Zürcher Hochschule
für Angewandte Wissenschaften

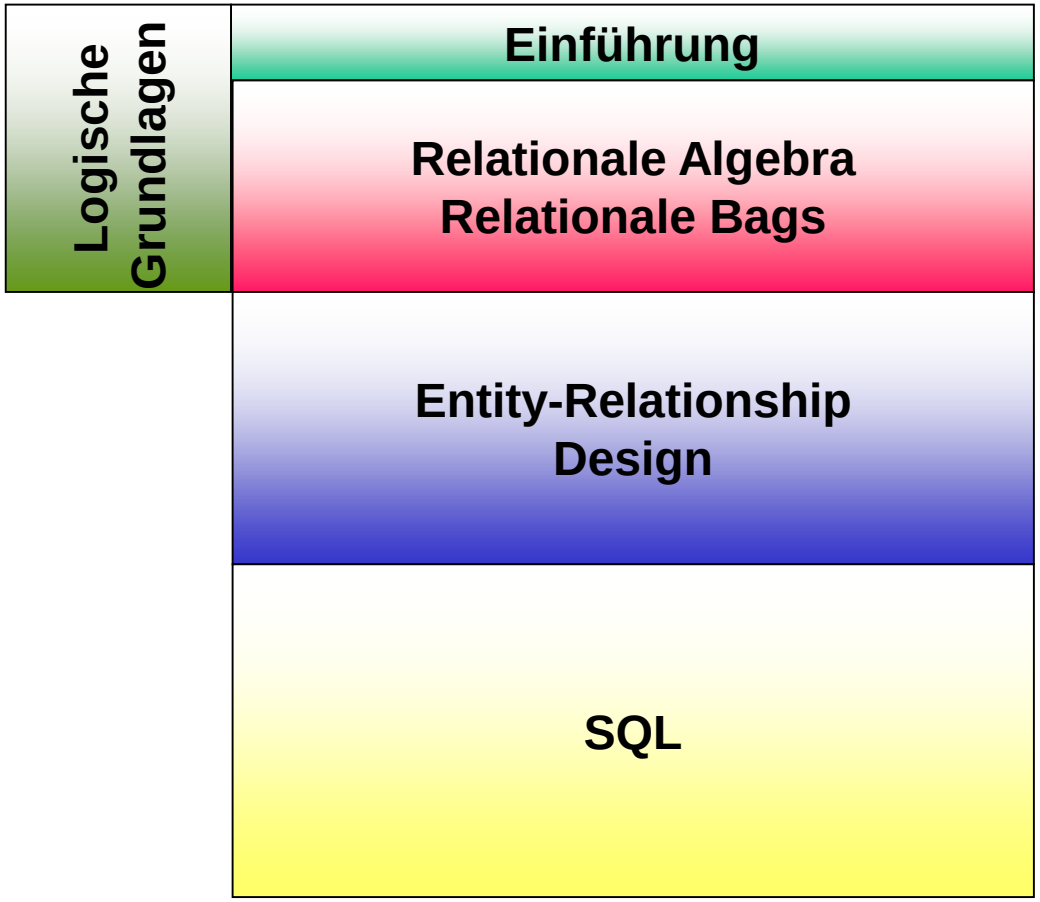
School of
Engineering
InIT Institut für angewandte
Informationstechnologie

DAB1 – Datenbanken 1

Dr. Daniel Aebi (aebd@zhaw.ch)

Lektion 2: Relationenmodell, relationale Algebra

Wo stehen wir?



"You are here"

Rückblick

Diskutiert im Unterricht. Machen Sie Ihre eigenen Notizen.

Lernziele Lektion 2

- Bedeutung relationaler Datenbanken für die Praxis kennen.
- Ursprung des relationalen Modelles kennen.
- Begriffe kennen: Relation, Tupel, Attribut, Wertebereich.
- Erste Grundoperationen auf Relationen verstehen.

Datenbanksysteme – Entwicklung

- Ganz verschiedene Datenmodelle & Technologien im Einsatz:
  - Hierarchisch (ab 70er Jahre, z.B. IMS von IBM)
  - Netzwerk (ab 70er Jahre, z.B. IDMS von CA, UDS von Siemens)
  - Relational (ab 80er Jahre)
  - Objekt-Relational
  - Objektorientiert
  - Logikbasiert
  - Deduktiv
  - XML
  - NoSQL
  - ...

Begriff: Strukturierte Daten

- Daten, die eine reguläre, fest vorgegebene Struktur besitzen. Mehrere «gleichartige Datensätze» mit identischem Aufbau. Gut «tabellarisch» darstellbar.

  Synonym: formatierte Daten.

