kadaster

GraphQL Tutorial

GraphQL Basics: Wat is het?

GraphQL: A query language for your API

- Net als een ‘traditionele’ API voldoet GraphQL aan de **HTTP** standaard (Ik haal data op met zgn. GET of POST requests)
- ‘Traditionele’ APIs vragen aan een gebruiker **specifieke** query parameters waarmee een **specifieke** vraag kan worden opgelost (“Geef mij alle informatie van adres met BAG identificatie XXXXXX)
- GraphQL vertelt een gebruiker welke data er beschikbaar is in de Graph, en de **gebruiker bepaalt** welke informatie hij/zij precies nodig is.

De kern: Typedefs

Wat betekent: welke data er beschikbaar is in de Graph?

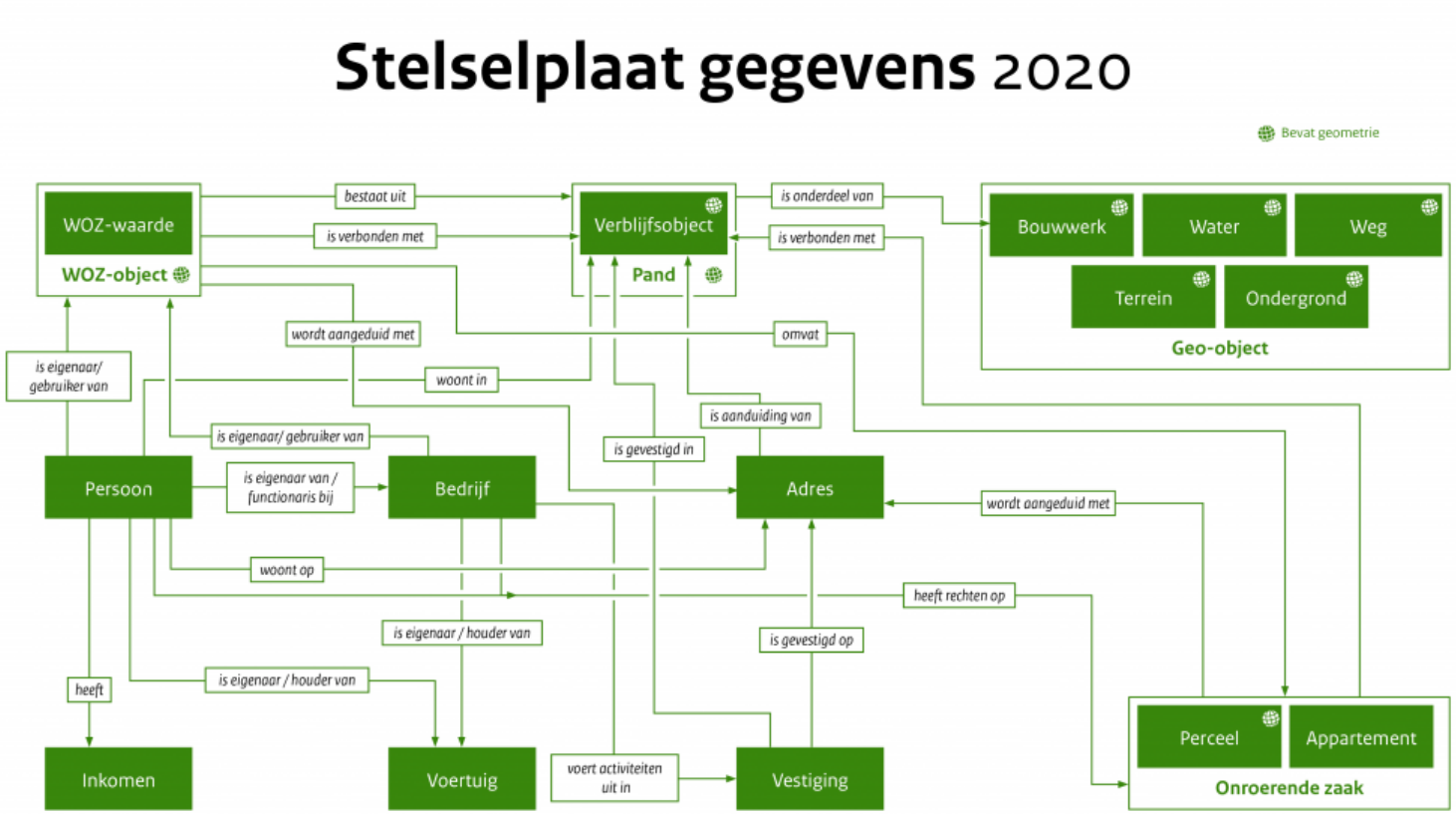
- **De kern van ieder GraphQL endpoint is het datamodel welke het endpoint serveert. Dit is beschreven in de zgn. typedefs. De typedefs beschrijven:**

