Vorlesungsmodul DbSys1
- VorlMod Datenbanksysteme 1 -

Matthias Ansorg

17. März 2003 bis 1. Juli 2003

Zusammenfassung

Studentische Mitschrift zur Vorlesung Datenbanksysteme I bei Prof. Dr. Volker Klement (Sommersemester 2003) im Studiengang Informatik an der Fachhochschule Gießen-Friedberg. Gegliedert identisch zum offiziellen Vorlesungsskript [1], es folgen noch Ergänzungen. DbSys1 besteht aus 4 SWS Vorlesungen, 2 SWS Übungen (d.i. ohne PC) bzw. Praktikum (d.i. mit PC). Für die Übungen wird eine Gruppeneinteilung vorgenommen. Man sollte die Übungen besuchen, muss aber nicht. Es wird hier Datenbankentwurf und Datenbankzugriff trainiert.

- **Bezugsquelle:** Die vorliegende studentische Mitschrift steht im Internet zum Download bereit: http://matthias.ansorgs.de/InformatikDiplom/Modul.DbSys1.Klement/DbSys1.pdf.
- **Lizenz:** Diese studentische Mitschrift ist public domain, darf also ohne Einschränkungen oder Quellenangabe für jeden beliebigen Zweck benutzt werden, kommerziell und nichtkommerziell; jedoch enthält sie keinerlei Garantien für Richtigkeit oder Eignung oder sonst irgendetwas, weder explizit noch implizit. Das Risiko der Nutzung dieser studentischen Mitschrift liegt allein beim Nutzer selbst. Einschränkend sind außerdem die Urheberrechte der angegebenen Quellen zu beachten.
- **Korrekturen und Feedback:** Fehler zur Verbesserung in zukünftigen Versionen, sonstige Verbesserungsvorschläge und Wünsche bitte dem Autor per e-mail mitteilen: Matthias Ansorg mailto:matthias@ansorgs.de.
- **Format:** Die vorliegende studentische Mitschrift wurde mit dem Programm LyX(graphisches Frontend zu LaTeX) unter Linux geschrieben und mit pdfLaTeXals pdf-Datei erstellt. Grafiken wurden mit dem Programm xfig unter Linux erstellt und als pdf-Dateien exportiert.
- **Dozent:** Prof. Dr. Volker Klement.
- **Verwendete Quellen:** Vollständig aufgenommen wurden [3], [4], [7].
- **Klausur:**

  Leistungsart: Prüfungsleistung

  Teilnahmevoraussetzung: zwei testierte Hausübungen zu SQL-Programmierung, nicht besonders umffangreich. Die Hausübungen werden auf Papier in der Vorlesung ausgeteilt, sind termingerecht abzugeben und werden vom Übungsleiter korrigiert. Je nach Übungsleiter müssen sie dann ggf. innerhalb einer Woche nachgebessert werden, bevor man das Testat erhält.

  Hilfsmittel: Keine. Die Klausur wird so leichter, da Wissensfragen so keine Detailfragen sein werden.. Aufgabensammlungen würden nicht helfen, da die Designaufgaben stets neu entwickelt werden. Die Art der Hilfsmittel kann jedoch noch diskutiert werden bis Semesterende. Die Klausur enthält eine einseitige SQL-Kurzreferenz, so dass man nicht die gesamte SQL-Syntax auswendig beherrschen muss.

  Punkteverteilung: je $\frac{1}{3}$ für allgemeine Fragen, Design-Aufgaben und SQL-Programmierung.

  Tipps:

  - Zuerst alle Aufgaben lesen und prüfen, welche man beherrscht.
  - Dann mit den Aufgaben beginnen, die man beherrscht, und die Zeit entsprechend den Punkten einteilen. So beißt man sich nicht an einer Aufgabe fest.
  - Es gibt ein System, wie die Designaufgaben und SQL-Aufgaben zusammenhängen. Man kann es aus alten Klausuren ableiten und es hilft in der Klausur.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

1.1 Ziel der Lehrveranstaltung

- Entwurf und Modellierung von Datenbankanwendungen.
- Interner Aufbau von Datenbanken.

- Kritische Sicht der Datenbanken. Heute wird für fast jede Anwendung eine Datenbank verwendet.

1.2 Begriffe

1.2.1 Was ist eine Datenbank?

Datenbank (DB, database). Der engl. Begriff (eigtl. »Datenbasis«) ist genauer.

Datenbanksystem (DBS, database system) Eine Kombination einer spezifischen Datenbank mit einem DBVS.

Datenbankverwaltungssystem (DBVS, database management system (DBMS)).

Datenbankanwendung (DBA, database application). Eine Anwendung, die eine Datenbank zur Datenhaltung benutzt.

Informationssystem Verwendet DBS als Basistechnologie zur Datenhaltung.

Datenbank-Tabelle (database table).

Datenbank-Datei (database file). Keine Datei des Dateisystems, sondern eine logische Datei, die vom DBVS verwaltet wird. Synonym für Datenbank-Tabelle, aber veraltet.

Datenmodell (data model). Modellvorstellung der Daten als Hilfsmittel vor der Implementierung der Datenbank.

Datenbank-Abfragesprache (database query language).

Datenbank-Administrator (DBA, database administrator)

1.2.2 Schalenmodell einer Datenbank

Daten können nicht mehr beliebig zugriffen werden, sondern nur über die durch das DBVS bereitgestellte Schnittstelle. Diese Kontrolle bringt alle Vorteile der Objektorientierung mit sich.

1.2.3 Übersicht zur Funktionalität eines Datenbankverwaltungssystems (DBVS)

Produkte aus dem DBVS-Umfeld

- Data Dictionary / Repository
- 4th Generation Language
- Kommunikations-Subsystem (dabei Bildschirmsteuerung, Teleprocessing-Monitor)
- Unterstützung von Netzwerkverbindungen
- Unterstützung von Schnittstellen (z.B. OLE), z.B. für graphische Ausgabe oder Integration Büroautomaten
- DBA-Utilities: Dienstprogramme für den Datenbank-Administrator

1.2.4 vereinfachtes Schichtenmodell

1.2.5 Grobarchitektur eines DBS

1.3 Literatur-Empfehlungen

Alle diese Bücher sind in der Bibliothek der FH Gießen-Friedberg ausleihbar.

1.3.1 empfohlene Lehrbücher

1.3.2 spezielle Literatur

1.3.3 Historische Literatur

1.4 Von der Dateiverarbeitung zur Datenbank

1.4.1 Nachteile der »konventionellen« Dateiverarbeitung

1.4.2 Verarbeitung mit Datenbanksystem

1.4.3 Definitionen

1.4.4 Allgemeine Anforderungen an eine Datenbank

1.4.5 Datenbankstrukturen

1.4.6 Gegenüberstellung Datei - Datenbank

1.4.7 erwartete Basisfunktionalität eines DBVS

1.4.8 Benutzertypen und Verantwortlichkeiten beim Betrieb einer Datenbank

1.4.9 Wirtschaftlichkeitsüberlegungen zum Einsatz von Datenbanken

1.4.10 »Klassische« typische Datenbank-Anwendungen

1.4.11 Entwicklungsgeschichte DBS

1.5 Terminologie (Beispiele)

5.3 Datenkompression

5.4 Zugriffspfade

5.5 Zielpunktlisten

5.6 Beispiel MS SQL Server

5.7 Beispiel ADABAS-C

6 Spezielle Funktionen eines DBVS

6.1 Datenintegrität

6.2 Transaktionslogik

6.3 Aspekte des Mehrbenutzerbetriebs

6.4 Datensicherung

6.5 Datenschutz

6.6 Data Dictionary

6.7 sonstige Funktionalität

7 Beispiele von DBVS

8 HowTo Aufgaben lösen

8.1 Vorbereitung

Rechner Für das Praktikum dieser Veranstaltung werden die NT-Rechner in Raum F113 verwendet.

Login Windows NT Anmeldung mit Username »`db`«, Passwort »`student`«. Dokumentiert in [2, P-INS 1].

Login Microsoft ISQL/w Server »`dbserv`«, Login Id »`db`«, Password »`db`«. Nicht dokumentiert in [2]!

Teilnehmer-Eintragung Auf den Aufgabenblättern der Hausübung steht: »Die Hausübung kann nur angenommen werden, wenn Sie sich in der Datenbank-Tabelle "Teilnehmer" korrekt eingetragen haben (Anleitung erfolgte in den Übungen).« Diese Eintragung geht in MS ISQL/w in der Datenbank »Praktikum« wie folgt (an einem fiktiven Beispiel):

```
INSERT INTO teilnehmer (name, vorname, matrikel_nr, semester)
  VALUES ('meinnachname', 'meinvorname', 123456, 4)
```

Erhält man mit diesem SQL-Kommando einen Fehler »Violation of UNIQUE KEY constraint [...]: Attempt to insert duplicate key in object 'Teilnehmer'« oder »Violation of PRIMARY KEY constraint: Attempt to insert duplicate key in object 'Teilnehmer'«, so erkennt man daran, dass das Kommando vorher bereits erfolgreich ausgeführt wurde. Denn das System verlangt eindeutige Matrikelnummern, weshalb erneutes Einfügen nicht möglich ist.

8.2 Benötigte Programme

Die Gruppe »Datenbanken I« auf dem Windows NT-Desktop der Client-Rechner in F113 enthält einige der hier aufgezählten Programme. Soweit sie nämlich in F113 benötigt werden, um Aufgaben zu lösen.

Microsoft ISQLw Die Online-Hilfe zu diesem Programm enthält alle benötigte Dokumentation zu dem in MS SQL Server verwendeten SQL-Dialekt Transact-SQL, inkl. einer vollständigen Dokumentation von Standard SQL.

Microsoft Query

Microsoft Access

Windows Explorer Öffnet das Verzeichnis »`D:\usr\db1`« im Windows Explorer. Das ganze Verzeichnis `D:\usr` ist über Netzwerk eingebunden und entspricht `Dbserv::\DB`. Mit dem Windows Explorer kann man auch direkt auf `Dbserv::\DB` und andere Verzeichnisse auf `Dbserv` zugreifen. Siehe Kapitel 8.3.

Microsoft SQL Server Dieses DBVS wird in den Übungen dieser Veranstaltung verwendet. Es läuft nur in der FH, es gibt auch keine Studentenversionen.

mySQL Herr Klement rät jedoch von der Verwendung von mySQL als Werkzeug zum selbständigen Lösen von Übungsaufgaben ab, da sich dieses System in einigen Dingen nicht so verhält wie benötigt und in MS SQL Server auch implementiert. Dies stört zu Anfang nicht, so dass man mySQL verwenden kann, um SQL zu lernen.

8.3 Nützliche Daten

Zugriff auf Dbserv von F113 aus: Man öffne im Windows Explorer `Netzwerkumgebung :: Gesamtes Netzwerk :: Microsoft Windows-Netzwerk :: Mni_dbserv :: Dbserv`. Laufwerksbuchstaben in der folgenden Liste beziehen sich auf die Laufwerke der Rechner in F113.

`Dbserv::\DB\Access_Daten` Daten alter MS Access Anwendungen aus vorigen Semestern.

`Dbserv::\DB\alte_Klausuren` alte Klausuren zu Datenbanksysteme 1

`Dbserv::\DB\Isql` Jede Menge Textdateien mit SQL-Befehlen, wohl Lösungen zu Aufgabenblättern aus vorigen Semestern, außerdem weitere Beispiele.

`Dbserv::\DB\Msquery` Einige Beispieldateien für Microsoft Query.

`Dbserv::\DB\Programs\screen_shots` Programm zum Erstellen von Bildschirmfotos.

`Dbserv::\DB\Screen_Shots` Screenshots als Dokumentation der Einrichtung von Microsoft SQL Server.

`Dbserv::\DB\Texte` Einige Beispieldaten im CSV-Format.

`Dbserv::\STUD` Daten anderer Praktikumsteilnehmer, ggf. als Beispiele oder Referenz. Auch Lösungen zu aktuellen Übungsaufgaben können dabei sein!

`C:\` Auf der Platte C der Client-Rechner im Raum F113 liegen oft einige nützliche Daten von Studierenden, etwa im Verzeichnis `C:\Temp`.

`D:\usr` Dies ist das über Netzwerk eingebundene Verzeichnis `DBbserv::\DB`.

8.4 HowTo Designaufgaben lösen

Das allgemeine Verfahren, um Designaufgaben zu lösen:

1. Was sind Entitäten? D.h.: Welche Typen von Objekten nennt die Aufgabenstellung?
2. Was sind Attribute? D.h.: Welche Attribute der Objekte nennt die Aufgabenstellung?
3. Was sind Beziehungen zwischen Entitäten?
4. Auflösen des bisherigen ERDs in SQL-kompatible Form. Das geschieht rein mechanisch, indem $m : n$-Beziehungen in zwei $1 : n$-Beziehungen aufgelöst werden. Die eigentliche Leistung ist, die Entitäten und ihre Beziehungen zu finden.

Tipps bei Designaufgaben:

- Für solche, die gerne etwas zu kompliziert denken: die Designaufgaben sind einfach gemeint. Deshalb steht auch unter jeder Aufgabe der Hinweis »Selbstverständlich können Sie die reale Welt so weit vereinfacht modellieren, wie dies die gestellte Aufgabe zulässt.«. Konkret bedeutet das:
  - Attribute statt Relationen. Beispiel: Vereinfachend Daten zur Leistungsabrechnung einfach als Attribut »Leistung« zu »Behandlung« speichern statt auf einen oder mehrere »BehandlungsFälle« pro Behandlung zu verweisen mit Hilfe einer Zwischenrelation »BehandlungsFallEinheit«.
  - Vereinfachte Komplexitäten annehmen wo die Aufgabenstellung nicht explizit anderes verlangt. Beispiel: An einer Behandlung ist immer genau ein Arzt beteiligt.
  - Weniger Relationen durch vereinfachte Komplexitäten. Bevor man eine neue Relation einführt, fragt man sich: ist sie durch vereinfachte Komplexitäten vermeidbar oder tatsächlich durch die Aufgabenstellung verlangt? Beispiel: Wenn nur genau ein Arzt an einer Behandlung teilnimmt, braucht man keine Zwischenrelation »BehandlungsPersonalEinheit« zwischen Behandlung und Arzt.
  - Weniger Relationen durch vereinfachte Modellierung. Bevor man eine neue Relation einführt, fragt man sich: ist sie durch ein vereinfachtes Modell vermeidbar oder tatsächlich durch die Aufgabenstellung verlangt? Beispiel: Statt für »Patient« eine Relation »Aufenthalt« einzuführen, um seine Ident-Nummer und Personaldaten auch nach Entlassung zu behalten, speichert man »Aufnahme« und »Entlassung« als Attribute zu Patient und legt bei erneuter Aufnahme einen weiteren Patienten mit neuer Ident-Nummer an.
  - Weniger Relationen durch Datenverlust. Bevor man eine neue Relation einführt, fragt man sich: brauchen wir die in ihr gespeicherten Daten überhaupt zur Lösung der Aufgabe? Beispiel: Es ist nicht erforderlich, dass die Datenbank weiß, in welchen Betten ein Patient jemals gelegen hat; statt also eine Relation »Aufenthalt« einzuführen und alle Aufenthalte zu speichern, gibt es eine $1 : 1$-Beziehung zwischen Patient und Bett. Ein Bettenwechel ändert einfach das beteiligte Bett, vergisst also in welchem Bett der Patient vorher lag.
- In der Aufgabenstellung sind alle Entitäten, die in Relationen umzusetzen sind, fettkursiv geschrieben. Zusätzlich müssen nur Zwischenrelationen für $m : n$-Beziehungen eingeführt werden.

8.5 HowTo SQL programmieren

8.5.1 SQL-Kurzreferenz

Vereinfachende Syntax aus SQL-Dialekten wird hier nicht erwähnt und sollte vermieden werden, um sich an Standard-SQL zu halten.

Cross Join Es gibt zwei äquivalente Möglichkeiten, dies zu schreiben:

```
SELECT *
  FROM Personal, Abteilung
SELECT *
  FROM Personal CROSS JOIN Abteilung
```

Inner Join Liefert nur die korrespondierenden Zeilen. Es gibt drei äquivalente Möglichkeiten, dies zu schreiben:

```
SELECT *
  FROM Personal, Abteilung
  WHERE Personal.AId = Abteilun.AId
SELECT *
  FROM Personal CROSS JOIN Abteilung
  WHERE Personal.AId = Abteilun.AId
SELECT *
  FROM Personal INNER JOIN Abteilung ON Personal.AId = Abteilung.AId
```

Outer Join Liefert die korrespondierenden Zeilen und alle fehlende Information.

Left Outer Join Liefert die korrespondierenden Zeilen und die fehlende Information rechts.

Right Outer Join Liefert die korrespondierenden Zeilen und die fehlende Information links.

8.5.2 Elemente von Aufgabenstellungen und zugehörige Elemente von SQL

9 Aufgaben und Lösungen

Hier wurden die Aufgaben der Dokumente [3], [4], [6, 1999-SS], [7], [8] vollständig integriert, außerdem einige Aufgaben aus [2]. Manche der Aufgaben ohne Quellenangabe wurden selbst erfunden.

9.1 1. Hausübung SS 2003, Aufgabe 1 von 1

Aufgabenstellung: [3]. Anders als dort verwenden wir hier vereinfachend nur die männlichen Wortformen: »In einer Datenbank sollen Informationen über alle an einer Fachhochschule abgelegten Prüfungen gespeichert werden. Bei diesen Prüfungen werden Studierende von Professoren geprüft. Vereinfacht möge je Prüfung nur ein Studierender von genau einem Professor geprüft werden. Erstellen Sie hierzu:«

9.1.1 Entity-Relationship-Diagramm

Siehe Abbildung 1. Für die Diagramme wurde eine Semantik entsprechend [1, 3.0-6] und [2, P-INS 4] verwendet und in einer Legende erklärt.

9.1.2 Tabellen-Grobentwürfe

Die Regeln für Tabellenbeschreibungen sind enhalten in [2, P-TAB].

