I Erläuterungen

Aufgabenart
materialgebundene Aufgabenstellung

Voraussetzungen gemäß Lehrplan und Erlass „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen im beruflichen Gymnasium (fachrichtungs-/schwerpunktbezogene Fächer)“ in der für den Abiturjahrgang geltenden Fassung

Implementierung
Die Implementierung soll grundsätzlich in einer objektorientierten Programmiersprache erfolgen. Die Musterlösungen sind in die Programmiersprache zu übertragen, die an der jeweiligen Schule verwendet wird.

Entity-Relationship-Modell
Die Notation für Entity-Relationship-Diagramme hat sich in den zurückliegenden Jahren stark verändert. Neben der ursprünglichen Darstellungsform, die auf P. P. Chen (1976) zurückgeht, unterstützen Softwaretools vielfältige Formen, bis hin zur Krähenfuß-Notation. Aus didaktischen Gründen wird die Notation nach Chen verwendet, die übersichtlich und einfach auch von Hand zu erstellen ist. Die Entitätsmengen werden durch Substantive, die Beziehungen durch Verben beschrieben und die Kardinalitäten in Min-Max-Schreibweise angegeben.

Relationales Modell
Die Primärschlüssel sind in den Musterlösungen unterstrichen, Fremdschlüssel mit einer Raute („#“) versehen.

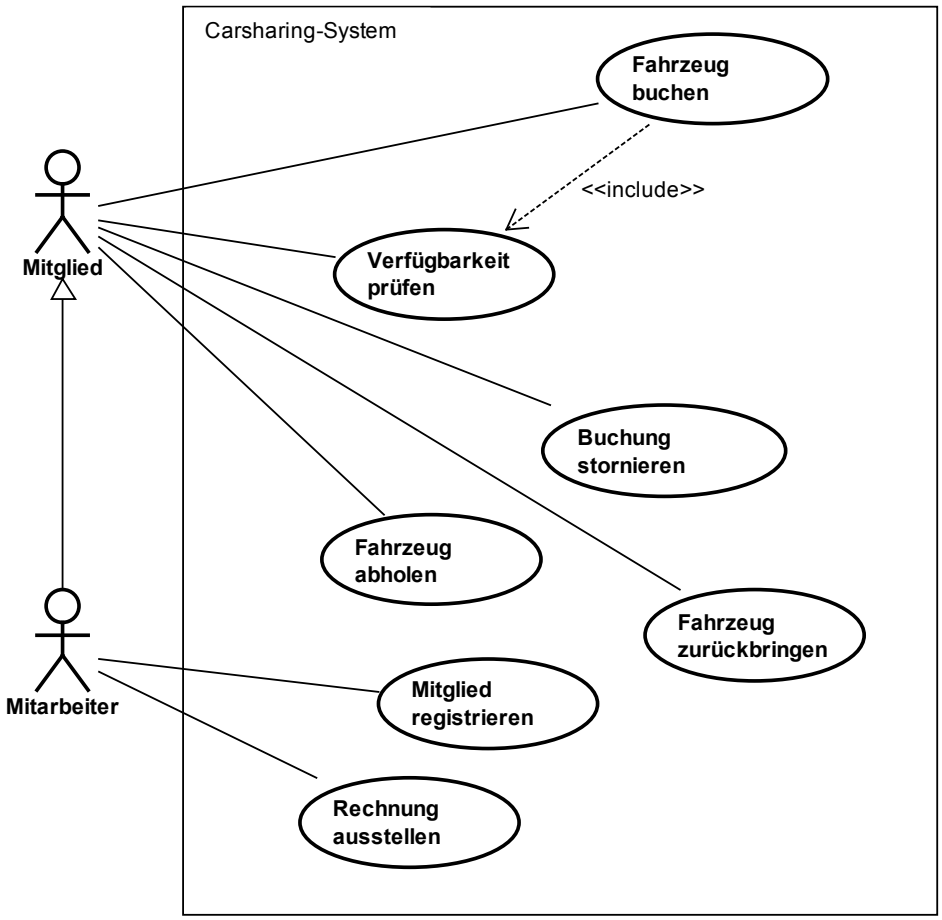
SQL-Syntax
Die verwendete SQL-Syntax orientiert sich an dem Standard SQL-92 (SQL2). Abweichungen aufgrund der im Unterricht verwendeten Datenbanksysteme sind möglich. Abweichende Lösungen aufgrund anderer Tabellen aus vorangegangenen Aufgabenteilen sind entsprechend zu bewerten.