1. **Objecten**
2. **Attributen**
3. **Relaties**

Typedefs (Objecten)

- Een object is vaak ook wel iets tastbaars;
  - Adres
  - Persoon
  - Bedrijf

- Een object kent in feiten twee belangrijke elementen
  - Historie / geldigheid (later meer)
  - Attributen



Typedefs (attributen)

- Attributen beschrijven de staat van een object (op een gegeven tijdstip)
- Attributen zijn primair gegevens van een bepaald datatype:
  - String
  - Int / Float
  - Date / Datetime
  - Boolean (True, False)

- Attributen kunnen enkelvoudig zijn of meervoudig
  → Waarbij dat laatste wordt uitgedragen met een []

Voorbeeld, voor een BAG2Verblijfsobject:
gebruiksdoel: [String]

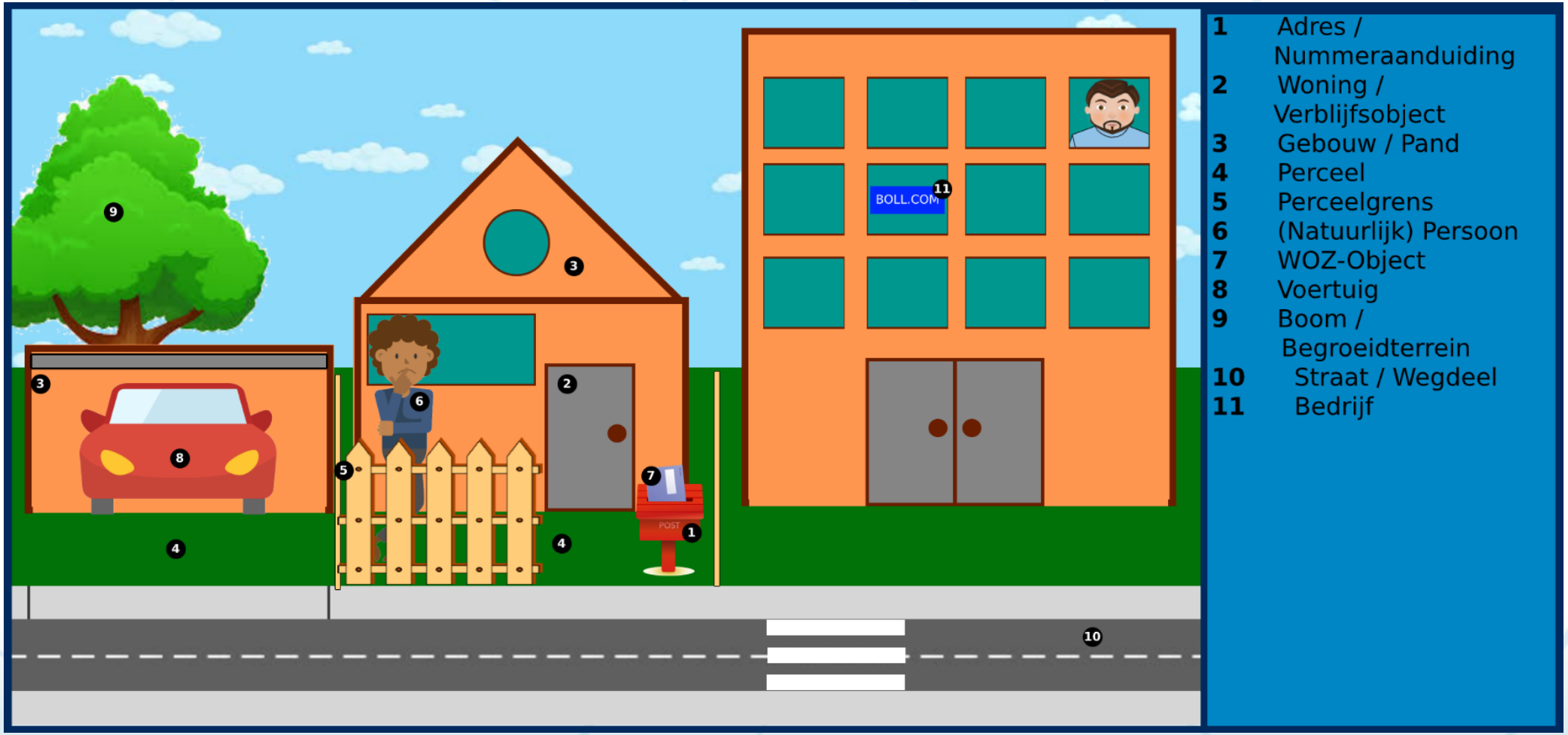
```
type BAG2Nummeraanduiding {
        lokaalID: String!
        postcode: String!
        huisnummer: Int!
        huisnummertoevoeging: String
        huisletter: String
        geconstateerd: Boolean
        begingeldigheid: Date
        eindgeldigheid: Date
        tijdstipregistratie: Datetime
        eindregistratie: Datetime
        voorkomenidentificatie: Int!
}
```

