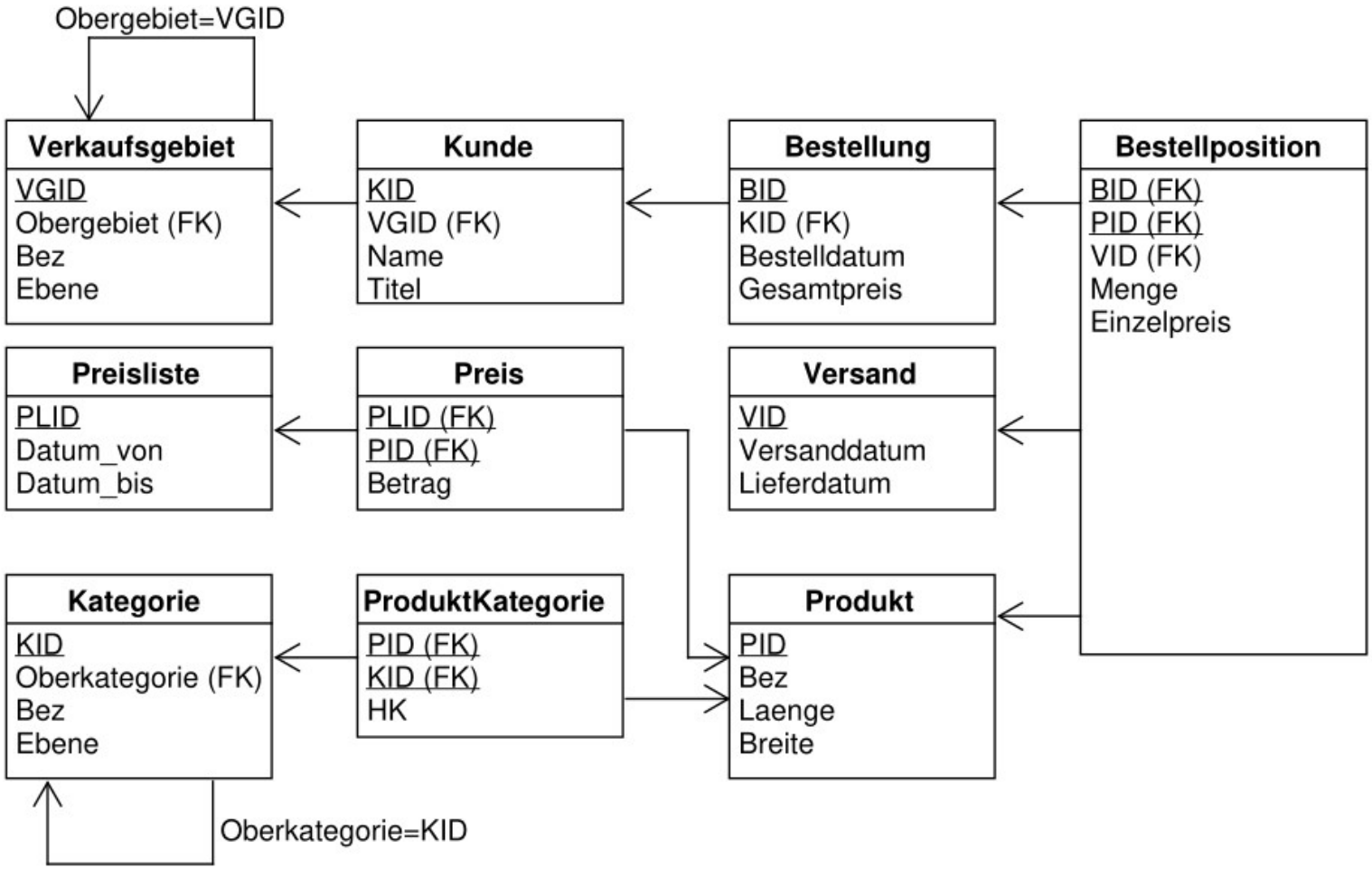
Implementierung von GraphQL-Abfragen mittels JSON-Funktionen in Postgres
GraphQL: https://graphql.org/

Datenmodell



P5-A1: Abfrage Ohne Schachtelung

Prof. Dr. Ingo Claßen

Mittwoch, 23. März 2022 15:11

Die Abfrage wird in der GraphQL-Syntax angegeben

```
query {
  kunde(kid: "10001") {
    name
    titel
  }
}
```

Folgendes Ergebnis sollte dann geliefert werden:

```
{
    "name": "Eck",
    "titel": "Dr."
}
```

