

Datenbanken (Quelle: Wikipedia)

Bedeutungen: [Wiktonary]

[1] [EDV](#): grosse Menge von Daten, die in einem Rechner nach bestimmten Kriterien organisiert komplexe Abfragen zulassen

[2] [allgemein](#): organisierte Sammlung von Daten, die nach verschiedenen Kriterien organisiert sind und abgerufen werden können

1 Einführung

Eine Datenbank, auch Datenbanksystem (DBS) genannt, ist ein System zur elektronischen Datenverwaltung. Die wesentliche Aufgabe eines DBS ist es, **grosse Datenmengen effizient, widerspruchsfrei und dauerhaft zu speichern** und benötigte Teilmengen in unterschiedlichen, bedarfsgerechten Darstellungsformen für Benutzer und Anwendungsprogramme bereitzustellen.

Ein DBS besteht aus zwei Teilen: der **Verwaltungssoftware**, genannt Datenbankmanagementsystem (DBMS) und der Menge der **zu verwaltenden Daten**, der Datenbank (DB) im engeren Sinn, zum Teil auch „Datenbasis“ genannt. Die Verwaltungssoftware organisiert intern die strukturierte Speicherung der Daten und kontrolliert alle lesenden und schreibenden Zugriffe auf die Datenbank. Zur Abfrage und Verwaltung der Daten bietet ein Datenbanksystem eine **Datenbanksprache** an.

Datenbanksysteme gibt es in verschiedenen Formen. Die Art und Weise, wie ein solches System Daten speichert und verwaltet, wird durch das Datenbankmodell festgelegt.

Zu unterscheiden ist der hier beschriebene Begriff der Datenbank (bestehend aus DBMS und Daten) von Datenbankanwendungen: Letzteres sind (häufig zur Anwendungssoftware gehörende) Computerprogramme, die ihre jeweils individuell erforderlichen Daten unter Nutzung eines DBS verwalten und speichern [Bre04].

2 Bedeutung

Datenbanksysteme sind heute ein zentraler Bestandteil der Unternehmenssoftware. Damit stellen sie einen kritischen Teil vieler Unternehmen und Behörden dar. Von der Verfügbarkeit, Vollständigkeit und Richtigkeit der Daten hängt die Aktionsfähigkeit eines Unternehmens ab. Die Datensicherheit ist daher ein wichtiger und gesetzlich vorgeschriebener Bestandteil der IT eines Unternehmens oder einer Behörde.

3 Komponenten eines Datenbanksystems

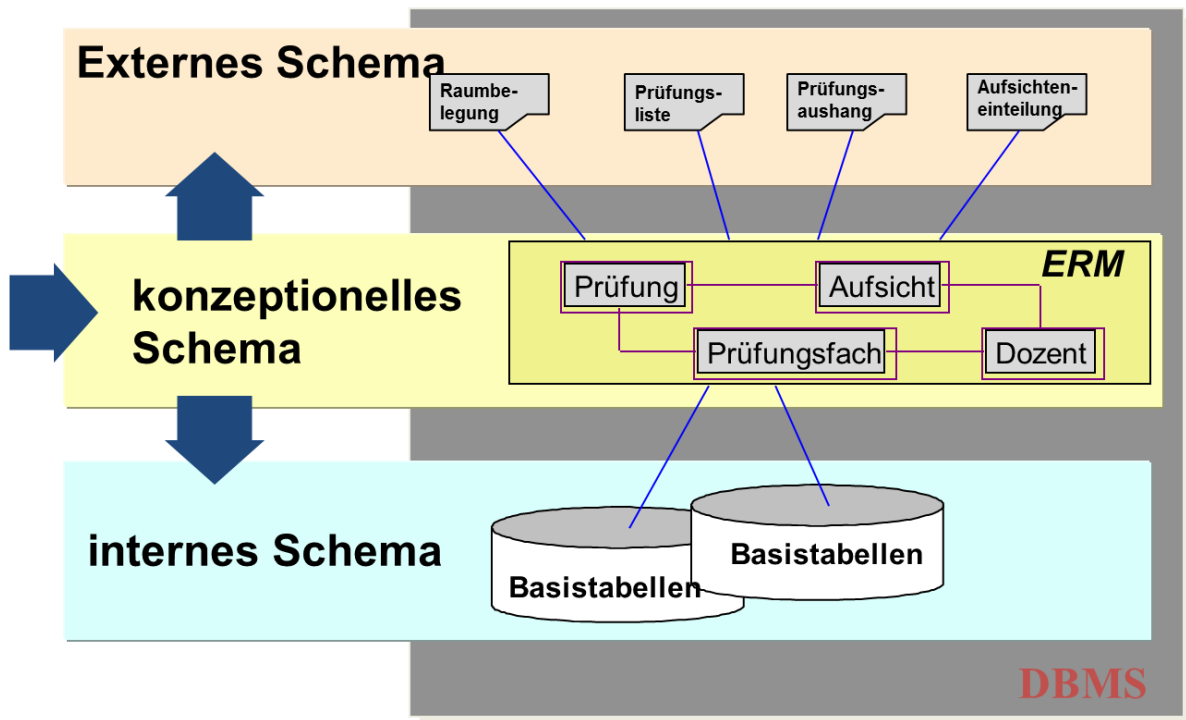
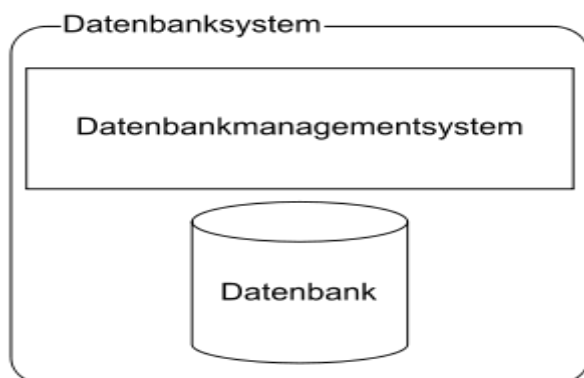


Abbildung 1: Schichtenmodell einer relationalen Datenbank

Das Datenbanksystem ist das ausgeführte DBMS zusammen mit den zu verwaltenden Daten der Datenbank. Ein DBS gewährleistet die persistente Speicherung sowie die Konsistenz der Nutzdaten einer Institution und bietet für die benutzenden Datenbankanwendungen mit dem DBMS Schnittstellen zur Abfrage, Auswertung, Veränderung und Verwaltung dieser Daten.



3.1 Datenbankmanagementsystem

Das Datenbankmanagementsystem (DBMS) ist die eingesetzte Software, die für das Datenbanksystem installiert und konfiguriert wird. Das DBMS legt das Datenbankmodell fest, hat einen Grossteil der unten angeführten Anforderungen zu sichern und entscheidet massgeblich über Funktionalität und Geschwindigkeit des Systems. Datenbankmanagementsysteme selbst sind hochkomplexe Softwaresysteme.

3.2 Datenbank

In der Theorie versteht man unter Datenbank (engl. database) einen logisch zusammengehörigen Datenbestand. Dieser Datenbestand wird von einem laufenden DBMS verwaltet und für Anwendungssysteme und Benutzer unsichtbar auf nichtflüchtigen Speichermedien abgelegt. Um einen effizienten Zugriff auf die Datenbank zu gewährleisten, verwaltet das DBMS in der Regel eine Speicherhierarchie, die insbesondere auch einen schnellen Zwischenspeicher (Pufferpool) umfasst. Zur Wahrung der Konsistenz des Datenbestandes müssen sich alle Anwendungssysteme an das DBMS wenden, um die Datenbank nutzen zu können. Allein administrativen Tätigkeiten, wie zum Beispiel der Datensicherung, ist der direkte Zugriff auf den Speicher erlaubt. Die logische Struktur der zu speichernden Daten wird bei der Datenmodellierung als Datenmodell erarbeitet und festgelegt.