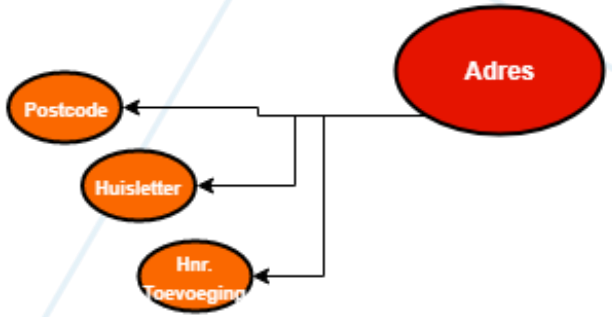
Typedefs (relaties)

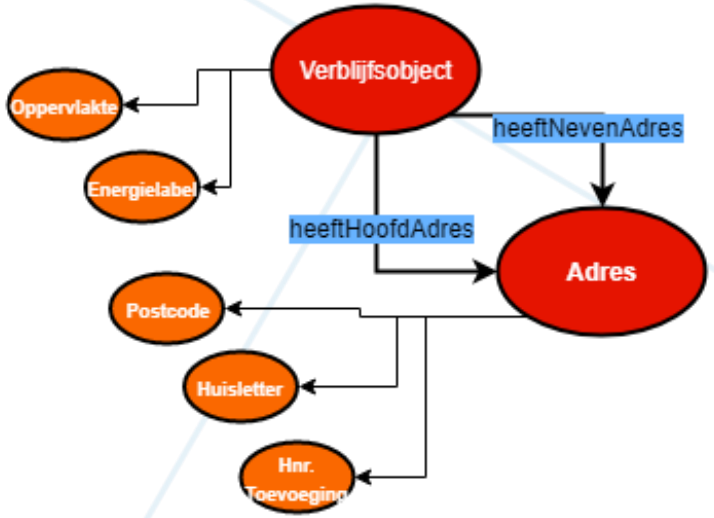
- **Vaak verwijzen objecten door naar andere objecten**
  - Een persoon woont op een adres (nummeraanduiding)
  - Een beperking rust op een perceel
  - Een BGT pand verwijst naar een BAG pand
- **Ook deze relaties zijn enkelvoudig of meervoudig**