Der Python-GraphQLParser liefert einen Syntaxbaum (AST=Abstract Sytax Tree)

```
from graphql.parser import GraphQLParser
parser = GraphQLParser()
ast1 = parser.parse("""
query {
  kunde(kid: "10001") {
    name
    titel
  }
}
""")
q1 = ast1.definitions[0].selections[0]
print(q1)
```

```
<Field: name=kunde, arguments=[<Argument: name=kid, value=10001>],
    selections=[
        <Field: name=name>,
        <Field: name=titel>
    ]
>
```

Gehen sie bei der Implementierung wie folgt vor:

- AST traversieren und eine Postgres-Query erzeugen, die JSON zurückliefert
- Diese Query an die Datenbank senden und Ergebnis ausgeben
- Beachten sie, dass sie die Namen der Tabellen und Spalten aus der GraphQL-Query entnehmen und nicht hart kodieren

P5-A2: Abfrage mit Schachtelung

Prof. Dr. Ingo Claßen

Mittwoch, 23. März 2022 15:35

Abfrage

```
query {
  kunde(kid: "10001") {
    name
    bestellung {
      bestelldatum
      gesamtpreis
    }
  }
}
```

AST

```
<Field: name=kunde, arguments=[<Argument: name=kid, value=10001>],
    selections=[
        <Field: name=name>,
        <Field: name=bestellung,
            selections=[
                <Field: name=bestelldatum>,
                <Field: name=gesamtpreis>
            ]
        >
    ]
>
```

Diese Form der Abfrage soll für beliebige Tabellen funktionieren, zwischen denen eine 1-zu-n-Beziehung besteht.

Ergebnis


```
{
    "name": "Eck",
    "bestellung": [
        {
            "gesamtpreis": 725.25,
            "bestelldatum": "2018-06-27"
        },
        {
            "gesamtpreis": 145.99,
            "bestelldatum": "2019-01-31"
        },
        {
            "gesamtpreis": 28.00,
            "bestelldatum": "2019-04-17"
        },
        {
            "gesamtpreis": 871.66,
            "bestelldatum": "2019-06-28"
        },
        {
            "gesamtpreis": 343.94,
            "bestelldatum": "2019-10-28"
        },
        {
            "gesamtpreis": 36.00,
            "bestelldatum": "2019-11-01"
        }
    ]
}
```

P5-A3: Abfrage mit Schachtelung und Bedingung

Prof. Dr. Ingo Claßen

Mittwoch, 23. März 2022 15:35

Abfrage

```
query {
  kunde(kid: "10001") {
    name
    bestellung (bestelldatum: "2019-01-31") {
      bestelldatum
      gesamtpreis
    }
  }
}
```

Ergebnis

```
{
    "name": "Eck",
    "bestellung": [
        {
            "gesamtpreis": 145.99,
            "bestelldatum": "2019-01-31"
        }
    ]
}
```

AST

```
<Field: name=kunde, arguments=[<Argument: name=kid, value=10001>],
    selections=[
        <Field: name=name>,
        <Field: name=bestellung, arguments=[<Argument: name=bestelldatum, value=2019-01-31>],
            selections=[
                <Field: name=bestelldatum>,
                <Field: name=gesamtpreis>
            ]
        >
    ]
>
```

Diese Form der Abfrage soll für beliebige Tabellen funktionieren, zwischen denen eine 1-zu-n-Beziehung besteht.

P5-A4: Abfrage mit n-zu-m-Beziehung

Prof. Dr. Ingo Claßen

Mittwoch, 23. März 2022 15:35

Abfrage

```
query {
  kunde(kid: "10001") {
    name
    bestellung {
      produkte {
          bez
          laenge
      }
      bestelldatum
      gesamtpreis
    }
  }
}
```

AST

```
<Field: name=kunde, arguments=[<Argument: name=kid, value=10001>],
    selections=[
        <Field: name=name>,
        <Field: name=bestellung,
            selections=[
                <Field: name=produkte,
                    selections=[
                        <Field: name=bez>,
                        <Field: name=laenge>
                    ]
                >,
                <Field: name=bestelldatum>,
                <Field: name=gesamtpreis>
            ]
        >
    ]
>
```

Ergebnis

```
{
    "name": "Eck",
    "bestellung": [
        {
            "produkt": [
                {
                    "bez": "Lucca-Esstisch",
                    "laenge": null
                },
                {
                    "bez": "Stehtisch rund",
                    "laenge": null
                },
                ...
            ],
            "gesamtpreis": 725.25,
            "bestelldatum": "2018-06-27"
        },
        ...
    ]
}
```

Diese Form der Abfrage soll für beliebige Tabellen funktionieren, zwischen denen eine n-zu-m-Beziehung besteht.