

Hinweise:

Bearbeitungszeit: 60 Minuten (DI: 75 Minuten inkl. der Aufgabe zu Datenbanken 1)

Erlaubte Hilfsmittel: im Anhang, sonst keine

Diese Klausur besteht aus 7 Aufgaben (Seiten 2 bis 8) und einem Anhang (Seiten 9 bis 11). (DI: zusätzlich noch eine Seite zu Datenbanken 1)

Bitte notieren Sie Ihre Antworten ausschließlich auf dem Aufgabenblatt! Bei Bedarf benutzen Sie die Rückseite! Für Skizzen und Entwürfe steht ebenfalls die Rückseite zur Verfügung. Entwürfe, die nicht gewertet werden sollen, sind durchzustreichen.

Hinter jeder Aufgabe steht die Anzahl der Bewertungseinheiten (BE), die für diese Aufgabe vergeben werden. Es gibt insgesamt 35 BE. Zum Bestehen benötigen Sie also 17,5 BE (DI: Ihre BE aus diesem Klausurteil werden zu 80 % gewertet, die aus dem Datenbanken 1-Klausurteil zu 20 %, insgesamt brauchen Sie die Hälfte). Die Zeitangabe entspricht der empfohlenen Zeit, die Sie sich mit der Aufgabe beschäftigen sollten, um in der Bearbeitungszeit von 60 (75) Minuten fertig zu werden. Sie ist nur als Richtwert zu sehen.

Viel Erfolg!

Aufgabe 1: Thema: Relationale und objektorientierte Realisierung eines ER-Modells (6 BE, 10 min)

Formulieren Sie zum ER-Modell in Anhang I ein objektorientiertes Modell nach dem ODMG-Standard: Geben Sie alle wesentlichen Elemente an (mit Typangabe)! Die Beachtung der genauen Syntax ist nicht erforderlich.

Lösung:

```
Objekt Kunde: Attribute int Nr, String Name, int Alter;
                relationship Kauf kauft (inverse Kauf::kauft).
Objekt Produkt: Attribute int Nr, String Name;
                relationship Kauf wirdGekauft (inverse Kauf::wirdGekauft).
Objekt Verkäufer: Attribute int Nr, String Name, int Alter;
                relationship Kauf verkauft (inverse Kauf::verkauft).
Objekt Kauf: Attribute float Preis, Date Datum;
                relationship Kunde kauft (inverse Kunde::kauft);
                relationship Produkt wirdGekauft (inverse Produkt::wirdGekauft);
                relationship Verkäufer verkauft (inverse Verkäufer::verkauft);
```

Alternative Lösungen:

```
Objekt Kunde: Attribute int Nr, String Name, int Alter;
                relationship Kauf kauft (inverse Kauf::kunde).
Objekt Produkt: Attribute int Nr, String Name;
                relationship Kauf wirdGekauft (inverse Kauf::produkt).
Objekt Verkäufer: Attribute int Nr, String Name, int Alter;
                relationship Kauf verkauft (inverse Kauf::verkäufer).
Objekt Kauf: Attribute float Preis, Date Datum;
                relationship Kunde kunde (inverse Kunde::kauft);
                relationship Produkt produkt (inverse Produkt::wirdGekauft);
                relationship Verkäufer verkäufer (inverse Verkäufer::verkauft);
```

Andere Namen werden in den relationships von Kauf ebenfalls akzeptiert. Eine BE Abzug gibt es, wenn die in Anhang I angegebenen relationship-Namen kauft, wirdGekauft und verkauft in keiner beteiligten Objektklasse aufgeführt sind.

Andere Typnamen werden ebenfalls akzeptiert, solange sie sinnvoll sind.

