Download Skript

Informatik 8. Klasse - Schuljahr 2022/23

Lehrinhalte

- Lehrplaninhalte

Remote-Zugriff auf Schulserver

Kapitel

- 1) Datenbanken

Leistungsbeurteilung

- **Schularbeiten (SA)**
  - 2x SA (2h) pro Semester
- **Mitarbeit (MA)**
  - Aktive Mitarbeit im Unterricht (aMA)
  - Mündliche Stundenwiederholungen (mMA)
  - Schriftliche Stundenwiederholungen (sMA)
- **Praktische Arbeiten (PA)**
  - 1x praktischer Arbeitsauftrag pro Woche

Stoff für die 1. Schularbeit in Informatik

24.11.2022 1+2.Stunde

Stoff für die 2. Schularbeit in Informatik

16.03.2023 1-3.Stunde

2) DATENBANKEN

2.1) Allgemeines

2.1.1) Definitionen

2.1.1.1) Datenbanksystem:

Ein Datenbanksystem ist ein computergestütztes System bestehend aus einer Datenbasis zur Beschreibung eines Ausschnitts der realen Welt sowie Programmen zum geregelten Zugriff auf die Datenbasis.

2.1.1.2) Datenbankverwaltungssystem (DBMS-data base management system)

Ist jener Teil der Software die zwischen den eigentlichen Daten und den Benutzern der Daten liegt und alle Anfragen der Benutzer verarbeitet. Sie stellt jene Einrichtung zur Verfügung die notwendig sind um neue Daten anzulegen, zu löschen, abzufragen und zu verändern

2.1.2) Motivation

Die ersten Computer unterstützten Informationssysteme, wurden in Form von Einzellösungen, d.h. durch einzelne Anwendungsprogramme mit privaten Dateien realisiert. Diese Programme verwendeten unmittelbar das zugrunde liegende Dateisystem auf den jeweiligen Rechner. Gleichartige Daten wurden in separaten Dateien gespeichert, die selbst wieder aus einzelnen Datensätzen bestanden.

Produktion
Angestellte
Teile

Verkauf
Kunden
Teile

Fakturierung
Kunden
Teile

Personalabteilung
Kunden
Gehalt

Diese Systeme waren schwer wartbar da mehrfach verwendete Daten auch mehrfach gespeichert wurden, deshalb entstand die integrierte Datenverarbeitung bei der Dateien in mehreren Anwendungsprogrammen verwendet werden. Doch auch die separate Abspeicherung von teilweise in Beziehung stehenden Daten würde zu schwerwiegenden Problemen führen:

- **Redundanz** - Dieselben Informationen werden doppelt gespeichert.
- **Inkonsistenz** - Dieselben Informationen werden in unterschiedlichen Versionen gespeichert.
- **Integritätsverletzung** - Die Einhaltung komplexer Intigritätsbedingungen fällt schwer.
- **Verknüpfungseinschränkung** - Logisch verwandte Daten sind schwer zu verknüpfen wenn sie in isolierten Dateien liegen.
- **Mehrbenutzerprobleme** - Gleichzeitiges Editieren der Datei führt zur Anomalie (lost update).
- **Verlust von Daten** - Außer einem kompletten Backup ist kein Recovery-Mechanismus vorhanden.
- **Sicherheitsprobleme** - Abgestufte Zugriffsrechte können nicht implementiert werden.
- **Hohe Entwicklungskosten** - Für jedes Anwendungsprogramm müssen die Fragen zur Dateiverwaltung erneut gelöst werden.

Heute werden Informationssysteme meist mit Hilfe von Datenbanksystemen realisiert.

Einerseits ermöglicht diese Trennung zwischen Anwendungsprogrammen und Daten die sogenannte **physische Datenunabhängigkeit**, d.h. Programme sind von den konkreten Speicher- und Zugriffsmethoden unabhängig. Andererseits stellen Datenbanksysteme ein Datenmodell, also eine Sprache zur Beschreibung von Datenstrukturen, zur Verfügung, die es ermöglicht einzelne Programme auf speziellen logischen Darstellungen der gespeicherten Datenbank arbeiten zu lassen (logische Unabhängigkeit).

2.1.3) Funktionalität von Datenbanksystemen

2.1.3.1) Persistente Datenhaltung

Ein DBMS muss Mechanismen zur Verfügung stellen, die eine **persistente Speicherung von Daten** garantieren, d.h. dass die Daten in der DB über die Ausführungszeit von Programmen hinaus erhalten bleiben. Die Daten werden üblicherweise auf einem Hintergrundspeicher (Sekundärspeicher - HDD) persistent gehalten. Nachdem Programme nur auf Daten im Hauptspeicher (Primärspeicher) direkt zugreifen können werden Ausschnitte der Datenbank zeitweise auch in einem Teil des Hauptspeichers, dem Datenbankpuffer, verwaltet.
Nachdem ein **Plattenzugriff sehr viel länger dauert als ein Hauptspeicherzugriff**, sind spezielle Techniken notwendig um unnötige Plattenzugriffe zu vermeiden.
Spezielle Puffersatzstrategien werden verwendet um bei Platzmangel im Datenbankpuffer zu entscheiden welche Blöcke wieder auf die Platte ausgelagert werden (z.B. **least recently used, least frequently used**). Aus Effizienzgründen gruppieren sogenannten Clustertechniken Datensätze so, dass jene Datensätze, auf die oft gemeinsam zugegriffen wird, physisch benachbart gespeichert werden. Weiters werden verschiedene Indextechniken verwendet um Daten auf einem Hintergrundspeicher rasch zu finden.

2.1.3.2) Recovery

DBMS unterstützen Änderungen in der Datenbank durch Transaktionen. Eine Transaktion ist eine Folge von Aktionen (Lese- und Schreibzugriffe auf Daten in der DB),die eine Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand überführt. Die Recoveryeinheit eines **DBMS gewährleistet die Atomarität und die Dauerhaftigkeit (Persistenz) von**

Transaktionen trotz eventuell bei der Transaktion aufgetretenen Hard- oder Softwarefehlern.

Atomarität bedeutet, dass **entweder alle Aktionen** einer Transaktion ausgeführt werden **oder keine**.

Dauerhaftigkeit bedeutet dass **alle Effekte** einer einmal erfolgten Transaktion **trotz aufgetretenen Fehlern erhalten bleiben**.

Bsp. zu Atomarität: Überweisung eines Geldbetrages von einem Konto auf ein Sparbuch

```
- Kontostand lesen
- Kontostand schreiben
- Sparbuch lesen
- Sparbuch schreiben
```

Angenommen während der Überweisung tritt nach der Abbuchung aber noch vor der Aufbuchung ein Systemabsturz ein, so möchten die Kunden davon ausgehen können, dass sich nach einem Wideranlauf der gesamte Geldbetrag noch auf dem Konto befindet.

Bsp. zu Dauerhaftigkeit:
Angenommen wir zahlen einen Millionengewinn im Lotto auf unser Konto ein, dann stellt die Dauerhaftigkeit von Transaktionen sicher, dass der Gewinn auch nach einem Systemneustart immer noch auf dem Konto liegt. Für den Wiederanlauf verwenden die meisten DBS ein Log-Protokoll. In diesem Log-Protokoll werden der Start, das Ende und der Abbruch von Transaktionen verzeichnet, sowie die von Transaktionen durchgeführten Modifikationen von Datensätzen (Einfügen, Löschen und Ändern von Datensätzen). Zu jeder Änderung wird der alte Datensatz (before image) und der neue Datensatz (after image) im Log-Protokoll verzeichnet. Beim Wiederanlauf werden alle nicht beendeten Transaktionen unter Verwendung der before-images zurückgesetzt und alle bereits erfolgreich abgeschlossenen Transaktionen nachgeholt.

2.1.3.3) Concurrency Control

Die Concurrency Control - Einheit eines DBMS ermöglicht mehreren Benutzern eine Datenbank gemeinsam zur selben Zeit zu nutzen ohne ihre Konsistenz zu gefährden. Das traditionell verwendete **Korrektheitskriterium für parallele (oder verzahnte) Ausführung von Transaktionen im Mehrbenutzerbetrieb ist die Serialisierbarkeit**.
Die Serialisierbarkeit garantiert folgende Eigenschaft:
Das Ergebnis der beliebigen Parallelausführung mehrerer Transaktionen entspricht dem Ergebnis irgendeiner Hintereinander-Ausführung dieser Transaktion.

Bsp.: Angenommen wir wollen unsere Telefonrechnung (50€) bezahlen. Wir gehen zur Bank, wo die Abbuchung vom Konto durchgeführt wird. Diese Abbuchung wird nun gleichzeitig mit der Gehaltsbuchung (1000€) auf das Konto durchgeführt.

Angenommen der aktuelle Kontostand beträgt 2000€. Nachdem beide Transaktionen abgeschlossen wurden, ist der Kontostand auf 1950€. Eine Hintereinanderausführung hätte aber 2950€ ergeben, d.h. diese verzahnte Ausführung ist nicht serialisierbar und daher nicht korrekt.

Um Serialisierbarkeit von Transaktionen zu gewährleisten verwenden die meisten DBMS Sperrverfahren.

Dabei legt eine Transaktion auf Datenobjekte die sie schreiben oder lesen soll, eine **Sperre**. Besitzt eine Transaktion auf einem Datenobjekt eine Sperre und fordert eine andere Transaktion für dieses Datenobjekt ebenfalls eine Sperre an, so wird diese Sperre nur dann gewährt, wenn die neu angeforderte Sperre mit der bereits bestehenden Sperre verträglich ist. Ist sie es nicht, so muss die neue Transaktion **auf die Freigabe der bestehenden Sperre warten**.