3.3 Funktionen eines DBMS

Die wesentlichen Funktionen von heutigen Datenbankmanagementsystemen sind:

- Speicherung, Überschreibung und Löschung von Daten
- Verwaltung der Metadaten
- Vorkehrungen zur Datensicherheit
- Vorkehrungen zum Datenschutz
- Vorkehrungen zur Datenintegrität
- Ermöglichung des Mehrbenutzerbetriebs durch das Transaktionskonzept
- Optimierung von Abfragen
- Ermöglichung von Triggern und Stored Procedures
- Bereitstellung von Kennzahlen über Technik und Betrieb des DBMS

4 Anforderungen an eine Datenbank [MUN99]



Redundanzfreiheit Kein Datenelement soll in der Datenbank mehrfach gespeichert sein; der Speicherplatz ist optimal auszunutzen.

Vielfachverwendbarkeit (Anwendungsunabhängigkeit) Die Datenbank soll so aufgebaut sein, dass sie möglichst viele Funktionsbereiche bedienen kann. Jeder Benutzer, für den die gespeicherten Daten von Bedeutung sind, soll mit der Datenbank arbeiten können.



Benutzerfreundlichkeit Der Umgang mit der Datenbank soll leicht erlernbar sein. Mit möglichst geringem Aufwand sollen möglichst viele Funktionen des Datenbanksystems eingesetzt werden können

Flexibilität Es muss möglich sein, die Struktur der Datenbank, unabhängig von den bestehenden Anwendungen, zu ändern. Es sollten Backups im laufenden Betrieb möglich sein.





Unabhängigkeit von Betriebssystemen Die Datenbank sollte ein offenes System sein, d.h. das Wechseln von Hard- und/oder Software sollte keine Anpassungsschwierigkeiten mit sich bringen.

Wirtschaftlichkeit Das System sollte möglichst geringe Betriebskosten verursachen. Abfragen und Datenmanipulationen sollten mit möglichst wenigen Plattenzugriffen erfolgen.



Datenintegrität Datenschutz, Datensicherheit und Datenkonsistenz sollten vom System weitgehend unterstützt werden. (Beispiele: Wiederanfahrhilfen, Transaktionserkennung, Passcodeverwaltung, Rollback, Logbuch usw.)

5 Geschichte



Ausgehend von Problemen bei der Verarbeitung von Daten in einfachen Dateien wurde in den 1960er Jahren das Konzept eingeführt, Daten durch eine separate Softwareschicht zwischen Betriebssystem (Dateiverwaltung) und Anwendungsprogramm zu verwalten. Dieses Konzept begegnete der Fehlentwicklung, dass Datenspeicher in Form von Dateien in der Regel für eine spezielle Anwendung konzipiert wurden und ein erheblicher Teil des Tagesgeschäfts mit Umkopieren, Mischen und Restrukturieren der Dateien belastet war.

Eines der ersten grossen DBMS war IMS mit der Sprache DL/I (Data Language One). Die damit verwalteten Datenbanken waren hierarchisch strukturiert. Parallel dazu definierte CODASYL (**C**onference on **D**Ata **S**ystems **L**anguages ab 1959) ein Modell für netzwerkartig strukturierte Datenbanken.

Einen wesentlichen Fortschritt erzielte in den 1960er und 1970er Jahren **Edgar F. Codd** mit seiner Forschungsarbeit am IBM Almaden Research Center. Codd entwickelte die Grundlagen des ersten experimentellen relationalen Datenbanksystems System R. [EFC70] Die Berkeley Group folgte mit Ingres und der Abfragesprache QUEL.

Oracle (damals noch unter den Firmennamen SDL und RSI) verwertete die Ergebnisse des System R und führte SQL zum kommerziellen Erfolg. IBM folgte mit SQL/DS und DB2. **Die relationalen**

Datenbanksysteme verdrängten in den 1980er Jahren die hierarchischen und netzwerkartigen Systeme und der Grossteil der Behörden, Konzerne, Institute und mittelständischen Unternehmen stellte seine IT auf Datenbanksysteme um.

Während in den 1990er Jahren wenige kommerzielle Hersteller von Datenbank-Software faktisch den Markt beherrschten (namentlich IBM, Informix, dBASE, Microsoft SQL Server und Oracle), erlangten in den 2000ern die Open-Source-Datenbankmanagementsysteme eine immer grössere Bedeutung. Vor allem MySQL und PostgreSQL erzielten signifikante Marktanteile. Als Reaktion begannen die führenden kommerziellen Hersteller, gebührenfreie Versionen ihrer Datenbank-Software anzubieten. Etwa seit 2001 ist aufgrund mangelnder Skalierbarkeit relationaler Datenbanken die Bedeutung der NoSQL-Systeme gewachsen.

Billige und schnelle Hardware erlaubt in der heutigen Zeit die Entwicklung von grossen IN-MEMORY Datenbanken, welche auf dem relationalen Modell basieren. Beispiele dafür sind HANA von SAP und Exadata von Oracle. Bei diesen beiden Produkten geht es aber auch stark um die Kundenbindung an einen Anbieter indem dieselbe Firma von der Hardware über das Betriebssystem und die Datenbank bis zur Anwendungssoftware alles in einem Guss liefert.

IN-MEMORY ist in der Zwischenzeit eine Option in fast allen Datenbanken (selbst SQLite verfügt darüber). Die Problematik der in-memory Technologie liegt in der permanenten Abspeicherung der Daten auf nicht-flüchtige Datenträger.

6 Architekturen von Datenbanksystemen [PAY97]

6.1 Hierarchisches Datenbankmodell

Eine hierarchische Datenbank ordnet verschiedene Datentypen verschiedenen Ebenen einer Datenstruktur zu. Die Verbindungen zwischen den Daten verschiedener Ebenen sind einfach.

Ein Nachteil des hierarchischen Modells ist, dass ein einzelner Dateneintrag u.U. mehrfach erscheinen muss, da er in verschiedenen hierarchischen Zusammenhängen steht. Ein weiterer Nachteil des hierarchischen Modells ist, dass es schwierig ist, im Nachhinein die Struktur der Datenbank wesentlich zu verändern.

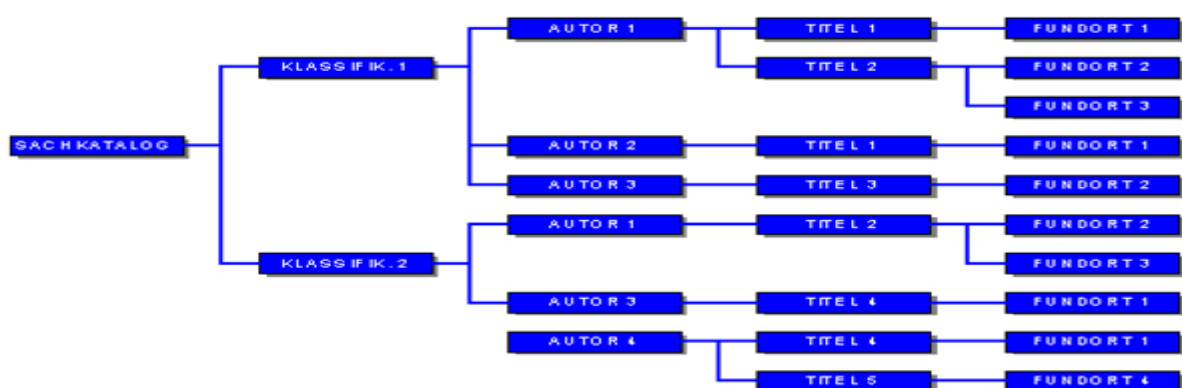


Abbildung 2: Hierarchische Datenbank

6.2 Netzwerkdatenbankmodell

In einer Datenbank nach dem Netzwerkmodell hat prinzipiell jeder Datenknoten unmittelbaren Zugang zu jedem anderen. Deswegen muss kein Knoten mehrfach vorhanden sein.