- Beispiel:

	Name	Letz.	Veränderung	Hoch	Tief	Volumen
	SMI PR	9'220.69	▼ -70.23 (-0.76 %)	9'272.50	9'220.69	-
Trade	ABB LTD N	25.67	▼ -0.36 (-1.38 %)	26.03	25.67	4'970'180
Trade	ADECCO N	75.30	▼ -1.08 (-1.41 %)	76.24	75.30	911'282
Trade	CS GROUP N	18.06	▼ -0.18 (-0.99 %)	18.22	17.955	8'142'755
Trade	GEBERIT N	435.60	▼ -3.90 (-0.89 %)	438.80	435.10	113'461
Trade	GIVAUDAN N	2'179.00	▼ -35.00 (-1.58 %)	2'216.00	2'178.00	35'706
Trade	JULIUS BAER N	63.58	▼ -0.80 (-1.24 %)	64.50	63.48	1'047'244
Trade	LAFARGEHOLCIM N	56.28	▼ -0.92 (-1.61 %)	57.30	56.28	2'152'747
Trade	LONZA N	251.60	▼ -0.40 (-0.16 %)	252.40	248.60	631'566
Trade	NESTLE N	79.42	▼ -1.00 (-1.24 %)	80.14	79.10	6'063'516
Trade	NOVARTIS N	82.64	▼ -0.88 (-1.05 %)	83.56	82.40	4'875'390
Trade	RICHEMONT N	88.78	▼ -1.14 (-1.27 %)	89.60	88.68	1'356'277
Trade	ROCHE GS	224.15	▲ 1.05 (0.47 %)	226.00	221.15	2'780'105
Trade	SGS N	2'469.00	▼ -29.00 (-1.16 %)	2'494.00	2'468.00	13'304
Trade	SIKA I	7'820.00	▼ -180.00 (-2.25 %)	8'010.00	7'820.00	8'814
Trade	SWATCH GROUP I	417.20	▼ -10.00 (-2.34 %)	426.50	417.20	211'735
Trade	SWISS LIFE HOLDING AG N	351.20	▼ -1.10 (-0.31 %)	352.40	348.60	146'636
Trade	SWISS RE N	92.22	▲ 0.52 (0.57 %)	92.92	90.90	2'298'221
Trade	SWISSCOM N	506.80	▼ -1.00 (-0.20 %)	508.80	504.00	188'734
Trade	UBS GROUP N	18.94	▼ -0.115 (-0.60 %)	19.145	18.93	8'793'161
Trade	ZURICH INSURANCE N	307.80	▲ 0.30 (0.10 %)	308.90	306.60	379'067

Begriff: Unstrukturierte Daten

- Daten, für deren Elemente keine bestimmte Anordnung bzw. Struktur vorgeschrieben ist. Sie haben also keine explizite Struktur (aber ev. eine implizite, z.B. Grammatikregeln für Text).

  Synonyme: unformatierte oder formatfreie Daten.

- Beispiele:
  - Texte (können auch strukturiert sein, z.B. gewisse Formulare)
  - Bilder
  - Filme
  - Audiodaten
  - …

Begriff: Semi-strukturierte Daten

- Daten, die teilweise irreguläre bzw. unvollständige Strukturen aufweisen, die sich öfters ändern können. Das Schema kann evtl. aus zusätzlicher Information (sog. „markup“) (re)konstruiert werden.
- Beispiel:

Datenarten ↔ Technologien

- Unterschiedliche Datenarten verlangen nach unterschiedlichen Technologien:
  - Strukturierte Daten: Relationale Datenbanken
  - Semi-Strukturierte Daten: XML, JSON, …
  - Unstrukturierte Daten: ??? (diverse proprietäre Technologien)

Relationale Datenbanken

- Relationale Datenbanken sind hervorragend geeignet um strukturierte Daten zu verwalten. Technologie und Produkte haben sich über Jahrzehnte entwickelt, sind entsprechend leistungsfähig und ausgereift. Für semi-strukturierte Daten eigenen sich vor allem XML-Technologien.

- Relationale Datenbanken sind – trotz Schwächen bei der Verwaltung von semi-strukturierten und unstrukturierten Daten – heute immer noch die weitaus wichtigste Datenbanktechnologie.

- Die Entwicklung der letzten paar Jahre geht in Richtung "Ausbau" der relationalen Technologie (aber Achtung: meist ohne mathematisch saubere Grundlage):
  - Erweiterungen für XML-Daten, "OO", BI, …
  - Skalierung (horizontal, vertikal) → "big data"

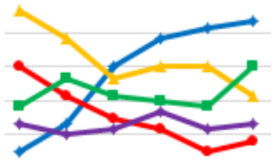
Relationale Datenbanken

- Verbreitung von relationalen Datenbankverwaltungssystemen:

DB-Engines Ranking

Das DB-Engines Ranking ist eine Rangliste der Popularität von Datenbankmanagement-systemen. Die Rangliste wird monatlich aktualisiert.

Lesen Sie mehr über die Berechnungsmethode der Bewertungen.



