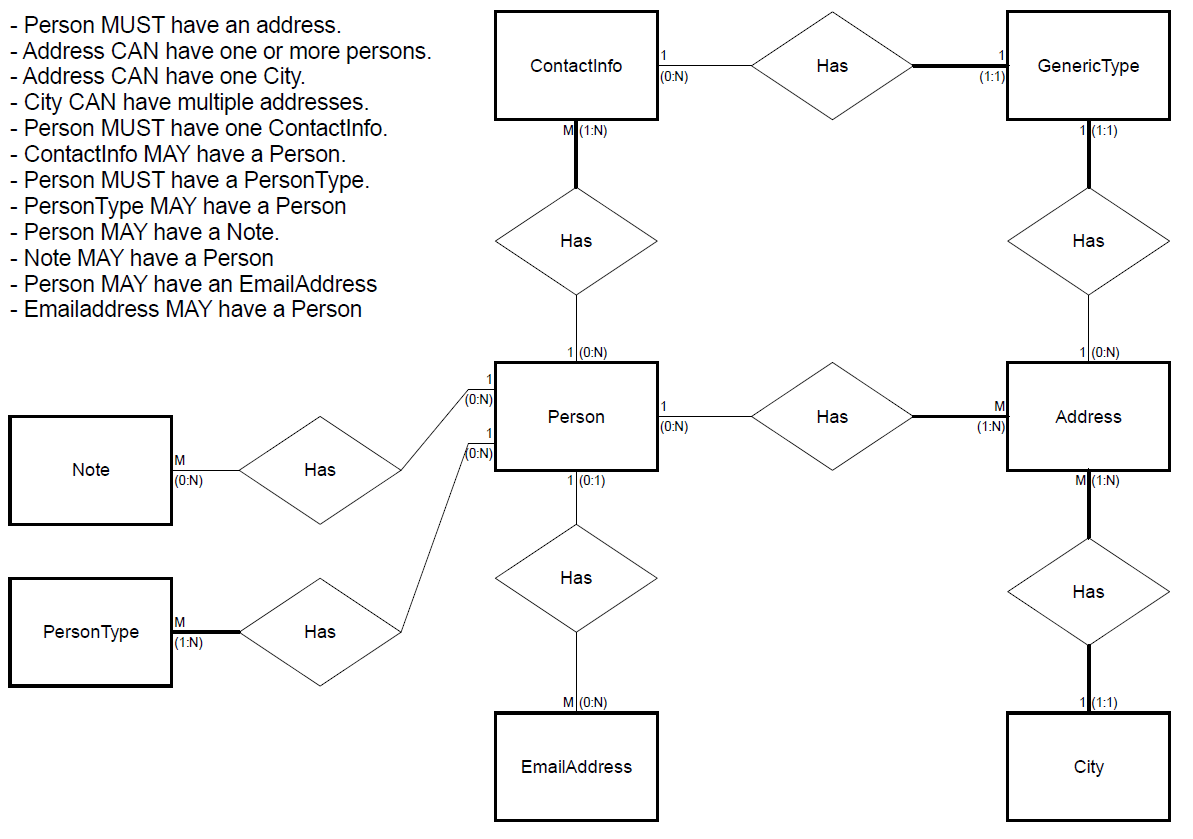


Figure 1: Entity Relationship Diagram of Domain Model

# Logical Schema from DDS Lite

The following Schema snippets has been generated from DDS-Lite and can be used to implement the Database in Visual Studio.

--

-- Target: Microsoft SQL Server

-- Syntax: isql /Uuser /Ppassword /Sserver -i\path\filename.sql

-- Date : Sep 29 2018 14:22

-- Script Generated by Database Design Studio 2.21.3

--

--

-- Create Table : 'Person'

-- fullName :

--

CREATE TABLE Person (

fullName CHAR NOT NULL,

CONSTRAINT pk\_Person PRIMARY KEY CLUSTERED (fullName))

GO

--

-- Create Table : 'City'

-- cityName :

-- zipCode :

--

CREATE TABLE City (

cityName CHAR(1) NOT NULL,

zipCode CHAR(1) NOT NULL,

CONSTRAINT pk\_City PRIMARY KEY CLUSTERED (cityName,zipCode))

GO

--

-- Create Table : 'Address'

-- streetAddress :

-- countryRegion :

-- cityName : (references City.cityName)

-- zipCode : (references City.zipCode)

-- fullName : (references Person.fullName)