Das Netzwerkmodell einer Datenbank wurde 1969 von der Data Base Task Group (DBTG) von CODASYL (Conference on Data Systems Languages) formuliert.

Der wichtigste Nachteil des Netzwerkmodells ist, dass die Struktur sehr schnell ziemlich undurchsichtig wird.

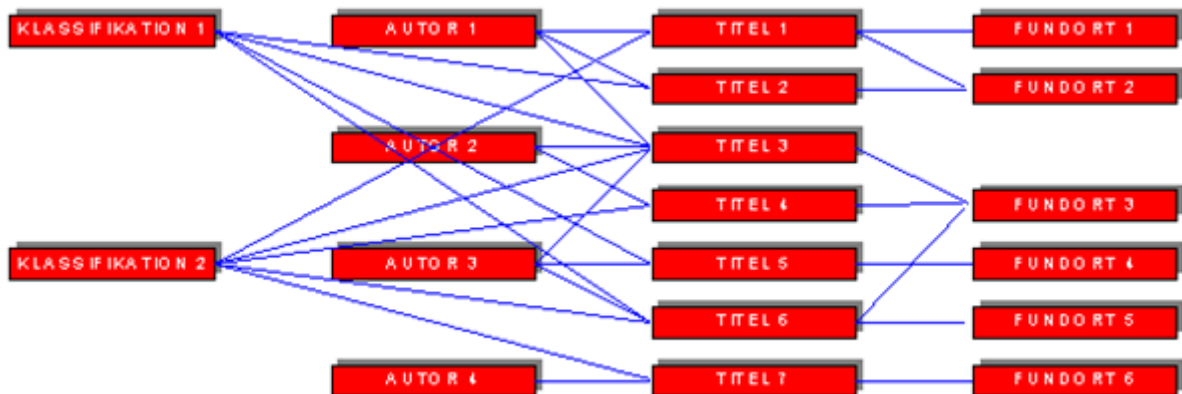


Abbildung 3: Netzwerk Datenbank

6.3 Relationales Datenbankmodell

Das Hauptmerkmal einer relationalen Datenbank ist, dass die Daten in *Tabellen (tables)*, akademisch *Relationen (relations)* genannt, organisiert sind, die voneinander weitgehend unabhängig sind. Eine Tabelle / Relation ist ein zweidimensionales Gebilde aus *Reihen (rows)* und *Spalten (columns)*. Alle Einträge in einer bestimmten Spalte haben dieselbe Bedeutung (z.B. Titel, Untertitel ...). Jede Spalte repräsentiert ein bestimmtes Attribut (attribute) der durch die Tabelle repräsentierten Objekte. Jede Reihe enthält die Attribute eines einzigen Objektes der Tabelle. Die Reihenfolge der Reihen und Spalten in einer Tabelle ist ohne Bedeutung. Verschiedene Tabellen sind dadurch direkt verknüpft, dass sie jeweils mindestens eine Spalte mit demselben Attribut enthalten. Über direkt verknüpfte Tabellen lassen sich indirekte Verknüpfungen herstellen.

Der Hauptvorteil des relationalen Datenbankmodells ist, dass die Struktur der Datenbank verändert werden kann (indem man Tabellen hinzufügt oder entfernt), ohne dass deswegen Anwendungen geändert werden müssen, die sich auf die ältere Struktur gründen.

Ein weiterer Vorteil des relationalen Modells ist, dass man beliebig viele *Sichtweisen (views)* oder *virtuelle Tabellen (virtual tables)* der Daten mit unterschiedlichster logischer Struktur schaffen kann, indem verschiedene Tabellen bzw. Teile von Tabellen kombiniert werden. Dazu muss die Datenbank physisch nicht verändert werden.

Die Meta-Daten der Struktur der ganzen Datenbank und ihrer Tabellen wird in sogenannten *Schemas (schemas)* definiert und gespeichert.

Der Bereich der Werte, die ein Attribut (d.h. ein Eintrag in einer bestimmten Spalte) annehmen kann, nennt man *Wertebereich (domain)* des Attributes. Dieser Wertebereich wird durch *Einschränkungen (constraints)* näher bestimmt.

Das relationale Datenbankmodell wurde 1970 von E. F. Codd (englischer Mathematiker) bei IBM entworfen. 1979 erschien dann mit *Oracle* die erste kommerzielle Implementierung. Heute gibt es über 100 relationale DBMS.

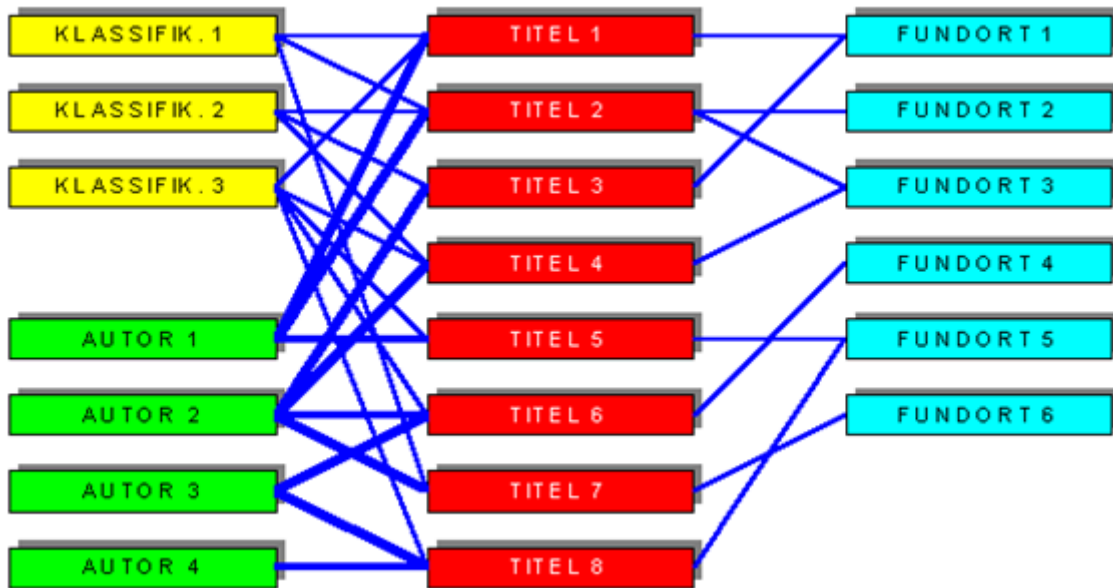
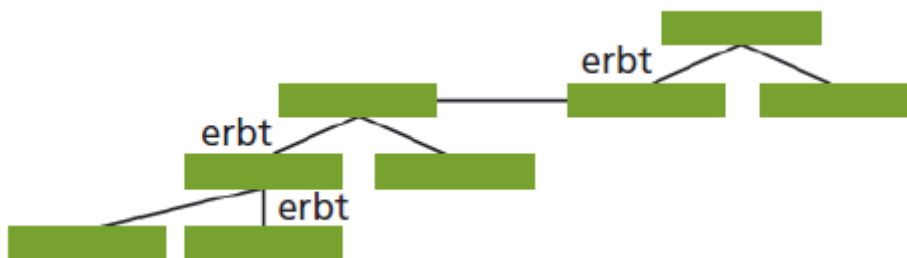


Abbildung 4: Relationale Datenbank

6.4 Objektorientierte Datenbank

Objekte sind modellhafte Abbilder der Wirklichkeit. Das Objekt-Datenmodell wird vor allem im Bereich der Softwareentwicklung eingesetzt.



7 Einführung in SQL [BRE04]

SQL = Structured Query Language (strukturierte Abfragesprache)

SQL ist eine standardisierte Sprache zum Erstellen, Bearbeiten und Kontrollieren von relationalen Datenbanken.

SQL ist keine prozedurale Computersprache: d.h. SQL kann nicht alleinstehend, sondern nur in Verbindung mit einer andern Sprache oder einer Anwendung genutzt werden.

SQL ...

