Name: _____

Abiturprüfung 2020

Informatik, Leistungskurs

Aufgabenstellung:

Die Firma *Serienflix* bietet Kundinnen und Kunden die Möglichkeit, (TV-)Serien zu streamen, d. h. über das Internet abzurufen. Eine Serie umfasst mehrere Episoden in mehreren Staffeln. Es wird erfasst, welche Kundin bzw. welcher Kunde sich welche Episode angesehen hat. Außerdem besteht die Möglichkeit für eine Kundin bzw. einen Kunden eine angesehene Episode zu bewerten.

Zur Verwaltung der Daten möchte das Unternehmen eine relationale Datenbank aufbauen, die der folgenden Teilmodellierung entspricht:

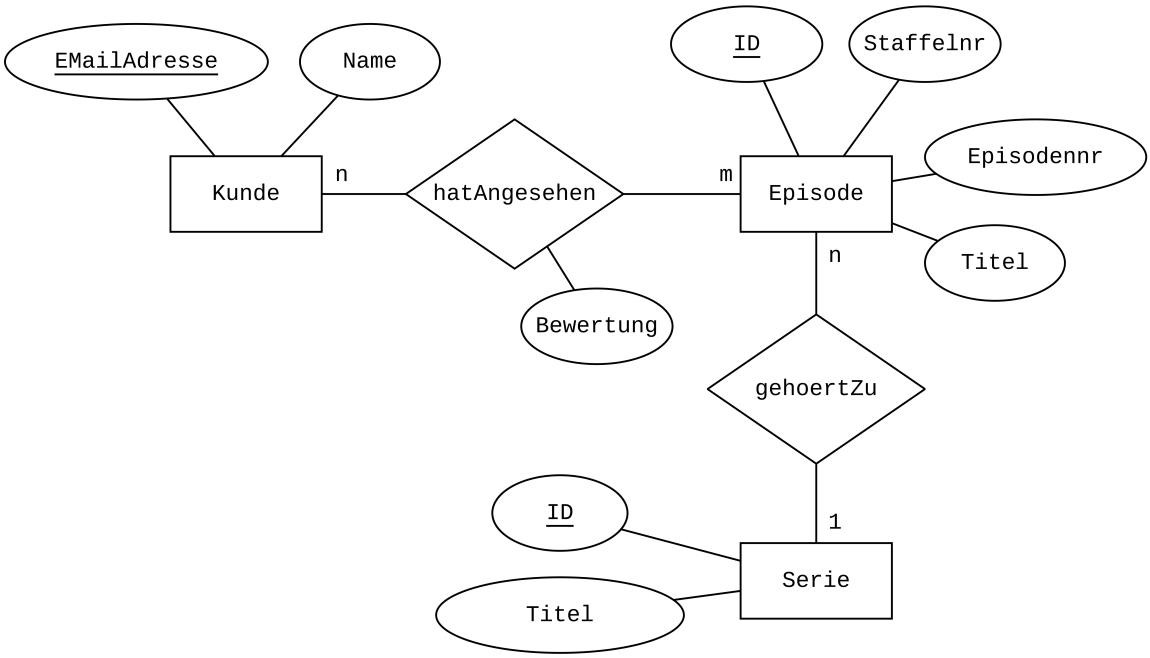


Abbildung 1: Teilmodellierung der Datenbank

Name: _____

Im Folgenden soll mit dem in Abbildung 2 gegebenen Datenbankschema gearbeitet werden. Es stellt einen Ausschnitt der Gesamtdatenbank dar. Beispieldaten zu diesem Datenbankschema sind in der Anlage zu finden.

Kunde(EMailAdresse, Name)
hatAngesehen(↑EMailAdresse, ↑EpisodenID, Bewertung)
Episode(ID, Titel, Episodennr, Staffelnr, ↑SerienID)
Serie(ID, Titel)

Abbildung 2: Datenbankschema eines Ausschnittes der Datenbank

a) *Beschreiben Sie die in Abbildung 1 als Entity-Relationship-Diagramm dargestellte Teilmodellierung.*

Begründen Sie die im Entity-Relationship-Modell gewählten Kardinalitäten für diesen Sachkontext.

