Probeklausur

Informationssysteme und Datenanalyse

Universität:	Technische Universität Berlin
Kurs/Modul:	Informationssysteme und Datenanalyse
Bearbeitungszeit:	90 Minuten
Erstellungsdatum:	September 19, 2025

Aufgabe 1.

(a) Definieren Sie das relationale Modell mit den Begriffen Relationen, Schema, Attribute, Tupel und Schlüssel. Erläutern Sie kurz, wieso das relationale Modell gut geeignet ist, Daten konsistent abzubilden.

(b) Gegeben seien die Relationen

$$R = (\text{id, name, city}), \quad S = (\text{order\_id, cust\_id, order\_date, amount}).$$

Bestimmen Sie in relationaler Algebra alle Kundinnen und Kunden, die mindestens eine Bestellung mit Betrag größer als 100 getätigt haben.

(c) Schreiben Sie eine SQL-Abfrage, die pro Kunde den Gesamtumsatz ermittelt und nur Kunden ausgibt, die einen Gesamtumsatz von mehr als 100 haben. Geben Sie Kunde und total_spend aus.

```
SELECT c.name AS kunde, SUM(o.amount) AS total_spend
FROM R c
JOIN S o ON o.cust_id = c.id
GROUP BY c.name
HAVING SUM(o.amount) > 100;
```

(d) Geben Sie eine grobe 3NF-Zerlegung eines typischen Transaktionsdatensatzes an. Nennen Sie beispielhaft die resultierenden Tabellen und Schlüssel.

Tabelle	Felder	Schlüssel
CUSTOMER	id PK, name, city	id
ORDER	order_id PK, customer_id FK, order_date	order_id
ORDER_ITEM	order_id FK, product_id FK, quantity	(order_id, product_id)

Aufgabe 2.

(a) Beschreiben Sie kurz den Unterschied zwischen Transaktionssystemen und Data-Warehouse-Systemen. Welche typischen Anforderungen ergeben sich daraus für Architektur und Design?

(b) Entwerfen Sie ein einfaches Star-Schema für eine Verkaufsanalyse. Definieren Sie eine Faktentabelle *FactSales* und drei Dimensionstabellen *DimDate*, *DimProduct*, *DimStore*. Benennen Sie Schlüsselattribute und typische Measure-Spalten.

FactSales(fact$_i dPK, date_k eyFK, product_k eyFK, store_k eyFK, units_s old, revenue$)
DimDate(date$_k eyPK, date, month, quarter, year$)
DimProduct(product$_k eyPK, product_n ame, category, price$)
DimStore(store$_k eyPK, store_n ame, city, region$)

(c) Schreiben Sie eine SQL-ähnliche Abfrage, die den Umsatz pro Store im aktuellen Monat ermittelt. Verwenden Sie das Star-Schema aus Teil (b) und geben Sie Store-Name sowie Revenue aus.

```
SELECT ds.store_name, SUM(fs.revenue) AS revenue
FROM FactSales fs
JOIN DimStore ds ON fs.store_key = ds.store_key
JOIN DimDate dd ON fs.date_key = dd.date_key
WHERE dd.date >= <start_of_month> AND dd.date < <start_of_next_month>
GROUP BY ds.store_name;
```

(d) Skizzieren Sie eine einfache ETL-Pipeline, die Daten aus operativen Quellsystemen in das Data-Warehouse-System überführt. Führen Sie die groben Schritte Extract, Transform, Load auf und nennen Sie typische Aufgaben in jedem Schritt.

Aufgabe 3.

(a) Unterscheiden Sie Begrifflichkeiten im Bereich Data Science: Unüberwachtes Lernen vs. überwachtes Lernen. Welche typischen Aufgaben fallen in jede Kategorie? Nennen Sie je ein gängiges Beispiel aus dem Praxisbereich.

(b) Beschreiben Sie den Grundalgorithmus von k-Means. Führen Sie die Schritte auf, einschließlich Initialisierung, Zuordnung, Neuberechnung der Zentren und Konvergenzkriterium. Welche Annahmen stehen hinter dem Algorithmus?

(c) Gegeben seien die Merkmale einer kleinen Stichprobe (2 Merkmale):

$$\{(1,2),\ (2,1),\ (3,4),\ (4,3)\}.$$

Skizzieren Sie grob, wie eine einfache lineare Trennregel aussehen könnte. Geben Sie eine mögliche Gerade in Koordinatenform an, die die Punkte in zwei Klassen trennt.