Meist werden 2 Typen von Sperren verwendet:

- Geteilte Lese-Sperren
- Exklusive Schreib-Sperren

Lesesperren verschiedener Transaktionen für dasselbe Datenobjekt sind miteinander verträglich, eine Schreibsperre ist mit keiner Sperre anderer Transaktionen verträglich.

Das Sperren von Datenobjekten ist alleine jedoch nicht ausreichend um die Serialisierbarkeit paralleler Transaktionen zu gewährleisten. Es muss darüber hinaus ein **Sperrprotokoll** eingehalten werden. Das am meisten gebräuchliche Sperrprotokoll ist das **2-Phasen-Sperrverfahren**. Eine Transaktion erfüllt das 2-Phasen-Sperrverfahren, wenn es nach der 1. Freigabe einer Sperre keine neue Sperre mehr anfordert. Weiters garantiert die Concurrency Control-Einheit eines DBMS die Isolation von Transaktionen. **Isolation** bedeutet, dass **Effekte nach einer Transaktion erst nach ihrem erfolgreichem Abschluss für andere Transaktionen sichtbar** werden.

2.1.3.4) ACID - Atomicity Consistency Isolation Durability (zu d. Dt. AKID - Atomarität, Konstistenz, Isolation, Dauerhaftigkeit)

Die Eigenschaften des Transaktionskonzeptes werden unter der Abkürzung **ACID** zusammengefasst:

- **Atomicity:** Eine Transaktion stellt eine nicht weiter zerlegbare Einheit dar, mit dem Prinzip: **„ALLES oder NICHTS“**
- **Consistency:** Nach Abschluss der Transaktion liegt wieder ein konsistenter Zustand vor, während der Transaktion sind inkonsistente Zustände erlaubt
- **Isolation:** Nebenläufig ausgeführte Transaktionen dürfen sich nicht beeinflussen, d.h. jede Transaktion hat den Effekt, den sie verursacht hätte, als ob sie allein im System gewesen wäre.
- **Durability:** Die Wirkung einer erfolgreich abgeschlossenen Transaktion bleibt dauerhaft in der Datenbank, auch nach einem späteren Systemfehler.

2.1.3.5) Datenschutz

DBMS bieten die Möglichkeit für einzelne Benutzer oder Benutzergruppen den Zugriff auf Ausschnitte der Datenbank zu beschränken. Dabei kann hinsichtlich der Art des Zugriffs zwischen Lese-, Änderungs-, Einfüge- und Löschzugriffe unterschieden werden.

2.1.4) Architektur eines Datenbanksystems

Moderne Datenbanksysteme unterstützen alle die **ANSI-SPARC-Architektur**:

Die Architektur wurde 1975 vom Standards Planning and Requirements Committee (SPARC) des American National Standards Institute (ANSI) entwickelt und hat das Ziel, den Benutzer einer Datenbank vor nachteiligen Auswirkungen von Änderungen in der Datenbankstruktur zu schützen.

Die drei Ebenen sind:

- **Die externe Ebene**, die den Benutzern und Anwendungen individuelle Benutzersichten bereitstellt. Beispiele: Formulare, Masken-Layouts, Listen, Schnittstellen.
- **Die konzeptionelle Ebene**, in der beschrieben wird, welche Daten in der Datenbank gespeichert sind, sowie deren Beziehungen zueinander. Designziel ist hier eine vollständige und redundanzfreie Darstellung aller zu speichernden Informationen. Hier findet die Normalisierung des relationalen Datenbankschemas statt.
- **Die interne Ebene (auch physische Ebene)**, die die physische Sicht der Datenbank im Computer darstellt. In ihr wird beschrieben, wie und wo die Daten in der Datenbank gespeichert werden. Designziel ist hier ein effizienter Zugriff auf die gespeicherten Informationen. Das wird meistens nur durch eine bewusst in Kauf genommene Redundanz erreicht (z. B. im Index werden die gleichen Daten gespeichert, die auch schon in der Tabelle gespeichert sind).

Die Vorteile des Drei-Ebenen-Modells liegen in der

- **physischen Datenunabhängigkeit**, da die interne von der konzeptionellen und externen Ebene getrennt ist. Physische Änderungen, z. B. des Speichermediums oder des Datenbankprodukts, wirken sich nicht auf die konzeptionelle oder externe Ebene aus.
- **logischen Datenunabhängigkeit**, da die konzeptionelle und die externe Ebene getrennt sind. Dies bedeutet, dass Änderungen an der Datenbankstruktur (konzeptionelle Ebene) keine Auswirkungen auf die externe Ebene, also die Masken-Layouts, Listen und Schnittstellen haben.

Allgemein kann also von einer **höheren Robustheit gegenüber Änderungen** gesprochen werden.

2.2) Datenmodellierung

Im Datenbankentwurf werden verschiedene Datenbankmodelle verwendet:

Konzeptionelle Datenmodelle (z.B.: ER-Modell) stehen **problemnahe Modellierungskonzepte** für die ersten Schritte des Datenbankentwurfs zur Verfügung und dienen zur **Kommunikation zwischen Endbenutzern und Datenbankentwicklern**.

Physische Modelle stellen **maschinennahe Konzepte** zur Verfügung und dienen zur **Beschreibung der Organisation von Daten in Dateien** sowie zur Beschreibung von **Zugriffsstrukturen** die ein rasches Einfügen, Suchen und Ändern von Daten ermöglichen.

Logische Datenmodelle dienen der **Überbrückung zwischen konzeptionellen und physischen Datenmodellen**. Sie werden oft auch als Implementierungsmodelle bezeichnet, das logische Datenmodell steht dem Entwickler zur Definition eines Datenbankschemas zur Verfügung und wird weitgehend automatisch vom DBMS in ein physisches Datenbankschema übersetzt.

Zur Formulierung des logischen Schemas stehen je nach zugrunde liegendem DBS folgende Möglichkeiten zur Wahl:

- Hierarchisches Modell
- Netzwerkmodell
- Relationales Modell
- Objektorientiertes Modell

- Anforderungen: Z.B. Banker erklärt, welche Daten (Kunden, etc.) benötigt werden,
- Konzeptioneller Entwurf/Konzeptionelles Schema: z.B. ER-Modell
- Logischer Entwurf: Ist konzeptioneller Entwurf überhaupt umsetzbar?
- Physischer Entwurf: Wo wirklich die Datenbank erstellt wird.

Ziel: Kunde soll Datenbank verstehen, nur Kunde weiß, wie das Unternehmen aufgebaut ist und welche Daten wie verarbeitet werden. Die Fehlerquote soll minimiert werden.

2.3) ENTITY-RELATIONSHIP-MODELL

Das ER-Modell wurde im Jahr 1976 von Chen eingeführt. In der Sicht des ER-Modells besteht die Welt aus Entities (Objekttypen) und Relationships (Beziehungen), zwischen den Objekten. Das ER-Modell wurde von Teorey im Jahr 1986 erweitert zum Extended ER-Modell (EER).

2.3.1) Grundlegende Objekte

Objekte bzw. Entity-Instanzen repräsentieren unterscheidbare Objekte des Problembereichs. Objekte mit den selben Eigenschaften werden zu Entities zusammen gefasst.

- **Entity**

- Jede Entität muss eindeutig sein, z.B. SVNr von Person, FIN von Auto, …
- **Weak Entity**

- Im Unterschied zur Entity ist die Weak Entity nicht eindeutig. Eine Weak Entity ist alleine nicht „überlebensfähig“. Z.B.: Die Speise braucht eine Zuordnung zum Restaurant.

Beziehungen zwischen Objekten haben keine physische oder konzeptionelle Existenz sondern ergeben sich aufgrund der vorhandenen Objekte.

Attribute sind Eigenschaften der Entities. Es wird unterschieden zwischen:

- **identifizierende Attribute = (Schlüssel, Key)** ⇒ sind jene Attribute, die die Objekte einer Entity eindeutig kennzeichnen.

- **beschreibende Attribute** ⇒ sind jene Attribute, die ein Objekt nicht eindeutig kennzeichnen.

Schlüssel (Keys) können aus **einem oder mehreren Attributen** bestehen. Jede Entität kann mehrere verschiedene Schlüssel haben, wir stellen im ER-Modell aber immer **einen ausgezeichneten Schlüssel als Primärschlüssel (=Primary Key)** dar. Entitäten deren Objekte, nur mit Hilfe von anderen Entitäten identifiziert werden können, nennt man Weak Entities. Sie haben

keine Attribute, die alleine den Schlüssel bilden können, obwohl sie dazu beisteuern können.

Weiters können Attribute mehrwertig sein oder komplexe Werte annehmen.

- **mehrwertige Attribute**

- **komplexe Attribute**

2.3.1.1) Komplexität von Beziehungen

Die Komplexität einer Beziehung zwischen Entities beschreibt mit vielen Instanzen, der jeweils anderen Entität jede Instanz einer Entität in Beziehung stehen kann. Bei einer Beziehung zwischen 2 Entitäten A und B gibt es folgende Möglichkeiten:

- **1:1** ⇒ Jedes Objekt von A gehört genau zu einem Objekt von B und umgekehrt
- **1:n** ⇒ Jedes Objekt von A gehört zu einem oder mehreren Objekten von B, aber jedes Objekt von B gehört genau zu einem Objekt von A.
- **n:m** ⇒ Jedes Objekt von A gehört zu einem oder mehreren Objekten von B und umgekehrt.