- Beruht auf dem relationalen Datenbankmodell
- Ist eine high Level Computersprache
- Ist interaktiv (ad hoc Datenbankabfragen)
- Ist hersteller-unabhängig (+/-)
- Es existieren Sprach-Standards
- Ist portabel zwischen verschiedenen Computersystemen

Befehlsgruppen:

- Definition von Daten (DDL = Data Definition Language)
Erstellen von Tabellen, Indizes, Views. Ändern von DB-Objekten. Löschen von DB-Objekten
- Manipulation von Daten (DML = Data Manipulation Language)
Daten anzeigen und auswerten, ändern, einfügen und löschen.
- Datenbankverwaltung (DCL = Data Control Language (DCL)

Für viele Anwendungen reichen die DML Befehle aus. Aus diesem Grund wird weiter nur auf diese Befehlsgruppe eingegangen.

SQL-Syntax

7.1 Auswahl von Daten (SELECT)

Mit der SELECT Anweisung können Selektionen und Projektionen aus Tabellen erstellt werden.

- Selektion → Auswahl Zeilen einer Tabelle
- Projektion → Auswahl der angezeigten Felder (Spalten)

SELECT [DISTINCT] Auswahlliste

welche Spalten?

FROM Quelle

aus welcher Tabelle?

[**WHERE** Where-Klausel]

welche Zeilen?

[**GROUP BY** (Group-by-Attribut)+

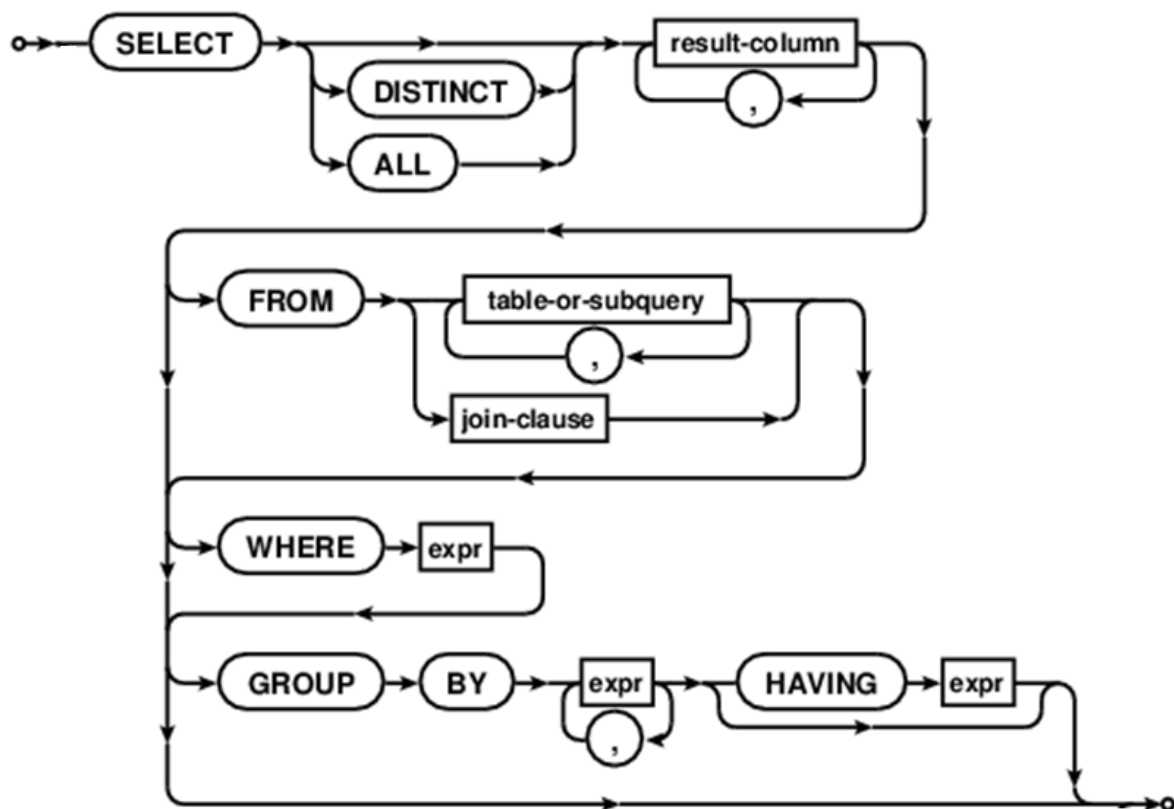
gruppiert nach?

[**HAVING** Having-Klausel]]

welche Gruppen?

[**ORDER BY** (Sortierungsattribut [ASC|DESC])+];

sortiert nach?



Einige Beispiele: (Tabelle „buch“)

BUCHID	TITEL	ISBN	AUFLAGE	JAHR	PREIS	VERLAGSID
1	Algorithmen in C++	3-89319-462-2	1. Aufl., 1., korr.	1994	89.9	1
2	Spreadsheets:	3-528-05256-2		1992	10	2
3	C und Assembler	3-8023-0371-7		1992	69	3
4	Programmiersysteme			1967	15	5
5	LATEX. Kompaktführer	3-89319-152-6	2. durchges. Aufl.	1991	20	1
6	Workgroups 3.11	3-87791-582-5		1994	25	4
7	DOS-Software	3-8023-1174-4	2. Aufl.	1992	30	3
9	OO Datenbanken :	3-89319-315-4	1. Nachdr.	1993	79.9	1
10	Learnig GNU Emacs:	0-937175-84-6		1991	35	7

Projektion: *Titel und ISBN Nummer der Bücher*

```
SELECT TITEL, ISBN
FROM buch
```

TITEL	ISBN
Algorithmen in C++	3-89319-462-2
Spreadsheets:	3-528-05256-2
C und Assembler	3-8023-0371-7
Programmiersysteme	
LATEX. Kompaktführer	3-89319-152-6
Workgroups 3.11	3-87791-582-5
DOS-Software	3-8023-1174-4
OO Datenbanken :	3-89319-315-4
Learnig GNU Emacs:	0-937175-84-6

Selektion: *Alle Bücher aus dem Jahr 1992 auswählen.*

```
SELECT *
FROM buch
WHERE JAHR = 1992
```

BUCHID	TITEL	ISBN	AUFLAGE	JAHR	PREIS	VERLAGSID
2	Spreadsheets:	3-528-05256-2		1992	10	2
3	C und Assembler	3-8023-0371-7		1992	69	3
7	DOS-Software	3-8023-1174-4	2. Aufl.	1992	30	3

Sortieren: *Alle Bücher sortiert nach Ausgabejahr.*

```
SELECT *
FROM buch
ORDER BY JAHR desc
```

BUCHID	TITEL	ISBN	AUFLAGE	JAHR	PREIS	VERLAGSID
1	Algorithmen in C++	3-89319-462-2	1. Aufl., 1., korr.	1994	89.9	1
6	Workgroups 3.11	3-87791-582-5		1994	25	4
9	OO Datenbanken :	3-89319-315-4	1. Nachdr.	1993	79.9	1
2	Spreadsheets:	3-528-05256-2		1992	10	2
3	C und Assembler	3-8023-0371-7		1992	69	3
7	DOS-Software	3-8023-1174-4	2. Aufl.	1992	30	3
5	LATEX. Kompaktführer	3-89319-152-6	2. durchges. Aufl.	1991	20	1
10	Learnig GNU Emacs:	0-937175-84-6		1991	35	7
4	Programmiersysteme			1967	15	5

Gruppieren: *Anzahl Bücher pro Ausgabejahr zählen.*

```
SELECT jahr, count(buchid)
FROM buch
GROUP BY jahr
```

JAHR	count(buchid)
1967	1
1991	2
1992	3
1993	1
1994	2

Aggregatsfunktion: *Durchschnittspreis, höchster und tiefster Preis für ein Buch.*

```
SELECT avg(preis) as Durchschnittspreis
, max(preis) as "Teuerstes Buch"
, min(preis) as "Billigstes Buch"
FROM buch
```