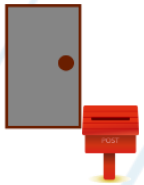
```
type BAG2Nummeraanduiding {
        lokaalID: String!
        postcode: String!
        huisnummer: Int!
        huisnummertoevoeging: String
        huisletter: String
        geconstateerd: Boolean
        begingeldigheid: Date
        eindgeldigheid: Date
        tijdstipregistratie: Datetime
        eindregistratie: Datetime
        voorkomenidentificatie: Int!
        hoofdadresvan: [BAG2AdresseerbaarObject]
        bewoners: [Persoon]
        wozobjecten [WOZObject]
}
```

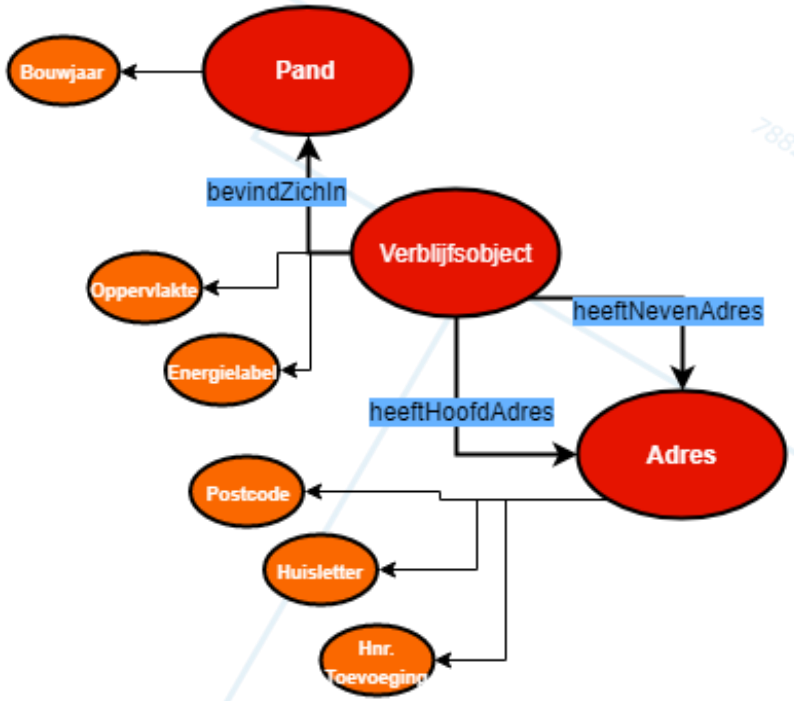
Object voor Object ontstaat er een Graph(QL) van de Nederlandse basisregistraties..

