Datenbanken

Theorie

Inhaltsverzeichnis

Version vom 27. März 2026

1 Einführung

Tabellen

Wir werden uns im Folgenden auf einen weit verbreiteten Typ von Datenbanken beschränken:

Dabei spielt die Datenstrukturierung durch Tabellen eine besondere Rolle.

MITARBEITER

Mitarbeiter_Nr	Name	Ort

- Ein Merkmal oder Attribut ordnet jedem Eintrag der Tabelle einen bestimmten Datenwert aus einem vordefinierten *Wertebereich* (*Domain*) zu.
- Ein *Identifikationsschlüssel* oder *Schlüssel* einer Tabelle ist ein Merkmal oder eine minimale Merkmalskombination, dessen Werte die Datensätze (Tupel) der Tabelle eindeutig identifizieren.

MITARBEITER

Mitarbeiter_Nr	Name	Ort
203	Schweizer	Stans
73	Becker	Buochs
191	Meier	Oberdorf
...	...	...

- Jeder Schlüsselwert identifiziert eindeutig einen Datensatz innerhalb der Tabelle. Verschiedene Tupel dürfen keine identischen Schlüssel aufweisen (*Eindeutigkeit*).
- Falls der Schlüssel eine Kombination von Merkmalen darstellt. muss diese minimal sein. D. h. kein Merkmal der Kombination kann weggelassen werden, ohne dass die Eindeutigkeit der Identifikation verloren geht (*Minimalität*).

Schlüsselattribute sind durch Kursivschrift oder durch Unterstreichen zu kennzeichnen.

Anstelle eines natürlichen Merkmals oder einer natürlichen Merkmalskombination kann ein künstliches Merkmal als Schlüssel eingeführt werden. Zum Beispiel eine Schüler-Nummer anstelle der Merkmalskombination Name, Vorname, Adresse)

Ein künstlicher Schlüssel sollte aufgrund dieser Überlegungen *anwendungsneutral* und ohne *Semantik* (Aussagekraft, Bedeutung) sein. Man bedenke auch, dass sich die Bedeutung eines „sprechenden Schlüssels“ mit der Zeit ändern kann.

Eine *Tabelle* oder *Relation* ist eine Menge von Tupeln, die tabellenförmig dargestellt werden und folgende Anforderungen erfüllen:

- Eine Tabelle besitzt einen eindeutigen Tabellennamen.
- Innerhalb einer Tabelle ist jeder Merkmalsname eindeutig und bezeichnet eine bestimmte Spalte mit der gewünschten Eigenschaft.
- Die Anzahl der Merkmale ist beliebig, die Ordnung der Spalten innerhalb der Tabelle ist bedeutungslos.
- Die Anzahl der Tupel ist beliebig. Die Ordnung der Tupel innerhalb der Tabelle ist bedeutungslos.
- Eines der Merkmale oder eine (minimale) Merkmalskombination identifiziert eindeutig die Tupel innerhalb der Tabelle und wird als *Primärschlüssel* bezeichnet.

Datenmodellierung

„Ein *Datenmodell* (engl. *data model*) beschreibt auf strukturierte und formale Weise die für eine Informationssystem notwendigen Daten und Datenbeziehungen.“ [Meier, S. 17]

Dazu werden die nötigen *Datenklassen* (*Datenkategorien*, *Entitätsmengen*) erfasst und zueinander in Beziehung gebracht.

1. *Datenanalyse*

   Die für das Informationssystem notwendigen Daten und deren Beziehungen werden gemeinsam mit den Benutzern ermittelt. (Fragebögen, Bedarfsanalysen, Formularsammlungen)

   Daraus wird eine Dokumentation zusammengestellt, welche die Zielsetzung und die wesentlichen Informationssachverhalte enthält.

2. *Entwurf eines Entitäten-Beziehungsmodells*

   Entitätsmengen (Datenkategorien) und Beziehungsmengen werden grafisch durch Rechtecke bzw. Rhomben dargestellt. (Dazu gleich mehr ....)

3. *Überführung des Entitäten-Beziehungsmodell in ein relationales Datenbankschema*

   Durch *Abbildungsregeln* wird das Entitäten-Beziehungsmodell in ein relationales Datenbankschema überführt.

2 Das Entity Relationship Model

Das Entity Relationship Model (ERM) ist ein graphisches Datenmodell zur formalisierten Darstellung von Beziehungen zwischen Daten mittels eines Diagramms. Diese Modellierungstechnik gehört heute zum Standard in der Entwicklung von Datenbanken. Im ERM soll ein Ausschnitt aus einer realen Geschäftsbeziehung dargestellt werden. Oft gibt es verschiedene Modellierungsmöglichkeiten eines bestimmten Sachverhaltes. Die zentralen Modellierungseinheiten sind der Entitätstyp, der Beziehungstyp und das Attribut. Einfach formuliert besteht die Welt in einem ERM aus Objekten (*Entities*), zwischen denen Beziehungen (*Relationships*) bestehen. Sowohl Entities als auch Relationships können mittels Attributen genauer spezifiziert werden. Ein ERM kann schliesslich in eine relationale (tabellarische) Datenbank überführt werden.

Das ERM wurde in seiner Grundform 1976 von Peter Chen vorgestellt. Mittlerweile existieren etliche Abwandlungen und/oder Erweiterungen der ursprünglichen Chen-Notation. In diesem Lehrgang wird die modifizierte Chen-Notation (Modified Chen Notation oder MC-Notation) verwendet. Sie ist eine Erweiterung der originalen Chen-Notation, bei der die Aussage „kein oder ein Element“ mit dem Buchstaben `c` (choice, can), und die Aussage „ein oder mehr Element(e)“ mit dem Buchstaben `m` (must, multiple) angegeben wird.

Im nächsten Kapitel wird beschrieben, welche Modellelemente zur Modellierung eines ERM eingesetzt werden.

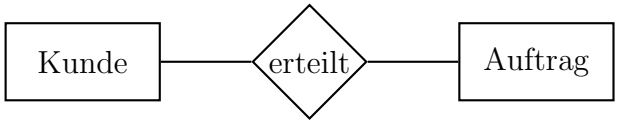
2.1 Modellelemente

Entitätstyp (*entity type*)

Ein Entitätstyp wird eingesetzt, um eine gleichartige Menge (Entitätsmenge) materieller oder immaterieller Objekte (Entitäten) abzubilden. Dabei wird jedem Entitätstyp ein Name zugeordnet. Beispiele sind „Kunde“, „Mitarbeiter“ und „Auftrag“; als graphisches Symbol wird das Rechteck verwendet.