--

CREATE TABLE Address (

streetAddress CHAR(1) NOT NULL UNIQUE,

countryRegion CHAR(1) NOT NULL,

cityName CHAR(1) NOT NULL,

zipCode CHAR(1) NOT NULL,

fullName CHAR NOT NULL,

CONSTRAINT pk\_Address PRIMARY KEY CLUSTERED (streetAddress,countryRegion),

CONSTRAINT fk\_Address2 FOREIGN KEY (cityName,zipCode)

REFERENCES City (cityName,zipCode)

ON DELETE NO ACTION

ON UPDATE CASCADE,

CONSTRAINT fk\_Address3 FOREIGN KEY (fullName)

REFERENCES Person (fullName)

ON DELETE NO ACTION

ON UPDATE CASCADE)

GO

--

-- Create Table : 'ContactInfo'

-- company :

-- phoneNumber :

-- fullName : (references Person.fullName)

--

CREATE TABLE ContactInfo (

company CHAR NOT NULL,

phoneNumber BIGINT NOT NULL UNIQUE,

fullName CHAR NOT NULL,

CONSTRAINT pk\_ContactInfo PRIMARY KEY CLUSTERED (company,phoneNumber),

CONSTRAINT fk\_ContactInfo FOREIGN KEY (fullName)

REFERENCES Person (fullName)

ON DELETE NO ACTION

ON UPDATE CASCADE)

GO

--

-- Create Table : 'EmailAddress'

-- email :

-- fullName : (references Person.fullName)

--

CREATE TABLE EmailAddress (

email CHAR(1) NOT NULL UNIQUE,

fullName CHAR NULL,

CONSTRAINT pk\_EmailAddress PRIMARY KEY CLUSTERED (email),

CONSTRAINT fk\_EmailAddress FOREIGN KEY (fullName)

REFERENCES Person (fullName)

ON UPDATE CASCADE)

GO

--

-- Create Table : 'Note'

-- noteId :

-- fullName : (references Person.fullName)

--

CREATE TABLE Note (

noteId BIGINT NOT NULL,

fullName CHAR NULL,

CONSTRAINT pk\_Note PRIMARY KEY CLUSTERED (noteId),

CONSTRAINT fk\_Note FOREIGN KEY (fullName)

REFERENCES Person (fullName)

ON UPDATE CASCADE)

GO

--

-- Create Table : 'PersonType'

-- type :

-- fullName : (references Person.fullName)

--

CREATE TABLE PersonType (

type CHAR NOT NULL,

fullName CHAR NOT NULL,

CONSTRAINT pk\_PersonType PRIMARY KEY CLUSTERED (type),

CONSTRAINT fk\_PersonType FOREIGN KEY (fullName)

REFERENCES Person (fullName)

ON DELETE NO ACTION

ON UPDATE CASCADE)

GO

--

-- Create Table : 'GenericType'

-- type :

-- streetAddress : (references Address.streetAddress)

-- countryRegion : (references Address.countryRegion)

-- company : (references ContactInfo.company)

-- phoneNumber : (references ContactInfo.phoneNumber)

--

CREATE TABLE GenericType (

type CHAR NOT NULL,

streetAddress CHAR(1) NULL,

countryRegion CHAR(1) NULL,

company CHAR NULL,

phoneNumber BIGINT NULL,

CONSTRAINT pk\_GenericType PRIMARY KEY CLUSTERED (type),

CONSTRAINT fk\_GenericType FOREIGN KEY (streetAddress,countryRegion)

REFERENCES Address (streetAddress,countryRegion)

ON UPDATE CASCADE,

CONSTRAINT fk\_GenericType2 FOREIGN KEY (company,phoneNumber)

REFERENCES ContactInfo (company,phoneNumber)

ON UPDATE CASCADE)

GO

--

-- Permissions for: 'public'

--

GRANT ALL ON Person TO public

GO

GRANT ALL ON City TO public

GO

GRANT ALL ON Address TO public

GO

GRANT ALL ON ContactInfo TO public

GO

GRANT ALL ON EmailAddress TO public

GO

GRANT ALL ON Note TO public

GO

GRANT ALL ON PersonType TO public

GO

GRANT ALL ON GenericType TO public

GO

-- Create Table : 'ContactInfo'

-- company :

-- phoneNumber :

-- fullName : (references Person.fullName)