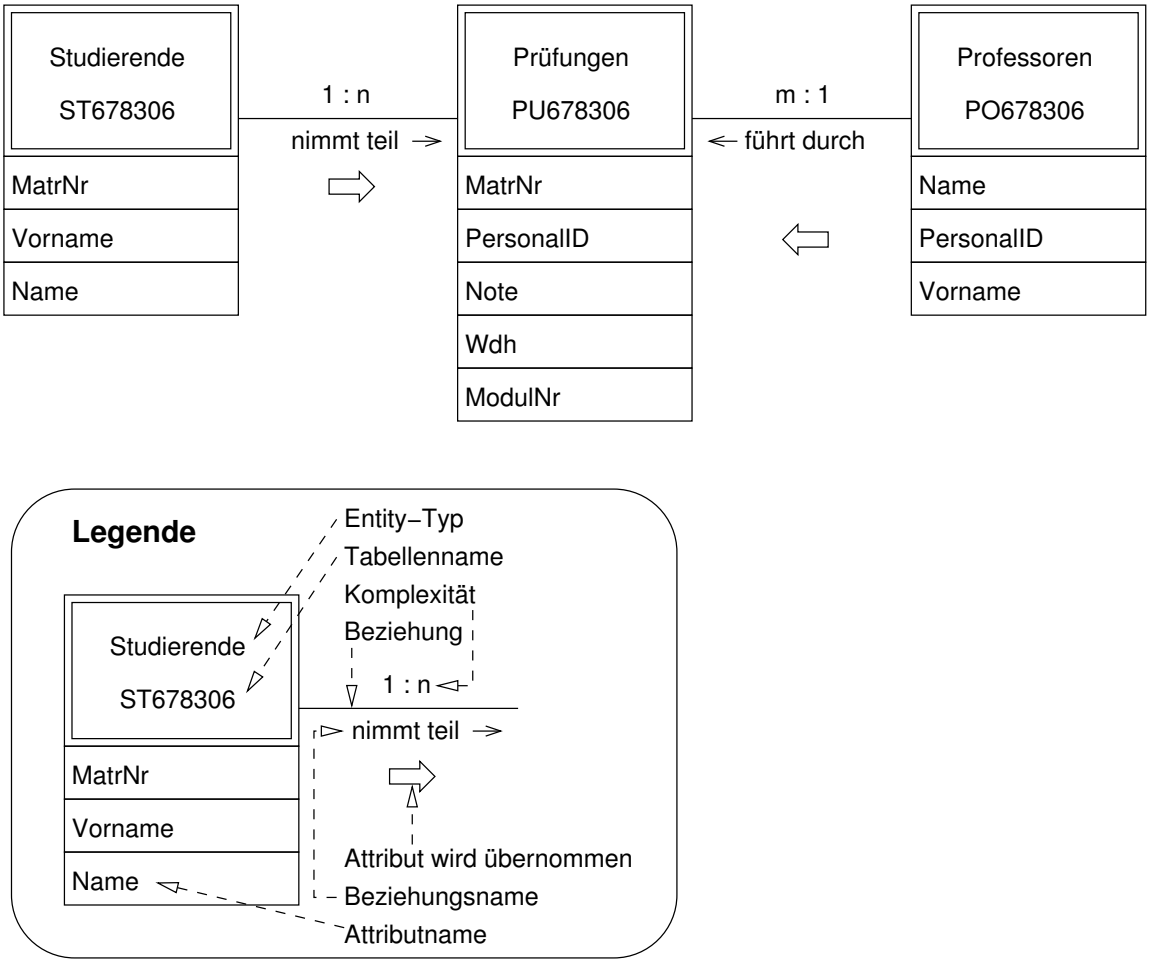

Abbildung 1: Entity-Relationship-Diagramm zu Hausübung 1

Tabelle »Studierende«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
×		MatrNr	int	4		
		Vorname	varchar	25	×	
		Name	varchar	25	×	

Primary Key / Identity	MatrNr
Foreign Keys	
Unique Constraints	
Keys	

Tabelle »Prüfungen«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
		MatrNr	int	4		
		PersonalID	int	4		
		Note	tinyint	1	×	
		Wdh	tinyint	1		1
		ModulNr	char	5		
×	×	lfdNr	int	4		

Primary Key / Identity	lfdNr
Foreign Keys	MatrNr, PersonalID
Unique Constraints	
Keys	MatrNr, ModulNr, Wdh

Tabelle »Professoren«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
		Name	varchar	25	×	
×		PersonalID	int	4		
		Vorname	varchar	25	×	

Primary Key / Identity	PersonalID
Foreign Keys	
Unique Constraints	
Keys	

9.1.3 Realisierung der Tabelle »Studierende« für MS Access

Die Lösung soll durch einen Screenshot dokumentiert werden. Siehe Abbildung 2.

9.1.4 Erzeugen aller Tabellen für MS SQL Server

Die Tabellen sollen mit SQL-Anweisungen erzeugt werden. Sie sind auf dem Server »DBSERV« für das DBVS »MS SQL Server« in der Datenbank »Scratch« abzulegen. Sie sind mit zwei Buchstaben zur Identifikation der Tabelle, gefolgt von der eigenen Matrikelnummer zur Identifikation des Autors zu benennen.

```
CREATE TABLE ST678306 (
  MatrNr      INT
    NOT NULL
    PRIMARY KEY,
  Vorname     VARCHAR(25),
  Name        VARCHAR(25)
)

CREATE TABLE PU678306 (
  MatrNr      INT
    NOT NULL,
  PersonalID  INT
    NOT NULL,
  Note        TINYINT,
```

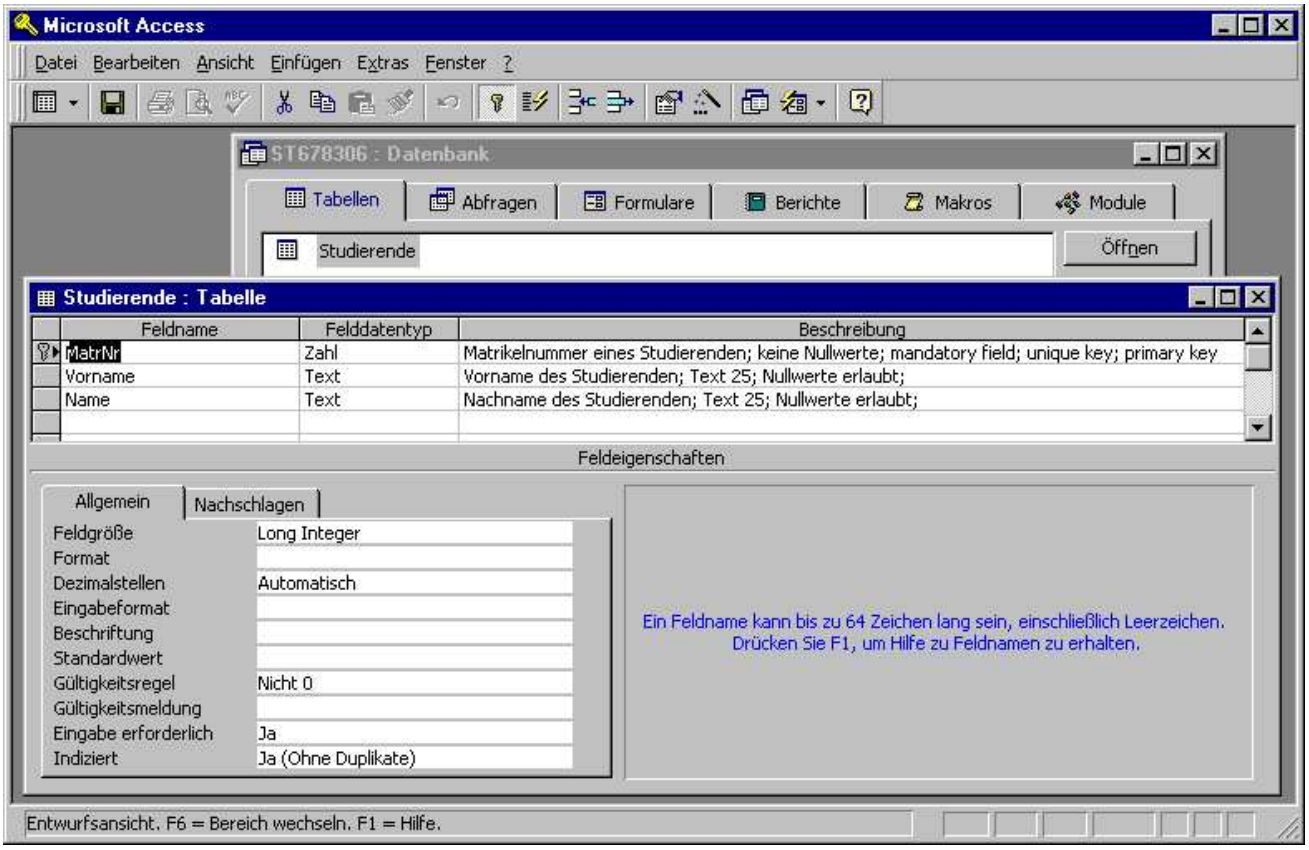

Abbildung 2: Realisierung der Tabelle Studierende in Access

```
  Wdh          TINYINT
    NOT NULL
    DEFAULT 1,
  ModulNr      CHAR(5)
    NOT NULL,
  lfdNr        INT
    NOT NULL
    PRIMARY KEY
    IDENTITY(1,1),
  FOREIGN KEY(PersonalID) REFERENCES PO678306(PersonalID),
  FOREIGN KEY(MatrNr) REFERENCES ST678306(MatrNr)
)

CREATE TABLE PO678306 (
  Name         VARCHAR(25),
  PersonalID   INT
    NOT NULL
    PRIMARY KEY,
  Vorname      VARCHAR(25)
)
```

9.1.5 Speichern einiger Beispieldaten in den Tabellen auf MS SQL Server mit einem SQL-Programm

Die Tabellen sollen also mit SQL-Anweisungen gefüllt werden.

```
/* insertions in table "Studierende" */
INSERT
```

```
  INTO ST678306 (MatrNr, Vorname, Name)
  VALUES (000001, 'Alfons', 'Alberts')
INSERT
  INTO ST678306 (MatrNr, Vorname, Name)
  VALUES (000002, 'Bernd', 'Baleppo')
INSERT
  INTO ST678306 (MatrNr, Vorname, Name)
  VALUES (000003, 'Carina', 'Curz')
INSERT
  INTO ST678306 (MatrNr, Vorname, Name)
  VALUES (000004, 'Daniel', 'Düsentrieb')
INSERT
  INTO ST678306 (MatrNr, Vorname, Name)
  VALUES (000005, 'Edgar', 'Eilig')

/* insertions in table "Professoren" */
INSERT
  INTO PO678306 (Name, PersonalID, Vorname)
  VALUES ('Ziegler', 000001, 'Zorro')
INSERT
  INTO PO678306 (Name, PersonalID, Vorname)
  VALUES ('Yukon', 000002, 'Yoko')
INSERT
  INTO PO678306 (Name, PersonalID, Vorname)
  VALUES ('Xray', 000003, 'Xaver')
INSERT
  INTO PO678306 (Name, PersonalID, Vorname)
  VALUES ('Wandersmann', 000004, 'Winfried')
INSERT
  INTO PO678306 (Name, PersonalID, Vorname)
  VALUES ('Vogler', 000005, 'Vriedrich')

/* insertions in table "Prüfungen" */
INSERT
  INTO PU678306 (MatrNr, PersonalID, Note, Wdh, ModulNr)
  VALUES (000001, 000001, 5, 1, 'IG011')
INSERT
  INTO PU678306 (MatrNr, PersonalID, Note, Wdh, ModulNr)
  VALUES (000001, 000002, 1, 2, 'IG011')
INSERT
  INTO PU678306 (MatrNr, PersonalID, Note, Wdh, ModulNr)
  VALUES (000002, 000003, 1, 1, 'MG010')
INSERT
  INTO PU678306 (MatrNr, PersonalID, Note, Wdh, ModulNr)
  VALUES (000003, 000004, 5, 1, 'MG011')
INSERT
  INTO PU678306 (MatrNr, PersonalID, Note, Wdh, ModulNr)
  VALUES (000004, 000005, 3, 1, 'IF004')
```

9.1.6 SQL-Programm, das eine Statistik erzeugt, welcher Professor wieviele Prüfungen hatte

Das »Programm« sind SQL-Anweisungen. Das Ergebnis soll eine Tabelle sein, deren eine Spalte den Namen des Professors und deren andere Spalte die Anzahl der Prüfungen enthält. Dabei sind mögliche Namensgleichheiten von Professoren zu beachten.

```
SELECT
  PO678306.Name AS ProfName,
  PO678306.Vorname AS ProfVorname,
```

```
PO678306.PersonalID AS ProfPersonalID,
COUNT(PU678306.PersonalID) AS Anzahl
FROM PU678306, PO678306
  WHERE PO678306.PersonalID = PU678306.PersonalID
  GROUP BY PO678306.Name, PO678306.Vorname, PO678306.PersonalID
```

9.2 2. Hausübung SS 2003, Aufgabe 1 von 2

Die hier gezeigte Lösung ist wesentlich zu kompliziert; die Aufgabenstellung ist einfacher gemeint. Die Relationen »Aufenthalt«, »BehandlungsFallEinheit«, »BehandlungsFall« und »BehandlungsPersEinheit« können mit den in Kapitel 8.4 gezeigten Vereinfachungen vermieden werden.

Aufgabenstellung: [4].

»Für das Informationssystem eines großen Krankenhauses soll in einer Datenbank gespeichert werden, wann welcher Patient von welchem Arzt behandelt wurde. Ferner sollen Informationen über die Belegung der Betten gespeichert werden.

Das Krankenhaus ist untergliedert in Stationen. Ärzte und Patientenzimmer sind jeweils einer Station zugeordnet. In den Patientenzimmern werden Betten aufgestellt. Patienten können von mehreren Ärzten behandelt werden und können, zeitlich nacheinander, in mehreren Betten / Zimmern / Stationen liegen.

Stationen haben neben ihrer Bezeichnung eine interne, aus bis zu 3 Zeichen bestehende Kurzbezeichnung. Örtlich sind sie lokalisiert durch Angabe des Gebäudes und des Stockwerks.

Patientenzimmer werden innerhalb einer Station durchnummeriert.

Betten haben eine interne vierstellige Nummerierung und eine Bauart, aus welcher mögliche spezielle Verwendungen hervorgehen.

Zu Ärzten sind u.a. das vertretene Fachgebiet, vorhandene Spezialkenntnisse sowie alle wichtigen Daten bezüglich der Erreichbarkeit zu speichern. Daneben Angaben zur Personalverwaltung wie Dienststellung, Gehaltsklasse, EInstellungsdatum, Datum Ausscheiden etc.

Zu Patienten sind administrative Daten, auch zur Leistungsabrechnung zu speichern. Sie werden krankenhausweit identifiziert durch eine »I-Zahl«, welche aus dem Anfangsbuchstaben des Familiennamens, dem Geburtsdatum, einem Kürzel für das Geschlecht und einer zusätzlichen 2-stelligen fortlaufenden Nummer gebildet wird.

Häufig vorkommende zu unterstützende Suchanfragen an die Datenbank können sein:

- Auskunft, ob es ein freies Bett gibt
- Auskunft, auf welcher Station und in welchem Zimmer ein Patient liegt
- mittlere Verweildauer Patienten, Auslastungsgrad Betten, Zahl Behandlungen je Monat und Arzt«

»Selbstverständlich können sie die »reale Welt« so weit vereinfacht modellieren, wei dies die gestellte Aufgabe zulässt.«

9.2.1 Entity-Relationship-Diagramm

»Zeichnen Sie ein Entity-Relationship-Diagramm, welches unmittelbar mit einer »relationalen« Datenbank implementierbar ist.« Lösung dazu in Abbildung 3. Erläuterungen dazu:

Arzt Die Komplexität der Beziehung »Arzt« zu »SpezialKenntnisName« ist $m : n$. Zur Umsetzung in relationalen Datenbanken wird deshalb eine weitere Tabelle eingeführt, hier »SpezialVerwendung«. Diese Tabelle enthält Tupel aus Primärschlüssel des Arztes und des Spezialkenntnisnamens, d.h. mögliche (Spezial-)Verwendungen von Ärzten. Daher die Namensgebung.

Behandlung Die Komplexität der Beziehung »Behandlung« zu »Arzt« ist $m : n$: an einer Behandlung können mehrere Ärzte gleichzeitig teilnehmen (OP, ...), und ein Arzt kann nacheinander an mehreren Behandlungen beteiligt gewesen sein. Es wird also wieder eine zusätzliche Tabelle zur Umsetzung in relationalen Datenbanken eingeführt, die hier »BehandlungsPersEinheit« genannt wurde, denn eine Behandlung kann mehrere solcher PersonalEinheiten benötigen.

BehandlungsFall Ein Behandlungsfall repräsentiert Diagnosen, die denselben Fall darstellen. Ein »Fall« ist ein Konzept zurAbrechnung: im derzeitigen Gesundheitswesen wird mit Krankenhäusern über Fallpauschalen abgerechnet. Die Komplexität der Beziehung »Behandlung« zu »BehandlungsFall« ist $m : n$. Zur Umsetzung in relationalen Datenbanken wird daher die Tabelle »BehandlungsFallEinheit« eingeführt.

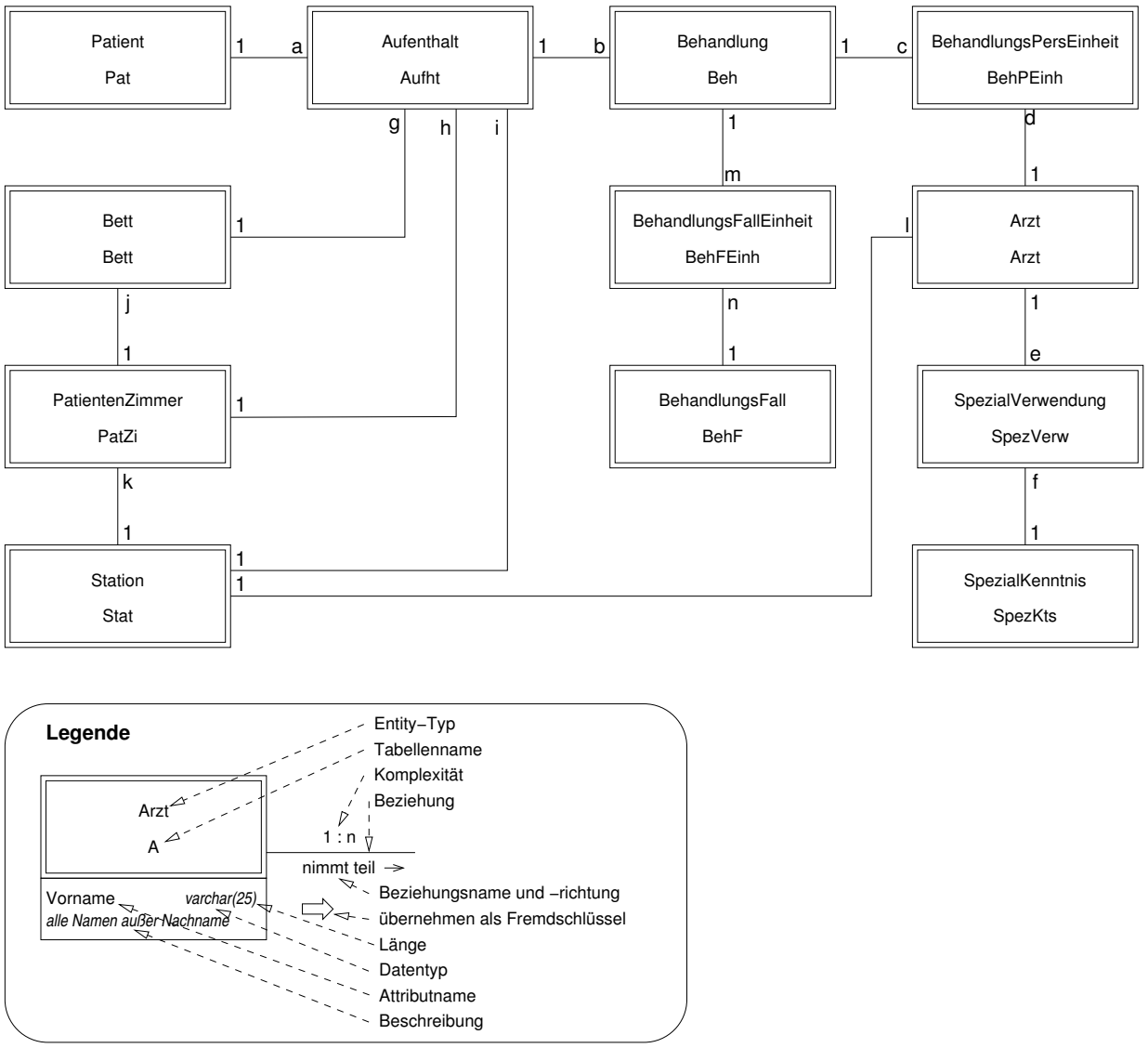

Abbildung 3: Entity-Relationship-Diagramm zu Hausübung 2 SS2003, Aufgabe 1a

Aufenthalt Wird ein Patient mehrmals in das Krankenhaus eingeliefert, so werden die schon erfassten Daten inkl. der »I-Zahl« wiederverwendet. Deshalb gibt es keine Attribute »Aufnahmedatum« und »Entlassungsdatum« für »Patient«, sondern ein Patient kann mehrere Aufenthalte haben, zu denen jeweils Beginn- und Endzeitpunkt gehören. Auch ein Bett-, Zimmer- oder Stationswechsel führt zum Anlegen eines neuen Aufenthalts.