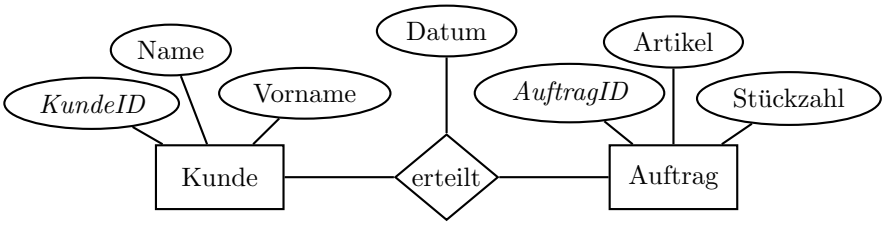
Beziehungstyp (*relationship type*)

Ein Beziehungstyp erlaubt es, eine gleichartige Menge von Zuordnungen zwischen Objekt-Beziehungen abzubilden: Entitätstypen werden durch einen Beziehungstyp miteinander verbunden. Als Name von Beziehungstypen werden Bezeichnungen eingesetzt, die die Zuordnung verbal darstellen. Beispiele sind „Kunde erteilt Auftrag“ und „Auftrag bezieht sich auf Produkt“. Als graphisches Symbol wird die Raute verwendet.



Attribut (*attribute*)

Attribute werden eingesetzt, um die Eigenschaften von Entitäts- und Beziehungstypen im Modell abzubilden. Entitätstypen müssen durch Attribute ergänzt werden, bei Beziehungstypen sind Attribute hingegen optional. Auch Attribute erhalten Namen. Attribute des Entitätstyps „Produkt“ können beispielsweise „Produktname“ oder „Lagerbestand“ sein. Als graphisches Symbol für ein Attribut wird eine Ellipse verwendet.

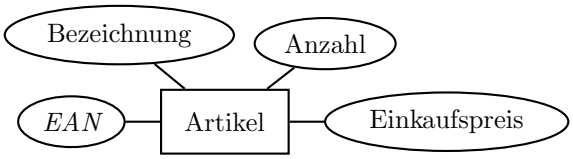


Schlüssel

Prinzipiell muss jeder Entitäts- und Beziehungstyp über einen Schlüssel verfügen, also über ein Attribut, das eine Identifikation der zugrunde liegenden Entitäten ermöglicht (= Primärschlüssel). Sofern zur Identifikation mehrerer Attribute bzw. deren Attributwerte kombiniert werden müssen, bilden diese Attribute gemeinsam den Schlüssel (= zusammengesetzter Schlüssel). Die Bezeichnungen der zum Schlüssel gehörenden Attribute werden unterstrichen oder kursiv dargestellt. Die einander direkt zugeordneten Entitäts- und Beziehungstypen werden durch ungerichtete Kanten miteinander verbunden.

Aufgabe 1

Betrachten Sie das folgende Diagramm und setzen Sie den richtigen Begriff an Stelle der Ausslassungspunkte ein.



(a) „Artikel“ ist ein/eine ...

(b) „Anzahl“ ist ein/eine ...

(c) „EAN“ (European Article Number) ist ein/eine ...

(d) „Artikel(4005800001192, Nivea Creme, 530, 2.20)“ ist ein/eine ...

Aufgabe 2

Zählen Sie fünf Attribute der Entitätsmenge „Buch“ auf. Welches Attribut wäre als Primärschlüssel geeignet?

-
-
-
-
-
-

Domänen

Die Domäne eines Attributs bezeichnet den Wertevorrat, aus dem die Werte eines Attributs stammen.

Beispiele:

- Die Domäne des Attributs „Geburtsjahr“:
  $\{1900, 1901, \ldots, 2012, 2013\}$
- Domäne des Attributs „Hausnummer“:
  $\{1, 2, 3, \ldots\}$.
- Domäne des Attributs „Vorname“:
  {Aadina, Aaron, Adalbert, …, Zacharias, Zita, Zeno, Zoé}.

Aufgabe 3

Gib die Domänen der folgenden Attribute an.

(a) Postleitzahl (Schweiz)

(b) Kanton (Schweiz)

(c) Betriebssystem

(d) Reptilien der Schweiz

Aufgabe 4

Welche Beziehungsmenge(n) kommen in Frage?

(a) Artikel – Kunde

(b) Schüler(in) – Lehrer(in)

(c) Käufer – Rechnung

(d) Frau – Mann

(e) Komponist – Musikstück

(f) Person – Ort

(g) Netzwerkkarte – MAC-Adresse

(h) Flughafen – Fluggesellschaft

(i) Kino – Kinofilm

(j) Pizza – Zutaten

(k) Firma – Firma

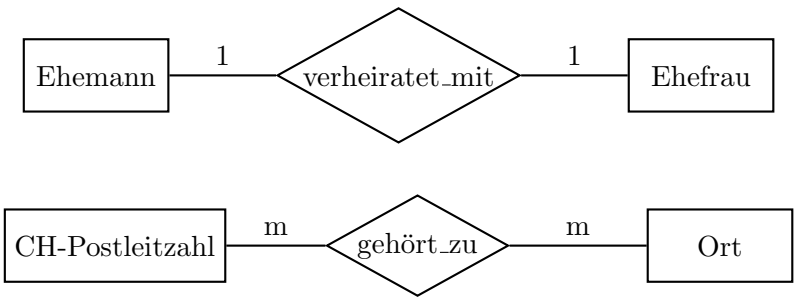
2.2 Kardinalität

Beziehungstyp

Ein Beziehungstyp beinhaltet immer eine Angabe darüber, wie viele Entitäten des einen Entitätstyps mit einer Entität des anderen Entitätstyps in Beziehung stehen können und umgekehrt. Diese Angabe bezeichnet man als Kardinalität des Beziehungstyps bzw. als Assoziationstyp. Dabei bedeuten

Typ	**Beschreibung**
`1`	genau ein(e)
`c`	höchstens ein(e)
`m`	mindestens ein(e)
`mc`	beliebig viele

Beispiele zum Beziehungstyp



Üblicherweise notiert man die Kardinalität zwischen der Beziehungsmenge und der Ziel-Entität. Im letzten Beispiel:

- Jede Person ist an *genau einem* Ort geboren und
- Jeder Ort ist Geburtsort von *beliebig vielen* Personen

Aufgabe 5

Skizzieren Sie die Diagramme zu den unten beschriebenen Beziehungen und geben Sie den Beziehungstyp richtig an. Die Attribute können weggelassen werden.

Aufgabe 5 (a)

Ein Hersteller produziert mehrere Artikel. Artikel werden immer nur von einem Hersteller produziert.

Aufgabe 5 (b)

Ein Mitarbeiter gehört einer Abteilung an. Eine Abteilung hat mindestens einen Mitarbeiter.

Aufgabe 5 (c)

Ein Lehrer unterrichtet mindestens einen Schüler. Ein Schüler wird von mindestens einem Lehrer unterrichtet.