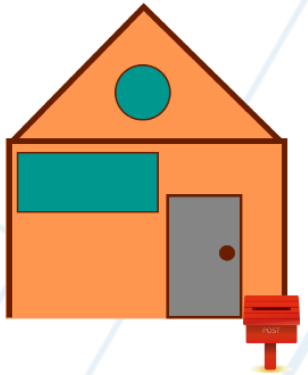


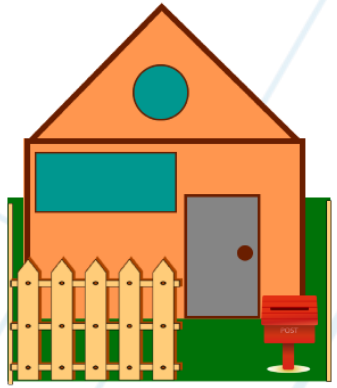

Adres
Postcode
Huisletter
Hnr.
Toevoeging

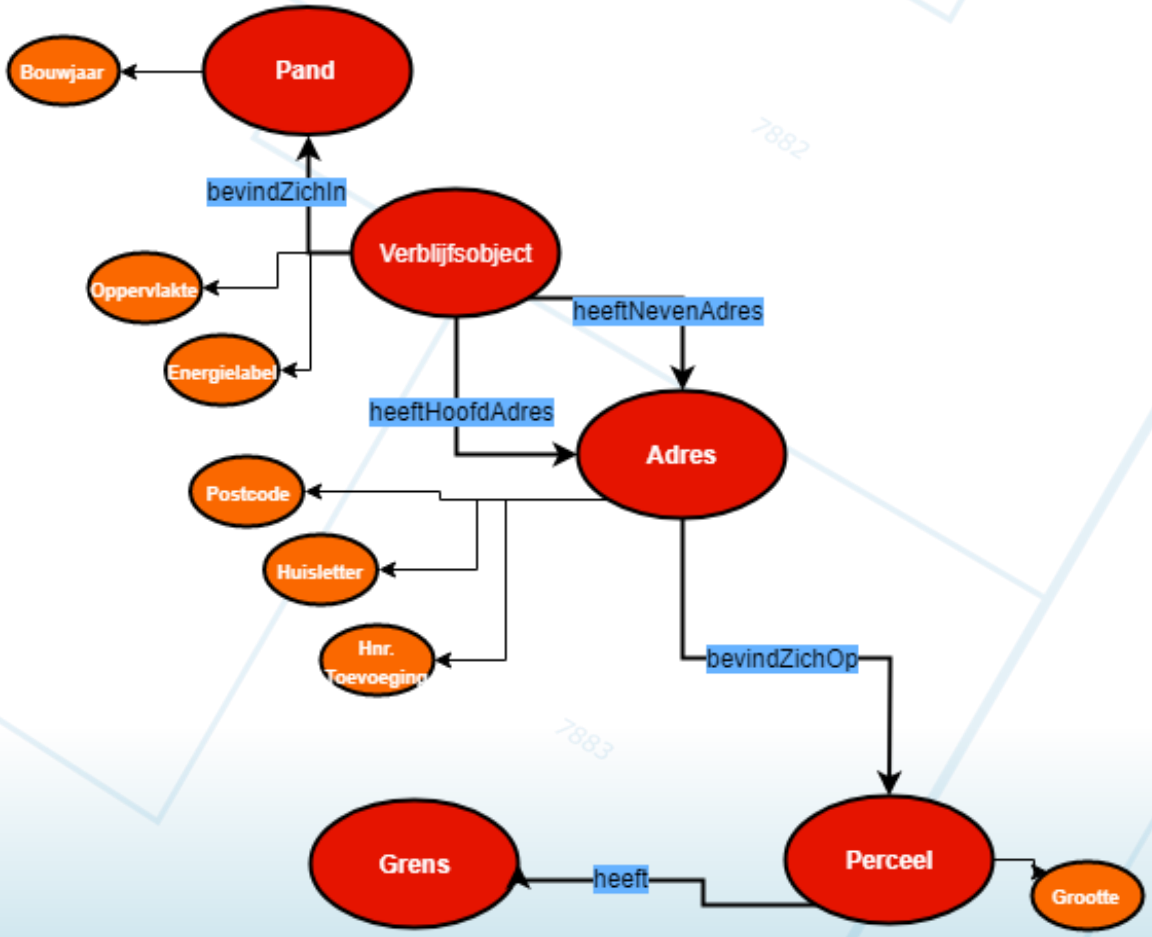

Verblijfsobject
Oppervlakte
Energielabel
heeftNevenAdres
heeftHoofdAdres
Adres
Postcode
Huisletter
Hnr.
Toevoeging