9.2.2 Tabellengrobentwürfe

»Skizzieren Sie zu jeder Tabelle die von ihnen für erforderlich gehaltenen Datenfelder (Name, Typ, Länge, ggf. Anmerkungen). Markieren Sie eventuelle Identifikationsschlüssel und Fremdschlüssel in den einzelnen Tabellen.«

Tabelle »Arzt«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
×		PersNr	int	4		
		StationID	smallint	2		
		Vorname	varchar	25	×	
		Name	varchar	25	×	
		StrasseNr	varchar	30	×	
		PLZ	char	5	×	
		Wohnort	varchar	30	×	
		TelNr	varchar	15	×	
		Mobiltelefon	varchar	15	×	
		email	varchar	30	×	
		Fachgebiet	varchar	30	×	
		Dienststellung	char	3		ARZ
		Gehaltsklasse	tinyint	1		
		BeschäftigtSeit	datetime	8		
		BeschäftigtBis	datetime	8	×	

Primary Key / Identity	PersNr
Foreign Keys	StationID
Unique Constraints	Mobiltelefon
Keys	

Tabelle »Aufenthalt«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
		IZahl	int	4		
		Aufnahme	datetime	8		
		Entlassung	datetime	8	×	
×	×	AufhtID	int	4		
		StationID	smallint	2	×	
		PatZiID	smallint	2	×	
		BettNr	smallint	2	×	

Primary Key / Identity	AufhtID
Foreign Keys	IZahl, StationID, PatZiID, BettID
Unique Constraints	
Keys	IZahl, Aufnahme, Entlassung

Tabelle »Behandlung«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
×	×	BehID	int	4		
		AufhtID	int	4		

Primary Key / Identity	BehID
Foreign Keys	AufhtID
Unique Constraints	
Keys	

Tabelle »BehandlungsFall«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
×	×	BehFID	int	4		
		FallName	varchar	35		
		FallPauschale	money	8	×	

Primary Key / Identity	BehFEinhID
Foreign Keys	BehID, BehFID
Unique Constraints	
Keys	

Tabelle »BehandlungsFallEinheit«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
×	×	BehFEinhID	int	4		
		BehID	int	4		
		BehFID	int	4		
		AbrechnungGestellt	datetime	8	×	
		AbrechnungErledigt	datetime	8	×	

Primary Key / Identity	BehFEinhID
Foreign Keys	BehID, BehFID
Unique Constraints	
Keys	

Tabelle »BehandlungsPersEinheit«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
		BehID	int	4		
		PersNr	int	4		

Primary Key / Identity	
Foreign Keys	BehID, PersNr
Unique Constraints	
Keys	BehID, PersNr

Tabelle »Bett«

Key	Identity	Column Name	Datatype	Size	Nulls	Default	Notes
×		BettNr	smallint	2			
		Bauart	char	4			Herstellertypbezeichung
		PatZiID	smallint	2			

Primary Key / Identity	BettNr
Foreign Keys	PatZiID
Unique Constraints	
Keys	

Tabelle »Patient«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
×		IZahl	int	4		
		Vorname	varchar	25	×	
		Name	varchar	25	×	
		StrasseNr	varchar	30	×	
		PLZ	char	5	×	
		Wohnort	varchar	30	×	
		TelNr	varchar	15	×	
		HausarztName	varchar	25	×	
		HausarztOrt	varchar	30	×	
		Geschlecht	char	1		
		GeburtsTag	datetime	8	×	
		GeburtsOrt	varchar	30	×	
		KrKasseNr	int	4	×	
		VersichertenNr	int	4	×	

Primary Key / Identity	IZahl
Foreign Keys	
Unique Constraints	
Keys	

Tabelle »PatientenZimmer«

Key	Identity	Column Name	Datatype	Size	Nulls	Default	Notes
×	×	PatZiID	smallint	2			
		PatZiNr	tinyint	1			ZiNummer stationsintern
		StationID	smallint	2			

Primary Key / Identity	PatZiID
Foreign Keys	StationID
Unique Constraints	
Keys	StationID, PatZiNr

Tabelle »SpezialKenntnis«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
×	×	SpezKtsID	int	4		
		SpezKtsName	varchar	30		

Primary Key / Identity	SpezKtsID
Foreign Keys	
Unique Constraints	
Keys	

Tabelle »SpezialVerwendung«

Key	Identity	Column Name	Datatype	Size	Nulls	Default
		PersNr	int	4		
		SpezKtsID	int	4		

Primary Key / Identity	
Foreign Keys	PersNr, SpezKtsID
Unique Constraints	
Keys	PersNr, SpezKtsID

Tabelle »Station«

Key	Identity	Column Name	Datatype	Size	Nulls	Default	Notes
×	×	StationID	smallint	2			
		KurzBezeichnung	char	3			
		Bezeichnung	varchar	25			
		Ort	char	3			2 Zeichen Gebäude, 1 Ziffer Stockwerk

Primary Key / Identity	StationID
Foreign Keys	
Unique Constraints	
Keys	

9.3 2. Hausübung SS 2003, Aufgabe 2 von 2

Aufgabenstellung: [4]. »Nachfolgend sehen Sie das ERD eines kleinen Systems zur Verwaltung von Gewinnen / Verlusten aus Wertpapierverkäufen. [vgl. Abbildung 4]. Sie finden die Datenbank für MS Access als Datei »Wertpapier.mdb« auf dem Server DBSERV im Verzeichnis D:\DB\Access_Daten. Erstellen Sie folgende SQL-Programme für MS Access (oder MS SQL Server). Abzugeben sind Listen der SQL-Quellcodes und Ausdrucke der Ergebnisse (Kapitel 9.3.2 bis Kapitel 9.3.5). Achten Sie auf geeignete Spaltenüberschriften!«

9.3.1 neue Tabelle GV

»Erstellen Sie eine neue Tabelle »GV«, in welcher Sie neben den in der Tabelle »TA« enthaltenen Spalten ene Spalte »G_V« (Gewinn / Verlust) einführen. Die Spalte errechnet sich aus »Verkaufspreis – Kaufpreis«. Die Tabelle soll nach »Datum_VK« sortiert sein.«

```
SELECT (Preis_VK-Preis_K) AS G_V, *
  INTO GV
  FROM TA
  ORDER BY TA.Datum_VK
```

9.3.2 Gesamt-Gewinn / -Verlust in 2001

»Errechnen Sie anhand der Tabelle »GV« den Gesamt-Gewinn / -Verlust für alle Verkäufe im Kalenderjahr 2001 . . . «

```
SELECT Sum(G_V) AS GesGV2001
  FROM GV
  WHERE Datum_VK BETWEEN #1/1/2001# AND #31/12/2001#
```

Ergebnisliste:

```
GesGV2001
16.192,43 DM
```

». . . sowie die Anzahl der Verkäufe[1] «.

```
SELECT COUNT(*) AS GesVerkZahlen2001
  FROM GV
  WHERE Datum_VK BETWEEN #1/1/2001# AND #31/12/2001#
```

Ergebnisliste:

```
GesVerkZahlen2001
24744
```

9.3.3 steuerlich anzugebender Gesamt-Gewinn / -Verlust in 2001

»Errechnen Sie analog zu Kapitel 9.3.2 den steuerlich anzugebenden Gesamt-Gewinn / -Verlust in 2001. Dieser wird nach derzeit geltendem Steuerrecht als »privater Veräußerungsgewinn« (früher: »Spekulationsgewinn«) bezeichnet. Dabei sind solche Gewinne / Verluste steuerlich freigestellt, bei denen zwischen Kauf und Verkauf mehr als ein Jahr liegt.«

```
SELECT Sum(G_V) AS SteuerGesGV2001
  FROM GV
  WHERE GV.Datum_VK BETWEEN #1/1/01# AND #12/31/01#
    AND (GV.Datum_VK - GV.Datum_K < 365)
```

[1] Gemeint ist hier nicht die Anzahl der verkauften einzelnen Aktien (erhältlich über `SUM(Anzahl)`), sondern die Anzahl der verkauften Aktienpakete.

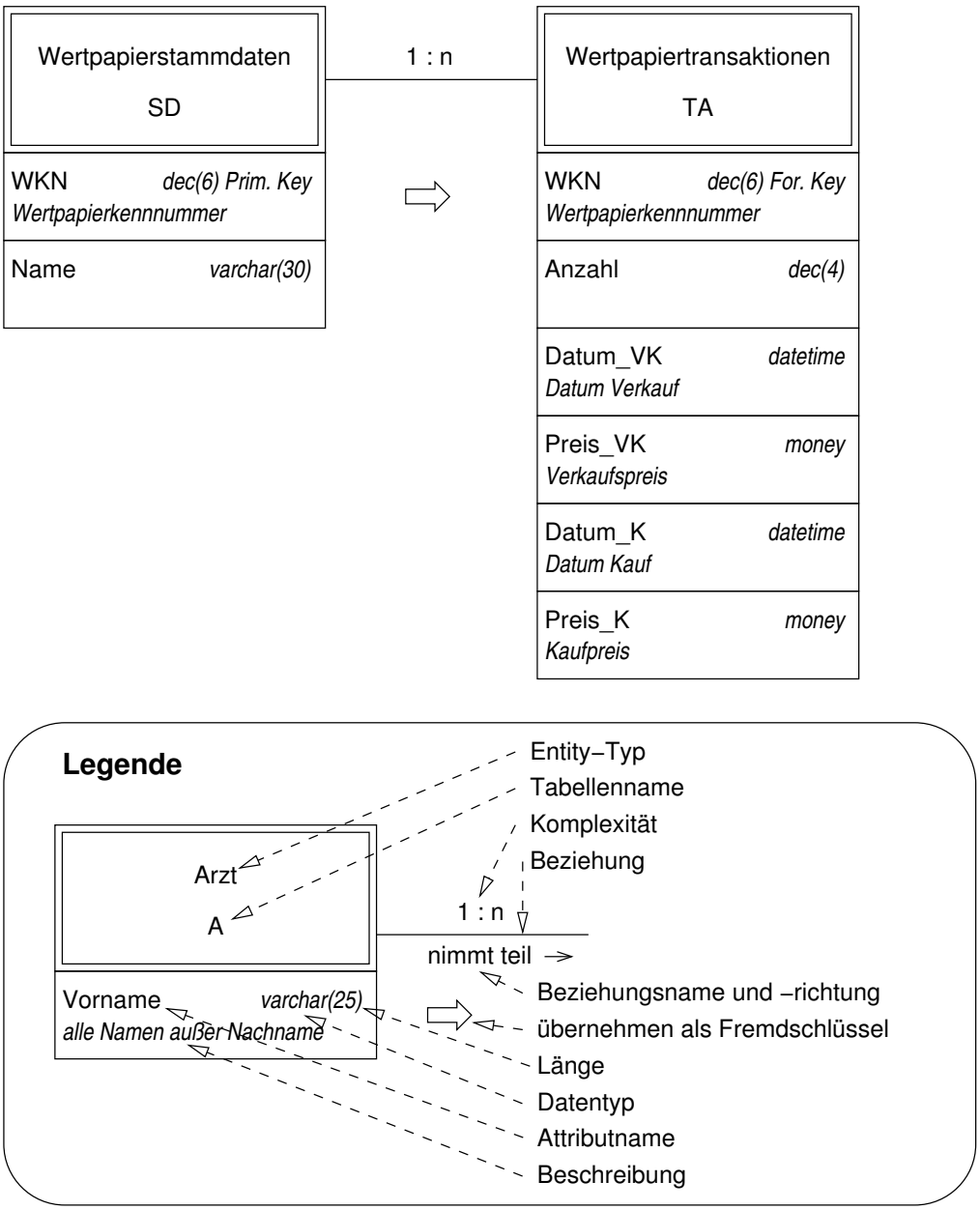

Abbildung 4: ERD eines kleinen Systems zur Verwaltung von Gewinnen / Verlusten aus Wertpapierverkäufen

Ergebnisliste:

```
SteuerGesGV2001
14.835,24 DM
```

9.3.4 Wertpapiere ohne Transaktionen

»Gibt es Wertpapiere in »SD«, zu denen es in »TA« keine Transaktionen gibt? Listen Sie solche Wertpapiere (WKN, Name) aufsteigend sortiert nach WKN«.

```
SELECT *
  FROM SD
  WHERE SD.WKN NOT IN (SELECT WKN FROM TA)
  ORDER BY SD.WKN
```

Ergebnisliste:

```
WKN        Name
581005     Dt. Börse
591068     Vivendi
593703     MAN Vz.
802200     Hypovereinsbank
850471     Boeing
855681     Intel
855686     Walt Disney
856958     Mc Donalds
```

9.3.5 Liste aller Gewinne / Verluste bei Verkäufen in 2001

»Erstellen Sie anhand der Tabellen »GV« und »SD« eine Liste aller Gewinne / Verluste bei Verkäufen im Kalenderjahr 2001, wobei Sie Verkäufe nach »WKN« zusammenfassen und die jeweilige Anzahl der Verkäufe mit angeben. Die Liste soll zum Gewinn / Verlust die »WKN« und den »Name« des jeweiligen Wertpapiers enthalten und nach Summe der Gewinne / Verluste für ein Wertpapier aufsteigend sortiert sein.«

```
SELECT GV.WKN, Name, sum(Anzahl) AS Gesamtanzahl, sum(G_V) AS GewinnVerlust
  FROM GV INNER JOIN SD ON GV.WKN = SD.WKN
  WHERE Datum_VK BETWEEN #1/1/2001# AND #31/12/2001#
  GROUP BY GV.WKN, Name
  ORDER BY sum(G_V)
```

Ergebnisliste:

```
WKN    Name             Gesamtanzahl GewinnVerlust
871111 SUN              1600         -3.240,90 DM
703003 Rheinmetall      1200         -1.956,96 DM
940602 Philips          600          -1.590,65 DM
870737 Nokia            500          -1.469,10 DM
716463 SAP              555          -1.314,12 DM
623100 Infineon         900          -1.201,62 DM
555750 Dt. Telekom      300          -634,57 DM
607920 Orange           400          26,49 DM
730790 Kolbenschmidt    400          47,38 DM
852362 Carrefour        100          83,36 DM
575200 Bayer            100          134,59 DM
840400 Allianz          20           198,85 DM
843002 Münchner Rück    20           234,70 DM
860028 Unilever         200          242,87 DM
628060 Kamps            300          270,25 DM
871028 Endesa           600          273,73 DM
875773 BBVA             300          276,84 DM
```

```
519003 BMW Vz.           300          291,76 DM
873102 Alcatel           200          303,76 DM
868400 AT&T              600          314,98 DM
851144 General Electric  200          336,16 DM
726430 Stinnes           400          339,23 DM
896356 TIM               500          340,26 DM
568480 EM TV             1300         366,30 DM
803200 Commerzbank       400          419,41 DM
851301 Hewlett Packard   400          423,73 DM
520000 Beiersdorf        100          441,91 DM
852009 Pfizer            400          469,66 DM
852491 Suez Lyon         30           517,21 DM
724264 Software AG       700          551,22 DM
804700 Depfa             200          625,83 DM
717200 Schering          400          641,66 DM
823212 Lufthansa         600          673,50 DM
870747 Microsoft         100          741,02 DM
515100 BASF              600          764,74 DM
620440 IWKA              450          792,13 DM
901626 Qiagen            1100         827,59 DM
514000 Deutsche Bank     100          937,71 DM
622700 Intershop         1300         984,83 DM
725750 Metro             300          1.123,69 DM
648300 Linde             600          1.192,30 DM
604843 Henkel            600          1.255,39 DM
710000 Daimler           700          1.267,89 DM
527800 Buderus           769          1.398,12 DM
766400 VW                600          1.472,73 DM
625700 IDS Scheer        700          1.476,59 DM
723132 Sixt              900          2.007,65 DM
593700 MAN St.           1100         2.512,33 DM
```

9.4 Was ist ein ERD?

9.4.1 Aufgabenstellung

Was wird durch ein Entity-Relationship-Diagramm dargestellt? Antworten Sie mit einem Satz!

9.4.2 Lösung

Ein ERD modelliert formal einen relevanten Teilbereich der Realität mit Entity-Typen (»Klassen«), ihre Attributen und Beziehungstypen (»Relationship-Typ«); es gehört zum konzeptionellen Modell beim DB-Entwurf.

9.5 Superdeskriptor

9.5.1 Aufgabenstellung

Erläutern Sie den Begriff »Super-Deskriptor« anhand eines Beispiels!

9.5.2 Lösung

Der Superdeskriptor ist eine Kombination von Feldern und Feldteilen, welche zur Suche verwendet werden können. Beispiel: in folgender Tabelle ist »PersNr« ein Deskriptor, die Kombination »Name UND Vorname« aber ein Superdeskriptor. Eine Suche wäre etwa »Name UND Vorname = Schulz, Werner«.

PersNr	Name	Vorname	Abteilung
1212	Müller	Hans	Marketing
3234	Schulz	Werner	Support
1242	Karl	Josef	Marketing

9.6 Regeln des Datenbankentwurfs verletzt?

9.6.1 Aufgabenstellung

Eine Datenbank-Tabelle »MUSIK« enthalte folgende Attribute:

Feldname	Datentyp(Länge)	Schlüssel?	Bemerkungen
Titel	varchar(30)	×	
Interpret_Langname	varchar(30)	×	
Interpret_Kurzname	varchar(30)	×	
Interpret_Aliasname	varchar(30)	×	multiples Feld mit bis zu 5 Realisierungen
E-Jahr	dec(4)		
Fundstelle	varchar(10)		

1. Wo sehen Sie Verstöße gegen die Regeln des Datenbankentwurfs, speziell bei Nutzung eines streng »relationalen« DBS? Erläutern Sie jeweils!
2. Wo ergeben sich hieraus konkrete Nachteile?
3. Wie müßten Sie umformen? Skizzieren Sie Ihre Lösung!
4. Bringt diese Umformung auch Nachteile mit sich? Wenn ja, welche?

9.6.2 Lösung

1. Verstöße gegen die Regeln des Datenbankentwurfs:
   - Für jeden Schlüssel wird eine »invertierte Liste« angelegt, die auf dem Massenspeicher ähnlich B*-Bäumen abgelegt werden. Deshalb sollte die Feldindizierung »weise« und nicht »verschwenderisch« vorgenommen werden. Im Beispiel ist die Indizierung der Aliasnamen nicht unbedingt »weise«.
   - Das Beispiel hält sich nicht an das relationale Datenmodell: es genügt nicht der 1. Normalform, weil es multiple Felder enthält.
   - Interpret_Kurzname ist hat gleiche Feldlänge wie Interpret_Langname. Das ist wahrscheinlich Platzverschwendung.
   - Der Primärschlüssel ist hier eine Kombination aus »Titel« und »Interpret_Langname«, man sollte sich also überlegen einen »künstlichen« Primärschlüssel einzuführen wie etwa »Musikstück-ID«.
2. Konkrete Nachteile:
   - Bei Umsetzung in ein streng »relationales« DBS: Es gibt keine multiplen Felder. Die Abfrage kann also nicht nach Art von »SELECT ... WHERE 'Mr. Joy' in Interpret_Aliasname« geschehen, sondern es müssen 5 Spalten für Aliasnamen angelegt werden und die umständliche Abfrage geschieht mit 5 OR-Verknüpfungen.
   - Die Speicherplatzverschwendung führt höchstens zu Effizienznachteilen.
3. Siehe Abbildung 5.
4. Durch diese Umformung in das relationale Datenmodell ergeben sich auch Nachteile: Etwas Redundanz wird durch Auflösung des multiplen Feldes gebracht, da nun zu jedem Aliasnamen die Stück_ID gespeichert ist. Ein weiterer Nachteil ist, daß man bei der Standard-SQL Ausgabe der Tabelle nicht mehr die Aliasnamen sieht, da diese sich in der anderen Tabelle befinden.