Aufgabe 5 (d)

Ein Drucker ist an genau einem Computer angeschlossen. Einem Computer ist höchstens ein Drucker angeschlossen.

Aufgabe 6

Erstellen Sie ein ERM einer DVD-Ausleihe unter folgenden Voraussetzungen.

- Die Kunden haben einen Namen, Vornamen sowie eine Adresse mit Strasse, PLZ und Ort
- Die DVDs haben einen Titel, eine Altersfreigabe (FSK) und eine International Standard Audiovisual Number (ISAN).
- Ein Kunde kann mehrere DVDs ausleihen. Von jeder DVD gibt es nur ein Exemplar.

Quellen

- http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/daten-wissen/Datenmanagement/Daten-/Entity-Relationship-Model–/index.html [6.10.2012]
- http://ebus.informatik.uni-leipzig.de/www/media/lehre/seminar-pioniere04/sem04swp-hartmann-vortrag.pdf [6.10.2012]
- http://de.wikipedia.org/wiki/Chen-Notation#Modifizierte_Chen-Notation [6.10.2012]

3 Abbildungsregeln

3.1 Tabellen

Einleitung

Da ein relationales Datenbankschema als Objekte nur Tabellen zulässt, müssen sowohl die Entitäts- als auch die Beziehungsmengen in Tabellenform ausgedrückt werden. Dabei muss nach folgenden Regeln, sog. Abbildungsregeln, vorgegangen werden.

Aufbau der Tabellen

Die Tabellen müssen für unsere Zwecke wie folgt aufgebaut sein:

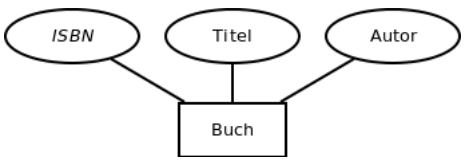
- Der Tabellenname wird in Grossbuchstaben geschrieben.
- In der Kopfzeile stehen die *Merkmalsnamen*. Jede Tabellenzeile mit Daten wird *Datensatz*, oder *Tupel* genannt.
- Jede *Spalte* eines Datensatzes enthält einen *Datenwert* aus dem *Wertebereich* (*Domäne*) des Merkmals.
- Jede Tabelle benötigt ein Merkmal (oder eine „minimale“ Merkmalskombminationen), das (die) jeden Datensatz eindeutig indentifiziert. Ein solcher Schlüssel wird *Identifikationsschlüssel* genannt und durch Kursivdruck oder Unterstreichen hervorgehoben.

3.2 Abbildungsregeln

Abbildungsregel 1

- Jede Entitätsmenge muss als eigenständige Tabelle mit einem eindeutigen Primärschlüssel und einem eindeutigen Namen definiert werden.
- Als Primärschlüssel der Tabelle dient entweder ein Attribut oder eine Attributs-Kombination.
- Die übrigen Merkmale der Entitätsmengen werden zu Attributen der Tabelle.

Beispiel zur Abbildungsregel 1



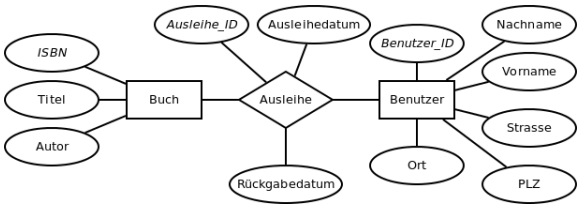
BUCH

ISBN	Titel	Autor
...	...	...

Abbildungsregel 2

- Jede Beziehungsmenge kann als eigenständige Tabelle definiert werden, wobei die Primärschlüssel der zugehörigen Entitätsmengen als sogenannte Fremdschlüssel in dieser Tabelle auftreten müssen.
- Der Primärschlüssel der Beziehungsmengentabelle kann der aus den Fremdschlüsseln zusammengesetzte Schlüssel oder ein künstlicher Schlüssel sein.
- Weitere Merkmale der Beziehungsmenge erscheinen als zusätzliche Attribute in der Tabelle.

Beispiel zur Abbildungsregel 2



BUCH

ISBN	Titel	Autor
...	...	...

AUSLEIHE

Ausleihe_ID	ISBN	Benutzer_ID	Ausgabe	Rückgabe
...	...	...	...	...

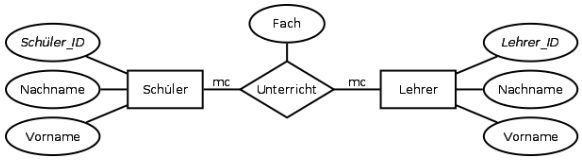
BENUTZER

	Nachname	Vorname	...
...	...	...	...

Abbildungsregel 3

- Jede komplex-komplexe Beziehungsmenge (komplex bedeutet: m oder mc) muss als eigenständige Tabelle definiert werden.
- Die Primärschlüssel der zugehörigen Entitätsmengen treten dabei als Fremdschlüssel auf.
- Der Primärschlüssel der Beziehungsmengentabelle ist entweder der aus den Fremdschlüsseln zusammengesetzte Schlüssel oder ein künstlicher Schlüssel.
- Weitere Merkmale der Beziehungsmenge erscheinen als zusätzliche Attribute in der Tabelle.

Beispiel zur Abbildungsregel 3



SCHÜLER

Schüler_ID	Nachname	Vorname
...	...	...

UNTERRICHT

Unterricht_ID	Schüler_ID	Lehrer_ID	Fach
...	...	...	...

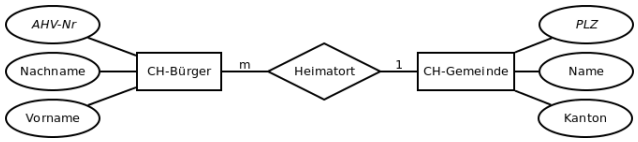
LEHRER

Lehrer_ID	Nachname	Vorname
...	...	...

Abbildungsregel 4

- Eine einfach-komplexe Beziehungsmenge (einfach bedeutet: 1 oder c) kann ohne eine eigenständige Beziehungsmengentabelle durch die beiden Tabellen der zugeordneten Entitätsmengen ausgedrückt werden.
- Dazu wird in der Tabelle mit der einfachen Kardinalität ein Fremdschlüssel auf die referenzierte Tabelle mit eventuell weiteren Merkmalen der Beziehungsmenge geführt.

Beispiel zur Abbildungsregel 4



CH-BÜRGER

AHV-Nr	Nachname	Vorname	PLZ_Heimatort
...	...	...	...

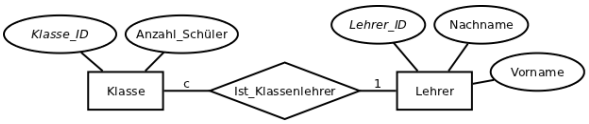
CH-GEMEINDE

PLZ	Name	Kanton
...	...	...

