Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Datenmanagement & -analyse

Übung 6 – Datenbankabfragen und Transaktionen

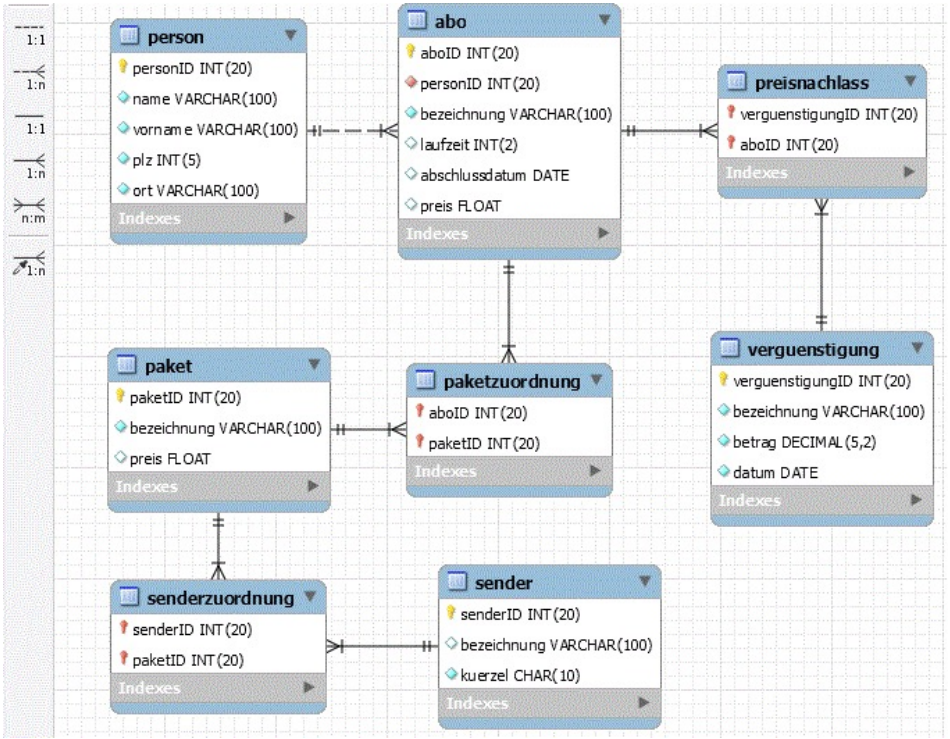
Dr. Nikolai Stein
Lehrstuhl für WI & BA
Julius-Maximilians-Universität Würzburg

Sommersemester 2021

Aufgabe 1 – Sky-Verwaltung

Das Datenbankmodell dient der Verwaltung von Abonnements. Personen sind dabei Abonnements zugeordnet, die sie zu einem bestimmten Preis erstanden haben. Für ein Abonnement können Vergünstigungen ausgegeben werden, die sich in Form von Preisnachlässen äußern.

Ein Abo integriert immer eine bestimmte Anzahl an Paketen, die wiederum bestimmte Sender integrieren.



Aufgabe 1 – Sky-Verwaltung

Lösen Sie die folgenden Aufgaben mit Hilfe von Datenbankabfragen:

a) Zeigen Sie für jedes Abo die dazugehörige Person an.

```sql
SELECT *
FROM abo
LEFT JOIN person
ON abo.personid = person.personID;
```

b) Sie möchten gerne wissen, welche Sender welchem Paket zugeordnet sind. Lassen Sie sich die senderID, die Senderbezeichung sowie die paketID, die Paketbezeichung und den Paketpreis anzeigen.

```sql
SELECT sender.senderid, paket.paketid, paket.preis, sender.bezeichnung AS sender_name, paket.bezeichnung AS paket_name
FROM paket
INNER JOIN senderzuordnung
ON paket.paketid = senderzuordnung.paketID
INNER JOIN sender
ON senderzuordnung.senderid = sender.senderID;
```

Aufgabe 1 – Sky-Verwaltung

Lösen Sie die folgenden Aufgaben mit Hilfe von Datenbankabfragen:

c) Lassen Sie sich die Anzahl aller Sender eines Pakets in einer neuen Spalte ausgeben.

```sql
SELECT paket.paketID, paket.bezeichnung, paket.preis, COUNT(*) AS anzahl_sender
FROM paket
INNER JOIN senderzuordnung
ON paket.paketid = senderzuordnung.paketID
GROUP BY paket.paketid;
```

d) Lassen Sie sich die Anzahl der Pakete anzeigen, welche mindestens einen Sender mit der Bezeichnung „Sport“ beinhaltet.

```sql
SELECT COUNT(DISTINCT(paket.paketid)) AS anzahl_sport_pakete
FROM paket
LEFT JOIN senderzuordnung
ON paket.paketID = senderzuordnung.paketid
INNER JOIN sender
ON senderzuordnung.senderid = sender.senderid
WHERE sender.bezeichnung LIKE '%Sport%';
```