Die Darstellung einer Komplexität von Beziehung nach Chen geschieht durch Anbringung von Ziffern an die jeweiligen Entitäten. Dabei wird nur zwischen den Werten 1 und viele (n,m) unterschieden. Die genaue Anzahl der zugeordneten Objekte, d.h. die Kardinalität wird selten verwendet, da diese sich bei jeder Instanz einer Beziehung unterscheiden kann. Teorey verwendet nicht mehr die Werte neben den Entitäten sondern färbt die Hälften der Beziehungen ein, die zu den n-Entitäten gedreht sind.

Konzept	Darstellung nach Chen	Darstellung nach Teorey
1:1		
1:n		
n:m		

2.3.1.2) Existenz einer Entität in einer Beziehung

Die Komplexität einer Beziehung gibt an mit wie vielen Instanzen der anderen Entität in Beziehung stehen kann. Dabei haben wir immer von einer oder von vielen Instanzen gesprochen. Das Konzept der Existenz einer Entität gibt uns die Möglichkeit auszudrücken, ob es immer mindestens eine Instanz geben muss oder eine Instanz geben kann.

Konzept	Darstellung nach Chen	Darstellung nach Teorey
zwingend		
optional		
unbekannt		

Bsp.:
zwingend ⇒ Ein Restaurant hat hat zumindest einen oder mehrere Mitarbeiter und jeder Mitarbeiter ist genau einem Restaurant zugeordnet.

optional ⇒ Ein Restaurant hat zwingend einen Geschäftsführer. Manche Mitarbeiter sind Geschäftsführer.

2.3.1.3) Grad einer Beziehung

Der **Grad einer Beziehung** ist die **Anzahl der Entitäten**, die in der **Beziehung miteinander verbunden** sind. **Unäre und binäre Beziehungen** kommen in der realen Welt **am häufigsten** vor.

Unäre Beziehungen (Beziehung einer Entität mit sich selbst) heißen auch binär rekursive Beziehungen.

Ternäre Beziehungen (Beziehungen zwischen 3 Entitäten) werden benötigt wenn binäre Beziehungen einen Sachverhalt nicht ausreichend beschreiben. Kann jedoch eine ternäre Beziehung mit 2 oder 3 binären Beziehungen ausgedrückt werden, so ist diese Darstellung vorzuziehen.

Grad der Beziehung	Darstellung nach Chen	Darstellung nach Teorey
unär		
binär		
ternär		

Bsp.:
Für eine konkrete Pizzabestellung wird für die Lieferfirma ein Auto und ein Mitarbeiter abgestellt. Daher folgende ternäre Beziehung:

1. Ein Mitarbeiter verwendet für eine Bestellung ein Auto
2. Ein Auto wird bei einer Bestellung von einem Mitarbeiter gefahren.
3. Ein Mitarbeiter kann mit einem Auto mehrere Bestellungen erledigen

2.3.1.4) Attribute einer Beziehung

Attribute können wie wir bisher gesehen haben zu Entitäten hinzugefügt werden. Es gibt aber auch die Möglichkeit Attribute zu Beziehungen hinzuzufügen.
So könnte man die Attribute Anzahl und Datum zu Beziehung zwischen Kunde und Speise schreiben. Dadurch trägt man den Umstand Rechnung, dass ein Kunde eine Speise mehrmals bestellen kann. Würde man Attribut Anzahl zum Kunden hinzufügen würde man ein mehrwertiges Attribut erhalten und zudem die Information verlieren, welche Speise der Kunde zu welchem Datum bestellt hat.

Attribute werden üblicherweise nur zu n:m Beziehungen hinzugefügt, da im Falle von 1:1 oder 1:n zumindest auf einer der beiden Seiten der Beziehung ein einziges Objekt steht und damit die Mehrdeutigkeit kein Problem darstellt.

Wenn Attribute zu n:m Beziehungen geschrieben werden, so muss man überlegen, ob man nicht an Stellt von n:m Beziehungen eine Weak-Entität modelliert.
Denn jede n:m Beziehung entspricht einer Weak-Entität die durch eine 1:n Beziehung mit den beiden anderen Entitäten verbunden ist.

2.3.2) Erweiterte ER-Konstrukte: Die Generalisierung

Bei der Modellierung der Entities Mitarbeiter und Kunden wird man feststellen, dass sie viele gemeinsame Attribute besitzen. Diese können zu einer Enitity Person verallgemeinert werden. Mitarbeiter und Kunden werden jeweils mit einer 1:1 Beziehung zu Person verbunden. Die Attribute der Person werden entlang der Hierarchie vererbt und spezielle Attribute wie die SVNR der Mitarbeiter nur bei den speziellen Entities gespeichert.

Die Generalisierung gibt an, dass mehrere Entities (Subtyp-Entity) mit bestehenden gemeinsame Attributen zu einer Entity auf einer höheren Ebene (Supertyp-Entity) generalisiert werden können.

Die Disjunktheit der Generalisierbarbeit beschreibt ob die einzelnen Subtyp-Entities **disjunkt** oder **überlappend** sind.

- **disjunkte Spezialisierung**: Entität kann zu maximal einem Untertyp gehören (Mitarbeiter ist entweder Angestellter oder Manager)

- **überlappende Spezialisierung**: Entität kann zu einem oder mehreren Untertypen gehören (Person kann ein Mitarbeiter oder/und eine Kunde sein)

2.3.1) Übungen

2.3.1.1) Universität

Ein Institut hat einen eindeutige Nummer, einen Namen und eine Adresse. Eine LektorIn identifiziert sich anhand seiner Sozialversicherungsnummer, und hat einen Namen. Er/Sie ist genau einem Institut zugeordnet, ein Institut kann keine oder mehrere Lektoren haben. Weiters gibt es Lehrveranstaltungen, wobei diese eine eindeutige Nummer haben und einen Titel. Eine LektorIn kann mehrere Lehrveranstaltungen leiten, eine Lehrveranstaltung kann von mehreren Lektoren geleitet werden, hat aber mindestens einen Leiter.

2.3.1.2) Familienstammbaum

Jemand will seine Vorfahren (und sonstige Verwandte) in einer Datenbank speichern. Jede Person wird durch ihren Vornamen, Geburtstag und Geburtsort identifziert. Des weiteren wird ihr Zuname gespeichert. Bei verstorbenen Personen wird auch der Sterbetag und -ort in der Datenbank verwaltet. Bei den Personen wird zwischen Männern und Frauen unterschieden. Für jede Person wird auch der Vater und die Mutter gespeichert, falls diese bekannt sind.
Personen können andere Personen adoptieren. Dabei wird der Tag der Adoption gespeichert.
Des weiteren sollen alle Ehen mit dem Tag der Hochzeit und, falls die Ehe schief gehen sollte, dem Tag der Scheidung vermerkt werden. Paare können nur einmal heiraten.

2.3.1.3) Friedhof

Für einen Friedhof wird zur Arbeitserleichterung der Verwaltung eine Datenbank erstellt. Unterstreichen Sie je Relation einen Schlüssel. Verwenden Sie nur die vorgegebenen Attributnamen. (Diese sind nur bei ihrer jeweils ersten Erwähnung angeführt.)

Auf dem Friedhof gibt es viele Gräber. Jedes Grab hat eine eindeutige Nummer (GNR), eine Lagebeschreibung (LAGE), einen Besitzer (BESITZER) und eine maximale Sarganzahl (MAXSARG). In einem Grab können sich mehrere Verstorbene befinden. Särge haben eine eindeutige Bestellnummer (BNR) und einen Hersteller (HERSTELLER).
Weiters gibt es Verstorbene, von denen die eindeutige Totenscheinnummer (TNR), das Sterbedatum (SDATUM), das Geburtsdatum (GDATUM), der Vorname (VORNAME) und der Nachname (NACHNAME) bekannt sind. Es ist auch bekannt, in welchem Sarg der Verstorbene liegt. Man kann die Position (POSITION) von Verstorbenen relativ zu einem anderen Verstorbenen im selben Grab angeben (z. B. kann ein Verstorbener rechts, links, unter . . . einem anderen Verstorbenen liegen).

Es gibt 3 Friedhofsgärtner mit eindeutiger Sozialversicherungsnummer (SVNR), und Vor- und Nachnamen (VORNAME, NACHNAME), die für die Betreuung der Gräber zuständig sind. Es ist bekannt, welcher von den Gärtnern für ein bestimmtes Grab verantwortlich ist.

Man kann bei der Friedhofsgärtnerei verschiedene Dienstleistungen bestellen (z. B. Pflanzen setzen, Kerzen zu Allerheiligen, . . .). Jede Dienstleistung wird durch eine Nummer (DNR), eine Beschreibung

(BESCHREIBUNG) und einen Preis (PREIS) beschrieben. Bestellungen beziehen sich immer auf Gräber und Dienstleistungen, es wird das Datum (DATUM) angegeben und die Person (PERSON), die die anfallende Rechnung bezahlt. Für die Statistik wird mitprotokolliert, wann (DATUM) welcher Gärtner welche Dienstleistung bei einem Grab durchgeführt hat und wie viele Stunden (STUNDEN) er dafür gebraucht hat.

2.3.1.4) Politik

Um den Überblick über das Kommen und Gehen der politischen Akteure zu behalten bittet Sie ein Freund um eine Datenbank. Zeichnen Sie aufgrund der vorliegenden Informationen ein ER-Diagramm. Achtung!! Beachten Sie, dass der unten beschriebene Sachverhalt stark vereinfacht ist und nicht notwendigerweise mit der Realitat ubereinstimmt. Modellieren Sie bitte auf jeden Fall den angegebenen Sachverhalt!