Trend-Diagramm

345 Systeme im Ranking, September 2018

Rang Sep 2018	Rang Aug 2018	Rang Sep 2017	DBMS	Datenbankmodell	Punkte Sep 2018	Punkte Aug 2018	Punkte Sep 2017
1.	1.	1.	Oracle	Relational DBMS	1309,12	-2,91	-49,97
2.	2.	2.	MySQL	Relational DBMS	1180,48	-26,33	-132,13
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1051,28	-21,37	-161,26
4.	4.	4.	PostgreSQL	Relational DBMS	406,43	-11,07	+34,07
5.	5.	5.	MongoDB	Document Store	358,79	+7,81	+26,06
6.	6.	6.	DB2	Relational DBMS	181,06	-0,78	-17,28
7.	↑ 8.	↑ 10.	Elasticsearch	Suchmaschine	142,61	+4,49	+22,61
8.	↓ 7.	↑ 9.	Redis	Key-Value Store	140,94	+2,37	+20,54
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	133,39	+4,30	+4,58
10.	10.	↓ 8.	Cassandra	Wide Column Store	119,55	-0,02	-6,65

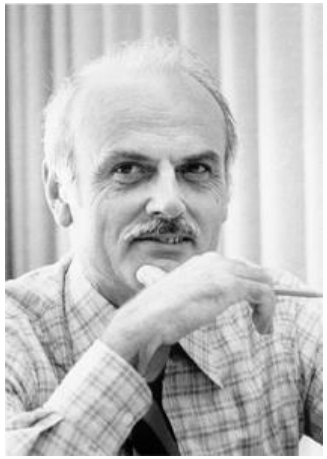
Quelle: http://db-engines.com/de/ranking

Gründe für den Erfolg des rel. Modelles

- Einfache Datenstruktur: Relation
- Mengenorientierte Verarbeitung der Daten, alle Operationen führen wieder zu Relationen. Wenige Grundoperationen zur Verarbeitung und dadurch eine klare Semantik ($\rightarrow$ relationale Algebra).
- Formale Theorie zur Modellierung und Anfrageverarbeitung.
- Implementationen sind relativ einfach zu benutzen:
  - DDL (data definition language): SQL
  - DML (data manipulation language): SQL
  - DQL (data query language): SQL

Wer hats erfunden?

- E. F. Codd: A Relational Model of Data for Large Shared Data Banks. Comm. of the ACM, 13(6): 377-387(1970)



- Codd hat dafür den bedeutendsten Preis in der Informatik gewonnen. Welchen?
- In der Praxis (als formale Grundlage von RDBMS) von sehr grosser Bedeutung!

Begriff: Wertebereich (Domäne)

- Menge einfacher bzw. „atomarer“ Werte (entspricht im wesentlichen einem unstrukturierten Datentyp einer höheren Programmiersprache).

- Beispiele:
  - Ganze Zahlen, Fixpunktzahlen (Dezimalbrüche), Gleitkommazahlen, ...
  - Menge aller Zeichenketten, evt. einer festen Länge
  - Aufzählungstypen, z.B. {red, green, blue}, {CHF, EUR, GBP, USD}
  - Unterbereichstypen, z.B. ganze Zahlen im Intervall [100 ... 999]
  - …

Begriff: Attribut

- «Eigenschaften, die uns interessieren» (siehe später: ERM)

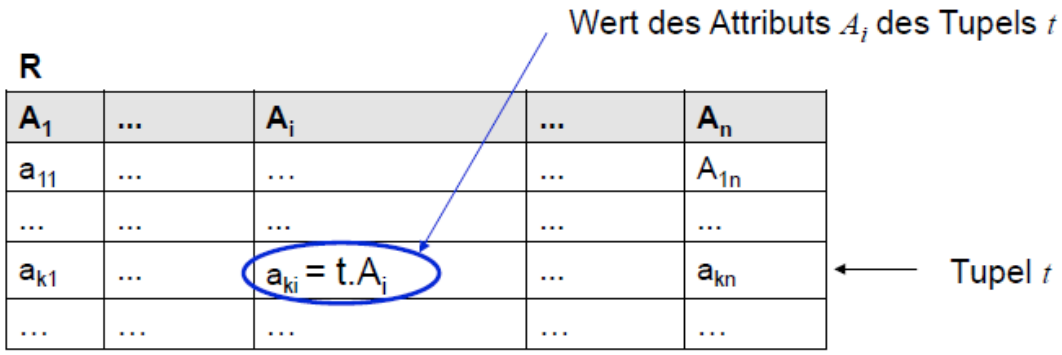
- Ein Attribut besteht aus zwei Teilen:
  - Bezeichnung/Name
  - Domäne/Wertebereich, aus der die zugehörigen Werte stammen können

- Bsp:
  - Anrede, {„Herr“, „Frau“}
  - Ort, string[30]
  - Gehalt, float
  - ...