(d) Erläutern Sie kurz, wie Streaming-Daten in der Data-Science-Pipeline nutzbar gemacht werden können. Welche Typen von Modellen eignen sich besonders für kontinuierliche Datenströme?

Aufgabe 4.

(a) Begriffsklärung: Was versteht man unter einem Data-Stream-Management-System (DSMS)? Welche Aufgabe hat es, und welche Unterschiede zu klassischen Datenbanken bestehen?

(b) Skizzieren Sie eine grobe Architektur eines DSMS. Nennen Sie zentrale Bausteine wie Quellen, Operatoren, Fenstermechanismen und Outputs. Erläutern Sie kurz, wozu Fenster in Streams dienen.

(c) Formulieren Sie eine einfache Streaming-Abfrage (CQL/SQL-ähnliche Syntax), die für jedes Produkt innerhalb eines 60-Minuten-Fensters den Durchschnittspreis berechnet. Geben Sie die Struktur der Abfrage an.

```
SELECT product_id, AVG(price) AS avg_price
FROM Stream(Sales)
WINDOW 60 MINUTES
GROUP BY product_id;
```

(d) Nennen Sie drei typische Anwendungsbereiche für DSMS in der Praxis.

Lösungen

Aufgabe 1.

(a) Lösung:

- Relationale Modell: Daten werden als Mengen von Tupeln in Relationen dargestellt. Eine Relation ist eine mathematische Tabelle, deren Spalten (Attribute) eine schlichte Ordnung besitzen und deren Zeilen (Tupel) einzigartige Instanzen eines konkreten Tupeltyps darstellen.
- Schema: Eine endliche Menge von Relationen, wobei jede Relation durch ihren Namen und ihr Tupelformat (Attributes) beschrieben wird.
- Attribute: Spalten einer Relation. Jedes Attribut hat einen Namen und einen Domänenbereich (Wertebereich).
- Tupel: Eine Zeile einer Relation, d. h. eine konkrete Instanz der in einem Schema beschriebenen Attributewerte.
- Schlüssel: Candidate Keys (Schlüsselmengen), die minimale Menge von Attributen, deren Werte Tupel eindeutig identifizieren. Häufiger wird ein Primary Key ausgewählt. Integritätsregeln wie Schlüsselgleichheit, Funktionale Abhängigkeiten und referentielle Integrität (FKs) sichern Konsistenz.
- Warum geeignet für Konsistenz: Das relationale Modell unterstützt Integritätsbedingungen, Normalformen und Transaktionsschemata, sodass Redundanz minimiert wird (bis zu 3NF/BCNF) und Update-Anomalien vermieden werden. Datenabhängigkeiten lassen sich explizit modellieren, und Abfragen bleiben deklarativ konsistent formuliert.

(b) Lösung: Die gesuchten Kundinnen und Kunden ergeben sich durch eine Selektion der Tupel aus der Verbindung von R und S mit Betragsanomalien > 100, gefolgt von Projektion der Kundendaten. Formal:

$$K \;=\; \pi_{R.\mathrm{id},\, R.\mathrm{name},\, R.\mathit{city}}\big(\sigma_{S.\mathrm{amount}>100}\big(R \;\bowtie_{R.\mathrm{id}=S.\mathrm{cust\_id}}\; S\big)\big).$$

Dabei liefert der Ausdruck K die Kundinnen und Kunden (identifiziert durch id, Name, Stadt), die mindestens eine Bestellung mit Betrag > 100 hatten.

(c) Lösung:

```
SELECT c.name AS kunde, SUM(o.amount) AS total_spend
FROM R c
JOIN S o ON o.cust_id = c.id
GROUP BY c.id, c.name
HAVING SUM(o.amount) > 100;
```

(d) Lösung:

Tabelle	Felder	Schlüssel
CUSTOMER	id PK, name, city	id
ORDER	order_id PK, customer_id FK, order_date	order_id
ORDER_ITEM	order_id FK, product_id FK, quantity	(order_id, product_id)

Aufgabe 2.