--

CREATE TABLE ContactInfo (

company CHAR NOT NULL,

phoneNumber BIGINT NOT NULL UNIQUE,

fullName CHAR NOT NULL,

CONSTRAINT pk\_ContactInfo PRIMARY KEY CLUSTERED (company,phoneNumber),

CONSTRAINT fk\_ContactInfo FOREIGN KEY (fullName)

REFERENCES Person (fullName)

ON DELETE NO ACTION

ON UPDATE CASCADE)

GO

--

-- Create Table : 'EmailAddress'

-- email :

-- fullName : (references Person.fullName)

--

CREATE TABLE EmailAddress (

email CHAR(1) NOT NULL UNIQUE,

fullName CHAR NULL,

CONSTRAINT pk\_EmailAddress PRIMARY KEY CLUSTERED (email),

CONSTRAINT fk\_EmailAddress FOREIGN KEY (fullName)

REFERENCES Person (fullName)

ON UPDATE CASCADE)

GO

--

-- Create Table : 'Note'

-- noteId :

-- fullName : (references Person.fullName)

--

CREATE TABLE Note (

noteId BIGINT NOT NULL,

fullName CHAR NULL,

CONSTRAINT pk\_Note PRIMARY KEY CLUSTERED (noteId),

CONSTRAINT fk\_Note FOREIGN KEY (fullName)

REFERENCES Person (fullName)

ON UPDATE CASCADE)

GO

--

-- Create Table : 'PersonType'

-- type :

-- fullName : (references Person.fullName)

--

CREATE TABLE PersonType (

type CHAR NOT NULL,

fullName CHAR NOT NULL,

CONSTRAINT pk\_PersonType PRIMARY KEY CLUSTERED (type),

CONSTRAINT fk\_PersonType FOREIGN KEY (fullName)

REFERENCES Person (fullName)

ON DELETE NO ACTION

ON UPDATE CASCADE)

GO

-- Create Table : 'GenericType'

-- type :

-- streetAddress : (references Address.streetAddress)

-- countryRegion : (references Address.countryRegion)

-- company : (references ContactInfo.company)

-- phoneNumber : (references ContactInfo.phoneNumber)

--

CREATE TABLE GenericType (

type CHAR NOT NULL,

streetAddress CHAR(1) NULL,

countryRegion CHAR(1) NULL,

company CHAR NULL,

phoneNumber BIGINT NULL,

CONSTRAINT pk\_GenericType PRIMARY KEY CLUSTERED (type),

CONSTRAINT fk\_GenericType FOREIGN KEY (streetAddress,countryRegion)

REFERENCES Address (streetAddress,countryRegion)

ON UPDATE CASCADE,

CONSTRAINT fk\_GenericType2 FOREIGN KEY (company,phoneNumber)

REFERENCES ContactInfo (company,phoneNumber)

ON UPDATE CASCADE)

GO

--

-- Permissions for: 'public'

--

GRANT ALL ON Person TO public

GO

GRANT ALL ON City TO public

GO

GRANT ALL ON Address TO public

GO

GRANT ALL ON ContactInfo TO public

GO

GRANT ALL ON EmailAddress TO public

GO

GRANT ALL ON Note TO public

GO

GRANT ALL ON PersonType TO public

GO

GRANT ALL ON GenericType TO public

GO

# Domain Model with Domain Driven Design (DDD)

Following figure shows the Domain Model with DDD implemented in Visual Studio 2017.