Abbildungsregel 5

- Eine einfach-einfache Beziehungsmenge kann ohne eine eigenständige Tabelle durch die beiden Tabellen der zugeordneten Entitätsmengen ausgedrückt werden, indem einer der Primärschlüssel der referenzierten Tabelle als Fremdschlüssel in die andere Tabelle eingebracht wird.
- Normalerweise fügen wir in die Tabelle mit der Kardinalität „1“ den Fremdschlüssel der referenzierten Tabelle ein.

Beispiel zur Abbildungsregel 5



KLASSE

Klasse_ID	Anzahl_Schüler	Lehrer_ID
...	...	...

LEHRER

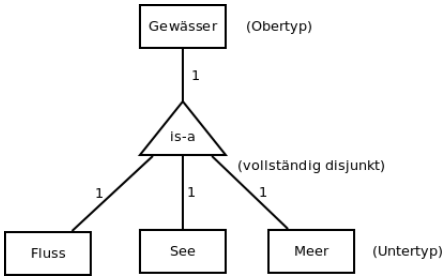
Lehrer_ID	Nachname	Vorname
...	...	...

Würde man umgekehrt den Primärschlüssel der Tabelle KLASSE als Fremdschlüssel in der Tabelle LEHRER einsetzen, so müsste man bei den Lehrern, die nicht Klassenlehrer sind, in der Spalte „Klassen_ID“ einen Nullwert einsetzen. Nullwerte sind in relationalen Datenbanken jedoch nicht unproblematisch.

Abbildungsregel 6

- Unter einer Generalisation versteht man ein Abstraktionsvorgang, bei dem einzelne Entitätsmengen zu einer übergeordneten Entitätsmenge verallgemeinert werden. Beispielsweise sind sowohl ein Manager, ein Lehrling und ein Fachspezialist Mitarbeiter einer bestimmten Firma. Umgekehrt lassen sich die in einer Generalisationshierarchie abhängigen Subentitätsmengen als Spezialisierung interpretieren.
- Jede Entitätsmenge einer Generalisationshierarchie verlangt eine eigenständige Tabelle, wobei der Primärschlüssel der übergeordneten Tabelle auch zum Primärschlüssel der untergeordneten Tabellen wird.
- Wenn sich die Subentitätsmengen der Spezialisierung gegenseitig ausschliessen (d. h. gegenseitig disjunkt sind), führen wir in der übergeordneten Tabelle z. B. das Merkmal „Kategorie“ ein, in welcher ein bestimmter Mitarbeiter nicht mehreren Kategorien gleichzeitig angehört.

Beispiel zur Abbildungsregel 6



- Fluss *is a* Gewässer (Generalisierung)
- Gewässer *can be a* Fluss (Spezialisierung)

GEWÄSSER

GewässerID	GewässerName	Kategorie
...	...	...

FLUSS

GewässerID	Länge
...	...

SEE

GewässerID	Fläche	Tiefe
...	...	...

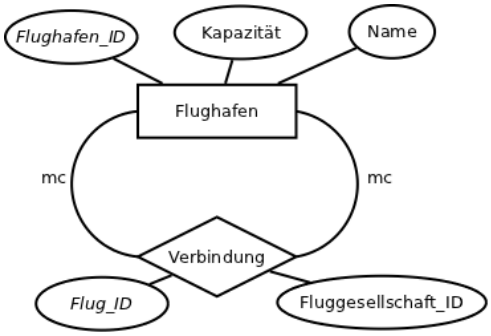
MEER

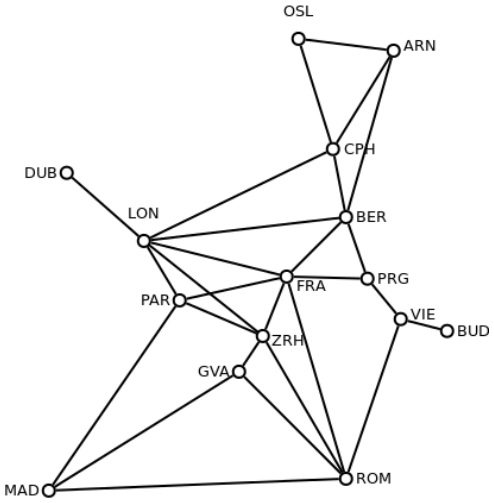
GewässerID	Fläche	Tiefe
...	...	...

Abbildungsregel 7

- Bei einer Aggregation (Vereinigung, Zusammenführen von Einzeldaten) müssen sowohl die Entitätsmenge als auch die Beziehungsmenge je als eigenständige Tabelle definiert werden, falls der Beziehungstyp komplex-komplex ist (= netzwerkartige Aggregation, z. B. das Fluglinniennetz). Die Tabelle der Beziehungsmenge enthält in diesem Fall zweimal den Schlüssel aus der Tabelle der zugehörigen Entitätsmenge als zusammengesetzten Schlüssel, mit entsprechenden *Rollennamen*.

Beispiel zur Abbildungsregel 7





FLUGHAFEN

FlughafenID	Name	Kapazität
...	...	...

VERBINDUNG

FlugID	FlughafenID.ab	FlughafenID.an	AirlineID
...	...	...	...

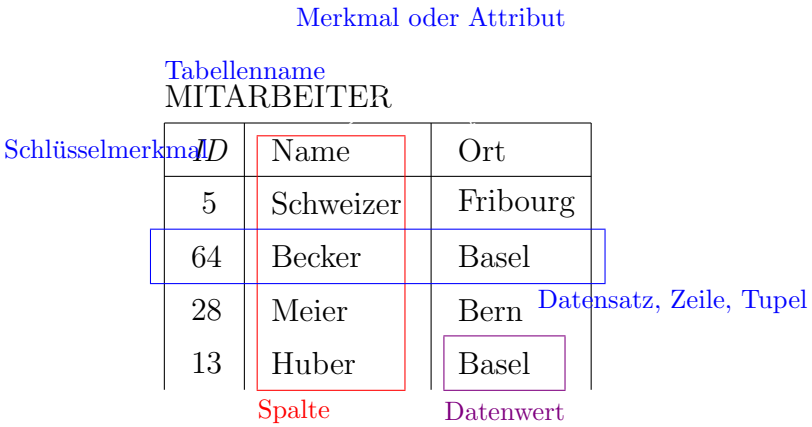
Im Falle einer einfach-komplexen Beziehung (=hierarchische Aggregation, z. B. die Stückliste) kann die Entitätsmenge mit der Beziehungsmenge zu einer einzigen Tabelle kombiniert werden. Beispielsweise könnten die beiden Tabellen „Artikel“ und „Stückliste“ in einer einzigen Tabelle „Artikelstruktur“ zusammengefasst werden. Dabei würde man zu den Artikeleigenschaften je die Artikelnummer des eindeutig übergeordneten Artikels aufführen.