Aufgabe 1: Im ersten Teil der Aufgabe ist ein Ausschnitt aus einem Use Case-Diagramms zu beschreiben, die Anwendungsfälle des Systems zu analysieren und das gegebene Use Case-Diagramm zu erweitern. Ein gegebenes UML-Klassendiagramm ist in Teilen zu implementieren, das vorliegende Objektmodell entsprechend den Anforderungen zu erweitern. Im zweiten Teil der Aufgabe ist die Kommunikation zwischen Objekten in einem UML-Sequenzdiagramm vorgegeben, der dargestellte Ablauf ist zu erläutern und zu implementieren. Die Aufgabe enthält Teile aus Q1 „Objektorientierte Softwareentwicklung“. Zur Bearbeitung der Aufgabe werden folgende verbindlichen Inhalte des Lehrplans gefordert: Objektorientierte Modellierung unter Einbeziehung der Modellierungssprache UML (Use Case-, Klassen-, und Sequenzdiagramm), Klassen, Kollektionen, Standardoperationen auf Kollektionen, iterative Verfahren. Zur Kommunikation mit der Außenwelt werden Inhalte aus Q2 „Datenkommunikation“ benötigt, um die Kommunikation über die serielle Schnittstelle mittels eines gegebenen Protokolls zu ermöglichen.

Aufgabe 2: Ein vorgegebenes Entity-Relationship-Modell (ER-Modell) ist in das relationale Modell zu überführen und für dieses SQL-Statements zur Datenmanipulation und -abfrage zu entwickeln. Darüber hinaus ist das Modell den Anforderungen entsprechende zu erweitern und als Diagramm darzustellen. An einem konkreten Fall sind Möglichkeiten zur Realisierung von Vererbungsbeziehungen zu diskutieren.
Die Aufgabe bezieht sich auf den Kurs „Datenbanken“ aus Q3. Von den verbindlichen Inhalten des Lehrplans werden zur Lösung der Aufgabe benötigt: die Modellierung von Mini-Welten mit ER-Modellen, die Überführung eines ER-Modells in das relationale Modell, Datenmanipulation und Abfragen mittels SQL-Anweisungen.

II Lösungshinweise

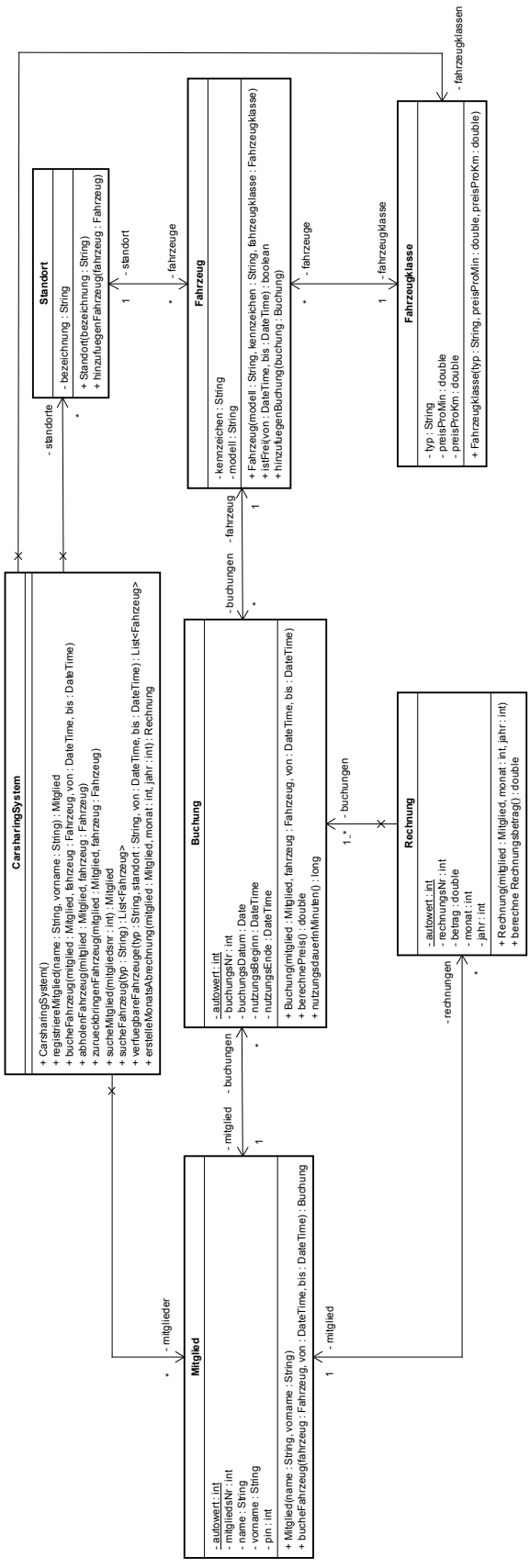
In den nachfolgenden Lösungshinweisen sind alle wesentlichen Gesichtspunkte, die bei der Bearbeitung der einzelnen Aufgaben zu berücksichtigen sind, konkret genannt und diejenigen Lösungswege aufgezeigt, welche die Prüflinge erfahrungsgemäß einschlagen werden. Selbstverständlich sind jedoch Lösungswege, die von den vorgegebenen abweichen, aber als gleichwertig betrachtet werden können, ebenso zu akzeptieren.