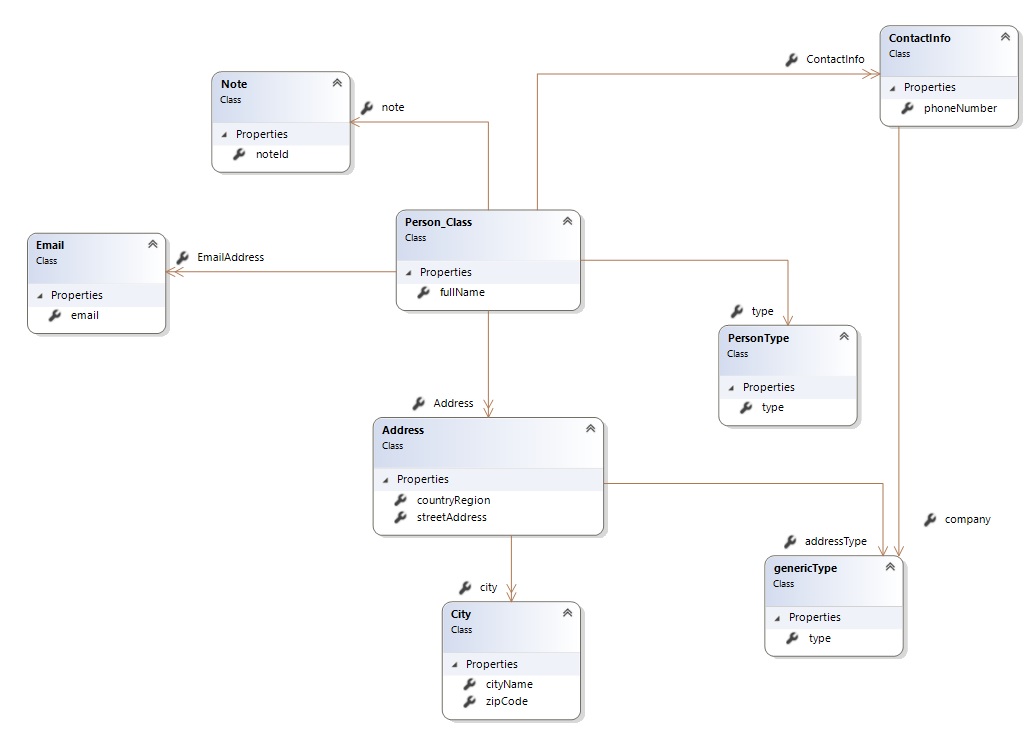


Figure 2: Doman Driven Design of Domain Model in UML

Person is the main object of interest; hence it would be defined as the primary entity. The person remains through all the states of the system and defines the lifetime.

## JSON Structure of Domain Model

This section explains the JSON structure of the domain model. The Main object in the root is Person and we have chosen EmailAddress, ContactInfo and Address to be array object.

Furthermore, we have a coded example of this model in the following snippet.

{

"Personkartotek": {

"Person": {

"fullName": "Ed Edd Eddy",

"PersonType": { "type": "One" },

"Note": { "noteId": 1 },

"EmailAddress": [

{

"email": "main\_email"

},

{

"email": "secondary\_email"

}

],

"ContactInfo": [

{

"phoneNumber": 11223344,

"GenericType": { "company": "Telenor" }

},

{

"phoneNumber": 55667788,

"GenericType": {"company": "TDC"}

}

],

"Address": [

{

"City": {

"cityName": "Aarhus",

"zipCode": "8200"

},

"countryRegion": "Danmark",

"GenericType": { "addressType": "Lejlighed" },

"streetAddress": "Kylling alle 11"

},

{

"City": {

"cityName": "Aarhus",

"zipCode": "8200"

},

"countryRegion": "Danmark",

"GenericType": { "addressType": "Sommerhus" },

"streetAddress": "Kylling alle 12"

}

]

}

}

}

Snippet 1: JSON Code Model of Domain Model

## XML Model of Domain Model

This section XML model of the Domain Model. We have defined person as the main element node. Attribute node with fullName and type has been attached to the person directly while EmailAddress, ContactInfo and Address has been designed as Meta Information to the person.

Following snippet shows an example of coded XML version of the model, it follows the same annotations from the JSON version.

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

<Person>

<fullName>Ed Edd Eddy</fullName>

<PersonType type="One"/>

<Note noteId="1"/>

<EmailAddress email="main\_email"/>

<EmailAddress email="secondary\_email"/>'

<ContactInfo

phoneNumber="11223344">

<GenericType company="Telenor"/>

</ContactInfo>

<ContactInfo

phoneNumber="55667788">

<GenericType company="TDC"/>

</ContactInfo>

<Address

streetAddress="Kylling Alle 11"

countryRegion="Danmark">

<City cityName="Aarhus" zipCode="8200"/>

<GenericType addressType="Lejlighed"/>

</Address>

<Address

streetAddress="Kylling Alle 12"

countryRegion="Danmark">

<City cityName="Aarhus" zipCode="8200"/>

<GenericType addressType="Sommerhus"/>

</Address>

</Person>

Snippet 2: XML Code Model of Domain Model

# Application Software Architecture

Following figure shows how the user can access the Personkartotek. The User must access it from the User Interface where the client connects to a server that contains the Personkartotek. Here the server has a Database which contains the given information of a Person from the Personkartotek.