Die Definition von set-relationships (gemäß der natürlichen Bedeutung der Begriffe) wird auch akzeptiert, da die Vielfachheit der Beziehungen in Anhang I nicht angegeben wurde. Beispiel:

```
Objekt Kunde: Attribute int Nr, String Name, int Alter;
                relationship set (kauft) (inverse Kauf::kunde).
Objekt Produkt: Attribute int Nr, String Name;
                relationship set (wirdGekauft) (inverse Kauf::produkt).
Objekt Verkäufer: Attribute int Nr, String Name, int Alter;
                relationship set (verkauft) (inverse Kauf::verkäufer).
Objekt Kauf: Attribute float Preis, Date Datum;
                relationship Kunde kunde (inverse Kunde::kauft);
                relationship Produkt produkt (inverse Produkt::wirdGekauft);
                relationship Verkäufer verkäufer (inverse Verkäufer::verkauft);
```

Aufgabe 2: Thema: Persistenzkonzepte

(6 BE, 10 min)

- a) Erklären Sie den Unterschied zwischen Persistenzfähigkeit und Persistenz!
(2 BE, 4 min)
- b) Erklären Sie den Unterschied zwischen Persistenzfähigkeit durch Vererbung und durch explizite Kennzeichnung! Welche Variante wird in der ODMG-Spezifikation für Java vorgenommen? Welche Variante verfolgt JDO?
(4 BE, 6 min)

Lösung:

- a) Eine Klasse heißt persistenzfähig, wenn man Instanzen von ihr persistent machen kann. Instanzen von nicht persistenzfähigen Klassen sind immer transient. Ein Objekt heißt persistent, wenn es in der Datenbank abgespeichert ist.
- b) In der Persistenzfähigkeit durch Vererbung ist jede Spezialisierung einer persistenzfähigen Klasse automatisch auch persistenzfähig. Bei der Persistenzfähigkeit durch explizite Kennzeichnung muss die Persistenzfähigkeit für jede Klasse gesondert deklariert werden. In ODMG für Java und JDO wird Persistenzfähigkeit durch explizite Kennzeichnung vorgenommen.

Aufgabe 3: Thema: Transaktionen

(4 BE, 5 min)

Erklären Sie die für eine Transaktion geforderten ACID-Eigenschaften!

Lösung:

Atomicity: Es werden entweder alle Operationen einer Transaktion ausgeführt oder gar keine.

Consistency: Nach Abschluss der Transaktion ist der Datenbestand konsistent

Isolation: Die Transaktionen beeinflussen sich nicht gegenseitig

Durability: Die Auswirkung einer Transaktion ist dauerhaft

Aufgabe 4: Thema: Anfragesprachen

(7 BE, 10 min)

Betrachten Sie die folgende OQL-Anfrage, die in einer Erweiterung des ER-Modells von Anhang I gestellt wird:

```
select v
from v in AlleVerkäufer
where exists k in v.Kunden: k.Alter > v.Alter
```

- Von welchem Strukturtyp muss `AlleVerkäufer` sein und von welchem Strukturtyp `v.Kunden`?
- Formulieren Sie umgangssprachlich, welche Daten durch diese Anfrage aus der Datenbank geholt werden!
- Folgende JDOQL-Befehle sollen dieselbe Anfrage in JDOQL realisieren:

```
String filter = "...";
Query query = pm.newQuery( AlleVerkäufer, filter);
Collection result = (Collection) query.execute();
```

Vervollständigen Sie den String für die Variable `filter`!

Lösung:

- `AlleVerkäufer` muss ein `Extent` sein und `v.Kunden` eine `Collection`.
- Es werden alle Verkäufer geladen, die einen Kunden haben, der älter ist als sie selbst.
- `filter = "Kunden.contains(k) && (k.Alter > Alter)"`

Alternative Lösung:

- `filter = "this.Kunden.contains(k) && (k.Alter > this.Alter)"`

Aufgabe 5: Thema: Datenidentitätskonzepte

(4 BE, 10 min)

Sie sollen ein Objekt mit Objektnamen "Joker" zur Klasse `MyClass` aus einer JDO-Datenbank holen. Welche Aktionen müssen Sie durchführen? Benutzen Sie die Methodendefinitionen aus Anhang II und geben Sie die Aufrufe der benötigten Javamethoden mit den richtigen Parametern in der richtigen Reihenfolge an! Hinweis: Die meisten in Anhang II aufgeführten Methoden werden nicht benötigt.