Aufgabe 1 – Sky-Verwaltung

Lösen Sie die folgenden Aufgaben mit Hilfe von Datenbankabfragen:

e) Herr Müller hat sich über seine Abrechnung beschwert. Er behauptet, dass er für Pakete bezahlt, die er nicht empfangen kann. Finden Sie heraus, welche Pakete seinem Abo zugeordnet sind.

```sql
SELECT person.personid, person.name, person.vorname, abo.aboid, abo.bezeichnung AS abo_bez, paket.paketid, paket.bezeichnung AS paket_bez
FROM person
INNER JOIN abo
ON person.personid = abo.personid
LEFT JOIN paketzuordnung
ON abo.aboid = paketzuordnung.aboid
LEFT JOIN paket
ON paketzuordnung.paketid = paket.paketID
WHERE person.name = 'Müller';
```

f) a.) Lassen Sie sich die Abo-Laufzeit anzeigen und korrigieren Sie diese falls nötig auf 12 Monate.

```sql
UPDATE abo
SET laufzeit = 12
WHERE abo.personid = (SELECT personid
                      FROM person
                      WHERE name = 'Maier');
```

Aufgabe 1 – Sky-Verwaltung

Lösen Sie die folgenden Aufgaben mit Hilfe von Datenbankabfragen:

f) b.) Ebenso sollen alle Vergünstigungen angezeigt werden, die die Kundin erhalten hat. Hierfür lassen Sie sich lediglich den Vor- und Nachnamen der Kunden, sowie die Bezeichnung und den Betrag der Vergünstigung anzeigen.

```sql
SELECT person.vorname, person.name, verguenstigung.bezeichnung, verguenstigung.betrag
FROM person
LEFT JOIN abo
ON person.personid = abo.personid
LEFT JOIN preisnachlass
ON abo.aboid = preisnachlass.aboid
LEFT JOIN verguenstigung
ON preisnachlass.verguenstigungID = verguenstigung.verguenstigungID
WHERE name = 'Maier';
```

g) Lassen Sie sich die Bezeichnung, den Betrag und die Häufigkeit aller Vergünstigungen ausgeben. Es sollen nur Vergünstigungen mit einem größeren Einzelbetrag als 12€ angezeigt werden.

```sql
SELECT verguenstigung.bezeichnung, verguenstigung.betrag, COUNT(aboid) AS haeufigkeit
FROM verguenstigung
LEFT JOIN preisnachlass
ON verguenstigung.verguenstigungID = preisnachlass.verguenstigungID
GROUP BY verguenstigung.verguenstigungid
HAVING betrag >= 12;
```

Aufgabe 1 – Sky-Verwaltung

Lösen Sie die folgenden Aufgaben mit Hilfe von Datenbankabfragen:

h) Lassen Sie sich die Abos und den ermittelten kumulierten Paketpreis anzeigen. Fügen Sie eine Spalte mit den tatsächlichen Kosten und der Differenz hinzu.

```sql
SELECT abo.aboid, abo.bezeichnung AS abo_name, abo.preis AS abo_preis, paket.bezeichnung AS paket_name,
SUM(paket.preis) AS sum_paket_preis, ROUND((SUM(paket.preis) - abo.preis), 2) AS differenz
FROM abo
LEFT JOIN paketzuordnung
ON abo.aboid = paketzuordnung.aboid
LEFT JOIN paket
ON paketzuordnung.paketid = paket.paketid
GROUP BY abo.aboid;
```

i) Lasse Sie sich alle Abos anzeigen, welche die höchste Vergünstigung (Betrag = 20) erhalten haben und gleichzeitig das Paket mit der ID 2 gebucht haben.

```sql
SELECT *
FROM abo
INNER JOIN preisnachlass
ON abo.aboID = preisnachlass.aboid
INNER JOIN verguenstigung
ON preisnachlass.verguenstigungID = verguenstigung.verguenstigungid
INNER JOIN paketzuordnung
ON abo.aboid = paketzuordnung.aboid
WHERE verguenstigung.betrag = 20 AND paketzuordnung.paketID = 2;
```