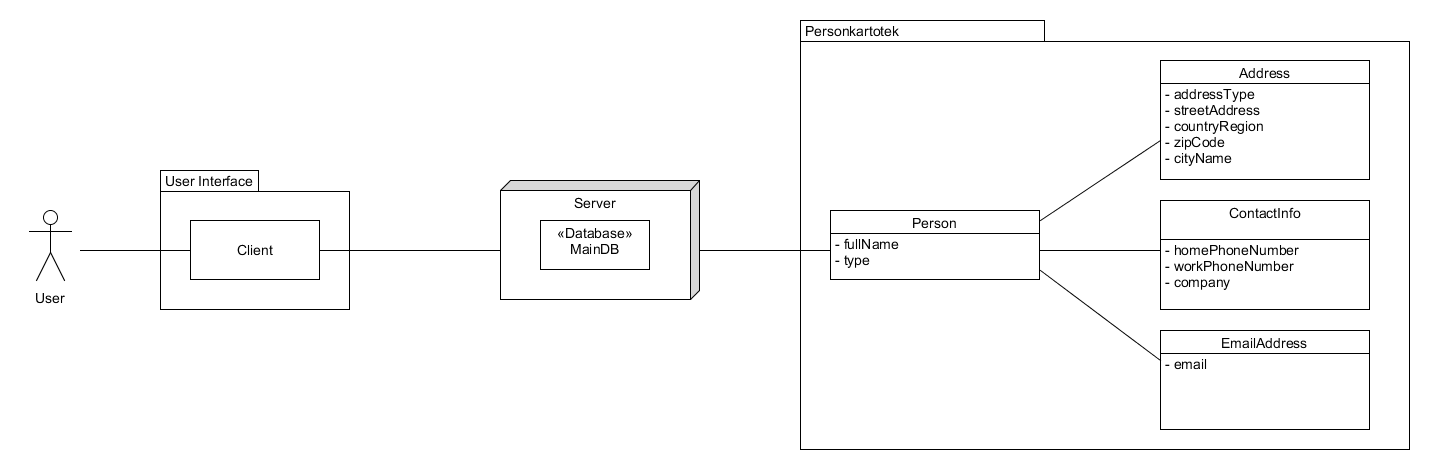


Figure 3: Application Software Architecture

# Scaffolding model

The scaffolding model shows the root and its objects in collections/containers, and they end with the triangular shaped objects.