9.7 Redundanzfreiheit

9.7.1 Aufgabenstellung

Eine der Anforderungen an eine Datenbank ist »Redundanzfreiheit«.

1. Was versteht man darunter?
2. Warum fordert man Redundanzfreiheit?
3. Warum verstößt man in realen Anwendungen gelegentlich gegen diese Forderung?

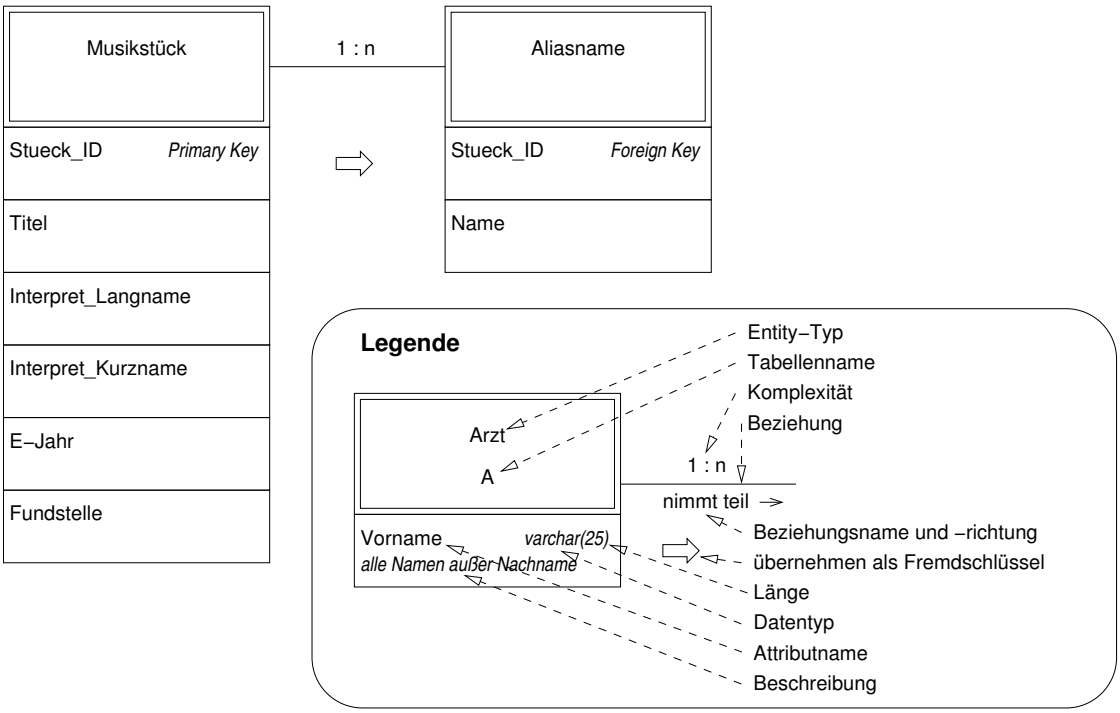

Abbildung 5: Zu Aufgabe »Regeln des Datenbankentwurfs verletzt?«

9.7.2 Lösung

1. Redundanz ist dann vorhanden, wenn ein Teil der Daten ohne Informationsverlust weggelassen werden kann, d.h. mehrfaches Auftreten des gleichen Datums. Durch geschickte Umformung im konzeptionellen Entwurf vermeidet man Redundanz.

2. Redundanz

   - kostet Speicherplatz
   - bedeutet einen hohen Pflegeaufwand
   - stellt ein Aktualitäts- und Geschwindigkeitsproblem dar
   - kann Ursache für Anomalien sein

3. Um dem verwendeten Datenmodell zu genügen, um Zugriffszeiten zu optimieren, meist aber durch fehlerhafte konzeptionelle Modellierung.

9.8 Schwächen relationaler DBS

9.8.1 Aufgabenstellung

Worin werden heute bei »relationalen« DBS Schwächen gesehen? Erläutern Sie nach Ihrer Wahl einen solchen wesentlichen Punkt, schildern sie auch eine typische Applikation!

9.8.2 Lösung

Jedes Datenmodell hat Stärken und Schwächen. Die Schwächen des relationalen Modells liegen in folgenden Punkten (siehe [1, 4.5-2]):

- keine einheitliche Unterstützung von BLOBs
- keine benutzerdefinierten Datentypen und Operationen (»abstrakte Datentypen«), kein Vererbungskonzept
- keine erweiterten Basis-Datentypen wie Vektoren, Listen, Matrizen

- keine »multiplen Felder«
- keine geordneten Relationen (d.h. mit fester Reihenfolge ihrer Tupel)
- keine Zeitversionsverwaltung (alte Zustände sollen abfragbar sein)

Ein wesentlicher Punkt ist die Unterstützung abstrakter Datentypen, also die Erweiterung zu einem semantischen Datenmodell Richtung ODBMS. Das ermöglicht bessere Modellbildung (»Abstraktion«) und natürlichere Programmierung, besonders aber die einfachere, direkte Bedienung aus den heute üblichen objektorientierten Programmiersprachen.

9.9 Allgemeine Ziele beim DB-Design

9.9.1 Aufgabenstellung

Welches sind allgemeine Ziele beim DB-Design[2]? Kreuzen Sie an!

9.9.2 Lösung

Nach [1, 3.0-7]:

×	Erreichung Flexibilität
	Reduzierung des Umfangs der Aufgabenstellung zur einfacheren Lösbarkeit
×	Erzielung minimaler Verarbeitungszeiten
×	Minimierung Speicherbedarf
	Minimierung der Zahl der Tabellen
	Verteilung der Zugriffe auf verschiedene Massenspeicher
	Wiederverwendbarkeit der Programme
×	Ermöglichung einfacher Programmlogik

9.10 Probleme bei Redundanz

9.10.1 Aufgabenstellung

Welche Probleme können auftreten, wenn eine Datenbank Redundanz enthält? Kreuzen Sie an!

9.10.2 Lösung

Nach [1, 4.4-6 bis 4.4-8]:

	Head-Crash Plattenspeicher
×	Update-Anomalien
×	mehr Speicherplatzbedarf
×	inkonsistente Daten
	größerer Suchaufwand
	häufiger Gruppenwechsel

9.11 Beziehungen in relationalen Datenbanken

9.11.1 Aufgabenstellung

Das klassische Relationenmodell kennt keine Beziehungen zwischen Relationen. Wie kann man dennoch solche für die Praxis unverzichtbaren Relationen darstellen? Erläutern Sie anhand eines Beispiels!

9.11.2 Lösung

Siehe [1, 4.4-4]. Im »erweiterten Relationenmodell« können $1 : n$-Beziehungen dargestellt werden. Dazu wird der Primärschlüssel der unabhängigen Relation in der abhängigen Relation als Fremdschlüssel wiederholt (»nach unten durchgereicht«). Beispiel: in Abbildung 5 ist »Musikstück« die unabhängige Relation, »Aliasname« die abhängige Relation in der $1 : n$-Beziehung. Deshalb wird der Primärschlüssel von »Musikstück« als Fremdschlüssel »Stueck_ID« von der Relation »Aliasname« wiederholt.

[2] »DB-Design« ist synonym zu »DB-Entwurf«; siehe [1, 3.0-1].

9.12 Datenkompression bei Speicherung durch ein DBVS

9.12.1 Aufgabenstellung

Nennen Sie Vorteile des Einsatzes von Datenkompressionstechniken bei der Datenspeicherung durch ein DBVS!

9.12.2 Lösung

Nach [1, 5.3-1]:

- geringerer Speicherbedarf
- dadurch geringere Zugriffszeiten
- und geringere Datenübertragungszeiten
- Einziger Nachteil: (De)Kompression verursacht höhere CPU-Last. Wird in Kauf genommen.

9.13 SQL-Anweisungen zum Datenretrieval

9.13.1 Aufgabenstellung

Nennen Sie die in Standard-SQL verfügbaren Anweisungen zum Retrieval von DB-Daten!

9.13.2 Lösung

SELECT

9.14 Architektur- und Datenmodelle

9.14.1 Aufgabenstellung

Was versuchen folgende Modelle zu beschreiben/modellieren?

- Relationenmodell
- ANSI/SPARC-Modell
- Hierachisches Modell
- Netzwerkmodell

9.14.2 Lösung

Vorbemerkung: Ein Datenmodell ist ein formaler Rahmen zur Beschreibung der Daten auf logischer, nicht physikalischer Ebene. Es beschreibt die Form, wie man die Daten und ihre Beziehungen sehen will. Nach [1, 4.1-1].

Relationenmodell: Ein Datenmodell. Sieht die Daten gespeichert in Tabellen (Relationen) mit Zeilen (Entities) und Spalten (Attribute).

ANSI/SPARC-Modell: Siehe [1, 2.0-1]. Ein Architekturmodell: eine Empfehlung für die Architektur von DB-Systemen. Unterscheidet 3 Ebenen von Benutzersichten, damit jeder Benutzer die Daten so sehen kann wie er sie braucht:

externe Sicht für Endbenutzer und Anwendungsprogrammierer

konzeptionelle Sicht für Datenbankplaner

interne Sicht für Datenbankverwalter

Hierarchisches Modell: Ein Datenmodell. Sieht die Daten abhängig von ihrem Typ in einer Hierarchie geordnet. [1, 4.2-1]

Netzwerk-Modell: Ein Datenmodell. Sieht die Daten als Knoten eines beliebigen gerichteten Graphen, die Beziehungen als Kanten dieses Graphen. [1, 4.3-2]

9.15 Graphische Darstellung des konzeptionellen Modells

9.15.1 Aufgabenstellung

Nennen Sie eine gängige graphische Darstellungsform zur Modellierung der konzeptionellen Ebene nach dem ANSI/SPARC-Modell! Welche graphischen Grundelemente kennt diese Darstellungsform?

9.15.2 Lösung

Das Entity-Relationship-Modell (ERM) erlaubt die formale Modellbildung zur Erstellung des konzeptionellen Modells. Dargestellt durch Entity-Relationship-Diagramme mit folgenden graphischen Grundelementen:

Objekttyp engl. »entity type«. Typ und gleichzeitig Menge von Entitäten mit gleichen Eigenschaften. Entitäten sind Modelle realer Objekte. Darstellungsform: Rechteck.

Beziehungstyp engl. »relationship type«. Typ und gleichzeitig Menge von Beziehungen zwischen Entitäten. Darstellungsform: Raute auf einer stumpfwinkligen Spitze stehend.

Kante Zuordnung von Objekttypen und Beziehungstypen. Darstellungsform: Linie. In vereinfachten Darstellungen gibt man Beziehungstypen nicht explizit an, so dass dann einfach Objekttypen durch Linien verbunden werden. Diese Linien enthalten nach moderner Notation auch keine kleine unbeschriftete Raute!

Attribut: Eigenschaft einer Entität. Darstellungsform: Im Objekttyp.

9.16 Invertierte Liste in ADABAS

9.16.1 Aufgabenstellung

Erläutern Sie Aufbau, Funktion und Eigenschaften einer invertierten Liste beim DBVS ADABAS. Skizzieren Sie dazu ein Beispiel!

9.16.2 Lösung

Zur eindeutigen Identifikation eines Datensatzes auf physikalischer Ebene verwendet ADABAS einen internen künstlichen Primärschlüssel, die ISN (internal sequence number). Für jeden Deskriptor wird eine invertierte Liste angelegt, die das Suchen unterstützt und optimiert. Sie wird auf dem Massenspeicher ähnlich B*-Bäumen abgelegt und ist logisch eine nach dem Deskriptorwert (»DE-Wert«) geordnete Tabelle der Struktur: (DE-Wert, Anzahl des Auftretens, Liste ISNs). Dabei ist die »Liste ISNs« jeweils aufsteigend sortiert.

Beispiel:

ISN	MatrNr	Name	Vorname	Alter
1	1001	Meyer	Elke	17
2	1003	Müller	Hans	19
3	4711	Meyer	Gustav	18
4	0003	Schulz	Herbert	19
5	1312	Sommer	Ulrike	16
6	1102	Müller	Susanne	21
7	2103	Meyer	Inge	19

Die invertierte Liste für den Deskriptor »Name«:

DE-Wert	Anzahl	Liste ISNs
Meyer	3	1, 3, 7
Müller	2	2, 6
Schulz	1	4
Sommer	1	5

9.17 Anforderungen an postrelationale Datenmodelle

9.17.1 Aufgabenstellung

Welches sind nach allgemeiner Überzeugung Anforderungen an künftige postrelationale Datenmodelle als Grundlage für Datenbankverwaltungssysteme? Kreuzen Sie an!

9.17.2 Lösung

Siehe [1, 4.5-1 bis 4.5-2].

×	Möglichkeit zur Darstellung von BLOBs
×	Verzicht auf die erste Normalform der Relationentheorie
	Integration eines C-Compilers in das DBVS
×	Unterstützung der Versionsverwaltung für Daten
	Verbot von Listen als Datenfelder
	Aufhebung des Zwangs zur Konsistenz zwischen Index- und Datenbereich der DB
	Verzicht auf Modellierung von $m : n$-Beziehungen
×	Zulassen geschachtelter Relationen
×?	erhöhte Flexibilität bei der Beschreibung von Entities (variable Satzbeschreibung)
×	Unterstützung benutzerdefinierter Datentypen

9.18 Relationale Datenbank?

9.18.1 Aufgabenstellung

Ein Hersteller bietet ein von ihm vermarktetes DBVS als »Relationale Datenbank« an. Was sagen Sie dazu, wenn Sie die Einhaltung exakter fachlicher Terminologie fordern?

9.18.2 Lösung

Er verkauft keine »Datenbank«, d.i. eine Sammlung von Daten, sondern ein System zur Verwaltung von Datenbanken, das DBVS. Er verkauft kein »relationales DBVS«, sondern ein DBVS, das das relationale Datenmodell[3] verwendet.

9.19 Adressumsetzung in ADABAS

9.19.1 Aufgabenstellung

Welche Datenstruktur beim DBVS ADABAS dient der Umsetzung zwischen logischen und physikalischen Adressen von Datensätzen? Kreuzen Sie genau eine Lösung an!

9.19.2 Lösung

Siehe [1, 5.7-3].

	invertierte Liste
	Assoziator
	Datenspeicher
	Logfile
×	Adresskonverter
	Nucleus
	Search Buffer

9.20 Gründe für die Überführung in die 1. Normalform

9.20.1 Aufgabenstellung

Was sind die Gründe dafür, einen DB-Entwurf in die 1. Normalform zu überführen? Kreuzen Sie an!

9.20.2 Lösung

Siehe [1, 4.4-6 bis 4.4-7].

[3] Es heißt nicht »relationales Datenbankmodell«, denn es werden Daten modelliert, nicht Datenbanken! Siehe [1, 4.4-1].

×	Eliminierung von Redundanz
×	Vermeidung von Anomalien
×	Speicherplatzersparnis
	Erhöhung der Performance
	Anpassung an das ERD
	objektorientierter Ansatz beim DB-Entwurf
	einfachere Struktur mit "flachen" Tabellen
	die Attribute in 1.NF ändern sich während der Existenz des Tupels nicht
×	um nach der Umformung auch nicht elementare Attribute zulassen zu können

9.21 Ordnung im Relationenmodell

9.21.1 Aufgabenstellung

Ein Nachteil der Datenmodellierung nach dem Relationenmodell ist das Fehlen einer Ordnung (Reihenfolge) für die Tupel einer Relation. Wie kann man dennoch im Relationenmodell Ordnung ausdrücken?

9.21.2 Lösung

Indem man ein Attribut einführt, das als eindeutiger Schlüssel verwendet werden kann und dessen Werte in Beziehung »größer« oder »kleiner« zueinander stehen.

9.22 Aufgaben eines Datenbankadministrators

9.22.1 Aufgabenstellung

Welches sind originäre Aufgaben eines Datenbankadministrators beim betrieblichen Einsatz einer DB? Kreuzen Sie an!

9.22.2 Lösung

Siehe [1, 2.0-1].

×	Einrichten der Datenbank
	regelmäßiges Reinigen der Datenbank mit einem antistatischen Tuch
×	Optimierung von dynamischen Parametern zur Verbesserung der Performance
×	Durchführen von Datensicherungsmaßnahmen
	Einspielen neuer Versionen des Betriebssystems
	Überwachung der Einhaltung der Arbeitszeit im Rechenzentrum
	Überwachung der Netzwerkverbindungen
×	regelmäßige Überwachung DBVS-interner Datenstrukturen auf bevorstehenden Überlauf
×	Einspielen neuer Versionen des DBVS

9.23 Bezeichnungen beim relationalen Datenmodell

9.23.1 Aufgabenstellung

Wie bezeichnet man beim relationalen Datenmodell:

1. die Auswahl einer Zeile
2. die Auswahl einer Spalte
3. die Verknüpfung einer Tabelle mit sich selbst
4. die Umformung einer konzeptionellen Datenbeschreibung derart, daß Tabellen nur noch einfache, unstrukturierte Attribute enthalten
5. ein Attribut, für welches ein bestimmter Wert nur ein einziges Mal in einer Relation vorkommen kann

9.23.2 Lösung

1. Selektion
2. Projektion
3. self join
4. Überführung in die 1. Normalform
5. eindeutiger Schlüssel, unique key

9.24 NF^2-Datenmodelle

9.24.1 Aufgabenstellung

Gelegentlich ist von NF^2-Datenmodellen die Rede.

1. Was bedeutet diese Abkürzung
2. Was versteht man darunter? Welche Vorteile bieten sie?

9.24.2 Lösung

Siehe [1, 4.4-3].

1. Non-First-Normal-Form
2. NF^2-Datenmodelle verlangen nicht die Einhaltung der 1. Normalform, im Gegensatz zum streng relationalen Datenmodell. Sie erlauben also Mengen als Attributwerte, d.h. multiple Felder. Anders formuliert: Sie erlauben geschachtelte Relationen, denn Attribute mit Mengen als Werten sind selbst Relationen. Vorteil: Problemlösungen werden oft einfacher; es wird ein Vorzug des Netzwerkmodell bzw. hierarchischen Datenmodells integriert.

9.25 Warum Transaktionsdauer begrenzen? (Textantwort)

9.25.1 Aufgabenstellung

Welchen Grund gibt es dafür, bei einem DBVS die maximal zulässige Dauer einer Transaktion zu begrenzen?

9.25.2 Lösung

Siehe [1, 6.2-5]. Damit es unmöglich ist, dass Datenbankteile gesperrt bleiben und so andere Benutzer damit nicht mehr arbeiten können. Langes Sperren von Datenbankteilen kann auftreten durch:

- Dialog-Benutzer, die mit dem Abschluss einer Transaktion trödeln.
- Absturz der DB-Applikation, so dass der Server kein »end of transaction« erhält.

9.26 Datensicherung und backout transaction

9.26.1 Aufgabenstellung

1. Skizzieren und erläutern Sie den Inhalt der zur Datensicherung benutzten Datei »log-file« bzw. »journal-file«!
2. Wie ist der Ablauf bei der Operation »backout transaction«?