Erläutern Sie, wie der `n:m`*-Beziehungstyp und der* `1:n`*-Beziehungstyp aus dem Entity-Relationship-Modell in diesem Datenbankschema umgesetzt wurden.*

(9 Punkte)

b) Aus der Datenbank sollen folgende Informationen abgefragt werden:

i. Gesucht sind die Serientitel aufsteigend sortiert, die das Wort `Spiel` beinhalten.

ii. Gesucht sind die Serientitel mit der jeweils zugehörigen Durchschnittsbewertung. Das Ergebnis soll nach der Durchschnittsbewertung der Serie absteigend sortiert sein. Serien, bei denen noch keine Episode angesehen wurde, müssen nicht berücksichtigt werden.

iii. Gesucht sind alle Namen der Kundinnen und Kunden – auch derjenigen, die noch gar keine Episode angesehen haben – mit der jeweiligen Anzahl der abgegebenen Bewertungen. Das Ergebnis soll absteigend nach der Anzahl der Bewertungen und dann aufsteigend nach den Kundennamen sortiert sein.

Entwerfen Sie für die obigen Anfragen i, ii und iii jeweils eine SQL-Anweisung.

(11 Punkte)

Name: _____

c) Folgende SQL-Anweisung ist gegeben:

```
 1  SELECT * FROM (
 2    SELECT Serie.Titel, COUNT(*) AS x
 3    FROM Serie
 4      INNER JOIN (
 5          SELECT DISTINCT Episode.SerienID, Episode.Staffelnr
 6          FROM Episode
 7        ) AS Temp
 8        ON Serie.ID = Temp.SerienID
 9    GROUP BY Temp.SerienID
10    ) AS Abfrage
11  WHERE Abfrage.x > 1
12  ORDER BY Abfrage.Titel ASC
```

Analysieren und erläutern Sie zunächst die Zeilen 5 bis 7, dann die Zeilen 2 bis 10 und anschließend die gesamte SQL-Anweisung.

Erläutern Sie im Sachzusammenhang, welche Information die gesamte SQL-Anweisung ermittelt.

(9 Punkte)

d) Die Datenbank soll um folgende Aspekte erweitert werden:

i. Für jede Episode soll verwaltet werden, in welchen Sprachen die Episode angeboten wird. Außerdem soll verwaltet werden, was die Originalsprache einer Episode ist.

ii. Um gezielt nach Angeboten mit dem Lieblingsschauspieler oder der Lieblingsschauspielerin suchen zu können, sollen alle Schauspielerinnen und Schauspieler mit ihren Vornamen und Nachnamen verwaltet werden. Außerdem soll verwaltet werden, wer in welcher Episode mitspielt.

Ein Praktikant hat für einen Teil des ersten Aspekts einen Vorschlag entwickelt:

Episode(ID, Titel, Episodennr, Staffelnr, ↑SerienID,
Sprache1, Sprache2, Sprache3, Sprache4, Sprache5)

Für jede Episode kann unter `Sprache1` bis `Sprache5` eingetragen werden, in welchen Sprachen diese Episode angeboten wird.

Name: _____

Begründen Sie, warum der Vorschlag des Praktikanten ungünstig ist.

Modellieren Sie die Erweiterung für das Entity-Relationship-Diagramm aus Abbildung 1 so, dass der neue Datenbankentwurf zusätzlich die beschriebenen Aspekte (vgl. i und ii) enthält.

Erläutern Sie, wie das von Ihnen entwickelte Modell die Erweiterung um die Verwaltung der Sprachen für Episoden mit der Originalsprache (Anforderung i) umsetzt.

Hinweis: Entitäts- und Beziehungstypen sowie Attribute, die für die Erweiterung nicht relevant sind, müssen nicht dargestellt werden.

(11 Punkte)

e) Die Firma erweitert ein bestehendes Programm, um eine neue Funktion zu testen. Abbildung 3 zeigt einen Ausschnitt des Implementationsdiagramms.