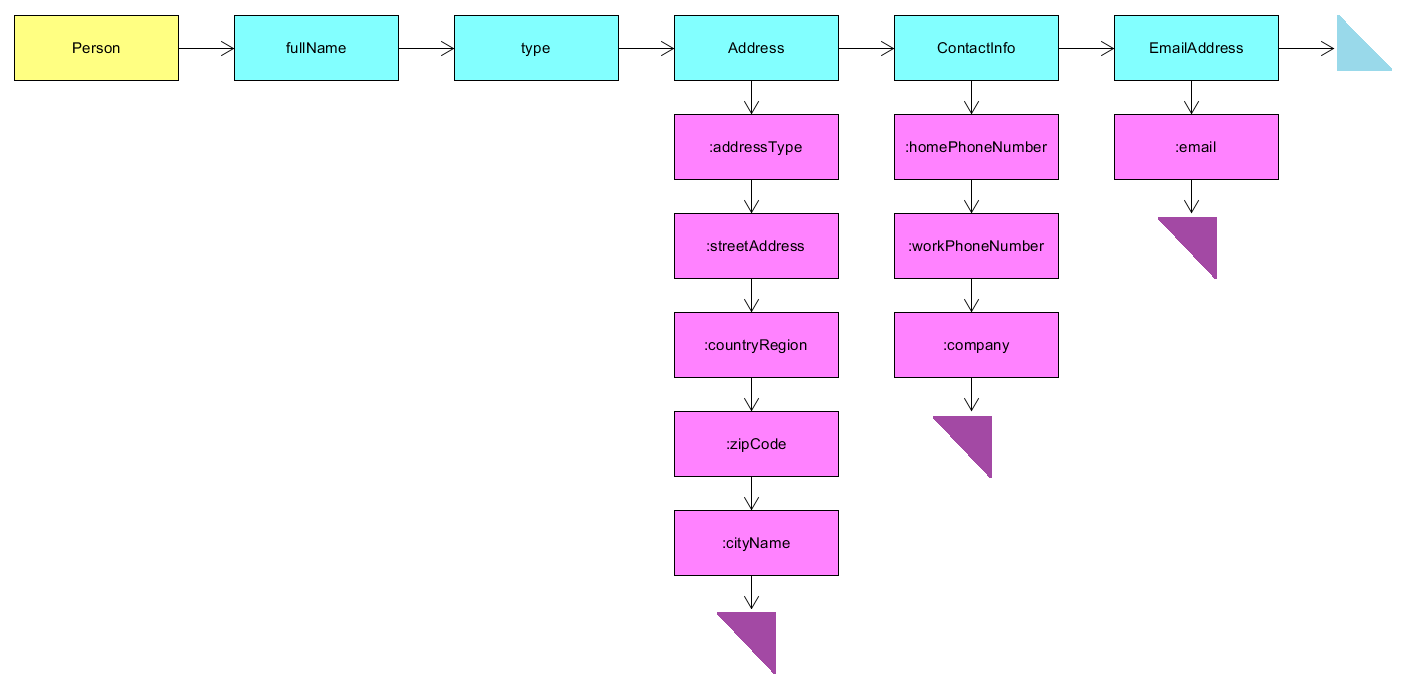


Figure 4: Scaffolding Model

# Graph-tree model

This is a Graph-Tree Model given from the Scaffolding Model.

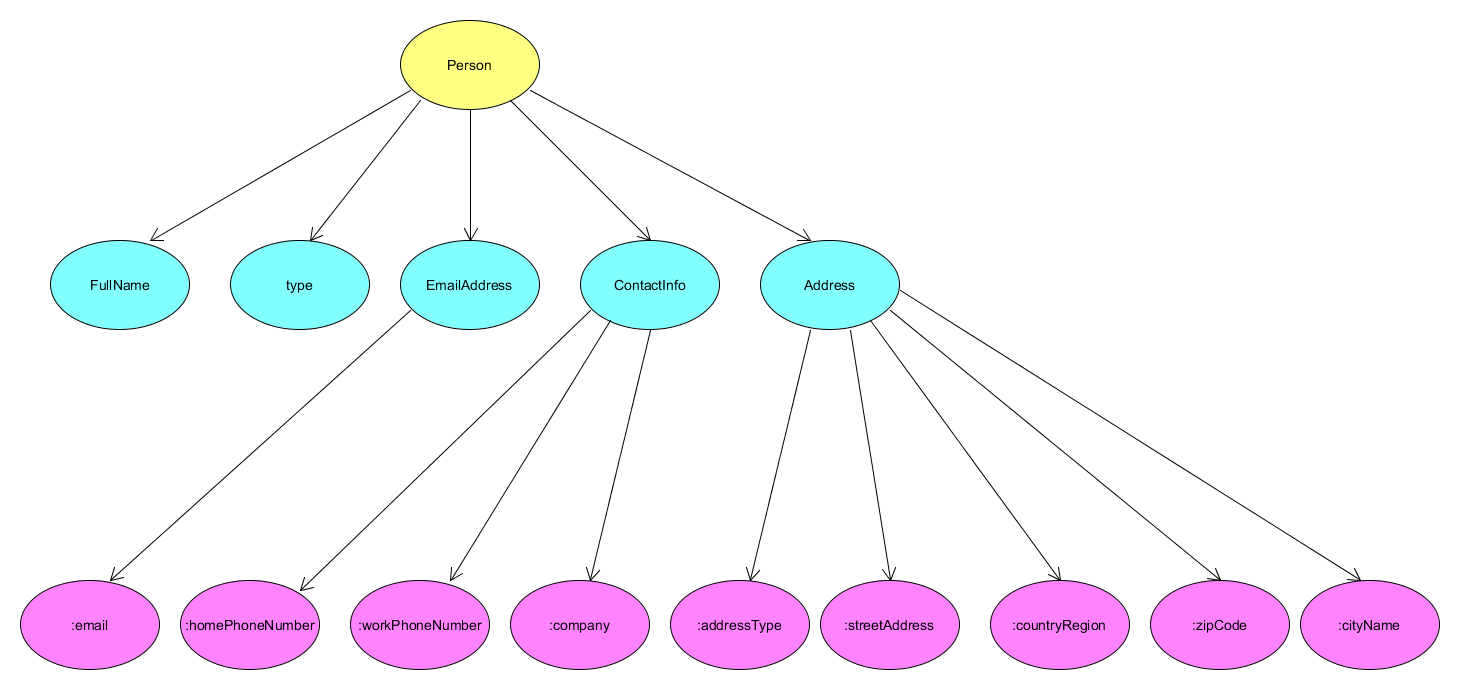


Figure 5: Graph-Tree Model