Aufg.	erwartete Leistungen	BE		
		I	II	III
1.1	UML-Anwendungsfalldiagramm beschreiben Das abgebildete Use Case-Diagramm enthält zwei Akteure, das zu entwickelnde System mit Systemgrenzen und 2 Anwendungsfälle. Die Beziehung Akteur und Anwendungsfall besagt, dass der Akteur an diesem Anwendungsfall beteiligt ist oder diesen auslösen kann. Zwischen den beiden Akteuren besteht eine Generalisierung. Diese Beziehung kann zwischen Akteuren und zwischen Anwendungsfällen modelliert werden und bedeutet, dass ein Anwendungsfall oder ein Akteur spezialisiert wird. Die Pfeilspitze zeigt auf den Akteur oder Anwendungsfall, der spezialisiert wird. Im vorliegenden Fall heißt das, dass der Akteur Mitarbeiter alle Anwendungsfälle verwenden kann, die auch ein Mitglied verwenden kann, darüber hinaus ist es aber nur dem Mitarbeiter möglich Rechnungen auszustellen.	2	2	
1.2	UML-Anwendungsfalldiagramm entwickeln und zeichnen  	2	2	2

Aufg.	erwartete Leistungen	BE		
		I	II	III
1.3	Klassen aus UML-Klassendiagramm in eine Programmiersprache überführen und Methoden implementieren	4	4	2

```
public class Mitglied {
  private static int autowert = 1;
  private int mitgliedsNr;
  private String name;
  private String vorname;
  private int pin;
  private List<Buchung> buchungen;

  public Mitglied(String name, String vorname) {
    mitgliedsNr = autowert++;
    this.name = name;
    this.vorname = vorname;
    buchungen = new List<Buchung>();
  }

  public Buchung bucheFahrzeug(Fahrzeug fahrzeug,
                               DateTime von, DateTime bis) {
    Buchung buchung = null;
    if (fahrzeug.istFrei(von, bis)) {
      buchung = new Buchung(this, fahrzeug, von, bis);
      buchungen.add(buchung);
    }
    return buchung;
  }
public class Buchung {
  private static int autowert = 1;
  private int buchungsNr;
  private Date buchungsDatum;
  private DateTime nutzungsBeginn;
  private DateTime nutzungsEnde;
  private Mitglied mitglied;
  private Fahrzeug fahrzeug;

  public Buchung(Mitglied mitglied, Fahrzeug fahrzeug,
                 DateTime von, DateTime bis) {
    buchungsNr = autowert++;
    nutzungsBeginn = von;
    nutzungsEnde = bis;
    this.mitglied = mitglied;
    this.fahrzeug = fahrzeug;
    fahrzeug.hinzufuegenBuchung(buchung);
  }

  public double berechnePreis() {
    long nutzungsdauer = nutzungsdauerInMinuten();
    double preis = nutzungsdauer * fahrzeug.getPreisProMin();
    return preis;
  }

  public long nutzungsdauerInMinuten() {
    long diff = (nutzungsBeginn.inMillis() -
                 nutzungsEnde.inMillis()) / 1000 / 60;
    return diff;
  }
}
```