Durchschnittspreis	Teuerstes Buch	Billigstes Buch
41.5333333333	89.9	10

Unterabfrage: *Welches ist das teuerste Buch?*

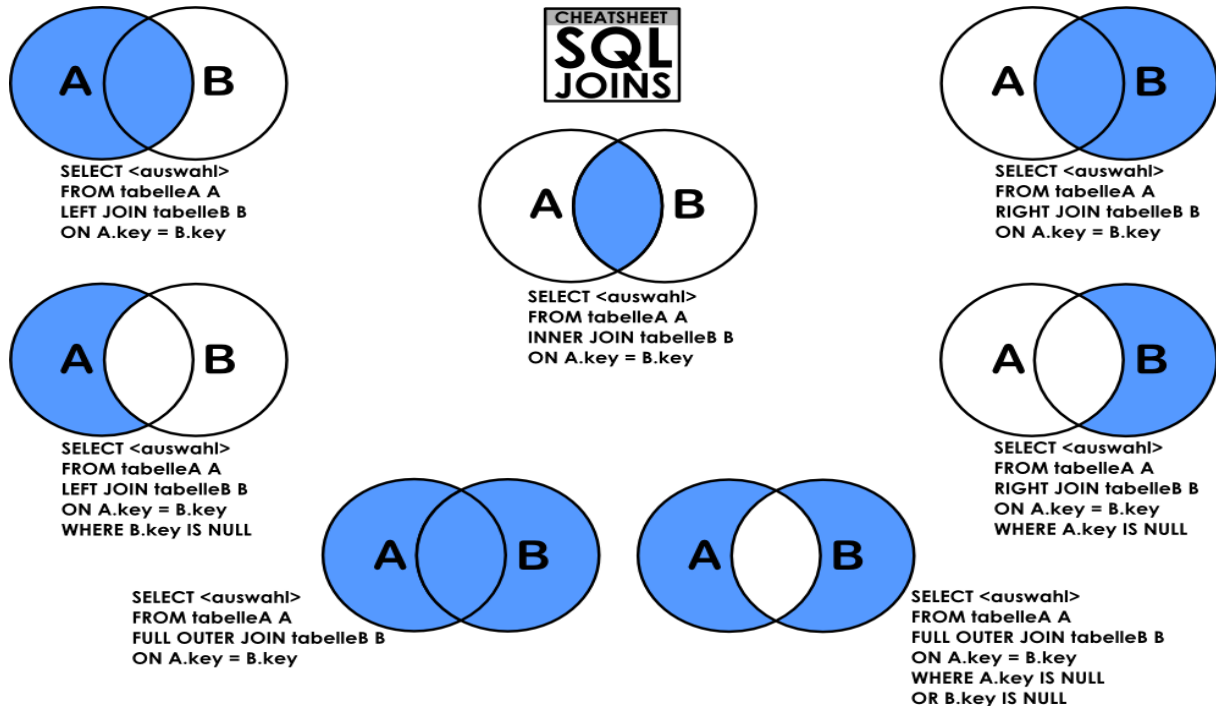
```
SELECT *
FROM buch
WHERE preis = (SELECT max(preis) FROM buch)
```

BUCHID	TITEL	ISBN	AUFLAGE	JAHR	PREIS	VERLAGSID
1	Algorithmen in C++	3-89319-462-2	1. Aufl., 1., korr.	1994	89.9	1

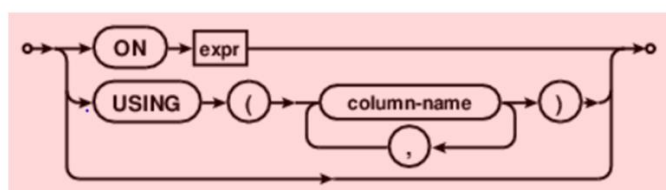
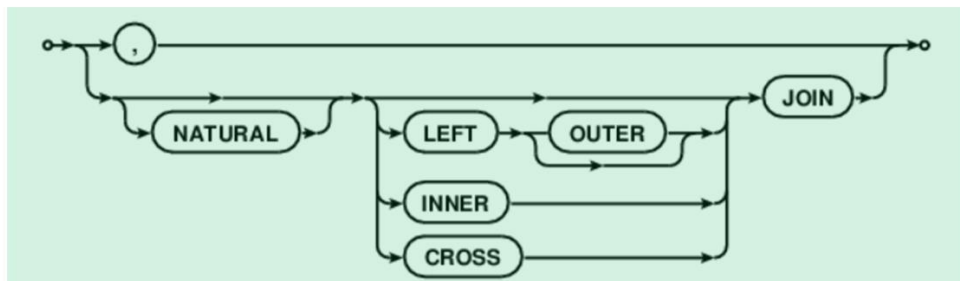
7.1.2 Verbinden von Tabellen

Da sich die meisten Abfragen auf mehr als eine Tabelle in einer relationalen Datenbank beziehen, ist die Verbindung von verschiedenen Tabellen von grosser Wichtigkeit. Mit SQL ist es möglich alle möglichen Mengen-Operationen abzubilden. Die weitaus am häufigsten verwendete Operation ist die Schnittmenge. Das bedeutet, dass Informationen zu Schlüsseln gesucht werden, die in beiden beteiligten Mengen (bzw. Tabellen) vorhanden sind.

Untenstehende Abbildung zeigt die verschiedenen Mengen-Operationen und die dazugehörige SQL Syntax.



Im Syntax Diagramm wird die Join Operation folgendermassen beschrieben:



Beispiel:

Um den Namen des Verlags zu jedem Buchtitel auszugeben, müssen die Tabellen „verlag“ und „buch“ verknüpft werden:

Der Schlüssel (verlagsID) der Tabelle „verlag“ ist als Fremdschlüssel in der Tabelle „buch“ abgespeichert. Dadurch wird ein eindeutiger Bezug von jedem Buch zum publizierenden Verlag hergestellt.

Da SQL historisch gewachsen ist, gibt es mehrere Möglichkeiten diese Verbindung auszuwerten:

JOIN Bedingung innerhalb der FROM-Klausel (moderne Variante):

```
SELECT buch.buchID, buch.titel, verlag.name
FROM buch NATURAL JOIN verlag
WHERE buch.jahr = 1992
```

```
SELECT buch.buchID, buch.titel, verlag.name
FROM buch JOIN verlag ON buch.verlagsID = verlag.verlagsID
WHERE buch.jahr = 1992
```

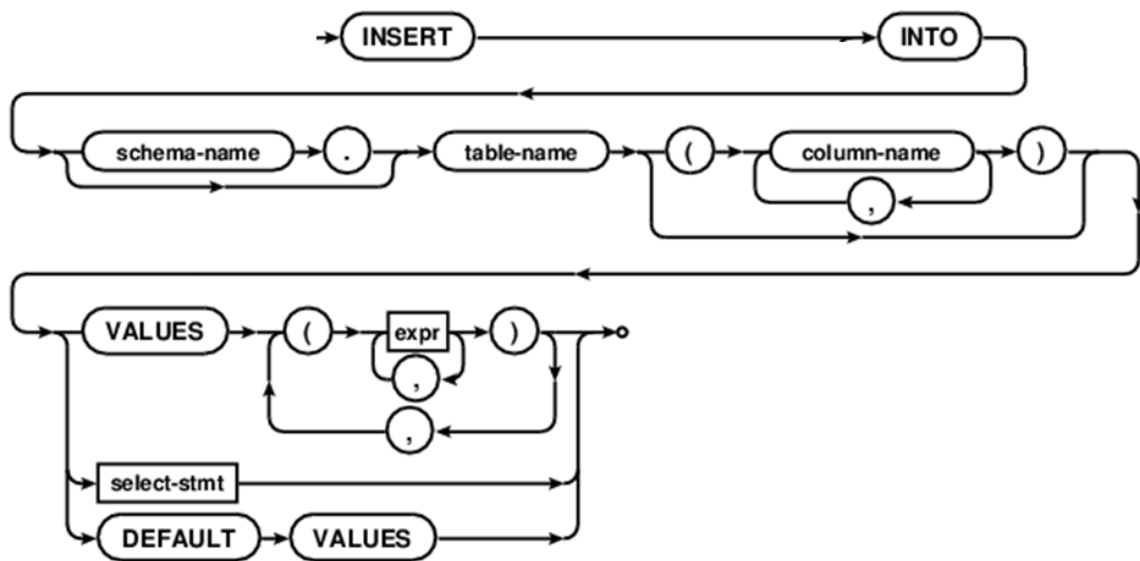
```
SELECT buch.buchID, buch.titel, verlag.name
FROM buch JOIN verlag USING(verlagsID)
WHERE buch.jahr = 1992
```