(a) Lösung: Transaktionssysteme (OLTP) fokussieren auf kurze, atomare Transaktionen mit starken Konsistenz- und Integritätsgarantien. Die Systeme bearbeiten schnell viele diskrete Schreib-/Lesezugriffe und unterstützen gleichzeitige Transaktionen unter konstanter Reaktionszeit. Data-Warehouse-Systeme (DWH) dienen der historischen, integrierten Analyse großer Datenmengen; sie bevorzugen Stützung von komplexen Abfragen und Analysen über große Zeiträume hinweg. Typische architecture/Design-Anforderungen:

- OLTP: normalisierte Schemata, transaktionale Integrität, niedrige Latenz pro Transaktion, hohes Parallelitätsniveau.
- DWH: denormalisierte oder schneeblatt-/sternförmige Schemata (Star/Snowflake), ETL/ELT-Prozesse, Historisierung, analytische Abfragen und Reporting, ggf. zeitbasierte Partitionierung.
- Architektur-Differenzen: Online-Transaktionsverarbeitung vs. Online-Analytik; Trade-offs bei Konsistenz (ACID) vs. Verfügbarkeit/Partitionstoleranz (CAP) je nach System, Einsatz von Implementierungen wie Snapshot-Isolation, MVCC, Materialized Views.

(b) Lösung:

Fact/Dim	Beschreibung
FactSales	$\text{fact}_i dPK, \text{date}_k eyFK, \text{product}_k eyFK, \text{store}_k eyFK, \text{units}_s old, revenue$
DimDate	$\text{date}_k eyPK, date, month, quarter, year$
DimProduct	$\text{product}_k eyPK, \text{product}_n ame, category, price$
DimStore	$\text{store}_k eyPK, \text{store}_n ame, city, region$

(c) Lösung: SQL-ähnliche Abfrage (Star-Schema, aktueller Monat):

```
SELECT ds.store_name, SUM(fs.revenue) AS revenue
FROM FactSales fs
JOIN DimStore ds  ON fs.store_key = ds.store_key
JOIN DimDate  dd   ON fs.date_key  = dd.date_key
WHERE dd.date  >= DATE_TRUNC('month', CURRENT_DATE)
  AND dd.date  <  DATE_TRUNC('month', CURRENT_DATE) + INTERVAL '1 month'
GROUP BY ds.store_name;
```

Hinweis: Die Datums-Syntax hängt vom jeweiligen SQL-D Dialekt ab. Alternativ können Start/Ende eines Monats als Parameter $(\text{start}_o f_m onth, start_o f_n ext_m onth) verwendetwerden.$

(d) Lösung: Eine grobe ETL-Pipeline umfasst:

- Extract (E): Extraktion relevanter Daten aus operativen Quellsystemen (OLTP-Systeme, Logs, Dateien) inkl. inkrementeller Loads, Handling von Schema-Änderungen.
- Transform (T): Datenbereinigung (Duplikate, Inkonsistenzen), Standardisierung von Formaten, Normalisierung/Denormalisierung je nach Ziel, Conform Dimensions, Berechnung von Kennzahlen, Handling slowly changing dimensions (SCD).

- Load (L): Laden in das Data Warehouse (erneute oder inkrementelle Loads), Implementierung von Indizes/Partitionen, Materialized Views, ggf. Aktualisierung von Aggregationen.

Aufgabe 3.

(a) Lösung:

- Überwacht (supervised) Lernen: Lernziel ist die Vorhersage basierend auf gekennzeichneten Trainingsdaten. Typische Aufgaben: Klassifikation (categorical labels, z. B. Spam/Kein Spam), Regression (kontinuierliche Zielwerte, z. B. Hauspreis).
- Unüberwacht (unsupervised) Lernen: Lernziel ohne gelabelte Zielwerte. Typische Aufgaben: Clustering (Gruppierung ähnlicher Objekte, z. B. Kundensegmentierung), Dimensionsreduktion (z. B. PCA) zur Visualisierung oder Vorverarbeitung.
- Praxisbeispiele: Supervised – Kreditrisiko-Klassifikation; Unsupervised – Kundensegmentierung mittels K-Means.