4 Relationenalgebra

4.1 Relationen

- Der Begriff der Relation kann mathematisch präzise definiert werden.
- Für uns genügt es zu wissen: *Relation = Tabelle*
- Wir werden sehen, wie man mit Relationen, d. h. mit Tabellen sinnvoll operieren kann.

Tabellen (Repetition)



4.2 Mengenorientierte Operatoren

Übersicht

- Vereinigung von Tabellen ($R \cup S$)
- Durchschnitt von Tabellen ($R \cap S$)
- Differenz von Tabellen ($R \setminus S$)
- Kartesisches Produkt von Tabellen ($R \times S$)

Für die ersten drei Operatoren müssen die Tabellen vereinigungsverträglich sein.

Zwei Tabellen sind *vereinigungsverträglich*, wenn sie folgende Eigenschaften haben.

- Beide Tabellen haben die gleiche *Anzahl* Merkmale.
- Die Datentypen korrespondierender Spalten sind identisch.

Aufgabe

Sind die folgenden Tabellen vereinigungsverträglich?

WAHLPFLICHTFACH

WID	Titel	Lehrperson
41	Geschichte der Neuzeit	A. Meier
65	Die Mathematik der Antike	L. Euler
19	Volleyball II	S. Portmann

ABENDKURS

AID	Titel	Kursleitung
108	Word für Fortgeschrittene	R. Ratlos
459	Metallbearbeitung	H. Phaistos
98	Advanced	T. Beutel

4.2.1 Der Vereinigungsoperator (union)

Zwei vereinigungsverträgliche Tabellen R und S werden mengentheoretisch vereinigt, indem sämtliche Einträge aus R und sämtliche Einträge aus S in die Resultattabelle eingefügt werden. Gleichzeitig werden identische Datensätze eliminiert.

ADRESSEN1

ID	Name	Vorname
12	Meier	Andreas
47	Müller	Michael
39	Schmid	Susanne

ADRESSEN2

ID	Name	Vorname
65	Bischof	Maria
39	Schmid	Susanne

ADRESSEN1 $\cup$ ADRESSEN2

ID	Name	Vorname
12	Meier	Andreas
47	Müller	Michael
39	Schmid	Susanne
65	Bischof	Maria

4.2.2 Der Durchschnittsoperator (intersection)

Zwei vereinigungsverträgliche Tabellen R und S werden geschnitten, indem sämtliche Einträge, die sowohl in R als auch in S vorhanden sind, in die Resultattabelle aufgenommen werden.

ADRESSEN1

ID	Name	Vorname
12	Meier	Andreas
47	Müller	Michael
39	Schmid	Susanne

ADRESSEN2

ID	Name	Vorname
65	Bischof	Maria
39	Schmid	Susanne

ADRESSEN1 ∩ ADRESSEN2

ID	Name	Vorname
39	Schmid	Susanne

4.2.3 Der Subtraktionsoperator \ (difference)

Sind R und S zwei vereinigungsverträgliche Tabellen, so wird die Differenz $R \setminus S$ gebildet, indem man aus R sämtliche Einträge entfernt, die in S enthalten sind.

ADRESSEN1

ID	Name	Vorname
12	Meier	Andreas
47	Müller	Michael
39	Schmid	Susanne

ADRESSEN2

ID	Name	Vorname
65	Bischof	Maria
39	Schmid	Susanne

ADRESSEN1 \ ADRESSEN2

ID	Name	Vorname
12	Meier	Andreas
47	Müller	Michael

4.2.4 Das kartesische Produkt

- Unter dem kartesischen Produkt $R \times S$ versteht man die Menge aller möglichen Kombinationen aus Tupeln aus R mit Tupeln aus S.
- Für das kartesische Produkt müssen die betrachteten Tabellen *nicht* vereinigungsverträglich sein.

ADRESSEN

AID	Name	Vorname
12	Meier	Andreas
47	Müller	Michael
39	Schmid	Susanne

HOBBY

HID	Bezeichnung
1	Wandern
2	Schwimmen

FREIZEIT = ADRESSEN × HOBBY

AID	Name	Vorname	*HID*	Bezeichnung
12	Meier	Andreas	1	Wandern
12	Meier	Andreas	2	Schwimmen
47	Müller	Michael	1	Wandern
47	Müller	Michael	2	Schwimmen
39	Schmid	Susanne	1	Wandern
39	Schmid	Susanne	2	Schwimmen

4.3 Die relationenorientierten Operatoren

Übersicht

- Die relationenorientierten Operatoren ergänzen die mengenorientierten Operatoren
- Wie beim kartesischen Produkt werden *keine* vereinigungsverträglichen Tabellen vorausgesetzt.

Diese Operatoren werden nachfolgende besprochen:

- Die Projektion einer Tabelle R auf ein Merkmal M
- Die Selektion einer Zeile aus einer Tabelle R anhand der Formel F
- Der Verbund zweier Tabellen R und S über das Prädikat P

Ein *Prädikat* ist eine Funktion, die einen Wahrheitswert (*wahr* oder *falsch*) zurückliefert.

4.3.1 Der Projektionsoperator $\pi_M(R)$

- Der Projektionsoperator bildet mit den in M angegebenen Merkmalsnamen die Tabelle R auf eine Teiltabelle ab.
- Die Merkmalsnamen dürfen in einer beliebigen Reihenfolge aufgelistet werden.

ADRESSEN

ID	Name	Vorname	PLZ	Ort
12	Meier	Andreas	6370	Stans
47	Müller	Michael	8000	Zürich
39	Schmid	Susanne	3000	Bern

π_{Vorname}(ADRESSEN)

Vorname
Andreas
Michael
Susanne

$\pi_{\text{Ort,Name}}$(ADRESSEN)

Ort	Name
Stans	Meier
Zürich	Müller
Bern	Schmid

4.3.2 Der Selektionsoperator $\sigma_F(R)$

- Unter $\sigma_F(R)$ versteht man alle Tupel aus R, welche die Selektionsbedingung F erfüllen.
- Eine Selektionsbedingung F besteht aus einer bestimmten Anzahl von Merkmalsnamen oder konstanten Werten, die durch Vergleichsoperatoren wie $<$, $>$ oder $=$ sowie durch logische Operatoren wie AND, OR und NOT miteinander kombiniert werden können.