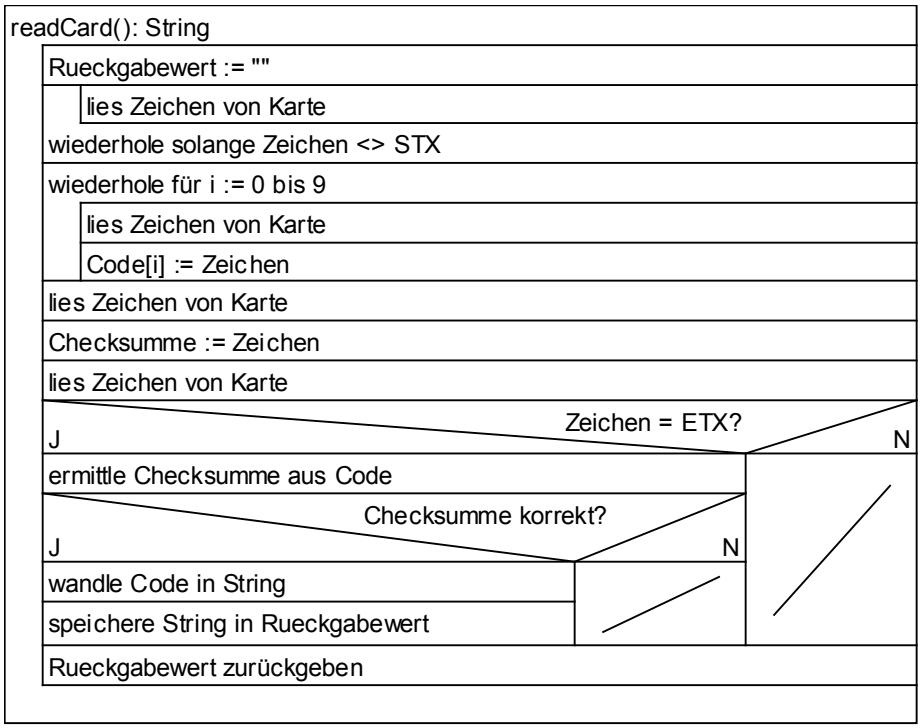
Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.4	UML-Klassendiagramm entwickeln und zeichnen	2	4	4



Aufg.	erwartete Leistungen	BE		
		I	II	III
1.5.1	Den in einem UML-Sequenzdiagramm dargestellten Ablauf beschreiben Bei der Abholung prüft der Bordcomputer die Gültigkeit der Mitgliedskarte. Hierzu wird in der Methode `fetchCar()` zunächst mittels RFID-Reader geprüft, ob eine Mitgliedskarte vorhanden ist (`rr.isCardAvailable()`). Wurde eine Karte erkannt, werden die Daten der Karte mittels RFID-Reader gelesen (`rr.readCard()`) und (über Funk) geprüft, ob die Mitgliedskarte gültig ist (`isCardValid()`). Ist die Karte gültig, wird (über Funk) geprüft, ob eine Buchung für das Fahrzeug erfolgt ist (`getOrderState()`). Ist dies der Fall, wird die Zentralverriegelung geöffnet (`openDoor()`). Anschließend wird die Zeitüberwachung aktiviert (`startTimer()`). Nun hat das Mitglied maximal 5 Minuten Zeit, die PIN über das Touchdisplay einzugeben (`td.readLine()`), dabei kann es sich 2 Fehlversuche leisten. Die PIN wird mittels der Methode `checkPin()` geprüft. Wurde die PIN korrekt eingeben wird das Fahrzeug freigegeben (`enable()`), andernfalls wird die Freigabe verweigert. In beiden Fällen wird eine entsprechende Meldung auf dem Touchdisplay ausgegeben (`td.printLine()`) und die Buchungszentrale informiert (`rm.sendData()`). Wird die PIN dreimal falsch eingegeben, bleibt die Wegfahrsperre aktiviert.	5		