Begriff: Tupel (n-Tupel)

- Sammlung von als zusammengehörig betrachteter Attribute:
  - Feste Zahl von Komponenten
  - Beliebige Anordnung (d.h. Reihenfolge ist erst wichtig, wenn einmal festgelegt)
  - Der Attributwert entstammt einer für jedes Attribut festgelegten Domäne

- Wichtig: Die Attributwerte verändern sich über die Zeit, die Attribute selbst, also Name, Bedeutung und Wertebereich, bleiben konstant
- Eine Menge von gleichartig strukturierten Tupeln bilden eine Relation:



Begriff: Relation

- Eine Relation R besteht aus zwei Teilen, einem:
  1. Einem sogenannten Relationenformat (Heading, Schema, Relationsvariable): Menge von Namen von Attributen $\{A_1, ..., A_n\}$, wobei jedem Attributnamen A_i ein Domain (Wertebereich) $dom(A_i) \in \{D_1, ..., D_m\}$ zugeordnet wird

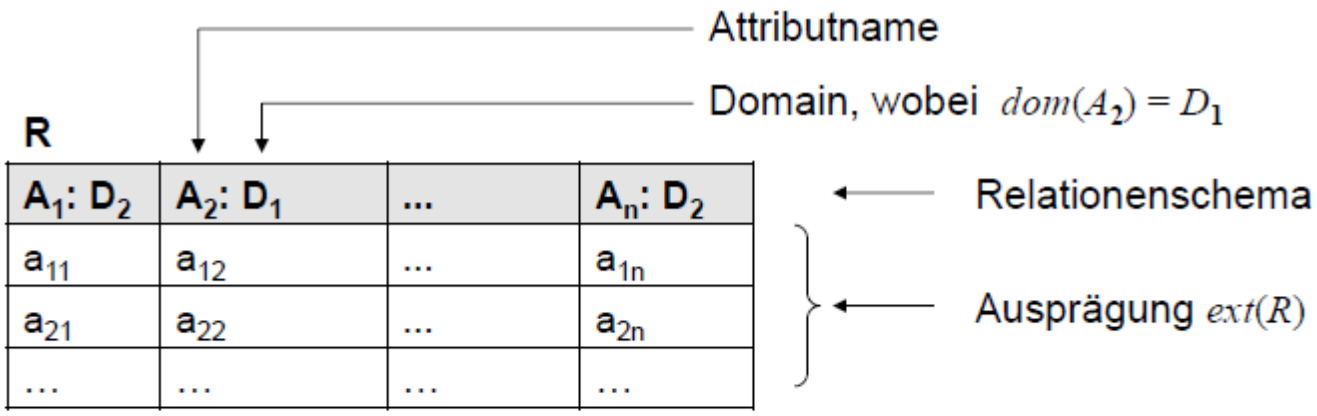
     und

  2. Einer sogenannten Ausprägung (Menge von Tupeln, die dem Relationenschema entsprechen): $ext(R) \subseteq dom(A_1) \times ... \times dom(A_n)$

- Etwas zum Aufschneiden: «Eine Relation ist eine Teilmenge des kartesischen Produktes von n Wertebereichen»

- Wichtig: Relationen sind Mengen, enthalten also (per Definition) keine doppelten Elemente!!

Begriff: Relation (Notationen)



Gleiche Relation ohne Angabe der Domains:

R

A_1	A_2	...	A_n
a_{11}	a_{12}	...	a_{1n}
a_{21}	a_{22}	...	a_{2n}
...	...	...	...

Kurznotation für Relationenschema: $\boldsymbol{R}(A_1, A_2, \ldots, A_n)$

Beispiel: Relation

Employees

EmpNo: EmpNos	Name: PersonNames	AdrCity: CityNames	YrSalAmt: INTEGER	YrSalCur: Currencies	YrBonAmt: INTEGER	YrBonCur: Currencies
1	'Bechtolsheim'	'Palo Alto'	100000	'USD'	50000	'USD'
2	'Khosla'	'Altos Hills'	200000	'USD'	100000	'USD'
3	'McNealy'	'Atherton'	150000	'USD'	200000	'USD'
5678	'Saaner'	'Thalwil'	120000	'CHF'	100000	'CHF'

- Domänen:
  - EmpNos = INTEGER
  - PersonNames = VARCHAR(30)
  - CityNames = VARCHAR(30)
  - Currencies = {‘USD’, ‘GBP’, ‘EUR’, ‘CHF’}

Relationenformat

- Relationen gehen im Laufe der Zeit in andere Relationen über. Aber: zeitlich konstant an den Beispielrelationen ist das gemeinsame Format.

- Wir nennen dieses Format Relationenformat und schreiben (z.B.):

  Studenttyp(StudentName, DateOfBirth, StudentsFee)

- In der Literatur wird oft auch die Bezeichnung Relationenschema oder Relationsvariable verwendet

- Die Komponenten des Relationenformats sind Attribute, im Beispiel: StudentName, DateOfBirth und StudentsFee

Bemerkung zu Domänen

- Zweck von Domänen
  - Theorie: Nur Attribute mit gleichen Domänen sind „kompatibel“, z.B. bei Vergleichen, arithmetischen Operationen etc., Attribute mit verschiedenen Domänen sind nicht kompatibel
  - Praxis: Nicht relevant, die meisten RDBMS unterstützen nur vordefinierte Domänen wie INTEGER, VARCHAR(n) etc. – und das ist auch gut so!