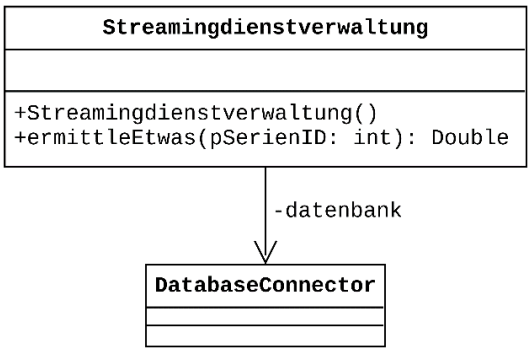


Abbildung 3: Teilmodellierung eines bestehenden Programms einer Streamingdienstverwaltung

Name: _____

Die Klasse `Streamingdienstverwaltung` verfügt über folgende Methode:

```
 1 public Double ermittleEtwas(int pSerienID) {
 2   String sql = "SELECT hatAngesehen.Bewertung "
 3              + "FROM hatAngesehen "
 4              + "  INNER JOIN Episode "
 5              + "    ON hatAngesehen.EpisodenID = Episode.ID "
 6              + "WHERE Episode.SerienID = " + pSerienID
 7              + "  AND hatAngesehen.Bewertung IS NOT NULL "
 8              + "ORDER BY hatAngesehen.Bewertung ASC";
 9
10   datenbank.executeStatement(sql);
11
12   QueryResult qr = datenbank.getCurrentQueryResult();
13   if (qr != null && qr.getRowCount() > 0) {
14     int index = qr.getRowCount() / 2;
15     if (qr.getRowCount() % 2 == 1) {
16       return Double.parseDouble(qr.getData()[index][0]);
17     } else {
18       return (Double.parseDouble(qr.getData()[index-1][0])
19             + Double.parseDouble(qr.getData()[index][0])) / 2;
20     }
21   } else {
22     return null;
23   }
24 }
```

Analysieren Sie die SQL-Anweisung (Zeilen 2 bis 8), indem Sie das Ergebnis der SQL-Anweisung auf Grundlage der Beispieldaten in der Anlage ermitteln, wenn Sie als Parameter `pSerienID` *einmal den Wert* `101` *und einmal den Wert* `222` *übergeben.*

Analysieren Sie die Methode `ermittleEtwas` *und ermitteln Sie die Rückgabe der Methode auf Grundlage der Beispieldaten in der Anlage, wenn Sie als Parameter* `pSerienID` *erneut einmal den Wert* `101` *und einmal den Wert* `222` *übergeben.*

Erläutern Sie die Funktionalität der Methode im Sachkontext.

(10 Punkte)

Zugelassene Hilfsmittel:

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anlage:

Ausschnitt aus den Beispieldaten zum Datenbankschema aus Abbildung 2

Kunde	
EMailAdresse	**Name**
ada@lovelace.de	Ada Lovelace
john@vonneumann.de	John von Neumann
alan@turing.de	Alan Turing
konrad@zuse.de	Konrad Zuse
tim@BernersLee.de	Tim Berners-Lee

Serie	
ID	**Titel**
101	Spiel der Throne
222	Auf die schiefe Bahn
369	Die Urknalltheorie
480	Kartenhaus

Episode				
ID	**Titel**	**Episodennr**	**Staffelnr**	**SerienID**
1184	Tief im Wald	1	7	101
1251	Die Mauer	2	7	101
8042	Angespannte Lage	1	8	101
13040	Wie alles begann	1	1	222
16060	Zwickmühle	2	1	222

Hinweis:
Episoden werden in Produktionsabschnitten, sogenannten Staffeln, produziert. Die `Staffelnr` gibt an, zu welcher Staffel eine Episode gehört.

hatAngesehen		
EMailAdresse	**EpisodenID**	**Bewertung**
ada@lovelace.de	1184	8
alan@turing.de	13040	1
john@vonneumann.de	1251	3
alan@turing.de	8042	9
konrad@zuse.de	13040	10
ada@lovelace.de	13040	9
john@vonneumann.de	13040	*NULL*
konrad@zuse.de	16060	8

Name: _____

Anhang:

Die Klasse DatabaseConnector

Ein Objekt der Klasse **`DatabaseConnector`** ermöglicht die Abfrage und Manipulation einer MySQL-Datenbank.
Beim Erzeugen des Objekts wird eine Datenbankverbindung aufgebaut, so dass anschließend SQL-Anweisungen an diese Datenbank gerichtet werden können.