Aufg.	erwartete Leistungen	BE		
		I	II	III
1.5.2	Methode nach UML-Sequenzdiagramm implementieren		5	5

```
public void fetchCar() {
  // prüfe, ob eine Kundenkarte vorhanden ist
  if (rr.isCardAvailable()) {
    // lies die Daten der Karte
    String s = rr.readCard();
    // prüfe (über Funk), ob die Kundenkarte gültig ist
    if (isCardValid(s)) {
      // prüfe (über Funk), ob die Buchung erfolgt ist
      if (getOrderState(s)) {
        // gib Zentralverriegelung frei
        openDoor();
        // Zeitüberwachung aktivieren
        startTimer();
        long wartezeit = 0;
        int versuche = 0;
        boolean freigabeOK = false;
        do {
          // lies Pin über die Tastatur ein
          String t = td.readLine();
          int pin = Integer.parseInt(t);
          // prüfe (über Funk), ob die Pin gültig ist
          if (checkPin(pin))
            freigabeOK = true;
          else
            versuche++;
          wartezeit = getTimer();
        } while (versuche < 3 && !freigabeOK &&
                 wartezeit <= 5 * 60);

        if (freigabeOK) {
          // Wegfahrsperre deaktivieren
          enable();
          td.printLine("Fahrzeug freigegeben.");
          rm.sendData("Fahrzeug abgeholt.");
        } else {
          // Wegfahrsperre bleibt aktiviert
          td.printLine("Freigabe verweigert.");
          // Buchungszentrale verständigen
          rm.sendData("Unberechtigter Nutzungsversuch.");
        }
      }
    }
  }
}
```

Aufg.	erwartete Leistungen	BE I	BE II	BE III
1.5.3	Struktogramm entwickeln und zeichnen 	2	4	4
1.5.4	Prinzip des Thread an einem Beispiel zeigen und beschreiben der Realisierung Threads sind leichtgewichtige Prozesse, die sich den gleichen Adressraum im Speicher teilen. Mit Hilfe von Threads lassen sich parallele Aktivitäten innerhalb eines Prozesses beschreiben. In einem Prozess können mehrere Threads gleichzeitig ablaufen. Da sowohl die Erfassung der Fahrdaten als auch die Kommunikation des Bordcomputers mit der Zentrale, die z. B. jederzeit die aktuellen Fahrdaten abrufen kann, nebenläufig ablaufen sollen, ist es günstig, beide Vorgänge als Threads zu realisieren. Um einen Thread in Java zu realisieren kann man eine Unterklasse der Klasse `Thread` ableiten, in dieser die Methode `run()` überschreiben und dann ein Exemplar von dieser neuen Unterklasse erzeugen. Der Thread kann dann durch den Aufruf der Methode `start()` gestartet werden. Diese ruft hierzu die überschriebene Methode `run()` asynchron auf, das heißt, die Methode `run()` wird nebenläufig ausgeführt. Der Thread ist beendet, sobald die `run()`-Methode beendet ist. In der `run()`-Methode des Threads zur Funk-Kommunikation mit der Zentrale werden die Anforderungen der Zentrale entgegen genommen und die geforderten Daten an die Zentrale zurückgeliefert. Parallel dazu kann der Bordcomputer noch weitere Aufgaben wahrnehmen, wie z. B. die Zeitüberwachung oder die Erfassung der Fahrdaten.		5	
	Summe 60	**17**	**26**	**17**

Aufg.	erwartete Leistungen	BE		
		I	II	III
2.1	ER-Modell in das relationale Modell überführen Mitglied (mitgliedsnr, name, vorname) Buchung (buchungsnr, nutzungsBeginn, nutzungsEnde, mitgliedsnr#,kennzeichen#) Fahrzeug (kennzeichen, hersteller, modell, baujahr, kmStand, klasse#) Fahrzeugklasse (bezeichnung, preisProMin, preisProKm)	4	1	