Zu jeder Person wird ihr Vorname (VNAME), Nachname (NNAME) sowie eine besondere Eigenschaft (EIGENSCHAFT) gespeichert. Es kann keine zwei Personen mit dem selben Namen (gleicher Vorname und gleicher Nachname) geben.
Parteien besitzen eine eindeutige Farbe (FARBE), und darüber hinaus ein (nicht notwendiger Weise eindeutiges) Kürzel (KRZL).
Jede Legislaturperiode ist durch ihren Beginn (VON) gemeinsam mit ihrem Ende (BIS) identifizierbar.
Jeder Aufgabenbereich der Regierung hat eine eindeutige Bezeichnung (BEZ). Außerdem gibt es eine Beschreibung (BESCHREIBUNG) zu jedem Aufgabenbereich.
Es wird vermerkt welche Person in welcher Legislaturperiode welche Aufgaben übernimmt. Außerdem wird gespeichert welcher Aufgabenbereich in welcher Legislaturperiode in welchem Ministerium angesiedelt war. Ministerien haben einen eindeutigen Namen (NAME) und ein Budget für Werbung (WBUDGET). In jedem Ministerium muss mindestens ein Aufgabenbereich angesiedelt sein (in irgendeiner Legislaturperiode). Außerdem gibt es in jeder Legislaturperiode mindestens drei Aufgabenbereiche.
Es soll außerdem vermerkt werden wie viele Stimmen jede Partei in den verschiedenen Legislaturperioden hatte.
Jeder Parteieintritt erhält eine innerhalb der entsprechenden Partei eindeutige Nummer (NR), es wird das Datum des Eintritts (DATUM) gespeichert, sowie welche Person eingetreten ist (bei jedem Parteieintritt tritt genau eine Person einer Partei bei).

2.3.1.5) Kreditkartenfirma

Als Teil ihrer Strategie zur Verbesserung des Dokumentations-Workflows bittet die Kreditkartenfirma „Schuldenfroh“ um ein Redesign ihrer Datenbank.
Nachdem Sie sich vergewissert haben, dass die Beschreibung wenigstens ein paar Anglizismen enthält, machen Sie sich an die Arbeit.

Jede Kreditkarte ist eindeutig identifiziert durch ihre Kartennummer (KaNR).
Zusätzlich muss das Ablaufdatum (ABLAUFDATUM) und der Sicherheitscode (CVC) vermerkt sein.
Ersetzt eine Karte eine ältere Karte, so wird gespeichert welche Karte ersetzt wird (jede Karte kann maximal eine andere Karte ersetzen und durch maximal eine andere Karte ersetzt werden), und ab wann die neue Karte gültig ist (AB). KundInnen haben eine eindeutige Kundennummer (KuNR).
Zusätzlich ist ihr Vor- und Nachname (VNAME, NNAME) und ihr Geburtsdatum (GEBDAT) bekannt,

wobei die Kombination aus Vorname, Nachname und Geburtsdatum ebenfalls eindeutig ist.
Außerdem muss für jede Kundin/jeden Kunden eine Kontonummer (IBAN) vorhanden sein.
Bei Kreditkarten wird unterschieden auf welche Kundin/welchen Kunden die Kreditkartet lautet auf der einen, und welche KundInnen für eine Karte zeichnungsberechtigt sind auf der anderen Seite. Jede Kreditkarte lautet auf genau eine Kundin/einen Kunden. Jede Kundin/Jeder Kunde kann jedoch für beliebig viele Karten zeichnungsberechtigt sein, wobei für jede Karte die Anzahl der Zeichnungsberechtigten auf 5 begrenzt ist.
Partner der Kreditkartenfirma sind sowohl durch ihren Markennamen (MNAME) eindeutig identifiziert, als auch durch ihre Partner-ID (PID). Bei Partnern wird zwischen Vertriebspartnern und Bezahlpartnern unterschieden. Vertriebspartner vergeben Kreditkarten, wobei jede Karte von genau einem Vertriebspartner vergeben werden muss. Für Vertriebspartner wird neben dem Preis (PREIS) einer Karte auch der Prozentsatz (ANTEIL) gespeichert, den der Vertriebspartner von jedem mit der Karte getätigten Umsatz erhält. Bezahlpartner sind Unternehmen welche Kreditkarten von „Schuldenfroh“ akzeptieren. Für jeden Bezahlpartner wird der Anteil des Umsatzes gespeichert der als Gebühr abgeführt werden muss (GEBUEHR). Manche Vertriebspartner gewähren einen Rabatt bei Zahlungen mit bestimmten Bezahlpartnern.
Es soll vermerkt werden, welcher Vertriebspartner bei welchem Bezahlpartner welchen Rabatt (PROZENT) gewährt.
Jeder Bezahlpartner erhält mindestens einen Account. Ein Account ist eindeutig identifiziert durch den Bezahlpartner sowie durch den Benutzernamen (BENUTZER). Zusätzlich soll für jeden Account das Passwort (PW), sowie sämtliche Loginversuche gespeichert werden. Ein Login(versuch) wird eindeutig identifiziert durch den Account, sowie durch das Datum (DATUM) des Logins, gemeinsam mit einer id (ID). Darüber hinaus wird die IP-Adresse des Logins gespeichert (IP), und ob der Login erfolgreich war (SUCCESS).
MitarbeiterInnen von Bezahlpartnern werden durch ihre Sozialversicherungsnummer (SNVR) eindeutig identifiziert. Darüber hinaus wird ihr Vorname (VNAME), Nachname (NNAME), ihr Geburtsdatum (GEBDAT), sowie ihr Gehalt (GEHALT) gespeichert. Es wird nicht gespeichert, welche MitarbeiterInnen bei welchem Partner arbeiten, sondern nur, welche MitarbeiterInnen Zugriff auf welchen Account haben, wobei es für jeden Account mindestens eineN MitarbeiterIn mit Zugriff geben muss.
Eine Transaktion ist eindeutig identifiziert durch die Kreditkarte auf welche sie gebucht wurde, den Account mit dem die Buchung durchgeführt wurde, sowie durch eine Transaktionsid (TID). Zu jeder Transaktion wird außerdem der Betrag (TOTAL) sowie das Datum (DATUM) gespeichert.

Jedes Security-Level ist eindeutig bezeichnet durch seinen Namen (NAME) in Kombination mit seiner Nummer (NR). Für jede Transaktion wird genau ein Security-Level verwendet.
Die Kreditkartenfirma beschäftigt Security-Agents. Zu jedem Security-Agent wird ein Name (NAME), sein Trust-Level (TLEVEL), sowie eine eindeutige Agent-ID (AID) verwaltet.
Zu Transaktionen, bei denen der Verdacht auf einen Betrugsfall besteht wird zusätzlich noch eine Bemerkung (BEM), die Art des Betrugs (TYP), das Datum an dem der Betrugsverdacht erkannt wurde (DATUM) gespeichert, sowie welcher Security-Agent den Verdacht gemeldet hat. Sobald einE KundeIn auf einen Verdachtsfall reagiert wird das Datum (DATUM) der Reaktion gespeichert, sowie ob es sich in der Tat um einen Betrug handelt oder nicht (OK).
Es soll weiter gespeichert werden, welcher Security-Agent welcheN MitarbeiterIn zu welchem Security-Level zugelassen hat, und wann das passiert ist (AM). Es gilt zu beachten, dass jedeR MitarbeiterIn zu mindestens einem Security-Level zugelassen sein muss.

2.4 Relationenmodell

Das Relationenmodell wurde 1970 von Codd entwickelt. Es **repräsentiert** die **Daten einer Datenbank als eine Menge von Relationen**. Eine **Relation** können wir uns dabei **als Tabelle vorstellen**, der **Zeilen Objekte oder Beziehungen zwischen Objekten beschreiben**. Der **Tabellenkopf** heißt **Relationenschema**, die einzelnen **Spalten** werden als **Attribute** bezeichnet, die einzelnen **Zeilen als Tupel**.

Restaurant

ID	Name	PLZ
R1	Mc	3300
R2	Hw	3300
R3	Kb	3300

„Restaurant“ ist ein Relationenschema. ID, Name, PLZ ist Attribut, R1/Mc/3300 etc. sind Tupel

2.4.1 Formalisierung

Ein **Relationenschema R** ist eine **endliche Menge von Attributnamen** $\{A_1, A_2, ... A_n\}$. Zu jedem Attributnamen A_1 gibt es eine Menge D_i, $1 \leq i \leq n$, den **Wertebereich (=domain)** von A_i, der auch mit $Dom(A_i)$ bezeichnet wird.

Eine **Relation r(R)** auf einem **Relationenschema R** ist eine endliche Menge von Abbildungen $\{t_1,....,t_m\}$ von R nach D. Die Abbildungen werden **Tupel** genannt.

Bsp.: Relationenschemata für RESTAURANT und SPEISE

```
Restaurant = {rnr,name,adresse,haube,typ}
Speise     = {name,preis,rnr}
```

rnr	name	adresse	haube	typ
1	…	…	…	…
2	…	…	…	…
3	„Green Cottage“	„Kettenbrückengasse 3, 1050 Wien“	2	„chinesisch“
4	…	…	…	…

Bsp.: Domänen für das Relationenschemata RESTAURANT:

```
Dom(rnr)   = Menge aller Integer
Dom(name)  = Menge aller Namen
Dom(haube) = {k|0<=k<=4}
```

Bsp.: Die Tabelle Restaurant hat mehrere Tupel. Eines von ihnen ist t3 mit:

```
t3            = 3, "Green Cottage", "Kettenbrückengasse 3, 1050 Wien", 2,
```

```
"chinesisch"
t3(rnr)       = 3
t3(name)      = "Green Cottage"
t3(haube,typ) = 2, "chinesisch"
```

Ein Schlüssel einer Relation r(R) ist eine Teilmenge K von R, sodass für zwei verschiedene Tupel t_1 und t_2 aus r(R) immer $t_1(K) != t_2(K)$ gilt und keine echte Teilmenge K' von K diese Eigenschaft hat.