Lösung:

```
Object oid = pm.newObjectIdInstance(MyClass.class, "Joker");  
MyClass myObject = (MyClass) pm.getObjectById (oid, false);
```

Lösungsalternative:

```
Object oid = pm.newObjectIdInstance(MyClass.class, "Joker");  
MyClass myObject = (MyClass) pm.getObjectById (oid, true);
```

Aufgabe 6: Thema: Lebenszykluszustände

(6 BE, 10 min)

- a) Was bedeutet der Zustand `hollow`? (2 BE, 3 min)
- b) Geben Sie alle obligatorischen Zustände an, aus denen man ohne Zwischenzustände in den Zustand `hollow` gelangen kann! (siehe Anhang III) (2 BE, 3 min)
- c) Unter welcher Bedingung kann man auf die Attribute von Objekten, die bisher im Zustand `hollow` waren, **nicht** im Java-Programm zugreifen? Unter welcher Bedingung kann man dagegen auf die Attribute von Objekten zugreifen, die bisher im Zustand `hollow` waren, und in welchen Zustand wechseln die Objekte, wenn ein solcher Zugriff erfolgt? (siehe Anhang III) (2 BE, 6 min)

Lösung:

- a) Im Zustand `hollow` befindet sich ein persistentes Objekt, dessen persistente Attribute nicht geladen sind. An ihrer Stelle wird nur die JDO-Id abgespeichert.
- b) `persistent-new`, `persistent-dirty`, `persistent-clean`, `persistent-deleted`
- c) Außerhalb von Transaktionen kann man nicht auf `hollow` Objekte zugreifen. Sie wechseln dort nämlich nicht automatisch in einen persistenten Zustand. Innerhalb von Transaktionen geht solch ein Zugriff, da ein `hollow` Objekt automatisch in den Zustand `persistent-clean` wechselt.

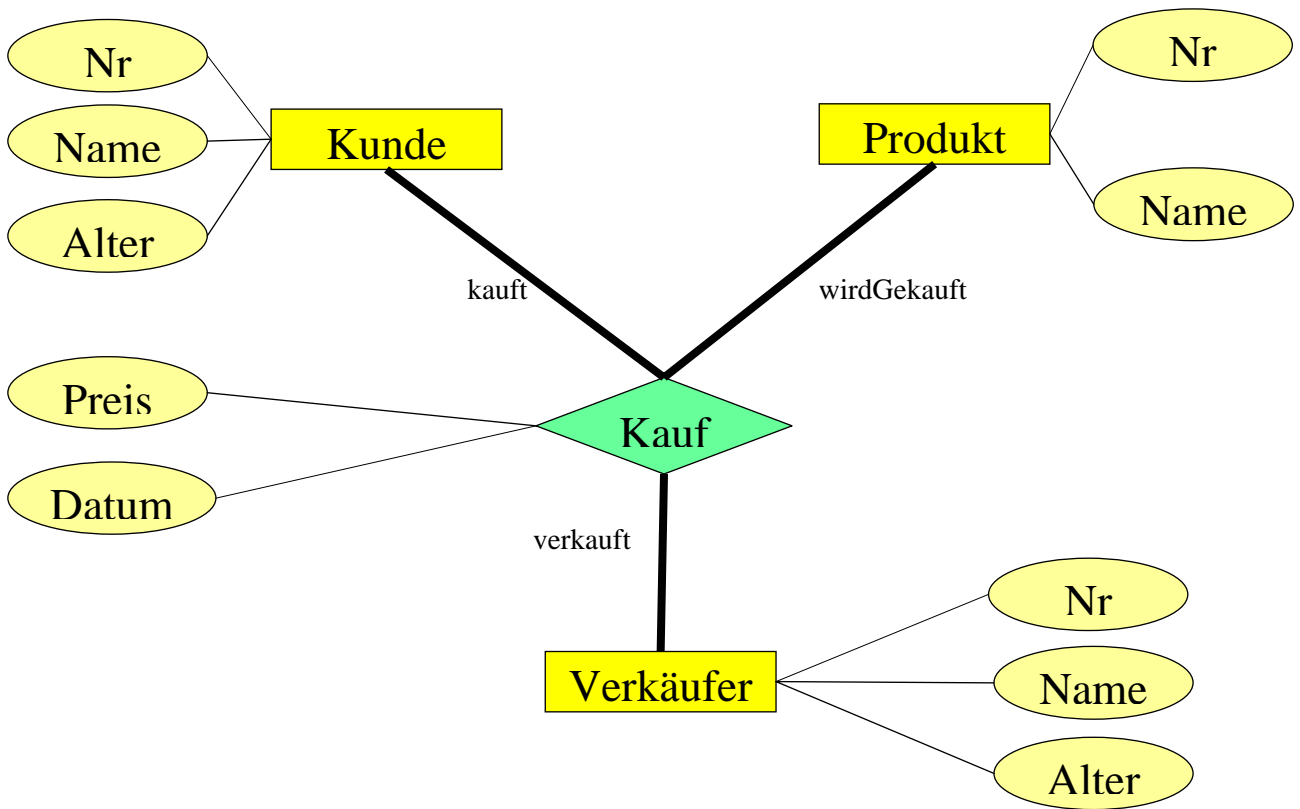
Aufgabe 7: Thema: Datenbanken in betriebswirtschaftlichen Anwendungen (2 BE, 5 min)