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Transaktionen und ACID

- Wozu Transaktionen?
  - Häufig wollen **mehrere Nutzer/Anwendungen parallel** auf DB zugreifen (z.B. ERP-System, Web Shop, …)
  - Kann zu **Konkurrenzsituationen** führen

  → **Gefahr für Konsistenz der Datenbank**

- Was ist eine Transaktion?
  - **Zusammenfassung von Operationen** auf DB als **eine Arbeitseinheit**
  - Überführt DB von konsistenten in **konsistenten Zustand**
  - Zwei grundlegende Anforderungen
    - **Recovery**: Behebung von (unvermeidbaren) Fehlersituationen
    - **Synchronisation**: Parallelisierung von Transaktionen

Kemper, Eickler (2013)

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Eigenschaften einer Transaktionen: ACID

- **Atomicity** (Abgeschlossenheit)
- **Consistency** (Konsistenzerhaltung)
- **Isolation** (Isoliertheit)
- **Durability** (Dauerhaftigkeit)

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Abgeschlossenheit und Konsistenzerhaltung

- Atomicity
  - Transaktion ist **unteilbar**
  - D.h. eine Transaktion wird entweder **ganz** oder **gar nicht** ausgeführt
  - erfolgreiche Transaktionen werden mit `commit` persistiert
  - Andernfalls gelten sie als abgebrochen (`abort`)
  - Änderungen der Transaktion an der Datenbank müssen rückgängig gemacht werden (`rollback`)

- Consistency
  - Transaktion versetzt Datenbank in einen **konsistenten** Zustand, sofern sie vor der Transaktion konsistent war
  - Inkonsistente Zustände werden **komplett** zurückgesetzt
  - Zwischenstände dürfen inkonsistent sein
  - Konsistent: Fachliche Korrektheit (**Integrität**)
  - Beispiel
    - Doppische Buchführung: Keine Buchung ohne Gegenbuchung
    - Redundante Daten müssen konsistent gehalten werden

Kemper, Eickler (2013)

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Isoliertheit und Dauerhaftigkeit

- Isolation
  - Nicht vollständig ausgeführte Transaktionen haben **keinen Einfluss auf parallele Transaktionen und sind nicht sichtbar**
  - Jede Transaktion wird behandelt, als wäre sie die einzige auf der Datenbank
  - Ergebnisse werden erst nach `commit` sichtbar
- Durability
  - Effekt einer Transaktion **bleibt wirksam**, sobald sie durch `commit` bestätigt wurde
  - Bezieht sich weniger auf fachliche Integrität sondern auf **Sicherung der Daten** vor
    - Systemausfällen
    - Hardwarefehlern
    - weiteren externen Risiken

Kemper, Eickler (2013)

Lost Update – Verlorene Aktualisierung (write-write-conflict)

Beschreibung:

- Transaktion 1 schreibt einen Wert
- Transaktion 2 überschreibt diesen Wert ohne die Aktualisierung zu berücksichtigen
- Beide **Transaktionen lesen konsistente Datenbankzustände**

Problem:

- Das **Update** durch Transaktion 1 **geht verloren**
- Datenbank wird möglicherweise **inkonsistent**
- Schwierig zu entdecken, da Datenbank nicht zwingend inkonsistent

Dirty Read (Schreib-Lese-Konflikt, write-read-conflict)

Beschreibung:

- Transaktion 1 liest einen von Transaktion 2 veränderten Wert (und arbeitet damit weiter)
- Transaktion 2 wurde aber noch nicht via **commit** bestätigt

Problem:

- Transaktion 2 kann durch **rollback** zurückgerollt werden
- Transaktion 1 arbeitet dann auf einem **ungültigen Wert**
- **Verstoß** gegen Eigenschaft der **Isoliertheit**

Non-repateable Read (nicht-wiederholbares Lesen, read-write-conflict)

Beschreibung:

- Transaktion 1 liest ein Datum mehrfach
- Transaktion 2 ändert dies währenddessen
- Beide **Transaktionen lesen konsistente Datenbankzustände**

Problem:

- Transaktion 1 operiert auf **veralteten Daten**
- Datenbank wird **möglicherweise inkonsistent**

Phantom Read (Inconsistent Read)

Beschreibung:

- Entsteht bei **Operationen auf mehreren Tupeln**
- Ähnlich dem non-repeatable Read
- Bezieht sich auf **Menge von Daten** nicht auf ein Datum

Problem:

- Operation auf Basis von **veralteten** Daten
- Datenbank wird **möglicherweise inkonsistent**

Aufgabe 2 - Transaktionen

Sie sind Betreiber eines Online Shops und speichern Produktinformationen (ID, Name, Lagerbestand) in einer Tabelle **Produkte**.