9.26.2 Lösung

1. Inhalt der Protokolldatei (»log file«, auch »journal file«) (siehe [1, 6.4-2]):

   before-image-journal Protokollierung des Zustandes *vor* der Änderung für alle in einer Transaktion erfolgenden Änderungen von Objekten. Struktur:

   (a) Marke für Transaktionsbeginn, mit Transaktions-ID
   (b) Kopie jedes geänderten Objektes (meist Datensätze als Einheit) *vor* der Änderung, mit Transaktions-ID, Objektidentifikation und Inhalt.
   (c) Marke für Transaktionsende, mit Transaktions-ID

   after-image-journal Protokollierung des Zustandes *nach* der Änderung für alle in einer Transaktion erfolgten Änderungen von Objekten. Struktur:

   (a) Marke für Transaktionsbeginn, mit Transaktions-ID
   (b) Kopie jedes geänderten Objektes (meist Datensätze als Einheit) *nach* der Änderung, mit Transaktions-ID, Objektidentifikation und Inhalt.
   (c) Marke für Transaktionsende, mit Transaktions-ID

2. Die Operation »Rücksetzen einer Transaktion«(siehe [1, 6.4-3]): alle Änderungen seit Beginn der Transaktion rückgängig machen, um die Konsistenz der DB wieder herzustellen.

   (a) Ermitteln der Transaktions-ID der rückzusetzenden Transaktion.
   (b) Das before-image-journal der Protokolldatei rückwärts lesen.
   (c) Für jeden Eintrag mit passender Transaktions-ID den Inhalt in die Datenbank zurückschreiben.
   (d) Ende wenn die »Marke für Transaktionsbeginn« mit der passenden Transaktions-ID erreicht wird.

9.27 Sortierte und ausgeglichene B-Baum-Varianten

9.27.1 Aufgabenstellung

Als Zugriffspfade werden oft Varianten von B-Bäumen eingesetzt. Diese werden in der Literatur als »sortierte Schlüsselbäume« und als »ausgeglichene« Bäume bezeichnet. Erläutern Sie, was mit den Bezeichnungen »sortiert« und »ausgeglichen« gemeint ist! Mit welchen Maßnahmen wird der Zustand »ausgeglichen« jederzeit gewährleistet?

9.27.2 Lösung

Siehe [1, 5.4-4 bis 5.4-5]. B-Bäume *sind* »sortierte Schlüsselbäume«, denn sie enthalten Schlüssel in sortierter Form, d.h. geordnet. Ein B-Baum ist »ausgeglichen«, wenn sichergestellt ist, dass alle seine Knoten immer zu einem gewissen Grad gefüllt sind. Das wird bei Bedarf durch Maßnahmen zum »Ausgleich« sichergestellt:

- Ist der Knoten zu gering gefüllt, wird ein Eintrag vom Nachbarn geholt oder der Knoten wird mit anderen Knoten zusammengelegt (melting).
- Bei Überlauf wird der Knoten und der neue Eintrag gleichmäßig in zwei Knoten geteilt (splitting).

9.28 Reaktion auf Überschreiten der maximalen Transaktionsdauer

9.28.1 Aufgabenstellung

Was passiert, wenn die maximal zulässige Dauer einer Transaktion überschritten wird? Gehen Sie auf die Reaktion des Anwendungsprogrammes und des DBVS ein!

9.28.2 Lösung

Siehe [1, 6.2-3]. Eine Transaktion muss »ganz oder gar nicht« ausgeführt werden. Bei Abbruch einer Transaktion muss das DBVS die Datenbank also in den Zustand vor Beginn der Transaktion zurückversetzen. Dazu werden alle Änderungen an Daten mit Hilfe der Protokolldatei rückgängig gemacht. Das Anwendungsprogramm erhält vom DBVS eine Fehlermeldung und wird den Programmablauf entsprechend ändern. Es kann die Fehlermeldung dem Endbenutzer z.B. in einem gesonderten Dialogfenster anzeigen.

9.29 Remote Database Access

9.29.1 Aufgabenstellung

Was verstehen Sie unter »Remote Database Access«? Erläutern Sie anhand einer Skizze! Gehen Sie auch darauf ein, ob dieser Begriff etwas mit CSA zu tun hat!

9.29.2 Lösung

Siehe [1, 6.3-3]. »Remote Database Access« ist eine mögliche Art von Client-Server-Architektur. Dabei übernimmt jeder Client Präsentation und Verarbeitung, der Server die Datenhaltung. Die Kommunikation über eine Systemgrenze geschieht durch »Remote Procedure Calls« durch den Client und entsprechende Antworten vom Server. Skizze siehe [1, 6.3-3].

9.30 Aussagen über B-Bäume

9.30.1 Aufgabenstellung

Welche Aussagen über B-Bäume sind im Zusammenhang mit DB richtig? Kreuzen Sie an!

9.30.2 Lösung

Siehe [1, 5.4-2 bis 5.4-5].

	B-Bäume wurden speziell für die Verwaltung großer Datenmengen im ASP entwickelt
×	B-Bäume sind sortierte Schlüsselbäume
	B-Bäume sind ausgeglichen
×	Die Höhe eines Baumes kann zeitlich varieren
	Die Länge der Wege von der Wurzel zu einem Blatt kann zum selben Zeitpunkt geringfügig varieren
	Die Knoten eines B-Baumes sollten möglichst eine variable Blocklänge haben
	Ein Knoten kann eine beliebige Zahl Nachfolger haben
×	Ein Knoten eines B-Baumes kann eine variable Zahl von Schlüsselwerten bzw. Dateneinheiten aufnehmen
×	Mit B-Bäumen realisierte Zugriffspfade müssen zur Erhaltung kurzer Zugriffszeiten von Zeit zu Zeit reorganisier

9.31 Funktionaltät eines Reportgenerators

9.31.1 Aufgabenstellung

Was ist typische Funktionalität eines im Zusammenhang mit einem DBS stehenden Reportgenerators? Kreuzen Sie an!

9.31.2 Lösung

Siehe [1, 9.3-7].

×	Tabellenausgabe mit Spaltenüberschriften
	Auflistung von Informationen über Datenstrukturen
	Erstellung von Berichten über die DB-Benutzung
×	Unterstützung Gruppenwechselfunktionen
	Werkzeug zur Unterstützung der Systemanalyse
×	statistische Zusammenfassungen der enthaltenen Daten
	Generierung von Standardprogrammen
	automatische Generierung von Testdaten
×	automatische Zeilennummerierung

9.32 Warum Transaktionsdauer begrenzen? (Multiple Choice)

9.32.1 Aufgabenstellung

Welchen Grund bzw. welche Gründe gibt es dafür, bei einem DBVS die maximal zulässige Dauer eine Transaktion zu begrenzen? Kreuzen Sie an!

9.32.2 Lösung

Siehe [1, 6.2-5].

×	Reduzierung möglicher Wartezeiten bei der Simultanarbeit von Benutzern
	Vermeidung von Deadlock-Situationen
	Reduzierung des CPU-Zeit-Verbrauchs
	Reduzierung der Programmlaufzeiten
	Reduzierung der Dauer temporärer Inkonsistenzen
×	Verkürzung der mittleren Dauer einer Transaktion

9.33 verteilte Verarbeitung

9.33.1 Aufgabenstellung

Was versteht man im Zusammenhang mit Client-Server-DB-Konzepten unter »verteilter Verarbeitung«? Erläutern Sie anhand einer Skizze!

9.33.2 Lösung

Siehe, auch für die Skizze: [1, 6.3-3]. »Verteilte Verarbeitung« ist eine mögliche Client-Server-Architektur. Dabei übernimmt jeder Client Präsentation und einen Teil der Verarbeitung, der Server den anderen Teil der Verarbeitung und die Datenhaltung. Die Verarbeitung ist also auf mehrere Rechner verteilt, die zusammenarbeiten müssen; deshalb spricht man auch von »kopperativer Verarbeitung«.

9.34 Massenspeicherorientierte Realisierung von Zugriffspfaden

9.34.1 Aufgabenstellung

Welche Realisierungen von Zugriffspfaden spielen bei DBVS mit Ablage der Daten auf Massenspeichern eine wichtige Rolle? Kreuzen Sie an!

9.34.2 Lösung

Siehe [1, 5.4-2].

×	Digitalbäume
×	Hash-Verfahren
	sequentielle Zugriffspfade
×	B-Bäume
	C-Bäume
	systematische Suche

9.35 Wann padding factor groß wählen?

9.35.1 Aufgabenstellung

Welches sind Kriterien den »padding factor« für eine DB-Datei groß zu wählen? Kreuzen Sie an!

9.35.2 Lösung

Siehe [1, 5.4-3].

	Es handelt sich um »Stamm-Daten«
×	Es handelt sich um »Bewegungs-Daten«
	Der Plattenspeicher hat eine große physikalische Blockgröße
	Der Plattenspeicher hat eine besonders kurze Zugriffszeit
	Es steht sehr viel Arbeitsspeicher zur Verfügung
	Es ist noch sehr viel Platz auf dem Massenspeicher vorhanden
×	Man erwartet häufiges Löschen und Neueinfügen von Datensätzen
	Man erwartet häufige Änderung von Datums- und Uhrzeit-Angaben in den Sätzen

9.36 Varianten des Transaktionsabbruchs

9.36.1 Aufgabenstellung

Manchmal muß eine begonnene Transaktion trotz stabil laufenden DBVS abgebrochen werden. Nennen und erläutern Sie 2 von verschiedenen Instanzen angestoßene Varianten des Abbruchs!

9.36.2 Lösung

Siehe [1, 6.2-3 und 6.2-5].

- Wunsch des Benutzers. Etwa wenn der Benutzer bemerkt, dass er weitere Daten erkundigen muss, um diese Transaktion durchzuführen. Ausgedrückt durch eine entsprechende Aktion im Dialogfeld. Das Anwendungsprogramm fordert daraufhin den Abbruch der Transaktion beim DBVS an (»rollback«).
- Abbruch durch das DBVS. Etwa aufgrund »time-out«, d.h. wenn eine Transaktion die maximale Transaktionsdauer überschritten hat. So verhindert das DBVS, dass Teile der Datenbank lange Zeit gesperrt bleiben und die Arbeit anderer Benutzer so behindert wird.

9.37 Warum nur eine Retrieval-Anweisung in SQL?

9.37.1 Aufgabenstellung

Was sind die Gründe, daß man sich im Sprachumfang von SQL auf nur eine einzige Anweisung zum Retrieval von Daten beschränkt hat? Wie versucht man, die daraus in einigen Fällen entstehenden Performance-Nachteile zu beseitigen?

9.37.2 Lösung

Siehe [1, 9.6-1]: SQL ist als Datenankabfragesprache für den Endbenutzer gedacht, i.d.R. also für den EDV-Laien. Deshalb soll es eine einfache Sprache sein. Dazu soll sie »ergebnisorientiert« sein: der Endbenutzer soll sagen, »was« er machen will und nicht »wie« es zu erreichen ist. Die Beschränkung auf eine einzige Retrieval-Anweisung verfolgt dieses Ziel von Einfachheit und Ergebnisorientierung.

Siehe [1, 5.6-3]: Zur Beseitigung von Ineffizienten enthält das DBVS enthält einen Optimierer, der für jede Retrieval-Anweisung einen endgültigen und maschinell optimierten Abfrageplan erstellt.

9.38 Nicht abgeschlossene Transaktionen bei Wiederanlauf

9.38.1 Aufgabenstellung

Beim Neustart eines DBVS nach Systemzusammenbruch (z.B. durch Netzausfall) ist als Reaktion unter anderem die Wiederherstellung eines konsistenten Zustandes der Datenbasis erforderlich. Hierzu zählt die Behandlung nicht abgeschlossener Transaktionen. Wie erkennt das DBVS solche nicht abgeschlossene Transaktionen und welche Handlungen führt es im Detail aus?

9.38.2 Lösung

Siehe [1, 6.4-3 bis 6.4-4]. Transaktionen, zu denen es im Logfile eine Marke »begin of transaction« gibt, aber keine Marke »end of transaction«, werden daran vom DBVS als nicht abgeschlossen erkannt. Alle nicht abgeschlossenen Transaktionen werden zurückgesetzt (undo recovery). Das bedeutet für jede Transaktion: das »before-image-journal« des Logfile wird rückwärts gelesen und für jedes Objekt mit passender Transaktions-ID wird der hier protokollierte alte Inhalt in die Datenbank zurückgeschrieben. Eine Transaktion ist vollständig zurückgesetzt, wenn die Marke »begin of transaction« gelesen wird.

9.39 Gründe für Client-Server-Architektur

9.39.1 Aufgabenstellung

Was sind Gründe für die Ablösung Host-basierter Systeme durch CSA? Kreuzen Sie an!

9.39.2 Lösung

Siehe [1, 6.3-3].

×	Verteilung der Rechenleistung auf Knoten in einem Netzwerk
×	Verfügbarkeit preiswerter Arbeitsplatzrechner (PC/Workstation)
	Wunsch nach graphischen Bedienungsoberflächen für Applikationsprogramme
×	Daten aus der DB müssen auch räumlich entfernt verfügbar sein
	Verzicht auf Middleware
×	Anpassung an dezentrale Organisationsstrukturen
	große Datenmengen
	höhere Betriebssicherheit und Verfügbarkeit
	CSA sind strukturell einfachere Systeme

9.40 Nachteile von Hash-Verfahren

9.40.1 Aufgabenstellung

Welche Nachteile hat die Realisierung von Zugriffspfaden mit Hash-Verfahren gegenüber Varianten von B*-Bäumen?

9.40.2 Lösung

Siehe [1, 5.4-8].

- Der Suchaufwand im »worst case« kann bei Hashverfahren sehr hoch sein, während er bei B*-Bäumen weit weniger variiert. Deshalb sind Hash-Verfahren die beste Wahl, wenn nur eine möglichst kurze durchschnittliche Suchzeit gefordert ist. Sie sind aber den B*-Bäumen unterlegen, wenn eine maximale Suchzeit garantiert sein muss.
- Nachträgliche Vergrößerung des Adressraums ist aufwändig (»re-hashing«), während B*-Bäume nie »aus Platzmangel« reorganisiert werden müssen.

9.41 Das before image journal

9.41.1 Aufgabenstellung

Was versteht man im Zusammenhang mit Datensicherungskonzepten unter einem »before image journal«? Was enthält es und wozu wird es benutzt?

9.41.2 Lösung

Siehe [1, 6.4-2]. Das »before image journal« ist Bestandteil des »log file«, also des Protokolls der DB-Änderung. Es enthält eine Kopie der durch eine Transaktion geänderten Datenobjekte *vor* ihrer Änderung. Zu jedem kopierten Datenobjekt wird die ID der zugehörigen Transaktion abgespeichert, zu jeder Transaktion Marken für Beginn und Ende. Das »before image journal« wird für undo-Operationen verwendet, etwa zum Rücksetzen nicht abgeschlossener Transaktionen. Dazu wird es rückwärts gelesen und für jedes veränderte Objekt der alte Inhalt vom Logfile in die DB zurückgeschrieben, bis man auf die Marke »begin of transaction« trifft.

9.42 Transaktionsendeerkennung durch DBVS?

9.42.1 Aufgabenstellung

Bei der Programmierung von ändernden DB-Zugriffen muss der Programmierer das Ende einer Transaktion angeben. Warum kann der Kern des DBVS nicht selbst erkennen, wann eine Transaktion zu Ende ist? Antworten Sie kurz!

9.42.2 Lösung

Siehe: [1, 6.2-2 und 6.2-6]. Eine Transaktion überführt die Datenbank zwischen zwei konsistenten Zuständen. Konsistenz meint »anwendungsbezogene logische Richtigkeit der Daten«. Anwendungsbezogen impliziert, dass das DBVS nicht alle Integritätsbedingungen kennt, damit Konsistenz und damit das Ende einer Transaktion nicht selbst erkennen kann.

9.43 Das after image journal

9.43.1 Aufgabenstellung

Was versteht man im Zusammenhang mit Datensicherungskonzepten unter einem »after image journal«? Was enthält es und wozu wird es benutzt?

9.43.2 Lösung

Siehe [1, 6.4-2]. Das »after image journal« ist Bestandteil des »log file«, also des Protokolls der DB-Änderung. Es enthält eine Kopie der durch eine Transaktion geänderten Datenobjekte *nach* ihrer Änderung. Zu jedem kopierten Datenobjekt wird die ID der zugehörigen Transaktion abgespeichert, zu jeder Transaktion Marken für Beginn und Ende. Das »after image journal« wird für redo-Operationen verwendet, etwa bei der Rekonstruktion einer zerstörten Datenbank zum »Nachfahren« aller abgeschlossenen Transaktionen seit der letzten Sicherungskopie.

9.44 Was ist der padding factor?

9.44.1 Aufgabenstellung

Erläutern Sie den Begriff »padding factor«!

9.44.2 Lösung

Siehe [1, 5.4-3]. »padding factor« bezeichnet den initial freien Platz eines B-Baum-Knotens in Prozent der Blocklänge. Er wird beim Einrichten der DB vom DBA festgelegt, typische Werte liegen zwischen 10% und 50%. Ein größerer padding factor senkt die Wahrscheinlichkeit für das aufwendige »block-splitting«.

9.45 Entwicklung und Diskussion der Client-Server-Architektur

9.45.1 Aufgabenstellung

Datenbanksysteme mit Client-Server-Architekturen, insbesondere heterogene CSA, bieten heute eine Reihe von wohlbekannten Vorzügen gegenüber Host-basierten Systemen.

1. Welche technologischen Entwicklungen haben zur derzeitigen Aktualität geführt?
2. Nennen Sie Vorzüge und mögliche Nachteile von CSA!

9.45.2 Lösung

Siehe [1, 6.3-3 bis 6.3-4].

1.
   - Verfügbarkeit preiswerter Arbeitsplatzrechner mit Netzwerkanbindung (Workstations und PCs)
   - weltweite Vernetzung
   - sinkende Datenübertragungskosten aufgrund der weltweiten Vernetzung
2. Vorteile:
   - Anpassung an dezentrale Organisationsstrukturen der Anwender
   - höhere lokale Verfügbarkeit
   - schnellerer lokaler Zugriff
   - Wegfall von Datenübertragungskosten, wenn Daten lokal verfügbar sind

   Nachteile:
   - potentielle Integritätsprobleme
   - ggf. Leitungskosten für Daten und zum Ablauf der Kommunikation
   - höherer Systemadministrationsaufwand
   - ggf. geringere Zuverlässigkeit

9.46 Was ist referentielle Integrität?

9.46.1 Aufgabenstellung

Was verstehen Sie unter »referentieller Integrität«? Geben Sie eine kurze Erklärung!

9.46.2 Lösung

Die »referentielle Integrität« ist ein Bestandteil der interrelationalen semantischen Integrität, der vom DBVS gewährleistet werden kann. In SQL wird ein Attribut dazu als Fremdschlüssel ausgezeichnet und der referenzierte Schlüssel angegeben. »Referentielle Integrität« verlangt: der Fremdschlüssel darf einen Wert nur annehmen, wenn ein Tupel existiert, in dem der referenzierte Schlüssel diesen Wert hat.

9.47 Transaktion und Notwendigkeit von Transaktionslogik

9.47.1 Aufgabenstellung

Wieso ist ein »relationaler« DB-Server zur Gewährleistung der jederzeitigen Datenkonsistenz darauf angewiesen, dass die Anwendungsprogramme konsequente Transaktionslogik einhalten? Erläutern Sie hierbei auch den Begriff »Transaktion« und die zu treffenden Maßnahmen im Anwendungsprogramm!

9.47.2 Lösung

Siehe [1, 6.2-2]. Eine Transaktion ist »die kleinste Einheit von Datenveränderungen, die grundsätzlich vollständig ausgeführt sein müssen, damit die anwendungsbezogene logische Richtigkeit der gesamten Daten erhalten bleibt«. Ein Anwendungsprogramm hält die Transaktionslogik ein, indem es alle Transaktionen als solche auszeichnet. Geschieht dies nicht, kann das DBVS die Konsistenz der Daten nicht gewährleisten, weil es nicht alle Integritätsbedingungen dafür kennt. Diese sind ja »anwendungsbezogen«: sie gehören zum Anwendungsprogramm und sind oft gar nicht mit den Mitteln der Datenbeschreibungssprache, die das DBVS versteht, ausdrückbar.