Im Allgemeinen wird dabei ein Schlüssel als Primärschlüssel ausgezeichnet und in den Relationenschemata durch Unterstreichen der Schlüsselattribute gekennzeichnet.
Bsp.:

```
Restaurant = {__rnr__, name, adresse, haube, typ}
Speise     = {__name__, preis, rnr}
```

2.4.2 Operationen auf Relationen

Um Operationen auf Relationen durchführen zu können, wurde die relationale Algebra eingeführt.

2.4.2.1 Mengenoperationen

Zu den Mengenoperationen gehören:

- Durchschnitt („∩“)
- Vereinigung („∪“)
- Differenz („-“)

von Relationen, die über der gleichen Attributmenge mit derselben Anordnung (identische Reihenfolge der Attribute) definiert sind:

Bsp.: Es sind 2 Relationen r und s gegeben:

r

A	B	C
a1	b1	c1
a1	b2	c1
a2	b1	c2

s

A	B	C
a1	b2	c1
a2	b2	c1
a2	b2	c2

Gesucht sind:

- Durchschnitt (r ∩ s)
- Vereinigung (r ∪ s)
- Differenz (r - s)
- Differenz (s - r)

Lösung:

- Durchschnitt (r ∩ s)

A	**B**	**C**
a1	b2	c1

- Vereinigung (r ∪ s)

A	**B**	**C**
a1	b1	c1
a1	b2	c1
a2	b1	c2
a2	b2	c1
a2	b2	c2

- Differenz (r - s)

A	**B**	**C**
a1	b1	c1
a2	b1	c2

- Differenz (s - r)

A	**B**	**C**
a2	b2	c1
a2	b2	c2

2.4.2.2 Die Selektion ("σ")

Bei der Selektion werden Zeilen ausgewählt, die einem bestimmten Kriterium entsprechen:

$ σ_{A=a}(r) = \{ t∈r|t(A)=a \} $

Beispiele:
a) $ σ_{A=a1}(r) = ? $

A	B	C
a1	b1	c1
a1	b2	c1

b) $ σ_{A=a1}(s) = ? $

A	B	C
a1	b2	c1

Noch allgemeiner wird die Selektion, wenn wir erlauben, wohldefinierte Operatoren auf den Attributwerten auszuführen, z.B.: arithmetische Operationen auf Zahlenwerte und logische Verknüpfungen von Attributen durch „und“ („∧“), „oder“ („∨“) und Negationen („¬“).

c) $ σ_{(B=b1)∧¬(A=a1)}(r)=? $
1. Schritt:
$ σ_{(B=b1)}(r)=? $

A	B	C
a1	b1	c1
a2	b1	c2

2. Schritt: $ σ_{(B=b1)∧¬(A=a1)}(r)=? $

A	B	C
a2	b1	c2

d) $ σ_{(haube > 1) ∧ ¬(typ="österreichisch" ∨ typ="international")}(Restaurant)=? $

Bsp: Relation Restaurant

rnr	name	adresse	haube	typ
1	Schnitzelhaus	Amstetten	1	österreichisch
2	Brauhof	Amstetten	3	international
3	Pizza Hollywood	Amstetten	2	italienisch

Lösung ⇒ Folgender Tupel:

3	Pizza Hollywood	Amstetten	2	italienisch

2.4.2.3 Die Projektion ("π")

Bei der Projektion werden gewisse Spalten einer Tabelle ausgewählt. Man projiziert nach einer Teilmenge der Attribute:

$π_X(r) = \{t(X)|t∈r\} für X ⊆ R $

Beispiele:

e) $ π_{A,B}(r)=? $

f) $ π_{name,adresse,haube}(Restaurant)= ?? $

name	**adresse**	**haube**
Schnitzelhaus	Amstetten	1
Brauhof	Amstetten	3
Pizza Hollywood	Amstetten	2

2.4.2.4 Der Verbund

2.4.2.4.1 Der natürliche Verbund

Der Verbundoperator verknüpft zwei Relationen über ihre gemeinsamen Attribute:

$ r ⋈ s = \{|\{R,S\}| ∃ t_r ∈ r \text{ und } ∃ t_s ∈ s:t_r =t(R) \text{ und } t_s=t(S) \} $

Der Verbundoperator ist kommutativ.

Beispiel:
Relation r

A	**B**
a1	b1
a2	b1
a3	b2

Relation s

B	**C**
b2	c1
b2	c2
b1	c3
b3	c4

Relation r ⋈ s

A	B	C
a1	b1	c3
a2	b1	c3
a3	b2	c1
a3	b2	c2

Aufgabe:

```
Restaurant  Speise
```

Lösung:

rnr	name	adresse	haube	typ	preis

Erklärung:
Der natürliche Verbund verknüpft beide Relationen über die gemeinsamen Attribute und gibt daher jene Tupel aus, bei denen sowohl der Name **name** des Restaurants und der Name **name** der Speise als auch die Restaurantnummer **rnr** des Restaurants und jene der Speise gleich sind. In unserem Fall ist das die leere Menge!

Projektionseigenschaften des Verbundoperators

Seien R und S zwei Relationenschema, $ q=r⋈s $ und $ r'=π_R(q) $, dann gilt $ r'⊆r $. Joinen wir also eine Relation r mit einer anderen und projizieren dann nach den ursprünglichen Attributen von r, so können unter Umständen Tupel verloren gehen.

Beispiel:

Relation r

A	B
a	b
a	b'

Relation s

B	C
b	c

Relation r⋈s=q

A	B	C
a	b	c

Relation $ π_{AB}(q)=r' $

A	B
a	b

2.4.2.4.2 Das Kartesische Produkt

Falls $R ∩ S = \{\}$,die beiden Relationenschemata also kein gemeinsames Produkt haben, so liefert die Verknüpfung r ⋈ s das Kartesische Produkt, geschrieben als r x s.

Beispiel:
Relation r

A	B
a1	b1
a2	b1

Relation s

C	D
c1	d1
c2	d1
c2	d2

Relation r x s = r ⋈ s

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d1
a1	b1	c2	d2
a2	b1	c1	d1
a2	b1	c2	d1
a2	b1	c2	d2

Wie wir oben gesehen haben, ist das Kartesische Produkt nur für den Fall definiert, dass $R ∩ S=\{\}$. Möchte man das Kartesische Produkt von Relationen bilden, die gemeinsame Attribute haben, so müssen diese in einer der Relationen umbenannt werden. Ist etwa R = {A,B,C} und S = {A,B,D}, so benennen wir die Attribute von S um, so dass S={A',B',D} oder kennzeichnen sie durch Voranstellen des Relationennamens, also S={S.A,S.B,D}.

2.4.2.4.3 Weitere Verbundarten

Weitere Verbundarten sind:

- der Gleichverbund (equi-join)
- der Theta-Verbund (theta-join)
- der Semi-Verbund (semi-join)

- der Äußere Verbund (outer-join)

2.4.2.5 Division

Möchte man die Relation r durch s dividieren, so muss die Attributmenge von s eine Teilmenge der Attributmenge von r sein. Das Ergebnis hat die Differenz der Attributmengen als Attribute und wählt jene Tupel aus r aus, die eingeschränkt auf die Differenz der Attribute R-S für alle Tupel aus s denselben Wert haben.

Beispiel:
Relation R

A	**B**	**C**	**D**
a	b	c	d
a	b	e	f
b	c	e	f
e	d	c	d
a	b	d	e
e	d	e	f
a	d	e	f

Relation S

C	**D**
c	d
e	f

Relation R % S

A	**B**
a	b
e	d

Sowohl bei |a|b| als auch bei |e|d| kommen beide Tupel der Relation S, nämlich |c|d| und |e|f| vor.

Übungen zum Relationenmodell

Übung 1

Relation Restaurant			
rnr	**RName**	**Adr**	**Haube**
1	McDonalds	Amstetten	0
2	McDonalds	Ybbs	0
3	Hollywood	Amstetten	1
4	Casa Venezia	Amstetten	2
5	grill.Bar	Amstetten	2
6	Schinakel	Grein	1

Relation Speise		
rnr	**SName**	**Preis**
1	Hamburger	1
2	Hamburger	1
2	Cheeseburger	1,30
3	Hawaii	8,6
4	Hawaii	9
5	Salat	5
1	Salat	1,80

Abfragen in relationaler Algebra:

a) alle Restaurants, die in Amstetten zu finden sind

b) alle Restaurants von Amstetten mit mindestens 2 Hauben

c) Namen aller Restaurants

d) Namen aller Restaurants die in Amstetten zu finden sind

e) alle Restaurants, die auch Speisen anbieten

f) Namen der Restaurants, die Speisen anbieten

g) Namen aller Restaurants, die einen Salat anbieten

h) Namen und Preise aller Speisen samt Name des Restaurants, von Restaurants, die einen Salat anbieten

Hier nochmals die Relationen:

Lösung Übung 1 a)

$ σ_{(Adr="Amstetten")}(Restaurant) $

Lösung Übung 1 b)

$ σ_{(Adr="Amstetten")∧(haube >= 2)}(Restaurant) $

Lösung Übung 1 c)

$ π_{RName}(Restaurant) $

Lösung Übung 1 d)

$ π_{RName}(σ_{(Adr="Amstetten")}(Restaurant)) $

Lösung Übung 1 e)

$ Restaurant ⋈ Speise $

Lösung Übung 1 f)

$ π_{RName}(Restaurant ⋈ Speise) $

Lösung Übung 1 g)

$ π_{RName}(σ_{SName="Salat"}(Restaurant ⋈ Speise)) $

Lösung Übung 1 h)

$ π_{RName, SName, Preis}(σ_{SName="Salat"}(Restaurant ⋈ Speise)) $

Übung 2

Gegeben sind einige Relationen und Abfragen. Formulieren Sie die Abfragen mittels Relationaler Algebra und berechnen Sie auch das Ergebnis der Abfragen.