Dokumentation der Klasse `DatabaseConnector`

Konstruktor **`DatabaseConnector(String pIP, String pPort, String pDatabase, String pUsername, String pPassword)`**

Ein Objekt vom Typ `DatabaseConnector` wird erstellt, und eine Verbindung zur Datenbank wird aufgebaut. Mit den Parametern `pIP` und `pPort` werden die IP-Adresse und die Port-Nummer übergeben, unter denen die Datenbank mit Namen `pDatabase` zu erreichen ist. Mit den Parametern `pUsername` und `pPassword` werden Benutzername und Passwort für die Datenbank übergeben.

Auftrag **`void executeStatement(String pSQLStatement)`**

Der Auftrag schickt den im Parameter `pSQLStatement` enthaltenen SQL-Befehl an die Datenbank ab.
Handelt es sich bei `pSQLStatement` um einen SQL-Befehl, der eine Ergebnismenge liefert, so kann dieses Ergebnis anschließend mit der Methode `getCurrentQueryResult` abgerufen werden.

Anfrage **`QueryResult getCurrentQueryResult()`**

Die Anfrage liefert das Ergebnis des letzten mit der Methode `executeStatement` an die Datenbank geschickten SQL-Befehls als Objekt vom Typ `QueryResult` zurück.
Wurde bisher kein SQL-Befehl abgeschickt oder ergab der letzte Aufruf von `executeStatement` keine Ergebnismenge (z.B. bei einem INSERT-Befehl oder einem Syntaxfehler), so wird `null` geliefert.

Anfrage **`String getErrorMessage()`**

Die Anfrage liefert `null` oder eine Fehlermeldung, die sich jeweils auf die letzte zuvor ausgeführte Datenbankoperation bezieht.

Auftrag **`void close()`**

Die Datenbankverbindung wird geschlossen.

Name: _____

Die Klasse `QueryResult`

Ein Objekt der Klasse **`QueryResult`** stellt die Ergebnistabelle einer Datenbankanfrage mit Hilfe der Klasse `DatabaseConnector` dar. Objekte dieser Klasse werden nur von der Klasse `DatabaseConnector` erstellt. Die Klasse verfügt über keinen öffentlichen Konstruktor.

Dokumentation der Klasse `QueryResult`

Anfrage **`String[][] getData()`**

Die Anfrage liefert die Einträge der Ergebnistabelle als zweidimensionales Feld vom Typ `String`. Der erste Index des Feldes stellt die Zeile und der zweite die Spalte dar (d.h. `String[zeile][spalte]`).

Anfrage **`String[] getColumnNames()`**

Die Anfrage liefert die Bezeichner der Spalten der Ergebnistabelle als Feld vom Typ `String` zurück.

Anfrage **`String[] getColumnTypes()`**

Die Anfrage liefert die Typenbezeichnung der Spalten der Ergebnistabelle als Feld vom Typ `String` zurück. Die Bezeichnungen entsprechen den Angaben in der MySQL-Datenbank.

Anfrage **`int getRowCount()`**

Die Anfrage liefert die Anzahl der Zeilen der Ergebnistabelle als `int`.

Anfrage **`int getColumnCount()`**

Die Anfrage liefert die Anzahl der Spalten der Ergebnistabelle als `int`.