2.2.1	Einfache Auswahlabfrage angeben
	`SELECT bezeichnung`
	`FROM Fahrzeugklasse`
	`WHERE preisProMin < 0.15`
	`AND preisProKm < 0.20;`
		2		

2.2.2	Auswahlabfrage über 3 Tabellen mit Sortierung formulieren
	`SELECT DISTINCT name, vorname`
	`FROM Mitglied, Buchung, Fahrzeug`
	`WHERE Mitglied.mitgliedsnr = Buchung.mitgliedsnr`
	`AND Buchung.kennzeichen = Fahrzeug.kennzeichen`
	`AND hersteller = 'Mercedes'`
	`AND modell = 'Smart'`
	`ORDER BY name, vorname;`
			3	

2.2.3	Auswahlabfrage über 3 Tabellen mit Aggregatfunktion und Gruppierung entwickeln
	`SELECT name, vorname, COUNT(*) AS Anzahl`
	`FROM Mitglied, Buchung, Fahrzeug`
	`WHERE Mitglied.mitgliedsnr = Buchung.mitgliedsnr`
	`AND Buchung.kennzeichen = Fahrzeug.kennzeichen`
	`AND Fahrzeug.klasse = 'Mittel'`
	`GROUP BY name, vorname`
	`HAVING anzahl >= 4;`
			2	2

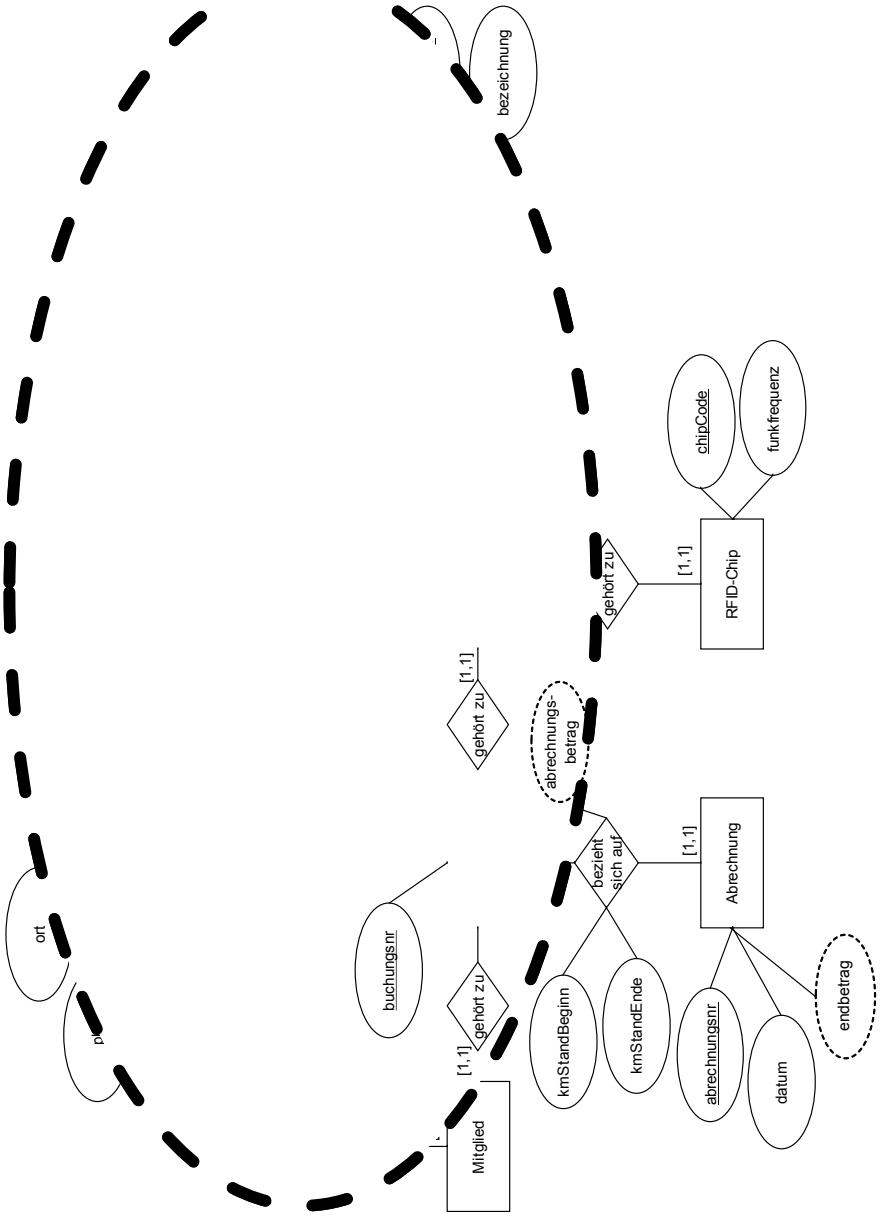
2.2.4	SQL-Anweisung erläutern Es werden Daten aus drei verbundenen Tabellen abgefragt. Hierzu werden mehrere „Joins" hintereinander ausgeführt. Zunächst werden die ersten beiden Tabellen miteinander verbunden und die Datensätze ausgewählt, die der in der ON-Spezifikation angegebenen Bedingung entsprechen. Anschließend wird diese Zwischentabelle mit der nächsten Tabelle verbunden usw. Durch den IN-Parameter in der WHERE-Klausel wird verglichen, ob der Inhalt einer Spalte in der angegebenen Liste enthalten ist. Die angegebene SQL-Abfrage liefert alle Buchungen (Mitgliedsnummer, Vor- und Nachname des Mitglieds, Buchungsnummer, Fahrzeugkennzeichen) der Mitglieder mit den Mitgliedsnummern 1002, 1007 und 1009		3	