Die Relation **Rechner** beschreibt die Rechner eines Institutes. **RNr** ist eine eindeutige Bezeichnung für den Rechner, **StudAss** ist der eindeutige Name des Studienassistenten, der den Rechner (und die

darauf installierten Programme) wartet, **Speicher** gibt die Größe der Festplatte des Rechners an und **Leist** ist ein Maß für die Leistungsfähigkeit des Rechners, wobei 1 die schlechteste und 5 die beste Leistung ist.

Relation Rechner			
RNr	**StudAss**	**Leist**	**Speicher**
R1	Huber	1	100
R2	Brunner	3	80
R3	Brunner	3	400
R4	Vogt	2	120
R5	Huber	2	500

In **Programm** sind die Programme, die das Institut besitzt gespeichert. **PNr** ist eine eindeutige Nummer des Programms, **PName** ist sein Name, **Bereich** gibt an, um was für eine Art von Programm es sich dabei handelt und **MinLeist** bezeichnet die Leistungsfähigkeit, die ein Rechner mindestens besitzen muss, damit das Programm auf ihm laufen kann. Hat ein Programm also die **MinLeist** 4, so kann man ihn nur in einem Rechner mit **Leist** 4 oder 5 einsetzen, nicht aber einem mit **Leist** 1,2 oder 3.

Relation Programm			
PNr	**PName**	**MinLeist**	**Bereich**
P1	DrawIt	1	Grafik
P2	AskIt	3	Datenbank
P3	WriteIt	1	Text
P4	ConnectIt	2	Internet
P5	PaintIt	2	Grafik
P6	StoreIt	3	Datenbank

Die Relation **Assistent** beschreibt die Assistenten, die an dem Institut arbeiten. (Diese sind von den Studienassistenten verschieden) Dabei ist **ANr** eine eindeutige Nummer für den Assistenten, **AName** ist dessen Name, **StudAss** ist der eindeutige Name des Studienassistenten, der den Assistenten bei seiner Tätigkeit unterstützt. **Gehalt** bezeichnet die Gehaltsstufe des Assistenten.

Relation Assistent			
ANr	**AName**	**StudAss**	**Gehalt**
A1	Novak	Brunner	3
A2	Dvorak	Vogt	1
A3	Husak	Vogt	1
A4	Pfeiffer	Brunner	2

In der Relation **Installation** ist verzeichnet, welche Programme auf welchen Rechnern installiert sind. **RNr** und **PNr** geben den entsprechenden Rechner und das Programm an, **Platz** gibt die Größe der Installation auf der Festplatte an und **Code** ist ein Code, den nur die Studienassistenten verstehen.

Relation Installation			
RNr	**_PNr_**	**Platz**	**Code**
R1	P1	500	X
R1	P3	300	z
R2	P6	300	X
R2	P2	200	pp
R3	P1	400	c
R3	P2	100	tt
R3	P3	500	c
R3	P4	200	pp
R3	P5	200	z
R3	P6	100	t
R4	P5	1000	T
R5	P1	200	p
R5	P5	100	ccc

In der Relation **Benutzung** wird vermerkt, wie lange die Assistenten die verschiedenen Programme benutzen. **ANr** verweist auf den Assistenten, **PNr** auf das Programm und **Stund** gibt an wieviele Stunden am Tag das Programm vom Assistenten pro Tag höchstens benötigt wird.

Relation Benutzung		
ANr	**_PNr_**	**Stund**
A1	P1	5
A1	P2	3
A2	P1	6
A2	P4	2
A2	P5	5
A3	P1	7
A3	P3	3
A4	P1	1
A4	P4	4

Abfragen:

a) Geben Sie den Namen der Assistenten aus, die ein Programm möglicherweise länger als 5 Stunden benutzen.

b) Wie heißen die Programme, die von Huber gewartet werden?

c) Wie heißen die Programme, die von allen Assistenten benutzt werden?

d) Auf welchem Rechner sind dieselben Programme installiert, wie auf dem Rechner R1?

e) Von welchen Studienassistenten wird das Programm WriteIt nicht gewartet?

f) Welche Paare von Assistenten werden vom gleichem Studienassistenten betreut? (Dabei soll jedes Paar nur einmal ausgegeben werden)

g) Welche Studienassistenten betreuen sowohl Assistenten, als auch Rechner?

h) Welche Paare von Rechner haben dieselbe RLeistung?

i) Welche Programme (gesucht sind die Namen) sind auf allen Rechnern installiert?

j) Welche Programme laufen auf einem Rechner, der genaue die minimale Leistungsfähigkeit für das Programm besitzt? (Gesucht sind die Paare aus Rechnernummer und Programmnummer)

Hier noch einmal alle Relationen auf einen Blick:

Relation Rechner

RNr	StudAss	Leist	Speicher
R1	Huber	1	100
R2	Brunner	3	80
R3	Brunner	3	400
R4	Vogt	2	120
R5	Huber	2	500

Relation Programm

PNr	PName	MinLeist	Bereich
P1	DrawIt	1	Grafik
P2	AskIt	3	Datenbank
P3	WriteIt	1	Text
P4	ConnectIt	2	Internet
P5	PaintIt	2	Grafik
P6	StoreIt	3	Datenbank

Relation Assistent

ANr	AName	StudAss	Gehalt
A1	Novak	Brunner	3
A2	Dvorak	Vogt	1
A3	Husak	Vogt	1
A4	Pfeiffer	Brunner	2

Relation Installation

RNr	_PNr_	Platz	Code
R1	P1	500	X
R1	P3	300	z
R2	P6	300	X
R2	P2	200	pp
R3	P1	400	c
R3	P2	100	tt
R3	P3	500	c
R3	P4	200	pp
R3	P5	200	z
R3	P6	100	t
R4	P5	1000	T
R5	P1	200	p
R5	P5	100	ccc

Relation Benutzung

ANr	_PNr_	Stund
A1	P1	5
A1	P2	3
A2	P1	6
A2	P4	2
A2	P5	5
A3	P1	7
A3	P3	3
A4	P1	1
A4	P4	4

2.5 Umsetzung des ER-Modells in ein Relationenmodell

2.5.1 1-zu-1 Beziehung

Bei einer 1-zu-1 Beziehung zwischen Entitäten wird die Beziehung aufgelöst, in dem der Schlüssel der einen Entität zur zweiten Entität hinzukommt. Welche Richtung hier verwendet wird, ist dem Designer überlassen. Wenn allerdings, eine optionale Beziehung besteht, wird der Schlüssel auf der optionalen Seite gespeichert.

2.5.2 1-zu-n Beziehung

Im Falle einer 1-zu-n Beziehung schreiben wir den Schlüssel der 1-Seite in die Relation die der Entität auf der n-Seite entspricht.

2.5.3 n-zu-m Beziehung

Bei einer n-zu-m Beziehung führen wir eine neue Relation ein, die die Schlüssel beider Entitäten als Schlüssel besitzt.

2.5.4 ternäre Beziehung

Bei einer ternären Beziehung (=Beziehung zwischen 3 Entities) muss immer eine neue Relation eingefügt werden, die die Schlüssel aller Entitäten als gemeinsamen Schlüssel besitzt.

Relationen

Schüler (ID, Vorname, Nachname)

Lehrer (Kürzel, Alter)

Klasse (RaumNr, Sitzplätze)

Unterricht (*Schüler.ID*, *Lehrer.Kürzel*, *Klasse.RaumNr*)

2.5.5 Generalisierung

2.5.5.1 nicht disjunkte Entitäten

2.5.5.1 disjunkte Entitäten

2.5.6 Weak-Entities

Bei schwachen Entitäten, bei denen die eigenen Attribute nicht ausreichen um ein Tupel eindeutig zu identifizieren, müssen die Schlüsselattribute der damit verbundenen Entitäten zum Schlüssel hinzugenommen werden.

Übungen zur Umsetzung von ER-Modell => Relationenmodell

Übung 1

Gegeben ist folgendes ER-Diagramm zur Verwaltung von Elektronikbauteilen. Entwickeln Sie daraus die Relationen der Datenbank.

Übung 2

Gegeben ist folgendes ER-Diagramm zur Verwaltung eines Videoverleihs. Entwickeln Sie daraus die Relationen der Datenbank.

Übung 3

Gegeben ist folgendes ER-Diagramm zur Verwaltung eines Videoverleihs. Entwickeln Sie daraus die Relationen der Datenbank.

Das Call Center System erlaubt es bestehenden oder zukünftigen Kunden sich über ein menügesteuertes Interface (Tastenkombinationen am Telefon, Handy, etc.) im Support Center einzuwählen. Das Menü ist dabei in mehrere Bereiche (Produktauskunft, Rechnungsinfo, Technische Probleme, etc.) gegliedert. Eingehende Calls werden entweder von Telefonisten entgegengenommen, welche in allgemeinen Bereichen ausgebildet sind, oder von Technikern bearbeitet, sofern es sich um technische Anfragen handelt. Technische Anfragen werden im Allgemeinen nur für bestehende Kunden bearbeitet. Die mit den Anfragen betrauten Techniker sind für bestimmte Support Levels ausgebildet. Support Levels sind kostenpflichtig, aber Stammkunden bzw. mehr zahlende Kunden können bestimmte Levels gratis in Anspruch nehmen. Technische Probleme können von verschiedenen Technikern auf verschiedene Levels bearbeitet und einer Lösung zugeführt werden.