Unterlagen für die Lehrkraft

Abiturprüfung 2020

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Implementation kontextbezogener Problemstellungen mit Schwerpunkt auf dem Inhaltsfeld Daten und ihre Strukturierung

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2020

Die Aufgaben weisen vielfältige Bezüge zu den Kompetenzerwartungen und Inhaltsfeldern des Kernlehrplans bzw. zu den in den Vorgaben ausgewiesenen Fokussierungen auf.
Im Folgenden wird auf Bezüge von zentraler Bedeutung hingewiesen.

1. *Inhaltsfelder und inhaltliche Schwerpunkte*
   Daten und ihre Strukturierung
   - Datenbanken
     - Klassen `DatabaseConnector`, `QueryResult`

   Algorithmen
   - Analyse, Entwurf und Implementierung von Algorithmen

   Formale Sprachen und Automaten
   - Syntax und Semantik einer Programmiersprache
     - SQL
     - Java

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Die in Abbildung 1 gegebene Teilmodellierung besteht aus drei Entitätstypen und zwei Beziehungstypen.

Im Entitätstyp `Kunde` wird als Primärschlüssel `EMailadresse` verwendet.
Im Entitätstyp `Episode` wird als Primärschlüssel `ID` verwendet.
Im Entitätstyp `Serie` wird als Primärschlüssel `ID` verwendet.

Entitäten des Typs `Kunde` haben das weitere Attribut `Name`.
Entitäten des Typs `Episode` haben die weiteren Attribute `Titel`, `Episodennr` und `Staffelnr`.
Entitäten des Typs `Serie` haben das weitere Attribut `Titel`.

Der Beziehungstyp `hatAngesehen` modelliert, welche Kundin oder welcher Kunde welche Episode angesehen hat. Zusätzlich hat der Beziehungstyp das Attribut `Bewertung`.

Der Beziehungstyp `gehoertZu` modelliert, welche Episode Bestandteil welcher Serie ist.

`n:m`-Beziehungstyp `hatAngesehen`:
Eine Kundin bzw. ein Kunde kann mehrere Episoden angesehen haben und eine Episode kann von mehreren Kundinnen bzw. Kunden angesehen werden.

`1:n`-Beziehungstyp `gehoertZu`:
Eine Episode gehört zu genau einer Serie. Eine Serie besteht aus mehreren Episoden.

Der `n:m`-Beziehungstyp `hatAngesehen` wurde mit dem Relationenschema `hatAngesehen` umgesetzt. In dieser Relation werden die zwei Fremdschlüsselattribute `EMailAdresse` und `EpisodenID` verwaltet, welche jeweils die Primärschlüssel einmal in der Relation `Kunde` und einmal in der Relation `Episode` darstellen.

Der `1:n`-Beziehungstyp `gehoertZu` wurde so umgesetzt, dass in der Relation `Episode` zusätzlich das Fremdschlüsselattribut `SerienID` verwaltet wird, welches den Primärschlüssel aus der Relation `Serie` abbildet.

Teilaufgabe b)

Die folgenden SQL-Anweisungen realisieren die Anfragen:

i.
```
SELECT Titel
FROM Serie
WHERE Titel LIKE '%Spiel%'
ORDER BY Titel ASC
```

ii.
```
SELECT Serie.Titel, AVG(hatAngesehen.Bewertung)
FROM Serie
  INNER JOIN Episode
    ON Serie.ID = Episode.SerienID
  INNER JOIN hatAngesehen
    ON Episode.ID = hatAngesehen.EpisodenID
GROUP BY Serie.ID
ORDER BY AVG(hatAngesehen.Bewertung) DESC
```

iii.
```
SELECT Kunde.Name, COUNT(hatAngesehen.Bewertung)
FROM Kunde
  LEFT JOIN hatAngesehen
    ON Kunde.EMailAdresse = hatAngesehen.EMailAdresse
GROUP BY Kunde.EMailAdresse
ORDER BY COUNT(hatAngesehen.Bewertung) DESC, Kunde.Name ASC
```

Teilaufgabe c)

Die Unterabfrage (vgl. Zeilen 5 bis 7) ermittelt redundanzfrei alle `SerienID`s und die dazugehörigen Staffelnummern aus der Tabelle `Episode`.