Alle drei SQL-Statements liefern dasselbe Ergebnis. Wenn das Schlüsselwort NATURAL verwendet wird, werden automatisch die gemeinsamen Schlüsselfelder auf Gleichheit verglichen. Mit der USING Klausel wird auch vorausgesetzt, dass gleichnamige Schlüsselattribute in beiden Tabellen vorhanden sind. Bei der Verwendung der ON-Klausel, können auch Schlüsselfelder verglichen werden, welche nicht gleichnamig sind.

Die JOIN Bedingung kann auch in der WHERE-Klausel formuliert werden (alte Variante, aber weit verbreitet):

```
SELECT buch.buchID, buch.titel, verlag.name
FROM buch, verlag
WHERE buch.verlagsID = verlag.verlagsID AND buch.jahr = 1992
```

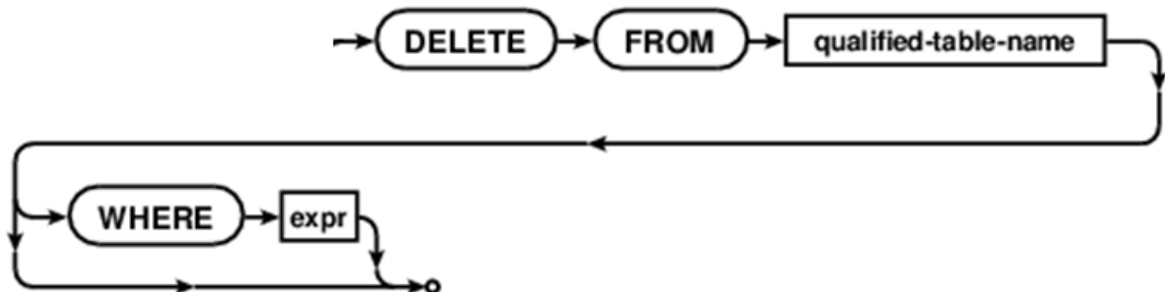
7.2 Einfügen von Daten (INSERT)



Beispiel:

```
INSERT INTO verlag (verlagsid, name, ort)
VALUES (812, 'Junior', 'Bern')
```

7.3 Löschen von Daten (DELETE)



Beispiel:

```
DELETE FROM verlag WHERE verlagsid = 812
```

Ganze Syntax ist in der SQLite Dokumentation zu finden.

<https://www.sqlite.org/lang.html>

8 Datenbankmodellierung [] – relationales Datenmodell

In Karteisystemen wurden Karteikarten erstellt

Nummer:	14555
Ar Nummer:	14556
Fa Art:	Schlüssel
Fu Farbe:	-
Fu Funddatum:	12.09.2000
Fi Fundort:	Bahnhof
Finder:	Hegi Roger
	Brandweg 12
	9876 Zuchwil

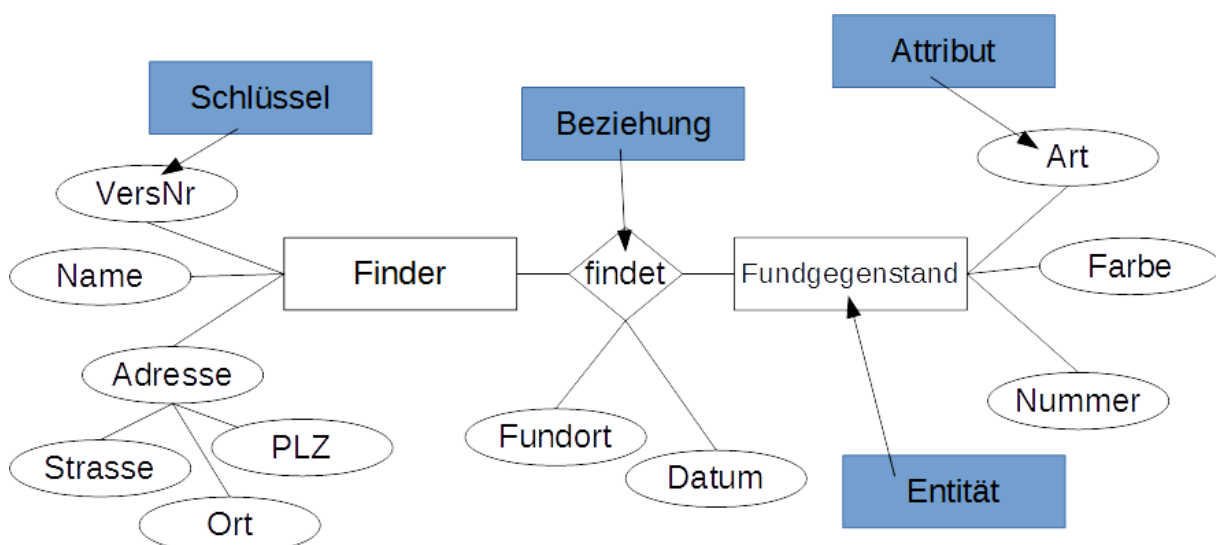
Als Beispiel schauen wir uns die Kartei eines Fundbüros an.

Definition einer Tabelle:

Fundgegenstand				
Feldname	Wert	Datentyp	leer?	Schlüssel
Nummer	14556	Zahl	Nein	Ja
Farbe	-	Text	Ja	Nein
Funddatum	12.09.2015	Datum	Nein	Nein
Fundort	Bahnhof	Text	Nein	Nein
Finder_Name	Hans Muster	Text	Ja	Nein
Finder_Strasse	Bachweg 12	Text	Ja	Nein
Finder_Ort	9630 Wattwil	Text	Ja	Nein

In einem relationalen Datenbanksystem werden die Tabellen auch Entitäten genannt (Entität = ein existierendes Ding). Relationen bezeichnen die Beziehungen, welche die Tabellen untereinander haben können.

Ein Modell beschreibt ein Abbild der Wirklichkeit. Die Modellierung relationaler Datenbanken erfolgt mit dem von Peter Chen entwickelten Entity-Relationship-Model (ERM).



Bestandteile des ERM

1. Entitäten sind eindeutig unterscheidbare Objekte. Exemplare (Instanzen) sind konkrete Ausprägungen einer Entität.
2. Beziehungen (Relationen) sind Assoziationen zwischen Entitäten. Auch Beziehungen haben Exemplare (Instanzen).
3. Attribute sind Eigenschaften von Entitäten oder Beziehungen
4. Die Attribute einer Entität sind vom Schlüsselattribut abhängig.



8.1 Zusammengesetzte Attribute

In der Entität „Finder“ gibt es das Attribut „Adresse“. Bei genauerer Betrachtung fällt auf, dass dieses Attribut aus drei Teilen besteht: „Strasse“, „PLZ“ und „Ort“.

8.2 Schlüsselattribut

Ein Schlüsselattribut ist ein Attribut oder eine Kombination von Attributen, die eine Entität eindeutig identifiziert.

Für die Verbindung von Tabellen wird der Schlüssel einer anderen Entität in der Tabelle als sogenannter Fremdschlüssel abgespeichert.