9.48 Maximale Einträge eines $(2, 2)$ B*-Baums

9.48.1 Aufgabenstellung

Wieviele verschiedene Einträge kann ein als B*-Baum der Klasse $(2, 2)$ organisierter Schlüsselbaum maximal enthalten? Die Ermittlung des Ergebnisses ist durch eine Skizze zu belegen!

9.48.2 Lösung

Siehe [1, 5.4-5 bis 5.4-6]. (k, h) bedeutet: $p = 2k$ Speicherplätze je Knoten, Höhe h des Baums (d.i. Länge des längsten Weges von der Wurzel zu einem Blatt, angegeben in Kanten und um 1 geringer als die Stufenzahl s). Einträge sind Datensätze und daher nur in Blattknoten zu finden. Ein B*-Baum enthält mehr Schlüssel als Einträge, da manche Schlüssel mehrfach vorkommen, nämlich auch vor Zeigern in Nicht-Blattknoten. Die Zahlen in Abbildung 6 sind Schlüssel von Einträgen, keine Nummerierung der Plätze für Schlüssel!

$$\begin{aligned} (2,2) &\Rightarrow k = 2,\ h = 2 \\ &\Rightarrow p = 2k = 4,\ s = h + 1 = 3 \end{aligned}$$

Damit ist die maximale Anzahl der Einträge: $(2k)^{h+1} = 4^3 = 64$. Siehe Abbildung 6.

9.49 Auswahl eines DBVS per tpmC?

9.49.1 Aufgabenstellung

Ohne sich schon auf eine Hardware-Plattform festgelegt zu haben, geben Sie vor der geplanten Beschaffung eines DBVS einen Benchmark zweier DBVS in Auftrag. Ihnen werden folgende Ergebnisse mitgeteilt: DBVS A: 1800 tpmC, DBVS B: 3000 tpmC.

1. Was bedeutet die Angabe »tpmC«?
2. Ist das Ergebnis für Sie aussagekräftig genug, um eine Entscheidung für die Beschaffung eines der DBVS zu treffen? Falls nein, welche weitere Angabe müssen Sie mindestens haben?

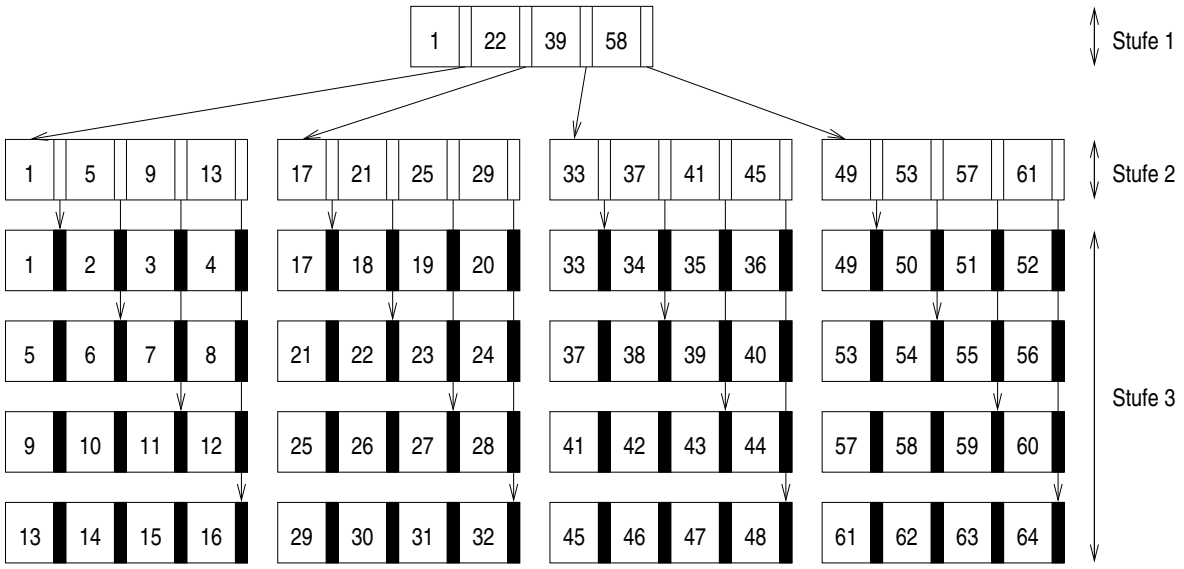

Abbildung 6: Zur Aufgabe »Maximale Einträge eines $(2, 2)$ B*-Baums«

9.49.2 Lösung

1. »tpmC« bedeutet: Transaktionen pro Minute gemessen nach der Methode »C«
2. Das Ergebnis ist hardwareabhängig, die Hardware-Plattform ist jedoch noch nicht festgelegt. Daher muss das Ergebnis noch hardwareunabhängig in »Kosten pro Transaktion« ausgedrückt werden, um eine fundierte Entscheidung der Beschaffung zu ermöglichen.

9.50 Wozu before image journaling?

9.50.1 Aufgabenstellung

Was wird durch das »before image journaling« unterstützt? Kreuzen Sie an!

9.50.2 Lösung

Siehe [1, 6.4-2 bis 6.4-5].

×	Rücksetzen einer begonnenen, aber nicht abgeschlossenen Transaktion
	Rekonstruktion der Datenbasis nach Plattencrash
×	Neustart nach Zusammenbruch
	Doppelführung (shadowing)
	Setzen von Checkpoints
×	rollback
	rollforward

9.51 Bau eines $(100, 2)$ B*-Baums

9.51.1 Aufgabenstellung

Bestimmen sie zu einem B*-Baum der Klasse $(100, 2)$ die Stufenzahl, die Anzahl der Schlüsseleinträge je Knoten und die Anzahl der Schlüsseleinträge im gesamten Baum!

9.51.2 Lösung

(k, h) bedeutet: $p = 2k$ Speicherplätze je Knoten, Höhe h des Baums (d.i. Länge des längsten Weges von der Wurzel zu einem Blatt, angegeben in Kanten und um 1 geringer als die Stufenzahl s).

$$\begin{aligned}(100, 2) &\Rightarrow k = 100,\ h = 2 \\ &\Rightarrow s = h + 1 = 3,\ p = 2k = 200\end{aligned}$$

Damit ist die maximale Anzahl der Einträge: $\sum_{i=1}^{h+1} (2k)^i = 200^1 + 200^2 + 200^3 = 8.040.200$.

9.52 Mögliche Konsequenz eines block splitting

9.52.1 Aufgabenstellung

Zu was kann ein »block splitting« führen? Kreuzen Sie an!

9.52.2 Lösung

	führt immer zu einer Erhöhung der Stufenzahl
×	kann zu einer Erhöhung der Stufenzahl führen
	kann zu einer Erhöhung der Anzahl möglicher Schlüsseleinträge je Knoten führen

9.53 Einsatzgebiete von EmbeddedSQL

9.53.1 Aufgabenstellung

Welches sind sinnvolle Einsatzgebiete für »Embedded SQL«? Kreuzen Sie an!

9.53.2 Lösung

Siehe [1, 9.4-1].

×	Anwendungen, in denen SQL nicht nach außen in Erscheinung treten soll
×	Anwendungen mit wenigen DB-Zugriffen und komplexer Anwendungslogik
×?	Wenn eine Kapselung der Programmiersprache vor dem Anwender geboten ist
×	Migration von COBOL-Altanwendungen mit bisheriger Datenhaltung im Dateisystem
	Anwendungen mit Einbettung in eine graphische Benutzeroberfläche
	Neuanwendungen, bei denen eine Codierung in 4GL zu aufwendig wäre

9.54 Aussagen zu Client-Server-Architektur

9.54.1 Aufgabenstellung

Welche Aussagen zu CSA sind richtig? Kreuzen Sie an!

9.54.2 Lösung

Siehe [1, 6.3-3].

	Sie bedeuten immer die Trennung der Präsentation (Client) von der Verarbeitungslogik und der Datenhaltung (
×	Der Server-Rechner wird entlastet
	Es werden Clients mit graphischen Benutzeroberflächen vorausgesetzt
×	Zur Wirtschaftlichkeit werden preiswerte Clients vorausgesetzt
	Sie reduzieren die Komplexität von Systemen
	Middleware-Komponenten sind nur auf dem Server erforderlich

9.55 Rücksetzen einer Transaktion

9.55.1 Aufgabenstellung

Nach einem »time-out« wegen Überschreitens der maximal zulässigen Dauer einer Transaktion soll eine begonnene, aber noch nicht abgeschlossene Transaktion zurückgesetzt werden.

1. Worin liegt der Sinn dieses Abbruchs einer Transaktion bei Überschreitung einer maximal zulässigen Dauer?

2. Was läuft dabei im Datenbank-Kern ab?

9.55.2 Lösung

1. Siehe [1, 6.2-5]. So verhindert das DBVS, dass Teile der Datenbank lange Zeit gesperrt bleiben und so die parallele Arbeit anderer Benutzer, die mit diesen Daten arbeiten wollen, behindert wird. Ein »time-out« macht also nur im Mehrbenutzerbetrieb wirklich Sinn. Ein »time-out« wird verursacht, wenn Benutzer mit dem Abschließen einer Transaktion trödeln oder wenn eine DB-Applikation abstürzt und deshalb das DBVS kein »end of transaction« erhält.

2. Siehe [1, 6.4-3]. Verfahren, mit dem das DBVS eine nicht beendete Transaktion rücksetzt:

   (a) Fehlermeldung »time-out« an das Anwendungsprogramm senden.

   (b) Transaktions-ID der rückzusetzenden Transaktion bestimmen.

   (c) Das »before-image-journal« des Logfile rückwärts lesen und für jedes Objekt mit passender Transaktions-ID den hier verzeichneten alten Inhalt in die Datenbank zurückgschreiben.

   (d) Die Transaktion ist vollständig rückgesetzt, wenn im »before image journal« die Marke »begin of transaction« gelesen wird.

   (e) Alle durch die nun rückgesetzte Transaktion gesperrten Objekte freigeben.

9.56 ANSI/SPARC-Achitekturmodell

9.56.1 Aufgabenstellung

Quelle: [6, 1999-SS :: Aufg. 8]. Skizzieren Sie das ANSI/SPARC-Architekturmodell!

9.56.2 Lösung

Siehe [1, 2.0-1]. Das ANSI/SPARC-Architekturmodell ist eine Empfehlung für die DB-System-Architektur. Damit jeder Benutzer die Daten sieht, wie er sie braucht und die Daten selbst maximal unabhängig von irgendeiner speziellen Darstellungsart sind, unterscheidet es drei Sichten:

interne Sicht für den DB-Administrator. Die Sicht auf die physikalische Organisation der Daten: Speicherungsform, Zugriffsmöglichkeiten ...

konzeptionelle Sicht für den DB-Planer. Die »logische Gesamtsicht« der Daten. Sie ist ein »Datenmodell« der Realität, geschrieben in der DDL des jeweiligen DBVS.

externe Sicht für den Endbenutzer und Anwendungsprogrammierer. Gruppe spezieller Benutzersichten, jede eingeschränkt auf den jeweils relevanten und / oder zulässigen Ausschnitt der Gesamtdaten.

9.57 Abkürzungen

9.57.1 Aufgabenstellung

Quelle: [6, 1999-SS :: Aufg. 1]. Nachfolgend stehen einige gängige Abkürzungen aus der Terminologie zu Datenbanksystemen. Für was stehen sie jeweils?

9.57.2 Lösung

Siehe [9, S. 1].

DBMS database management system

4GL 4th generation language

DDL data definition language

OQL object query language

ODBC open database connectivity

ACID atomicity, consistency, isolation, durability

9.58 Tabelle für Zeitschriftenaufsätze umformen

9.58.1 Aufgabenstellung

Quelle: [6, 1999-SS :: Aufg. 11]. Gegeben sei der Entwurf einer Tabelle »Zeitschriftenaufsaetze«:

Feldname	Datentyp(Länge)
TitelAufsatz	varchar(80)
NameZeitschrift	varchar(60)
Band	dec(3)
SeiteAnfang	dec(4)
Jahrgang	dec(4)
NameAutor1	varchar(40)
NameAutor2	varchar(40)
NameAutor3	varchar(40)

1. Überführen Sie diesen Tabellenentwurf in die 1 . Normalform.
2. Formulieren Sie eine SQL-Abfrage nach Zeitschriftenaufsätzen mit Beteiligung eines Autors 'Meyer':
   (a) für den Originalentwurf
   (b) für den in die 1. NF überführten Entwurf.

9.58.2 Lösung

1. Tabelle »Zeitschriftenaufsaetze«

Feldname	Datentyp(Länge)
AufsatzID	int(4)
TitelAufsatz	varchar(80)
NameZeitschrift	varchar(60)
Band	dec(3)
SeiteAnfang	dec(4)
Jahrgang	dec(4)

Tabelle »ZA«

Feldname	Datentyp(Länge)
AufsatzID	int(4)
AutorID	int(4)

Tabelle »Autoren«

Feldname	Datentyp(Länge)
AutorID	int(4)
AutorName	varchar(40)

2.

(a)

```
SELECT TitelAufsatz, NameZeitschrift, Jahrgang, Band, SeiteAnfang
  FROM Zeitschriftenaufsaetze
  WHERE NameAutor1 = 'Meyer'
    OR NameAutor2 = 'Meyer'
    OR NameAutor3 = 'Meyer'
```

(b)

```
SELECT TitelAufsatz, NameZeitschrift, Jahrgang, Band, SeiteAnfang
  FROM Zeitschriftenaufsaetze ZAufs
    INNER JOIN ZA ON ZAufs.AufsatzID = ZA.AufsatzID
    INNER JOIN Autoren ON ZA.AutorID = Autoren.AutorID
  WHERE AutorName = 'Meyer'
```

9.59 DB-System einer Bank

9.59.1 Aufgabenstellung

Quelle: [6, 1999-SS :: Aufg. 12]. Eine bundesweit operierende Bank möchte Daten über Ihre Kunden und deren Konten in einer Datenbank speichern. Es ergeben sich folgende Forderungen:

- Kunden sind regional zuständigen Niederlassungen zugeordnet, welche jeweils Zweigstellen unterhalten. Ein Kunde kann mehrere Konten bei der Bank haben.
- Aufbau der (numerischen) Konto-Nr.:
  - 2-stellige Niederlassungs-Nr.
  - 6-stellige Kunden-Stamm-Nr. (unternehmensweit eindeutig)
  - 2-stellige fortlaufende Nr.
- Zu einem Kunden sollen alle für die Geschäftsbeziehung erforderlichen Daten gespeichert werden, hierzu zählen auch die Angabe eines Kreditlimits und des Beginns der Kundenbeziehung.
- Zu den Niederlassungen sind neben der Niederlassungs-Nr. der Ort und ein Statusfeld (Zeichenkette, Länge 1) zu speichern.

Selbstverständlich können Sie die »reale Welt« so weit vereinfacht modellieren, wie dies die gestellte Aufgabe zulässt. Aufgaben:

1. Zeichnen Sie zu dieser Anwendung ein Entity-Relationship-Diagramm, welches unmittelbar mit einer »relationalen« Datenbank implementierbar ist.
2. Erstellen Sie SQL-Anweisungen zur Erzeugung aller Tabellen. Geben Sie hierbei auch Primärschlüssel, eindeutige Schlüssel sowie Fremdschlüsselbeziehungen an!

Anmerkungen:

- Gehen Sie von den Gestaltungsmöglichkeiten aus, die Ihnen das DBVS MS SQL Server bietet (s. hierzu auch Arbeitsunterlage).
- Eindeutige und Primär-Schlüssel können auch aus einer Kombinationen mehrerer Spalten gebildet werden.
- Ein Fremdschlüssel muss in der fremden Tabelle Primärschlüssel oder eindeutiger Schlüssel sein!

9.59.2 Lösung

9.60 Informationssystem einer Hotelkette

9.60.1 Aufgabenstellung

Quelle: [6, 1999-SS :: Aufg. 13]. Das Informationssystem einer großen internationalen Hotelkette habe (vereinfacht) folgende Tabellen mit den angegebenen Feldern (DBVS MS SQL Server): siehe Abbildung 7.

Anmerkungen:

- In der Tabelle Zimmer ist die Kombination von H_ID und Z_ID eindeutig.
- Das Feld Dat_Abreise in der Tabelle Belegung wird bei der Anreise des Gastes, soweit noch unbekannt, mit dem Datum 31.12.9999 besetzt.

Erstellen Sie zu nachfolgenden Aufgabenstellungen in Standard-SQL codierte Programme:

1. Ändern der falsch erfaßten Bezeichnung des Hotels "Tandreahs" in Giessen auf "Tandreas".
2. Ist heute, am 12.7.1999, im Hotel mit der H_ID "TGI" ein Gast des Namens "Eva Ford" eingetragen? Wenn ja, welches Zimmer belegt sie und zu welchem Preis.

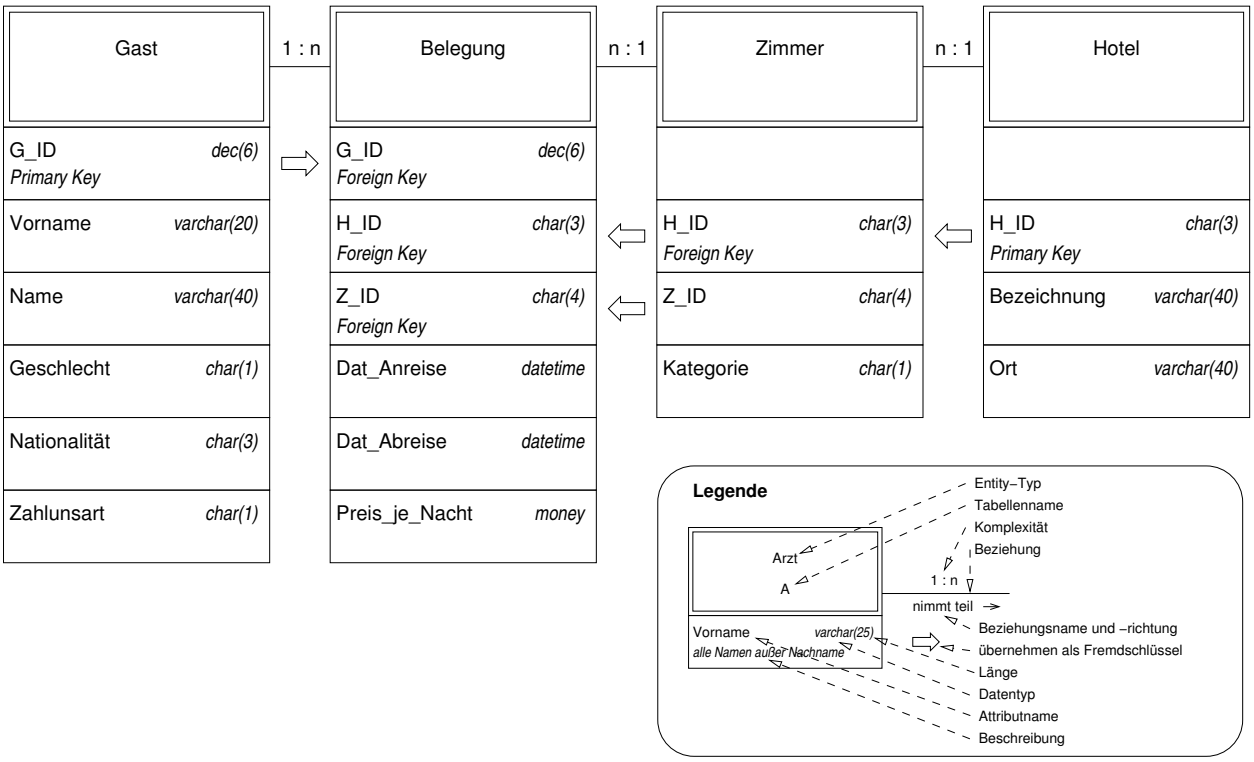

Abbildung 7: Zu Aufgabe »Informationssystem einer Hotelkette«

3. Erstellen Sie eine nach Name und Vorname alphabetisch sortierte Liste aller weiblichen Gäste der Hotelkette, welche nicht deutscher Nationalität (Codierung mit KFZ-Länderkennzeichen) sind.