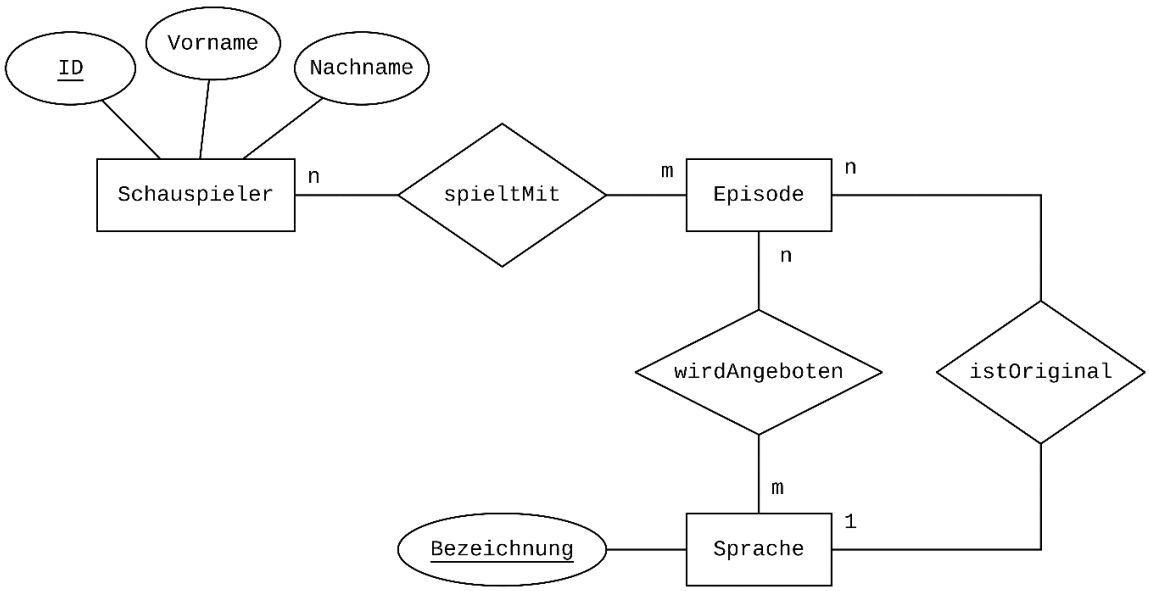
Die Abfrage in Zeilen 2 bis 10 verknüpft mit einem `INNER JOIN` die Tabelle `Serie` mit der Ergebnistabelle aus der genannten Unterabfrage über die `SerienID`. Außerdem gruppiert die SQL-Anweisung die Ergebnisse nach der `SerienID` und ermittelt mit einer Aggregatfunktion die Anzahl der Staffeln.

Die gesamte SQL-Anweisung filtert aus der genannten Abfrage alle Serien mit mehr als einer Staffel und sortiert diese nach dem Serientitel aufsteigend.

Im Sachzusammenhang liefert die gesamte SQL-Anweisung für alle Serien mit mehr als einer Staffel eine Übersicht über alle Serientitel aufsteigend sortiert mit der Anzahl der zur Verfügung stehenden Staffeln.

Teilaufgabe d)

Der Vorschlag des Praktikanten ist ungünstig, weil die Struktur der Relation `Episode` für jede neue Sprache verändert werden muss und die Originalsprache nicht erkennbar ist.



Die Informationen über die zur Verfügung stehenden Sprachen werden in Entitäten des Typs `Sprache` verwaltet. Entitäten dieses Typs werden durch den Primärschlüssel `Bezeichnung` eindeutig identifiziert.

Der `n:m`-Beziehungstyp `wirdAngeboten` zwischen den Entitätstypen `Episode` und `Sprache` modelliert, welche Episode in welchen Sprachen angeboten wird und zu welcher Sprache es welche Episoden gibt.

Der `1:n`-Beziehungstyp `istOriginal` zwischen den Entitätstypen `Episode` und `Sprache` modelliert, was die eine Originalsprache einer Episode ist. Eine Episode kann nur eine Originalsprache haben und eine Sprache kann die Originalsprache in mehreren Episoden sein.