Bilden Sie den folgenden Prozess in der Tabelle ab:

1. Kunde 1 öffnet die Seite für Produkt 1 und sieht den Lagerbestand von 12
2. Kunde 2 öffnet die Seite für Produkt 1 und sieht den aktuellen Warenbestand
3. Kunde 2 legt drei Einheit von Produkt 1 in den Warenkorb
4. Kunde 2 wechselt zum Check-Out und der Bestand in der Datenbank wird aktualisiert
5. Kunde 2 bezahlt den Artikel und beendet die Transaktion
6. Kunde 1 legt zwei Einheiten von Produkt 1 in den Warenkorb
7. Kunde 1 wechselt zum Check-Out und der Bestand in der Datenbank wird aktualisiert
8. Kunde 1 bezahlt den Artikel und beendet die Transaktion

Transaktion 1	Transaktion 2	
read a1 <- A		a1 = 12
	read a2 <- A	a2 = 12
	a2 = a2 - 3	a2 = 9
	write a2 -> A	A = 9
	commit	
a1 = a1 - 2		a1 = 10
write a1 -> A		A = 10
commit		

Aufgabe 2 - Transaktionen

Sie sind Betreiber eines Online Shops und speichern Produktinformationen (ID, Name, Lagerbestand) in einer Tabelle **Produkte**.

Bilden Sie den folgenden Prozess in der Tabelle ab:

1. Kunde 1 öffnet die Seite für Produkt 1 und sieht den Lagerbestand von 12
2. Kunde 2 öffnet die Seite für Produkt 1 und sieht den aktuellen Warenbestand
3. Kunde 2 legt drei Einheit von Produkt 1 in den Warenkorb
4. Kunde 2 wechselt zum Check-Out und der Bestand in der Datenbank wird aktualisiert
5. Kunde 2 bezahlt den Artikel und beendet die Transaktion
6. Kunde 1 legt zwei Einheiten von Produkt 1 in den Warenkorb
7. Kunde 1 wechselt zum Check-Out und der Bestand in der Datenbank wird aktualisiert
8. Kunde 1 bezahlt den Artikel und beendet die Transaktion

Transaktion 1	Transaktion 2	
read a1 ←A		a1 = 12
	read a2 ← A	a2 = 12
	a2 = a2 - 3	a2 = 9
	write a2→A	A=9
	commit	
a1 = a1 - 2		a1 = 10
write a1→A		A = 10
commit		

Lost Update

Aufgabe 3 - Transaktionen

Sie sind Betreiber eines Online Shops und speichern Produktinformationen (ID, Name, Lagerbestand) in einer Tabelle **Produkte**.

Bilden Sie den folgenden Prozess in der Tabelle ab:

1. Kunde 1 öffnet die Seite für Produkt 1 und sieht den Lagerbestand von 12
2. Kunde 1 legt eine Einheit von Produkt 1 in den Warenkorb
3. Kunde 1 wechselt zum Check-Out und der Bestand in der Datenbank wird aktualisiert
4. Kunde 2 öffnet die Seite für Produkt 1 und sieht den aktuellen Warenbestand
5. Kunde 2 legt eine Einheit von Produkt 1 in den Warenkorb
6. Kunde 2 wechselt zum Check-Out und der Bestand in der Datenbank wird aktualisiert
7. Kunde 2 bezahlt den Artikel und beendet die Transaktion
8. Die Kreditkarte von Kunde 1 wird abgelehnt und der Kunde bricht die Transaktion ab

Transaktion 1	Transaktion 2	
read a1 <- A		a1 = 12
a1 = a1 – 1		a1 = 11
write a1 -> A		A = 11
	read a2 <- A	a2 = 11
	a2 = a2 – 1	a2 = 10
	write a2 -> A	A = 10
	commit	
rollback		

Aufgabe 3 - Transaktionen

Sie sind Betreiber eines Online Shops und speichern Produktinformationen (ID, Name, Lagerbestand) in einer Tabelle **Produkte**.