Aufg.	erwartete Leistungen	BE		
		I	II	III

2.2.5	SQL-Anweisungen zum Einfügen von Daten angeben
	`INSERT INTO Fahrzeugklasse VALUES ("Van", 0.15, 0.25);`
	`INSERT INTO Fahrzeug`
	`VALUES ("F-UT 421", "VW", "Touran", 2016, 120, "Van");`
	Hinweis: die Reihenfolge der Anweisungen ist zu beachten
		3		

2.3	ER-Modell auf die Einhaltung der 3. Normalform hin untersuchen
	Die Relationen `Mitglied` und `Fahrzeugklasse` befinden sich in der 3. Normalform. Alle Attribute der beiden Relationen sind vom Schlüsselattribut voll funktional abhängig, es gibt keine transitiven Abhängigkeiten. Die Relation `Fahrzeug` befindet sich in der 2. Normalform. Die Wertebereiche der Attribute sind atomar. Das Attribute `modell` hängt transitiv vom Attribute `hersteller` ab. Aus der Relation `Fahrzeug` werden die Attribute `modell` und `hersteller` entfernt und in neue Relationen `Fahrzeugmodell` und `Hersteller` übertragen. Die ursprünglichen Attribute `modell` und `hersteller` werden durch Fremdschlüssel ersetzt. Darüber hinaus muss der Fremdschlüssel `klasse` aus der Relation `Fahrzeug` entfernt und in die neue Relation `Fahrzeugmodell` übertragen werden.
	`Fahrzeug (fahrzeugnr, kennzeichen, baujahr, kmStand, modellnr#)`
	`Fahrzeugmodell (modellnr, modell, herstellernr#, klasse#)`
	`Hersteller (herstellernr, hersteller)`
		2	2	1