Teilaufgabe e)

Die SQL-Abfrage ermittelt für die übergebene `pSerienID` die jeweils zugehörigen Bewertungen, die aufsteigend sortiert sind.

Ergebnistabelle mit dem Parameteraufruf `pSerienID = 101`
3
8
9

Ergebnistabelle mit dem Parameteraufruf `pSerienID = 222`
1
8
9
10

Die Methode `ermittleEtwas` ermittelt für die übergebene `pSerienID` aus den aufsteigend sortierten Bewertungen eine mittlere Bewertung einer Serie.

Rückgabe der Methode mit dem Parameteraufruf `pSerienID = 101`
8

Rückgabe der Methode mit dem Parameteraufruf `pSerienID = 222`
8.5

Die Methode ermittelt im Sachkontext zu einer bestimmten Serie den Median der Bewertungen. Es wird auf Grundlage der aufsteigend sortierten Bewertungen zu einer Serie die zentrale Bewertung in der Mitte zurückgeliefert. Bei einer ungeraden Anzahl von Datensätzen ist es das mittlere Element. Bei einer geraden Anzahl von Datensätzen ist es das arithmetische Mittel aus den beiden mittleren Elementen.

Falls die gesuchte Serie nicht vorhanden ist oder keine Bewertungen vorliegen, wird `null` zurückgeliefert.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**[2]	**ZK**	**DK**
1	beschreibt die in Abbildung 1 als Entity-Relationship-Diagramm dargestellte Teilmodellierung.	3			
2	begründet die im Entity-Relationship-Modell gewählten Kardinalitäten für diesen Sachkontext.	3			
3	erläutert, wie der n:m-Beziehungstyp und der 1:n-Beziehungstyp aus dem Entity-Relationship-Modell in diesem Datenbankschema umgesetzt wurden.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …… ……					
	Summe Teilaufgabe a)	**9**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft für die erste Anfrage i eine SQL-Anweisung.	3			
2	entwirft für die zweite Anfrage ii eine SQL-Anweisung.	4			
3	entwirft für die dritte Anfrage iii eine SQL-Anweisung.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (11) …… ……					
	Summe Teilaufgabe b)	**11**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert und erläutert die Zeilen 5 bis 7.	2			
2	analysiert und erläutert die Zeilen 2 bis 10.	3			
3	analysiert und erläutert die gesamte SQL-Anweisung.	2			
4	erläutert im Sachzusammenhang, welche Information die gesamte SQL-Anweisung ermittelt.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …… ……					
	Summe Teilaufgabe c)	**9**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	begründet, warum der Vorschlag des Praktikanten ungünstig ist.	2			
2	modelliert die Erweiterung für das Entity-Relationship-Diagramm so, dass der neue Datenbankentwurf zusätzlich die beschriebenen Aspekte enthält.	5			
3	erläutert, wie das entwickelte Modell die Erweiterung, um die Verwaltung der Sprachen für Episoden mit der Originalsprache (Anforderung i) umsetzt.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (11) …… ……					
	Summe Teilaufgabe d)	**11**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert die SQL-Anweisung, indem er das Ergebnis der SQL-Anweisung auf Grundlage der Beispieldaten in der Anlage ermittelt, wenn er als Parameter `pSerienID` einmal den Wert `101` und einmal den Wert `222` übergibt.	4			
2	analysiert die Methode `ermittleEtwas` und ermittelt die Rückgabe der Methode auf Grundlage der Beispieldaten in der Anlage, wenn er als Parameter `pSerienID` erneut einmal den Wert `101` und einmal den Wert `222` übergibt.	2			
3	erläutert die Funktionalität der Methode im Sachkontext.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe e)	**10**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	**EK**	**ZK**	**DK**
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

Berechnung der Endnote nach Anlage 4 der Abiturverfügung auf der Grundlage von § 34 APO-GOSt

Die Klausur wird abschließend mit der Note _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0