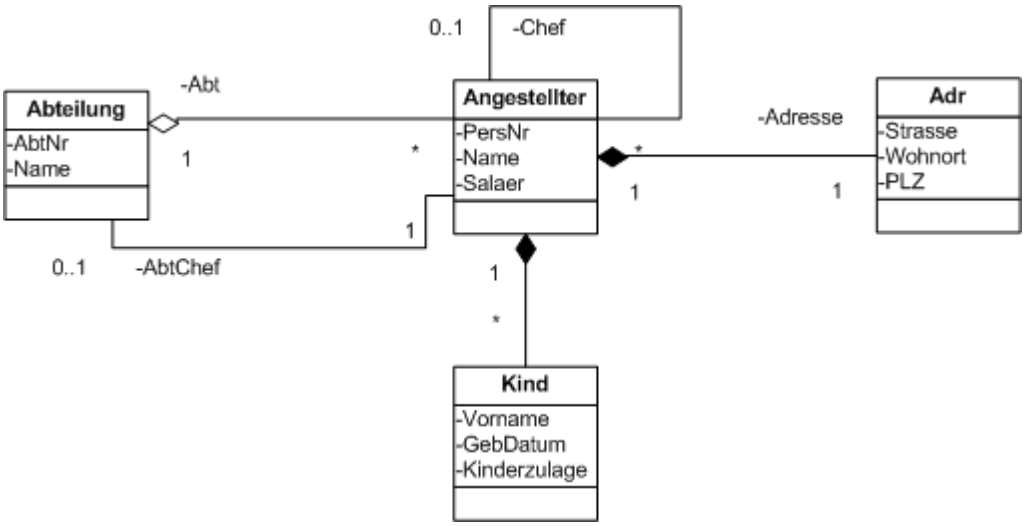
Aufgabe: Objektrelationale Erweiterungen (Oracle)

1 Einleitung

Lernziele	Objektrelationalen Erweiterungen von Oracle kennenlernen - Einige objektrelationale Erweiterungen (von Oracle) beherrschen - Arrays (VARARRAY) in Oracle kennenlernen - Nested Tables erstellen und abfragen - Eigene Typen mit Methode ergänzen.
Ausgangslage	Ein UML Klassendiagramm (Angestellter Verwaltung).
Aufgabenstellung	Ein UML Klassendiagramm objektrelational in einer Oracle DB implementieren.
Hilfsmittel	Oracle SQL Developer
Erwartete Resultate	Korrekte SQL Script Datei mit Befehlen für die Implementierung
Zeitbedarf	120 min
Lösungsdatei	Ausführbare SQL Script Dateien.

2 Schema (UML)



3 UML Beziehungen

Die UML Beziehungen werden wie folgt abgebildet:

Aggregation ➔ als Referenz

Composition 1:1 ➔ Objekttyp als Spaltenobjekt

Composition 1:n ➔ als Nested Table oder VARRAY

Assoziation ➔ als Referenz

4 Create / Insert

Implementation der Klassen Abteilung, Angestellter und Adr mit den dazugehörenden Assoziationen.

4.1 Typen deklarieren

- Deklarieren Sie die Typen AdrTyp, AbteilungTyp, AngestellterTyp.
- **Achtung**: Das Konzept Kind wird in dieser Aufgabe noch nicht implementiert.
- Modellieren Sie die Assoziationen mit Referenzen. So hat z.B. der Typ AbteilungTyp ein Attribut AbtLeiter vom Typ "Referenz auf Angestellter".
- Ergänzen Sie den Typ AngestellterTyp mit einer Map-Methode für die Sortierordnung mit den Ordnungskriterien: ***Name, Salaer***.