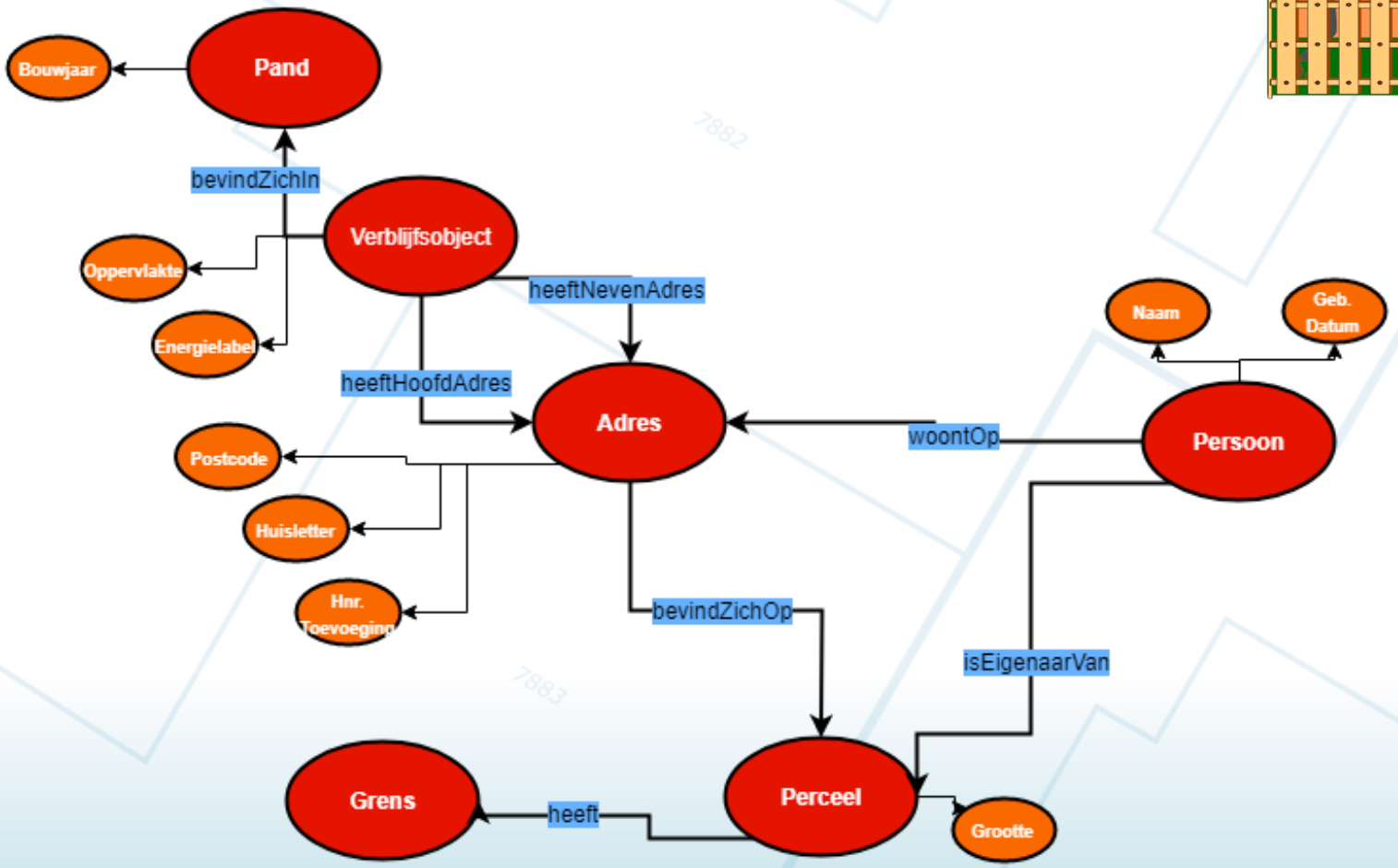


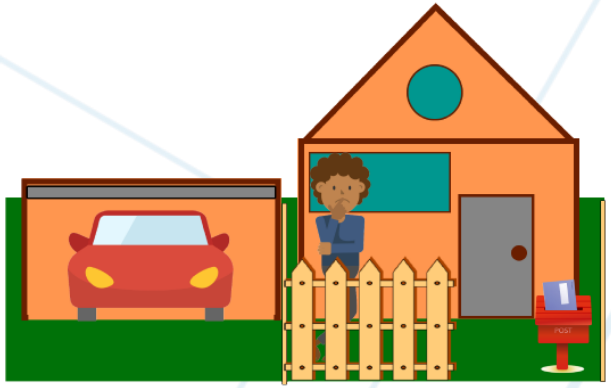

Pand
Bouwjaar
bevindZichIn
Verblijfsobject
Oppervlakte
Energielabel
heeftNevenAdres
heeftHoofdAdres
Adres
Postcode
Huisletter
Hnr.