- Aber: Domänen dürfen nur sogenannte «atomare» Werte enthalten, d.h. strukturierte Werte (Mengen, Listen, Vektoren u.ä. nicht zulässig)

UStudents

StudNo	Name	Faculty	ProgrammingSkills
‘87–604–I’	‘Meier’	‘Comp Science’	{ ‘Java’, ‘C’, ‘SQL’ }
‘91–872–I’	‘Schmid’	‘Comp Science’	{ ‘Pascal’, ‘Java’ }
‘91–109–I’	‘Anderegg’	‘Comp Science’	{ ‘Java’ }
‘94–555–L’	‘Imboden’	‘Chemistry’	{ }

Attribute und Wertebereiche

- In der Theorie ist es sinnvoll, die erlaubten Werte einer Domäne möglichst genau zu beschreiben und einzuschränken. Heutige RDBMS unterstützen diese Einschränkungen aber nur teilweise.

- Vorteile? Nachteile?

Attribute und Wertebereiche

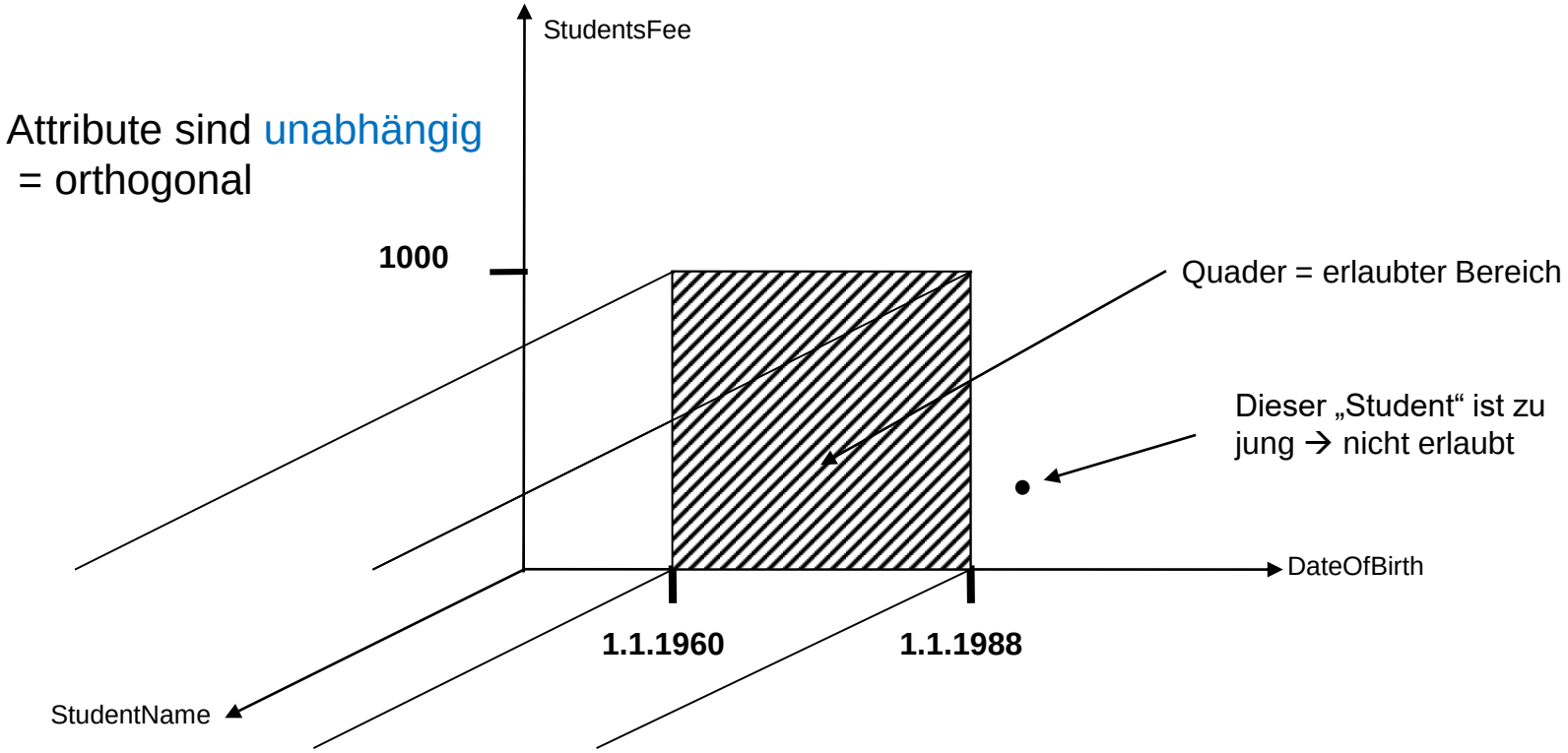
- Vorteile von Datentypen: Damit können schon eine Reihe von möglichen Eingabefehlern zentral (statt in jedem einzelnen Anwendungsprogramm) abgefangen werden.