2.6) Normalformen

Die Kriterien eines guten Datenbankentwurfs sind einerseits möglichst geringe **Redundanz** und andererseits von **Einfügen, Lösch- und Änderungsanomalien** abhängig. Um Redundanzen und Anomalien zu vermeiden, führen wir den Begriff **Normalform** ein, der uns hilft unerwünschte Eigenschaften der Datenbank zu vermeiden.

1. Normalform (1NF)

Ein Relationenschema R ist in 1NF wenn der Wertebereich aller Attribute von R atomar sind. D.h. wenn keine mehrwertigen Attribute vorkommen.

Beispiel - CD Lieder

CD_ID	Name	Titelliste
4711	All That You Can Leave Behind (U2)	Beautiful Day, Walk On, Kite, Wild Honey
4712	Tattou You (Rolling Stones)	Start Me Up, Hang Fire, Slave

- Das Feld **Name** beinhaltet Album und Interpret
- Das Feld **Titelliste** enthält eine Aufzählung aller Titel

Fehler/Problem

- Zur Sortierung nach Interpret muss das Feld **Name** in Album und Interpret aufgeteilt werden
- Den Titel können (mit einfachen Mitteln) nur alle gleichzeitig als Titelliste oder gar nicht dargestellt werden

→ Das Relationenschema ist **nicht** in 1. Normalform!

Lösung 1NF

CD_ID	Album	Interpret	Titel
4711	All That You Can Leave Behind	U2	Beautiful Day
4711	All That You Can Leave Behind	U2	Walk On
4711	All That You Can Leave Behind	U2	Kite
4711	All That You Can Leave Behind	U2	Wild Honey
4712	Tattoo You	Rolling Stones	Start Me Up
4712	Tattoo You	Rolling Stones	Hang Fire
4712	Tattoo You	Rolling Stones	Slave

2. Normalform (2NF)

Ein Relationenschema R ist in zweiter Normalform, wenn es in 1NF ist, und jedes Nicht-

Schlüsselfeld vom ganzen Primärschlüssel (der auch aus mehreren Feldern bestehen kann) abhängig ist.
Wichtig ist, dass Datenfelder nicht nur von einem Teilschlüsselfeld, sondern vom gesamten Schlüssel funktional abhängig sind.

Jedes Tupel muss immer vom gesamten Schlüssel abhängen, sonst ist es nicht in 2. Normalform!

Beispiel 1

Gegeben ist folgende Tabelle:

CD_ID	Album	Interpret	Track	Titel
4711	All That You Can Leave Behind	U2	1	Beautiful Day
4711	All That You Can Leave Behind	U2	2	Walk On
4711	All That You Can Leave Behind	U2	3	Kite
4712	Tattoo You	Rolling Stones	1	Start Me Up
4712	Tattoo You	Rolling Stones	2	Hang Fire

Problem

z.B.: Durch ein Update des Albums kommt es zu einer Dateninkonsistenz

CD_ID	Album	Interpret	Track	Titel
4711	All That You Can Leave Behind	U2	1	Beautiful Day
4711	All That You Can Leave Behind	U2	2	Walk On
4711	All That You Can Leave Behind	U2	3	Kite
4712	**Sticky Fingers**	Rolling Stones	1	Start Me Up
4712	Tattoo You	Rolling Stones	2	Hang Fire

Lösung 2NF

CD_ID	Album	Interpret
4711	All That You Can Leave Behind	U2
4712	Tattoo You	Rolling Stones

CD_ID	Track	Titel
4711	1	Beautiful Day
4711	2	Walk On
4711	3	Kite
4712	1	Start Me Up
4712	2	Hang Fire

Beispiel 2

PLZ	Nachname	Ort	Geburtsdatum
3300	Muster	Amstetten	03.05.2003
3300	Maier	Amstetten	05.06.2002

PLZ	Nachname	Ort	Geburtsdatum
3304	Wimmer	St. Georgen	03.05.2003

Problem: Der Ort hängt nicht vom gesamten Schlüssel (PLZ,Nachname) ab, sondern nur von einem Teil des Primärschlüssels (PLZ)!

3. Normalform (3NF)

Ein Relationenschema R ist genau dann in 3NF, wenn es die 1NF und die 2NF erfüllt, und wenn kein nicht primes Attribut von einem Schlüssel in R transitiv abhängt.

Transitivitätsregel K→X, X→A : K→A

Beispiel 1

Gegeben sei eine Tabelle mit folgenden Feldern:

_CD_ID_	Album	Interpret	Band-Gründungsjahr
4711	All That You Can Leave Behind	U2	1976
4712	Sticky Fingers	Rolling Stones	1962
4713	Tattou You	Rolling Stones	1962

Offensichtlich lässt sich der **Interpret** einer CD aus der **CD_ID** bestimmen. Das **Band-Gründungsjahr** der Band hängt dagegen vom **Interpret** und damit nur **!!transitiv!!** von der **CD_ID** ab.

Problem

Auch hier besteht eine Datenredundanz, wodurch Inkonsistenzen beim Ändern auftreten können.

_CD_ID_	Album	Interpret	Band-Gründungsjahr
4711	All That You Can Leave Behind	U2	1976
4712	Sticky Fingers	Rolling Stones	**1972**
4713	Tattou You	Rolling Stones	1962

Lösung

_CD_ID_	Album	*Interpret*
4711	All That You Can Leave Behind	U2
4712	Sticky Fingers	Rolling Stones
4713	Tattoo You	Rolling Stones

Interpret	Band-Gründungsjahr
U2	1976
Rolling Stones	1962

Beispiel 2

_ID_	PLZ	Nachname	Ort	Geburtsdatum
1	3300	Muster	Amstetten	03.05.2003
2	3300	Maier	Amstetten	05.06.2002
3	3304	Wimmer	St. Georgen	03.05.2003

Problem: Der Ort ist eigentlich von der PLZ abhängig

Lösung

Zwei Tabellen:

PLZ	Ort
3300	Amstetten
3304	St. Georgen

_ID_	PLZ	Nachname	Geburtsdatum
1	3300	Muster	03.05.2003
2	3300	Maier	05.06.2002
3	3304	Wimmer	03.05.2003

Übung Normalisierung

1. Firmendatenbank

normalisierung_datenbank-herdt.xlsx

2. Klassendatenbank

- Öffne die Excel-Datei und führe die Datenbank im Tabellenblatt „Nicht-normalisiert“ schrittweise in 1., 2. und 3. Normalform über.
- Die Lösungen befinden sich in den weiteren Tabellenblättern.

normalisierung-datenbank-klasse.xlsx

3. Schuldatenbank

SchülerNr	Name	Vorname	Klasse	Klassenlehrer	LernangebotsNr	Beschreibung	Zeit in h
1	Jürgens	Ina	11a	Lempel	2	Tanz	12
2	Schmidt	Tom	12a	Breier	3	Chor	22
3	Jäger	Franz	11a	Lempel	1, 2, 3	Elektronik, Tanz, Chor	15, 12, 2
4	Olsen	Ina	11b	Sommer	2	Tanz	5
5	Jürgens	Paula	12a	Breier	1	Elektronik	23

a) Führe die Datenbank in die 1. Normalform über und lege einen neuen Primärschlüssel fest. (1 Relation)

b) Führe die Datenbank in die 2. Normalform über.

Hinweis

Die Lösung besteht aus drei Relationen:

- Lernangebot
- Schüler
- Lernangebotsübersicht

c) Führe die Datenbank in die 3. Normalform über.

Hinweis

Die Lösung besteht aus vier Relationen. Teile die Relation Schüler in 2 Relationen auf:

- Schüler
- Klasse

2.7) SQL - Structured Query Language

SQL (**Structured Query Language**) hat sich als Standardabfragesprache für relationale Datenbanken etabliert. SQL stellt die Schnittstelle zwischen der relationalen Datenbank und dem Anwendungsprogramm dar. Es lassen sich damit alle Operationen der Relationenalgebra aus Kapitel 4 realisieren. Die Sprachelement von SQL lassen sich in mehrere Kategorien unterteilen, die allerdings im Standard nicht festgeschrieben sind.

Allgemeines

Öffnen der MySQL-Konsole

- Xampp starten mit MySQL
- cmd
- in C:\xampp\mysql\bin wechseln
- mysql -u root

MySQL Befehlsreferenz

- mysql-befehlsreferenz.pdf

Anzeigen von Datenbanken

```
SHOW DATABASES;
```

Erstellen von Datenbanken

```
CREATE DATABASE DBName;
```

Selektieren von Datenbanken

```
USE DBName;
```

Anzeigen von Tabellen in einer Datenbank

```
SHOW TABLES;
```

Anzeigen von Spalten in einer Tabelle

```
SHOW COLUMNS FROM tabellenname;
```

Löschen von Datenbanken

```
DROP DATABASE DBName;
```

Backup und Restore von Datenbanken

- Backup

```
C:\xampp\mysql\bin>mysqldump -u root db8ai > db8ai.sql
oder alle Datenbanken:
C:\xampp\mysql\bin>mysqldump -u root –all-databases > sicherung.sql
```

- Restore

```
mysql> CREATE DATABASE db8ai;
mysql> exit;
C:\xampp\mysql\bin>mysql -u root db8ai < db8ai.sql
bzw. alle Datenbanken:
C:\xampp\mysql\bin>mysql -u root -p < sicherung.sql

oder:

mysql> CREATE DATABASE db8ai;
mysql> USE db8ai;
mysql> SOURCE C:\xampp\mysql\bin\db8ai.sql
```

2.7.1) DDL (Data Definition Language)

- Anweisungen zur Anlage und Verwaltung von Datenbankschemata
- Anweisungen zur Definition von Relationen einschließlich der Konsistenzbedingungen
- Anweisungen zur Anlage von Datensichten (Views)

- 2.7.1) DDL
    - 2.7.1.1) DDL - Übungen

2.7.2) DQL (Data Query Language)

- Abfrage von Daten

- 2.7.2) Data Query Language
    - 2.7.2.1) DQL- Übungen

2.7.3) DML (Data Manipulation Language)

- Eingabe von Daten in eine vorhandene Tabelle
- Änderung von Daten in einer Tabelle
- Löschung von Daten in einer Tabelle

2.7.4) DCL (Data Control Language)

- Anlegen von Benutzern
- Vergabe von Zugriffsrechten

Abschlussübung

Folgendes ER-Modell ist gegeben und dient als Grundlage für eine Datenbank, welche die fertiggestellten Häuser dokumentiert. Annahme: Jeder ORT ist direkt abhängig von der PLZ. D.h. Jeder Ort besitzt eine eigene PLZ!