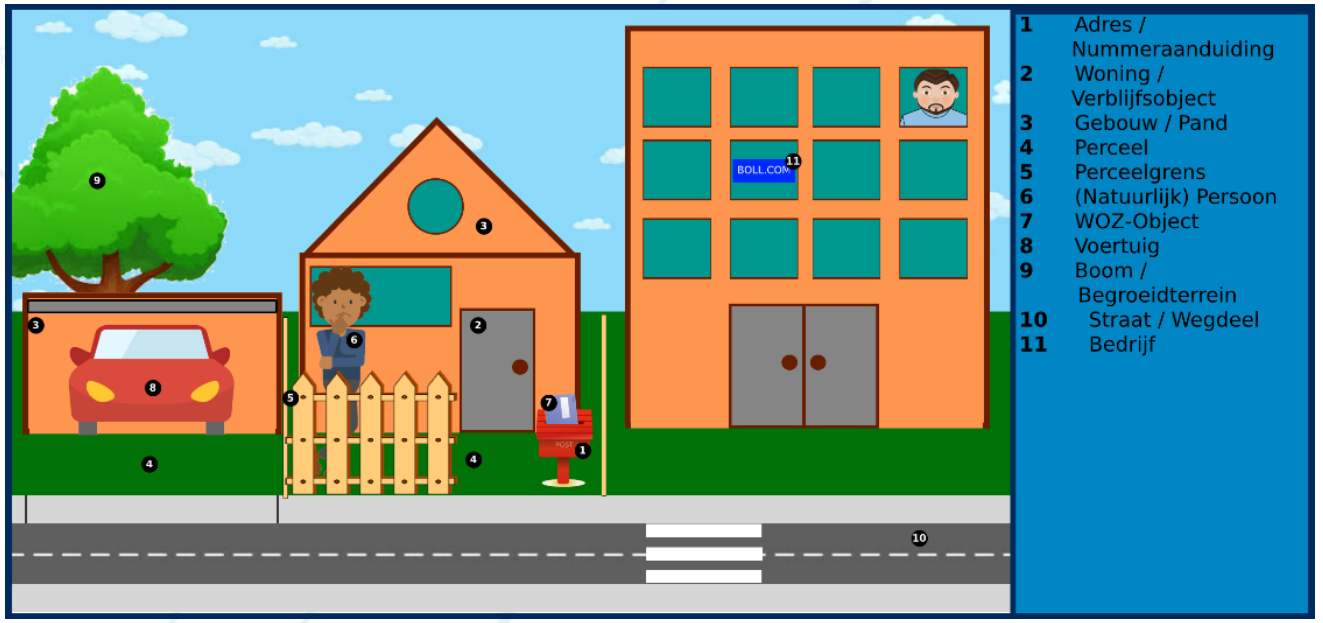



Pand
Bouwjaar
bevindZichIn
Verblijfsobject
Oppervlakte
Energielabel
heeftNevenAdres
heeftHoofdAdres
Adres
Postcode
Huisletter
Hnr. Toevoeging
bevindZichOp
Grens
heeft
Perceel
Grootte


Pand
Bouwjaar
bevindZichIn
Verblijfsobject
Oppervlakte
Energielabel
heeftNevenAdres
heeftHoofdAdres
Adres
Postcode
Huisletter
Hnr. Toevoeging
bevindZichOp
woontOp
Naam
Geb. Datum
Persoon
isEigenaarVan
Grens
heeft
Perceel
Grootte



WOZ Waarde
WOZ-Object
heeftBetrekkingOp
Pand
Bouwjaar
heeftBetrekkingOp
bevindZichIn
Verblijfsobject
heeftBetrekkingOp
Oppervlakte
heeftNevenAdres
Energielabel
isSubjectVan
Naam
Geb. Datum
heeftHoofdAdres
Adres
Persoon
Kenteken
Postcode
woontOp
staatOpNaamVan
Voertuig
Merk
Huisletter
Model
Hnr. Toevoeging
bevindZichOp
isEigenaarVan
isVerbondenAan
Grens
heeft
Perceel
Grootte