- Beispiel: Studenttyp(*StudentName*, *DateOfBirth*, *StudentsFee*)

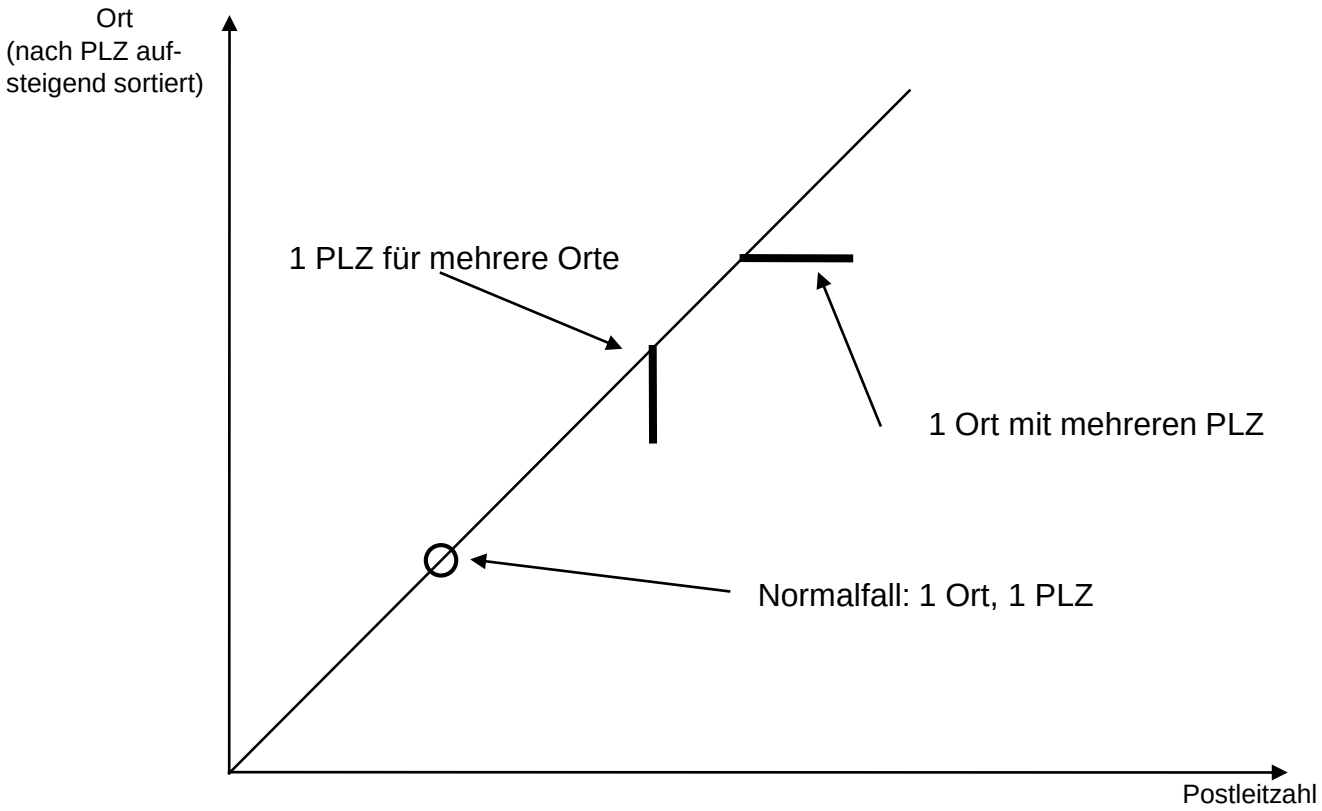
  DateOfBirth >= ‘1.1.1960’ $\wedge$ DateOfBirth <= ‘1.1.1988’
  StudentsFee >= 0 $\wedge$ StudentsFee <= 1000

Unabhängigkeit von Attributen

- Die Attributwerte von Studenttyp(StudentName, DateOfBirth, StudentsFee) können «räumlich» aufgezeichnet werden:



Unabhängigkeit von Attributen



Ein Vorgriff auf kommende Themen: Ort und Postleitzahl sind nicht unabhängig, die Zuordnung ist nicht eindeutig
→ Könnten theoretisch als 2 Entitätstypen mit einer m:m–Beziehung modelliert werden

Übersicht, etwas formaler

Wertebereich (domain):