ADRESSEN

ID	Name	Vorname	PLZ	Ort
12	Meier	Andreas	6370	Stans
47	Müller	Michael	8000	Zürich
39	Schmid	Susanne	3000	Bern
23	Meier	Stefan	6000	Luzern

$\sigma_{\text{Name=Meier}}$(ADRESSEN)

ID	Name	Vorname	PLZ	Ort
12	Meier	Andreas	6370	Stans
23	Meier	Stefan	6000	Luzern

$\sigma_{\text{Name=Meier AND PLZ<6100}}$(ADRESSEN)

ID	Name	Vorname	PLZ	Ort
23	Meier	Stefan	6000	Luzern

4.3.3 Der Verbundoperator $R \bowtie_P S$

- Der Verbundoperator der beiden Tabellen R und S über das Prädikat P ist die Menge aller Tupel aus dem kartesischen Produkt $R \times S$, die das Verbundprädikat P erfüllen.
- Das Verbundprädikat P enthält je ein Merkmal aus der Tabelle R und eines aus S. Diese beiden Merkmale werden durch die Vergleichsoperatoren $<$, $>$ und $=$ in Beziehung gesetzt, damit die Tabellen kombiniert werden können.
- Enthält das Verbundprädikat P den Vergleichsoperator $=$, so spricht man von einem *Gleichheitsverbund*. (engl. *equi-join*)
- Lässt man das Verbundprädikat weg ($P = \{\}$), so erhält man als Spezialfall das kartesische Produkt: $R \bowtie_{P=\{\}} S = R \times S$.

KUNDE

KID	Name	Ort
21312	Meier	Stans
5945	Müller	Zürich
45101	Schmid	Bern

EINKAUF

EID	Datum	KID	Betrag
135	3.9.2011	92415	123.70
136	4.9.2011	5945	95.00
137	7.9.2011	21312	69.35

KUNDE $\bowtie_{\text{KUNDE.KID=EINKAUF.KID}}$ EINKAUF

KID	Name	Ort	EID	Datum	KID	Betrag
21312	Meier	Stans	137	7.9.2011	21312	69.35
5945	Müller	Zürich	136	4.9.2011	5945	95.00

4.3.4 Der Verbundoperator $R \bowtie_P S$ (Fortsetzung)

KUNDE × EINKAUF

KID	Name	Ort	EID	Datum	KID	Betrag
21312	Meier	Stans	135	3.9.2011	92415	123.70
21312	Meier	Stans	136	4.9.2011	5945	95.00
21312	*Meier*	*Stans*	*137*	*7.9.2011*	*21312*	*69.35*
5945	Müller	Zürich	135	3.9.2011	92415	123.70
5945	*Müller*	*Zürich*	*136*	*4.9.2011*	*5945*	*95.00*
5945	Müller	Zürich	137	7.9.2011	21312	69.35
45101	Schmid	Bern	135	3.9.2011	92415	123.70
45101	Schmid	Bern	136	4.9.2011	5945	95.00
45101	Schmid	Bern	137	7.9.2011	21312	69.35

KUNDE $\bowtie_{\text{KUNDE.KID=EINKAUF.KID}}$ EINKAUF

KID	Name	Ort	EID	Datum	KID	Betrag
21312	*Meier*	*Stans*	*137*	*7.9.2011*	*21312*	*69.35*
5945	*Müller*	*Zürich*	*136*	*4.9.2011*	*5945*	*95.00*

Quelle

- Andreas Meier, *Relationale und Postrelationale Datenbanken*, Springer, 2007

5 SQLite

5.1 Relationales Datenbank Managment System

RDBMS

- Für die praktische Arbeit mit den Daten muss das *relationale Datenmodell* als *relationale Datenbank* implementiert werden.
- Ein Datenbankprogramm, das nicht nur das Speichern und Abfragen sondern auch das Verwalten der Daten und der Benutzer ermöglicht, wird *Relationales Datenbank Management System (RDBMS)* genannt.

Aufgaben eines RDBMS

- Datenbanken anlegen
- Tabellen erstellen und löschen
- Datensätze in Tabellen einfügen
- Datensätze aus Tabellen löschen
- Datenbankabfragen tätigen
- DB-Benutzer verwalten (Passwörter, Berechtigungen, ...)
- Transaktionsüberwachung ($\rightarrow$ ACID)

5.2 Structured Query Language

In der Praxis existieren viele verschiedene RDBMS. Ihre Anwendungsschnittstelle ist in der Regel die Datenbanksprache *Structured Query Language (SQL)*.

Auch wenn der SQL-Sprachkern standardisiert ist, gibt es je nach Datenbank(-Hersteller) verschiedene SQL-Dialekte:

- IBM DB2 (kommerziell)
- Oracle Database (kommerziell und eingeschränkte Version für kostenlose Nutzung)
- Microsoft SQL Server (kommerziell)
- MySQL (Open Source und kommerziell)
- PostgreSQL (Open Source)
- SQLite (Open Source)
- ...

5.3 SQLite

SQLite ist ein kostenloses leichtgewichtiges RDBMS, das im Gegensatz zu seinen grossen Geschwistern

- keinen Server benötigt,
- ohne Konfiguration auskommt,
- auf vielen Plattformen verfügbar ist,
- in einer einzelnen Datei integriert ist,
- sparsam mit Speichplatz umgeht.

Dennoch unterstützt SQLite einen grossen Teil des SQL92-Standards und ist sehr zuverlässig.

5.3.1 Information und Download

`http://www.sqlite.org/`

5.3.2 GUI

Für SQLite gibt es auch grafische Benutzerschnittstellen (GUI), die die Arbeit mit dem RDBMS vereinfachen.

Wir werden dafür das Firefox Add-On *SQLite Manager* verwenden. Dieses Programm verwendet den Browser als GUI und erlaubt das Speichern der Datenbank im lokalen Dateisystem.

5.3.3 Datentypen

Jeder Wert, der in einer SQLite-DB gespeichert oder von ihr verändert wird, gehört zu einer der folgenden Datentypen:

- **NULL** der NULL-Wert ($\neq 0$)
- **INTEGER** eine ganze Zahl mit Vorzeichen (je nach Grössenordnung 1, 2, 3, 4, 6, oder 8 Bytes)
- **REAL** Fliesskommazahl (8 Byte, IEEE 754-Format)
- **TEXT** Zeichenkette (je nach DB-Kodierung UTF-8, UTF-16BE oder UTF-16LE)
- **BLOB** (Binary Large Object) exakte Kopie des Inputs

5.3.4 Datum und Zeit

Datum- und Zeitangaben werden am einfachsten als `TEXT` gespeichert. Es sind auch Darstellungen als `REAL` möglich, die jedoch Umrechnungen erfordern.

Hier einige Darstellungen, die von SQLite erkannt werden:

- *YYYY-MM-DD*
- *YYYY-MM-DD HH:MM*
- *YYYY-MM-DD HH:MM:SS*
- *YYYY-MM-DD HH:MM:SS.sss*
- *HH:MM*
- *HH:MM:SS*
- *HH:MM:SS.sss*
- ...