1 Adres / Nummeraanduiding
2 Woning / Verblijfsobject
3 Gebouw / Pand
4 Perceel
5 Perceelgrens
6 (Natuurlijk) Persoon
7 WOZ-Object
8 Voertuig
9 Boom / Begroeidterrein
10 Straat / Wegdeel
11 Bedrijf

WOZ Waarde
WOZ-Object
heeftBetrekkingOp
Pand
Bouwjaar
heeftBetrekkingOp
bevindZichIn
Verblijfsobject
heeftBetrekkingOp
Oppervlakte
heeftNevenAdres
Energielabel
isSubjectVan
Naam
Geb. Datum
heeftHoofdAdres
Adres
Persoon
Kenteken
Postcode
woontOp
staatOpNaamVan
Voertuig
Merk
Huisletter
Model
isGevestigdOp
staatOpNaamVan
Hnr. Toevoeging
bevindZichOp
isEigenaarVan
isVerbondenAan
Bedrijf
Grens
heeft
Perceel
Grootte
Naam
SBI Code

Een GraphQL query: de ingang

- Het datamodel beschrijft dus welke objecten, attributen en relaties er in de data zijn te vinden. Maar hoe haal ik de door mij gezochte objecten eruit?
- Als ontwikkelaar definiëren we de ingangen van een zoektocht in het zgn. Query object. Deze beschrijft:

- Het **type object** dat kan worden opgehaald
- De **argumenten** waarmee deze set aan objecten wordt beperkt

Voorbeelden:

1. Bag2woonplaats(naam: String): [BAG2Woonplaats] **GEEF MIJ ALLE WOONPLAATS OBJECTEN MET DEZE naam**
2. Bgtbord(first: Int): [BGTBord] **GEEF MIJ DE EERSTE first BORD OBJECTEN**
3. bag2verblijfsobject(filter: String): [BAG2Verblijfsobject] **GEEF MIJ ALLE VERBLIJFSOBJECTEN DIE VOLDOEN AAN DIT filter**

Een GraphQL query: de query

```
query TutorialQuery {
  bag2nummeraanduiding(identificatiecode: "0518200001646739", peilDatum:"2021-05-10"){
    lokaalID
  }
}
```

1. Ingang (Query object)
2. Attributen

Een GraphQL query: de query

```
query TutorialQuery {
  bag2nummeraanduiding(identificatiecode: "0518200001646739", peilDatum:"2021-05-10"){
    lokaalID
    postcode
    huisnummer
    hoofdadresvan{
      lokaalID
      oppervlakte
      gebruiksdoel
    }
  }
}
```

1. Ingang (Query object)
2. Attributen
3. Relaties

Historische gegevens en peilDatum

- Tot op heden hebben we het mogelijk gemaakt om ook historische gegevens rondom objecten te bevragen (Wat zijn alle ‘voorkomens’ van een bepaald object)
- In de praktijk kennen objecten binnen de Nederlandse basisregistratie vaak een Formele- en Materiële historie
  → Zie bijv. https://imbag.github.io/praktijkhandleiding/artikelen/hoe-bepaal-ik-welke-gegevens-in-een-levenscyclus-van-een-object-geldig-zijn

- We abstraheren deze formele- en materiële historie weg door de gebruiker een mogelijkheid te geven een peilDatum op te voeren op iedere initiële bevraging in de vorm ‘YYYY-MM-DD’.
- Zo krijg je alleen de actuele voorkomens op deze datum te zien.

Complexiteit van een GraphQL query

- GraphQL is – onder water – een set aan queries / bevragingen die serieel (achter elkaar) worden afgevuurd)
- Iedere sprong naar nieuwe object(en) is één bevraging
  → Dit werkt dan ook exponentiëel
- Indien een totale query langer duurt dan 60 seconden kappen wij hem af en krijgt de gebruiker een Time-out te zien.