Bilden Sie den folgenden Prozess in der Tabelle ab:

1. Kunde 1 öffnet die Seite für Produkt 1 und sieht den Lagerbestand von 12
2. Kunde 1 legt eine Einheit von Produkt 1 in den Warenkorb
3. Kunde 1 wechselt zum Check-Out und der Bestand in der Datenbank wird aktualisiert
4. Kunde 2 öffnet die Seite für Produkt 1 und sieht den aktuellen Warenbestand
5. Kunde 2 legt eine Einheit von Produkt 1 in den Warenkorb
6. Kunde 2 wechselt zum Check-Out und der Bestand in der Datenbank wird aktualisiert
7. Kunde 2 bezahlt den Artikel und beendet die Transaktion
8. Die Kreditkarte von Kunde 1 wird abgelehnt und der Kunde bricht die Transaktion ab

Transaktion 1	Transaktion 2	
read a1 ←A		a1 = 12
a1 = a1 - 1		a1 = 11
write a1→A		A = 11
	read a2 ←A	a2 = 11
	a2 = a2 - 1	a2 = 10
	write a2→A	A = 10
	commit	
rollback		

Dirty Read

Aufgabe 4 - Transaktionen

Sie sind Betreiber eines Online Shops und speichern Produktinformationen (ID, Name, Lagerbestand) in einer Tabelle **Produkte**.

Bilden Sie den folgenden Prozess in der Tabelle ab:

1. Kunde 1 öffnet die Seite für Produkt 1 und sieht den Lagerbestand von 1
2. Kunde 1 legt eine Einheit von Produkt 1 in den Warenkorb legen und System prüft Verfügbarkeit
3. Kunde 2 öffnet die Seite für Produkt 1 und sieht den Lagerbestand
4. Kunde 2 legt eine Einheit von Produkt 1 in den Warenkorb legen und System prüft Verfügbarkeit
5. Kunde 2 wechselt zum Check-Out und System liest aktuellen Bestand aus der Datenbank
6. System reduziert Bestand um eine Einheit und Datenbank wird aktualisiert
7. Kunde 2 bezahlt den Artikel und beendet die Transaktion
8. Kunde 1 wechselt zum Check-Out und System liest aktuellen Bestand aus der Datenbank
9. System reduziert Bestand um eine Einheit und Datenbank wird aktualisiert
10. Kunde 1 bezahlt den Artikel und beendet die Transaktion

Transaktion 1	Transaktion 2	
read a1 <- A		a1 = 1
v1 = a1 - 1		v1 = 0
	read a2 <- A	a2 = 1
	v2 = a2 – 1	v2 = 0
	read a2 <- A	a2 = 1
	a2 = a2 - 1	a2 = 0
	write a2 -> A	A = 0
	commit	
read a1 <- A		a1 = 0
a1 = a1 - 1		a1 = -1
write a1 -> A		A = -1
commit		

Aufgabe 4 - Transaktionen

Sie sind Betreiber eines Online Shops und speichern Produktinformationen (ID, Name, Lagerbestand) in einer Tabelle **Produkte**.

Bilden Sie den folgenden Prozess in der Tabelle ab:

1. Kunde 1 öffnet die Seite für Produkt 1 und sieht den Lagerbestand von 1
2. Kunde 1 legt eine Einheit von Produkt 1 in den Warenkorb legen und System prüft Verfügbarkeit
3. Kunde 2 öffnet die Seite für Produkt 1 und sieht den Lagerbestand
4. Kunde 2 legt eine Einheit von Produkt 1 in den Warenkorb legen und System prüft Verfügbarkeit
5. Kunde 2 wechselt zum Check-Out und System liest aktuellen Bestand aus der Datenbank
6. System reduziert Bestand um eine Einheit und Datenbank wird aktualisiert
7. Kunde 2 bezahlt den Artikel und beendet die Transaktion
8. Kunde 1 wechselt zum Check-Out und System liest aktuellen Bestand aus der Datenbank
9. System reduziert Bestand um eine Einheit und Datenbank wird aktualisiert
10. Kunde 1 bezahlt den Artikel und beendet die Transaktion

Transaktion 1	Transaktion 2	
read a1 ←A		a1 = 1
v1 = a1 - 1		v1 = 0
	read a2 ←A	a2 = 1
	v2 = a2 - 1	v2 = 0
	read a2 ←A	a2 = 1
	a2 = a2 – 1	a2 = 0
	a2→A	A = 0
	commit	
read a1 ←A		a1 = 0
a1 = a1 – 1		a1 = -1
a1→A		A = -1
commit		

Non-repeatable Read