1) Relationale Algebra

Entwickle die zugehörigen Relationen in der 3. NF. Kennzeichne die Primär (z.B.: unterstreichen und PK dazuschreiben)- und Fremdschlüssel (z.B.: Relation mit . davor schreiben) eindeutig.

2) SQL -> DDL

Erstelle die Datenbank mithilfe der SQL-Statements!

3) SQL -> DML

Befülle die Datenbank mit folgenden Daten

Personen
Franz Huber wohnt in der Anzengruberstrasse 6
Max Müller wohnt in der Anzengruberstrasse 8
Maxima Muster wohnt in der Anzengruberstrasse 7
Franz Müller wohnt in der Anzengruberstrasse 9

Häuser
Massivhaus mit Satteldach auf Grundstück 1010 mit 15000m2 in der Anzengruberstrasse 6 3300 Amstetten
Fertighaus (Ausbauhaus) auf Grundstück 1011 mit 4000m2 in der Anzengruberstrasse 7 3300

Amstetten
Fertighaus (schlüsselfertig) auf Grundstück 1012 mit 500m2 in der Anzengruberstrasse 8 3300 Amstetten
Fertighaus (Rohbau) auf Grundstück 1013 mit 1000m2 in der Anzengruberstrasse 9 3300 Amstetten

4) SQL -> DQL

Entwickle folgende SQL-Abfragen.

- Liste alle Personen auf, die den Vornamen Franz haben und in einem Haus mit einer Grundstücksfläche über 1000m2 wohnen.
- Liste jene Grundstücksnummer auf, wo ein Fertighaus mit der Ausbaustufe „Schlüsselfertig“ gebaut wurde.

2.8) SQL-Datenbankzugriff mit PHP

Um auf eine MySQL-Datenbank zuzugreifen, muss zuerst eine Verbindung (***mysqli_connect***) hergestellt werden. Dazu ist der **Host**, auf dem der MySQL-Server läuft, der **Benutzername** sowie das **Kennwort** anzugeben. Anschließend muss noch eine Datenbank auf dem DBS ausgewählt werden (***mysqli_select_db***). Danach können die SQL-Anweisungen folgen (***mysqli_query***). Im Falle einer SQL-Abfrage liefert diese Funktion alle Tupel zurück, welche der Reihe nach mit ***mysqli_fetch_array*** abgearbeitet werden können.

Im Folgenden soll als Beispiel ein Gästebuch realisiert werden. Dazu wurde eine Tabelle **meldung** erstellt, welche die einzelnen Nachrichten aufnehmen soll:

```
CREATE TABLE meldung (
  id INT NOT NULL AUTO_INCREMENT,
  datum DATETIME DEFAULT NULL,
  name VARCHAR(200) DEFAULT NULL,
  eintrag TEXT,
  CONSTRAINT PK_id PRIMARY KEY (id)
);
```

Zusätzlich zu den Daten **datum, name** und **eintrag** wurde ein Attribut **id** eingeführt, welches als Schlüssel dient.

Das folgende Skript **show.php** liest alle Einträge der Datenbank aus und gibt sie in Tabellenform aus:

```
<?php
    $DBHost = "127.0.0.1";
    $DBUser = "root";
    $DBPasswd = "";
    $DBName = "db";

    //Verbindung zu DB-Server herstellen
    $con=mysqli_connect($DBHost, $DBUser, $DBPasswd, $DBName)
      OR die("Konnte DB-Server nicht erreichen!");
    mysqli_select_db($con,$DBName);
?>
<html>
    <head>
        <title>Alle Meldungen</title>
    </head>
    <body>
    <?php
        $erg=mysqli_query($con, "SELECT datum, name, eintrag FROM meldung
ORDER BY datum DESC");
        echo mysqli_error($con);


                echo "<table bgcolor=\"#dddddd\" border=\"0\" width=\"600\">
\n";
```

```
        while($row=mysqli_fetch_array($erg))
        {

            echo "<tr><td>Datum: </td><td>".$row["datum"]."</td></tr> \n";
            echo "<tr><td>Name:</td><td>" .$row ["name"]. "</td></tr> \n";
            echo "<tr><td valign=\"top\">Eintrag:</td> \n";
            echo "<td>".nl2br(htmlentities($row["eintrag"])."</td></tr>
\n";

        }
                echo "</table>\n<br>\n";
    ?>
    <hr><a href="insert.php">neuen Eintrag hinzufügen</a>
    </body>
</html>
```

Um neue Einträge für das Gästebuch zu erstellen, existiert ein weiteres Skript insert.php:

```
<?php
    $DBHost = "127.0.0.1";
    $DBUser = "root";
    $DBPasswd = "";
    $DBName = "db";

    //Verbindung zu DB-Server herstellen
    $con=mysqli_connect($DBHost, $DBUser, $DBPasswd)
      OR die("Konnte DB-Server nicht erreichen!");
    mysqli_select_db($con,$DBName);
?>
<html>
    <head>
        <title>Neuer Eintrag in unser Gästebuch</title>
    </head>
    <body>
        <?php

            if(isset($_GET["submit"]))
            {
                $submit = $_GET["submit"];
                $name = $_GET["name"];
                $eintrag = $_GET["eintrag"];

                //Der Submit-Button wurde gedrückt --> die Werte müssen
überprüft werden
                //und bei Gültigkeit in die DB eingefügt werden

                $DatenOK=1;         //wir gehen prinzipiell von der
Gültigkeit der Daten aus
                $error="";         //es gab noch keine Fehlermeldung bis hier
```

```
hier

                if($name=="")    //Kein Name eingegeben
                {
                    $DatenOK=0;
                    $error.="Es muss ein Name eingegeben werden!<br>\n";
                }
                if($eintrag=="")    //Kein Kommentar eingegeben
                {
                    $DatenOK=0;
                    $error.="Ein Eintrag ohne Kommentar macht nicht viel
Sinn!<br>\n";
                }
                if($DatenOK)        //Daten OK -> also in DB eintragen
                {
                                        $timestamp=date("Y-m-d h:i:s",
time());
                    mysqli_query($con,"INSERT INTO eintraege (datum, name,
eintrag) VALUES (\"$timestamp\", \"$name\", \"$eintrag\" );");
                    echo mysqli_error($con);
                    echo "<b>Daten wurden eingetragen.<b>";
                }
                else
                {
                    echo "<h2>Fehler: </h2>\n"; //Fehlermeldung
                    echo $error;
                }
            }

            //Formular
        ?>



        <form action="insert.php" method="GET">
            Name: <input type="text" name="name" size="30" maxlength="200"
value=""><br>
            Text: <br><textarea rows="10" cols="50" wrap="virtual"
name="eintrag"></textarea>
            <br><input type="submit" name="submit" value="Absenden">
        </form>
        <a href="show.php"> Alle Einträge anzeigen</a>
    </body>
</html>
```

SQL Injection

SQL injection ist eine Einschleusung von Code der die Datenbank möglicherweise zerstört

SQL injection ist eine von den meist genutzten Hacking Methoden

SQL injection bezeichnet das Einschleusen von bösartigen Code in die SQL-Statements mithilfe von Formulardaten in Webseiten

SQL Injection basierend auf 1=1 ist immer wahr (true)

Nehmen wir an, es soll innerhalb einer Webseite die UserId angegeben werden, um die jeweiligen Informationen des Users auszugeben.
Ist diese Eingabe nicht beschränkt, so kann der Benutzer eingeben was er will. Füllt er diese Eingabe „intelligent“ aus, so kann er die ursprüngliche Funktion des SQL-Statements vollkommen verändern. z.B.:

Dadurch wird das SQL-Statement wie folgt verändert:

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

Somit wird dieses SQL-Statement alle Zeilen der Tabelle Users zurückgeben, da ja OR 1=1 immer erfüllt ist.
Das wäre besonders heikel, falls die Users Tabelle Benutzernamen & Passwörter beinhaltet. Dann würde das manipulierte SQL-Statement dasselbe verursachen wie folgendes SQL-Statement:

```
SELECT UserID, Name, Passowrd FROM Users WHERE UserId=105 OR 1=1
```

Durch diese Manipulation würde der Hacker Zugriff zu allen Benutzernamen und Passwörter in der Datenbank erlangen, durch einfaches hinzufügen von OR 1=1.

Aufgabe 1

Erstellen Sie eine php-Seite welche dem Benutzer ein Formular (siehe Abbildung) präsentiert. In diesem Formular kann der Benutzer auswählen, welche der 6 Aufgaben (a bis f) aus der DQL-Übung angezeigt werden sollen.