5.3.5 CREATE TABLE

Beispiel

```
CREATE TABLE personal (
   pid       INTEGER PRIMARY KEY AUTOINCREMENT,
   nachname  TEXT NOT NULL,
   vorname   TEXT CHECK(vorname != 'Konstanze'),
   eintritt  TEXT NOT NULL DEFAULT CURRENT_DATE,
   garderobe INTEGER UNIQUE
);
```

Beispiel

```
CREATE TABLE wohnungen (
   wohnungsnummer INTEGER NOT NULL,
   gebaeudenummer INTEGER NOT NULL,
   PRIMARY KEY (wohnungsnummer, gebaeudenummer)
);
```

5.3.6 DROP TABLE

```
DROP TABLE [IF EXISTS] db_name.table_name;
```

Beispiel

```
DROP TABLE IF EXISTS personal;
```

5.3.7 INSERT INTO

```
INSERT INTO db_name.tab_name (col1, col2, ...)
VALUES (val1, val2, ...)
```

Beispiel

```
INSERT INTO personal (pid, nachname, vorname, eintritt, garderobe)
VALUES (11, 'Meier', 'Andreas', '04-06-2009', 17);
```

Beispiel

```
INSERT INTO personal (eintritt, nachname, garderobe, vorname)
VALUES ('25-11-2018', 'Bernasconi', 39, 'Maria');
```

5.3.8 ALTER TABLE

Übliche Verwendung:

```
ALTER TABLE db_name.table_name
RENAME TO new_table_name
```

und

```
ALTER TABLE db_name.table_name
ADD COLUMN column_def
```

5.3.9 DELETE

```
DELETE FROM db_name.table_name;
```

oder

```
DELETE FROM db_name.table_name
WHERE condition;
```

5.3.10 ACID-Transaktionen

- **A**tomic: *Die Transaktion darf nicht in kleinere Teile zerlegt werden.*
- **C**onsistent: *Die Transaktion muss die DB in einem konsistenten (widerspruchsfreien) Zustand belassen.*
- **I**solated: *Die Transaktion muss von denen anderer Clients abgeschirmt werden.*
- **D**urable: *Nach dem erfolgreichen Abschluss der Transaktion muss diese ein permanenter und unwiderruflichr Teil der DB werden.*

BEGIN/COMMIT TRANSACTION

Normalerweise befindet sich SQLite im *autocommit*-Modus: jede Eingabe wird implizit als einzelne Transaktion verarbeitet.

Problem: *Verlangsamt die Ausführung von SQL-Befehlen*

```
BEGIN [DEFERRED|IMMEDIATE|EXCLUSIVE] [TRANSACTION]
DB-Befehle
COMMIT [TRANSACTION] oder END [TRANSACTION]
```

5.3.11 Die SELECT-Pipeline

Überblick

```
SELECT [DISTINCT] select_heading
FROM source_tables
WHERE filter_expression
GROUP BY grouping_expressions
HAVING filter_expression
ORDER BY ordering_expressions
LIMIT count
OFFSET count
```

Auswertungsreihenfolge

1. `FROM` *source_tables*

   Legt eine oder mehrere Ausgangstabellen fest und kombiniert sie zu einer grossen Arbeitstabelle.

2. `WHERE` *filter_expressions*

   Filtert bestimmte Zeilen aus der Arbeittstabelle heraus.

3. `GROUP BY` *grouping_expressions*

   Fasst Zeilen zusammen, die in bestimmten Werten übereinstimmen.

4. `SELECT` *select_heading*

   Definiert die Kolonnen der Resultatmenge sowie die gruppierenden Funktionen (Aggregatsfunktionen), sofern diese anwendbar sind.

5. `HAVING` *filter_expression*

   Filtert bestimmte Zeilen aus der gruppierten Tabelle. Verlangt ein `GROUP BY`.

6. `DISTINCT`

   Eliminiert identische Zeilen.

7. `ORDER BY` *ordering_expressions*

   Sortiert die Zeilen der Resultatmenge.

8. `OFFSET` *`count`*

   Überspringt *`count`* Zeilen am anfang der Resultatmenge. Verlangt ein `LIMIT`.

9. `LIMIT` *`count`*

   Begrenzt die Anzahl der auszugebenden Zeilen der Resultatmenge auf *`count`*.

5.3.12 Der FROM-Abschnitt

Für die Kombination von Ausgangstabellen gibt es folgende Möglichkeiten:

CROSS JOIN

Ein ungefiltertes Kreuzprodukt der Tabellen. *Achtung!*

a	b
1	2
3	7
4	5

$\times$

c	d
2	7
1	2

$=$

a	b	c	d
1	2	2	7
1	2	1	2
3	7	2	7
3	7	1	2
4	5	2	7
4	5	1	2

```
SELECT * FROM tab1 CROSS JOIN tab2;
```

[INNER] JOIN

Ein Kreuzprodukt, das durch eine Bedingung gefiltert wird, die nach dem Schlüsselowrt `ON` steht.

a	b
1	2
3	7
4	5

$\bowtie_{b=c}$

c	d
2	7
1	2

$=$

a	b	c	d
1	2	2	7

```
SELECT * FROM tab1 INNER JOIN tab2 ON b=c;
```

Da ein `INNER JOIN` häufig verwendet wird, kann man dafür die Abkürzung `JOIN` verwenden.

NATURAL JOIN

Filtert im Kreuzprodukt automatisch nach Kolonnen mit identischen Kolonnenamen. Dies ist bequem, kann aber gefährlich werden, wenn man seine Kolonnen unsorgfältig benennt.

a	b
1	2
3	7
4	5

$\bowtie$

c	b
2	7
1	2

$=$

a	b	c
1	2	1
3	7	2

```
SELECT * FROM tab1 NATURAL JOIN tab2;
```

Ein `NATURAL JOIN` führt Kolonnen mit gemeinsamen Merkmalsnamen nur einmal auf.

LEFT OUTER JOIN

Ergänzt einen `INNER JOIN` um diejenigen Zeilen in der ersten Tabelle, zu denen es keine passenden Zeilen in der zweiten Tabelle gibt und füllt die Kollonnen mit NULL-Werten auf.

a	b
1	2
3	7
4	5

$\rtimes_{b=d}$

c	d
2	7
1	2

=

a	b	c	d
1	2	1	2
3	7	2	7
4	5	NULL	NULL

```
SELECT * FROM tab1 LEFT OUTER JOIN tab2 on b=d;
```

5.3.13 Der WHERE-Abschnitt

Nach `WHERE` kann man eine (oder mehrere) Bedingung(en) angeben, nach denen man die Resultattabelle filtert (*Selektion*). Beispielsweise mit den Vergleichsoperatoren (`=`, `!=`, `<`, `<=`, `>`, `>=`, `BETWEEN ... AND ...`), den logischen Operatoren (`AND`, `OR`, `NOT`) oder mit `LIKE`.

```
SELECT a, c FROM tab WHERE b >= 5;
```

a	b	c
3	5	7
9	3	1
2	6	4

⇒

a	c
3	7
2	4

5.3.14 Der GROUP BY-Abschnitt

`GROUP BY` fasst Zeilen zusammen, die in bestimmten Werten übereinstimmen.