Aufg.	erwartete Leistungen	BE		
		I	II	III
2.4	ER-Modell weiter entwickeln		3	7



Aufg.	erwartete Leistungen	BE		
		I	II	III

2.5	Möglichkeiten zur Realisierung einer Spezialisierung zeigen und diskutieren
	Möglichkeit 1:
	`Fahrzeug(kennzeichen, …, anzahlPlaetze, ladevolumen)`
	Alle Attribute in eine Tabelle `Fahrzeug` aufnehmen.
	Vorteil: nur eine Tabelle
	Nachteil: im Allgemeinen wenig speichereffizient (null-Einträge)
	Möglichkeit 2:
	`Pkw(kennzeichen, …, anzahlPlaetze)`
	`Transporter(kennzeichen, …, ladevolumen)`
	Zwei eigene Entitätstypen PKW und Transporter.
	Vorteil: einfacher Zugriff auf Attribute
	Nachteil: es kann u.U. Pkws und Transporter mit gleichen Kennzeichen geben (Fehleinträge). Dies ist in Möglichkeit 1 und 3 ausgeschlossen.
	Möglichkeit 3:
	`Fahrzeug(kennzeichen, …)`
	`Pkw(kennzeichen#, anzahlPlaetze)`
	`Transporter(kennzeichen#, ladevolumen)`
	Jeder Entitätstyp wird in eine Tabelle überführt:
	Spezialisierungen besitzen dieselben Primärschlüssel wie die Generalisierung, diese sind zugleich auch Fremdschlüssel.
	Weitere Merkmale werden in die Tabelle als Attribute aufgenommen.
	Vorteil: einfach und systematisch
	Nachteil: umständliche Navigation für bestimmte Attributwerte
	Hinweis: Nur zwei der drei Lösungsmöglichkeiten sind gefragt.
			2	3

	Summe 40	**11**	**16**	**13**

III Bewertung und Beurteilung

Die Bewertung und Beurteilung erfolgt unter Beachtung der nachfolgenden Vorgaben nach § 33 der Oberstufen- und Abiturverordnung (OAVO) vom 20. Juli 2009 (ABl. S. 408), zuletzt geändert durch Verordnung vom 13. Juli 2016 (ABl. S. 306). Nach § 52 (Übergangsregelungen) sind bei der Bewertung und Beurteilung der sprachlichen Richtigkeit in der deutschen Sprache die Bestimmungen des § 9 Abs. 12 OAVO in Verbindung mit Anlage 9b in der seit 16. August 2016 geltenden Fassung anzuwenden. In den modernen Fremdsprachen sowie den alten Sprachen gelten die Bestimmungen des § 9 Abs. 13 in Verbindung mit den Anlagen 9b und c bzw. 9d der Verordnung in der bis zum 15. August 2016 geltenden Fassung. Bei der Berechnung von Prozentwerten und Fehlerindizes gemäß Anlage 9 OAVO werden die berechneten Werte nicht gerundet. Für die Umrechnung von Prozentanteilen der erbrachten Leistungen in Notenpunkte ist Anlage 9a zu § 9 Abs. 12 OAVO in der bis zum 15. August 2016 geltenden Fassung anzuwenden. Darüber hinaus sind die Vorgaben der Erlasse „Hinweise zur Vorbereitung auf die schriftlichen Abiturprüfungen im beruflichen Gymnasium (fachrichtungs-/schwerpunktbezogene Fächer)“ und „Durchführungsbestimmungen zum Landesabitur“ in der für den Abiturjahrgang geltenden Fassung zu beachten.

Bei der Bewertung und Beurteilung ist auch die Intensität der Bearbeitung zu berücksichtigen. Als Bewertungskriterien dienen über das Inhaltliche hinaus qualitative Merkmale wie Strukturierung, Differenziertheit und Schlüssigkeit der Argumentation.

Im Fach Datenverarbeitungstechnik besteht die Prüfungsleistung aus der Bearbeitung eines Vorschlags, wofür insgesamt maximal 100 BE vergeben werden können. Ein Prüfungsergebnis von **5 Punkten (ausreichend)** setzt voraus, dass insgesamt 46 BE, ein Prüfungsergebnis von **11 Punkten (gut)**, dass insgesamt 76 BE erreicht werden.

Gewichtung der Aufgaben und Zuordnung der Bewertungseinheiten zu den Anforderungsbereichen

Aufgabe	Bewertungseinheiten in den Anforderungsbereichen			Summe
	AFB I	AFB II	AFB III	
1	17	26	17	**60**
2	11	16	13	**40**
Summe	**28**	**42**	**30**	**100**

Die auf die Anforderungsbereiche verteilten Bewertungseinheiten innerhalb der Aufgaben sind als Richtwerte zu verstehen.