4. Gibt es heute (am 12.7.1999) ein freies Zimmer im Hotel mit der H_ID "TGI"? Wenn ja, wie ist die Zimmer-Nr. und die Kategorie? (Hinweis: Denken Sie an "between"!)

5. Wieviele Gäste haben welche Zahlungsart benutzt?

6. Erstellen Sie eine nach Namen alphabetisch sortierte Liste aller Gäste der Hotelkette, welche mehr als 5 Aufenthalte hatten.

7. Nach einer Umorganisation soll geprüft werden, ob im gespeicherten Datenbestand die referentielle Integrität zwischen den Tabellen Gast und Belegung gewährleistet ist. Die beanstandeten Zeilen sind vollständig auszugeben.

8. Erstellen Sie eine Liste aller Hotels mit ihren Zimmern, welche auch solche Hotels beinhaltet, zu denen noch kein Zimmer gespeichert wurde. Die Liste soll nach H_ID des Hotels und Z_ID des Zimmers sortiert sein und folgende Felder beinhalten: H_ID, Bezeichnung, Ort, Z_ID, Kategorie.

9.60.2 Lösung

9.61 DB-System einer Universität

9.61.1 Aufgabenstellung

Quelle: [2, A-DES 20]. Eine Universität möchte Daten zum Lehrbetrieb in einer Datenbank speichern. In einer Vorbesprechung ergeben sich folgende Anforderungen:

Professoren halten Vorlesungen, Assistenten sind jeweils einem Professor zugeordnet. Sie betreuen Übungen, welche (ausschließlich) in Verbindung mit einer Vorlesung stattfinden. Studierende hören Vorlesungen und nehmen an Übungen teil.

1. Zeichnen Sie zu dieser Anwendung ein Entity-Relationship-Diagramm, welches unmittelbar mit einer »relationalen« Datenbank implementierbar ist.

2. Geben Sie an, zu welchen Entity-Typen (Tabellen) Sie welche Identifikationsschlüssel vorsehen und wie Sie die Beziehungen zwischen den Entity-Typen (Tabellen) realisieren.
   Ebenfalls im ERD enthalten.

3. Erstellen Sie SQL-Anweisungen zur Erzeugung aller Tabellen. Geben Sie hierbei auch Primärschlüssel, eindeutige Schlüssel und Fremdschlüsselbeziehungen an!

Hinweis: Selbstverständlich können Sie die »reale Welt« so weit vereinfacht modellieren, wie dies die gestellte Aufgabe zulässt.

9.61.2 Lösung

1. Entity-Relationship-Diagramm

2. Identifikationsschlüssel und Beziehungen

3. SQL-Anweisungen zur Erzeugung aller Tabellen

```
CREATE TABLE Professor (
  Pers_ID decimal(4) PRIMARY KEY
)

CREATE TABLE Vorlesung (
  VlfdNr         decimal(6) PRIMARY KEY,
  UlfdNr         decimal(6),
  Pers_ID        decimal(6),
  Semesterbeginn datetime,
  VeranstgAbk    char(6),
  FOREIGN KEY (Pers_ID) REFERENCES Professor(Pers_ID),
  FOREIGN KEY (UlfdNr) REFERENCES Uebung(UlfdNr)
)

CREATE TABLE VorlBesuch (
  MatrNr decimal(6),
  VlfdNr decimal(6),
  FOREIGN KEY (MatrNr) REFERENCES Student(MatrNr),
  FOREIGN KEY (VlfdNr) REFERENCES Vorlesung(VlfdNr)
)

CREATE TABLE Assistent (
  Pers_ID decimal(4),
  Prof_ID decimal(4),
  FOREIGN KEY (Prof_ID) REFERENCES Professor(Prof_ID)
)

CREATE TABLE Student (
  MatrNr decimal(6) PRIMARY KEY
)

CREATE TABLE Uebung (
  UlfdNr decimal(6) PRIMARY KEY,
  VlfdNr decimal(6),
  Pers_ID decimal(6),
  Semesterbeginn datetime,
  VeranstgAbk char(6),
  FOREIGN KEY (VlfdNr) REFERENCES Vorlesung(VlfdNr),
  FOREIGN KEY (Pers_ID) REFERENCES Assistent(Pers_ID)
)
```

9.62 Übung zur Datenspeicherung

9.62.1 Aufgabenstellung

Quelle: [2, A-DES 61]. Übung zur Datenspeicherung. Eine Tabelle »Schüler« hat folgende Feldbeschreibungstabelle:

Schüler-Nr	decimal(4)
Jahrgang	decimal(2)
Begabtenförderung	varchar(2)
Vorname_1	varchar(20)
Vorname_2	varchar(20)
Vorname_3	varchar(20)
Nachname	varchar(40)
Geschlechtscode	char(1)

Beim Laden (Import) aus einer sequentiellen Datei mit Sätzen fester Länge enthalte ein Eingabedatensatz folgende Daten:

0312	00		Hans	Otto		Rost	M

1. Wie groß ist die Satzlänge des Eingabedatensatzes?
2. Wie groß ist die Satzlänge des in der Datenbanktabelle gespeicherten Satzes?
3. Welcher Kompressionsfaktor wurde durch Ansatz des Datentyps »varchar« bei geeigneten Feldern erreicht (Verhältnis unkomprimierte / komprimierte Satzlänge)?
4. Überlegen Sie den Arbeitsablauf im Kern des DBVS für den Datenbank-Zugriff: »Ändere in dem gespeicherten Satz den dritten Vornamen auf den Wert 'August'«.

9.62.2 Lösung

1. Satzlänge des Eingabedatensatzes

$$4byte + 2byte + 2byte + 20byte + 20byte + 20byte + 40byte + 1byte = 109byte$$

2. Satzlänge des in der Datenbanktabelle gespeicherten Satzes
   Wir gehen davon aus, dass eine Stelle im »packed decimal« Datentyp mit $4bit$ dargestellt wird. Zur Satzlänge gehören auch Informationen über die Feldlängen. Weil wir das dafür benutzte Verfahren nicht kennen (vorangestellte Länge, Begrenzungssymbol oder Zeiger), bleiben diese Informationen unberücksichtigt.

$$4 \cdot 4bit + 2 \cdot 4bit + 4byte + 4byte + 4byte + 1byte = 16byte$$

3. Kompressionsfaktor
   Die unkomprimierte Satzlänge beträgt:

$$4 \cdot 4bit + 2 \cdot 4bit + 2byte + 20byte + 20byte + 20byte + 40byte + 1byte = 106byte$$

   Der gegebene Eingabedatensatz hat $16byte$ komprimierte Satzlänge (siehe vorige Teilaufgabe). Der Kompressionsfaktor beträgt damit:

$$f = \frac{106Byte}{16Byte} = 6,625$$

4. Arbeitsablauf im Kern des DBVS
   (a) Bisherigen Satz lesen, dabei dekomprimieren. Der Satz steht nun mit fester Satzlänge im Hauptspeicher.
   (b) Aus einem Zeigersystem den Beginn und die Länge des Feldes für den dritten Vornamen ermitteln.
   (c) Prüfen, ob das Feld lang genug ist für den hineinzuschreibenden Wert, ansonsten Fehlermeldung.
   (d) Das Feld initialisieren mit SQL NULL (wer weiß was vorher darin stand?)
   (e) Die Zeichenkette 'August' in dieses Feld schreiben.
   (f) Eintrag im before image journal.

(g) Den bisherigen Satz löschen. Dies geschieht zuerst nur im Puffer und wirkt sich später auf den Sekundärspeicher aus.

(h) Den neuen Satz komprimieren und schreiben. Dies geschieht zuerst nur im Puffer und wirkt sich später auf den Sekundärspeicher aus.

(i) Eintrag im after image journal.

9.63 Was sind Normalformen?

9.63.1 Aufgabenstellung

Definieren Sie in einem Satz: Was sind Normalformen?

9.63.2 Lösung

Normalformen sind eine definierte Vorgehensweise, um Redundanz im konzeptionellen Entwurf eines DBS zu erkennen und zu eliminieren.

9.64 Elemente von SQL

9.64.1 Aufgabenstellung

Nennen Sie die Namen der Befehle von SQL, gruppiert in die drei in SQL enthaltenen Sprachen.

9.64.2 Lösung

DDL Datendefinitionssprache

- CREATE
- ALTER
- DROP

DML Datenmanipulationssprache

- SELECT
- UPDATE
- INSERT
- DELETE

DCL Datensteuerungssprache

- GRANT
- REVOKE
- LOCK
- COMMIT
- ROLLBACK

Programmierspracheneinbettung

- DECLARE CURSOR
- OPEN
- CLOSE
- FETCH

9.65 Referentielle Integrität prüfen

9.65.1 Aufgabenstellung

Nennen Sie zwei Alternativen, wie man die referentielle Integrität zwischen zwei Tabellen mit SQL prüfen kann!

9.65.2 Lösung

Erste Alternative:

```
SELECT Kd#&Konto#
  FROM Kontobewegung
  WHERE Kd#&Konto# NOT IN (SELECT Kd#&Konto# FROM Konto)
```

Zweite Alternative:

```
SELECT Kd#, Konto#
  FROM Kontobewegung
    LEFT OUTER JOIN Konto ON Kontobewegung.Kd# = Konto.Kd# AND Kontobewegung.Konto# = Konto.Konto#
  WHERE Konto.Kd# IS NULL AND Konto.Kd# IS NULL
```

9.66 Suppliers-Tabelle erzeugen

9.66.1 Aufgabenstellung

Quelle: [2, A-SQL 1]. Erzeuge Tabelle »S« mit spezifizierten Datenfeldern.

9.66.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
CREATE TABLE S (
  S#     varchar(3)   PRIMARY KEY,
  SName  varchar(12),
  Status smallint     DEFAULT 0,
  City   varchar(12)
)
```

9.67 Suppliers-Tabelle ändern

9.67.1 Aufgabenstellung

Quelle: [2, A-SQL 2]. Ändere Tabelle »S« durch Hinzufügen eines neuen Feldes.

9.67.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
ALTER TABLE S ADD COLUMN Street varchar(12)
```

9.68 Index für Feld City

9.68.1 Aufgabenstellung

Quelle: [2, A-SQL 3]. Erzeuge Index für das Feld »City«.

9.68.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
CREATE INDEX (City) ON S
```

9.69 Tabelle Suppliers löschen

9.69.1 Aufgabenstellung

Quelle: [2, A-SQL 4]. Lösche Tabelle »S«.

9.69.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
DROP TABLE S
```

9.70 Alle Lieferanten in Paris, absteigend nach Status

9.70.1 Aufgabenstellung

Quelle: [2, A-SQL 9]. Zeige Liefranten-Nr. und Status für alle Lieferanten in Paris, sortiert in absteigender Reihenfolge nach Status.

9.70.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT S#, Status
  FROM Suppliers
  WHERE City = 'Paris'
  ORDER BY Status DESC
```

9.71 Paare von Lieferanten aus derselben Stadt

9.71.1 Aufgabenstellung

Quelle: [2, A-SQL 14]. Zeige alle möglichen Paare von Lieferanten-Nr., bei denen die beiden Lieferanten aus derselben Stadt sind.

9.71.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT S1.S#, S2.S#
  FROM S S1 INNER JOIN S S2 ON S1.City = S2.City
```

9.72 Alle Lieferanten, die P2 liefern

9.72.1 Aufgabenstellung

Quelle: [2, A-SQL 15]. Zeige den Lieferanten-Namen für alle Lieferanten, die das Teil »P2« liefern.

9.72.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT SName
  FROM S
    INNER JOIN SP ON S.S# = SP.S#
    INNER JOIN P ON SP.P# = P.P#
  WHERE PName = 'P2'
```

9.73 Jeweilige Gesamtmenge aller Teile

9.73.1 Aufgabenstellung

Quelle: [2, A-SQL 20]. Zeige die jeweilige Gesamtmenge für alle gelieferten Teile.

9.73.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT P#, sum(QTY)
  FROM SP
  GROUP BY P#
```

9.74 Alle Teile von mehr als einem Lieferanten

9.74.1 Aufgabenstellung

Quelle: [2, A-SQL 21]. Zeige die Teile-Nr. für alle Teile, die von mehr als einem Lieferanten geliefert werden.

9.74.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT P#
  FROM SP
  GROUP BY P#
  HAVING count(DISTINCT S#)>1
```

9.75 Status des Lieferanten S2 ändern

9.75.1 Aufgabenstellung

Quelle: [2, A-SQL 24]. Ändere den Status des Lieferanten »S2« auf »50«.

9.75.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
UPDATE S
  SET Status=50
  WHERE SName='S2'
```

9.76 Lieferanten S4 löschen

9.76.1 Aufgabenstellung

Quelle: [2, A-SQL 26]. Lösche den Lieferanten »S4«.

9.76.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
DELETE
  FROM S
  WHERE SName='S4'
```

9.77 Liste aller Bücher zu Datenbanken ab 1994

9.77.1 Aufgabenstellung

Quelle [2, A-SQL 31]. Erstellen Sie eine Liste aller Bücher mit dem Schlagwort »Datenbank«, welche 1994 oder später erschienen sind. Die Liste soll folgende Spalten enthalten: Lit_ID, Titel, Verlag, Erscheinungsort, Erscheinungsjahr (abgekürzte Spaltenüberschrift!), ISBN. Sortieren Sie alphabetisch nach Titel!

9.77.2 Lösung

```
SELECT L_Lit.Lit_ID, Titel, Verlag, Erscheinungsort, Erscheinungsjahr AS Jahr,
    ISBN, Schlagwort
  FROM L_Lit
    INNER JOIN L_LS ON L_Lit.Lit_ID=L_LS.Lit_ID
    INNER JOIN L_Sch ON L_LS.Sch_ID=L_Sch.Sch_ID
  WHERE L_Sch.Schlagwort = 'Datenbank' AND Erscheinungsjahr >= '1994'
  ORDER BY Titel
```

9.78 Liste aller Bücher mit Schlagwort SQL

9.78.1 Aufgabenstellung

Quelle [2, A-SQL 32]. Erstellen Sie eine Liste aller Bücher mit dem Schlagwort »SQL«. Die Liste soll folgende Spalten enthalten: Lit_ID, Titel, Erscheinungsjahr (abgekürzte Spaltenüberschrift!), Name und Vorname Autor(en).

9.78.2 Lösung

```
SELECT L_Lit.Lit_ID, Titel, Erscheinungsjahr AS Jahr, Name, Vorname
  FROM L_Lit
    INNER JOIN L_LS ON L_Lit.Lit_ID = L_LS.Lit_ID
    INNER JOIN L_Sch ON L_LS.Sch_ID = L_Sch.Sch_ID
    INNER JOIN L_LA ON L_Lit.Lit_ID = L_LA.Lit_ID
    INNER JOIN L_Aut ON L_LA.Aut_ID = L_Aut.Aut_ID
  WHERE Schlagwort = 'SQL'
```

9.79 Liste aller Bücher mit Schlagwort Datenbank, sortiert nach Erscheinungsjahr

9.79.1 Aufgabenstellung

Quelle: [2, A-SQL 33]. Erstellen Sie eine Liste aller Bücher mit dem Schlagwort »DATENBANK«. Die Liste soll folgende Spalten enthalten: Lit_ID, Titel, Erscheinungsjahr (abgekürzte Spaltenüberschrift!), Name und Vorname Autor(en). Sie soll sortiert sein nach Erscheinungsjahr.

9.79.2 Lösung

```
SELECT L_Lit.Lit_ID, Titel, Erscheinungsjahr AS Jahr, Name, Vorname
  FROM L_Lit
    INNER JOIN L_LS ON L_Lit.Lit_ID = L_LS.Lit_ID
    INNER JOIN L_Sch ON L_LS.Sch_ID = L_Sch.Sch_ID
    INNER JOIN L_LA ON L_Lit.Lit_ID = L_LA.Lit_ID
    INNER JOIN L_Aut ON L_LA.Aut_ID = L_Aut.Aut_ID
  WHERE Schlagwort = 'DATENBANK'
  ORDER BY Erscheinungsjahr
```

9.80 Liste aller Bücher

9.80.1 Aufgabenstellung

Quelle: [2, A-SQL 34]. Erstellen Sie eine Lister aller Bücher. Die Liste soll folgende Spalten enthalten: Lit_ID, Titel, Aut_ID, Name und Vorname Autor(en), alternativ mit / ohne Schlagwort. Sie soll alphabetisch aufsteigend nach Titel sortiert sein. Warum treten einige Bücher in der Liste doppelt auf? Wie können Sie in der Alternative »ohne Ausgabe Schlagwort« die Zahl der doppelt auftretenden Bücher reduzieren?

9.80.2 Lösung

Bücher treten doppelt auf, wenn sie mehrere Autoren oder Schlagwort haben. Jede Kombination wird dann als einzelner Datensatz gelistet. Alternative »mit Ausgabe Schlagwort«:

```
SELECT L_Lit.Lit_ID, Titel, L_Aut.Aut_ID, Name, Vorname, Schlagwort
  FROM L_Lit
    INNER JOIN L_LS ON L_Lit.Lit_ID = L_LS.Lit_ID
    INNER JOIN L_Sch ON L_LS.Sch_ID = L_Sch.Sch_ID
    INNER JOIN L_LA ON L_Lit.Lit_ID = L_LA.Lit_ID
    INNER JOIN L_Aut ON L_LA.Aut_ID = L_Aut.Aut_ID
  ORDER BY Titel
```

Alternative »ohne Ausgabe Schlagwort«; doppelte Bücher wurden durch »`SELECT DISTINCT`« reduziert:

```
SELECT DISTINCT L_Lit.Lit_ID, Titel, L_Aut.Aut_ID, Name, Vorname
  FROM L_Lit
    INNER JOIN L_LS ON L_Lit.Lit_ID = L_LS.Lit_ID
    INNER JOIN L_Sch ON L_LS.Sch_ID = L_Sch.Sch_ID
    INNER JOIN L_LA ON L_Lit.Lit_ID = L_LA.Lit_ID
    INNER JOIN L_Aut ON L_LA.Aut_ID = L_Aut.Aut_ID
  ORDER BY Titel
```

9.81 Doppelt gespeicherter Autor

9.81.1 Aufgabenstellung

Quelle: [2, A-SQL 35]. Gibt es einen Autor oder eine Autorin, der oder die in 'L_Aut' doppelt gespeichert ist? Berücksichtigen Sie hierbei Name und Vorname! Wenn ja, geben Sie Name und Vorname zusammen mit der zugehörigen Aut_ID aus.

9.81.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT Aut1.Aut_ID, Aut1.Name, Aut1.Vorname
  FROM L_Aut Aut1
    INNER JOIN L_Aut Aut2 ON Aut1.Name = Aut2.Name AND Aut1.Vorname = Aut2.Vorname
```

9.82 Doppelt gespeichertes Schlagwort

9.82.1 Aufgabenstellung

Quelle: [2, A-SQL 36]. Gibt es ein Schlagwort, welches in 'L_Sch' doppelt gespeichert ist? Wenn ja, geben Sie es zusammen mit der zugehörigen Sch_ID aus!