Ein Versandhaus, dessen Datenbestand sich häufig ändert, möchte seinen Kunden einen sicheren und schnellen on-line-Zugriff ermöglichen. Der Projektleiter hat gehört, dass Datawarehouses eine moderne und effiziente Technologie zur Vernetzung der Daten zur Verfügung stellen. Würden Sie ihm zum Einsatz eines Datawarehouses raten? (mit Begründung)

Lösung:

Es ist dringend abzuraten, denn Datawarehouses sind nicht für viele Kunden und häufige Aktualisierungen geeignet.

Anhang I



Anhang II

Method Summary of Class PersistenceManager (Auszug)	
<u>Extent</u>	<u>getExtent</u> (java.lang.Class persistenceCapableClass, boolean subclasses) The PersistenceManager manages a collection of instances in the data store based on the class of the instances.
java.lang.Object	<u>getObjectById</u> (java.lang.Object oid, boolean validate) This method retrieves a persistent instance specified by the object id oid in the cache of instances managed by this PersistenceManager.
java.lang.Object	<u>getObjectId</u> (java.lang.Object pc) The ObjectId returned by this method represents the JDO identity of the instance.
java.lang.Class	<u>getObjectIdClass</u> (java.lang.Class cls) Return the Class that implements the JDO Identity for the specified PersistenceCapable class.
void	<u>makePersistent</u> (java.lang.Object pc) Make the transient instance persistent in this PersistenceManager.
void	<u>makeTransient</u> (java.lang.Object pc) Make an instance transient, removing it from management by this PersistenceManager.
java.lang.Object	<u>newObjectIdInstance</u> (java.lang.Class pcClass, java.lang.String objectName) This method returns an object id instance corresponding to the Class and String arguments.
<u>Query</u>	<u>newQuery</u> () Create a new Query with no elements.
<u>Query</u>	<u>newQuery</u> (java.lang.Class cls) Create a new Query specifying the Class of the candidate instances.
<u>Query</u>	<u>newQuery</u> (java.lang.Class cls, java.util.Collection cln) Create a new Query with the candidate Class and Collection.
<u>Query</u>	<u>newQuery</u> (java.lang.Class cls, java.util.Collection cln, java.lang.String filter) Create a new Query with the Class of the candidate instances, candidate Collection, and filter.
<u>Query</u>	<u>newQuery</u> (java.lang.Class cls, java.lang.String filter) Create a new Query with the Class of the candidate instances and filter.
<u>Query</u>	<u>newQuery</u> (<u>Extent</u> cln) Create a new Query with the Class of the candidate instances and candidate Extent.
<u>Query</u>	<u>newQuery</u> (<u>Extent</u> cln, java.lang.String filter) Create a new Query with the candidate Extent and filter; the class is taken from the Extent.

Obligatorische Lebenszykluszustände:

- **transient**
- **persistent-new**
- **hollow**
- **persistent-clean**
- **persistent-dirty**
- **persistent-deleted**
- **persistent-new-deleted**

Optionale Lebenszykluszustände:

- **transient-clean**
- **transient-dirty**
- **persistent-nontransactional**