(b) Lösung: Grundalgorithmus von k-Means:

1. Initialisierung: Wähle k Start-Zentren $\mu_1, \ldots, \mu_k$ (z. B. zufällig aus den Daten oder per k-means++).
2. Zuordnung: Weisen Sie jedes Datenpunkt x_i dem nächsten Zentrum zu:
$$c(i) = \arg\min_j \| x_i - \mu_j \|^2.$$
3. Neuberechnung der Zentren: Für jede Cluster-Klasse C_j berechne das neue Zentrum
$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i.$$
4. Konvergenz: Wiederhole Schritte 2–3, bis sich die Zuordnungen stabilisieren oder die Zentren sich nicht mehr wesentlich bewegen.

Annahmen: Abstandsmaß (häufig euklidisch), kugelförmige, ähnliche Clustergrößen; Distanzbasierte Zuordnung; Konvergenz zu lokalen Optima.

(c) Lösung: Gegebene Punkte: $(1,2), (2,1), (3,4), (4,3)$. Eine einfache Trennregel ist die Geradentypisierung über die Summe der Koordinaten:

$$x + y = 5 \quad \text{bzw.} \quad y = -x + 5.$$

Beispiele:

- Punkte mit $x + y < 5$ (z. B. (1,2) mit Summe 3) auf einer Seite.
- Punkte mit $x + y > 5$ (z. B. (4,3) Summe 7) auf der anderen Seite.

Damit lässt sich eine einfache lineare Trennregel darstellen, die die Punkte () in zwei Klassen trennt. Hinweis: Mehrere Geraden könnten je nach Zuschneidung gewählt werden; diese hier ist eine plausible einfache Wahl.

(d) Lösung: Streaming-Daten können in der Data-Science-Pipeline durch kontinuierliche Ingestion, zeitbasierte Fenster und Online-/Incremental-Learning nutzbar gemacht werden. Typische Modelle für kontinuierliche Datenströme:

- Online-Learn algorithms (z. B. SGD, Online Newton, incremental Logistic Regression)
- Modelle mit Fensterbausteinen (Sliding/Temporal Windows) zur Concept-Drift-Anpassung
- Drift-aware oder adaptive Modelle, Online Clustering (z. B. streaming k-Means)

Dazu gehört auch das ständige Monitoring der Metriken, das Handling von Konzeptdrift und das Aktualisieren von Modellen in kurzen Abständen.

Aufgabe 4.

(a) Lösung: Ein Data-Stream-Management-System (DSMS) ist eine Engine zur kontinuierlichen Verarbeitung endloser Datenströme. Es unterstützt kontinuierliche Abfragen, Streaming-Operatoren (Filter, Join, Aggregation), Fenstermechanismen (zeit- bzw. ereignisbasiert) und Outputs in nahezu Echtzeit. Unterschiede zu klassischen Datenbanken:

- Unendliche, zeitbasierte oder ereignisbasierte Datenströme vs. finite Tabellen.
- Fokus auf geringe Latenz und kontinuierliche Abfrageausführung statt auf vollständige Transaktionsverarbeitung.
- Fenster- und streaming-spezifische Operatoren (z. B. Sliding/Tumbling Windows) statt rein relationaler Abfragen.

(b) Lösung: Grobe Architektur eines DSMS:

- Quellen (Streams): Datenquellen, Sensoren, Log-Dateien, Message-Brokers.
- Operatoren: Filter, Projection, Join, Aggregation, Windowing, Annotationslogik.
- Fenstermechanismen: zeitbasierte Fenster (z. B. 60 Minuten), tuple-basiert (Anzahl der Tupel) oder hybride Fenster.
- Outputs: Sinks, Dashboards, Alerts, persistierte Storages.

Fenster dienen dazu, endliche Teilmengen aus einem kontinuierlichen Strom zu definieren, so dass deterministische, rechenbare Aggregationen über begrenzte Sequenzen von Tupeln möglich werden (z. B. Durchschnitt pro Produkt in einem 60-Minuten-Fenster).

(c) Lösung:

```
SELECT product_id, AVG(price) AS avg_price
FROM Stream(Sales)
WINDOW 60 MINUTES
GROUP BY product_id;
```

(d) Lösung: Drei typische Anwendungsbereiche für DSMS in der Praxis:

- Echtzeit-Überwachung und Alarmierung (z. B. Fraud Detection, Netzwerk-Sicherheitswarnungen)
- Online-Analytik und Dashboards (Streaming BI, Echtzeit-KPI-Überblick)
- IoT- und Ereignisbasierte Anwendungen (Sensoren, Industrie 4.0, Smart Cities)