Üblicherweise wendet man auf die gruppierten Zeilen im SELECT-Teil eine der Aggregatsfunktionen `count(...)`, `sum(...)`, `min(...)`, `max(...)` oder `avg(...)` auf eines der Attribute an.

```
SELECT a, b, sum(c) FROM tab GROUP BY a;
```

a	b	c
3	12	7
1	27	5
3	30	4
2	14	9
2	19	8
1	32	0

⇒

a	b	c
3	12	7
3	30	4
1	27	5
1	32	0
2	14	9
2	19	8

⇒

a	b	c
1	32	5
2	19	17
3	30	11

Die mittlere Tabelle stellt einen Zwischenschritt in der Berechnung dar und wird nicht ausgegeben.

Achtung: Einträge, die weder von der Zusammenfassung noch von der Aggregatsfunktion betroffen sind (hier: Attribut *b*), erhalten jeweils den Eintrag der letzten Zeile in der Gruppe. Sie sind von der Reihenfolge der Daten abhängig und damit nicht vorhersehbar.

5.3.15 Die SELECT-Kopfzeile

Die SELECT-Kopfzeile dient dazu, das Format und den Inhalt der Resultattabelle zu definieren.

`SELECT` *`expression`* `[AS` *`column_name`*`] [,...]`

Mit dem Optionalen Argument `AS` kann ein Alias definiert werden, der dann in Resultattabelle erscheint. Wenn der Aliasname Leerzeichen enthält, so muss er von Anführungs- und Schlusszeichen eingeschlossen sein.

Als *`expression`* kann auch ein Stern (`*`) verwendet werden, der jede Kolonne im Resultat anzeigt.

5.3.16 Der HAVING-Abschnitt

Die `HAVING`-Klausel ist im Grunde identisch mit der `WHERE`-Klausel.

Der Unterschied besteht darin, dass die `WHERE`-Klausel *vor* einem `GROUP BY` und die `HAVING`-Klausel nach einem `GROUP BY` ausgewertet wird.

Deshalb sollte man die `HAVING`-Klausel nur als Filterinstrument nach einem `GROUP BY` verwenden.

Beispiel

```
SELECT a, sum(c) AS summe FROM tab
GROUP BY a
HAVING summe > 10;
```

a	b	c
3	12	7
1	27	5
3	30	4
2	14	9
2	19	8
1	32	0

⇒

a	summe
3	11
1	5
2	17

⇒

a	summe
2	17
3	11

5.3.17 Das DISTINCT-Schlüsselwort

Mit dem Schlüsselwort `DISTINCT` werden in der Resultatabelle identischen Zeilen allfällige duplizierte Zeilen entfernt.

Beachte: Da SQLite alle Zeilen paarweise auf Identität prüfen muss, ist es eine *teure* Operation, die bei grossen Tabellen einen grossen Zeitaufwand verursachen kann.

Beispiel

```
SELECT DISTINCT a from tab;
```

a	b	c
3	6	9
1	4	8
3	5	7
2	4	9
2	6	7
1	5	8

⇒

a
3
1
2

5.3.18 Der ORDER BY-Abschnitt

Die `ORDER BY`-Klausel dient dazu, die Zeilen der Resultattabelle zu sortieren bzw. zu ordnen.

`ORDER BY` *`expression`*

`[COLLATE` *`collation_name`*`] [ASC|DESC] [,...]`

`COLLATE` bezeichnet die vorgegebene Ordnung in der die Symbole des Zeichensatzes verglichen werden. Beispielsweise werden mit `COLLATE NOCASE` alle Grossbuchstaben in Kleinbuchstaben verwandelt.

Ohne Angabe einer Sortierreihenfolge wird standardmässig `ASC` (aufsteigend) verwendet.

Beispiel

```
SELECT * FROM tab
ORDER BY a ASC, b DESC;
```

a	b	c
3	6	9
1	4	8
3	5	7
2	4	9
2	6	7
1	5	8

⇒

a	b	c
1	5	8
1	4	8
2	6	7
2	4	9
3	6	9
3	5	7

5.3.19 Die LIMIT- und OFFSET-Abschnitte

`LIMIT` und `OFFSET` ermöglichen eine spezielle Teilmenge der Resultattabelle anzuzeigen.

`LIMIT` beschränkt die Anzahl der ausgegebenen Zeilen.

`OFFSET` sagt, wie viele Zeilen vor der ersten Ausgabe *weggelassen* werden sollen.

Beispiele:

`LIMIT 10`	gibt die Zeilen 1–10 aus
`LIMIT 10 OFFSET 3`	gibt die Zeilen 4–13 aus
`LIMIT 3 OFFSET 20`	gibt die Zeilen 21–23 aus

Achtung: Es gibt noch eine Abkürzung, bei der OFFSET implizit an erster Stelle von `LIMIT` steht.

`LIMIT 3, 20` gibt die Zeilen 4–23 aus

5.3.20 Aggregatsfunktionen

- `count(`*`col`*`)`: Anzahl der Zeilen in *`col`*
- `min(`*`col`*`)`: kleinster Wert in *`col`*
- `max(`*`col`*`)`: grösster Wert in *`col`*
- `sum(`*`col`*`)`: Summe aller Werte in *`col`*

- `avg(col)`: Durchschnitt der Werte in *col*

Darüber hinaus kann man die üblichen Operatoren `+`, `-`, `*` und `/` zum Operieren auf den Kolonnen verwenden.

5.3.21 Subqueries (Unterabfragen)

Anstelle einer gegebenen Tabelle (nach `FROM`) darf auch ein weiteres `SELECT`-Statement stehen, das in runde Klammern eingeschlossen werden muss.

```
SELECT ... FROM (SELECT ...) WHERE ...;
```

5.3.22 Mengenoperationen

Die Resultate von jeweils zwei `SELECT`-Ausdrücken können durch die folgenden Mengenoperationen im Sinne der Mengenalgebra verknüpft werden, sofern sie vereinigungsverträglich sind.

- `SELECT ... UNION SELECT ...`

  Bildet die Vereinigungsmenge ($\cup$) der Resultate der beiden `SELECT`-Ausdrücke.

- `SELECT ... INTERSECT SELECT ...`

  Bildet die Durchschnittsmenge ($\cap$) der Resultate der beiden `SELECT`-Ausdrücke.

- `SELECT ... EXCEPT SELECT ...`

  Bildet die Mengendifferenz ($\setminus$) von der ersten und der zweiten Resultatmenge.