PID	Name	Vorname	AID
1	Kunz	Christian	7
2	Meier	Hans	53

AID	Strasse	PLZ	Ort
7	Ulmiz	3088	Biel
8	Strasse	9000	St.G

Ein Pfeil verbindet den Wert '53' in der Spalte 'AID' der ersten Tabelle mit der Spalte 'AID' der zweiten Tabelle, was die Referenzierung eines Fremdschlüssels darstellt.

- Ein Primärschlüssel identifiziert ein Exemplar einer Entität oder Beziehung eindeutig.
- Ein Fremdschlüssel verknüpft ein Exemplar mit einem Primärschlüssel eines anderen.

8.3 Kardinalität

Beziehungen zwischen Tabellen unterscheiden sich in der Anzahl möglicher Beziehungen, die ein Element einer Entität mit Elementen der andern Entität eingehen kann. Diese Anzahl der möglichen Beziehungen wird Kardinalität genannt. Mögliche Ausprägungen sind die 1:1, die 1:n und die n:m-Beziehung.



Eine Klasse hat EINEN Klassensprecher. Ein Klassensprecher spricht für EINE Klasse.
Dies ist eine 1:1 – Beziehung.



In einer Klasse hat es mehrere (=n) Schüler. Ein Schüler ist in EINER Klasse.
Dies ist eine 1:n Beziehung.

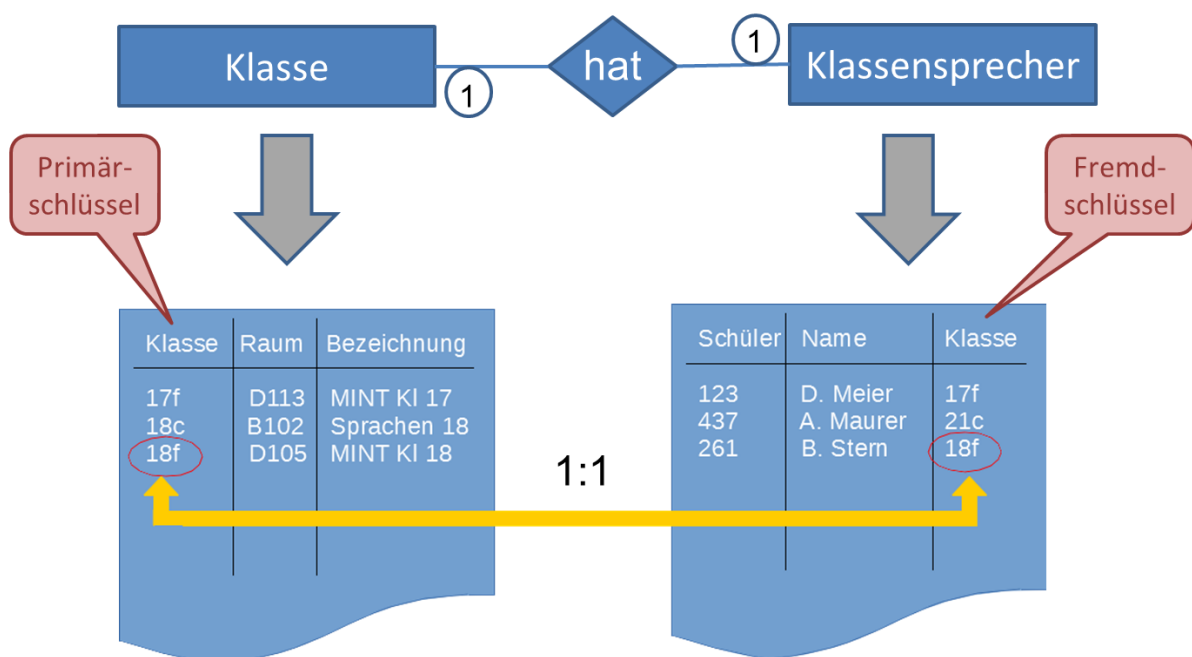


Eine Klasse wird von mehreren (=n) Lehrern unterrichtet. Ein Lehrer unterrichtet mehrere (=m) Klassen.
Dies ist eine n:m Beziehung.

8.4 Relationale Auflösung

Da die Datenbank die Information in Tabellen darstellt, müssen wir aus den Entitäten und Relationen entsprechende Tabellen definieren. Diese Umwandlung nennt man die relationale Auflösung.

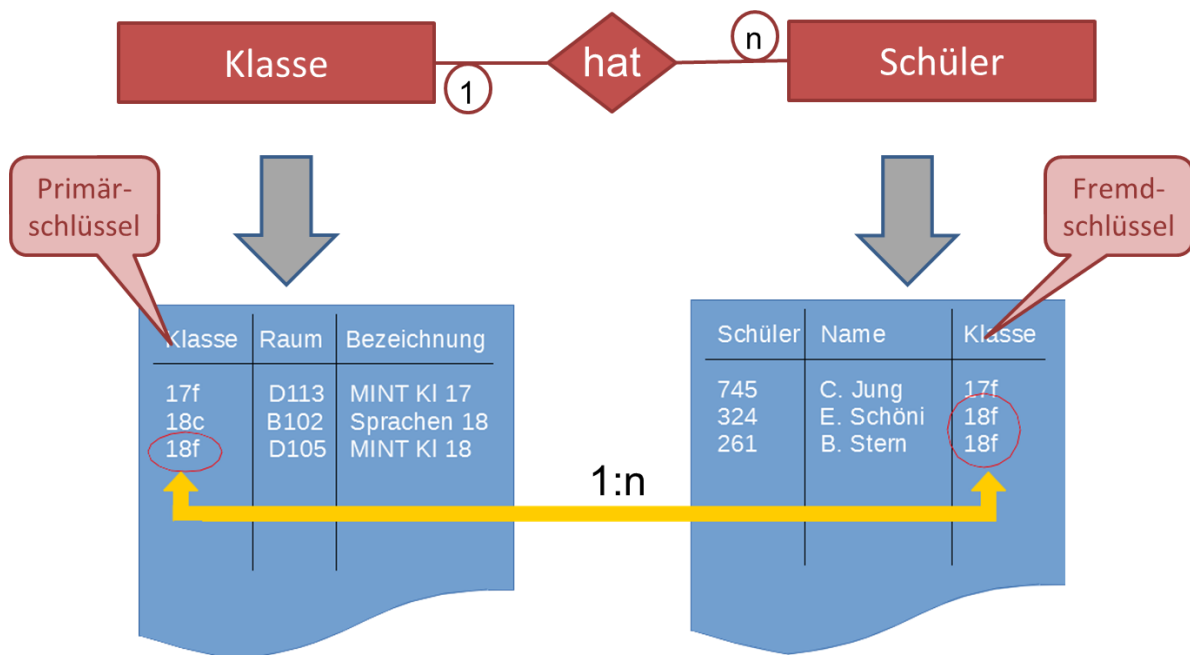
Auflösung der 1:1 Beziehung:



Die Entitäten Klasse und Klassensprecher werden zu Tabellen. Die Attribute werden zu Spalten der Tabellen. Die Primärschlüssel werden als Primärschlüsselfelder in den Tabellen festgelegt (Klasse und Schüler).

Als Fremdschlüssel kann bei einer 1:1-Beziehung einer der beiden Primärschlüssel in der jeweils anderen Tabelle verwendet werden. Hier wurde die Klasse in der Tabelle Klassensprecher herangezogen.