9.82.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT Sch1.Sch_ID, Sch1.Schlagwort
  FROM L_Sch Sch1 INNER JOIN L_Sch Sch2 ON Sch1.Schlagwort = Sch2.Schlagwort
```

9.83 Verschiedene Bücher, Autoren und Schlagworte

9.83.1 Aufgabenstellung

Quelle: [2, A-SQL 37]. Wieviele verschiedene a) Bücher, b) Autoren und c) Schlagworte gibt es?

9.83.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT count(DiSTINCT ISBN)
  FROM L_Lit

SELECT count(DISTINCT Vorname&Name)
  FROM L_Aut

SELECT count(DISTINCT Schlagwort)
  FROM L_Sch
```

9.84 Verschiedene Verlage

9.84.1 Aufgabenstellung

Quelle: [2, A-SQL 38]. Wieviele verschiedene Verlage sind in den gespeicherten Literaturdaten enthalten?

9.84.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT count(DISTINCT Verlag)
  FROM L_Lit
```

9.85 Wieviele Bücher von welchem Verlag?

9.85.1 Aufgabenstellung

Quelle: [2, A-SQL 39]. Von welchem Verlag sind wieviele Bücher in den gespeicherten Literaturdaten enthalten?

9.85.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT Verlag, count(Lit_ID)
  FROM L_Lit
```

9.86 Statistik Bücher nach Erscheinungsjahr

9.86.1 Aufgabenstellung

Quelle: [2, A-SQL 40]. Erstellen Sie eine statistische Verteilung der Bücher nach Erscheinungsjahr.

9.86.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT Erscheinungsjahr, count(Lit_ID)
  FROM L_Lit
  ORDER BY Erscheinungsjahr
```

9.87 Wer schrieb wieviele Bücher?

9.87.1 Aufgabenstellung

Quelle: [2, A-SQL 41]. Erstellen Sie eine Liste, aus welcher hervorgeht, welcher Autor (Name und Vorname) wieviele Bücher geschrieben hat (auch Mit-Autorenschaft zählt).

9.87.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT Name, Vorname, count(Lit_ID)
  FROM L_Lit
    INNER JOIN L_LA ON L_Lit.Lit_ID = L_LA.Lit_ID
    INNER JOIN L_Aut ON L_Aut.Aut_ID = L_LA.Au_ID
  GROUP BY Name, Vorname
```

9.88 Bücher mit Titel »Client-Server. . . «

9.88.1 Aufgabenstellung

Quelle: [2, A-SQL 42]. Erstellen Sie eine Liste aller Bücher, bei denen der Titel mit der Zeichenkette 'Client-Server' beginnt. Die Liste soll folgende Spalten enthalten: Titel, Verlag, Erscheinungsort, Erscheinungsjahr. Die Schreibweise von 'Client-Server' soll hierbei verschiedene Variationen zulassen, z.B. mit Bindestrich, Schrägstrich oder Leerzeichen. Wie müsst die Anweisung ohne Verfügbarkeit des Operators 'like' heißen? (Anmerkung: Benutzen Sie den Operator *like*, Symbol '%' steht für kein oder beliebig viele Zeichen, Symbol '_' steht für genau ein Zeichen.)

9.88.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT *
  FROM L_Lit
  WHERE Titel LIKE('Client_Server%')
```

9.89 Bücher mit mindestens einem Schlagwort

9.89.1 Aufgabenstellung

Quelle: [2, A-SQL 49]. Wieviele Bücher gibt es, die durch mindestens ein Schlagwort beschrieben sind?

9.89.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT count(Lit_ID)
  FROM L_Lit
    INNER JOIN L_LS ON L_Lit.Lit_ID = L_LS.Lit_ID
    INNER JOIN L_Sch ON L_LS.Sch_ID = L_Sch.Sch_ID
```

9.90 Bücher ohne Schlagwort

9.90.1 Aufgabenstellung

Quelle: [2, A-SQL 50]. Wieviele Bücher gibt es, die durch kein Schlagwort beschrieben sind?

9.90.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT count(Lit_ID)
  FROM L_Lit
    LEFT OUTER JOIN L_LS ON L_Lit.Lit_ID = L_LS.Lit_ID
    LEFT OUTER JOIN L_Sch ON L_LS.Sch_ID = L_Sch.Sch_ID
  WHERE Schlagwort IS NULL
```

9.91 Referentielle Integrität zwischen Büchern und Schlagworten

9.91.1 Aufgabenstellung

Quelle: [2, A-SQL 51]. Gibt es in der Tabelle L_LS Zuordnungen zwischen einem Buch und einem Schlagwort, zu welchen es kein Buch in der Tabelle L_Lit gibt? (Prüfung auf »referentielle Integrität«)

9.91.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
SELECT *
  FROM L_LS
  WHERE Lit_ID NOT IN (SELECT Lit_ID FROM L_Lit)
```

9.92 Liste aller Kombinationen Abteilung - Mitarbeiter

9.92.1 Aufgabenstellung

Quelle: [2, A-SQL 61]. Erstellen Sie eine Liste mit allen möglichen Kombinationen zwischen Abteilungen und Mitarbeitern. Die Liste soll nach der Abkürzung des Abteilungs-Namens und innerhalb der Abteilung nach den Namen der Mitarbeiter alphabetisch sortiert sein.

9.92.2 Lösung

```
SELECT Abkuerzung, Name
  FROM F_Abt, F_Per
  ORDER BY Abkuerzung, Name
```

9.93 Daten aller Mitarbeiter und Abteilungen (Inner Join)

9.93.1 Aufgabenstellung

Quelle: [2, A-SQL 62]. Erstellen Sie eine Liste mit allen Daten aller Abteilungen und aller Mitarbeiter. Die Liste soll nach der Abkürzung des Abteilungs-Namens und innerhalb der Abteilung nach den Namen der Mitarbeiter alphabetisch sortiert sein. Die Liste soll nur diejenigen Mitarbeiter enthalten, welche zu einer Abteilung gehören, und nur diejenigen Abteilungen, welche Mitarbeiter haben.

9.93.2 Lösung

```
SELECT *
  FROM F_Abt INNER JOIN F_Per ON F_Abt.A_ID = F_Per.A_ID
  ORDER BY Abkuerzung, Name
```

9.94 Daten aller Mitarbeiter, ggf. mit Abteilungsdaten

9.94.1 Aufgabenstellung

Quelle: [2, A-SQL 63]. Erstellen Sie eine Liste mit allen Daten aller Mitarbeiter, welche auch die Bezeichung und den abgekürzten Namen der Abteilung enthält, zu welcher der jeweilige Mitarbeiter gehört. Die Liste soll nach der Abkürzung des Abteilungs-Namens und innerhalb der Abteilung nach den Namen der Mitarbeiter alphabetisch sortiert sein.

9.94.2 Lösung

```
SELECT Abkuerzung, Bezeichnung, F_Per.A_ID, P_ID, Name, Vorname,
    Geschlecht, Geburtsdatum
  FROM F_Per LEFT OUTER JOIN F_Abt ON F_Per.A_ID = F_Abt.A_ID
  ORDER BY Abkuerzung, Name
```

9.95 Daten aller Abteilungen, ggf. mit Mitarbeiterdaten

9.95.1 Aufgabenstellung

Quelle: [2, A-SQL 64]. Erstellen Sie eine Liste mit allen Daten aller Abteilungen, welche auch die Daten der Mitarbeiter enthält, die zu der jeweiligen Abteilung gehören. Die Liste soll nach der Abkürzung des Abteilungs-Namens und innerhalb der Abteilung nach den Namen der Mitarbeiter alphabetisch sortiert sein.

9.95.2 Lösung

```
SELECT *
  FROM F_Per RIGHT OUTER JOIN F_Abt ON F_Per.A_ID = F_Abt.A_ID
  ORDER BY Abkuerzung, Name
```

9.96 Daten aller Mitarbeiter und Abteilungen (Full Outer Join)

9.96.1 Aufgabenstellung

Quelle: [2, A-SQL 65]. Erstellen Sie eine Liste mit allen Daten aller Abteilungen und aller Mitarbeiter. Die Liste soll nach der Abkürzung des Abteilungs-Namens und innerhalb der Abteilung nach den Namen der Mitarbeiter alphabetisch sortiert sein. Die Liste soll auch Mitarbeiter enthalten, welche zu keiner Abteilung gehören sowie Abteilungen ohne Mitarbeiter.

9.96.2 Lösung

```
SELECT *
  FROM F_Per FULL OUTER JOIN F_Abt ON F_Abt.A_ID = F_Per.A_ID
  ORDER BY Abkuerzung, Name
```

9.97 Unterschiedliche Familiennamen des Personals

9.97.1 Aufgabenstellung

Quelle: [2, A-SQL 71]. Selektieren Sie aus der Tabelle Personal (F_Per) alle Männer und alle Frauen in jeweils eine neue Tabelle. Die Namen dieser Tabellen sollen mit dem Buchstaben 'M' bzw. 'F' beginnen und mit Ihrer Matrikel-Nr. fortgesetzt werden, z.B. 'F123456!'.

Erstellen Sie eine Liste der in den beiden Tabellen enthaltenen voneinander verschiedenen Namen (Familiennamen). Ermitteln Sie zum Vergleich die Anzahl der voneinander verschiedenen Namen in der ursprünglichen Tabelle!

9.97.2 Lösung

```
SELECT *
  INTO M123456
  FROM F_Per
  WHERE Geschlecht = 'M'

SELECT *
  INTO W123456
  FROM F_Per
  WHERE Geschlecht = 'W'

SELECT Name
  FROM M123456
UNION
SELECT Name
  FROM W123456

SELECT count(Name)
  FROM F_Per
```

9.98 Index und eindeutiger Index für Personal

9.98.1 Aufgabenstellung

Quelle: [2, A-SQL 72]. Erstellen Sie eine Kopie der Tabelle Personal (F_Per). Der Name dieser neuen Tabelle soll mit dem Buchstaben 'P' beginnen und mit Ihrer Matrikel-Nr. fortgesetzt werden, z.B. 'P123456'. Erstellen Sie für die Spalte »Geburtsdatum« der von Ihnen erzeugten Tabelle

1. einen Index,
2. einen Index mit eindeutigen Schlüsselwerten.

Welche Probleme treten auf, wenn Sie versuchen, einen Index mit eindeutigen Schlüsselwerten für die Spalte »Name« zu erzeugen?

9.98.2 Lösung

```
SELECT *
  INTO P678306
  FROM F_Per

CREATE INDEX P678306IDX
  ON P678306(Geburtsdatum)
```

```
CREATE UNIQUE INDEX P678306IDXU
  ON P678306(Geburtsdatum)

CREATE UNIQUE INDEX P678306IDXN
  ON P678306(Name)
```

Es ist nicht möglich, einen Index mit eindeutigen Schlüsselwerten für die Spalte »Name« zu erzeugen, weil es mehrere Datensätze mit gleichem Wert in »Name« gibt.

9.99 Neue Tabelle Praktikanten

9.99.1 Aufgabenstellung

Quelle: [2, A-SQL 73]. Erzeugen Sie zu den bestehenden DB-Tabellen »Firma« eine neue Tabelle »Praktikanten« mit den Spalten P_ID, A_ID, Name, Vorname, Eintrittsdatum, Austrittsdatum. Der Name dieser neuen Tabelle soll mit den Buchstaben 'PR' beginnen und mit ihrer Matrikel-Nr. fortgesetzt werden, z.B. 'PR123456'. Die Spalte 'A_ID' (Abteilungs-Identifikation) soll hierbei als Fremdschlüssel aus der Tabelle Abteilung (F_Abt) spezifiziert werden, um die referentielle Integrität zwischen den beiden Tabellen Praktikanten und Abteilun zu gewährleisten.

9.99.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
CREATE TABLE PR123456 (
  P_ID           dec(6) NOT NULL PRIMARY KEY,
  A_ID           dec(6),
  Name           varchar(20),
  Vorname        varchar(20),
  Eintrittsdatum datetime,
  Austrittsdatum datetime,
  FOREIGN KEY (A_ID) REFERENCES F_Abt(A_ID)
)
```

9.100 Eintrag lesen als Stored Procedure

9.100.1 Aufgabenstellung

Quelle: [2, A-SQL 74]. Prüfen Sie, ob Sie in der Tabelle »Teilnehmer_public« mit Ihrem Nmen eingetragen sind. Wenn ja, geben Sie die Spalten »Name« und »Vorname« aus. Lösen Sie diese Aufgabe durch eine »Stored Procedure«, welcher Sie beim Aufruf den Namen als Parameter übergeben.

9.100.2 Lösung

```
CREATE PROCEDURE ##binichteilnehmer
    @myName varchar, @myVorname varchar
  AS
    SELECT Name, Vorname
      FROM Teilnehmer_public
      WHERE Name=@myName AND Vorname=@myVorname

EXEC ##binichteilnehmer @myName='Ansorg', @myVorname='Matthias'
```

9.101 Trigger beim Speichern in Tabelle Praktikanten

9.101.1 Aufgabenstellung

Quelle: [2, A-SQL75]. Erstellen Sie einen Trigger, welcher beim Speichern eines neuen Datensatzes in die Tabelle Praktikanten (siehe Kapitel 9.99) eine Meldung ausgibt, gefolgt von einer Liste aller in der Tabelle »F_Abt«

enthaltenen A_ID's. Testen Sie den Trigger durch ein Programm zum Speichern eines neuen Datensatzes in »Praktikanten«.

9.101.2 Lösung

Diese Lösung wurde noch nicht getestet.

```
CREATE TRIGGER PR123456_save
  ON PR123456
  FOR INSERT
  AS
    SELECT A_ID
      FROM F_Abt
```

Literatur

[1] Prof. Dr. Volker Klement: »Datenbanksysteme I«; SS 2003. Das offizielle Skript der Vorlesung, verkauft in der Vorlesung für 2 EUR. Man kann ohne wesentliche Einbußen Skripte der vorigen Semester verwenden. Wer will, kann das Skript auch in digitaler Form bekommen. Dieses Skript enthält alle wichtigen Dinge, es fehlen jedoch Erläuterungen und Erklärungen aus der Vorlesung. Wer nur dieses Skript lernt, kann die Klausur bestehen, wohl aber mit geringerem Erfolg.

[2] Prof. Dr. Volker Klement: Übungsblätter zur Veranstaltung »Datenbanksysteme I«; SS 2003. Kostenlos verteilt in der Vorlesung. Bestehend aus den Teilen »Einrichtungen zum Praktikum (P-INS)«, »Regeln für Tabellenbeschreibungen (P-TAB)«, »SQL-Sprachumfang (Auszug) (P-SQL)«, »Aufgaben zur SQL-Programmierung (A-SQL)« und »Übungsaufgaben zum DB-Design (A-DES)«. Wer will, kann diese Übungsblätter auch in digitaler Form bekommen. Man sollte möglichst die aktuelle Ausgabe der Übungsblätter verwenden.

[3] Prof. Dr. Volker Klement: »Datenbanksysteme I; 1. Hausübung«; SS 2003. Ausgeteilt in Papierform in der Vorlesung Datenbanksysteme I.

[4] Prof. Dr. Volker Klement: »Datenbanksysteme I; 2. Hausübung«; SS 2003. Ausgeteilt in Papierform in der Vorlesung Datenbanksysteme I am 26. Mai 2003.

[5] Homepage von Prof. Dr. Volker Klement http://homepages.fh-giessen.de/~hg6331/. Direkter Link zum Teil »Datenbanksysteme I«: http://mm00.mni.fh-giessen.de/klement/db.html

[6] Klausuren in Datenbanksysteme I von SS 1999 bis WS 2002 im WinWord-Format. Auf [5] wird die Quelle `Dbserv::\DB\alte_Klausuren` angegeben. Zum Zugriff siehe Kapitel 8.3. Quelle im Internet Homepage von Matthias Ansorg http://matthias.ansorgs.de/InformatikDiplom/Modul.DbSys1.Klement/Klausuren/.

[7] Roger: »Klausurfragen:«; 52 Klausurfragen der Veranstaltung Datenbanksysteme 1 bei Prof. Dr. Volker Klement, FH Gießen-Friedberg, Studienort Gießen. Erstellt am 2000-02-25. Dateiname `DB-FAQ.DOC`, Größe 74752 Byte. Quelle: Homepage von Martin Müller http://homepages.fh-giessen.de/~hg11474/dateien/Datenbanken/DB-FAQ.DOC. Dieser Link ist nirgendwo auf der Homepage referenziert, sondern muss direkt angegeben werden! Dieses Dokument wurde vollständig in das vorliegende Dokument integriert, wird also zur Klausurvorbereitung nicht mehr benötigt. Die Antworten auf die Fragen wurden sehr weitgehend neuformuliert bzw. korrigiert und überarbeitet, so dass sich keine Probleme bzgl. des Urheberrechts ergeben.

[8] Tim Pommerening und Martin Müller: 16 Multiple-Choice-Klausurfragen zur Veranstaltung Datenbanksysteme 1 bei Prof. Dr. Volker Klement, FH Gießen-Friedberg, Studienort Gießen. Erstellt am 2002-06-13. Dateiname `Multiple_Choice.doc`, Größe 32256 Byte. Dieses Dokument ist lediglich ein Auszug der Multiple-Choice-Fragen aus [7] und daher zur Klausurvorbereitung nicht notwendig. Quelle: Homepage von Martin Müller http://homepages.fh-giessen.de/~hg11474/dateien/Datenbanken/Multiple_Choice.doc. Dieser Link ist nirgendwo auf der Homepage referenziert, sondern muss direkt angegeben werden!

[9] Roger, und Stephanie Weg: Kurz-Script V1.2 zur Veranstaltung Datenbanksystem 1 bei Prof. Dr. Volker Klement, FH Gießen-Friedberg, Studienort Gießen. Erstellt am 2001-07-01. Dateiname `Kurz-ScriptV1_2.DOC`, Größe 148480 Byte. Eine kompakte Zusammenstellung von klausurrelevantem Stoff aus [1], meist in eigener Formulierung. Enthält auch wertvolle Ergänzungen wie ein ausführliches Abkürzungsverzeichnis. Quelle: Homepage von Martin Müller http://homepages.fh-giessen.de/~hg11474/dateien/Datenbanken/Kurz-ScriptV1_2.DOC. Dieser Link ist nirgendwo auf der Homepage referenziert, sondern muss direkt angegeben werden!

[10] Frank Kirchner: »Datenbanken; Zusammenfassung Skript«. Zusammenfassung von [1] in 9 Seiten studentischer Mitschrift. zur Veranstaltung »Datenbanksysteme 1« bei Prof. Dr. Klement aus dem SS 1998 an der FH Gießen-Friedberg. Enthalten im Studienmaterial auf der Homepage von Andreas Ditze als http://awd.ods.org/studium/Klement-Datenbanken-SS98.zip, darin `Zusammenfassung_Script.doc`.

[11] Frank Kirchner: »Datenbanken; Zusammenfassung SQL-Commandos«. Vierseitige SQL-Kurzreferenz als studentische Mitschrift zur Veranstaltung »Datenbanksysteme 1« bei Prof. Dr. Klement aus dem SS 1998 an der FH Gießen-Friedberg. Enthalten im Studienmaterial auf der Homepage von Andreas Ditze als http://awd.ods.org/studium/Klement-Datenbanken-SS98.zip, darin `SQL.doc`.

[12] Referenz der SQL-Kommandos http://www.us.postgresql.org/users-lounge/docs/7.0/users/sql-commands.htm

[13] SQL-Tutorial http://www.w3schools.com/sql/default.asp