- Menge einfacher bzw. „atomarer“ Werte (entspricht im wesentlichen einem unstrukturierten Datentyp einer höheren Programmiersprache)
- Gegeben sei eine Menge $\{D_1, ..., D_m\}$ von Wertebereichen
- Relation r besteht aus:
  - Relationenformat: Menge von Namen von Attributen $\{A_1, ... , A_n\}$ wobei jedem Attributnamen A_i eine Domain $dom(A_i) \in \{D_1, ..., D_m\}$ zugeordnet wird
  - Ausprägung (Wert, Extension): $ext(r) \subseteq dom(A_1) \times ... \times dom(A_n)$
- Als Domains werden häufig Zahlen (ganze, Gleitkomma-, Fixpunkt-Zahlen) oder Zeichenketten verwendet. Oft werden sie nicht explizit aufgeführt
- Der Wertebereich gilt als zusätzliche Einschränkung der möglichen Attributswerte

Äquivalenz

- Zwei Relationen sind äquivalent wenn sich die eine durch Umordnen der Attribute der anderen herstellen lässt
- Darstellung: R1 ~ R2
- Bsp:
  - BUCH1(ISBN, Titel, Typ, Jahr, Auflage, Fachgebiet)
  - BUCH2(Titel, Typ, ISBN, Auflage, Jahr, Fachgebiet)
  - BUCH1 ~ BUCH2

Eigenschaften

- Konsequenzen aus Definition, Relationen sind:
  - Duplikatfrei
  - Ungeordnet (Attribute & Tupel)
  - Endlich

- Veranschaulichung in Tabellenform:
  - Spalten = Attribute
  - Zeilen = Tupel

Begriff: Relationale Algebra

- Operatoren und Regeln für das «Rechnen» mit Relationen (Analogie: Arithmetik mit Addition, Subtraktion… auf ganzen Zahlen).
- Bis jetzt: Relationenschemas mit Relationen (Attributwerten), die in einer Datenbank gespeichert sind (wie sie dahin kommen folgt später).
- Es fehlt: «Abgeleitete» Relationenschemas mit Relationen, die aus den «Basisrelationen» berechnet werden.
- Die Rechenvorschriften definieren eine «Abfragesprache», d.h. eine Sprache, mit der beliebige Abfragen an eine Datenbank gestellt werden können.

Begriff: Relationale Algebra

- Mathematik:
  - Algebra: Definiert durch Wertebereich und auf diesem definierte Operatoren.
  - Operand: Variablen oder Werte aus denen neue Werte konstruiert werden können.
  - Operator: Symbole, die Prozeduren repräsentieren, die aus gegebenen Werten neue Werte produzieren.

- Für Datenbankanfragen:
  - Inhalt der Datenbank (Relationen) sind Operanden.
  - Operatoren definieren Funktionen zum Berechnen von Anfrageergebnissen: «Grundlegenden Dinge, die wir mit Relationen tun wollen».
  - Relationale Algebra (Relationenalgebra, RA).
  - Anfragesprache für das relationale Modell.

Kriterien für Abfragesprachen

- Ad-Hoc-Formulierung:
  - Benutzer sollen eine Anfrage formulieren können, ohne ein vollständiges Programm schreiben zu müssen.
- Deskriptivität / Deklarativität:
  - Benutzer sollen formulieren „Was will ich haben?“ und nicht „Wo finde ich das, was ich haben will?“
- Optimierbarkeit:
  - Sprache besteht aus wenigen Operationen.
  - Optimierungsregeln für die Operatorenmenge.

Kriterien für Abfragesprachen

- Abgeschlossenheit:
  - Anfragen erfolgen auf Relationen.
  - Anfrageergebnis ist wiederum eine Relation und kann als Eingabe für die nächste Anfrage verwendet werden → Schachtelung möglich.
- Mengenorientiertheit:
  - Operationen auf Mengen von Daten.
  - Nicht navigierend nur auf einzelnen Elementen („tuple-at-a-time“).
- Sicherheit:
  - Keine Anfrage, die syntaktisch korrekt ist, darf in eine Endlosschleife geraten oder ein unendliches Ergebnis liefern.
- …

Klassifikation der rel. Operatoren

- Entfernende Operatoren:
  - Selektion, Projektion
- Mengenoperatoren:
  - Vereinigung, Schnittmenge, Differenz
- Kombinierende Operatoren:
  - Kartesisches Produkt, Join, Joinvarianten
- Umbenennung:
  - Verändert nicht Tupel, sondern Schema
- Ausdrücke der relationalen Algebra: „Anfragen“ (queries)

Einschub: Griechisches Alphabet

Name	Zeichen	Name	Zeichen	Name	Zeichen
Alpha	A, α	Iota	I, ι	Rho	P, ρ
Beta	B, β	Kappa	K, κ	Sigma	Σ, σ
Gamma	Γ, γ	Lambda	Λ, λ	Tau	T, τ
Delta	Δ, δ	My	M, μ	Ypsilon	Υ, υ
Epsilon	E, ϵ	Ny	N, ν	Phi	Φ, ϕ
Zeta	Z, ζ	Xi	Ξ, ξ	Chi	X, χ
Eta	H, η	Omikron	O, o	Psi	Ψ, ψ
Theta	Θ, ϑ	Pi	Π, π	Omega	Ω, ω

Symbole der relationalen Algebra

Übersicht der Operationen:

- σ Selektion
- π Projektion
- x Kreuzprodukt
- $\bowtie$ Join (Verbund)
- \ Mengendifferenz
- $\cup$ Mengenvereinigung
- $\cap$ Mengendurchschnitt
- $\div$ Division
- ρ Umbenennung
- Linker äusserer Verbund (Symbol, oft weggelassen: $⟕$)
- Rechter äusserer Verbund (Symbol, oft weggelassen: $⟖$)
- Voller äusserer Verbund (Symbol, oft weggelassen: $⟗$)

Entfernend: Selektion, σ

- Unärer Operator (d.h. nur ein Operand)
- Erzeugt neue Relation mit gleichem Schema aber einer Teilmenge der Tupel.
- Nur Tupel, die der sogenannten Selektionsbedingung entsprechen, werden übernommen.
- Selektionsbedingung: Kombination von logischen Ausdrücken bestehend aus Attributen und/oder Konstanten (das Ergebnis des Ausdrucks muss wahr oder falsch sein).
- Prüft Selektionsbedingung für jedes Tupel der Relation.

Entfernend: Selektion, σ

Filme					
Titel	**Jahr**	**Länge**	**inFarbe**	**Studio**	**ProduzentID**
Total Recall	1990	113	True	Fox	12345
Basic Instinct	1992	127	True	Disney	67890
Dead Man	1995	90	False	Paramount	99999

$\sigma_{\text{Länge}\geq 100}(\text{Filme})$

Titel	**Jahr**	**Länge**	**inFarbe**	**Studio**	**ProduzentID**
Total Recall	1990	113	True	Fox	12345
Basic Instinct	1992	127	True	Disney	67890

Entfernend: Selektion, σ

Customers

CNo	Name	City	Balance	Discount
1	'Legrand'	'Genève'	0.00	0.10
2	'Studer'	'Zürich'	–800.00	0.20
3	'Huber'	'Zürich'	–20.00	0.05

Finde alle Zürcher Kunden, die einen Rabatt von 15% oder mehr erhalten

$\sigma_{\text{City = 'Zürich'} \wedge \text{Discount} \geq 0.15}$ Customers

CNo	Name	City	Balance	Discount
2	'Studer'	'Zürich'	–800.00	0.20

Entfernend: Projektion, π

- Unärer Operator (d.h. nur ein Operand).
- Erzeugt neue Relation mit einer Teilmenge der ursprünglichen Attribute
  - $\pi_{A1,A2,\ldots,Ak}(R)$ ist eine Relation mit den Attributen A1,A2,…,Ak
- Achtung: Es können Duplikate entstehen, die entfernt werden müssen.
- Die Projektion wählt eine Menge von Spalten aus (und entfernt die Übrigen).
- Mit einer Projektion lässt sich auch die Reihenfolge der Spalten anpassen.

Entfernend: Projektion, π

Filme					
Titel	**Jahr**	**Länge**	**inFarbe**	**Studio**	**ProduzentID**
Total Recall	1990	113	True	Fox	12345
Basic Instinct	1992	127	True	Disney	67890
Dead Man	1995	121	False	Paramount	99999

$\pi_{Titel,Jahr,Länge}(Filme)$

Titel	**Jahr**	**Länge**
Total Recall	1990	113
Basic Instinct	1992	127
Dead Man	1995	121

$\pi_{inFarbe}(Filme)$

inFarbe
True
False

Aufgabe zur Projektion

- Aufgabe: Es sei **Clubs** = {<Tischtennis, 1.10.1990, 15.00>,
  <Schwimmclub, 31.7.1988, 10.00>,
  <PC, 1.10.1990, 15.00>}.

- Relationenformat Clubtyp(Clubname, Gründungsdatum, Jahresbeitrag)

? Wie sieht die Relation $\pi_{Gründungsdatum,\ Jahresbeitrag}$ (Clubs) aus?

Resultat

- Man beachte:

- $\pi_{Gründungsdatum,\ Jahresbeitrag}$ (Clubs) = {<1.10.1990, 15.00>,
  <31.7.1988, 10.00>}

- Die Ergebnisrelation besteht aus zwei Tupeln, nicht etwa drei.

? Wieso?

Bemerkung

- Relationen sind Mengen. Mengen können keine doppelten Elemente enthalten!
- Die Projektion eliminiert Duplikate.
- Aber Achtung: Relationale Datenbankverwaltungsysteme eliminieren mehrfach vorkommende Tupel bei der Projektion nicht automatisch. Man spricht dann (präziser) nicht von Relationen sondern von (relationalen) Bags (= Multimengen)
- Es gilt auch: $\sigma_{\Phi 1}(\sigma_{\Phi 2}(r)) = \sigma_{\Phi 2}(\sigma_{\Phi 1}(r))$
- Hingegen gilt nicht: $\pi_A(\pi_B(r)) = \pi_B(\pi_A(r))$

Und weiter..

Das nächste Mal: Relationale Algebra (Fortsetzung)