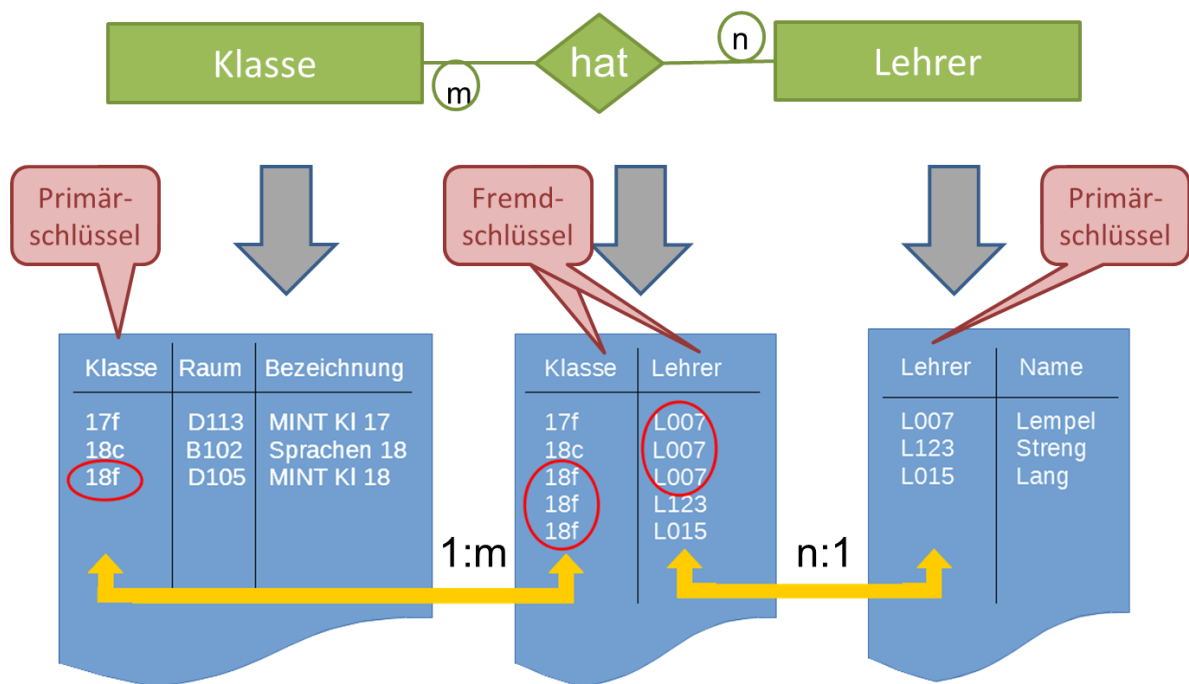
Auflösung der 1:n-Beziehung:



Die Entitäten Klasse und Schüler werden zu Tabellen. Die Attribute werden zu Feldern der Tabellen. Die Primärschlüssel werden als Primärschlüsselfelder in den Tabellen festgelegt (Klasse und Schüler).

Bei einer 1:n-Beziehung wird der Primärschlüssel der 1-Tabelle als Fremdschlüssel der n-Tabelle festgelegt. Hier ist der Fremdschlüssel Klasse in der Tabelle Schüler.

Auflösung der n:m-Beziehung:



Die Datenbank erlaubt nur 1:1 und 1:n-Beziehungen zwischen Tabellen. Daher müssen wir eine n:m-Beziehung in zwei 1:n-Beziehungen auflösen:

Die Entitäten werden zunächst in Tabellen aufgelöst. Ihre Attribute werden zu Feldern der beiden Tabellen. Die Primärschlüssel werden die Schlüsselattribute. Hier sind das Klasse und Lehrer.

Im nächsten Schritt erstellen wir aus der Beziehung eine neue Tabelle: die Beziehungstabelle. Sie bekommt den Namen KlasseHatLehrer, also eine Kombination aus den beiden Tabellen, die sie verknüpft, und dem Namen der Beziehung. (diese Namensgebung ist nicht zwingend. Es ist aber hilfreich, wenn der Name der Tabelle, deren Funktion erfasst.) In dieser Tabelle legen wir als Fremdschlüssel die beiden Primärschlüssel aus Klasse und Lehrer an. Die Fremdschlüssel repräsentieren die n-Teile (mehrere gleiche Schlüssel).

Die Beziehungstabelle enthält in unserem Beispiel die beiden Fremdschlüssel Klasse und Lehrer. Aber wo befindet sich der Primärschlüssel?

Der Primärschlüssel ist eine Kombination der beiden Fremdschlüssel, also ein zusammengesetzter Schlüssel.

10 Vermeiden von Redundanz → Normalisierung

Um den verschiedenen Anforderungen an eine Datenbank (siehe Kap. 4) gerecht zu werden, müssen die Tabellen nach gewissen Kriterien definiert werden. In der Datenbank-Theorie unterscheidet man sogenannte Normalformen, die eine Hierarchie von Bedingungen an die Tabellendefinitionen stellen.

Wir beschreiben hier nur die erste, zweite und dritte Normalform. Viele reale Datenbanken halten sich an diese Normalformen. Die strengeren Anforderungen der weiteren Normalformen werden in der Praxis nicht beachtet.

Die wichtigsten Gründe für das Normalisieren einer Datenbank sind:

1. Vermeidung von Redundanz (doppelte Einträge)
2. Vermeidung von Anomalien (widersprüchliche Daten)
3. Konsistenz (Vollständigkeit und Richtigkeit)
4. Vereinfachung der Wartung
- 5.

Erste Normalform (1NF): Jedes Attribut einer Tabelle ist unteilbar.

Eine Information in einem Attribut ist dann unteilbar, wenn sie nicht weiter in Einzelinformationen zerlegt werden kann. Als Attributwerte sind keine Aufzählungen, Listen oder zusammengesetzte Datenelement erlaubt.

Beispiel:

Das zusammengesetzte Attribut Name wird in die Attribute Vorname und Nachname aufgeteilt.

Zweite Normalform (2NF): Die Tabelle ist in 1NF und alle Attribute müssen vom gleichen Primärschlüssel abhängen.

Das heisst, dass es keine Attribute geben darf, die nur von einem Teil des Schlüssels abhängig sind.

Beispiel

CD-Nr	Titel	Interpret	Gründung	Track-Nr	Song
12334556	Freak of nature	Anastasia	1999	1	Freak of nature
12334556	Freak of nature	Anastasia	1999	2	Not that kind
00032556	Rattle'n'Hum	Dire Straits	1982	7	Bloody Sunday

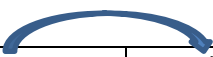
In dieser Tabelle sind die Felder Titel, Interpret und Gründung nur von einem Teil des Schlüssels abhängig, nämlich der CD-Nr. Nur die Spalte Song ist vom ganzen Schlüssel (CD-Nr/Track-Nr) abhängig.

Die Tabelle muss also in zwei Tabellen aufgeteilt werden. Einerseits die CD-Tabelle und andererseits die Song-Tabelle.

Dritte Normalform (3NF): Die Tabelle ist in 2NF und es gibt keine transitiven Abhängigkeiten.

Das heisst, dass kein Attribut direkt von einem andern Attribut abhängig ist.

Im Beispiel unserer CD-Tabelle existiert eine solche Abhängigkeit. Das Gründungsjahr der Band ist direkt vom Interpret abhängig!



CD-Nr	Titel	Interpret	Gründung
12334556	Freak of nature	Anastasia	1999
12334556	Freak of nature	Anastasia	1999
00032556	Rattle'n'Hum	Dire Straits	1982

Die Tabelle muss also in eine CD-Tabelle und eine Band-Tabelle aufgeteilt werden!

11 Literaturverzeichnis

[Bre04] Bremen, Aus- und Fortbildungszentrum. (2004). *Entwicklung von Datenbankapplikationen - Datenbankgrundlagen*.

[EFC70] Codd, E. F. (13. 6 1970). A relational model of data for large shared data banks. *Communications of the ACM*, S. 377-387.

[MUN99] Munz, Udo Matthias, RBS Ulm

[PAY97] Payer, Margarete <1942 - >: Datenbankaufbau : Skript / Margarete Payer & Alois Payer. -- Kapitel 1: Grundkonzepte von Datenbanken. -- Fassung vom 1997-06-02

Genealogy of Relational Database Management Systems: [1] beim Hasso-Plattner-Institut.

Genealogy of Relational Database Management Systems