4.2 Tabellen erstellen

Tabelle	Attribute
Abteilung	AbtNr (PK)
	AbtName NOT NULL, UNIQUE
Angesteller	PersNr (PK)
	Abt NOT NULL
	NAME NOT NULL

4.3 Daten abfüllen

Abteilung

AbtNr	Name
1	Entwicklung
2	Marketing

Angestellter

PersNr	Name	Salaer	Abt	Chef	Addresse (Strasse, Wohnort, PLZ)
110	Müller	8000	1	Null	Seestrasse, Zürich, 8008
120	Meier	7000	1	110	Breitenstrasse, Horgen, 8080
130	Krauer	9000	1	110	Hauptstrasse, Au, 8081

4.4 Hinweis und Tipps

Bei der Modellierung der Assoziationen werden Sie auf ein Problem stossen, hervorgerufen durch zirkuläre Referenzen:

AbteilungTyp referenziert *AngestellterTyp* und *AngestellterTyp* referenziert *AbteilungTyp*. Dieses Problem lässt sich mit Forward Declarations lösen:

```
CREATE TYPE AbteilungTyp; --forward declaration
CREATE OR REPLACE TYPE AngestellterTyp AS OBJECT (
PersNr Integer,
Abt REF AbteilungTyp
);
CREATE OR REPLACE TYPE AbteilungTyp AS OBJECT (
AbtNr Integer,
AbtChef REF AngestellterTyp
)
```

5 Abfragen

Erstellen Sie für folgende Ausgaben die Abfragen:

- Liste: Name des Angestellten, Name seines Chefs.
- Liste: Name des Angestellten, Name seines Chefs, sortiert.

  ... from Angestellter t order by value(t);
- Liste der Angestellten der Abteilung "Entwicklung".
- Liste: Name und Wohnort der Angestellten der Abteilung "Entwicklung".
- Müller (PersNr 110) wird Leiter der Abteilung "Marketing". Führen Sie diese Änderung mit dem Update-Befehl auf der Tabelle Abteilung durch.

6 VARRAY

Erweitern Sie die Angestellten, damit jeder eine Liste von zehn TelefonNr hat. Verwenden Sie dazu ein VARRAY.

Testen Sie die Erweiterung:

- Fügen Sie einen neuern Angestellten ein. Dieser Angestellte sollte zwei Telefonnummern besitzen.
- Geben Sie eine List aus, in der alle Angestellten mit ihren Telefonnummern aufgelistet sind.

6.1 Hinweise und Tipps

Um den Typ *AngestellterTyp* zu ändern, können sie den Befehl ALTER TYPE ADD ATTRIBUTE verwenden.

7 Nested Table

In dieser Aufgabe erweitern Sie das DB-Modell.

Zusätzlich soll die Klasse Kind mit der Aggregation Angestellter-Kind als Nested Table implementiert werden.

- Definieren Sie zuerst einen Typ *KindTyp*, anschliessend einen Tabellentyp *KindTabelleTyp*.
- Erweitern Sie dann den Typ AngestellterTyp um das Attribut Kinder vom Typ KindTabelleTyp (ALTER TYPE.. ADD ATTRIBUTE..).
- Führen Sie einen Update auf die Tabelle *Angestellter* aus. Das Attribut *Kinder* soll dabei mit dem Default-Konstruktor initialisiert werden.

Auffüllen der Daten in die Tabelle *Angestellter*:

- Schenken Sie einem Mitarbeiter 2 Kinder.
- Schenken Sie einem anderen Mitarbeiter 1 Kind.

Abfragen mit dem Table Operator:

- Liste mit den Daten aller Kinder.
- Liste mit *PersNr*, *Name* und Kinderdaten aller Angestellten mit Kindern, die eine Kinderzulage grösser 20 haben.

8 Methoden

- Definieren Sie eine Methode *sumKinderzulagen()*. Diese Methode traversiert die Kinder eines Angestellten und summiert die einzelnen Kinderzulagen.
- Definieren Sie eine Methode *compareSalary()*. Diese soll den Lohn des aufrufenden Mitarbeiters mit dem Lohn des Chefs des Mitarbeiters vergleichen.
- Erstellen Sie die Abfrage für folgende Ausgaben:
- Liste der Angestellten mit PersNr, Name, Salaer, sumKinderzulagen()
- Liste der Angestellten mit PersNr, Name, Salaer, compareSalary()

8.1 Hinweise und Tipps

- Folgendermassen können Sie über alle Kinder iterieren:
- FOR i in 1..SELF.Kinder.COUNT LOOP
- Da PL/SQL das Navigieren über REF Attribute nicht erlaubt, müssen sie den Chef mit Hilfe der folgenden Funktion selektieren:
- UTL_REF.SELECT_OBJECT(SELF.Chef, chefobj);